

Reasoning with Separation Logics

Complexity, Expressive Power, Proof Systems

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580, Sciences et technologies de l'information
et de la communication (STIC)

Spécialité de doctorat: Informatique

Unité de recherche: CNRS, LSV, ENS Paris-Saclay, Université Paris-Saclay

Référent: ENS Paris-Saclay

**Thèse présentée et soutenue en visioconférence totale,
le 10 Décembre 2020, par**

Alessio MANSUTTI

Composition du jury:

Erich Grädel Professor, RWTH Aachen, Germany	Président
Radu Iosif Chargé de recherche CNRS, Université Grenoble Alpes, France	Rapporteur
David Pym Professor, University College London, UK	Rapporteur
Leonid Libkin Professor, École Normale Supérieure, France	Examinateur
Mihaela Sighireanu Professor, ENS Paris-Saclay, France	Examinateuse
Stéphane Demri Directeur de recherche CNRS, ENS Paris-Saclay, France	Directeur
Étienne Lozes Professor, Université Côte d'Azur	Codirecteur

Abstract

This thesis proposes an in-depth study of classical decision problems, such as satisfiability and validity, for separation logics: well-known assertion languages developed to verify sequential and concurrent programs with dynamic memory allocation. The main features of separation logics are given by two binary connectives: the separating conjunction $*$ and the separating implication $\rightarrow*$. These two connectives use second-order features in order to express spatial properties, and allow the verifier to modularly analyse the memory.

The first part of the thesis focuses on the notion of reachability predicates in separation logics. A location ℓ_1 (a.k.a. a memory address) is said to be reachable from a second location ℓ_2 whenever subsequent dereferentiation of ℓ_2 yield ℓ_1 . Together with first-order quantification, reachability predicates allow to check for several *robustness properties* of the memory, such as acyclicity (i.e. the absence of cycles) and garbage freedom (i.e. the absence of dangling pointers). Our goal is to devise a separation logic featuring these types of predicates, while keeping the complexity of its satisfiability problem in check. This is easier said than done, as we show that several separation logics are largely intractable (undecidable or non-elementary decidable). Nonetheless, by studying the sources of intractability we are able to design a PSPACE fragment of first-order separation logic that is able to express the desired robustness properties. The PSPACE membership is shown through the *core formulae technique*, a model theoretic approach that is reminiscent of Gaifman's locality theorem for first-order logic.

The second part of the thesis tackles the open problem of designing Hilbert-style axiomatisations for separation logics. An axiomatisation is said to be Hilbert-style (or internal) whenever its axioms and inference rules are built solely from formulae of the logic, without any external machinery such as labels or nominals. In particular, we introduce the first sound and complete internal proof system for the quantifier-free fragment of separation logic. The completeness of the system is shown by taking advantage of the core formulae technique introduced in the first part of the thesis. In order to show that this technique can be reused on other logics, we design an axiomatisation for a modal logic enriched with the composition operator from ambient logic (a logic to verify distributed systems). The two proof systems reveal interesting connections between separation logics and ambient logics.

Motivated by the similarities in their proof systems, in the third part of the thesis we dig deep in the connections between separation logic and ambient logic. To carry out our comparison, we devise a suitable framework based on modal logic. This framework gives us the common ground needed in order to study the two logics from the point of view of their spatial connectives: the separating conjunction $*$ for separation logic and the parallel composition $|$ for ambient logic. Surprising similarities and differences are discovered, both in terms of expressive power and computational complexity.

Contents

Contents	iii
1 Introduction	1
1.1 Scalable Reasoning about Programs	3
1.2 Pointer Program Verification and Separation Logic	5
1.3 Between Theory and Practice	7
1.4 Our Work	8
1.5 Prerequisites and Basic Notations	12
2 Separation Logic	15
2.1 A Logic for Shared Mutable Data Structures	17
2.2 Fragments of $SL(\exists, *, \neg*)$ and Second-Order Logic	28
2.3 Other Separation Logics and Bunched Logics	33
I Reachability Queries in Separation Logic	41
Introduction: Robustness Properties of Logical Assertions	43
3 Extensionality and Reachability Leads to Non-enumerability	47
3.1 Encoding Assignments as Memory Cells	51
3.2 Simulating the First-order Quantification	55
3.3 Reachability Predicates can Quantify	65
4 Intensionality and Reachability Leads to Non-elementary Logics	73
4.1 The Hardness of Reachability and Submodel Reasoning	77
4.2 On the Expressive Power of ALT	81
4.3 The Complexity of ALT	97
4.4 Revisiting TOWER-hard Logics with ALT	103
5 Deciding Robustness Properties in PSpace	115
5.1 Taming the Robustness Properties	119
5.2 Towards Small Models: The Core Formulae Technique	122
5.3 A Family of Core Formulae Capturing the Fragment w	127
5.4 Recap: How to Apply the Core Formulae Technique	144
5.5 A Family of Core Formulae Capturing the Fragment s	146
5.6 Connecting the Two Families of Core Formulae	221

Conclusion	269
II Internal Calculi for Spatial Logics	273
Introduction: Internal Proof Systems via Core Formulae	275
6 A Complete Axiomatisation for Quantifier-free Separation Logic	279
6.1 Axiomatising $SL(*, \neg*)$, Internally	283
6.2 An Hilbert-style proof system for $SL(*, \neg*)$	285
6.3 Main ingredients of the method	290
6.4 A Simple Calculus for the Core Formulae	291
6.5 Syntactical elimination of the Separating Conjunction	295
6.6 Syntactical elimination of the Separating Implication	316
7 Axiomatising a Modal Logic Featuring Ambient-like Composition	337
7.1 A Taste of Ambient Logic	341
7.2 The Modal Logic $ML()$	342
7.3 Towards an Hilbert-style proof system for $ML()$	346
7.4 Graded Modalities as Core Formulae	348
7.5 Syntactical Elimination of the Composition Operator	352
Conclusion	367
III Mixing Multiplicative Connectives and Modalities	369
Introduction: Two Ways to Chop a Tree	371
8 The Complexity of the Modal Logic $ML()$	373
8.1 $ML()$ and GML as fragments of second-order ML	377
8.2 Checking satisfiability for $ML()$, in $AEXP_{POL}$	380
8.3 $ML()$ is $AEXP_{POL}$ -complete	390
8.4 An $AEXP_{POL}$ -complete Static Ambient Logic	397
9 The Complexity and Expressive Power of the Modal Logic $ML(*)$	403
9.1 $ML(*)$: when $*$ replaces $ $	407
9.2 $ML(*)$ is Strictly Less Expressive than $ML()$	411
9.3 The complexity of $ML(*)$	432
9.4 Revisiting TOWER-hard Logics with $ML(*)$	459
Conclusion	465
References	469
A Appendix of Chapter 3	481
B Appendix of Chapter 4	491
C Appendix of Chapter 5	511

D Appendix of Chapter 6	551
E Appendix of Chapter 7	559
F Appendix of Chapter 8	567
G Appendix of Chapter 9	575
List of Notations and Symbols	585
List of Definitions	589
List of Figures	591
Index	595

1

Introduction

Contents

1.1	Scalable Reasoning about Programs	3
1.2	Pointer Program Verification and Separation Logic	5
1.3	Between Theory and Practice	7
1.4	Our Work	8
1.4.1	Part I: Reachability queries in separation logic.	9
1.4.2	Part II: Internal calculi for spatial logics.	10
1.4.3	Part III: Mixing multiplicative connectives and modalities.	11
1.5	Prerequisites and Basic Notations	12

1.1 SCALABLE REASONING ABOUT PROGRAMS

In the craft of programming, resource management plays a central role. The finiteness of resources such as time and space led computer scientists to an ever going quest for faster and more efficient algorithms that eventually evolved to what is now known as Complexity Theory. In fact, as noticed by M. Y. Vardi in the May 2020 Communications of the ACM [140]:

“generations of computer scientists [were taught] that analysis of algorithm only means analyzing their computational efficiency. As Wikipedia states: “In computer science, the analysis of algorithms is the process of finding the computational complexity of algorithms—the amount of time, storage, or other resources needed to execute them.” In other words, efficiency is the sole concern in the design of algorithms.”

Efficiency is however only one facet of software and, due to the ubiquitousness of technology in our society, the craft of programming is naturally evolving to consider other qualities of programs, such as security and resilience. Broadly speaking, both security and resilience evaluate the robustness of systems against disruptive changes in the environment. Both these qualities are more abstract than efficiency. The latter is evaluated quantitatively: time-wise, we are interested in the numbers of operations needed in order to perform an algorithm, whereas space-wise we are interested in numbers of memory cells that are simultaneously being used. Not only efficiency is relatively simple to assess, it is also compositional: if on an input of size n two programs f and g respectively require n and m operations, then running g and f sequentially on the same input requires $n + m$ operations. On the other hand, there is no “magic number” to evaluate the security or resilience of software. These are not absolute properties of all software, and they heavily depend on the environment and its interactions with the computer program. The problem of formalising these properties is thus far from trivial, and becomes even harder if we want the specifications to be easily composed as it is done in the case of running time. Summarising the central message of [140], the trade-off between efficiency and security/resilience has now become a central problem in computer science.

The formal specification and analysis of qualitative properties of software (correctness, resilience, security etc.) is known as Program Verification. Even though the first attempts of verifying routines trace back to A. Turing and J. von Neumann (see e.g. [134]), the mathematical foundation of this field were set at the end of the sixties by C. A. R. Hoare and R. W. Floyd [87, 71]. In their works, the central idea is to formalise the properties of a system by means of logical assertions, and track their evolution as the instructions of the program fire. This idea is formalised with a proof system known as Hoare logic, where judgements are given by *Hoare triples* of the form $\{\varphi\} P \{\psi\}$, which should be read as:

“Every model that satisfies the assertion φ will, after being modified by the program P , satisfy the assertion ψ (if P terminates).”

Here, φ and ψ are called the precondition and postcondition of the Hoare triple, respectively, and a *model* is a mathematical structure that abstracts the resources that the program uses. The semantics above follows the notion of *partial correctness*, which in contrast with total correctness does not require P to terminate: if P does not terminate, then the postcondition is assumed to be satisfied. As an example, a possible model could be the domain of functions going from program variables to real numbers, and a program could be as simple as an assignment instruction of the form $x \leftarrow \text{expr}$, which assigns to the variable x the result of an arithmetic expression expr . A possible assertion language could be given by a logic featuring the classical Boolean connectives

(conjunction \wedge , disjunction \vee and negation \neg) together with atomic propositions of the form $expr_1 = expr_2$ stating that the result of two arithmetical expressions $expr_1$ and $expr_2$ coincide. With respect to these objects, the following Hoare triple is valid:

$$\{x = 1 \wedge y = 1\} \ x \leftarrow x - y \ \{x = 0 \wedge y = 1\}.$$

Indeed, the precondition $x = 1 \wedge y = 1$ restricts the set of possible models to the class of functions f such that $f(x) = 1$ and $f(y) = 1$. The instruction $x \leftarrow x - y$ modifies $f(x)$ so that it is equivalent to the value of $f(x) - f(y)$ (before the update, i.e. 0). The resulting function satisfies the postcondition $x = 0 \wedge y = 1$. To formally prove the validity of Hoare triples without semantical arguments (as we just did), Hoare logic provides a set of axioms and inference rules to syntactically manipulate and derive new Hoare triples. For instance, with respect to the assignment $x \leftarrow expr$, Hoare logic features the axiom schema

$$(\text{ASSIGN}) \quad \{\varphi[x \leftarrow expr]\} \ x \leftarrow expr \ \{\varphi\},$$

where $\varphi[x \leftarrow expr]$ is the assertion obtained from φ by syntactically replacing all occurrences of x with $expr$ (without evaluating $expr$). By instantiating the axiom **(ASSIGN)** so that φ corresponds to the assertion $x = 0 \wedge y = 1$, we deduce that $\{x - y = 0 \wedge y = 1\} \ x \leftarrow x - y \ \{x = 0 \wedge y = 1\}$ is a valid Hoare triple. Notice that this triple is equivalent to the previous one we proved semantically, even though its precondition is syntactically different. To solve this discrepancy, we can rely on the *(left) weakening rule*

$$(\text{WEAK}) \quad \frac{\varphi \models \varphi' \quad \{\varphi'\} P \{\psi\}}{\{\varphi\} P \{\psi\}}.$$

Here, the *entailment* $\varphi \models \varphi'$ between logical assertions φ and φ' states that every model satisfying φ also satisfies φ' . Given a valid triple $\{\varphi'\} P \{\psi\}$ and a precondition φ that is stronger than φ' , i.e. $\varphi \models \varphi'$, the rule **(WEAK)** allows us to deduce the validity of the triple $\{\varphi\} P \{\psi\}$. Since the entailment $x = 1 \wedge y = 1 \models x - y = 0 \wedge y = 1$ holds, by instantiating **(WEAK)** as follows

$$\frac{x = 1 \wedge y = 1 \models x - y = 0 \wedge y = 1 \quad \{x - y = 0 \wedge y = 1\} x \leftarrow x - y \ \{x = 0 \wedge y = 1\}}{\{x = 1 \wedge y = 1\} x \leftarrow x - y \ \{x = 0 \wedge y = 1\}}$$

we derive the triple $\{x = 1 \wedge y = 1\} x \leftarrow x - y \ \{x = 0 \wedge y = 1\}$ purely syntactically.

Hoare logic works very well when used to verify programs manipulating primitive data types such as numeric domains, but struggles when dealing with more structured data and, in particular, when dealing with programs that manipulates pointers [115]. There are several reasons for this, but the crucial one has to do with the modularity of the proof system. As previously mentioned, the efficiency of an algorithm can be studied in terms of its components, which allows us to scale the computational analysis to large programs. When it comes to Hoare logic, this type of modularity is deeply tangled with the concept of independence between variables. For instance, let us take the two-instructions program $x \leftarrow x - y; z \leftarrow x$. The effects of the first instruction does not depend on the value assigned to the variable z , whereas the second instruction is independent from y . To achieve modularity, we would like to reason separately on these instructions, and only consider pre- and post-conditions containing variables that are meaningful for the evaluation of the instruction under analysis. Intuitively, this property would allow us to scale the verification to larger programs, as atomic instructions only consider a tiny subsets of all variables and data structures used by the program. More formally, modularity can be achieved in Hoare logic as soon as we add the following *rule of constancy*:

$$\text{(CONST)} \quad \frac{\{\varphi\} P \{\psi\} \quad \text{all variables in } \chi \text{ are not modified by } P}{\{\varphi \wedge \chi\} P \{\psi \wedge \chi\}}.$$

Fundamentally, this rule allows us to drop from the precondition all superfluous information (represented by the assertion χ) about variables that are not considered by P , carry out the proof of $\{\varphi\} P \{\psi\}$ that features the simpler assertion φ and ψ , and then push back the superfluous information directly in the postcondition. Intuitively, we expect this very natural inference rule to hold: if P does not modify the variables occurring in χ , then this assertion should still be satisfied after the execution of P . Unfortunately, (CONST) fails when considering programs manipulating pointers (or more generally having instructions with side effects).

The main problem is due to pointer aliasing. Borrowing the syntax of the C programming language, let us write $*x$ for the result of dereferencing the pointer variable x , so that the instruction $*x \leftarrow *y$ change the value stored in the address referenced by x to the value stored in the address referenced by y . Intuitively, the Hoare triple $\{*x = 1\} *x \leftarrow 0 \{*x = 0\}$ is valid. Since the variable y does not occur in the instruction $*x \leftarrow 0$, we can apply the rule (CONST) instantiated so that χ corresponds to the formula $*y = 0$, and derive the triple

$$\{*x = 0 \wedge *y = 0\} *x \leftarrow 1 \{*x = 1 \wedge *y = 0\}.$$

However, this triple is not semantically valid, as the pointer variables x and y could reference the same address, i.e. they could be in aliasing, which implies that the instruction $*x \leftarrow 1$ has the side effect of setting $*y$ to 1, invalidating the assertion $*y = 0$ in the postcondition.

1.2 POINTER PROGRAM VERIFICATION AND SEPARATION LOGIC

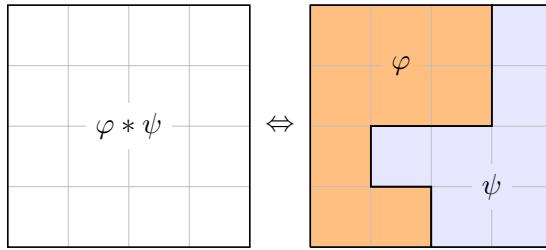
Because of the inconsistency between (CONST) and pointer aliasing, Hoare logic does not scale well when dealing with the verification of pointer programs. As most software use pointers on a regular basis, up to the 2000s this prevented the use of Hoare logic to check industrial level code. As pointed out by P. W. O’Hearn, J. C. Reynolds and H. Yang in [114, 115], the main problem is

“[that] there is a mismatch between simple intuitions about the way that pointer operations work and the complexity of their axiomatic treatments [in Hoare logic]”.

Indeed, the atomic instructions manipulating pointers modify the memory only locally, leaving most of it unaffected. As remarked for the rule (CONST), this feature shapes our intuition, which tells us that most of the properties expressed in a precondition should still be true after an atomic instruction is executed. The struggles in using Hoare logic for pointer programs lie in counter-intuitive fact that, again quoting [114],

“an alteration of a [single memory] cell may affect the values of many syntactically unrelated expressions”.

From the seventies until the end of the nineties, several works tried to partially solve this problem. First attempts were focused on describing the memory directly in the assertion language [96, 106, 14]. What was clear at the time was that the assertion language should be able to describe the notion of spatiality of the memory, where two pointer variables that are not in aliasing correspond, spatially, to syntactically distinct objects in the assertions. In this sense, the first notable result was put forward by R. M. Burstall in [31]. In his Distinct Non-Repeating List system (DNRL), Burstall introduced assertions that implicitly represent the notion of non-

Figure 1.1: The separating conjunction $*$.

aliasing in a compact way. For instance, the assertion language of DNRL features tuples of the form $*(x \hookrightarrow y, y \hookrightarrow z, v \hookrightarrow z)$ that are semantically equivalent to the assertion

$$*x = y \wedge *y = z \wedge *v = z \wedge x \neq y \wedge x \neq v \wedge y \neq v.$$

Essentially, not only the DNRL tuple expresses properties such as “ x points to y ”, which means that the value stored by the addresses referenced by x is the address referenced by y (i.e. $*x = y$), it also represents the notion of non-aliasing: if $x \hookrightarrow y$ and $v \hookrightarrow z$ belong to a DNRL tuple (in distinct positions), then x and v refer to different addresses. Then, a DNRL n -tuple divides the memory in n spaces by implicitly encoding $\mathcal{O}(n^2)$ inequalities between pointer variables.

Based on the ideas of Burstall, in the early 2000s P. W. O’Hearn and J. C. Reynolds, together with C. Calcagno, D. Distefano, D. Pym, H. Yang and the contribution of several other researchers, developed *separation logic* [124]: an extension of Hoare logic that permits scalable reasoning of pointer programs. This extension affects both the assertion language used for pre/postconditions and the set of Hoare-style axioms and inference rules.

Separation logic generalises the notion of spatiality given by DNRL tuples to a solid mathematical theory where the memory is seen as a resource that can be partitioned. To achieve this, the logic applies in a fundamental way the developments on resource reasoning given by the logic of Bunched Implication (BI) introduced by Pym and O’Hearn [113]. The essential feature of BI and of the *assertion language of separation logic* (which, despite de ambiguity, we refer to simply as separation logic) is given by a binary connective $*$, called *separating conjunction*, which roughly stands for “and, separately”. A formula $\varphi * \psi$ of BI is satisfied by a resource r (in the case of separation logic, the memory) if r can be partitioned into two pieces, one satisfying φ and the other satisfying ψ . Figure 1.1 intuitively depicts the spatial partitioning performed by the separating conjunction. Intuitively, one can then encode the DNRL tuple $*(x \hookrightarrow y, y \hookrightarrow z, v \hookrightarrow z)$ in separation logic¹, with a simple change of notation:

$$x \hookrightarrow y * y \hookrightarrow z * v \hookrightarrow z.$$

Indeed, thanks to the separating conjunction $*$, which partitions the memory in distinct pieces, the three variables x , y and v are implicitly assumed to refer to three distinct addresses.

Most importantly, the connective $*$ solves the mismatch between the local effect of atomic instructions on the memory and their global effect on the pre- and post-conditions, as it allows us to elegantly rephrase the rule (CONST) as follows:

$$\text{(FRAME)} \quad \frac{\{\varphi\} P \{\psi\} \quad \text{all variables in } \chi \text{ are not modified by } P}{\{\varphi * \chi\} P \{\psi * \chi\}}.$$

¹In this thesis, we use the term separation logic for both the Hoare proof system and its assertion language.

This rule, known as the *frame rule*, recovers the modularity of the rule of constancy lost when dealing with pointer programs, and paves the way for analysing large programs. It reflects our intuition that if P does not modify the variables in χ , then the satisfaction of χ should not change after executing P . In the context of industrial applications, one cannot stress enough the benefits on scalability given by (FRAME). Not only this rule allows us to verify large programs in the first place, it also enables a quick analysis of successive updates to an already verified software, by only considering the portion of the code that has effectively changed. Last but not least, the frame rule allows us to split the proof in numerous independent subproofs, which can be checked in parallel.

1.3 BETWEEN THEORY AND PRACTICE

The scalable reasoning enabled by separation logic had a vast impact in the full spectrum of program verification, from academic and theoretical research to industrial applications. After 20 years from its first theoretical developments, separation logic is now deployed in numerous industrial tools of program analysis. A vast literature has been written on the subject, and we invite the reader to look at the insightful paper “*Why separation logic works*”, by D. Pym, J. M. Spring and P. W. O’Hearn [116], as well as the article by O’Hearn in the February 2019 Communications of the ACM [115], to better understand the effects that separation logic had on the field. It is worth noting that scalable reasoning is only one of the reasons why separation logic is successful. Below, we summarise two other qualities of this logic.

Catastrophic errors. Despite doable in theory, the industrial tools using separation logic offers limited assistance when it comes to verify the correctness of the results given by an algorithm. However, these tools shine in the analysis of exhaustible resources, such as memory. They allow us to verify whether software generates memory leaks or null pointer exceptions, and they can do so with little to no guidance provided by the programmer. As stated at the beginning of the introduction, resource management plays a central role in computer science, and error due to exhausting or misusing available resources are perhaps the most severe ones. As explained in [116], “*a program with a memory management error will behave erratically or fail suddenly*”, which can not only lead to a fatal failure of the whole computer system, but also results in serious security flaws.

Automation and bi-abduction. Hoare logic often requires the programmer to guide the verification by annotating the code with pre- and post-conditions, in particular for loop invariants. This prevents the practical use of Hoare logic on old unannotated software, and it is a bottleneck in industrial deployment, as the programmer must provide both the implementation and part of its verification. In order to scale the verification to millions of lines of code, tools based on separation logic can be built to be completely automatic (see e.g. INFER [38]). To reach automation, separation logic relies on an inference problem, called *bi-abduction*, introduced in [37] by D. Distefano, C. Calcagno, P. W. O’Hearn and H. Yang. While the formal definition of bi-abduction will follow in Section 2.3.1, we can informally describe solutions of this problem as ways of “stitching” the postcondition ψ_1 of a Hoare triple $\{\varphi_1\} P_1 \{\psi_1\}$ with the precondition φ_2 of a second Hoare triple $\{\varphi_2\} P_2 \{\psi_2\}$ so that the entailment $\psi_1 \models \varphi_2$ holds, which allows us to then deduce the validity of $\{\varphi_1\} P_1; P_2 \{\psi_2\}$ by simply applying the rule of

sequential composition of Hoare logic. Thanks to bi-abduction, the problem of deriving suitable pre- and post-conditions can be fully automated.

Research on separation logic essentially branches out into two directions, the one studying and developing the Hoare-style calculus, and the one focused on its assertion language. The major achievement of the first direction, which is at the intersection of programming languages, concurrency theory and static/symbolic analysis, was perhaps showing the feasibility of modular reasoning to the program verification community. The philosophy behind separation logic was quickly picked up by several other fields outside of pointer programs analysis, as shown for instance by the recent works [7, 6] on probabilistic programming languages. As a testament of the impact that this direction had in the field of program analysis, the 2016 Göedel prize was awarded to S. Brookes and P. W. O’Hearn for their work on concurrent separation logic [24, 23], an extension of separation logic for program verification of concurrent programs with shared-memory. The Hoare calculi based on separation logics form the foundation of several tools, from the first solvers SMALLFOOT and SPACEINVADER [13, 144] and their successors INFER and SLAYER [38, 12], to the concurrent separation logic framework IRIS [94] and the new language-independent framework GILLIAN [126].

The research on the assertion language, which is rooted in mathematical logic, has also been undoubtedly active and fruitful. As shown by the weakening rule (WEAK), Hoare logic (and separation logic) require to solve classical decision problems, e.g. the entailment between two assertions, in order to build the proof. The first iteration of separation logic [124] featured an assertion language with undecidable satisfiability, validity and entailment problems, which of course limited its automation. Aiming for efficiency, J. Berdine, C. Calcagno and P. W. O’Hearn studied a lightweight fragment of separation logic, known as the symbolic heap fragment (SH), that can be decided in PTIME [10]. The tractability of SH made it the go to assertion language for several separation logic tools, as for instance SMALLFOOT and INFER. Subsequent works made a huge efforts to analyse assertion languages that sits between SH and the first separation logic from [124]. For practical purposes, several authors studied the addition to SH of (user-defined) inductive predicates in order to reason on data structures such as lists or trees [44, 28, 92, 65, 93, 101]. In this direction, very recently J. Pagel, C. Matheja and F. Zuleger defined a 2EXPTIME algorithm for the entailment problem of SH with inductive definitions of bounded treewidth [118], matching the 2EXPTIME-hardness by M. Echenim, R. Iosif and N. Peltier [64]. On the more theoretical side, it is known from [53] that the first-order separation logic restricted to two quantified variables is already undecidable, in contrast with the NEXPTIME-completeness of the two-variable fragment of first-order logic [84]. The restriction of first-order separation logic to one quantified variable is however PSPACE [55].

The broadness of works on this topic is also due to the fact that separation logic appears to lie on a sweet spot between theory and practice. Besides its practical applications that led to its industrial success, separation logic is based on the rigorous and elegant theory of the logic of bunched implications, which attracted attention in different fields of mathematical logic such as topology, proof theory and modal logic. On this matters, we invite the reader to look at the PhD thesis of S. Docherty [60].

1.4 OUR WORK

This thesis focus on the assertion language of separation logic. Broadly speaking, our goal is to methodically analyse features of separation logic, primarily the separating conjunction $*$ and the

separating implication \rightarrow (i.e. the right-adjoint of $*$), to improve our understanding of the logic from a computational and proof theoretical point of view. In doing so, we draw connections between separation logic and other logics, most notably ambient logics [34] and modal logics [15].

The thesis is naturally split into three parts (not counting the technical background on separation logic given in Chapter 2), each having its own introduction and conclusion. The first part consider the role of reachability predicates in separation logic, and study their computational complexity. To start, we substantially refine the analysis on the separating implication \rightarrow started with [22] and [53] in order to understand the reasons behind of the high computational status of separation logic. We show a way of handling the operator \rightarrow , together with reachability predicates, to design an expressive separation logic whose satisfiability, validity and entailment problems are “only” PSPACE-complete. In the second part of the thesis, we apply proof techniques introduced in the first part to tackle the open problem of designing Hilbert-style calculi for separation logics and other spatial logics. Interestingly, this part reveals connections between separation logic and ambient logics. In the third part of the thesis, we propose an in-depth analysis of these connections from the point of view of complexity and expressive power. Each chapter should be sufficiently self-contained, with perhaps the exception of Chapters 7 and 8.

Below, we give a more detailed view on the contributions of the thesis.

1.4.1 Part I: Reachability queries in separation logic.

Reachability predicates, such as the list segment predicate $1s(x, y)$, are perhaps the most studied types of inductive predicate in separation logic. Roughly speaking, $1s(x, y)$ holds in a memory that can be represented as a linear structure that starts with the address corresponding to the variable x and ends, through dereferentiation, in the address corresponding to the variable y . It is known that the entailment problem of the symbolic heap fragment enriched with $1s(x, y)$ can be solved in PTIME [44]. However, when considering richer separation logics that are closed under Boolean connectives and feature both the separating conjunction $*$ and implication \rightarrow , the addition of reachability predicates has not been studied. This is quite surprising, as these types of logics are able to express several properties of the memory that are fundamental in program analysis, such as acyclicity and garbage freedom. In Chapters 3, 4 and 5, we undertake a journey through these types of logics, with the aim of finding one that is decidable in PSPACE.

Here is a roadmap of Part I:

Chapter 3. We show that the quantifier-free separation logic $SL(*, \rightarrow, 1s)$ featuring Boolean connectives, both operators $*$ and \rightarrow , and the reachability predicate $1s(x, y)$, admits non recursively enumerable satisfiability and validity problems. This result is extended to several other separation logics, most notably $SL(*, \rightarrow, \hookrightarrow^2, \hookrightarrow^3)$, i.e. the fragment of $SL(*, \rightarrow, 1s)$ only featuring the bounded reachability predicates $x \hookrightarrow^\delta y$, where $\delta \in \{2, 3\}$, stating that dereferencing δ times the address corresponding to the variable x yields the address corresponding to y . This chapter covers the first part of the work published in:

- [56] S. Demri, E. Lozes, and A. Mansutti, “The effects of adding reachability predicates in propositional separation logic,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 10803. Springer, 2018, pp. 476–493.

Chapter 4. Distressed by the results in Chapter 3, we introduce an *Auxiliary Logic on Trees* (ALT), a modal logic that allows us to focus on the interactions between reachability predicates

and submodel reasoning, without the specificity of separation logic. After looking at the expressive power of ALT by defining a suitable notion of Ehrenfeucht-Fraïssé games, we show that the satisfiability problem of ALT is TOWER-complete, and thus non-elementary decidable [128]. This result extends to several logics interpreted on tree-like structures that were independently found to be TOWER-complete: quantified computation tree logic, modal separation logic and modal logic of heaps. Moreover, it shows that the separation logic $\text{SL}([\exists]_1, *, \mathbf{x} \hookrightarrow _, \hookrightarrow^+)$ already admits a non elementary satisfiability problem. This logic features Boolean connectives, the separating conjunction $*$, one quantified variable, the predicate alloc $\mathbf{x} \hookrightarrow _$ and the reachability predicate $\mathbf{x} \hookrightarrow^+ \mathbf{y}$. Chapter 4 covers the work published in:

- [108] A. Mansutti, “An auxiliary logic on trees: on the tower-hardness of logics featuring reachability and submodel reasoning,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 12077. Springer, 2020, pp. 462–481.

Chapter 5. The negative results of Chapter 3 and Chapter 4 guide us to the definition of the separation logic $\text{SL}([\exists]_1, *, [\neg*], \hookrightarrow^+_{\mathcal{W}})$ that is able to express properties such as acyclicity and garbage freedom, and admits a PSPACE-complete satisfiability problem. To show the PSPACE upper bound of the satisfiability problem for $\text{SL}([\exists]_1, *, [\neg*], \hookrightarrow^+_{\mathcal{W}})$, we extend the proof method of the *core formulae* introduced by E. Lozes in [104]. More precisely, we show that every formula of $\text{SL}([\exists]_1, *, [\neg*], \hookrightarrow^+_{\mathcal{W}})$ can be translated into a Boolean combination of “core” formulae, for which we show a polynomial small model property. Understanding this technique, which is connected to the first-order locality theorem proved by H. Gaifman in [73], gives us a new line of attack on the problem of designing Hilbert-style proof systems for separation logic. This chapter covers the second part of the work published in:

- [107] A. Mansutti, “Extending propositional separation logic for robustness properties,” in *Foundations of Software Technology and Theoretical Computer Science*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 42:1–42:23.

1.4.2 Part II: Internal calculi for spatial logics.

Despite the amount of research done on the computational complexity of separation logics, we know comparatively very little in terms of its proof systems. In the second part of the thesis, we try to shrink this gap by considering the problem of designing sound, complete and internal (a.k.a. Hilbert-style) proof systems for separation logics and similar logics. We recall that a proof system is complete whenever it can derive every semantically valid formula, and it is internal if its axioms and rules only use formulae of the logic, without relying on any external (and richer) theory. Fortunately, the insights given by the core formulae technique discussed in Chapter 5 facilitate our tasks, allowing us to design natural axiomatisations for these logics.

Here is a roadmap of Part II:

Chapter 6. We present the first Hilbert-style proof system for the quantifier-free separation logic $\text{SL}(*, \neg*)$, featuring Boolean connectives and both the separating conjunction and implication. Thanks to the core formulae technique, our axiomatisation is modular: we start from a Boolean algebra for the core formulae, and then extend it (twice) to support the connectives $*$ and $\neg*$. This allows us to derive a form of constructive completeness, as advocated in [61]. This chapter covers the first part of the work published in:

- [58] S. Demri, E. Lozes, and A. Mansutti, “Internal calculi for separation logics,” in *Computer Science Logic*, ser. LIPIcs, vol. 152. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020, pp. 19:1–19:18.

Chapter 7. In order to show that our methodology is reusable in practice, we apply it to axiomatise an ambient logic called $\text{ML}(\|)$. Ambient logics [39, 34] are modal logics introduced to verify properties of distributed systems specified in the calculus of Mobile Ambients [40]. Their main feature is given by the composition operator $\varphi\|\psi$ that, similarly to the separating conjunction, asks to spatially split the distributed process into two pieces, one satisfying the formula φ and the other satisfying the formula ψ . The axiomatisation of $\text{ML}(\|)$ follows the same principles as the one of $\text{SL}(*, \neg)$, and reveals interesting similarities between the two logics. Moreover, it shows that $\text{ML}(\|)$ is as expressive as graded modal logic, a well-known extension of modal logic K [79, 70]. The results in this chapter are not yet published.

1.4.3 Part III: Mixing multiplicative connectives and modalities.

Puzzled by the relationships between separation logics and ambient logics emerged from the proof systems of $\text{SL}(*, \neg)$ and $\text{ML}(\|)$, we undertake a comprehensive analysis on the differences, in terms of computational complexity and expressive power, between the separating conjunction $*$ and the composition operation $\|$. We rely on the framework of modal logic in order to carry out the comparison, and introduce the logic $\text{ML}(*)$ which is essentially obtained from $\text{ML}(\|)$ by replacing the composition operator with the separating conjunction. Despite the semantical similarities between $\text{ML}(*)$ and $\text{ML}(\|)$, we identify surprising differences in terms of their expressiveness and complexity. Part III roughly covers the work published in:

- [9] B. Bednarczyk, S. Demri, R. Fervari, and A. Mansutti, “Modal logics with composition on finite forests: Expressivity and complexity,” in *Logic in Computer Science*. ACM, 2020, pp. 167–180.

Here is a roadmap of Part III:

Chapter 8. We complete the analysis on $\text{ML}(\|)$ started in Chapter 7. We show that the satisfiability problem for $\text{ML}(\|)$ is AEXP_{POL}-complete, i.e. complete for the class of decision problems solvable by an alternating Turing machine with exponential runtime and polynomial number of alternations. Whereas the lower bound is quite simple to establish, the upper bound is derived from a refined translation to graded modal logic, using fundamental properties of the proof system designed in Chapter 7. The AEXP_{POL}-completeness of $\text{ML}(\|)$ transfers to other ambient logics from [34].

Chapter 9. We analyse $\text{ML}(*)$ in terms of expressive power and complexity. First of all, we prove that $\text{ML}(*)$ is strictly less expressive than $\text{ML}(\|)$. This is shown by relying on semantical connections between the two logics, together with an ad-hoc notion of Ehrenfeucht-Fraïssé games for $\text{ML}(*)$. Surprisingly, even though $\text{ML}(*)$ is less expressive than $\text{ML}(\|)$, we discover that its satisfiability problem is TOWER-complete. The chapter ends by formalising the connections between $\text{ML}(*)$ and separation logic, which allows us to solve some open problems in the realm of modal separation logics.

1.5 PREREQUISITES AND BASIC NOTATIONS

This thesis takes for granted basic notions of “naïve set theory” and mathematical logic (e.g. the Boolean algebra, standard constructions on sets and relations, first and second-order logics), as well as an understanding of the main concepts of complexity theory (e.g. complexity classes, many-one reduction, Turing reduction). Some familiarity in finite model theory and proof theory is also helpful, but not required. Below, we introduce the notations used throughout the thesis for the basic concepts from set theory. Their definitions are given only when necessary to avoid confusion. The complete list of symbols is given at the end of the thesis.

Sets. We use the standard notation for the algebra of sets:

- \emptyset : empty set;
- \times : Cartesian product;
- \subsetneq : strict inclusion;
- \cup : union;
- \subseteq : inclusion;
- $\text{card}(\cdot)$: cardinality;
- \cap : intersection;
- $=$: equality;
- $2^{(\cdot)}$: powerset;
- \setminus : difference;
- \neq : inequality;
- \in : membership.

The symbol \mathbb{N} denotes the set of natural numbers. Given $i, j \in \mathbb{N}$, $[i, j] \stackrel{\text{def}}{=} \{k \in \mathbb{N} \mid i \leq k \leq j\}$. We write $i \dot{-} j$ for the subtraction on natural numbers, i.e. $i \dot{-} j \stackrel{\text{def}}{=} \max(0, i - j)$.

Binary Relations. Aside from the operations above, for binary relations we also use:

- $(\cdot)^{-1}$: converse;
- $\pi_2(\cdot)$: 2nd projection;
- $(\cdot)^+$: Kleene plus;
- $\pi_1(\cdot)$: 1st projection;
- $(\cdot)^\delta$: δ th composition;
- $(\cdot)^*$: Kleene closure.

Let us recall the definition of δ th composition, Kleene plus and Kleene closure of a relation.

Definition 1.1 (δ th composition). Let $R \subseteq S \times S$ be a binary relation. The 0th composition R^0 is defined as the identity map id_S on S . Given $\delta \in \mathbb{N}$, the $(\delta+1)$ th composition is defined as:

$$R^{\delta+1} \stackrel{\text{def}}{=} \{(s, t) \in S \times S \mid \text{there is } e \in S \text{ such that } (s, e) \in R^\delta \text{ and } (e, t) \in R\}.$$

Definition 1.2 (Kleene plus and Kleene closure). Let $R \subseteq S \times S$ be a binary relation. Its Kleene plus is $R^+ \stackrel{\text{def}}{=} \bigcup_{\delta \geq 1} R^\delta$. Its Kleene closure (also called Kleene star) is $R^* \stackrel{\text{def}}{=} R^0 \cup R^+$.

Notice that, if R is a functional relation, then so is R^δ .

Functions. We use standard notation for functions:

- $S \rightharpoonup T$: partial functions from S to T ,
- $S \rightarrow T$: functions from S to T ,
- $S \rightharpoonup_{\text{fin}} T$: partial functions from S to T defined on finitely many values of S .

By seeing a partial function $f : S \rightharpoonup T$ as a weakly functional binary relation $f \subseteq S \times T$, f inherits all the operations defined for sets and binary relations. We write $\text{dom}(f)$ and $\text{ran}(f)$ for its domain and range (or image), respectively. Their definitions are recalled below.

Definition 1.3 (Domain, image of a subset, range). Let $f : S \rightharpoonup T$ be a partial function.

- (I) $\text{dom}(f) \stackrel{\text{def}}{=} \{s \in S \mid \text{there is } t \in T \text{ such that } f(s) = t\}$,
- (II) given $S' \subseteq S$, $f(S') \stackrel{\text{def}}{=} \{t \in T \mid \text{there is } s' \in S' \text{ such that } f(s') = t\}$,

(III) $\text{ran}(\mathfrak{f}) \stackrel{\text{def}}{=} \mathfrak{f}(S)$.

Notice that $\text{dom} = \pi_1$ and $\text{ran} = \pi_2$. Both notations are kept to make the exposition clearer.

Formulae. As usual, formulae are syntactical object from a vocabulary made of constant symbols c_1, c_2, \dots (e.g. atomic formulae), and operators (predicate or function symbols) P_1, P_2, \dots , each with an associated arity.

We use the standard notions of positions, subformulae, and substitutions, recalled below.

Definition 1.4 (Positions, subformulae and substitutions). Given a formula φ , the set $\text{Pos}(\varphi)$ of its *positions* is the subset of \mathbb{N}^* inductively defined as follows (ϵ stands for the empty sequence):

$$\text{Pos}(\varphi) \stackrel{\text{def}}{=} \begin{cases} \{\epsilon\} & \text{if } \varphi \text{ is a constant symbol,} \\ \{\epsilon\} \cup \{i\rho \mid i \in [1, n], \rho \in \text{Pos}(\varphi_i)\} & \text{if } \varphi = P(\varphi_1, \dots, \varphi_n), \text{ for an } n\text{-ary operator } P. \end{cases}$$

The *subformula* of φ at position $\rho \in \text{Pos}(\varphi)$, written $\varphi|_\rho$, is defined as follows

$$\begin{aligned} \varphi|_\epsilon &\stackrel{\text{def}}{=} \varphi, \\ (P(\varphi_1, \dots, \varphi_n))|_{i\rho} &\stackrel{\text{def}}{=} \varphi_i|_\rho. \end{aligned}$$

Lastly, we write $\varphi[\psi]_\rho$ for the formula obtained from φ by *replacing* its subformula at position ρ with a formula ψ .

Separation Logic

Contents

2.1	A Logic for Shared Mutable Data Structures	17
2.1.1	Memory allocation and reachability predicates.	21
2.1.2	The (not so) classical decision problems.	24
2.2	Fragments of $SL(\exists, *, \neg*)$ and Second-Order Logic	28
2.2.1	Fragments of $SL(\exists, *, \neg*)$	28
2.2.2	$SL(\exists, *, \neg*)$ as a Fragment of Second-Order Logic.	30
2.3	Other Separation Logics and Bunched Logics	33
2.3.1	Symbolic-Heaps and (bi)abduction.	33
2.3.2	Modal Separation Logics.	34
2.3.3	Boolean BI.	37

In this preliminary chapter

We present the first-order fragment of separation logic, that serves us as a way to introduce standard notions from the separation logic literature. After familiarising with these notions, we examine landmark results on the decidability of classical decision problems (e.g. satisfiability) for first-order separation logic. The last part of the chapter is dedicated to a round-up of the separation logic literature. First of all, we connect separation logic with weak second-order logics, which allows us to get a better grasp on the decidability status of first-order separation logic. Afterwards, we introduce well-known fragments of first-order separation logic, such as the symbolic-heap fragment and modal separation logic. The chapter ends by placing separation logic in the more general framework of the logic of bunched implications, which deepens our understanding of the various components of the logic.

2.1 A LOGIC FOR SHARED MUTABLE DATA STRUCTURES

As one can expect, twenty years of research in separation logic led to numerous variants and extensions of the original logic, some of which are discussed in Section 2.3. For our purposes, a good starting point is given by the separation logic $\text{SL}(\exists, *, \neg*)$ defined in [22], which closely follow the initial presentation given by J. Reynolds in its seminal work [124].

Syntax. We use $\text{VAR} = \{x, y, z, \dots\}$ to denote the countably infinite set of *program variable names* (or *variables*, in short). The formulae φ of the *first-order separation logic* $\text{SL}(\exists, *, \neg*)$ and its atomic formulae π are built from the grammars below (where $x, y, z \in \text{VAR}$):

$\pi :=$	\top	(true)	$\varphi :=$	π	(atomic formulae)
	emp	(empty predicate)		$\varphi \wedge \varphi$ $\neg\varphi$	(Boolean connectives)
	$x = y$	(equality predicate)		$\varphi * \varphi$	(separating conjunction)
	$x \rightarrow y$	(points-to predicate)		$\varphi \neg* \varphi$	(separating implication)
				$\exists z \varphi$	(first-order quantification)

The separating conjunction $*$ and the separating implication $\neg*$ are also called the *star* and the *magic wand*, respectively. Following the terminology of the logic of bunched implications [113] (see Section 2.3) as well as linear logic [78], we often refer to these two operators as two *multiplicative connectives*.

Given a formula φ , we write $\text{bv}(\varphi)$ and $\text{fv}(\varphi)$ for the sets of *bound variables* and *free variables* occurring in φ , respectively. A variable x is *bound* (resp. *free*) if it occurs (resp. does not occur) in the scope $\exists x$ of a first-order quantification. A formula is said to be *closed* whenever $\text{fv}(\varphi) = \emptyset$. Unless otherwise specified, the *size* $|\varphi|$ of a formula φ is understood as its tree size, i.e. the number of symbols needed to encode it as a tree.

Memory States. Separation logic is interpreted on memory states, which can be seen as abstractions of the heap/RAM memory model used as a backbone for the semantics of many programming languages (e.g. Java and C++). Addresses and data in the memory are all abstracted with a countably infinite set of *locations*, denoted by LOC . Furthermore, a memory state contains information on the location assigned to each program variable, as well as dependencies between locations to represent pointers and their stored addresses. Memory states are formally defined as follows.

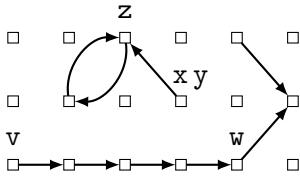


Figure 2.1: A memory state.

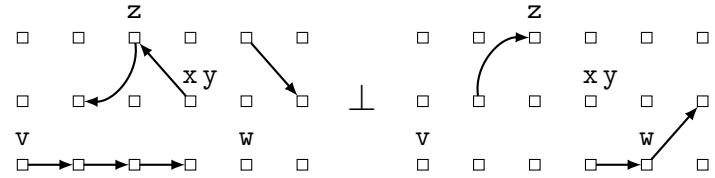


Figure 2.2: Two disjoint memory states.

Definition 2.1 (Memory state). A *memory state* is a pair (s, h) where $s : \text{VAR} \rightarrow \text{LOC}$ is called the *store*, and $h : \text{LOC} \rightarrow_{\text{fin}} \text{LOC}$ is a partial function with finite domain, called the *heap*.

Given a heap h , each element in $\text{dom}(h)$ is understood as a *memory cell* of h . Informally, we can see the heap as a finite functional graph, having locations as vertices. The set of locations is however infinite, and the store assigns some of them to program variables. As expected, two distinct locations cannot be assigned to the same variable. Figure 2.1 shows a memory state. Locations are denoted by small boxes (\square), and arrows represent the heap. Sometimes, we write the pair of locations (ℓ_1, ℓ_2) using the notation $\ell_1 \mapsto \ell_2$ and calling it an *arrow*, as it stresses that the pair belongs to a heap. For the same reason, given $\ell_1, \dots, \ell_{n-1}$ distinct locations, and a location ℓ_n (possibly equal to one of the first $n-1$ locations), we write $\{\ell_1 \mapsto \ell_2 \mapsto \ell_3 \mapsto \dots \mapsto \ell_n\}$ for the heap $\{(\ell_1, \ell_2), (\ell_2, \ell_3), \dots, (\ell_{n-1}, \ell_n)\}$. This heap witnesses a directed path going from the location ℓ_1 to the location ℓ_n , where cycles are permitted.

Definition 2.2 (Path). Given a heap h , a (*finite*) *path* is a sequence of locations (ℓ_1, \dots, ℓ_n) such that $h(\ell_j) = \ell_{j+1}$ holds for all $j \in [1, n - 1]$. Such a path *goes from* ℓ_1 *to* ℓ_n .

As already discussed in Section 1.1 separation logic allows for modular analysis of memory states by means of its two multiplicative connectives $*$ and \dashv . On the model side, in order to achieve its modularity the class of memory states is endowed with a union operator on heaps that allows to simulate how memory states can be composed or decomposed.

Definition 2.3 (Disjoint heaps and their union). (*Disjointness*) Two heaps h_1 and h_2 are said to be *disjoint*, written $h_1 \perp h_2$, whenever their domains are disjoint, i.e. $\text{dom}(h_1) \cap \text{dom}(h_2) = \emptyset$. (*Union*) When $h_1 \perp h_2$ holds, the *heap-union* $h_1 + h_2$ of h_1 and h_2 is defined as the set union $h_1 \cup h_2$. If $h_1 \perp h_2$ does not hold, then $h_1 + h_2$ is not defined.

More explicitly, given two disjoint memory states h_1 and h_2 , for every location $\ell \in \text{LOC}$, we have $(h_1 + h_2)(\ell) = \text{if } \ell \in \text{dom}(h_1) \text{ then } h_1(\ell) \text{ else } h_2(\ell)$. Two paths are said to be *disjoint* if their underlying heaps are disjoint. We lift the notion of disjoint heaps to memory states, and say that two memory states (s, h_1) and (s, h_2) are *disjoint* whenever $h_1 \perp h_2$ holds. Notice that this notion of disjointness requires the two memory states to share the same store. Figure 2.2 shows two disjoint memory states. As we can see, disjointness intuitively means that every two arrows taken from different heaps cannot share the same source. One can check that performing the heap-union on the heaps in Figure 2.2 yields the memory state in Figure 2.1. The notion of heap-union naturally leads to the one of subheap.

Definition 2.4 (Subheap). The heap h' is a *subheap* of the heap h whenever $h' \subseteq h$ holds, i.e. when $\text{dom}(h') \subseteq \text{dom}(h)$ and $h'(\ell) = h(\ell)$ holds for every memory cell $\ell \in \text{dom}(h')$. Similarly, the heap h' is a *strict subheap* of h whenever $h' \subsetneq h$ holds.

$(s, h) \models \top$	always,
$(s, h) \models \text{emp}$	iff $h = \emptyset$ (i.e. the heap is empty),
$(s, h) \models x = y$	iff $s(x) = s(y)$,
$(s, h) \models x \leftarrow y$	iff $h(s(x)) = s(y)$,
$(s, h) \models \varphi \wedge \psi$	iff $(s, h) \models \varphi$ and $(s, h) \models \psi$,
$(s, h) \models \neg \varphi$	iff $(s, h) \not\models \varphi$,
$(s, h) \models \varphi * \psi$	iff there are h_1 and h_2 s.t. $h_1 + h_2 = h$, $(s, h_1) \models \varphi$ and $(s, h_2) \models \psi$,
$(s, h) \models \varphi \multimap \psi$	iff for every heap h' , if $h' \perp h$ and $(s, h') \models \varphi$ then $(s, h + h') \models \psi$,
$(s, h) \models \exists z \varphi$	iff there is $\ell \in \text{LOC}$ such that $(s[z \leftarrow \ell], h) \models \varphi$.

Figure 2.3: Satisfaction relation for $\text{SL}(\exists, *, \multimap)$, with respect to a memory state (s, h) .

Alternatively, the notion of subheap can be characterised as follows:

$$h_1 \subseteq h \text{ iff there is } h_2 \text{ such that } h_1 \perp h_2 \text{ and } h_1 + h_2 = h.$$

In this characterisation, h_1 is found to be a subheap of h if another heap can be added to h_1 by means of heap-union, yielding h . From the definition of heap-union, it should be evident that if such a heap exists, then it must be $h \setminus h_1$ (where \setminus is the set difference and we see h and h' as binary relations). The two heaps represented in Figure 2.2 are both subheaps of the one represented in Figure 2.1.

We borrow the notion of (non-empty) weakly connected component of a graph, and redefine it for heaps. This notion is quite useful when reasoning on the structure of a memory state.

Definition 2.5 (Weakly connected component). A *weakly connected component* of a heap h is a non-empty subheap h_1 of h such that

1. the reflexive, symmetric and transitive closure of h_1 forms a clique, i.e.

$$(h_1 \cup h_1^{-1})^* = (\text{dom}(h_1) \cup \text{ran}(h_1)) \times (\text{dom}(h_1) \cup \text{ran}(h_1)),$$

2. h_1 is maximal, i.e. the property (1) does not hold for heaps h_2 satisfying $h_1 \subsetneq h_2 \subseteq h$.

The heap in Figure 2.1 has two weakly connected components:



Semantics. Let us consider a memory state (s, h) . We introduce the satisfaction relation \models for the formulae of $\text{SL}(\exists, *, \multimap)$, and say that (s, h) *satisfies* φ whenever $(s, h) \models \varphi$ holds. As usual, if (s, h) satisfies φ then (s, h) is called a *model* of φ . This notion is extended to sets of formulae: (s, h) is a model for the set S whenever it is a model for every formula in S . If S has a model, then it is said to be *consistent*. The definition of \models is formalised in Figure 2.3. Skipping the tautology \top and the Boolean connectives, with classical semantics, a first interesting

ingredient of $\text{SL}(\exists, *, \neg)$ is already given by the formula emp . This formula holds on (s, h) whenever the heap h is empty, and provides a useful tool for program verification. Roughly speaking, with emp we can check for memory leaks by verifying that a function F called on the empty heap also returns on the empty heap. In a Floyd–Hoare proof system this corresponds to a proof of validity for the triple $\{\text{emp}\} F \{\text{emp}\}$. The formula $x = y$ simply states that the same location is assigned to both the variables x and y . The formula $x \hookrightarrow y$ goes one step further, and states that the location assigned to the variable x points to the location assigned to y . For instance, the memory state in Figure 2.1 satisfies $x \hookrightarrow z$ and $y \hookrightarrow z$. Moreover, as the two variables x and y correspond to the same location, this memory state also satisfies the formula $x = y$. For the separating conjunction, $(s, h) \models \varphi * \psi$ states that it is possible to split the heap h into two disjoint heaps h_1 and h_2 so that the memory state (s, h_1) satisfies φ , whereas the memory state (s, h_2) satisfies ψ . Then, we can easily see that the following relation holds:

$$\text{if } (s, h) \models x \hookrightarrow y * z \hookrightarrow v \text{ then } (s, h) \models x \neq z,$$

where $x \neq z$ is a shortcut for $\neg(x = z)$. Indeed, suppose that h can be split into two disjoint heaps h_1 and h_2 such that $(s, h_1) \models x \hookrightarrow y$ and $(s, h_2) \models z \hookrightarrow v$. Then, both $s(x) \in \text{dom}(h_1)$ and $s(y) \in \text{dom}(h_2)$ hold, leading to $s(x) \neq s(y)$ by disjointness of the two heaps. This implication perfectly shows one of the useful properties of separation logic: the ability to encode inequalities in linear space. Indeed, the formula $x_1 \hookrightarrow y_1 * x_2 \hookrightarrow y_2 * \dots * x_n \hookrightarrow y_n$ implicitly states that for every two distinct $i, j \in [1, n]$, x_i and x_j do not correspond to the same location. We can translate this formula into one that only uses classical connectives, but unfortunately this requires a quadratic amount of inequalities:

$$x_1 \hookrightarrow y_1 \wedge x_2 \hookrightarrow y_2 \wedge \dots \wedge x_n \hookrightarrow y_n \wedge \bigwedge_{i < j \in [1, n]} x_i \neq x_j.$$

where given a set $S = \{e_1, \dots, e_n\}$, we write $\psi(e_1) \wedge \dots \wedge \psi(e_n)$ using the standard notation $\bigwedge_{e \in S} \psi(e)$. Thanks to this property of the separating conjunction, when it comes to Floyd–Hoare proof systems, assertions written using separation logic can be manipulated in an easy and modular way that seems out of reach for classical logics.

Let us now consider the separating implication. $(s, h) \models \varphi \multimap \psi$ states that whenever we consider a heap h' disjoint from h and fulfilling $(s, h) \models \varphi$, the heap $h + h'$ fulfills $(s, h + h') \models \psi$. As we show in the next section and chapters, the ability to extend the current heap with the magic wand leads to very expressive logics. We introduce another standard connective of separation logic, called *sepraction* (see e.g. [137]). It is denoted by \multimap and defined as the right dual of the separating implication, i.e. $\varphi \multimap \psi \stackrel{\text{def}}{=} \neg(\varphi \multimap \neg\psi)$. Its semantics is as follows:

$$(s, h) \models \varphi \multimap \psi \text{ iff there is a heap } h' \text{ such that } h' \perp h, (s, h') \models \varphi \text{ and } (s, h + h') \models \psi.$$

Lastly, let us focus on the first-order quantification. In the definition given in Figure 2.3, the notation $s[z \leftarrow \ell]$ stands for the store obtained from s by only changing the evaluation of z to ℓ . Formally, for every $x \in \text{VAR}$, $s[z \leftarrow \ell](x) \stackrel{\text{def}}{=} \text{if } x = z \text{ then } \ell \text{ else } s(x)$. Then, $(s, h) \models \exists z \varphi$ whenever it is possible to assign a location to z so that the resulting memory state satisfies φ .

The contradiction, classical connectives, and the universal quantifier are derived as usual:

$$\begin{array}{lll}
\perp & \stackrel{\text{def}}{=} & \neg \top & (\textit{false}) \\
\varphi \Rightarrow \psi & \stackrel{\text{def}}{=} & \neg(\varphi \wedge \neg \psi) & (\textit{implication}) \\
\varphi \vee \psi & \stackrel{\text{def}}{=} & \neg \varphi \Rightarrow \psi & (\textit{disjunction}) \\
\forall z \varphi & \stackrel{\text{def}}{=} & \neg \exists z \neg \varphi & (\textit{universal quantifier}) \\
\varphi \Leftrightarrow \psi & \stackrel{\text{def}}{=} & (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi) & (\textit{double implication})
\end{array}$$

As shown by these formulae, we adopt the standard precedence between classical connectives, and extend it for the other operators as follows: $\{\neg, \exists\} > \{\wedge, \vee, *\} > \{\Rightarrow, \Leftrightarrow, \neg*, \otimes\}$. For example, the separating conjunction $*$ has a higher precedence than the separating implication $\neg*$, and it has the same precedence as the (classical) conjunction \wedge . So, the formula $\varphi * \psi \neg* \chi$ should be read as $(\varphi * \psi) \neg* \chi$.

2.1.1 Memory allocation and reachability predicates.

We now focus our attention at some well-known formulae of the separation logic literature (see e.g. [124, 22, 56]) that are expressible in $\text{SL}(\exists, *, \neg*)$. Beside providing a way to become more familiar with the various ingredients of the logic, some of these formulae are extensively used in the following chapters of the thesis. In what follows, let (s, h) be a memory state.

Alloc. We start with the formula $x \hookrightarrow _$, generally referred to as *alloc*. This formula is intended to hold when the location assigned to the variable x is a memory cell. Formally,

$$(s, h) \models x \hookrightarrow _ \quad \text{if and only if} \quad s(x) \in \text{dom}(h).$$

By relying on the first-order quantification, $x \hookrightarrow _$ can be easily defined as $\exists y x \hookrightarrow y$, where y is an arbitrary variable that is syntactically different from x . However, we use another (common) definition, which allows us to show some flavour of the magic wand: $x \hookrightarrow _ \stackrel{\text{def}}{=} x \hookrightarrow x \neg* \perp$.

Proposition 2.6. $(s, h) \models x \hookrightarrow x \neg* \perp$ if and only if $s(x) \in \text{dom}(h)$.

Proof. (\Rightarrow): Suppose $(s, h) \models x \hookrightarrow x \neg* \perp$ and, *ad absurdum*, assume that $s(x)$ is not a memory cell of h . Then, the heap $h' = \{(s(x), s(x))\}$ is disjoint from h and fulfils $(s, h) \models x \hookrightarrow x$. From $(s, h) \models x \hookrightarrow x \neg* \perp$ we then reach the contradictory statement $(s, h + h') \models \perp$.

(\Leftarrow): If $s(x)$ is a memory cell of h then every heap h' fulfilling $(s, h') \models x \hookrightarrow x$ cannot be disjoint from h , leading directly to $(s, h) \models x \hookrightarrow x \neg* \perp$. \square

Size. Another formula that is often considered in the literature is the *size* formula $\text{size} \geq \beta$, where $\beta \in \mathbb{N}$. Again, let us first introduce its intended semantics, to then see its definition:

$$(s, h) \models \text{size} \geq \beta \quad \text{if and only if} \quad \text{card}(h) \geq \beta.$$

This formula can be defined as $\neg \text{emp} * \dots * \neg \text{emp}$ where $\neg \text{emp}$ appears β times, hence essentially stating that the heap can be split into β disjoint non-empty subheaps. Formally,

$$\text{size} \geq 0 \stackrel{\text{def}}{=} \top, \quad \text{size} \geq 1 \stackrel{\text{def}}{=} \neg \text{emp}, \quad \text{for every } \beta \geq 1, \text{size} \geq \beta + 1 \stackrel{\text{def}}{=} \text{size} \geq \beta * \neg \text{emp}.$$

The correctness of this definition should be transparent. We write $\text{size} = \beta$ as a shortcut for the formula $\text{size} \geq \beta \wedge \neg \text{size} \geq \beta + 1$, i.e. the formula satisfied by (s, h) if and only if $\text{card}(h) = \beta$.

Alloc-back and Reach-plus. We introduce the *alloc-back* formula $_ \hookrightarrow x \stackrel{\text{def}}{=} \exists y y \hookrightarrow x$ (where y is syntactically different from x), which states that there is a memory cell pointing to the location assigned to x , i.e. $s(x) \in \text{ran}(h)$. In view of the similarities with the formula $x \hookrightarrow _$, it is quite normal to ask ourselves if also $_ \hookrightarrow x$ can be rewritten without using the first-order quantification. Even though the answer is no (a proof of this is given in Chapter 6), we can achieve an equivalent quantifier-free formula by enriching $\text{SL}(\exists, *, \neg)$ with the *reachability predicate* \hookrightarrow^+ (which we call *reach-plus*) that corresponds to the transitive closure of \hookrightarrow . The semantics of this predicate is defined as follows:

$$(s, h) \models x \hookrightarrow^+ y \text{ if and only if } (s(x), s(y)) \in h^+,$$

where we recall that the relation h^+ corresponds to the transitive closure of h , as defined in Section 1.5. Informally, $(\ell, \ell') \in h^+$ means that h witnesses a non-empty directed path going from the location ℓ to the location ℓ' . For example, by considering (s, h) as the memory state in Figure 2.1, $(s(v), s(w))$ and $(s(z), s(z))$ are in h^+ , but $(s(x), s(y)) \notin h^+$ (as the path must be non-empty) and $(s(w), s(v)) \notin h^+$ (as there is no directed path going from $s(w)$ to $s(v)$). By using this operator, we can capture the semantics of $_ \hookrightarrow x$ with the formula

$$x \hookrightarrow x \vee (\top * (\neg x \hookrightarrow _ \wedge ((\text{size} = 1 \wedge \neg x \hookrightarrow^+ x) \oslash x \hookrightarrow^+ x))).$$

Proving this equivalence, as partially done below, is a good exercise to familiarise with the multiplicative connectives of separation logic.

Proposition 2.7. $s(x) \in \text{ran}(h)$ holds if and only if (s, h) satisfies the following formula

$$x \hookrightarrow x \vee (\top * (\neg x \hookrightarrow _ \wedge ((\text{size} = 1 \wedge \neg x \hookrightarrow^+ x) \oslash x \hookrightarrow^+ x))).$$

Proof. (\Leftarrow): If $(s, h) \models x \hookrightarrow x$ holds then $s(x) \in \text{ran}(h)$ follows trivially. Hence, let us consider the case where $(s, h) \models \top * (\neg x \hookrightarrow _ \wedge ((\text{size} = 1 \wedge \neg x \hookrightarrow^+ x) \oslash x \hookrightarrow^+ x))$. By definition of the operator $*$, there is a subheap $h_1 \subseteq h$ such that $(s, h_1) \models \neg x \hookrightarrow _ \wedge ((\text{size} = 1 \wedge \neg x \hookrightarrow^+ x) \oslash x \hookrightarrow^+ x)$. Then, following the definitions of conjunction and subtraction, we obtain

- (A) $(s, h_1) \models \neg x \hookrightarrow _$, therefore $s(x) \notin \text{dom}(h_1)$ holds by Proposition 2.6;
- (B) $(s, h_2) \models \text{size} = 1 \wedge \neg x \hookrightarrow^+ x$, therefore $\text{card}(h_2) = 1$ and $(s(x), s(x)) \notin h_2^+$ hold;
- (C) $(s, h_1 + h_2) \models x \hookrightarrow^+ x$, therefore $(s(x), s(x)) \in (h_1 + h_2)^+$ holds.

From (C), there is $\delta \geq 1$ such that $(s(x), s(x)) \in (h_1 + h_2)^\delta$. However, δ must be greater than 1, as otherwise either $h_1(s(x)) = s(x)$ or $h_2(s(x)) = s(x)$ hold, in contradiction with (A) and (B), respectively. Thus, $h_1 + h_2$ witnesses a path of length $\delta \geq 2$ going from $s(x)$ to $s(x)$. Let us picture this path as follows:

$$\{s(x) \mapsto \ell_1 \mapsto \ell_2 \mapsto \dots \mapsto \ell_{\delta-2} \mapsto \ell_{\delta-1} \mapsto s(x)\}, \quad \text{where } \ell_{\delta-1} \neq s(x) \text{ (since } \delta \geq 2\text{).}$$

From (A), it cannot be that the first arrow of the path, i.e. $s(x) \mapsto \ell_1$ in the representation above, is an element of h_1 . Hence, it must be an element of h_2 . More precisely, it must be the only element of h_2 (directly by (B)). Therefore, the last arrow of the path, i.e. $\ell_{\delta-1} \mapsto s(x)$ in the representation above, can only be an element of h_1 . We conclude that $s(x) \in \text{ran}(h_1)$ holds, which yields $s(x) \in \text{ran}(h)$ by $h_1 \subseteq h$. We leave the proof of the other direction to the reader. \square

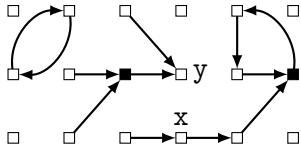


Figure 2.4: A memory state satisfying
 $\forall x (x \neq y \wedge _ \hookrightarrow x \Rightarrow x \hookrightarrow _).$

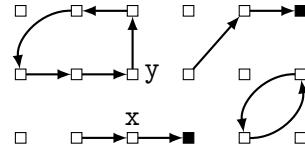


Figure 2.5: A memory state satisfying
 $\forall x \neg(_ \hookrightarrow x * _ \hookrightarrow x).$

Reachability predicates. Interestingly enough, the reachability predicate $x \hookrightarrow^+ y$ introduced for alloc-back can be defined in $SL(\exists, *, \neg)$. More precisely, this predicate can be defined in the two-variable fragment the logic, i.e. the fragment where VAR is restricted to only two variable names (in our case, case x and y). In order to stay in this fragment, in the definition of $x \hookrightarrow^+ y$ provided below the formulae $_ \hookrightarrow x$ and $_ \hookrightarrow y$ stand for $\exists y y \hookrightarrow x$ and $\exists x x \hookrightarrow y$, respectively.

$$x \hookrightarrow^+ y \stackrel{\text{def}}{=} \top * (x \hookrightarrow _ \wedge (_ \hookrightarrow x \Rightarrow x = y) \wedge \forall x \neg(_ \hookrightarrow x * _ \hookrightarrow x) \wedge \forall x (x \neq y \wedge _ \hookrightarrow x \Rightarrow x \hookrightarrow _)).$$

Since equivalent formulae are already defined in [22, 52, 53], instead of formally proving the correctness of $x \hookrightarrow^+ y$, let us try to understand the intention behind its definition. Let (s, h') be a memory state satisfying $x \hookrightarrow^+ y$. With its prefix “ $\top *$ ”, the formula $x \hookrightarrow^+ y$ imposes the existence of a subheap $h \subseteq h'$ such that the memory state (s, h) satisfies the following three formulae

1. $x \hookrightarrow _ \wedge (_ \hookrightarrow x \Rightarrow x = y),$
2. $\forall x \neg(_ \hookrightarrow x * _ \hookrightarrow x),$
3. $\forall x (x \neq y \wedge _ \hookrightarrow x \Rightarrow x \hookrightarrow _).$

Let us first consider the formula (3). Essentially, this formula states that for every location $\ell \in \text{dom}(h)$, either $h(\ell) = s(y)$ or $h(\ell) \in \text{dom}(h)$. Since the heap is finite, this property means that if h witnesses a non-empty directed path ending on a location $\ell \notin \text{dom}(h)$, then ℓ must be assigned to y . In other words, $\text{ran}(h) \setminus \text{dom}(h) \subseteq \{s(y)\}$. Figure 2.4 shows a memory state satisfying (3). Notice that every weakly connected component in this memory state has a cycle, with the exception of the one involving the location $s(y)$. On the other hand, the memory state in Figure 2.5, say (s_1, h_1) , does not satisfy this property, as there are two locations (highlighted with a black box \blacksquare) that are different from $s_1(y)$ but belong to $\text{ran}(h_1) \setminus \text{dom}(h_1)$. Let us move to the formula (2), which states that no location can be pointed by two distinct memory cells, i.e. for every $\ell, \ell' \in \text{dom}(h)$, if $h(\ell) = h(\ell')$ then $\ell = \ell'$. Figure 2.5 shows a memory state satisfying (2). Notice how every weakly connected component is either a linear structure or a cycle. This property is not satisfied by the memory state in Figure 2.4, as there are locations (again, highlighted with black boxes) pointed by multiple memory cells. Now, since (s, h) satisfies (1), we have $s(x) \in \text{dom}(h)$. As this memory state also satisfies (2), $s(x)$ either belongs to a cycle or to a linear structure. Since (s, h) satisfies (3), if $s(x)$ belongs to a linear structure, then the linear structure ends in $s(y)$ (which is then not in $\text{dom}(h)$) and therefore $(s(x), s(y)) \in h^+$. If instead $s(x)$ belongs to a cycle (i.e. $(s(x), s(x)) \in h^+$) then by definition $s(x)$ is pointed by a memory cell in $\text{dom}(h)$. Thus, from the right conjunct $_ \hookrightarrow x \Rightarrow x = y$ of the formula (1) we derive that $s(x) = s(y)$, which allows us to conclude again that $(s(x), s(y)) \in h^+$.

We write $x \hookrightarrow^* y$ for the formula $x = y \vee x \hookrightarrow^+ y$ (called *reach-star*). As the notation might suggest, the semantics of $x \hookrightarrow^* y$ can be given in terms of the Kleene closure of h , as shown below:

$$(s, h) \models x \hookrightarrow^* y \quad \text{if and only if} \quad (s(x), s(y)) \in h^*.$$

Informally, $(s, h) \models x \hookrightarrow^* y$ if h witnesses a (possibly empty) path going from $s(x)$ to $s(y)$.

Strict variants of points-to and reachability. Very often (see e.g. [124, 44, 93]), separation logic is defined by considering the so-called *strict predicates* $x \mapsto y$ and $\text{ls}(x, y)$ (where ls stands for *list-segment*). Given a memory state (s, h) , these two predicates are defined as

$$(s, h) \models x \mapsto y \quad \text{iff} \quad \{s(x)\} = \text{dom}(h) \text{ and } h(s(x)) = s(y).$$

$$(s, h) \models \text{ls}(x, y) \quad \text{iff} \quad \text{for every } \delta \in \mathbb{N}, h^\delta(s(x)) = s(y) \text{ if and only if } \delta = \text{card}(h).$$

Roughly speaking, these formulae are analogous of $x \hookrightarrow y$ and $x \hookrightarrow^* y$ but their satisfaction also requires that the formula cannot hold in any strict subheap. For instance, $\text{ls}(x, y)$ is satisfied whenever h describes a list (hence the name list-segment): a directed linear structure starting from $s(x)$ and ending with $s(y)$. In particular, if $s(x) = s(y)$ then the heap must be empty. For fragments of separation logic featuring emp , the separating conjunction and the classical negation, taking \hookrightarrow and \hookrightarrow^* instead of \mapsto and ls is just a matter of taste. Indeed, the two suites of operators are interdefinable. To show this, given a formula φ , we write $\text{strict}(\varphi)$ for the formula $\varphi \wedge \neg(\neg\text{emp} * \varphi)$, that has the following semantics:

$$(s, h) \models \text{strict}(\varphi) \quad \text{if and only if} \quad (s, h) \models \varphi \text{ and for every } h' \subsetneq h, (s, h') \not\models \varphi.$$

Then, \mapsto and \hookrightarrow , as well as ls and \hookrightarrow^* , are related following the four identities below:

$$(s, h) \models x \hookrightarrow y \quad \text{iff} \quad (s, h) \models x \mapsto y * \top, \quad (s, h) \models x \mapsto y \quad \text{iff} \quad (s, h) \models \text{strict}(x \hookrightarrow y),$$

$$(s, h) \models x \hookrightarrow^* y \quad \text{iff} \quad (s, h) \models \text{ls}(x, y) * \top, \quad (s, h) \models \text{ls}(x, y) \quad \text{iff} \quad (s, h) \models \text{strict}(x \hookrightarrow^* y).$$

2.1.2 The (not so) classical decision problems.

We conclude this introductory section on separation logic by displaying some features that make $\text{SL}(\exists, *, \neg)$ theoretically interesting. First, let us recall the concepts of validity and entailment. For simplicity, we define them for (separation) logics interpreted on memory states, and refer the reader to [17] for their general definition. We write $\models \varphi$ to state that a formula φ is *valid* (alternatively, φ is a *tautology*), i.e. φ is satisfied by every memory state. Instead, φ is said to *entail* the formula ψ , written $\varphi \models \psi$, whenever every memory state satisfying φ also satisfies ψ . We write $\varphi \equiv \psi$ when the two formulae φ and ψ are *equivalent*, that is when both $\varphi \models \psi$ and $\psi \models \varphi$ hold. Together with the *model-checking* problem and the *satisfiability* problem, deciding (semi-)algorithmically whether *validity* and *entailment* hold are among the most classical decision problems in logic. The description of these four problems is given in Figure 2.6.

For logics that are able to express the classical implication, as for instance $\text{SL}(\exists, *, \neg)$, it is well-known that validity and entailment are equireducible under many-one reductions. Indeed, to check whether $\models \varphi$ holds we can alternatively ask whether the entailment $\top \models \varphi$ is true. Similarly, the entailment $\varphi \models \psi$ can be verified by checking for the truth of $\models \varphi \Rightarrow \psi$. No further many-one reductions can be established between these four decision problems without requiring additional hypothesis on their complexity/decidability or on the expressive power of the logic. For instance, the following relation holds between validity and satisfiability:

$$\varphi \text{ is not valid} \quad \text{if and only if} \quad \neg\varphi \text{ is satisfiable.}$$

Therefore, we can check for the validity of φ by simply querying a procedure for satisfiability on input $\neg\varphi$. If the procedure replies “yes” then φ is not valid. If the procedure replies “no” then the formula is valid. However, this is not a many-one reduction (it is actually an instance of a Turing reduction), as we are negating the answer obtained from the satisfiability procedure.

-
- model-checking:** *Input:* A formula φ and a memory state (s, h) .
Question: Does (s, h) satisfy φ ? (i.e. is $(s, h) \models \varphi$ true?)
- satisfiability:** *Input:* A formula φ .
Question: Is there a memory state (s, h) satisfying φ ?
- validity:** *Input:* A formula φ .
Question: Does $\models \varphi$ hold?
- entailment:** *Input:* A pair of formulae (φ, ψ) .
Question: Does $\varphi \models \psi$ hold?

Figure 2.6: The decision problems of model-checking, satisfiability, validity and entailment.

In fact, the above double implication leads to a many-one reduction from unValidity (i.e. the complement of validity) to satisfiability.

Surprisingly, for $\text{SL}(\exists, *, \rightarrow)$ (as well as in many other separation logics) the landscape is quite different: the four decision problems in Figure 2.6 are all many-one equireducible.

Theorem 2.8. The problems of model-checking, satisfiability, validity and entailment for the logic $\text{SL}(\exists, *, \rightarrow)$ are all many-one equireducible (under log-space reductions).

We already saw that entailment reduces to validity (and vice versa). To prove this result it is then sufficient to show that the model-checking problem reduces to the entailment problem, that the validity problem reduces to the satisfiability problem, and that the satisfiability problem reduces to model-checking. To perform these reductions we first need to introduce the notions of X -heap-isomorphic memory states [56].

Definition 2.9 (X -heap-isomorphism). Let $X \subseteq \text{VAR}$. Two memory states (s_1, h_1) and (s_2, h_2) are X -heap-isomorphic, written $(s_1, h_1) \simeq_X (s_2, h_2)$, if there is a function $f : \text{LOC} \rightarrow \text{LOC}$ s.t.

1. f is a bijection,
2. $h_2 = \{(f(\ell_1), f(\ell_2)) \mid (\ell_1, \ell_2) \in h_1\}$,
3. for every $x \in X$, $f(s_1(x)) = s_2(x)$.

A function f satisfying these conditions is called a X -heap-isomorphism from (s_1, h_1) to (s_2, h_2) .

In this definition, the conditions (1) and (2) essentially state that f is a graph isomorphism between h_1 and h_2 , while the condition (3) extends this isomorphism to the variables in X . It is easy to check that \simeq_X is an equivalence relation. A folklore result states that no formula of $\text{SL}(\exists, *, \rightarrow)$ written with free-variables in X can distinguish between two X -heap-isomorphic memory states. The precise statement is given below.

Proposition 2.10. Let $X \subseteq \text{VAR}$ and let $(s_1, h_1), (s_2, h_2)$ be memory states s.t. $(s_1, h_1) \simeq_X (s_2, h_2)$. Given φ in $\text{SL}(\exists, *, \rightarrow)$ whose free variables are among X , $(s_1, h_1) \models \varphi$ if and only if $(s_2, h_2) \models \varphi$.

Since a slight generalisation of this result is proven in Section 3.1 (see Lemma 3.3), we omit the proof of this proposition (which carries out by structural induction on the formula). Given a formula φ , notice that this result shows us how to define a finite representation of a memory state (s, h) satisfying φ : it is sufficient to restrict s to the variables occurring in φ (thus including the quantified variables, in order to preserve the semantics of the first-order quantification), and

then encode the memory state as a pair of finite binary relations. This is important for the model-checking problem, where the memory state is part of the input. The *size* of this finite representation of (s, h) is defined naturally by considering a reasonably succinct encoding of the binary relations s (restricted as discussed above) and h . We are now ready to reduce the model-checking problem to the entailment problem.

Lemma 2.11. Model-checking for $\text{SL}(\exists, *, -*)$ is log-space reducible to its entailment problem.

Proof. In short, the reduction is achieved by internalising a memory state as a formula of $\text{SL}(\exists, *, -*)$ and relying on Proposition 2.10. Let (s, h) and φ be a memory state and a formula of $\text{SL}(\exists, *, -*)$, respectively. We start by constructing a polynomial-size formula describing (s, h) . Let $X = \text{fv}(\varphi)$ and let L be the finite set of locations in $\text{dom}(h) \cup \text{ran}(h) \cup \{s(x) \mid x \in X\}$. For every location $\ell \in L$, we introduce a distinct variable name x_ℓ that does not appear in φ . We write $\Gamma\langle s, h \rangle$ for the following formula in $\text{SL}(\exists, *, -*)$:

$$\Gamma\langle s, h \rangle \stackrel{\text{def}}{=} (\bigstar_{(\ell_1, \ell_2) \in h} x_{\ell_1} \mapsto x_{\ell_2}) \wedge (\bigwedge_{x \in X} x = x_{s(x)}) \wedge (\bigwedge_{\substack{\ell_1, \ell_2 \in L \\ \ell_1 \neq \ell_2}} x_{\ell_1} \neq x_{\ell_2})$$

where given a set $S = \{e_1, \dots, e_n\}$, we write $\bigstar_{e \in S} \psi(e)$ for the formula $\psi(e_1) * \dots * \psi(e_n)$ (similarly to the definition of $\bigwedge_{e \in S} \psi(e)$). Informally, $\Gamma\langle s, h \rangle$ describes the memory state (s, h) by internalising every $(\ell_1, \ell_2) \in h$ with the strict points-to predicate $x_{\ell_1} \mapsto x_{\ell_2}$, and requiring that $x_{\ell_1} \neq x_{\ell_2}$ holds for every two distinct locations $\ell_1, \ell_2 \in L$. Moreover, since the strict points-to predicates are separated with the operator $*$, every heap of a memory state satisfying $\Gamma\langle s, h \rangle$ must be graph-isomorphic to h . Lastly, the store is internalised by means of the conjunctions $\bigwedge_{x \in X} x = x_{s(x)}$. The formula $\Gamma\langle s, h \rangle$ enjoys the two following properties:

- A. $\Gamma\langle s, h \rangle$ is satisfiable,
- B. for every (s', h') , if $(s', h') \models \Gamma\langle s, h \rangle$ then $(s', h') \simeq_X (s, h)$.

The proofs of (A) and (B) are quite straightforward. For (A), from the definition of $\Gamma\langle s, h \rangle$ it is quite clear that this formula is satisfied by the memory state $(s[x_\ell \leftarrow \ell \mid \ell \in L], h)$, where $s[x_\ell \leftarrow \ell \mid \ell \in L]$ is the store obtained from s by assigning each location $\ell \in L$ to x_ℓ . For (B), given a memory state (s', h') satisfying $\Gamma\langle s, h \rangle$, we can obtain a X -heap-isomorphism from (s', h') to (s, h) by simply constructing a bijection f such that $f(s'(x_\ell)) = \ell$ holds for every $\ell \in L$. Details are omitted for the sake of brevity.

We can now derive the central property of $\Gamma\langle s, h \rangle$:

$$(s, h) \models \varphi \text{ if and only if the entailment } \Gamma\langle s, h \rangle \models \varphi \text{ holds.}$$

(\Rightarrow): Taking the contrapositive, suppose that $\Gamma\langle s, h \rangle \models \varphi$ does not hold, and therefore that there is a memory state (s', h') such that $(s', h') \models \Gamma\langle s, h \rangle$ but $(s', h') \not\models \varphi$. From (B), $(s', h') \simeq_X (s, h)$ and directly from Proposition 2.10, we conclude that $(s, h) \models \varphi$ does not hold.

(\Leftarrow): Suppose that $\Gamma\langle s, h \rangle \models \varphi$ holds. Let us consider a memory state (s', h') satisfying $\Gamma\langle s, h \rangle$ (its existence is guaranteed by (A)). From $\Gamma\langle s, h \rangle \models \varphi$ we derive $(s', h') \models \varphi$. Moreover, from $(s', h') \models \Gamma\langle s, h \rangle$ we have $(s', h') \simeq_X (s, h)$ by (B). Then, since $X = \text{fv}(\varphi)$, by Proposition 2.10 we derive $(s, h) \models \varphi$.

The above equivalence leads to a many-one reduction: to check whether $(s, h) \models \varphi$ holds we can check for the entailment $\Gamma\langle s, h \rangle \models \varphi$, where $\Gamma\langle s, h \rangle$ is a log-space computable polynomial-size (w.r.t. $\text{card}(\text{fv}(\varphi))$ and $\text{card}(h)$) encoding of (s, h) into a formula of $\text{SL}(\exists, *, -*)$. \square

In order to achieve the two other reductions (from validity to satisfiability and from satisfiability to model-checking), we internalise the quantification on memory states required by validity and satisfiability by using the first-order quantification and the separating implication.

Lemma 2.12. (I) Validity for $\text{SL}(\exists, *, \neg*)$ is log-space reducible to its satisfiability problem.
 (II) Satisfiability for $\text{SL}(\exists, *, \neg*)$ is log-space reducible to its model-checking problem.

Proof of (I). Let φ be a formula of $\text{SL}(\exists, *, \neg*)$ such that $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} = \text{fv}(\varphi)$. In order to prove (I) it is sufficient to show the following equivalence:

$$\varphi \text{ is valid if and only if the formula } \mathbf{emp} \wedge \forall \mathbf{x}_1 \dots \forall \mathbf{x}_n (\top \neg* \varphi) \text{ is satisfiable.}$$

(\Rightarrow): Suppose φ to be valid. Hence, for all n locations $\ell_1, \dots, \ell_n \in \text{LOC}$, every store s and every heap h , we have $(s[\mathbf{x}_i \leftarrow \ell_i \mid i \in [1, n]], h) \models \varphi$. Thanks to the semantics of the operator $\neg*$, we can internalise the universally quantified heap h and derive that for every store s and all n locations $\ell_1, \dots, \ell_n \in \text{LOC}$, $(s[\mathbf{x}_i \leftarrow \ell_i \mid i \in [1, n]], \emptyset) \models \top \neg* \varphi$ holds. Directly from the definition of the first-order quantification we derive that $(s, \emptyset) \models \forall \mathbf{x}_1 \dots \forall \mathbf{x}_n (\top \neg* \varphi)$ holds for every store s . Moreover, $(s, \emptyset) \models \mathbf{emp}$ so that the formula $\mathbf{emp} \wedge \forall \mathbf{x}_1 \dots \forall \mathbf{x}_n (\top \neg* \varphi)$ is satisfiable.

(\Leftarrow): Suppose that $\mathbf{emp} \wedge \forall \mathbf{x}_1 \dots \forall \mathbf{x}_n (\top \neg* \varphi)$ is satisfied by a memory state (s, h) . Since we have $(s, h) \models \mathbf{emp}$, the heap h must be empty. Thus, from the semantics of $\forall \mathbf{x}_1 \dots \forall \mathbf{x}_n (\top \neg* \varphi)$ we conclude that

A. for all n locations $\ell_1, \dots, \ell_n \in \text{LOC}$ and every heap h' , $(s[\mathbf{x}_i \leftarrow \ell_i \mid i \in [1, n]], h') \models \varphi$.

Let us now consider a memory state (s_1, h_1) , and show that $(s_1, h_1) \models \varphi$ (leading to φ being valid). From (A), the memory state $(s[\mathbf{x}_i \leftarrow s_1(\mathbf{x}_i) \mid i \in [1, n]], h_1)$ satisfies φ . It is quite clear that (s_1, h_1) and $(s[\mathbf{x}_i \leftarrow s_1(\mathbf{x}_i) \mid i \in [1, n]], h_1)$ are $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ -heap isomorphic. Then, since $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} = \text{fv}(\varphi)$, by Proposition 2.10 we conclude that $(s_1, h_1) \models \varphi$. \square

Proof of (II). Let φ be a formula of $\text{SL}(\exists, *, \neg*)$ such that $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} = \text{fv}(\varphi)$. Similarly to (I), in order to prove (II) it is sufficient to show the following equivalence:

$$\varphi \text{ is satisfiable if and only if } (s, \emptyset) \models \exists \mathbf{x}_1 \dots \exists \mathbf{x}_n (\varphi \neg* \top) \text{ holds.}$$

where s is an arbitrary (fixed) store.

(\Rightarrow): Suppose that φ is satisfied by a memory state (s_1, h_1) . First, we update the store s into $s[\mathbf{x}_i \leftarrow s_1(\mathbf{x}_i) \mid i \in [1, n]]$, so that the memory state $(s[\mathbf{x}_i \leftarrow s_1(\mathbf{x}_i) \mid i \in [1, n]], h_1)$ is $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ -heap isomorphic with (s_1, h_1) . By Proposition 2.10, we then conclude that the memory state $(s[\mathbf{x}_i \leftarrow s_1(\mathbf{x}_i) \mid i \in [1, n]], h_1)$ satisfies φ . Weakening this last statement, we have that there are n locations ℓ_1, \dots, ℓ_n and a heap h_1 such that $(s[\mathbf{x}_i \leftarrow \ell_i \mid i \in [1, n]], h_1) \models \varphi$. Thanks to the semantics of the subtraction $\neg*$, we can internalise the existentially quantified heap h_1 and derive that there are n locations ℓ_1, \dots, ℓ_n such that $(s[\mathbf{x}_i \leftarrow \ell_i \mid i \in [1, n]], \emptyset) \models \varphi \neg* \top$. From the definition of first-order quantifier we conclude that $(s, \emptyset) \models \exists \mathbf{x}_1 \dots \exists \mathbf{x}_n (\varphi \neg* \top)$ holds.
 (\Leftarrow): Suppose that $(s, \emptyset) \models \exists \mathbf{x}_1 \dots \exists \mathbf{x}_n (\varphi \neg* \top)$ holds. Then, directly from the semantics of first-order quantification and subtraction, we conclude that there are n locations $\ell_1, \dots, \ell_n \in \text{LOC}$ and a heap h such that $(s[\mathbf{x}_i \leftarrow \ell_i \mid i \in [1, n]], h) \models \varphi$. Thus, φ is satisfiable. \square

The two Lemmata 2.11 and 2.12 directly prove Theorem 2.8. It should be noted that similar results can be achieved for many variants of separation logics. For instance, an analogous theorem can already be shown for the quantifier-free fragment of $\text{SL}(\exists, *, \neg*)$ (as discussed in Chapter 6). The fact that, in separation logic, the four classical decision problems are all many-one equireducible has a profound impact on the results and techniques used in this thesis. For example, a standard way to provide an algorithm for the satisfiability problem consists into showing a upper bound on the smallest model that must satisfy a formula φ , and then check for $(s, h) \models \varphi$ on every memory state (s, h) below that bound. However, this requires

an algorithm for the model-checking problem, which by Theorem 2.8 is equivalent to satisfiability. Moreover, Theorem 2.8 implies that it is not possible to provide sound and complete axiom system of undecidable separation logics. Recall that an axiomatisable logic must have a *recursively enumerable* (RE) validity problem. Suppose the validity problem of $\text{SL}(\exists, *, \neg*)$ to be undecidable but recursively enumerable. Then, by Theorem 2.8 its satisfiability problem is RE, which implies that unvalidity is RE. However, as both validity and its complement are found to be RE, we conclude that validity is recursive, hence decidable, a contradiction. The satisfiability problem of $\text{SL}(\exists, *, \neg*)$ has been proven undecidable in [22], which leads to the following result.

Theorem 2.13. Model-checking, satisfiability, validity and entailment of $\text{SL}(\exists, *, \neg*)$ are not RE.

2.2 FRAGMENTS OF $\text{SL}(\exists, *, \neg*)$ AND SECOND-ORDER LOGIC

Now that we are quite familiar with $\text{SL}(\exists, *, \neg*)$, we look at various known complexity results concerning its fragments. For the moment, we only consider fragments that are closed under Boolean connectives. Some of these results can be motivated by looking at the translation of $\text{SL}(\exists, *, \neg*)$ into second-order logic. We take this opportunity to introduce weak (monadic) second-order logic and recall some landmark results on its decidability status.

2.2.1 Fragments of $\text{SL}(\exists, *, \neg*)$.

Since a good chunk of this thesis deals with the computational complexity of various separation logics, we need to be rather systematic with the notation in order not to get lost in the various fragments, extensions and variants we consider. Unless otherwise specified, we propose to write SL for a logic interpreted on memory states and featuring \top , the three predicates emp , $x = y$ and $x \hookrightarrow y$, and Boolean connectives. Additional elements are added in brackets following the lexeme SL . We already used this notation for the first-order separation logic $\text{SL}(\exists, *, \neg*)$, which is indeed obtained from the logic SL by adding both multiplicative connectives and the first-order quantifier. Given $n \geq 1$, we write $[\exists]_n$ for the restriction of the first-order quantifier to n distinct program variable names. For instance, $\text{SL}([\exists]_2, *, \neg*)$ is the restriction of $\text{SL}(\exists, *, \neg*)$ to only two quantified variable names. This restriction is purely syntactical: given a formula φ in $\text{SL}(\exists, *, \neg*)$, it is in $\text{SL}([\exists]_2, *, \neg*)$ if and only if $\text{card}(\text{bv}(\varphi)) \leq 2$. Given a logic \mathcal{L} featuring first-order quantification, we write $\text{pnf-}\mathcal{L}$ for the subset of its formulae in prenex normal form, i.e. formulae where the quantification appears as a prefix of an otherwise quantifier-free formula. For instance, $\text{pnf-}\text{SL}(\exists, *, \neg*)$ is the set of formulae in $\text{SL}(\exists, *, \neg*)$ that are in prenex normal form. If a specific quantifier alternation is considered, we write its characteristic language instead of the prefix pnf- . For example, $\exists^*\forall^*\text{SL}(\exists, *, \neg*)$ is the set of prenex formulae of $\text{SL}(\exists, *, \neg*)$ with quantifier prefix from the language $\exists^*\forall^*$. As in classical first-order logic, we call the class of formulae with this prefix the Bernays-Schönfinkel-Ramsey (BSR) fragment of $\text{SL}(\exists, *, \neg*)$. Lastly, following [104], we categorise separation logics depending on the presence of the separating implication $\neg*$. Separation logics featuring this connective are said to be *extensional*, and otherwise they are said to be *intensional*.

Figure 2.7 recalls known complexity results for fragments of first-order separation logic interpreted on memory states (s, h) as in Definition 2.1, where $s : \text{VAR} \rightarrow \text{LOC}$ and $h : \text{LOC} \rightarrow_{\text{fin}} \text{LOC}$. The results in the figure refer for the three decision problems of satisfiability, validity and entailment, and every problem is complete for the complexity class the respective logic is placed in. An arrow going from a logic \mathcal{L}_1 to a logic \mathcal{L}_2 means that \mathcal{L}_1 is a syntactical fragment of \mathcal{L}_2 .

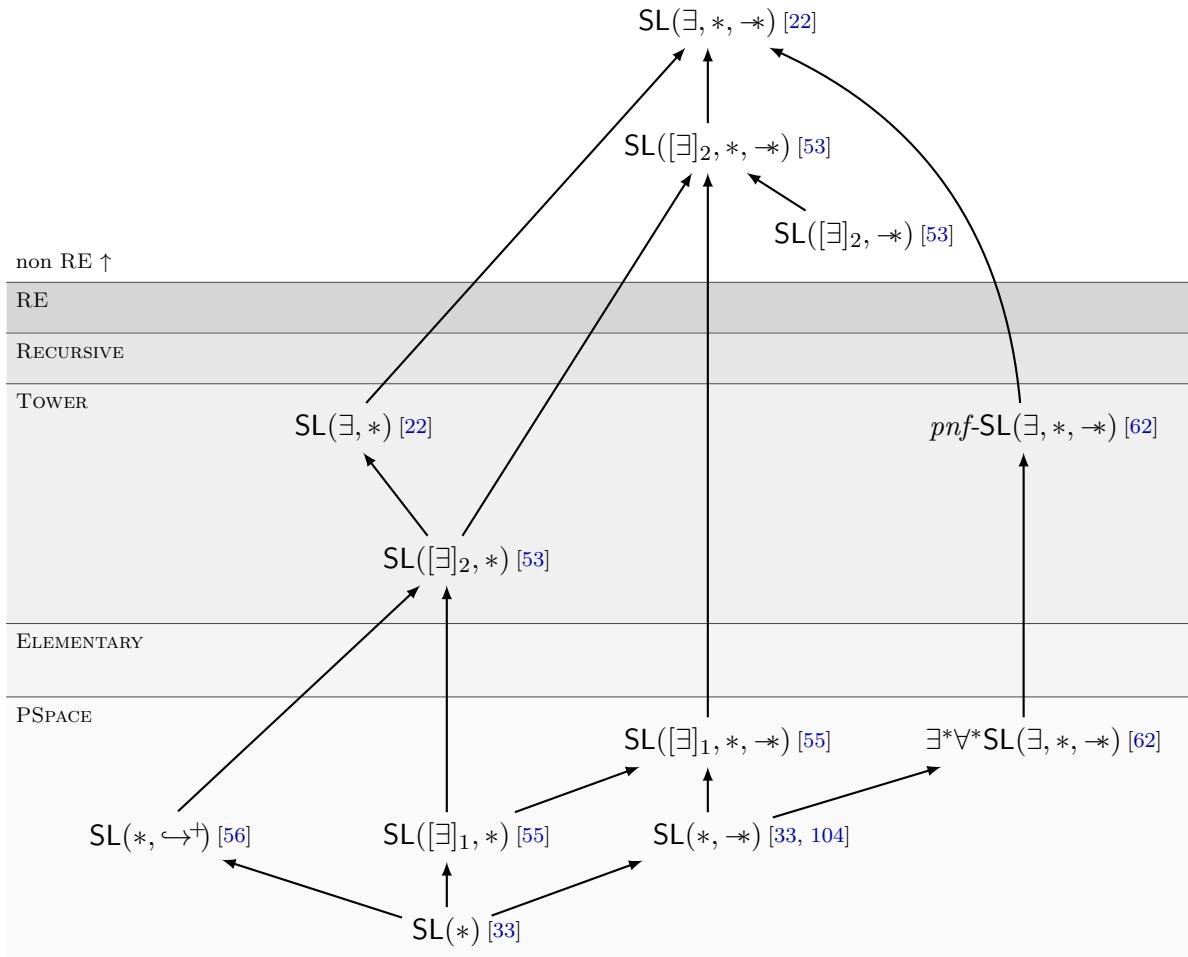


Figure 2.7: The complexity of Separation Logics.

This is by no means a complete list, but it will definitely help to place our work in the right context.

As we can see, the satisfiability problem of the known fragments of $\text{SL}(\exists, *, \neg)$ tends to belong to three classes: the class of PSPACE-complete problems, the one of TOWER-complete problems and the class of non recursively enumerable problems. We recall that PSPACE is the complexity class of all the decision problems that can be solved by a deterministic Turing machine using a polynomial amount of space, with respect to the size of the input. Instead, the TOWER complexity class is way up in the complexity hierarchy. It stands for the class of all decision problems that can be solved by a deterministic Turing machine in time bounded by a tower of exponentials of height depending elementarily on the size of the input. It is equivalent to the computational class \mathbf{F}_3 of the hierarchy of non-elementary complexity classes introduced by S. Schmitz in [128]. Interestingly, adding small features to a separation logic can cause quite a jump in terms of its complexity. For instance, the logic $\text{SL}([\exists]_1, *, \neg)$, i.e. $\text{SL}(\exists, *, \neg)$ restricted to just one quantified variable, admits a PSPACE-complete satisfiability problem [55], which jumps to non RE when a second quantified variable is allowed [53]. More precisely, non recursive enumerability already holds for the set of closed formulae of $\text{SL}([\exists]_2, \neg)$ [53].

$$\begin{aligned}
 (\mathcal{D}, r) \models S(\mathbf{x}_1, \dots, \mathbf{x}_n) &\quad \text{iff} \quad (r(\mathbf{x}_1), \dots, r(\mathbf{x}_n)) \in r(S), \\
 (\mathcal{D}, r) \models \exists \mathbf{z} \varphi &\quad \text{iff} \quad \text{there is } d \in \mathcal{D} \text{ such that } (\mathcal{D}, r[\mathbf{z} \leftarrow d]) \models \varphi, \\
 (\mathcal{D}, r) \models \exists S \varphi &\quad \text{iff} \quad \text{there is } R \subseteq_{\text{fin}} \mathcal{D}^{\text{ar}(S)} \text{ such that } (\mathcal{D}, r[S \leftarrow R]) \models \varphi.
 \end{aligned}$$

Figure 2.8: Satisfaction relation for WSO, with respect to a structure (\mathcal{D}, r) .

2.2.2 $\text{SL}(\exists, *, -*)$ as a Fragment of Second-Order Logic.

In order to understand the non-elementary complexity results of Figure 2.7 it is helpful to look at the translation of $\text{SL}(\exists, *, -*)$ into weak second-order logic. Indeed, as we now show, both multiplicative connectives are second-order in nature, with only $*$ being expressible in monadic second-order logic. In what follows, we briefly recall notions of weak second-order logic, and refer the reader to [17] for a complete investigation of this topic. The formulae of weak second-order logic are built using two kinds of variables: *individual* (or first-order) variables and *relation* (or second-order) variables. The first set of variable is denoted by $\mathbf{x}, \mathbf{y}, \mathbf{z} \dots$ (as done for separation logic), whereas for the relation variables we use S, H, \dots . Every second-order variable S is endowed with its arity $\text{ar}(S)$, which is a positive natural number. The formulae φ of the *weak second-order logic* WSO are defined from the following grammar:

$$\begin{array}{llll}
 \pi := \top & \quad (\text{true}) & \varphi := \pi & \quad (\text{atomic formulae}) \\
 | \quad \mathbf{x} = \mathbf{y} & \quad (\text{equality predicate}) & | \quad \varphi \wedge \varphi & \quad (\text{Boolean connectives}) \\
 | \quad S(\mathbf{x}_1, \dots, \mathbf{x}_{\text{ar}(S)}) & \quad (\text{relational predicate}) & | \quad \exists \mathbf{z} \varphi & \quad (\text{first-order quantifier}) \\
 & & | \quad \exists S \varphi & \quad (\text{second-order quantifier})
 \end{array}$$

A classical interpretation for the formulae of WSO is given by a relational structure over a non-empty domain \mathcal{D} , as we formally define below.

Definition 2.14 (WSO structure). A WSO *structure* is a pair (\mathcal{D}, r) where \mathcal{D} is a non-empty set called *domain* and r is an *assignment* function that maps every first-order variable \mathbf{x} to an element $r(\mathbf{x}) \in \mathcal{D}$, and every second-order variable S to a finite $\text{ar}(S)$ -ary relation $r(S) \subseteq_{\text{fin}} \mathcal{D}^{\text{ar}(S)}$.

The satisfaction relation \models for the formulae of WSO is formalised in Figure 2.8, omitting the standard clauses for \top and Boolean connectives. In particular, notice that the semantics of $\exists S \varphi$ updates the assignment function r so that S is mapped to a new finite $\text{ar}(S)$ -ary relation R . As usual, the first and second-order universal quantifications $\forall \mathbf{z} \varphi$ and $\forall S \varphi$ are defined as the dual of the existential quantifiers, i.e. $\neg \exists \mathbf{z} \neg \varphi$ and $\neg \exists S \neg \varphi$, respectively.

As shown in [22], weak second-order logic can internalise the semantics of $\text{SL}(\exists, *, -*)$ by using relation variables to simulate the heap and second-order quantification to simulate the multiplicative connectives. Without loss of generality, we assume VAR to be the set of first-order variables of WSO. We use second order variables H, H_1, H_2, \dots of arity 2 in order to represent heaps. To correctly characterise a heap, H must be a weakly functional binary relation, which can be enforced thanks to the following formula:

$$\text{fun}(H) \stackrel{\text{def}}{=} \forall \mathbf{x} \forall \mathbf{y} \forall \mathbf{z} (H(\mathbf{x}, \mathbf{y}) \wedge H(\mathbf{x}, \mathbf{z}) \Rightarrow \mathbf{y} = \mathbf{z}).$$

The correctness of this formula is quite easy to grasp, especially if we think in terms of $\text{SL}(\exists, *, -*)$, where it corresponds to the tautology $\forall \mathbf{x} \forall \mathbf{y} \forall \mathbf{z} (\mathbf{x} \hookrightarrow \mathbf{y} \wedge \mathbf{x} \hookrightarrow \mathbf{z} \Rightarrow \mathbf{y} = \mathbf{z})$. To internalise the

$\tau_H(\text{emp})$	$\stackrel{\text{def}}{=} H \perp H,$	$\tau_H(\neg\psi)$	$\stackrel{\text{def}}{=} \neg\tau_H(\psi),$
$\tau_H(x = y)$	$\stackrel{\text{def}}{=} x = y,$	$\tau_H(\psi_1 \wedge \psi_2)$	$\stackrel{\text{def}}{=} \tau_H(\psi_1) \wedge \tau_H(\psi_2),$
$\tau_H(x \rightarrowtail y)$	$\stackrel{\text{def}}{=} H(x, y),$	$\tau_H(\exists z \psi)$	$\stackrel{\text{def}}{=} \exists z \tau_H(\psi),$
$\tau_H(\psi_1 * \psi_2)$	$\stackrel{\text{def}}{=} \exists H_1 \exists H_2 ([H : H_1, H_2] \wedge \tau_{H_1}(\psi_1) \wedge \tau_{H_2}(\psi_2)),$		
$\tau_H(\psi_1 \multimap \psi_2)$	$\stackrel{\text{def}}{=} \forall H_1 \forall H_2 (\text{fun}(H_1) \wedge [H_2 : H, H_1] \wedge \tau_{H_1}(\psi_1) \Rightarrow \tau_{H_2}(\psi_2)).$		

Figure 2.9: Translating $\text{SL}(\exists, *, -*)$ to WSO. H , H_1 and H_2 are syntactically different.

multiplicative connectives we need to express the notion of union of two heaps. Given second-order variables H , H_1 and H_2 corresponding to three weakly functional binary relations, We first define a formula $H_1 \perp H_2$ that captures the notion of disjointness of two heaps, and that is then used to define the formula $[H : H_1, H_2]$ stating that H is the union of H_1 and H_2 :

$$\begin{aligned} H_1 \perp H_2 &\stackrel{\text{def}}{=} \forall x \forall y \forall z (\neg H_1(x, y) \vee \neg H_2(x, z)), \\ [H : H_1, H_2] &\stackrel{\text{def}}{=} H_1 \perp H_2 \wedge \forall x \forall y (H_1(x, y) \vee H_2(x, y) \Leftrightarrow H(x, y)). \end{aligned}$$

Both formulae closely follow the set-theoretical notions of disjointness and union of heaps. Notice that the formula $H \perp H$ is satisfiable only in models where H corresponds to the empty relation, and can be used to characterise the predicate `emp`. Thanks to these formulae we can translate a formula φ in $\text{SL}(\exists, *, -*)$ into an equivalent formula $\tau_H(\varphi)$ in WSO, where H is a second-order variable that represents the heap. The translation is given in Figure 2.9, and one can check that it simply rewrites the semantics of the various ingredients of $\text{SL}(\exists, *, -*)$ directly in WSO. A last ingredient is needed: the domain \mathcal{D} of a WSO structure must be infinite, so that it is isomorphic to `LOC`. This can be done with the formula $\text{Dom}[\omega] \stackrel{\text{def}}{=} \forall S \exists x \neg S(x)$ stating that no finite set can contain all the elements of \mathcal{D} (here, S and x are arbitrary variables). The following proposition, whose proof can be found in [22], shows the correctness of this translation.

Proposition 2.15 (From [22]). Let φ be a formula of $\text{SL}(\exists, *, -*)$.

- I. Let (s, h) be a memory state. Let (LOC, r) be a WSO structure such that $r(H) = h$ and for every $x \in \text{VAR}$ $r(x) = s(x)$. We have, $(s, h) \models \varphi$ if and only if $(\text{LOC}, r) \models \tau_H(\varphi)$.
- II. φ and $\text{Dom}[\omega] \wedge \text{fun}(H) \wedge \tau_H(\varphi)$ are equisatisfiable.
- III. φ and $\text{Dom}[\omega] \wedge \text{fun}(H) \Rightarrow \tau_H(\varphi)$ are equivalent.

It is well-known that the satisfiability and validity problems of WSO are not recursively enumerable (here, Theorem 2.13 reproves this result). Now, we can ask ourselves under which conditions the translation can be revisited so that it stays in a decidable fragment of WSO. Between the two multiplicative connectives, we notice that the magic wand seems to be the more delicate one. Given a structure (\mathcal{D}, r) , the translation of $\tau_H(\psi_1 * \psi_2)$ ask to find a partition $\{R_1, R_2\}$ of $r(H)$ so that $\tau_{H_1}(\psi_1)$ and $\tau_{H_2}(\psi_2)$ are satisfied by $(\mathcal{D}, r[H_1 \leftarrow R_1, H_2 \leftarrow R_2])$. Therefore, when the interpretation for H is fixed, there are only a finite number of such partitions. Unfortunately, this fundamental property does not hold for the operator $-*$: even when the interpretation of H is fixed, the set of interpretations of H_1 and H_2 is a priori infinite. Fundamentally, it is thanks to this property that we reduced the satisfiability of $\text{SL}(\exists, *, -*)$ to its model-checking problem (Lemma 2.12(II)). Indeed, starting from H interpreted as the empty

$$\begin{aligned}
(\mathcal{D}, r, \dot{\mathfrak{f}}) \models S(x_1) &\quad \text{iff} \quad r(x_1) \in r(S), \\
(\mathcal{D}, r, \dot{\mathfrak{f}}) \models f(x, y) &\quad \text{iff} \quad \dot{\mathfrak{f}}(r(x)) = r(y), \\
(\mathcal{D}, r, \dot{\mathfrak{f}}) \models \exists z \varphi &\quad \text{iff} \quad \text{there is } d \in \mathcal{D} \text{ such that } (\mathcal{D}, r[z \leftarrow d], \dot{\mathfrak{f}}) \models \varphi, \\
(\mathcal{D}, r, \dot{\mathfrak{f}}) \models \exists S \varphi &\quad \text{iff} \quad \text{there is } R \subseteq_{\text{fin}} \mathcal{D} \text{ such that } (\mathcal{D}, r[S \leftarrow R], \dot{\mathfrak{f}}) \models \varphi.
\end{aligned}$$

Figure 2.10: Satisfaction relation for $\text{WMSO}^{\dot{\mathfrak{f}}}$, with respect to a structure (\mathcal{D}, r) .

relation, $\tau_H(\varphi \multimap \top)$ asks if it is possible to find, among the infinite interpretations of H_1 , one that is weakly functional and makes $\tau_{H_1}(\varphi)$ true. As extensively studied in [22], one needs in fact WSO in order to express the separating implication of $\text{SL}(\exists, *, \multimap)$, and this operator together with first-order quantification is enough to express every property of WSO. This property essentially leads to all the complexity results in the “non RE” area of Figure 2.7.

What happens if we forbid the separating implication? If we consider the logic $\text{SL}(\exists, *)$, we can exploit the fact that $\tau_H(\psi_1 * \psi_2)$ only considers partitions of the interpretation of H in order to push the translation $\tau_H(\varphi)$ into the monadic fragment of WSO with an additional unary function symbol ($\text{WMSO}^{\dot{\mathfrak{f}}}$). Monadic WSO (WMSO) is the fragment of WSO where every relation variable S has arity $\text{ar}(S) = 1$. $\text{WMSO}^{\dot{\mathfrak{f}}}$ extends the vocabulary of WMSO by adding a symbol $\dot{\mathfrak{f}}$ of arity 2 and endowing a structure (\mathcal{D}, r) with an interpretation $\dot{\mathfrak{f}} : \mathcal{D} \rightarrow \mathcal{D}$ for $\dot{\mathfrak{f}}$. Formulae of $\text{WMSO}^{\dot{\mathfrak{f}}}$ do not quantify over $\dot{\mathfrak{f}}$, as shown by the satisfaction relation defined in Figure 2.10. One can show that $\text{WMSO}^{\dot{\mathfrak{f}}}$ is still a fragment of WSO, as $\dot{\mathfrak{f}}$ can be substituted with a relation variable F of arity 2 satisfying the axioms of functions:

- $\forall x \exists y F(x, y)$, i.e. F is interpreted by a binary relation R such that $\pi_1(R) = \mathcal{D}$,
- $\text{fun}(F)$, i.e. F is interpreted by a weakly functional relation.

The satisfiability and validity problems of $\text{WMSO}^{\dot{\mathfrak{f}}}$ are TOWER-complete. The upper bound was famously shown by M. O. Rabin in [122], whereas the non-elementary lower-bound can be traced back to the seminal works of A. R. Meyer and L. J. Stockmeyer [110, 111]. See [128] for the TOWER-characterisation of these problems.

Let φ be a formula of $\text{SL}(\exists, *)$. The main idea that allows us to modify the translation in Figure 2.9 so that $\tau_H(\varphi)$ is in $\text{WMSO}^{\dot{\mathfrak{f}}}$ is to notice that the notion of disjointness of two heaps only depends on their domain. Instead of characterising heaps entirely, we now use the unary relation variables H , H_1 and H_2 to only describe their domains. We rely on the symbol $\dot{\mathfrak{f}}$ in order to encode the heap. The formulae $H_1 \perp H_2$ and $[H : H_1 + H_2]$ are updated as follows:

$$H_1 \perp H_2 \stackrel{\text{def}}{=} \forall x \neg(H_1(x) \wedge H_2(x)), \quad [H : H_1 + H_2] \stackrel{\text{def}}{=} H_1 \perp H_2 \wedge \forall x (H_1(x) \vee H_2(x) \Leftrightarrow H(x)).$$

With these new definitions, the translation $\tau_H(\varphi)$ is defined as in Figure 2.9, the only two differences being that there is no case for the operator \multimap and that $\tau_H(x \leftarrow y)$ is now defined as $H(x) \wedge \dot{\mathfrak{f}}(x, y)$. Proposition 2.15 is updated accordingly, again as shown in [22].

Proposition 2.16 (From [22]). Let φ be a formula of $\text{SL}(\exists, *)$.

- I. Let (s, h) be a memory state. Let $(\text{LOC}, r, \dot{\mathfrak{f}})$ be a $\text{WMSO}^{\dot{\mathfrak{f}}}$ structure s.t. $r(H) = \text{dom}(h)$ and for every $x \in \text{VAR}$ $r(x) = s(x)$. We have, $(s, h) \models \varphi$ if and only if $(\text{LOC}, r, \dot{\mathfrak{f}}) \models \tau_H(\varphi)$.
- II. φ and $\text{Dom}[\omega] \wedge \tau_H(\varphi)$ are equisatisfiable.

III. φ and $\text{Dom}[\omega] \Rightarrow \tau_H(\varphi)$ are equivalid.

This result has been used for all the upper bounds of the problems in the “TOWER area” of Figure 2.7. Notice that this area contains the prenex separation logic $\text{pnf-SL}(\exists, *, -*)$, which features the magic wand. For this logic, some work is needed in order to rely on Proposition 2.16. Essentially, in [62] it is shown that the quantifier-free part of a prenex formula of $\text{SL}(\exists, *, -*)$ can be replaced with a formula of $\text{SL}([\exists]_2, *)$, so that the whole formula is translated into $\text{SL}(\exists, *)$. Most importantly, this result shows that the complexity of $\text{SL}(\exists, *, -*)$ really depends on the alternation between the first-order quantification and the separating implication.

2.3 OTHER SEPARATION LOGICS AND BUNCHED LOGICS

Various lines of research led to the definition of multiple separation logics, to the point that “separation logic” is more of an umbrella term to capture a family of logics that use multiplicative connectives in order to verify properties of a memory model. In this section, we briefly recall some of these logics, to then place them in the framework of the logic of bunched implications.

2.3.1 Symbolic-Heaps and (bi)abduction.

As depicted in the previous sections, the prohibiting complexity of $\text{SL}(\exists, *, -*)$ does not make it suitable for automated deduction. A more scalable solution that is well-suited for program verification is given by the frameworks of *symbolic-heap separation logics* [11]. An example of these logics is given by the following syntactical fragment of $\text{SL}(\exists, *, -*)$, denoted here with $\text{SH}(1s)$ and studied in [44]:

$$\begin{aligned}\varphi &:= \Pi \wedge \Sigma \\ \Pi &:= \top \mid x = y \mid x \neq y \mid \Pi \wedge \Pi \quad (\text{pure formulae}) \\ \Sigma &:= \top \mid \text{emp} \mid x \mapsto y \mid 1s(x, y) \mid \Sigma * \Sigma \quad (\text{spatial assertions})\end{aligned}$$

Notice that the formulae of $\text{SH}(1s)$ are conjunctions of one pure formula, i.e. a conjunction of (dis)equalities, and one spatial assertion, i.e. a set of empty, points-to and list-segment predicates connected via the separating conjunction. In particular, the fragment is not closed under negation and does not feature the separating implication. All these restrictions come with a major computational benefit: the model checking, satisfiability, validity and entailment problems of $\text{SH}(1s)$ can be solved in PTIME [44].

The literature regarding symbolic-heaps is particularly vast, ranging from the gentle addition of array predicates and pointer arithmetic [26, 29], to the more expressive extensions with user-defined inductive predicates [28, 92, 65, 93] and Presburger constraints [101]. At their core, all these variants are existential in nature: the negation is only allowed in order to express disequalities between variables, and the separating implication is not considered. This leads to a family of logics that are very well suited for compositional shape analysis, i.e. reason on the shapes of data structures in the heap encountered during the execution of a program, and allows to perform compositional reasoning via (bi)abduction.

Abduction and biabduction are forms of logical inference closely related to entailment. The *abduction problem* is formalised as follows:

abduction: *Input:* A pair of formulae (φ, ψ) .

Question: Is there a formula χ_A such that $\varphi * \chi_A \models \psi$ holds?

Notice that this question has a spatial connotation: its solution χ_A , called *antiframe*, describes a portion of the heap that is missing in order to make the entailment between φ and ψ true. Symbolic-heap separation logics are very relevant for these types of questions, as their formulae closely describe the structure of the heap. Among the possible solutions χ_A of the abduction problem, we generally look at one satisfying the following three properties:

- (*compatible*) $\varphi * \chi_A$ is satisfiable,
- (*weakest*) for every antiframe χ' if $\chi_A \models \chi'$ then $\chi_A \equiv \chi'$,
- (*minimal*) it does not exist an antiframe χ' such that $\chi_A \models \chi' * \neg \text{emp}$.

The separating implication allows us to compute the weakest antiframe very easily: $\chi_A = \varphi \dashv \psi$. This result follows directly from the identity of separation logic shown below:

$$\varphi * \chi \models \psi \text{ if and only if } \chi \models \varphi \dashv \psi.$$

In other words, the operator \dashv is the right-adjoint of $*$, in the same way that \Rightarrow is the right-adjoint of \wedge . We will meet this identity numerous times during the thesis (see e.g. Section 2.3.3).

The *biabduction problem* extends the abduction problem so that also a *frame* χ_F is required:

biabduction: *Input:* A pair of formulae (φ, ψ) .

Question: Are there two formulae χ_A, χ_F such that $\varphi * \chi_A \models \psi * \chi_F$ holds?

Similarly to abduction, among the possible solutions χ_A and χ_F of a biabduction problem, we generally look at one satisfying the following properties:

- (*optimal antiframe*) the antiframe χ_A is compatible, weakest and minimal for the abduction problem $(\varphi, \psi * \top)$,
- (*strongest frame*) for every χ' , if $\varphi * \chi_A \models \psi * \chi'$ and $\chi' \models \chi_F$ then $\chi_F \equiv \chi'$.

The biabduction problem was introduced in [37]. It allows to automatically infer preconditions, so that the proof of a Hoare triple can be further automated. We refer to [115] and [116] for a discussion on the benefits of biabduction in program verification. On this topic, the readers can find there more answers than this thesis could ever provide.

From a technical point of view, symbolic-heaps are quite far from the logic considered in this thesis. In particular, we will mostly deal with separation logics that are closed under Boolean connectives (hence, with negation) and that feature the separating implication. Nevertheless, some of the results presented in this text can be transferred to the symbolic-heap fragment.

2.3.2 Modal Separation Logics.

Continuing with our round-up of the separation logic literature, another interesting line of research (from a theoretical point of view) is given by the framework of modal separation logics introduced by S. Demri and M. Deters in [52]. The original motivation for this work is to emphasise the similarities between various separation logics, modal logics and temporal logics, with a focus on proof techniques that could lead to transfer results between these three areas. Similar directions are followed by J. Courtault, D. Galmiche, and D. Pym in [46] and [45]. Both Chapter 4 and Part III of this thesis fall in this research agenda.

To connect separation logic with modal and temporal logics, a first idea in [52] is to generalise the notion of memory state so that it falls into the realm of Kripke structures used by the latter formalisms. More precisely, the *Modal Separation Logic MSL* is interpreted on Kripke-style finite functions [52, 54]. Let $\text{AP} = \{p, q, \dots\}$ be a countably infinite set of propositional symbols.

Definition 2.17 (Kripke-style finite function). A (*Kripke-style*) *finite function* $(\mathcal{W}, R, \mathcal{V})$ is a triple where \mathcal{W} is a countably infinite set of *worlds*, $R \subseteq \mathcal{W} \times \mathcal{W}$ is a finite weakly functional¹ *accessibility relation*, and $\mathcal{V} : \text{AP} \rightarrow 2^{\mathcal{W}}$ is a *labelling function* assigning to every propositional symbol p the set of worlds satisfying it.

Given a binary relation $R \subseteq \mathcal{W} \times \mathcal{W}$, we write $R(w) \stackrel{\text{def}}{=} \{w' \in \mathcal{W} \mid (w, w') \in R\}$ for the set of *successors* of w . Similarly, $R^{-1}(w)$ is the set of its *predecessors*. If R is weakly functional, then $\text{card}(R(w)) \leq 1$. Since \mathcal{W} and LOC are isomorphic and the accessibility relation R is equivalent to a heap, a Kripke-style finite function can be essentially seen as a memory state where the structure of the store is relaxed so that each variable (a propositional symbol in MSL) corresponds to multiple locations. This analogy with memory states lead to a natural definition of disjointness and union of Kripke-style finite functions.

Definition 2.18 (Disjoint finite functions and their union). Two Kripke-style finite functions $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ are *disjoint* if $R_1 \cap R_2 = \emptyset$. When this holds, their *union* $\mathcal{K}_1 + \mathcal{K}_2$ is the Kripke-style finite function $(\mathcal{W}, R_1 \cup R_2, \mathcal{V})$. We write $\mathcal{K}_1 \subseteq \mathcal{K}_2$ if $R_1 \subseteq R_2$.

Notice that in the definition of disjointness and union the two finite functions \mathcal{K}_1 and \mathcal{K}_2 share the same set of worlds \mathcal{W} and labelling function \mathcal{V} . The formulae φ of MSL belongs to the following grammar (where $p \in \text{AP}$):

$\pi :=$	\top	(true)	$\varphi :=$	π	(atomic formulae)
	p	(propositional symbol)		$\varphi \wedge \varphi$	$\neg \varphi$ (Boolean connectives)
	emp	(empty predicate)		$\varphi * \varphi$	(separating conjunction)
				$\varphi \multimap \varphi$	(separating implication)
				$\Diamond \varphi$	(modality of possibility)
				$\Diamond^{-1} \varphi$	(converse modality of possibility)
				$\langle \neq \rangle \varphi$	(elsewhere modality)

As we can see, with respect to the grammar of $\text{SL}(\exists, *, \neg*)$, the logic MSL drops the points-to and the equality predicates between program variables, and replaces them with three well-known modalities from modal logic. MSL is interpreted on *pointed finite function* (\mathcal{K}, w) where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a Kripke-style finite function and $w \in \mathcal{W}$ is one of its worlds, called the *current world*. With respect to such a model, the satisfaction relation \models for formulae of MSL is given in Figure 2.11 (omitting standard cases for \top and Boolean connectives). The semantics of the formula p simply checks whether the propositional symbol p is satisfied in the current world w . The formula `emp` and the two multiplicative connectives $*$ and $\neg*$ are defined as in separation logic. The operator \Diamond is the standard alethic modality of possibility of modal logic, stating that φ holds in one *successor* of w , i.e. a world w' such that $(w, w') \in R$. Conversely, the modality \Diamond^{-1} asks whether φ holds in one *predecessor* of w . Lastly, the elsewhere modality $\langle \neq \rangle$ asks whether φ holds in a world different from w . Given a formula φ , we write $\langle U \rangle \varphi$ for the formula $\varphi \vee \langle \neq \rangle \varphi$ stating that φ is satisfied by an arbitrary world. The lexeme $\langle U \rangle$ is often called *somewhere* modality, and was introduced by V. Goranko and S. Passy in [80]. As introduced in [52], we call *Modal Logic of Heaps* (MLH) the logic obtained from MSL by removing the propositional symbols from the grammar above.

¹ R is finite and for every $w, w', w'' \in \mathcal{W}$, if $(w, w') \in R$ and $(w, w'') \in R$ then $w' = w''$.

$(\mathcal{K}, w) \models p$	iff $w \in \mathcal{V}(p)$,
$(\mathcal{K}, w) \models \text{emp}$	iff $R = \emptyset$,
$(\mathcal{K}, w) \models \varphi * \psi$	iff there are \mathcal{K}_1 and \mathcal{K}_2 s.t. $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$, $(\mathcal{K}_1, w) \models \varphi$ and $(\mathcal{K}_2, w) \models \psi$,
$(\mathcal{K}, w) \models \varphi -* \psi$	iff for all $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$, if $\mathcal{K}' \perp \mathcal{K}$ and $(\mathcal{K}', w) \models \varphi$ then $(\mathcal{K} + \mathcal{K}', w) \models \psi$,
$(\mathcal{K}, w) \models \Diamond \varphi$	iff there is $w' \in \mathcal{W}$ such that $w' \in R(w)$ and $(\mathcal{K}, w') \models \varphi$,
$(\mathcal{K}, w) \models \Diamond^{-1} \varphi$	iff there is $w' \in \mathcal{W}$ such that $w' \in R^{-1}(w)$ and $(\mathcal{K}, w') \models \varphi$,
$(\mathcal{K}, w) \models \langle \neq \rangle \varphi$	iff there is $w' \in \mathcal{W}$ such that $w' \neq w$ and $(\mathcal{K}, w') \models \varphi$.

Figure 2.11: Satisfaction relation for MSL, with respect to (\mathcal{K}, w) where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

Quite a few separation logics can be shown to be fragments of MSL. The reason for this is that the logic is expressive enough to capture the concept of program variables, hence overcoming the differences between labelling functions and stores. To do so, MSL borrows the concept of *nominals* from hybrid logics, a family of logics that add further expressive power to modal logic [2]. Essentially, a nominal is a propositional symbol that is true exactly in one world (like a program variable). Given a propositional symbol p , we can check whether it encodes a nominal with the formula $\text{nom}(p) \stackrel{\text{def}}{=} \langle U \rangle(p \wedge \neg \langle \neq \rangle p)$. Its formal semantics is recalled below.

Proposition 2.19. Let (\mathcal{K}, w) be a pointed finite function where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. We have, $(\mathcal{K}, w) \models \text{nom}(p)$ if and only if there is exactly one world $w' \in \mathcal{W}$ such that $(\mathcal{K}, w') \models p$.

Using the syntax of hybrid logics, we write $@_p \varphi$ for the formula $\langle U \rangle(p \wedge \varphi)$. Under the hypothesis that p is a nominal, $@_p \varphi$ is satisfied whenever the world corresponding to p satisfies φ .

Nominals allow us to capture the predicates $x = y$ and $x \hookrightarrow y$ of separation logic with the formulae $@_x y$ and $@_x \Diamond y$, respectively, where x and y are seen as nominals. Similarly, we can define all the formulae of $\text{SL}(\exists, *, -*)$ introduced in Section 2.1.1. The alloc predicate $x \hookrightarrow -$ and the alloc-back predicate $- \hookrightarrow x$ correspond to the formulae $@_x \Diamond \top$ and $@_x \Diamond^{-1} \top$, respectively. The reach-plus predicate $x \hookrightarrow^+ y$ is instead captured with the formula below:

$$\top * (x \hookrightarrow - \wedge (- \hookrightarrow x \Rightarrow x = y) \wedge \langle U \rangle \neg(\Diamond^{-1} \top * \Diamond^{-1} \top) \wedge \langle U \rangle ((\neg y \wedge \Diamond^{-1} \top) \Rightarrow \Diamond \top)),$$

where the alloc, alloc-back and equality predicates can be seen now as shortcuts in MSL. One can clearly see the correspondence between this formula and the definition of $x \hookrightarrow^+ y$ given in Section 2.1.1: the subformula $\langle U \rangle \neg(\Diamond^{-1} \top * \Diamond^{-1} \top)$ corresponds to $\forall x \neg(- \hookrightarrow x * - \hookrightarrow x)$, whereas the subformula $\langle U \rangle ((\neg y \wedge \Diamond^{-1} \top) \Rightarrow \Diamond \top)$ corresponds to $\forall x (x \neq y \wedge - \hookrightarrow x \Rightarrow x \hookrightarrow -)$. Thus, MSL is an extension of the quantifier-free separation logic $\text{SL}(*, -*, \hookrightarrow^+)$.

Proposition 2.20 (From [54]). $\text{SL}(*, -*, \hookrightarrow^+)$ is a fragment of MSL.

The connections between MSL and separation logic are exploited in [52] to transfer complexity results from MSL (more specifically, MLH) to $\text{SL}(\exists, *, -*)$. In particular, the authors notice how the fragment of MLH without separating implication admits a TOWER-complete satisfiability problem. Subsequently, it is sufficient to rely on the standard translation of modalities into two-variable logics [15] to conclude that the satisfiability problem of the separation logic $\text{SL}([\exists]_2, *)$ is TOWER-complete already on closed formulae. Figure 2.12 summarises the

Grammar:	Satisfiability:
$\varphi := \top \mid \text{emp} \mid \varphi \wedge \varphi \mid \neg \varphi \mid \varphi * \varphi \mid \Diamond \varphi \mid \Diamond^{-1} \varphi \mid \langle \neq \rangle \varphi$	TOWER-complete [52]
$\varphi := \top \mid p \mid \text{emp} \mid \varphi \wedge \varphi \mid \neg \varphi \mid \varphi * \varphi \mid \Diamond \varphi \mid \langle \neq \rangle \varphi$	TOWER-complete [54]
$\varphi := \top \mid p \mid \text{emp} \mid \varphi \wedge \varphi \mid \neg \varphi \mid \varphi * \varphi \mid \Diamond \varphi$	NP-complete [54]
$\varphi := \top \mid p \mid \text{emp} \mid \varphi \wedge \varphi \mid \neg \varphi \mid \varphi * \varphi \mid \langle \neq \rangle \varphi$	NP-complete [54]

Figure 2.12: The complexity of MSL.

known complexities on the satisfiability problem of fragments of MSL, excluding those that are proven using results that are presented in this thesis.

2.3.3 Boolean BI.

Even though the development of separation logic has been pragmatic in nature, the reader should be confident that the logic is rooted in the solid mathematical theory of *Bunched logics* [113]. More precisely, separation logic instantiates the framework of Bunched Implications under the classical interpretation of Boolean connectives, also known as Boolean Bunched Implications (BBI). In this short section, we recall the syntax, semantics and axiomatisation of BBI, as well as connecting this formalism to separation logic. We follow the presentation in [75], and refer the reader to S. Docherty PhD thesis for a complete description of Bunched Logics [60].

The formulae of BBI belongs to the following grammar, where p is a propositional symbol taken from a countably infinite set AP (as in MSL):

$$\begin{array}{lll} \pi := \top & \quad \varphi := \pi & \quad (\text{atomic formulae}) \\ | \quad p & \quad | \quad \varphi \Rightarrow \varphi \mid \neg \varphi & \quad (\text{Boolean connectives}) \\ | \quad \text{emp} & \quad | \quad \varphi * \varphi & \quad (\text{separating conjunction}) \\ & \quad | \quad \varphi -* \varphi & \quad (\text{separating implication}) \end{array}$$

The purpose of BBI is to provide a framework to reason on resource composition. To achieve this objective elegantly, resources are abstracted with a monoidal algebraic structure, which leads to a natural notion of composition via the binary operation of the monoid. Let \mathcal{M} be a set and $\circ : \mathcal{M} \times \mathcal{M} \rightarrow 2^{\mathcal{M}}$ be a binary operation. Given two subsets S and T of \mathcal{M} we extend \circ and write $S \circ T$ for $\{a \circ b \mid a \in X, b \in Y\}$. Given $m \in \mathcal{M}$, we write $m \circ T$ and $S \circ m$ for $\{m\} \circ T$ and $S \circ \{m\}$, respectively. We introduce the notion of non-deterministic monoid.

Definition 2.21 (Non-deterministic monoid, [75]). A non-deterministic monoid is a triple $(\mathcal{M}, \circ, \epsilon)$ where $\epsilon \in \mathcal{M}$, $\circ : \mathcal{M} \times \mathcal{M} \rightarrow 2^{\mathcal{M}}$ and

- (identity) $\epsilon \circ m = \{m\}$, for every $m \in \mathcal{M}$,
- (associativity) $a \circ (b \circ c) = (a \circ b) \circ c$, for every $a, b, c \in \mathcal{M}$,
- (commutativity) $a \circ b = b \circ a$, for every $a, b \in \mathcal{M}$.

The formulae of BBI are interpreted over the elements of a non-deterministic monoid $(\mathcal{M}, \circ, \epsilon)$, together with an evaluation $\llbracket . \rrbracket : \text{AP} \rightarrow 2^{\mathcal{M}}$ for propositional symbols. The satisfaction relation \models , implicitly parametrised on $(\mathcal{M}, \circ, \epsilon)$ and $\llbracket . \rrbracket$, is given in Figure 2.13. We stress that the

$m \models \top$	always,
$m \models p$	iff $m \in \llbracket p \rrbracket$,
$m \models \text{emp}$	iff $m = \epsilon$,
$m \models \neg\varphi$	iff $m \not\models \varphi$,
$m \models \varphi \Rightarrow \psi$	iff (if $m \models \varphi$ then $m \models \psi$),
$m \models \varphi * \psi$	iff there are $a, b \in \mathcal{M}$ such that $m \in (a \circ b)$, $a \models \varphi$ and $b \models \psi$,
$m \models \varphi -* \psi$	iff for every $a, b \in \mathcal{M}$, if $b \in (a \circ m)$ and $a \models \varphi$ then $b \models \psi$.

Figure 2.13: Satisfaction relation for BBI.

implication $\varphi \Rightarrow \psi$ of BBI has the standard semantics from classical logic, and thus should not be confused with the intuitionistic implication of the (original) logic of Bunched Implications [113]. Moreover, we notice that the conjunction $\varphi \wedge \psi$ (which is a primitive connective in $\text{SL}(\exists, *, -*)$) is defined in BBI as $\varphi \wedge \psi \stackrel{\text{def}}{=} \neg(\varphi \Rightarrow \neg\psi)$.

Several logics introduced in recent years can be seen as an instantiation of BBI, as for instance ambient logics [103, 32], team logics [135, 136] and, of course, separation logics. We will meet again both ambient logics and team logics in Chapters 7 and 8 of the thesis. For separation logics, it is quite easy to see that the $\text{SL}(*, -*)$, i.e. the quantifier-free fragment of $\text{SL}(\exists, *, -*)$, instantiate BBI for a specific monoid and evaluation of propositional symbols. As a monoid, we consider the set of all heaps $[\text{LOC} \rightarrow_{\text{fin}} \text{LOC}]$, and we let ϵ be the empty heap and \circ be the binary operation

$$h_1 \circ h_2 = \begin{cases} \{h_1 + h_2\} & \text{if } \text{dom}(h_1) \cap \text{dom}(h_2) = \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$$

Proposition 2.22. $([\text{LOC} \rightarrow_{\text{fin}} \text{LOC}], \circ, \epsilon)$ is a non-deterministic monoid.

For the evaluation of propositional symbols, we first consider a bijection f from atomic formulae of the form $x = y$ and $x \hookrightarrow y$ ($x, y \in \text{VAR}$) to the set of propositional symbols AP. Since both VAR and AP are countably infinite, f exists. Then, given a store s , we define the evaluation $\llbracket \cdot \rrbracket_s$:

$$\llbracket p \rrbracket_s = \begin{cases} [\text{LOC} \rightarrow_{\text{fin}} \text{LOC}] & \text{if there are } x, y \in \text{VAR} \text{ s.t. } f^{-1}(p) = x = y \text{ and } s(x) = s(y), \\ \emptyset & \text{if there are } x, y \in \text{VAR} \text{ s.t. } f^{-1}(p) = x = y \text{ and } s(x) \neq s(y), \\ \{h \mid h(\ell) = \ell'\} & \text{if there are } x, y \in \text{VAR} \text{ s.t. } f^{-1}(p) = x \hookrightarrow y, s(x) = \ell \text{ and } s(y) = \ell'. \end{cases}$$

Lastly, given a quantifier-free formula φ of $\text{SL}(\exists, *, -*)$, we write φ^f for the formula in BBI obtained from φ by replacing every atomic formula π of the form $x = y$ of $x \hookrightarrow y$ by $f(\pi)$. The following proposition, whose proof (by structural induction on φ) is left to the reader, connects the semantics of $\text{SL}(\exists, *, -*)$ with the one of BBI.

Proposition 2.23. Let φ be in $\text{SL}(*, -*)$ and (s, h) be a memory state. $(s, h) \models \varphi$ in $\text{SL}(*, -*)$ iff $h \models \varphi^f$ in BBI, w.r.t. the non-deterministic monoid $([\text{LOC} \rightarrow_{\text{fin}} \text{LOC}], \circ, \epsilon)$ and the evaluation $\llbracket \cdot \rrbracket_s$.

The correspondence between separation logic and BBI depicted in Proposition 2.23 allows us to grasp some of the properties of separation logic directly by looking at BBI. In our case, we are

Propositional Calculus:

$$\begin{aligned} (\text{L}_1) \quad & (\neg\varphi \Rightarrow \varphi) \Rightarrow \varphi \\ (\text{L}_2) \quad & \varphi \Rightarrow (\neg\varphi \Rightarrow \psi) \\ (\text{L}_3) \quad & (\varphi \Rightarrow \psi) \Rightarrow ((\psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \chi)) \end{aligned}$$

$$(\text{MP}) \quad \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$$

Axioms of the non-deterministic monoid:

$$\begin{aligned} (\text{ID}_L) \quad & \varphi \Rightarrow \text{emp} * \varphi \\ (\text{ID}_R) \quad & \text{emp} * \varphi \Rightarrow \varphi \end{aligned}$$

$$\begin{aligned} (\text{ASSOC}) \quad & \varphi * (\psi * \chi) \Leftrightarrow (\varphi * \psi) * \chi \\ (\text{COM}) \quad & \varphi * \psi \Rightarrow \psi * \varphi \end{aligned}$$

Rules of inference for the multiplicative connectives:

$$(*) \quad \frac{\varphi \Rightarrow \chi}{\varphi * \psi \Rightarrow \chi * \psi} \quad (*_1) \quad \frac{\varphi \Rightarrow (\psi \dashv \chi)}{\varphi * \psi \Rightarrow \chi} \quad (*_2) \quad \frac{\varphi * \psi \Rightarrow \chi}{\varphi \Rightarrow (\psi \dashv \chi)}$$

Figure 2.14: Hilbert-style axiomatisation of BBI [75].

particularly interesting in Hilbert-style proof systems for BBI, which, following Proposition 2.23, only contains axioms and rules that are also admissible in separation logic. We have yet to define what is an Hilbert-style proof system (this is done formally in Section 6.1). However, for the time being we invite the reader to think about it as a set of valid formulae (of BBI), together with rules of the form

$$\frac{\varphi_1, \dots, \varphi_n}{\psi}$$

which should be read as “if the formulae $\varphi_1, \dots, \varphi_n$ are all valid, then so is ψ ”. Let us look at the Hilbert-style proof system of BBI provided in [75] and recalled in Figure 2.14. The proof system includes an axiomatisation of classical propositional logic, due to J. Lukasiewicz [18] and made of three axioms (L_1) – (L_3) together with the rule of Modus ponens (MP) . More interesting, the second part of the proof system in Figure 2.14 axiomatises the notion of non-deterministic monoids given in Definition 2.21. The first two axioms (ID_L) and (ID_R) state that, in BBI, emp behaves as the identity element of the operator $*$, which in turn is an associative and commutative binary connective (axioms (ASSOC) and (COM)). Thanks to the correspondence between BBI and separation logic, these four axioms are also valid in separation logic, and capture in a neat syntactical way the essence of Proposition 2.22. The last part of Figure 2.14 is made of three rules of inference. The first one, denoted by $(*)$, is sometimes called “frame rule” by analogy with the rule of the same name in program logic. Essentially, it entails that logical equivalence is a congruence for $*$. The second and third rules state that the separating implication is the right-adjoint of the $*$, exactly as we found out in Section 2.3.1.

The axiomatisation of BBI given in Figure 2.14 provides the foundation to derive Hilbert-style proof systems for the quantifier-free fragment of $\text{SL}(\exists, *, \dashv)$ and similar logics, as we will see in Chapters 6 and 7 of the thesis.

Part I

Reachability Queries in Separation Logic

Robustness Properties of Logical Assertions

In Chapters 3, 4 and 5, we propose an in-depth study of separation logics featuring reachability predicates such as $\text{ls}(x, y)$ and $x \hookrightarrow^+ y$ (see Section 2.1.1). In program analysis, these predicates provide the foundation for verifying programs manipulating lists, but in the context of separation logic their use is often limited to the symbolic-heap fragment (Section 2.3.1). This restriction, while being beneficial on a computational level, severely limits the range of properties we are able to check. We pay particular attention to two properties that are not expressible in the symbolic heap fragment: the acyclicity property and the garbage freedom property. As done in [93], we refer to these two properties as the *robustness properties* of memory states. We say that a memory state (s, h) is *acyclic* whenever no location can *reach* itself by traversing the heap a positive amount of times. As explained in [76], being able to check whether the acyclicity property holds is useful in analysing the termination of a program. Indeed, starting from an acyclic memory state, any loop that traverses the heap is bound to terminate. Given a finite set $X \subseteq_{\text{fin}} \text{VAR}$ of program variables, we say that (s, h) is *X-garbage free* whenever, all memory cells of $\text{dom}(h)$ are reached by a location corresponding to a variable in X . In programming languages that do not feature garbage collection (e.g. C), this property can be used to prove that the program does not leak memory by generating unreachable portions of the heap. As described in [93], both acyclicity and garbage freedom come with homonymous decision problems that, given a formula φ , ask whether all memory states satisfying φ are acyclic and $\text{fv}(\varphi)$ -garbage free, respectively. The formal definition of these problems is given in Figure 2.15.

Motivations.

At their core, both robustness properties rely on reachability predicates. Indeed, in $\text{SL}(\exists, *, \neg)$, the class of acyclic memory states is characterised by the formula $\forall x \neg(x \hookrightarrow^+ x)$ whereas the set of X -garbage free memory states is characterised by the formula $\forall x (x \hookrightarrow_+ \Rightarrow \bigvee_{y \in X} y \hookrightarrow^* x)$. Unfortunately, the undecidability result given by Theorem 2.13 prevents us from using $\text{SL}(\exists, *, \neg)$ for automatic program analysis. This leads us to journey through various fragments of this logic, with the goal of designing a separation logic that is expressive enough capture the notions of acyclicity and garbage freedom, while still being decidable. More precisely, we aim for a logic that extends $\text{SL}(*, \neg)$, i.e. the quantifier-free fragment of $\text{SL}(\exists, *, \neg)$, and admits a satisfiability problem that can be solved in PSPACE, exactly as $\text{SL}(*, \neg)$. This goal, which we eventually reach in Chapter 5, reveals to be quite ambitious, as in both Chapters 3 and 4 we show how very small extensions of $\text{SL}(*, \neg)$ lead to negative results in terms of computational complexity.

-
- acyclicity:** *Input:* A formula φ .
Question: Is every memory state (s, h) satisfying φ acyclic
(i.e. for every $\ell \in \text{LOC}$ and $\delta \geq 1$, $h^\delta(\ell) \neq \ell$)?
- garbage freedom:** *Input:* A formula φ .
Question: Is every memory state (s, h) satisfying φ $\text{fv}(\varphi)$ -garbage free
(i.e. for every $\ell \in \text{dom}(h)$ there is $\delta \in \mathbb{N}$ and $x \in \text{fv}(\varphi)$
such that $h^\delta(s(x)) = \ell$)?

Figure 2.15: The decision problems for the properties of acyclicity and garbage freedom.

Contribution of Chapter 3.

Our journey starts by simply adding reachability predicates to $\text{SL}(*, -*)$. We consider the standard reachability predicates $1s$, \hookrightarrow^+ and \hookrightarrow^* already introduced in Section 2.1.1, as well as the bounded reachability predicates $x \hookrightarrow^\delta y$, where $\delta \geq 1$, that are satisfied by a memory state (s, x) whenever the minimal path in h going from $s(x)$ to $s(y)$ has length δ . Very surprisingly, we show that as soon as both bounded reachability predicates $x \hookrightarrow^2 y$ and $x \hookrightarrow^3 y$ are added to $\text{SL}(*, -*)$, the satisfiability problem jumps from PSPACE to non RE. This result extends to several separation logics featuring reachability predicates, among which:

- $\text{SL}([\exists]_2, *, -*)$, i.e. the two quantified variable restriction of $\text{SL}(\exists, *, -*)$,
- $\text{SL}(*, -*)$ augmented with one predicate among $1s$, \hookrightarrow^+ or \hookrightarrow^* ,
- $\text{SL}(*, -*, \hookrightarrow^2, \hookrightarrow^3)$ and all the logics above, restricted to 4 program variables.

The main cause of the computational blow-up is traced back to the interactions between the reachability predicates and the separating implication $-*$, which allows us to encode first-order quantifications by means of heap updates.

Contribution of Chapter 4.

In view of the results in Chapter 3, we remove for the time being the separating implication and focus on the separation logic $\text{SL}([\exists]_1, *, x \hookrightarrow_-, \hookrightarrow^+)$ featuring one quantified variable name, the separating conjunction $*$, the predicate alloc $x \hookrightarrow_-$ and the reachability predicate $x \hookrightarrow^+ y$. This logic is a fragment of $\text{SL}(\exists, *)$, which admits a TOWER-complete satisfiability problem [22]. Unfortunately, we show that $\text{SL}([\exists]_1, *, x \hookrightarrow_-, \hookrightarrow^+)$ is already TOWER-hard. Actually, this result is proved in a more general settings, as we show a set of features centred around reachability and submodel reasoning which causes logics interpreted on trees to be TOWER-hard. These features are formally described through a new modal logic which we call ALT (short for Auxiliary Logic on Trees). Apart from $\text{SL}([\exists]_1, *, x \hookrightarrow_-, \hookrightarrow^+)$, ALT is captured by several logics that were independently found to be TOWER-hard, as quantified computation tree logic [99] interpreted on trees (QCTL^t), modal separation logics [54] and modal logic of heaps [52]. New fragments of these logics are discovered to be TOWER-complete:

- $\text{SL}(*, -*, 1s)$ where $-*$ only occurs in the from $\text{size} = 1 -* \varphi$,
- $\text{QCTL}^t(\text{EU}^0)$, i.e. the fragment of QCTL^t only featuring the exists-until temporal operator, which cannot be nested,

- $\text{QCTL}^t(\text{EF}^1)$, i.e. the fragment of QCTL^t only featuring the exists-finally temporal operator, which can be nested only once,
- the common fragment of modal separation logic and modal logic of heaps, featuring the separating conjunction and the modalities \Diamond and $\langle U \rangle$.

Contribution of Chapter 5.

At last, the negative results of Chapter 3 and Chapter 4 guide us to the definition of a separation logic that satisfies all the conditions we have imposed: (I) it extends $\text{SL}(*, \neg*)$ with reachability predicates, (II) it can express both robustness properties, and (III) its satisfiability problem is PSPACE-complete, exactly as for $\text{SL}(*, \neg*)$. This logic, which is denoted by $\text{SL}([\exists]_1, *, [\neg*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}})$, is a syntactical fragment of $\text{SL}(\exists, *, \neg*)$ specifically crafted to avoid the interactions between reachability and the spatial connectives $*$ and $\neg*$ that, during Chapter 3 and Chapter 4, were discovered causing computational blow ups. To show the PSPACE upper bound of the satisfiability problem for $\text{SL}([\exists]_1, *, [\neg*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}})$, we extend the *core formulae technique*, a proof method that is often used in separation logic in order to obtain complexity results (see e.g. [104, 55, 56, 62]).

3

Extensionality and Reachability Leads to Non-enumerability

Contents

3.1	Encoding Assignments as Memory Cells	51
3.1.1	Generalised memory states.	52
3.1.2	The encoded-by relation \triangleright_Y^X	54
3.2	Simulating the First-order Quantification	55
3.2.1	Translating $SL(\exists, \neg)$ into $SL(n(x), *, \neg)$	56
3.2.2	$SL(n(x), *, \neg)$ is not recursively enumerable.	64
3.3	Reachability Predicates can Quantify	65
3.3.1	Bounded reachability.	65
3.3.2	Using $SL(n(x), *, \neg)$ to prove that $SL(*, \neg, \leftrightarrow^2, \leftrightarrow^3)$ is not RE.	68
3.3.3	Other separation logics with non RE decision problems.	70
3.3.4	Modal separation logic is non RE.	71

In this chapter

We look at the complexity of satisfiability and validity problems for extensions of the quantifier-free separation logic $\text{SL}(*, \neg*)$ featuring reachability predicates. We show the non-recursive enumerability of the satisfiability problem for $\text{SL}(*, \neg*)$ enriched with the bounded reachability predicates $x \rightarrow^2 y$ and $x \rightarrow^3 y$, where the predicate $x \rightarrow^\delta y$ is satisfied by a memory state (s, h) whenever the minimal path in h going from $s(x)$ to $s(y)$ has length δ . As bounded reachability predicates are expressible as soon as one predicate among 1s , \rightarrow^+ or \rightarrow^* are added to $\text{SL}(*, \neg*)$, several other separation logics are found to be non-recursively enumerable.

In order to show these results, our investigation starts by noticing that, in a memory state (s, h) , the role of the store s can be internalised in a heap, essentially leading to a (generalised) heap of the form $s + h : (\text{VAR} + \text{LOC}) \rightarrow \text{LOC}$. The multiplicative connectives $*$ and $\neg*$ of separation logic can be used to update the region of $s + h$ encoding the store s in a way that simulates the variable assignments done by the first-order quantification. This detour naturally leads us to a quantifier-free separation logic, denoted by $\text{SL}(n(x), *, \neg*)$, where these heap updates can be effectively checked and where the first-order quantification can be broadly simulated.

We show that the satisfiability and validity problems for $\text{SL}(n(x), *, \neg*)$ are non-recursively enumerable. Thanks to the simulation of first-order quantification via heap updates, this result can be shown by reduction from the satisfiability and validity problems of the first-order separation logic $\text{SL}(\exists, \neg*)$ shown non-recursively enumerable in [22]. Afterwards, we return to our original goal and design a semantically faithful translation from $\text{SL}(n(x), *, \neg*)$ to $\text{SL}(*, \neg*)$ enriched with the bounded reachability predicates $x \rightarrow^2 y$ and $x \rightarrow^3 y$.

Here is a roadmap of the chapter.

Section 3.1. We formalise the idea of encoding the store as part of the heap, and to simulate first-order quantification as heap updates. This is done by introducing the notion of generalised memory state (Definition 3.1) and encoding between generalised memory states (Definition 3.4).

Section 3.2. We introduce the separation logic $\text{SL}(n(x), *, \neg*)$ which is interpreted on generalised memory states. Afterwards, we move to the main technical contribution of the chapter, and design a reduction from the satisfiability (resp. validity) problem for $\text{SL}(\exists, \neg*)$ to the satisfiability (resp. validity) problem for $\text{SL}(n(x), *, \neg*)$. The following result is derived.

Theorem 3.5. The satisfiability and validity problems of $\text{SL}(n(x), *, \neg*)$ are not RE.

Section 3.3 We show that $\text{SL}(n(x), *, \neg*)$ can be translated into $\text{SL}(*, \neg*)$ enriched with $x \rightarrow^2 y$ and $x \rightarrow^3 y$ and, as a by-product, to several other separation logics, such as $\text{SL}(*, \neg*)$ augmented with either 1s , \rightarrow^+ or \rightarrow^* . We conclude that the satisfiability problem of all these logics is non RE (Corollary 3.19). Lastly, we transfer these results to the realm of modal separation logics, and show the following theorem by translation from $\text{SL}(*, \neg*)$ enriched with $x \rightarrow^2 y$ and $x \rightarrow^3 y$.

Theorem 3.20. MSL without \diamond^{-1} , $\langle \neq \rangle$ and emp has non RE satisfiability and validity problems.

3.1 ENCODING ASSIGNMENTS AS MEMORY CELLS

In Section 2.1.1, we saw how the first-order quantification of $\text{SL}(\exists, *, \neg*)$ can be used in order to express the standard reach-plus predicate \hookrightarrow^+ . We also noticed that there are instances where the opposite direction also holds: the first-order quantification can be avoided by using this reachability predicate together with the multiplicative connectives $*$ and $\neg*$. In particular, we showed that this is the case for the formula $_ \hookrightarrow \mathbf{x}$, analysed in Proposition 2.7. We ask ourselves if this can be done systematically. That is, we want to study whether first-order quantification can be broadly simulated by reachability predicates and multiplicative connectives. As this chapter answers this question positively, we conclude that enriching the quantifier-free separation logic $\text{SL}(*, \neg*)$ with \hookrightarrow^+ , makes the associated satisfiability and validity problems jump from PSPACE-complete to non RE (by Theorem 2.13).

In what follows, we give a rough explanation of the idea behind this result. For the moment we do not formally fix the main separation logics considered in the chapter, but simply think in terms of $\text{SL}(\exists, *, \neg*)$. Recall that in this logic, a first-order existentially quantified formula $\exists \mathbf{z} \varphi$ essentially (re)assign a location to \mathbf{z} , and then proceed with the evaluation of φ :

$$(s, h) \models \exists \mathbf{z} \varphi \text{ iff } \text{there is } \ell \in \text{LOC} \text{ such that } (s[\mathbf{z} \leftarrow \ell], h) \models \varphi.$$

Regardless of what we can express with the \hookrightarrow^+ predicate, if we want to mimic $\exists \mathbf{z} \varphi$ in a systematic way we need to find a way to simulate the effects that the first-order quantification has on the store by means of updates done to the heap (hence, using the multiplicative connectives). Let (s, h) be a memory state and let φ be in $\text{SL}(\exists, *, \neg*)$. As depicted by Proposition 2.10, in order to decide whether $(s, h) \models \varphi$ holds it is sufficient to consider the part of the store that corresponds to the variables occurring in φ , say \mathbf{X} . Let $s|_{\mathbf{X}}$ be the domain-restriction of the store to the variables in \mathbf{X} , i.e. $s|_{\mathbf{X}} \stackrel{\text{def}}{=} \{(x, \ell) \in s \mid x \in \mathbf{X}\}$. Roughly speaking, a simple but effective idea to simulate the role of the store s in the satisfaction of φ is to view $s|_{\mathbf{X}}$ as a heap. This means seeing the variables in \mathbf{X} as locations, and consider the memory state-like structure $(\text{id}_{\mathbf{X}}, s|_{\mathbf{X}} + h)$, where $\text{id}_{\mathbf{X}}$ is the identity map on \mathbf{X} . By doing this, and assuming $\mathbf{z} \in \mathbf{X}$, we can interpret the first-order quantified formula $\exists \mathbf{z} \varphi$ not as a reassignment done on the store, but as a local update of the “memory cell” \mathbf{z} , done on the heap. Approximately, its semantics could look as follows:

$$(\text{id}_{\mathbf{X}}, s|_{\mathbf{X}} + h) \models \exists \mathbf{z} \varphi \text{ iff } \text{there is } \ell \in \text{LOC} \text{ such that } (\text{id}_{\mathbf{X}}, (s|_{\mathbf{X}} + h)[\mathbf{z} \leftarrow \ell]) \models \varphi,$$

where $(s|_{\mathbf{X}} + h)[\mathbf{z} \leftarrow \ell]$ stands for $((s|_{\mathbf{X}} + h) \setminus \{(\mathbf{z}, s(\mathbf{z}))\}) + \{(\mathbf{z}, \ell)\}$, which can be seen as the heap obtained by first removing the memory cell $(\mathbf{z}, s(\mathbf{z}))$ and then adding (\mathbf{z}, ℓ) . Using the formulae $\mathbf{z} \hookrightarrow _$ and $\text{size}=1$ introduced in Section 2.1.1, this interpretation of $\exists \mathbf{z} \varphi$ can be characterised with the formula $(\mathbf{z} \hookrightarrow _ \wedge \text{size}=1) * (\mathbf{z} \hookrightarrow _ \wedge \text{size}=1 \multimap \varphi)$. Indeed, suppose that $(\text{id}_{\mathbf{X}}, s|_{\mathbf{X}} + h)$ satisfies this formula. The left conjunct $\mathbf{z} \hookrightarrow _ \wedge \text{size}=1$ separates $(\mathbf{z}, s(\mathbf{z}))$ from the rest of the heap, say $h' = (s|_{\mathbf{X}} + h) \setminus \{(\mathbf{z}, s(\mathbf{z}))\}$. Then, the right conjunct $\mathbf{z} \hookrightarrow _ \wedge \text{size}=1 \multimap \varphi$ realises the semantics given above by stating that there is a heap $\{(\mathbf{z}, \ell)\}$, for some $\ell \in \text{LOC}$, such that $(\text{id}_{\mathbf{X}}, h' + \{(\mathbf{z}, \ell)\}) \models \varphi$.

However, this formula and the semantics we introduced are not entirely correct. Indeed, the subtraction $\mathbf{z} \hookrightarrow _ \wedge \text{size}=1 \multimap \varphi$ can add the arrow (\mathbf{z}, \mathbf{z}) to the heap, leading to a structure $(s|_{\mathbf{X}} + h)[\mathbf{z} \leftarrow \mathbf{z}]$ that cannot be obtained by any well-formed (classical) memory state. In order to correct this, the subheap $s|_{\mathbf{X}}$ that is reserved to simulate the store must remain unreachable from the rest of the heap h . This can be done by forcing the alloc-back predicate $_ \hookrightarrow \mathbf{z}$ to not hold after an update (the details are given in Section 3.2). Figures 3.1 to 3.3 sketch this

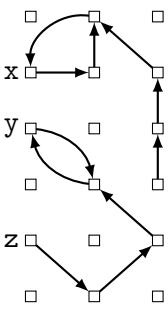


Figure 3.1: A memory state.

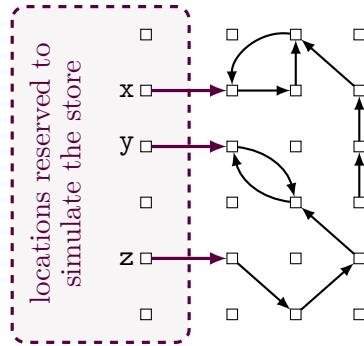


Figure 3.2: Injecting a store.

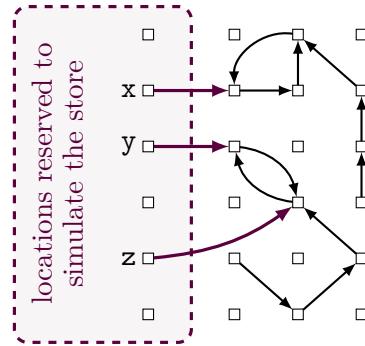


Figure 3.3: Reassignment.

idea. Given the memory state presented in Figure 3.1, we extend the heap with the portion of the store we are interested in (i.e. $s|_x$ in the previous description), leading to the structure in Figure 3.2. The store of this new structure is a simple identity map. Then, a quantification on z is achieved by modifying the location pointed by z , as shown in Figure 3.3.

Quite a few technicalities are omitted in this otherwise simple idea. First, because of this model transformation both the predicates $x = y$ and $x \hookrightarrow y$ must be revised accordingly. For example, $z \hookrightarrow y$ must be updated from “the location assigned to z points to the location assigned to y ” to the rather cumbersome “ z points to a location that points to the location that is pointed by y ” (a pattern that holds in Figure 3.3). In the next section, this property is formally captured by the *next-points-to* predicate $n(z) \hookrightarrow n(y)$. Similarly, the *next-equality* predicate $n(x) = n(y)$ capture the notion of “ x and y points to the same location”, which is a natural update to the formula $x = y$ stating that “the same location is assigned to both x and y ”. In Section 3.3, we show that both $n(x) = n(y)$ and $n(x) \hookrightarrow n(y)$ can be expressed in terms of reachability predicates. Another problem arises when dealing with the multiplicative connectives. For example, $(s, h) \models \varphi \multimap \psi$ is evaluated by considering heaps h' that are disjoint from h and fulfill $(s, h') \models \varphi$. As in our encoding the store is a part of the heap h , for the moment it is not clear how we can simultaneously encode it inside h' and keep the two heaps disjoint. The solution we consider relies on introducing one variable \bar{x} for each variable x appearing in φ . The store encoded in h' considers these new variables, so that $h \perp h'$ still holds. All these technical developments are given in Section 3.2.

3.1.1 Generalised memory states.

Let (s, h) be a memory state and let X be a finite set of variables. Albeit simple, the idea of seeing $s|_X$ as a heap must be fully formalised. For instance, the structure $s|_X \cup h$ is not technically speaking a heap, but a function in $[(\text{VAR} \cup \text{LOC}) \multimap_{\text{fin}} \text{LOC}]$. To solve this issue, in this chapter we consider a slight alternative semantics for $\text{SL}(\exists, *, \multimap)$ and its fragments, which does not modify the notion of satisfiability/validity and such that the set of formulae and the definition of the satisfaction relation \models given in Section 2.1 remain unchanged.

Definition 3.1 (Generalised memory state). A *generalised memory state* is a triple (G, s, h) where G is a countably infinite set, $s : \text{VAR} \rightarrow G$ and $h : G \multimap_{\text{fin}} G$.

So far, memory states are pairs of the form (s, h) with $s : \text{VAR} \rightarrow \text{LOC}$ and $h : \text{LOC} \multimap_{\text{fin}} \text{LOC}$ for a fixed countably infinite set of locations LOC . A generalised memory state is instead parametric

on the set of locations. One can see this variation as simply a way of exhibiting the set of locations directly as an element of the memory state. Indeed, a (standard) memory state (s, h) is equivalent to the generalised memory state (LOC, s, h) . Given a bijection $f : G_1 \rightarrow G_2$ and a heap $h_1 : G_1 \rightarrow_{\text{fin}} G_1$, we write $f(h_1)$ for the heap $h_2 : G_2 \rightarrow_{\text{fin}} G_2$ defined as $h_2 \stackrel{\text{def}}{=} \{f(\ell_1) \mapsto f(\ell_2) \mid h_1(\ell_1) = \ell_2\}$. The satisfaction relation \models of Figure 2.3 can be updated to generalised memory states in a natural way. As one can expect, considering generalised memory states instead of standard ones does not change the notion of satisfiability and validity, as we show next. First, let us revisit the notion of X -isomorphic memory states (Definition 2.9).

Definition 3.2 (g - X -heap-isomorphism). Two generalised memory states $(G_1, s_1, h_1), (G_2, s_2, h_2)$ are said to be *g - X -heap-isomorphic* ($X \subseteq \text{VAR}$), written $(G_1, s_1, h_1) \simeq_X^g (G_2, s_2, h_2)$, if there is a bijection $f : G_1 \rightarrow G_2$ such that (1) $f(h_1) = h_2$ and (2) for every $x \in X$, $f(s_1(x)) = s_2(x)$.

As in the case of X -heap-isomorphism, \simeq_X^g is an equivalence relation. In particular, note that if f is a g - X -heap-isomorphism from (G_1, s_1, h_1) to (G_2, s_2, h_2) (i.e. a bijection as in Definition 3.2), then f^{-1} is a g - X -heap-isomorphism from (G_2, s_2, h_2) to (G_1, s_1, h_1) . We now extend Proposition 2.10 to generalised memory states, thus proving that no formula of $\text{SL}(\exists, *, \neg)$ written with free-variables in X can distinguish between g - X -heap-isomorphic memory states.

Lemma 3.3. Let $X \subseteq \text{VAR}$. Consider two generalised memory states $(G_1, s_1, h_1) \simeq_X^g (G_2, s_2, h_2)$. For every φ in $\text{SL}(\exists, *, \neg)$ with $\text{fv}(\varphi) \subseteq X$, $(G_1, s_1, h_1) \models \varphi$ iff $(G_2, s_2, h_2) \models \varphi$.

Proof. The proof is by induction on the tree structure of φ (with the natural induction hypothesis stating that the property holds for strict subformulae of φ). Let X be a set of variables that includes the free variables from φ . Let $f : G_1 \rightarrow G_2$ be a g - X -heap-isomorphism from (G_1, s_1, h_1) to (G_2, s_2, h_2) , as defined in Definition 3.2. Since \simeq_X^g is a symmetric relation (as it is an equivalence relation), it is sufficient to prove one direction of the lemma (the other direction holds by considering f^{-1} instead of f). Recall that f^{-1} is a g - X -heap-isomorphism from (G_2, s_2, h_2) to (G_1, s_1, h_1) . The base cases for `emp`, $x = y$ and $x \hookrightarrow y$ pose no difficulty. Thus, we only show the case for $x \hookrightarrow y$, and omit the other two.

base case: $x \hookrightarrow y$. Suppose $(G_1, s_1, h_1) \models x \hookrightarrow y$, and therefore $h_1(s_1(x)) = s_1(y)$. Thanks to the properties g - X -heap-isomorphism, the following sequence of equalities is satisfied:

$$f^{-1}(h_2(s_2(x))) = h_1(f^{-1}(s_2(x))) = h_1(s_1(x)) = s_1(y) = f^{-1}(s_2(y)).$$

As f is a bijection, we derive $h_2(s_2(x)) = s_2(y)$ and thus $(G_2, s_2, h_2) \models x \hookrightarrow y$ holds.

Concerning the inductive cases, we omit the obvious cases when the outermost connective is a Boolean connective, leaving us with formulae of the form $\psi * \chi$, $\psi \neg * \chi$ and $\exists z \psi$.

induction step: case with $*$. Suppose $(G_1, s_1, h_1) \models \psi * \chi$. Then, there are two heaps h'_1 and h''_1 such that $h_1 = h'_1 + h''_1$, $(G_1, s_1, h'_1) \models \psi$ and $(G_1, s_1, h''_1) \models \chi$. Consider $h'_2 = f(h'_1)$ and $h''_2 = f(h''_1)$ to be the images of h'_1 and h''_1 via f . As f is an g - X -heap-isomorphism between h_1 and h_2 , we have:

$$h_2 = f(h_1) = f(h'_1 + h''_1) = f(h'_1) + f(h''_1) = h'_2 + h''_2$$

and moreover $h'_2 \perp h''_2$. Lastly, both $(G_1, s_1, h'_1) \simeq_X^g (G_2, s_2, h'_2)$ and $(G_1, s_1, h''_1) \simeq_X^g (G_2, s_2, h''_2)$ hold, as one can check that f is an g - X -heap-isomorphism also for these structures. By the induction hypothesis, $(G_2, s_2, h'_2) \models \psi$ and $(G_2, s_2, h''_2) \models \chi$. Thus, $(G_2, s_2, h_2) \models \psi * \chi$.

induction step: case with $\rightarrow*$. Suppose that $(G_1, s_1, h_1) \models \psi \rightarrow* \chi$ hold. Then, for every heap h'_1 , if $h'_1 \perp h_1$ and $(G_1, s_1, h'_1) \models \psi$ then $(G_1, s_1, h_1 + h'_1) \models \chi$. In order to show that $(G_2, s_2, h_2) \models \psi \rightarrow* \chi$ holds, let us consider a heap h'_2 s.t. $h'_2 \perp h_2$ and $(G_2, s_2, h'_2) \models \psi$ hold. We show that then $(G_2, s_2, h_2 + h'_2) \models \chi$ holds. First, by recalling that f^{-1} is a bijection from G_2 to G_1 , we construct the heap $h'_1 = f^{-1}(h'_2)$. By definition of h'_1 , the two following properties are satisfied:

1. f is an g - X -heap-isomorphism (from (G_1, s_1, h'_1) to (G_2, s_2, h'_2)),
2. $h'_1 \perp h_1$ holds (from (1), $(G_1, s_1, h'_1) \simeq_X^g (G_2, s_2, h'_2)$, $h'_2 \perp h_2$ and $f^{-1}(h'_2) = h'_1$).

By the induction hypothesis, (1) allows us to derive that $(G_1, s_1, h'_1) \models \psi$. Then, from (2) together with the initial hypothesis $(G_1, s_1, h_1) \models \psi \rightarrow* \chi$, we obtain that $(G_1, s_1, h_1 + h'_1)$ satisfies χ . By definition $f(h_1 + h'_1) = f(h_1) + f(f^{-1}(h'_2)) = h_2 + h'_2$ and therefore by induction hypothesis we get $(G_2, s_2, h_2 + h'_2) \models \chi$. Thus, $(G_2, s_2, h_2) \models \psi \rightarrow* \chi$.

induction step: case with \exists . Suppose $(G_1, s_1, h_1) \models \exists z \psi$. Then, there is a location $\ell \in G_1$ such that $(G_1, s_1[z \leftarrow \ell], h_1) \models \psi$. We need to prove that $(G_2, s_2, h_2) \models \exists z \psi$, which is true whenever there is a location $\ell' \in G_2$ such that $(G_2, s_2[z \leftarrow \ell'], h_2) \models \psi$. Let us consider the location $\ell' = f(\ell)$ in G_2 . By definition of g - $(X \cup \{z\})$ -heap-isomorphism it holds that $(G_1, s_1[z \leftarrow \ell], h_1) \simeq_{X \cup \{z\}}^g (G_2, s_2[z \leftarrow \ell'], h_2)$. Moreover, from $\text{fv}(\exists z \psi) \subseteq X$ we derive $\text{fv}(\psi) \subseteq X \cup \{z\}$. This allows us to apply the induction hypothesis, leading to $(G_2, s_2[x \leftarrow \ell'], h_2) \models \psi$. Consequently, $(G_2, s_2, h_2) \models \exists x \psi$. \square

As a direct consequence of this lemma, satisfiability in $\text{SL}(\exists, *, \rightarrow*)$ defined in Section 2.1 is equivalent to satisfiability with generalised memory states. Indeed, if φ is satisfied by the memory state (s, h) , then it is satisfied by the generalised memory state (LOC, s, h) . Similarly, suppose that φ is satisfiable in the generalised memory state (G, s, h) . As G is countably infinite, there is a bijection $f : G \rightarrow \text{LOC}$. Consider the memory state (s', h') defined as follows:

- for every $x \in \text{VAR}$, $s'(x) \stackrel{\text{def}}{=} f(s(x))$,
- the heap h' is defined as $f(h)$.

From the definition of (s', h') , we have $(G, s, h) \simeq_{\text{fv}(\varphi)}^g (\text{LOC}, s', h')$. By Lemma 3.3, we conclude $(\text{LOC}, s', h') \models \varphi$, which is equivalent to $(s', h') \models \varphi$.

3.1.2 The encoded-by relation \triangleright_Y^X

We now formalise the idea discussed at the beginning of this section on how to encode the store as a part of the heap. We do this by defining a relation \triangleright_Y^X between generalised memory states and say that (G_1, s_1, h_1) is encoded by (G_2, s_2, h_2) with respect to two sets of variables $X \subseteq Y$, whenever $(G_1, s_1, h_1) \triangleright_Y^X (G_2, s_2, h_2)$ holds. For instance, we will see that the memory state in Figure 3.1 is encoded by the one in Figure 3.2. As previously stated, when considering the satisfaction of a formula φ with respect of the encoding (G_2, s_2, h_2) , we rely on additional variables not appearing in φ in order to deal with the separating implication. Informally, X keeps track of the free variables in φ , whereas the auxiliary set Y represent all the variables needed to perform the encoding (thus $X \subseteq Y$). Let us define the encoded-by relation \triangleright_Y^X .

Definition 3.4 (Encoded-by relation). Let $X \subseteq Y \subseteq \text{fin VAR}$ be a finite set of variables. Let (G_1, s_1, h_1) and (G_2, s_2, h_2) be generalised memory states. (G_1, s_1, h_1) is *encoded-by* (G_2, s_2, h_2) with respect to X and Y , written $(G_1, s_1, h_1) \triangleright_Y^X (G_2, s_2, h_2)$, if the following conditions hold:

- $G_1 = G_2 \setminus \{s_2(x) \mid x \in Y\}$,
- $h_2 = h_1 + \{s_2(x) \mapsto s_1(x) \mid x \in X\}$.
- for $x, y \in Y$, if $x \neq y$ then $s_2(x) \neq s_2(y)$,

Notice that G_2 is obtained from G_1 by adding the locations in $\{s_2(x) \mid x \in Y\}$. Locations in the latter set are reserved to simulate the store. Moreover, h_2 is equal to the heap h_1 augmented with the heap $\{s_2(x) \mapsto s_1(x) \mid x \in X\}$, which encodes the store s_1 restricted to the domain X . In particular, notice that this heap is a subset of $\{s_2(x) \mid x \in X\} \times G_1$, which means that all its arrows have sources in the region reserved to simulate the store, and targets in the set of locations G_1 . Figure 3.2 represents quite clearly an encoding of the memory state in Figure 3.1, in the case where $X = \{x, y, z\}$. The set Y made of locations reserved to simulate the store is highlighted. In the figure, it includes the three locations that are assigned to x , y and z , plus three unlabelled locations which represent elements in $Y \setminus X$. Furthermore, the heap $\{s_2(x) \mapsto s_1(x) \mid x \in X\}$ is represented by the three arrows leaving the highlighted region.

The encoded-by relation formally describes the memory states we had in mind when considering the problem of simulating first-order quantification by means of heap manipulations. In the following section, we formalise this manipulation inside a fragment of $SL(*, -, \hookrightarrow^+)$.

3.2 SIMULATING THE FIRST-ORDER QUANTIFICATION

As informally described in the previous section, encoding a store as a part of the heap has some impact on the satisfaction of $x \hookrightarrow y$ and $x = y$. Indeed, given a generalised memory state (G, s, h) , these two properties should be now checked with respect to the locations that are pointed by $s(x)$ and $s(y)$. For instance, instead of checking whether $h(s(x)) = s(y)$ as required by $x \hookrightarrow y$, we must now check for $h^2(s(x)) = h(s(y))$. Besides, in well-formed encodings the locations that are reserved to encode the store are not pointed by any location, i.e. they do not satisfy the alloc-back predicate $_ \hookrightarrow (\cdot)$. All these adaptations naturally lead us to consider a separation logic, denoted by $SL(n(x), *, -*)$, whose formulae are from the following grammar:

$$\begin{array}{lll} \pi := & \top \mid \text{emp} \mid x = y & \varphi := \pi & \text{(atomic formulae)} \\ | & n(x) = n(y) & | & \varphi \wedge \varphi \mid \neg \varphi \\ | & n(x) \hookrightarrow n(y) & | & \varphi * \varphi \\ | & _ \hookrightarrow x & | & \varphi \multimap \varphi \end{array}$$

Given a generalised memory state (G, s, h) , the satisfaction relation \models for the formulae of $SL(n(x), *, -*)$ is defined as in Figure 2.3 for the syntactical elements that the logic has in common with $SL(\exists, *, -*)$, whereas it is given in Figure 3.4 for the other predicates. For instance, the memory state in Figure 3.5 satisfies $n(x) = n(y)$, $n(z) \hookrightarrow n(y)$ and $n(x) \hookrightarrow n(z)$, but it does not satisfy $_ \hookrightarrow v$, for any $v \in \{x, y, z\}$. A small remark: the equivalence relation $=$ used to define the semantics of $n(x) = n(y)$, and $n(x) \hookrightarrow n(y)$ is a binary relation on the set of locations. In particular, given two locations $\ell, \ell' \notin \text{dom}(h)$, $h(\ell) = h(\ell')$ does not hold. This means that the formula $n(x) = n(x)$ is equivalent to the alloc predicate $x \hookrightarrow _$ introduced in Section 2.1.1.

The logic $SL(n(x), *, -*)$ is a quantifier-free syntactical fragment of $SL(\exists, *, -*)$. We already saw how to express alloc-back predicate in Section 2.1.1. Concerning the predicates $n(x) = n(y)$ and $x \hookrightarrow y$, we have the two following equivalences:

$$n(x) = n(y) \equiv \exists z (x \hookrightarrow z \wedge y \hookrightarrow z), \quad n(x) \hookrightarrow n(y) \equiv \exists z \exists v (x \hookrightarrow z \wedge z \hookrightarrow v \wedge y \hookrightarrow v).$$

Despite being quantifier-free, we prove the following result.

$$\begin{aligned}
 (G, s, h) \models n(x) = n(y) &\quad \text{iff} \quad h(s(x)) = h(s(y)), \\
 (G, s, h) \models n(x) \hookrightarrow n(y) &\quad \text{iff} \quad h(h(s(x))) = h(s(y)), \\
 (G, s, h) \models _ \hookrightarrow x &\quad \text{iff} \quad s(x) \in \text{ran}(h).
 \end{aligned}$$

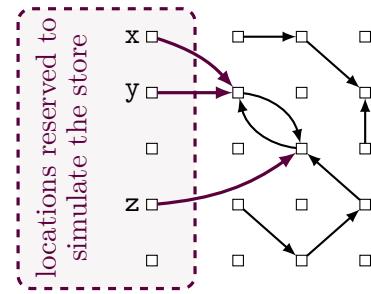
Figure 3.4: The semantics of $\text{SL}(n(x), *, _*)$ predicates.

Figure 3.5: An encoding.

Theorem 3.5. The satisfiability and validity problems of $\text{SL}(n(x), *, _*)$ are not RE.

This theorem is quite interesting when compared to other well-known complexity results. For instance, recall that $\text{SL}([\exists]_1, *, _*)$, i.e. the one quantified variable fragment of $\text{SL}(\exists, *, _*)$, is proven to admit a PSPACE-complete satisfiability problem [55]. This logic can express the predicates $_ \hookrightarrow x$ and $n(x) = n(y)$ by directly using the definition we gave for $\text{SL}(\exists, *, _*)$. Theorem 3.5 shows that enriching $\text{SL}([\exists]_1, *, _*)$ with the predicate $n(x) \hookrightarrow n(y)$ makes the satisfiability problem does a remarkable jump: from PSPACE to non RE. Moreover, in Section 3.3 we show that the three predicates $n(x) = n(y)$, $n(x) \hookrightarrow n(y)$ and $_ \hookrightarrow x$ can be expressed using a bounded variant of the list-segment predicate $1s$, leading to various extensions of the quantifier-free separation logic $\text{SL}(*, _*)$ to admit non RE satisfiability and validity problems.

The proof of Theorem 3.5 achieved by showing that $\text{SL}(n(x), *, _*)$ can simulate the first-order quantification of $\text{SL}(\exists, _*)$, i.e. the fragment of $\text{SL}(\exists, *, _*)$ without the separating conjunction. The satisfiability problem of this logic is proven undecidable in [22], and one can check that this implies that both satisfiability and validity are not RE by replaying the arguments used for $\text{SL}(\exists, *, _*)$ in Section 2.1.2. The formulae φ for the separation logic $\text{SL}(\exists, _*)$ considered in [22] are built from the following grammar (where $x, y, z \in \text{VAR}$):

$$\varphi ::= x = y \mid x \hookrightarrow y \mid \varphi \wedge \varphi \mid \neg \varphi \mid \varphi \dashv \varphi \mid \exists z \varphi.$$

In this presentation of $\text{SL}(\exists, _*)$, notice that the logic does not directly feature the atomic formulae T and emp of $\text{SL}(\exists, *, _*)$, which can be defined as $x = x$ and $\neg \exists x \exists y x \hookrightarrow y$, respectively. We recall the satisfaction relation \models for the formulae of $\text{SL}(\exists, _*)$ in Figure 3.6, adapting it to generalised memory states. As $\text{SL}(\exists, _*)$ is a fragment of first-order separation logic, by Lemma 3.3 we know that this adaptation does not change the notion of satisfiability.

3.2.1 Translating $\text{SL}(\exists, _*)$ into $\text{SL}(n(x), *, _*)$.

We define a translation of a formula of $\text{SL}(\exists, _*)$ into a quantifier-free formula of $\text{SL}(n(x), *, _*)$. For simplicity, we assume every formula φ in $\text{SL}(\exists, _*)$ to be *well-quantified*, meaning that every two distinct quantifiers appearing in φ involve distinct variables. This assumption, often called Barendregt's convention, can be done without loss of generality. Indeed, every subformula $\exists z \psi$ of φ can be replaced with the equivalent formula $\exists v \psi[z \leftarrow v]$, where v is a variable not appearing in ψ and $\psi[z \leftarrow v]$ is the formula obtained from ψ by replacing every occurrence of z with v .

Assumption 3.6. The formulae in $\text{SL}(\exists, _*)$ are well-quantified.

$$\begin{aligned}
(G, s, h) \models x = y &\quad \text{iff } s(x) = s(y), \\
(G, s, h) \models x \hookrightarrow y &\quad \text{iff } h(s(x)) = s(y), \\
(G, s, h) \models \varphi \wedge \psi &\quad \text{iff } (G, s, h) \models \varphi \text{ and } (G, s, h) \models \psi, \\
(G, s, h) \models \neg \varphi &\quad \text{iff } (G, s, h) \not\models \varphi, \\
(G, s, h) \models \varphi \multimap \psi &\quad \text{iff } \text{for every heap } h' : G \multimap_{\text{fin}} G, \text{ if } h' \perp h \text{ and } (G, s, h') \models \varphi \\
&\quad \text{then } (G, s, h + h') \models \psi, \\
(G, s, h) \models \exists z \varphi &\quad \text{iff there is } \ell \in G \text{ such that } (G, s[z \leftarrow \ell], h) \models \varphi.
\end{aligned}$$

Figure 3.6: Satisfaction relation for $\text{SL}(\exists, \multimap)$, for a generalised memory state.

Let $X \subseteq Y \subseteq_{\text{fin}} \text{VAR}$. Below, let us fix a (well-quantified) formula φ of $\text{SL}(\exists, \multimap)$ with free variables from X and bound variables from $Y \setminus X$. To correctly setup a framework that can deal with the separating implication, every variable $x \in Y$ is paired with a distinct *copy* \bar{x} . Formally, we pick a set $\bar{Y} \subseteq_{\text{fin}} \text{VAR}$ such that $\bar{Y} \cap Y = \emptyset$ and $\text{card}(\bar{Y}) = \text{card}(Y)$. We write $\bar{Y} \stackrel{\text{def}}{=} Y \cup \bar{Y}$ to denote the union of these two sets. Next, we build a correspondence between variables and their copies by defining an *involution* $(\cdot) : \bar{Y} \rightarrow \bar{Y}$ associating every $x \in Y$ with its copy $\bar{x} \in \bar{Y}$. Recall that (\cdot) being an involution means that it is a bijection such that $\bar{\bar{x}} = x$.

The translation of φ into a formula $\tau_{X, Y}(\varphi)$ of $\text{SL}(n(x), *, \multimap)$ is divided into two cases: a base case for atomic formulae and “inductive” case for non-atomic ones. This division is quite natural, since we aim at proving the following correctness lemma by structural induction on φ .

Lemma 3.7. Let $(G_1, s_1, h_1) \triangleright_{\bar{Y}}^X (G_2, s_2, h_2)$. $(G_1, s_1, h_1) \models \varphi$ iff $(G_2, s_2, h_2) \models \tau_{X, Y}(\varphi)$.

Base cases. The definition of $\tau_{X, Y}(\varphi)$ is straightforward when φ is a formula of the form $x = y$ or $x \hookrightarrow y$: we simply need to rely on the properties of the encoding given in the previous section.

$$\tau_{X, Y}(x = y) \stackrel{\text{def}}{=} n(x) = n(y), \quad \tau_{X, Y}(x \hookrightarrow y) \stackrel{\text{def}}{=} n(x) \hookrightarrow n(y).$$

The correctness of these two cases of the translation follows from the lemma below. Besides, this lemma corresponds to the base case of the proof by induction we employ to show Lemma 3.7.

Lemma 3.8. Let $X \subseteq Y$. Let φ be an atomic formula of the form $x = y$ or $x \hookrightarrow y$, where $x, y \in X$. Given $(G_1, s_1, h_1) \triangleright_{\bar{Y}}^X (G_2, s_2, h_2)$, $(G_1, s_1, h_1) \models \varphi$ iff $(G_2, s_2, h_2) \models \tau_{X, Y}(\varphi)$.

Proof. Let us consider two generalised memory states (G_1, s_1, h_1) and (G_2, s_2, h_2) such that $(G_1, s_1, h_1) \triangleright_{\bar{Y}}^X (G_2, s_2, h_2)$. We split the proof into two cases, following whether φ is an equality predicate or a points-to predicate.

case: $\varphi = x = y$. The following equivalences hold:

$$\begin{aligned}
(G_1, s_1, h_1) \models x = y &\quad \text{iff } s_1(x) = s_1(y) && \text{(by def. of } \models\text{)} \\
&\quad \text{iff } h_2(s_2(x)) = s_1(x) = s_1(y) = h_2(s_2(y)) && \text{(by def. of } \triangleright_{\bar{Y}}^X\text{)} \\
&\quad \text{iff } (G_2, s_2, h_2) \models n(x) = n(y). && \text{(by def. of } \models\text{)}
\end{aligned}$$

case: $\varphi = x \hookrightarrow y$. The following equivalences hold:

$$\begin{aligned}
(G_1, s_1, h_1) \models x \hookrightarrow y &\text{ iff } h_1(s_1(x)) = s_1(y) && (\text{by def. of } \models) \\
&\text{ iff } h_2(s_2(x)) = s_1(x), h_2(s_2(y)) = s_1(y) \\
&\quad \text{and } h_2(s_1(x)) = s_1(y) && (\text{by def. of } \triangleright_{\mathbb{Y}}^X) \\
&\text{ iff } (G_2, s_2, h_2) \models n(x) \hookrightarrow n(y). && (\text{by def. of } \models) \quad \square
\end{aligned}$$

Well-formed encodings. Before moving to the inductive cases, let us consider a generalised memory state (G_1, s_1, h_1) and one of its encodings (G_2, s_2, h_2) with respect to $X \subseteq \mathbb{Y}$, i.e. $(G_1, s_1, h_1) \triangleright_{\mathbb{Y}}^X (G_2, s_2, h_2)$. We recall that the definition of the encoded-by relation realises our intuitive idea of isolating a finite amount of locations in order to mimic the store s_1 by using the heap h_2 . This is done by enforcing the following three properties:

1. $G_1 = G_2 \setminus \{s_2(x) \mid x \in \mathbb{Y}\}$,
2. given $x, y \in \mathbb{Y}$, if $x \neq y$ then $s_2(x) \neq s_2(y)$,
3. $h_2 = h_1 + \{s_2(x) \mapsto s_1(x) \mid x \in X\}$.

These properties must be checked during the translation, so that the first-order quantification and the separating implication only consider generalised memory states that are *well-formed* with respect to the encoded-by relation (that is, they encode some memory state). For this reason, given a set $X \subseteq \mathbb{Y}$, we introduce the formula $\text{Store}_{\mathbb{Y}}(X)$ defined as follows

$$\text{Store}_{\mathbb{Y}}(X) \stackrel{\text{def}}{=} \left(\bigwedge_{\substack{x, y \in \mathbb{Y} \\ x \neq y}} x \neq y \right) \wedge \left(\bigwedge_{x \in \mathbb{Y}} \neg _ \hookrightarrow x \right) \wedge \left(\bigwedge_{x \in X} x \hookrightarrow _ \right) \wedge \left(\bigwedge_{x \in \mathbb{Y} \setminus X} \neg x \hookrightarrow _ \right).$$

where $x \hookrightarrow _ \stackrel{\text{def}}{=} n(x) = n(x)$ in $\text{SL}(n(x), *, -*)$. We can see that this formula captures some essential trait of the properties above: its first conjunct is equivalent to the second property, whereas the other three conjuncts essentially verify the first and third properties. The relationship between $\text{Store}_{\mathbb{Y}}(X)$ and the encoded-by relation is formalised in the next lemma.

Lemma 3.9. Let (G_2, s_2, h_2) be a generalised memory state. There is a generalised memory state (G_1, s_1, h_1) such that $(G_1, s_1, h_1) \triangleright_{\mathbb{Y}}^X (G_2, s_2, h_2)$ if and only if $(G_2, s_2, h_2) \models \text{Store}_{\mathbb{Y}}(X)$.

Proof. For both directions we use the properties (1), (2) and (3) of the relation $\triangleright_{\mathbb{Y}}^X$ above.

(\Rightarrow): Suppose $(G_1, s_1, h_1) \triangleright_{\mathbb{Y}}^X (G_2, s_2, h_2)$. (G_2, s_2, h_2) satisfies the four conjuncts of $\text{Store}_{\mathbb{Y}}(X)$:

(first conjunct) Directly from property (2).

(second conjunct) Let us consider a variable $x \in \mathbb{Y}$. From property (1) we derive $s_2(x) \notin G_1$, which means that $s_2(x) \notin \text{ran}(h_1)$ (indeed, h_1 is in $[G_1 \rightarrow_{\text{fin}} G_1]$). Similarly, since for every $y \in \mathbb{Y}$ it holds that $s_1(y) \in G_1$, we conclude that $s_2(x) \notin \text{ran}(\{s_2(x) \mapsto s_1(x) \mid x \in X\})$. Thus, from property (3) we have $s_2(x) \notin \text{dom}(h_2)$, which implies $(G_2, s_2, h_2) \models \neg _ \hookrightarrow x$.

(third conjunct) Directly from property (3).

(fourth conjunct) Let $x \in \mathbb{Y} \setminus X$. By property (1), $s_2(x) \notin G_1$ and therefore $s_2(x) \notin \text{dom}(h_1)$. Besides, by definition $s_2(x) \notin \text{dom}(\{s_2(x) \mapsto s_1(x) \mid x \in X\})$. From the property (3), it holds that $s_2(x) \notin \text{dom}(h_2)$, or equivalently $(G_2, s_2, h_2) \models \neg x \hookrightarrow _$.

(\Leftarrow): Suppose $(G_2, s_2, h_2) \models \text{Store}_{\mathbb{Y}}(X)$. Let us define the set $G_1 \stackrel{\text{def}}{=} G_2 \setminus \{s_2(x) \mid x \in \mathbb{Y}\}$, the heap $h_1 \stackrel{\text{def}}{=} \{(\ell, \ell') \in h_2 \mid \ell, \ell' \in G_1\}$ and a store s_1 such that $s_1(x) \stackrel{\text{def}}{=} h_2(s_2(x))$ for every $x \in X$. From the third conjunct of $\text{Store}_{\mathbb{Y}}(X)$, s_1 is well-defined. Moreover, G_1 is countably infinite and therefore the structure (G_1, s_1, h_1) is a generalised memory state. We show that $(G_1, s_1, h_1) \triangleright_{\mathbb{Y}}^X (G_2, s_2, h_2)$. The property (1) holds by definition, whereas the property (2) follows directly from the first conjunct of $\text{Store}_{\mathbb{Y}}(X)$. To prove the property (3), we consider the

$$\begin{aligned}
\tau_{X,Y}(\neg\psi) &\stackrel{\text{def}}{=} \neg\tau_{X,Y}(\psi), \\
\tau_{X,Y}(\psi_1 \wedge \psi_2) &\stackrel{\text{def}}{=} \tau_{X,Y}(\psi_1) \wedge \tau_{X,Y}(\psi_2), \\
\tau_{X,Y}(\exists z \psi) &\stackrel{\text{def}}{=} (z \hookrightarrow _ \wedge \mathbf{size} = 1) \multimap (\mathbf{Store}_Y(X \cup \{z\}) \wedge \tau_{X \cup \{z\}, Y}(\psi)), \\
\tau_{X,Y}(\psi_1 \multimap \psi_2) &\stackrel{\text{def}}{=} (\mathbf{Store}_Y(\bar{V}) \wedge \tau_{\bar{V}, Y}(\psi_1[x \leftarrow \bar{x} \mid x \in Y])) \multimap \\
&\quad ((\wedge_{\bar{x} \in \bar{V}} n(x) = n(\bar{x})) \Rightarrow (\mathbf{size} = \mathbf{card}(\bar{V}) \wedge \wedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _) * \tau_{X,Y}(\psi_2)), \\
&\quad \text{where } \bar{V} \stackrel{\text{def}}{=} \{\bar{x} \mid x \in \mathbf{fv}(\psi_1)\}.
\end{aligned}$$

Figure 3.7: Inductive cases of the translation to $\mathbf{SL}(n(x), *, \multimap)$.

structure $h' \stackrel{\text{def}}{=} \{s_2(x) \mapsto s_1(x) \mid x \in X\}$ and prove that it is a heap such that $h_2 = h_1 + h'$. From the definition of $s_1(x)$, we have $h' = \{s_2(x) \mapsto h_2(s_2(x)) \mid x \in X\}$, which implies that $h' \subseteq h_2$. Thus, h' is a heap. To prove that $h_2 = h_1 + h'$ holds, it is sufficient to show that $h_1 = h_2 \setminus h'$. The left-to-right inclusion $h_1 \subseteq h_2 \setminus h'$ holds as by definition $h_1 = h_2 \cap (G_1 \times G_1)$ and $\mathbf{dom}(h') \cap G_1 = \emptyset$. For the right-to-left inclusion, we prove that $(h_2 \setminus h') \subseteq G_1 \times G_1$. Let $(\ell, \ell') \in h_2 \setminus h'$. *Ad absurdum*, suppose $(\ell, \ell') \notin G_1 \times G_1$. By definition of G_1 , $\ell \in \{s_2(x) \mid x \in Y\}$ or $\ell' \in \{s_2(x) \mid x \in Y\}$. The latter case is contradictory as it implies $s_2(x) \in \mathbf{ran}(h_2)$ for some $x \in Y$, which is inconsistent with the second conjunct of $\mathbf{Store}_Y(X)$. The case $\ell \in \{s_2(x) \mid x \in Y\}$ is also contradictory. Indeed, $(\ell, \ell') \in h_2 \setminus h'$ implies $\ell \in \mathbf{dom}(h_2)$ and, by definition of h' , $\ell \in Y \setminus X$. However, from the fourth conjunct of $\mathbf{Store}_Y(X)$, every location in $Y \setminus X$ is not in the domain of h_2 . \square

Inductive cases. We are now ready to analyse the inductive cases of the translation. Their definition is given in Figure 3.7. As it is often the case with semantical preserving translations, $\tau_{X,Y}(\varphi)$ is homomorphic for the Boolean connectives, which fits nicely in the proof by structural induction of Lemma 3.7. The definition of $\tau_{X,Y}(\exists z \psi)$ captures quite closely our initial idea of seeing the first-order quantification as an update of the heap. More precisely, given a generalised memory state (G_2, s_2, h_2) satisfying $\tau_{X,Y}(\exists z \psi)$, it is possible to find a location ℓ such that $(G_2, s_2, h_2 + \{s_2(z) \mapsto \ell\}) \models \mathbf{Store}_Y(X \cup \{z\})$. When considering a generalised memory state (G_1, s_1, h_1) such that $(G_1, s_1, h_1) \triangleright_X^Y (G_2, s_2, h_2)$, this means that ℓ is in G_1 , so that $(G_1, s_1[z \leftarrow \ell], h_2) \triangleright_Y^{X \cup \{z\}} (G_2, s_2, h_2 + \{s_2(z) \mapsto \ell\})$. In a nutshell, adding the heap $\{s_2(z) \mapsto \ell\}$ to h_2 corresponds to the case where, after an existential quantification, ℓ is assigned to z .

For the formula $\psi_1 \multimap \psi_2$, we cannot follow a similar idea and simply translate it to the formula $\tau_{X,Y}(\psi_1) \multimap (\mathbf{Store}_Y(X) \Rightarrow \tau_{X,Y}(\psi_2))$. Indeed, the evaluation of $\tau_{X,Y}(\psi_1)$ in a disjoint heap may need the values of free variables occurring in ψ_1 , but our encoding of the variable valuations via the heap does not allow to preserve these values through disjoint heaps. Here is where the set \bar{V} of copies of the variables comes into play. Consider a generalised memory state (G, s, h) satisfying $\mathbf{Store}_Y(X)$ and $\mathbf{fv}(\psi_1 \multimap \psi_2) \subseteq X$. In particular, for every $x \in \mathbf{fv}(\psi_1)$ the location $s(x)$ is a memory cell of h . To check whether (G, s, h) satisfies $\tau_{X,Y}(\psi_1 \multimap \psi_2)$, the left side of the magic wand considers heaps h' encoding a store with respect to the copies of the variables in $\mathbf{fv}(\psi_1)$, i.e. such that (G, s, h') satisfies $\mathbf{Store}_Y(\bar{V})$ were $\bar{V} \stackrel{\text{def}}{=} \{\bar{x} \mid x \in \mathbf{fv}(\psi_1)\}$ (as in the translation). On this heap, instead of asking whether $\tau_{X,Y}(\psi_1)$ holds, we can check for the satisfaction of $\tau_{X,Y}(\psi_1[x \leftarrow \bar{x} \mid x \in Y])$, where $\psi_1[x \leftarrow \bar{x} \mid x \in Y]$ denotes the formula obtained from ψ_1 by replacing every variable $x \in X$ with its copy, and vice versa (recall that $\bar{(\cdot)}$ is an involution). We need however to be careful and only consider h' if the store it encodes is

compatible with the one encoded by h : for every $\bar{x} \in \bar{V}$, $h'(s_2(\bar{x}))$ must be the same location as $h(s_2(x))$. We check this property directly on the heap $h + h'$ by using the formula $n(x) = n(\bar{x})$. When this property holds, we can employ the separating conjunction to remove the store encoded by h' from $h + h'$, and ask for the satisfaction of $\tau_{X,Y}(\psi_2)$ on the resulting memory state. Notice that, by removing the store encoded by h' , we can reuse \bar{x} to deal with the occurrences of the magic wand inside ψ_2 . Similarly, the occurrences of the magic wand inside $\psi_1[x \leftarrow \bar{x} \mid x \in Y]$ are dealt with by using the copies of the variables appearing in this formula (recall that (\cdot) is an involution), which does not correspond to memory cells in h' .

Example 3.10. Let us consider the formula $\varphi \stackrel{\text{def}}{=} (x \hookrightarrow y \multimap x \hookrightarrow z) \multimap y = z$ and show its translation. Let $X = \{x, y, z\}$ and $Y = \{x, y, z, \bar{x}, \bar{y}, \bar{z}\}$. The translation leads to the following equivalences:

$$\begin{aligned} \tau_{X,Y}(\varphi) &= (\text{Store}_Y(\bar{X}) \wedge \tau_{\bar{X},Y}(\bar{x} \hookrightarrow \bar{y} \multimap \bar{x} \hookrightarrow \bar{z})) \multimap \\ &\quad ((\bigwedge_{\bar{v} \in \bar{X}} n(v) = n(\bar{v})) \Rightarrow (\text{size} = 3 \wedge \bigwedge_{\bar{v} \in \bar{X}} \bar{v} \hookrightarrow _) \multimap \tau_{X,Y}(y = z)), \\ \tau_{X,Y}(\bar{x} \hookrightarrow \bar{y} \multimap \bar{x} \hookrightarrow \bar{z}) &= (\text{Store}_Y(\{x, y\}) \wedge \tau_{\{x,y\},Y}(x \hookrightarrow y)) \multimap \\ &\quad ((\bigwedge_{v \in \{x,y\}} n(v) = n(\bar{v})) \Rightarrow (\text{size} = 2 \wedge \bigwedge_{v \in \{x,y\}} v \hookrightarrow _) \multimap \tau_{X,Y}(\bar{x} \hookrightarrow \bar{z})), \\ \tau_{X,Y}(x \hookrightarrow y) &= n(x) \hookrightarrow n(y), \\ \tau_{X,Y}(\bar{x} \hookrightarrow \bar{z}) &= n(\bar{x}) \hookrightarrow n(\bar{z}), \\ \tau_{X,Y}(y = z) &= n(y) = n(z). \end{aligned}$$

Notice how the translation alternates between the variables in X and their copies when imbricating the separating implication on its left side (as it is the case for the formula φ).

The following lemma subsumes Lemma 3.7 and thus proves the correctness of our translation. The statement is given with respect to the disjoint finite sets of variables Y and \bar{Y} , as well as the set Y endowed with the involution (\cdot) , as defined in this section. We recall that for the formula φ , we assume that distinct quantifications involve distinct variables.

Lemma 3.11. Let Z be either Y or \bar{Y} and let $X \subseteq Z$. Let φ be a formula in $SL(\exists, \multimap)$ s.t. $fv(\varphi) \subseteq X$ and $bv(\varphi) \subseteq Z \setminus X$. Given $(G_1, s_1, h_1) \triangleright_Y^X (G_2, s_2, h_2)$, $(G_1, s_1, h_1) \models \varphi$ iff $(G_2, s_2, h_2) \models \tau_{X,Y}(\varphi)$.

Proof. The proof goes by structural induction on φ (with the natural induction hypothesis stating that the property holds for strict subformulae of φ). The base cases for φ of the form $x = y$ and $x \hookrightarrow y$, where $x, y \in X$, hold from Lemma 3.8. We omit the obvious cases for the Boolean connectives, which leaves us with the two cases for $\varphi = \exists z \psi$ and $\varphi = \psi_1 \multimap \psi_2$. Let us consider two memory states (G_1, s_1, h_1) and (G_2, s_2, h_2) such that $(G_1, s_1, h_1) \triangleright_Y^X (G_2, s_2, h_2)$.

case: $\varphi = \exists z \psi$. By definition of φ , the variable z is in $Z \setminus X$, which implies $s_2(z) \notin \text{dom}(h_2)$ by definition of \triangleright_Y^X . We refer to Figure 3.7 for the definition of $\tau_{X,Y}(\exists z \psi)$.

(\Rightarrow) : Suppose $(G_1, s_1, h_1) \models \exists z \psi$, which implies that there is a location $\ell \in G_1$ such that $(G_1, s_1[z \leftarrow \ell], h_1) \models \psi$ holds. Let us consider the structure $h' \stackrel{\text{def}}{=} \{s_2(z) \mapsto \ell\}$. The location ℓ is from G_1 whereas, from the definition of \triangleright_Y^X , $s_2(z)$ is a variable in $G_2 \setminus G_1$. This makes h' a heap in $[G_2 \rightarrow_{\text{fin}} G_2]$. Moreover, as $s_2(z) \notin \text{dom}(h_2)$, h' is disjoint from h_2 . Clearly, $(G_2, s_2, h') \models z \hookrightarrow _ \wedge \text{size} = 1$ holds, which means that to conclude this direction of the proof is sufficient to show that $(G_2, s_2, h_2 + h')$ satisfies the formula $\text{Store}_Y(X \cup \{z\}) \wedge \tau_{X \cup \{z\},Y}(\psi)$. For this, the essential step is to establish that

$$(G_1, s_1[z \leftarrow \ell], h_1) \triangleright_{\mathbb{Y}}^{X \cup \{z\}} (G_2, s_2, h_2 + h'),$$

which amounts to prove that

1. $G_1 = G_2 \setminus \{s_2(x) \mid x \in \mathbb{Y}\}$,
2. given $x, y \in \mathbb{Y}$, if $x \neq y$ then $s_2(x) \neq s_2(y)$,
3. $h_2 + h' = h_1 + \{s_2(x) \mapsto s_1[z \leftarrow \ell](x) \mid x \in X \cup \{z\}\}$.

The properties (1) and (2) follow directly from the hypothesis that $(G_1, s_1, h_1) \triangleright_{\mathbb{Y}}^X (G_2, s_2, h_2)$. Moreover, this hypothesis implies $h_2 = h_1 + \{s_2(x) \mapsto s_1(x) \mid x \in X\}$. By simply adding h' to both sides of this equation we obtain

$$h_2 + h' = h_1 + \{s_2(x) \mapsto s_1(x) \mid x \in X\} + \{s_2(z) \mapsto \ell\}.$$

By definition of $s_1[z \leftarrow \ell]$, the heap $\{s_2(x) \mapsto s_1(x) \mid x \in X\} + \{s_2(z) \mapsto \ell\}$ is equivalent to $\{s_2(x) \mapsto s_1[z \leftarrow \ell](x) \mid x \in X \cup \{z\}\}$, which concludes the proof of the property (3). This concludes the proof of $(G_1, s_1[z \leftarrow \ell], h_1) \triangleright_{\mathbb{Y}}^{X \cup \{z\}} (G_2, s_2, h_2 + h')$, which in turn allows us to conclude $(G_2, s_2, h_2 + h') \models \text{Store}_{\mathbb{Y}}(X \cup \{z\})$ directly by Lemma 3.9, as well as $(G_2, s_2, h_2 + h') \models \tau_{X \cup \{z\}, \mathbb{Y}}(\psi)$, by induction hypothesis.

(\Leftarrow): Suppose $(G_2, s_2, h_2) \models \tau_{X, \mathbb{Y}}(\exists z \psi)$. By definition of $\tau_{X, \mathbb{Y}}(\exists z \psi)$, there is $h' : G_2 \rightarrow_{\text{fin}} G_2$ disjoint from h_2 and such that

- A. $(G_2, s_2, h') \models z \hookrightarrow _ \wedge \text{size} = 1$,
- B. $(G_2, s_2, h_2 + h') \models \text{Store}_{\mathbb{Y}}(X \cup \{z\}) \wedge \tau_{X \cup \{z\}, \mathbb{Y}}(\psi)$.

From (A), $h' = \{s_2(z) \mapsto \ell\}$ for some $\ell \in G_2$. More precisely, from the satisfaction of $\text{Store}_{\mathbb{Y}}(X \cup \{z\})$ in (B), ℓ is not assigned to any of the variables in \mathbb{Y} and therefore, by $(G_1, s_1, h_1) \triangleright_{\mathbb{Y}}^X (G_2, s_2, h_2)$, ℓ can only be a location in G_1 . As done for the left-to-right direction, this allows us to prove that $(G_1, s_1[z \leftarrow \ell], h_1) \triangleright_{\mathbb{Y}}^{X \cup \{z\}} (G_2, s_2, h_2 + h')$, which in turn implies $(G_1, s_1[z \leftarrow \ell], h_1) \models \psi$ by induction hypothesis (using (B)). Lastly, by definition of $\exists z \psi$ we conclude: $(G_1, s_1, h_1) \models \exists z \psi$.

The proof of the case for $\varphi = \psi_1 \multimap \psi_2$ requires the following substitution lemma, which holds for every formula in either $\text{SL}(n(x), *, \multimap)$ or $\text{SL}(\exists, \multimap)$:

(Sub) Let χ be a formula with variables in Z , where $Z = \mathbb{Y}$ or $Z = \bar{\mathbb{Y}}$. Let (G, s, h) be a generalised memory state. $(G, s, h) \models \chi$ iff $(G, s[\bar{x} \leftarrow s(x) \mid x \in Z], h) \models \chi[x \leftarrow \bar{x} \mid x \in Z]$.

This result essentially states that renaming all the variables in a formula with its copies and updating the store s adequately, i.e. $s(\bar{x}) = s(x)$ for every x in the formula, does not change the notion of satisfiability. A quick way to prove (Sub) is noticing that the two memory states (G, s, h) and $(G, s[\bar{x} \leftarrow s(x) \mid x \in Z], h)$ are g-Z-isomorphic, and therefore they equisatisfy χ by Lemma 3.3. Moreover, $(G, s[\bar{x} \leftarrow s(x) \mid x \in Z], h) \models x = \bar{x}$ for every $x \in Z$, which allows us to conclude (Sub) by relying on the following well-known tautology of separation logic:

$$\models x = \bar{x} \wedge \chi \Rightarrow \chi[x \leftarrow \bar{x}].$$

Details are omitted to let us focus on the case of $\varphi = \psi_1 \multimap \psi_2$, which we now develop.

case: $\varphi = \psi_1 \multimap \psi_2$. We refer to Figure 3.7 for the definition of $\tau_{X, \mathbb{Y}}(y_1 \multimap y_2)$. As in this figure, we use $\bar{\mathbb{Y}}$ to denote $\{\bar{x} \mid x \in \text{fv}(\psi_1)\}$. Moreover, \bar{Z} stands for the set $\{\bar{x} \mid x \in Z\}$ (recall that by hypothesis Z is \mathbb{Y} or $\bar{\mathbb{Y}}$), so that $\bar{\mathbb{Y}} \subseteq \bar{Z}$. As extensively used for both directions of the proof, we recall that the hypothesis $(G_1, s_1, h_1) \triangleright_{\mathbb{Y}}^X (G_2, s_2, h_2)$ implies that

- \triangleright_1 . $G_1 = G_2 \setminus \{s_2(x) \mid x \in \mathbb{Y}\}$,
- \triangleright_2 . given $x, y \in \mathbb{Y}$, if $x \neq y$ then $s_2(x) \neq s_2(y)$,

$\triangleright_3. h_2 = h_1 + \{s_2(x) \mapsto s_1(x) \mid x \in X\}.$

Below, the indices (\triangleright_1) , (\triangleright_2) and (\triangleright_3) refer to these three properties.

(\Rightarrow) : Suppose $(G_1, s_1, h_1) \models \psi_1 \multimap \psi_2$. By definition, for every heap $h' : G_1 \rightarrow_{fin} G_1$, if $h' \perp h_1$ and $(G_1, s_1, h') \models \psi_1$ then $(G_1, s_1, h_1 + h') \models \psi_2$. Let us prove $(G_2, s_2, h_2) \models \tau_{X,Y}(\varphi_1 \multimap \varphi_2)$. By definition of $\tau_{X,Y}(\varphi_1 \multimap \varphi_2)$, this holds whenever for all heaps $h'_2 : G_2 \rightarrow_{fin} G_2$ satisfying the following properties

$$H1. h'_2 \perp h_2,$$

$$H2. (G_2, s_2, h'_2) \models \text{Store}_Y(\bar{V}),$$

$$H3. (G_2, s_2, h'_2) \models \tau_{\bar{V}, Y}(\psi_1[x \leftarrow \bar{x} \mid x \in Y]),$$

$$H4. (G_2, s_2, h_2 + h'_2) \models \bigwedge_{\bar{x} \in \bar{V}} n(x) = n(\bar{x}),$$

it holds that

$$T. (G_2, s_2, h_2 + h'_2) \models (\text{size} = \text{card}(\bar{V}) \wedge \bigwedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _) * \tau_{X,Y}(\psi_2).$$

Therefore, let h'_2 be some heap that satisfies the premises of the implication, i.e. $(H1)$ – $(H4)$. We show that (T) holds. By $(H2)$, for every variable $x \in Y$, $s_2(x) \notin \text{ran}(h'_2)$ and moreover $s_2(x) \in \text{dom}(h'_2)$ if and only if $x \in \bar{V}$. This implies that there is a heap h'_1 such that $h'_2 = h'_1 + \{s_2(\bar{x}) \mapsto h'_2(s_2(\bar{x})) \mid \bar{x} \in \bar{V}\}$ and for every $(\ell, \ell') \in h'_1$, both ℓ and ℓ' are not assigned to variables in Y . Moreover, from (\triangleright_1) we have $G_2 = G_1 \cup \{s_2(x) \mid x \in Y\}$, which in turn implies that the locations ℓ and ℓ' above are in G_1 . Thus, h'_1 is a heap in $[G_1 \rightarrow_{fin} G_1]$. This allows us to derive that

$$(G_1, s_1[\bar{x} \leftarrow h'_2(s_2(\bar{x})) \mid \bar{x} \in \bar{V}], h'_1) \triangleright_{\bar{Y}} (G_2, s_2, h'_2).$$

We can then apply the induction hypothesis and, from $(H3)$, conclude that

$$(G_1, s_1[\bar{x} \leftarrow h'_2(s_2(\bar{x})) \mid \bar{x} \in \bar{V}], h'_1) \models \psi_1[x \leftarrow \bar{x} \mid x \in Y]. \quad (a)$$

From $(H4)$ and by definition of h'_2 , we have $h'_2(s_2(\bar{x})) = h_2(s_2(x))$ for every $\bar{x} \in \bar{V}$. This means that (a) can be rewritten as $(G_1, s_1[\bar{x} \leftarrow h_2(s_2(x)) \mid \bar{x} \in \bar{V}], h'_1) \models \psi_1[x \leftarrow \bar{x} \mid x \in Y]$. As ψ_1 is written with variables from Z , we can apply (Sub) to swap every $\bar{x} \in \bar{Z}$ with $x \in Z$, and derive that

$$(G_1, (s_1[\bar{x} \leftarrow h_2(s_2(x)) \mid \bar{x} \in \bar{V}])[x \leftarrow s_1(\bar{x}) \mid x \in Z], h'_1) \models \psi_1. \quad (b)$$

The store $(s_1[\bar{x} \leftarrow h_2(s_2(x)) \mid \bar{x} \in \bar{V}])[x \leftarrow s_1(\bar{x}) \mid x \in Z]$ considered in (b) is such that for every $\bar{x} \in \bar{V}$, $h_2(s_2(x))$ is assigned to x . From (\triangleright_3) , the same holds for s_1 . Here, recall in particular that \bar{V} is the set of copies of the variables in $\text{fv}(\psi_1) \subseteq X$. We derive that the following g-fv(ψ_1)-isomorphism holds:

$$(G_1, (s_1[\bar{x} \leftarrow h_2(s_2(x)) \mid \bar{x} \in \bar{V}])[x \leftarrow s_1(\bar{x}) \mid x \in Z], h'_1) \simeq_{\text{fv}(\psi_1)}^g (G_1, s_1, h'_1)$$

which allows us to conclude $(G_1, s_1, h'_1) \models \psi_1$ directly by Lemma 3.3. Moreover, the following inclusions allow us to conclude that $h'_1 \perp h_1$:

$$h'_1 \subseteq h'_2, \quad (\text{by def. of } h'_1)$$

$$h_1 \subseteq h_2, \quad (\text{by } (G_1, s_1, h_1) \triangleright_{\bar{Y}}^X (G_2, s_2, h_2))$$

$$h_2 \cap h'_2 = \emptyset. \quad (\text{by } (H1))$$

Thus, by $(G_1, s_1, h_1) \models \psi_1 \multimap \psi_2$, $(G_1, s_1, h_1 + h'_1) \models \psi_2$ holds. By (\triangleright_3) and $h'_1 \in [G_1 \rightarrow_{fin} G_1]$, we have $h'_1 \perp h_2$ which, by (\triangleright_1) and (\triangleright_2) , implies that $(G_1, s_1, h_1 + h'_1) \triangleright_{\bar{Y}}^X (G_2, s_2, h_2 + h'_1)$.

By induction hypothesis, $(G_2, s_2, h_2 + h'_1) \models \tau_{X,Y}(\psi_2)$. Let us now consider the generalised memory state $(G_2, s_2, h_2 + h'_2)$, for which we need to prove (T). By definition of h'_2 ,

$$h_2 + h'_2 = h + h'_1 + \{s_2(\bar{x}) \mapsto h'_2(s_2(\bar{x})) \mid \bar{x} \in \bar{V}\}. \quad (\text{c})$$

As (\triangleright_2) implies that for every two distinct variables $x, y \in \bar{V}$, $s_2(x) \neq s_2(y)$, and moreover by (H2) for every variable $x \in \bar{V}$, the location $s_2(x)$ is in $\text{dom}(h'_2)$, we have

$$(G_2, s_2, \{s_2(\bar{x}) \mapsto h'_2(s_2(\bar{x})) \mid \bar{x} \in \bar{V}\}) \models \text{size} = \text{card}(\bar{V}) \wedge \bigwedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _.$$

Together with $(G_2, s_2, h_2 + h'_1) \models \tau_{X,Y}(\psi_2)$ and (c), this implies (T).

(\Leftarrow): Suppose $(G_2, s_2, h_2) \models \tau_{X,Y}(\varphi_1 * \varphi_2)$. This means that for every heap $h'_2 : G_2 \rightarrow_{\text{fin}} G_2$, if the following properties are satisfied

- | | |
|--|---|
| A1. $h'_2 \perp h_2$, | A3. $(G_2, s_2, h'_2) \models \tau_{\bar{Y}, Y}(\psi_1[x \leftarrow \bar{x} \mid x \in \bar{Y}])$, |
| A2. $(G_2, s_2, h'_2) \models \text{Store}_Y(\bar{V})$, | A4. $(G_2, s_2, h_2 + h'_2) \models \bigwedge_{\bar{x} \in \bar{V}} n(x) = n(\bar{x})$, |

then so is

$$\text{C. } (G_2, s_2, h_2 + h'_2) \models (\text{size} = \text{card}(\bar{V}) \wedge \bigwedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _) * \tau_{X,Y}(\psi_2).$$

Let us prove that $(G_1, s_1, h_1) \models \psi_1 * \psi_2$, which by definition means that for every heap $h'_1 : G_1 \rightarrow_{\text{fin}} G_1$, if $h'_1 \perp h_1$ and $(G_1, s_1, h'_1) \models \psi_1$, then $(G_1, s_1, h_1 + h'_1) \models \psi_2$. To prove it, let us consider a heap $h'_1 : G_1 \rightarrow_{\text{fin}} G_1$ disjoint from h_1 and such that $(G_1, s_1, h'_1) \models \psi_1$. We show that $(G_1, s_1, h_1 + h'_1) \models \psi_2$. Since ψ_1 is written with variables from Z (as it is a subformula of φ), we can apply (Sub) and consider $(G_1, s_1[\bar{x} \leftarrow s_1(x) \mid x \in Z], h'_1) \models \psi_1[x \leftarrow \bar{x} \mid x \in Z]$ instead of $(G_1, s_1, h'_1) \models \psi_1$. Notice that, again due to the fact that ψ_1 is written with variables from Z , the formula $\psi_1[x \leftarrow \bar{x} \mid x \in Z]$ is syntactically equivalent to $\psi_1[x \leftarrow \bar{x} \mid x \in \bar{Y}]$. Let us consider the heap $h'_2 = h'_1 + \{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\}$. As $\bar{V} \subseteq \bar{Z}$, we can show the following relation:

$$(G_1, s_1[\bar{x} \leftarrow s_1(x) \mid x \in Z], h'_1) \triangleright_{\bar{Y}}^{\bar{V}} (G_2, s_2, h'_2). \quad (\text{a})$$

In particular, recall that this means that:

- $G_1 = G_2 \setminus \{s_2(x) \mid x \in \bar{Y}\}$,
- given $x, y \in \bar{Y}$, if $x \neq y$ then $s_2(x) \neq s_2(y)$,
- $h_2 = h_1 + \{s_2(\bar{x}) \mapsto s_1[\bar{x} \leftarrow s_1(x) \mid x \in Z](\bar{x}) \mid \bar{x} \in \bar{V}\}$.

The first two properties are direct from (\triangleright_1) and (\triangleright_2). For the third property, we notice that for every $\bar{x} \in \bar{V}$, $s_1[\bar{x} \leftarrow s_1(x) \mid x \in Z](\bar{x}) = s_1(x)$, which implies that

$$\{s_2(\bar{x}) \mapsto s_1[\bar{x} \leftarrow s_1(x) \mid x \in Z](\bar{x}) \mid \bar{x} \in \bar{V}\} = \{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\}.$$

Then, the property follows directly by definition of h'_2 . From (a), we deduce that (A3) holds by induction hypothesis, whereas (A2) is satisfied by Lemma 3.9. Moreover, (A1) follows directly from the inclusions below:

$$\begin{aligned} \text{dom}(h_2) &= \text{dom}(h_1) \cup \text{dom}(\{s_2(x) \mapsto s_1(x) \mid x \in X\}), && \text{(by } (\triangleright_3) \text{)} \\ \text{dom}(h'_2) &= \text{dom}(h'_1) \cup \text{dom}(\{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\}), && \text{(by def. of } h'_2 \text{)} \\ \text{dom}(h_1) \cap \text{dom}(h'_1) &= \emptyset, && \text{(by def. of } h'_1 \text{)} \\ \text{dom}(h_1) \cup \text{dom}(h'_1) &\subseteq G_1, && \text{(by def. of } h_1, h'_1 \text{)} \\ \text{dom}(\{s_2(x) \mapsto s_1(x) \mid x \in X\}) \cap \text{dom}(\{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\}) &= \emptyset, && \text{(by } (\triangleright_2) \text{)} \\ \text{dom}(\{s_2(x) \mapsto s_1(x) \mid x \in X\}) \cup \text{dom}(\{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\}) &\subseteq G_2 \setminus G_1. && \text{(by } (\triangleright_1) \text{)} \end{aligned}$$

Therefore, $h_2 + h'_2$ is defined. The definition of h_2 and h'_2 implies the following inclusion

$$\{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\} + \{s_2(x) \mapsto s_1(x) \mid x \in X\} \subseteq h_2 + h'_2.$$

Then, for every $\bar{x} \in \bar{V}$, $(h_2 + h'_2)(s(x)) = (h_2 + h'_2)(s(\bar{x}))$, which in turn implies (A4). (A1)–(A4) imply (C), and therefore there are two disjoint heaps h_L and h_R such that

1. $h_L + h_R = h_2 + h'_2 \stackrel{\text{by def}}{=} h_1 + h'_1 + \{s_2(x) \mapsto s_1(x) \mid x \in X\} + \{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\}$,
2. $(G_2, s_2, h_L) \models \mathbf{size} = \mathbf{card}(\bar{V}) \wedge \bigwedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow \dots$,
3. $(G_2, s_2, h_R) \models \tau_{X, Y}(\psi_2)$.

Since, by (D2), $\mathbf{card}(\{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\}) = \mathbf{card}(\bar{V})$, in order to satisfy (2), h_L must be $\{s_2(\bar{x}) \mapsto s_1(x) \mid \bar{x} \in \bar{V}\}$. By (1), this means that $h_R = h_1 + h'_1 + \{s_2(x) \mapsto s_1(x) \mid x \in X\}$. One can check that, by definition of $\triangleright_{\bar{Y}}^X$, $(G_1, s_1, h_1 + h'_1) \triangleright_{\bar{Y}}^X (G_2, s_2, h_R)$, which allows us to derive $(G_1, s_1, h_1 + h'_1) \models \psi_2$ by induction hypothesis from (3), concluding the proof. \square

3.2.2 $\mathbf{SL}(n(x), *, \neg*)$ is not recursively enumerable.

Now that we established Lemma 3.11, showing that the satisfiability and validity problem for $\mathbf{SL}(n(x), *, \neg*)$ are not RE comes almost effortlessly. Let us still consider the sets of program variables Y and \bar{Y} defined earlier. Given a formula φ in $\mathbf{SL}(\exists, \neg*)$, written with variables from Y , we write $\mathcal{T}_{\text{SAT}}(\varphi)$ and $\mathcal{T}_{\text{VAL}}(\varphi)$ for the two following formulae in $\mathbf{SL}(n(x), *, \neg*)$:

$$\mathcal{T}_{\text{SAT}}(\varphi) \stackrel{\text{def}}{=} \mathbf{Store}_{\bar{Y}}(\mathbf{fv}(\varphi)) \wedge \tau_{\mathbf{fv}(\varphi), \bar{Y}}(\varphi), \quad \mathcal{T}_{\text{VAL}}(\varphi) \stackrel{\text{def}}{=} \mathbf{Store}_{\bar{Y}}(\mathbf{fv}(\varphi)) \Rightarrow \tau_{\mathbf{fv}(\varphi), \bar{Y}}(\varphi).$$

As a consequence of Lemmata 3.9 and 3.11, φ and $\mathcal{T}_{\text{SAT}}(\varphi)$ are shown equisatisfiable, whereas φ and $\mathcal{T}_{\text{VAL}}(\varphi)$ are shown equivalent.

Corollary 3.12. (I) φ and $\mathcal{T}_{\text{SAT}}(\varphi)$ are equisatisfiable. (II) φ and $\mathcal{T}_{\text{VAL}}(\varphi)$ are equivalent.

The proofs of (I) and (II) are very similar. Below, we just show the proof of (I).

Proof of (I). First, suppose that φ is satisfiable, and let (G_1, s_1, h_1) be a generalised memory state satisfying φ . Let us define a set of locations $G_2 = G_1 \cup \{\ell_x \mid x \in Y\}$, where $\{\ell_x \mid x \in Y\}$ is a set of $\mathbf{card}(Y)$ locations not in G_1 . Let us also consider a store $s_2 : \mathbf{VAR} \rightarrow G_2$ such that, for every $x \in Y$, $s_2(x) = \ell_x$, and let $h_2 : G_2 \rightarrow G_2$ be the heap $h_2 \stackrel{\text{def}}{=} h_1 + \{s_2(x) \mapsto s_1(x) \mid x \in \mathbf{fv}(\varphi)\}$. A quick check to the conditions required by the encoded-by relation reveals that

$$(G_1, s_1, h_1) \triangleright_{\bar{Y}}^{\mathbf{fv}(\varphi)} (G_2, s_2, h_2).$$

By Lemmata 3.9 and 3.11 we conclude that $(G_2, s_2, h_2) \models \mathcal{T}_{\text{SAT}}(\varphi)$.

Conversely, suppose that there is a generalised memory state $(G_2, s_2, h_2) \models \mathcal{T}_{\text{SAT}}(\varphi)$. As (G_2, s_2, h_2) satisfies $\mathbf{Store}_{\bar{Y}}(\mathbf{fv}(\varphi))$, by Lemma 3.9 we derive that $(G_1, s_1, h_1) \triangleright_{\bar{Y}}^{\mathbf{fv}(\varphi)} (G_2, s_2, h_2)$ holds for some generalised memory state (G_1, s_1, h_1) . From Lemma 3.11, $(G_1, s_1, h_1) \models \varphi$. \square

As stated at the beginning of the section, the satisfiability and validity problems for $\mathbf{SL}(\exists, \neg*)$ are both non RE. Corollary 3.12 shows that this result carries over to $\mathbf{SL}(n(x), *, \neg*)$, thus proving Theorem 3.5. In the next section we show how to translate $\mathbf{SL}(n(x), *, \neg*)$ into separation logics with classical reachability predicates, transferring this result even further.

3.3 REACHABILITY PREDICATES CAN QUANTIFY

We come back to our original goal of simulating the first-order quantification by using reachability predicates. During Section 2.1.1 we defined three standard reachability predicates: the reach-plus predicate \hookrightarrow^+ , the reach-star predicate \hookrightarrow^* and the list-segment predicate ls . We showed that, as soon as we consider a separation logic featuring Boolean connectives, emp and the separating conjunction, \hookrightarrow^+ can be used to define the other two interdefinable predicates \hookrightarrow^* and ls . This means that the logic $\text{SL}(*, -, \hookrightarrow^+)$ obtained from the quantifier-free separation logic $\text{SL}(*, -)$ by adding the reach-plus predicate captures $\text{SL}(*, -, \hookrightarrow^*)$, which in turn is equivalent to $\text{SL}(*, -, \text{ls})$. Thanks to $\text{SL}(\text{n}(x), *, -)$, we are now able to show that a non-trivial restriction of $\text{SL}(*, -, \text{ls})$ already admits non RE satisfiability and validity problems. In this section we use the standard memory states of separation logic, knowing that (s, h) corresponds to the generalised memory state (LOC, s, h) , and that by Lemma 3.3, the distinction between generalised and standard memory states does not change the notion of satisfiability and validity.

3.3.1 Bounded reachability.

Given a memory state (s, h) and $\delta \in \mathbb{N}$, we introduce the *bounded reachability predicate* $x \hookrightarrow^\delta y$, with the following intended semantics:

$$(s, h) \models x \hookrightarrow^\delta y \quad \text{if and only if} \quad h^\delta(s(x)) = s(y) \text{ and for every } \delta' \in [0, \delta - 1], h^{\delta'}(s(x)) \neq s(y).$$

To define this formula, we first introduce the auxiliary formula $[\varphi]_\beta$ defined as $(\text{size} = \beta \wedge \varphi) * \top$. Clearly, $(s, h) \models [\varphi]_\beta$ holds whenever there is a heap $h' \subseteq h$ such that $\text{card}(h') = \beta$ and $(s, h') \models \varphi$. Then, the predicate $x \hookrightarrow^\delta y$ can be simply defined as $[\text{ls}(x, y)]_\delta$. We leave the proof of correctness of this definition to the reader. Since \hookrightarrow^δ is defined in terms of ls , it can be defined in the three separation logics $\text{SL}(*, -, \hookrightarrow^+)$, $\text{SL}(*, -, \hookrightarrow^*)$ and $\text{SL}(*, -, \text{ls})$.

We prove that the restriction of $\text{SL}(*, -, \text{ls})$ to the two predicates \hookrightarrow^2 and \hookrightarrow^3 , i.e. bounded reachability of depth two and three, already admits non RE satisfiability and validity problems. Precisely, we look at the logic $\text{SL}(*, -, \hookrightarrow^2, \hookrightarrow^3)$ whose formulae φ are from the grammar:

$$\varphi := \top \mid \text{emp} \mid x = y \mid x \hookrightarrow y \mid x \hookrightarrow^2 y \mid x \hookrightarrow^3 y \mid \varphi \wedge \varphi \mid \neg \varphi \mid \varphi * \varphi \mid \varphi -* \varphi.$$

Modulo some technical details, the result is shown by simulating the predicates $\hookrightarrow x$, $\text{n}(x) = \text{n}(y)$ and $\text{n}(x) \hookrightarrow \text{n}(y)$ in $\text{SL}(*, -, \hookrightarrow^2, \hookrightarrow^3)$, and relying on Corollary 3.12. This means that the proof mainly consists in defining these predicates in $\text{SL}(*, -, \hookrightarrow^2, \hookrightarrow^3)$ and then simply check for the correctness of the definitions. In order to avoid a repetitive list of definitions and proofs, in this section we simply introduce the formulae and informally explain the idea behind their definition. Their correctness is formally proven in Appendix A.

Alloc-back. In Section 2.1.1 we showed how to use the reachability predicate $x \hookrightarrow^+ x$ in order to express the predicate $\hookrightarrow x$ (see Proposition 2.7). We would like to change this definition so that it uses $x \hookrightarrow^\delta x$, with $\delta \in \{2, 3\}$, instead of $x \hookrightarrow^+ x$. This is unfortunately not possible, as by definition $x \hookrightarrow^\delta x$ is equivalent to \perp for every $\delta \geq 1$. To get around this problem, we need to take advantage of an auxiliary variable y whose location is different from the one assigned to x . We introduce the formula $\text{alloc}_y^{-1}(x)$, defined as follows

$$\text{alloc}_y^{-1}(x) \stackrel{\text{def}}{=} x \hookrightarrow x \vee y \hookrightarrow x \vee (\top * (y \hookrightarrow _ \wedge \text{size} = 1 \rightsquigarrow y \hookrightarrow^2 x)).$$

where we notice that $y \hookrightarrow _ \stackrel{\text{def}}{=} y \hookrightarrow y \dashv \perp$ is in $\mathbf{SL}(*, _, \hookrightarrow^2, \hookrightarrow^3)$. The formula $\text{alloc}_y^{-1}(x)$ is equivalent to $_ \hookrightarrow x$ under the hypothesis that x and y correspond to different locations. The idea behind its definition is quite simple. The first two disjuncts of the formula take care of the cases where the location that points to $s(x)$ is either $s(x)$ itself or $s(y)$. Every other case is taken care of by the third disjunct, which states that it is possible to find a subheap $h' \subseteq h$ and a location ℓ such that $h' + \{s(y) \mapsto \ell\}$ witness a path of length two going from $s(y)$ to $s(x)$. Clearly, this means that $\{\ell \mapsto s(x)\} \subseteq h'$, leading to the correctness of the formula.

Lemma 3.13. Let (s, h) be a memory state such that $s(x) \neq s(y)$. We have,

$$(s, h) \models \text{alloc}_y^{-1}(x) \text{ if and only if } s(x) \in \text{ran}(h).$$

The additional hypothesis $s(x) \neq s(y)$ does not influence our results, as the formulae $\mathcal{T}_{\text{SAT}}(\varphi)$ and $\mathcal{T}_{\text{VAL}}(\varphi)$ used in Corollary 3.12 keep their satisfiability and validity status when $_ \hookrightarrow x$ is replaced with $\text{alloc}_{\bar{x}}^{-1}(x)$. In particular, one can notice that the alloc-back predicate appears in these two formulae only inside a subformula $\text{Store}_{\mathbb{Y}}(X)$, which forces x and \bar{x} to correspond to different locations. We explicit this property by defining the formula of $\mathbf{SL}(*, _, \hookrightarrow^2, \hookrightarrow^3)$:

$$\text{Store}_{\mathbb{Y}}^*(X) \stackrel{\text{def}}{=} \text{Store}_{\mathbb{Y}}(X)[_ \hookrightarrow x \leftarrow \text{alloc}_{\bar{x}}^{-1}(x) \mid x \in \mathbb{Y}],$$

Lemma 3.14. $(s, h) \models \text{Store}_{\mathbb{Y}}(X)$ if and only if $(s, h) \models \text{Store}_{\mathbb{Y}}^*(X)$.

Proof. Directly from Lemma 3.13 and the fact that both $\text{Store}_{\mathbb{Y}}(X)$ and $\text{Store}_{\mathbb{Y}}^*(X)$ are conjunctive formulae witnessing one conjunct $x \neq \bar{x}$ for every $x \in \mathbb{Y}$. \square

Next-equality. Let us now move to the predicate $n(x) = n(y)$, which we recall being satisfied by a memory state (s, h) whenever $h(s(x)) = h(s(y))$ holds. As done for the predicate $_ \hookrightarrow x$, we define a formula that respects this semantics only under additional hypothesis. In particular, we assume $s(x)$ and $s(y)$ to be locations that do not belong to $\text{ran}(h)$. This formula is denoted by $\text{next}(x = y)$ and it is defined as:

$$\text{next}(x = y) \stackrel{\text{def}}{=} x \hookrightarrow _ \wedge \left(x \neq y \Rightarrow \left[x \hookrightarrow _ \wedge y \hookrightarrow _ \wedge \neg(\top \oplus x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x) \right]_2 \right).$$

Let (s, h) be a memory state. The first conjunct of $\text{next}(x = y)$ handles the case where $s(x) = s(y)$, as $n(x) = n(y)$ then becomes equivalent to $x \hookrightarrow _$ (or analogously, $y \hookrightarrow _$). The second conjunct considers the case $s(x) \neq s(y)$. It states that it is possible to find a subheap $h' \subseteq h$ of cardinality exactly two and where both $s(x)$ and $s(y)$ are memory cells. Under the hypothesis that $s(x)$ and $s(y)$ are not in $\text{ran}(h)$, this means that h' witness one of the two following shapes:



The subformula $\neg(\top \oplus x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x)$ of $\text{next}(x = y)$ excludes the leftmost memory state while allowing the rightmost one, i.e. the one satisfying $n(x) = n(y)$. Indeed, this formula states that it is not possible to find a heap h'' such that the union $h' + h''$ witnesses two paths of exactly length two, one going from $s(x)$ to $s(y)$ and one going from $s(y)$ to $s(x)$. By considering the

heap $h'' = \{\ell_1 \mapsto s(y), \ell_2 \mapsto s(x)\}$ we see that this property does not hold for the leftmost memory state. For the rightmost memory state, in order to create a path of length two from $s(x)$ to $s(y)$ we are obliged to consider a heap h'' such that $\{\ell \mapsto s(y)\} \subseteq h''$. Therefore, it is impossible to create a path of length 2 going from $s(y)$ to $s(x)$, as $h' + h''$ contains the cycle $\{s(y) \mapsto \ell \mapsto s(y)\}$. We conclude that the rightmost memory state satisfies $\neg(\top \multimap x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x)$. This leads to the following correctness result.

Lemma 3.15. Let (s, h) be a memory state such that $\{s(x), s(y)\} \cap \text{ran}(h) = \emptyset$.

$$(s, h) \models \text{next}(x = y) \text{ if and only if } h(s(x)) = h(s(y)).$$

As it holds for the formula $\text{alloc}_y^{-1}(x)$, the additional hypothesis $\{s(x), s(y)\} \cap \text{ran}(h) = \emptyset$ of Lemma 3.15 does not influence our result, and $n(x) = n(y)$ can be safely replaced by $\text{next}(x = y)$ in both the formulae $\mathcal{T}_{\text{SAT}}(\varphi)$ and $\mathcal{T}_{\text{VAL}}(\varphi)$. Again, this replacement does not change the satisfiability and validity of these formulae, since $\text{Store}_{\mathbb{Y}}^*(X)$ implies that all variables in \mathbb{Y} do not belong to the range of the heap. We formalise this result in Section 3.3.2

Next-points-to. Lastly, we consider the predicate $n(x) \hookrightarrow n(y)$, which we recall being satisfied by a memory state (s, h) whenever $h(h(s(x))) = h(s(y))$ holds. Again, we define this predicate in $\text{SL}(*, _, \hookrightarrow^2, \hookrightarrow^3)$ modulo some additional hypothesis. As in the case of $\text{alloc}_y^{-1}(x)$, we rely on an auxiliary variable z whose assigned location is assumed to be different from $s(x)$ and $s(y)$. Similarly to $\text{next}(x = y)$, we also require $s(x)$, $s(y)$ and $s(z)$ not to belong to the range of h . Under these conditions, the equivalent formula $\text{next}_z(x \hookrightarrow y)$ is defined as follows:

$$\text{next}_z(x \hookrightarrow y) \stackrel{\text{def}}{=} (\text{next}(x = y) \wedge [x \hookrightarrow _ \wedge \neg(\top \multimap x \hookrightarrow^3 z)]_2) \vee [\text{size} = 1 \multimap x \hookrightarrow^3 z \wedge y \hookrightarrow^2 z]_3.$$

This formula is somewhat more involved than $\text{alloc}_y^{-1}(x)$ and $\text{next}(x = y)$. Its definition is split into two disjuncts: the left one capturing the cases where $n(x) = n(y)$ holds, and the right one capturing the cases where $n(x) = n(y)$ does not hold. Let us consider a memory state (s, h) . Under the condition that $(s, h) \models n(x) = n(y)$, we notice that $h(h(s(x))) = h(s(y))$ holds if and only if there is a location ℓ such that $\{s(x) \mapsto \ell, \ell \mapsto \ell\} \subseteq h$. In particular, the location $h(s(x))$ points to itself. The subformula $[x \hookrightarrow _ \wedge \neg(\top \multimap x \hookrightarrow^3 z)]_2$ exactly checks for this pattern. It states that it is possible to find a subheap $h' \subseteq h$ made of two memory cells, one of which is $s(x)$, that cannot be extended so that it contains a path of length exactly three, that goes from $s(x)$ to $s(z)$. One can check this extension cannot be performed only in the case where the heap h' has a cycle that can be reached from $s(x)$. As however $s(x) \notin \text{ran}(h)$, the only possibility left is that $h'(s(x))$ points to itself. This leads to $h' = \{s(x) \mapsto \ell, \ell \mapsto \ell\}$.

The second disjunct of $\text{next}_z(x \hookrightarrow y)$ states that there is a subheap $h_1 \subseteq h$ of exactly three memory cells that can be extended with a heap h_2 such that $\text{card}(h_2) = 1$ and $h_1 + h_2$ witnesses the following two paths:

$$\{s(x) \mapsto \ell_1^x \mapsto \ell_2^x \mapsto s(z)\}, \quad \{s(y) \mapsto \ell_1^y \mapsto s(z)\}.$$

As $s(z) \notin \text{ran}(h)$ and $h_1 \subseteq h_2$, it can only be the case that $h_2 = \{\ell_1^y \mapsto s(z)\}$ and $\ell_2^x = \ell_1^y$, which leads to $h(h(s(x))) = h(s(y))$. The formula $\text{next}_z(x \hookrightarrow y)$ is then found to be correct.

Lemma 3.16. Let (s, h) be such that $s(x) \neq s(z) \neq s(y)$ and $\{s(x), s(y), s(z)\} \cap \text{ran}(h) = \emptyset$.

$$(s, h) \models \text{next}_z(x \hookrightarrow y) \text{ if and only if } h(h(s(x))) = h(s(y)).$$

Again, the additional hypothesis needed for the correctness of the formula $\text{next}_z(x \hookrightarrow y)$ does not influence our result, as we now show.

3.3.2 Using $\text{SL}(n(x), *, \neg*)$ to prove that $\text{SL}(*, \neg*, \hookrightarrow^2, \hookrightarrow^3)$ is not RE.

We now show that $\text{SL}(*, \neg*, \hookrightarrow^2, \hookrightarrow^3)$ is not RE by relying on the translation from $\text{SL}(\exists, \neg*)$ to $\text{SL}(n(x), *, \neg*)$ carried out in Section 3.2.1. As defined in that section, given a finite set of variables $Y \subseteq_{\text{fin}} \text{VAR}$, we consider a copy \bar{x} for every variable $x \in Y$. We denote with $\bar{Y} \subseteq_{\text{fin}} \text{VAR}$ the set of these copies, so that $\text{card}(\bar{Y}) = \text{card}(Y)$ and $\bar{Y} \cap Y = \emptyset$. Then, we denote by \bar{Y} the set $Y \cup \bar{Y}$, and consider an involution $(\cdot) : \bar{Y} \rightarrow \bar{Y}$ associating every $x \in Y$ with its copy $\bar{x} \in \bar{Y}$. In Section 3.2.1, we have shown that we can translate a given formula φ in $\text{SL}(\exists, \neg*)$, written with variables in Y , into an equisatisfiable formula $\mathcal{T}_{\text{SAT}}(\varphi)$ and an equivalent formula $\mathcal{T}_{\text{VAL}}(\varphi)$, both in $\text{SL}(n(x), *, \neg*)$. The main ingredient of these formulae is given by the translation $\tau_{X,Y}(\varphi)$, which uses the multiplicative connectives of $\text{SL}(n(x), *, \neg*)$ in order to simulate the first-order quantification of $\text{SL}(\exists, \neg*)$. We now modify this translation in order to produce an equivalent formula in $\text{SL}(*, \neg*, \hookrightarrow^2, \hookrightarrow^3)$. In particular, it is sufficient to replace every occurrence of the predicates $_ \hookrightarrow x$, $n(x) = n(y)$ and $n(x) \hookrightarrow n(y)$ in $\text{SL}(n(x), *, \neg*)$ with the formulae $\text{alloc}_{\bar{x}}^{-1}(x)$, $\text{next}(x = y)$ and $\text{next}_{\bar{x}}(x \hookrightarrow y)$, respectively, obtaining the following formula:

$$\begin{aligned}\tau_{X,Y}^*(\varphi) &\stackrel{\text{def}}{=} \tau_{X,Y}(\varphi)[_ \hookrightarrow x \leftarrow \text{alloc}_{\bar{x}}^{-1}(x) \mid x \in \bar{Y}] \\ &\quad [n(x) = n(y) \leftarrow \text{next}(x = y) \mid x, y \in \bar{Y}] \\ &\quad [n(x) \hookrightarrow n(y) \leftarrow \text{next}_{\bar{x}}(x \hookrightarrow y) \mid x, y \in \bar{Y}].\end{aligned}$$

One can check that $\tau_{X,Y}^*(\varphi)$ is indeed a formula of $\text{SL}(*, \neg*, \hookrightarrow^2, \hookrightarrow^3)$. Thanks to the lemmata shown in the previous section (from Lemma 3.13 to Lemma 3.16), we can prove that $\tau_{X,Y}^*(\varphi)$ is equivalent to $\tau_{X,Y}(\varphi)$. We recall that $\text{Store}_{\bar{Y}}^*(X) = \text{Store}_{\bar{Y}}(X)[_ \hookrightarrow x \leftarrow \text{alloc}_{\bar{x}}^{-1}(x) \mid x \in \bar{Y}]$.

Lemma 3.17. Let Z be either Y or \bar{Y} and let $X \subseteq Z$. Let φ be a formula in $\text{SL}(\exists, \neg*)$ s.t. $\text{fv}(\varphi) \subseteq X$ and $\text{bv}(\varphi) \subseteq Z \setminus X$. Given (s, h) satisfying $\text{Store}_{\bar{Y}}(X)$, $(s, h) \models \tau_{X,Y}(\varphi)$ iff $(s, h) \models \tau_{X,Y}^*(\varphi)$.

Proof. From $(s, h) \models \text{Store}_{\bar{Y}}(X)$, the memory state (s, h) satisfies the following properties:

- | | |
|--|--|
| 1. for all distinct $x, y \in \bar{Y}$, $s(x) \neq s(y)$, | 3. for every $x \in X$, $s(x) \in \text{dom}(h)$, |
| 2. for every $x \in \bar{Y}$, $s(x) \notin \text{ran}(h)$, | 4. for every $x \in \bar{Y} \setminus X$, $s(x) \notin \text{dom}(h)$. |

Below, the indices (1), (2), (3), (4) refer to these four properties.

As done for Lemma 3.11, the proof is by structural induction on φ . The base cases for $\varphi = x = y$ and $\varphi = x \hookrightarrow y$ holds directly from Lemmata 3.15 and 3.16, respectively, which can be applied thanks to (1) and (2). We omit the obvious cases for the Boolean connectives, and focus on the cases where $\varphi = \exists z \psi$ or $\varphi = \psi_1 \neg* \psi_2$.

case: $\varphi = \exists z \psi$. Let us make explicit the definitions of $\tau_{X,Y}(\varphi)$ and $\tau_{X,Y}^*(\varphi)$:

$$\begin{aligned}\tau_{X,Y}(\exists z \psi) &= (z \hookrightarrow _ \wedge \text{size} = 1) \multimap (\text{Store}_{\bar{Y}}(X \cup \{z\}) \wedge \tau_{X \cup \{z\}, Y}(\psi)), \\ \tau_{X,Y}^*(\exists z \psi) &= (\text{next}(z = z) \wedge \text{size} = 1) \multimap (\text{Store}_{\bar{Y}}^*(X \cup \{z\}) \wedge \tau_{X \cup \{z\}, Y}^*(\psi)).\end{aligned}$$

where we recall that $z \hookrightarrow _$ is defined as $n(z) = n(z)$ in $\text{SL}(n(x), *, \neg*)$.

(\Rightarrow): Suppose $(s, h) \models \tau_{X,Y}(\exists z \psi)$. By definition of the subtraction operator \multimap , there is a location ℓ and a heap $h' = \{s(z) \mapsto \ell\}$ such that $(s, h + h')$ satisfies both $\text{Store}_{\bar{Y}}(X \cup \{z\})$ and $\tau_{X \cup \{z\}, Y}(\psi)$. Thanks to $(s, h + h') \models \text{Store}_{\bar{Y}}(X \cup \{z\})$ we have:

- $s(z) \notin \text{ran}(h + h')$, which implies $s(z) \notin \text{ran}(h')$. Directly from Lemma 3.15 and the definition of h' we conclude that $(s, h') \models \text{next}(z = z) \wedge \text{size} = 1$ holds.
- By Lemma 3.14, $(s, h + \{s(z) \mapsto \ell\}) \models \text{Store}_{\mathbb{Y}}^*(X \cup \{z\})$. Then, by induction hypothesis we derive $(s, h + \{s(z) \mapsto \ell\}) \models \tau_{X \cup \{z\}, \mathbb{Y}}^*(\psi)$.

Again by definition of the subtraction operator, we conclude: $(s, h) \models \tau_{X, \mathbb{Y}}^*(\exists z \psi)$.

(\Leftarrow): Analogous to the other direction.

case: $\varphi = \psi_1 \multimap \psi_2$. Let $\bar{V} \stackrel{\text{def}}{=} \{\bar{x} \mid x \in \text{fv}(\psi_1)\}$. We recall the definitions of $\tau_{X, \mathbb{Y}}(\varphi)$ and $\tau_{X, \mathbb{Y}}^*(\varphi)$:

$$\begin{aligned}\tau_{X, \mathbb{Y}}(\psi_1 \multimap \psi_2) &\stackrel{\text{def}}{=} (\text{Store}_{\mathbb{Y}}(\bar{V}) \wedge \tau_{\bar{V}, \mathbb{Y}}(\psi_1[x \leftarrow \bar{x} \mid x \in \mathbb{Y}])) \multimap \\ &((\wedge_{\bar{x} \in \bar{V}} n(x) = n(\bar{x})) \Rightarrow (\text{size} = \text{card}(\bar{V}) \wedge \wedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _) * \tau_{X, \mathbb{Y}}(\psi_2)), \\ \tau_{X, \mathbb{Y}}^*(\psi_1 \multimap \psi_2) &\stackrel{\text{def}}{=} (\text{Store}_{\mathbb{Y}}^*(\bar{V}) \wedge \tau_{\bar{V}, \mathbb{Y}}^*(\psi_1[x \leftarrow \bar{x} \mid x \in \mathbb{Y}])) \multimap \\ &((\wedge_{\bar{x} \in \bar{V}} \text{next}(x = \bar{x})) \Rightarrow (\text{size} = \text{card}(\bar{V}) \wedge \wedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _) * \tau_{X, \mathbb{Y}}^*(\psi_2)).\end{aligned}$$

(\Rightarrow): Suppose $(s, h) \models \tau_{X, \mathbb{Y}}(\psi_1 \multimap \psi_2)$. In order to show that $(s, h) \models \tau_{X, \mathbb{Y}}^*(\psi_1 \multimap \psi_2)$, let us consider a heap h' disjoint from h and such that

$$\begin{aligned}H1. \quad (s, h') &\models \text{Store}_{\mathbb{Y}}^*(\bar{V}), & H3. \quad (s, h + h') &\models \wedge_{\bar{x} \in \bar{V}} \text{next}(x = \bar{x}), \\ H2. \quad (s, h') &\models \tau_{\bar{V}, \mathbb{Y}}^*(\psi_1[x \leftarrow \bar{x} \mid x \in \mathbb{Y}]),\end{aligned}$$

and prove that then it follows that

$$T. \quad (s, h + h') \models (\text{size} = \text{card}(\bar{V}) \wedge \wedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _) * \tau_{X, \mathbb{Y}}^*(\psi_2).$$

From (H1) and by Lemma 3.14, (s, h') satisfies $\text{Store}_{\mathbb{Y}}(\bar{V})$. Thanks to Section 3.3.2, we can then apply the induction hypothesis and conclude that $(s, h') \models \tau_{\bar{V}, \mathbb{Y}}(\psi_1[x \leftarrow \bar{x} \mid x \in \mathbb{Y}])$. Let us now consider $x \in \mathbb{Y}$. By (H1) we know that $s(x) \notin \text{ran}(h')$. Moreover, $s(x) \notin \text{ran}(h)$ also holds (by (2)), and therefore $s(x) \notin \text{ran}(h + h')$. Thus, by (H3) and Lemma 3.15, $(s, h + h') \models \wedge_{\bar{x} \in \bar{V}} n(x) = n(y)$. This allows us to derive, by $(s, h) \models \tau_{X, \mathbb{Y}}(\psi_1 \multimap \psi_2)$, that

$$(s, h + h') \models (\text{size} = \text{card}(\bar{V}) \wedge \wedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _) * \tau_{X, \mathbb{Y}}^*(\psi_2),$$

which means that there are two disjoint heaps h_1 and h_2 such that $h + h' = h_1 + h_2$, $(s, h_1) \models \text{size} = \text{card}(\bar{V}) \wedge \wedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _$ and $(s, h_2) \models \tau_{X, \mathbb{Y}}(\psi_2)$. To conclude the proof, it is sufficient to show that $(s, h_2) \models \text{Store}_{\mathbb{Y}}(X)$, so that we can apply the induction hypothesis to conclude $(s, h_2) \models \tau_{X, \mathbb{Y}}^*(\psi_2)$, which in turn shows (T). So, we show the four properties:

$$\begin{array}{ll} S1. \text{ for all distinct } x, y \in \mathbb{Y}, s(x) \neq s(y), & S3. \text{ for every } x \in X, s(x) \in \text{dom}(h_2), \\ S2. \text{ for every } x \in \mathbb{Y}, s(x) \notin \text{ran}(h_2), & S4. \text{ for every } x \in \mathbb{Y} \setminus X, s(x) \notin \text{dom}(h_2). \end{array}$$

The property (S1) holds directly from (1). For (S2) it is sufficient to recall that for every $x \in \mathbb{Y}$, $s(x) \notin \text{ran}(h + h')$, and $h_2 \subseteq h + h'$. In order to prove (S3) and (S4), we equivalently show that for every $x \in \mathbb{Y}$, $s(x) \in \text{dom}(h_2)$ if and only if $x \in X$. From the properties (3) and (4) of $\text{Store}_{\mathbb{Y}}(X)$, we derive that for every $x \in \mathbb{Y}$, $s(x) \in \text{dom}(h)$ if and only if $x \in X$. Similarly, from $(s, h') \models \text{Store}_{\mathbb{Y}}(\bar{V})$, given a variable $x \in \mathbb{Y}$, $s(x) \in \text{dom}(h')$ if and only if $x \in \bar{V}$. At the same time, from (1), $\text{card}(\{s(\bar{x}) \mid \bar{x} \in \bar{V}\}) = \text{card}(\bar{V})$. Together with $(s, h_1) \models \text{size} = \text{card}(\bar{V}) \wedge \wedge_{\bar{x} \in \bar{V}} \bar{x} \hookrightarrow _$, this implies that given a variable $x \in \mathbb{Y}$, $s(x) \in \text{dom}(h_1)$ if and only if $x \in \bar{V}$, exactly as in the case of h' . Since $h + h' = h_1 + h_2$, we conclude: for every $x \in \mathbb{Y}$, $s(x) \in \text{dom}(h_2)$ if and only if $x \in X$.

(\Leftarrow): Analogous to the other direction. \square

Let us now consider the formulae $\mathcal{T}_{\text{SAT}}(\varphi)$ and $\mathcal{T}_{\text{VAL}}(\varphi)$ of $\text{SL}(\mathbf{n}(x), *, \neg*)$ defined in section Section 3.2.2. We define analogous formulae in $\text{SL}(*, \neg*, \hookrightarrow^2, \hookrightarrow^3)$:

$$\mathcal{T}_{\text{SAT}}^*(\varphi) \stackrel{\text{def}}{=} \text{Store}_Y^*(\text{fv}(\varphi)) \wedge \tau_{\text{fv}(\varphi), Y}^*(\varphi), \quad \mathcal{T}_{\text{VAL}}^*(\varphi) \stackrel{\text{def}}{=} \text{Store}_Y^*(\text{fv}(\varphi)) \Rightarrow \tau_{\text{fv}(\varphi), Y}^*(\varphi).$$

Directly from Lemmata 3.14 and 3.17, $\mathcal{T}_{\text{SAT}}(\varphi)$ is equivalent to $\mathcal{T}_{\text{SAT}}^*(\varphi)$, whereas $\mathcal{T}_{\text{VAL}}(\varphi)$ is equivalent to $\mathcal{T}_{\text{VAL}}^*(\varphi)$. We can therefore lift Theorem 3.5 to $\text{SL}(*, \neg*, \hookrightarrow^2, \hookrightarrow^3)$.

Theorem 3.18. The satisfiability and validity problems of $\text{SL}(*, \neg*, \hookrightarrow^2, \hookrightarrow^3)$ are not RE.

3.3.3 Other separation logics with non RE decision problems.

Let us analyse Theorem 3.5 and Theorem 3.18 further. As already stated, it is known that the restriction of $\text{SL}(\exists, *, \neg*)$ to one quantified variable, i.e. $\text{SL}([\exists]_1, *, \neg*)$, admits PSPACE-complete satisfiability and validity problems [55], whereas the logic with just two quantified variables, i.e. $\text{SL}([\exists]_2, *, \neg*)$, is already non recursively enumerable on closed formulae [53]. As we already discussed when we introduced Theorem 3.5, $\text{SL}([\exists]_1, *, \neg*)$ can easily express both the predicates $\mathbf{n}(x) = \mathbf{n}(y)$ and $_\hookrightarrow x$. The distance between the decidability for $\text{SL}([\exists]_1, *, \neg*)$ and the undecidability for $\text{SL}([\exists]_2, *, \neg*)$ (not restricted to closed formulae) is best witnessed by Corollary 3.19(I) below, which solves an open problem [55, Section 6]. Moreover, Theorem 3.18 implies that adding **1s** to the quantifier-free separation logic $\text{SL}(*, \neg*)$ also leads to non RE satisfiability and validity problems. All these undecidability results are stated below for the record, without claiming that all these variants happen to be interesting in practice.

Corollary 3.19. The satisfiability and validity problems of the following logics are non RE:

- (I) $\text{SL}([\exists]_1, *, \neg*)$ augmented with $\mathbf{n}(x) \hookrightarrow \mathbf{n}(y)$,
- (II) $\text{SL}([\exists]_2, *, \neg*)$,
- (III) $\text{SL}(*, \neg*)$ augmented with either **1s**, \hookrightarrow^+ or \hookrightarrow^* ,
- (IV) $\text{SL}(\mathbf{n}(x), *, \neg*)$, $\text{SL}(*, \neg*, \hookrightarrow^2, \hookrightarrow^3)$ and all the logics above, restricted to 4 variables.

Proof of (I). As already stated, this is a consequence of Theorem 3.5 by observing that the predicate $\mathbf{n}(x) = \mathbf{n}(y)$ is equivalent to $\exists z (x \hookrightarrow z \wedge y \hookrightarrow z)$, whereas $_\hookrightarrow x$ is $\exists z z \hookrightarrow x$. \square

Proof of (II). Consequence of (I), as $\mathbf{n}(x) \hookrightarrow \mathbf{n}(y)$ can be expressed with two quantified variables with the formula $\exists z \exists v (x \hookrightarrow z \wedge z \hookrightarrow v \wedge y \hookrightarrow v)$. \square

Proof of (III). For $\text{SL}(*, \neg*, \mathbf{1s})$, it is a consequence of Theorem 3.18, as \hookrightarrow^δ is definable as soon as a separation logic features Boolean connectives, **emp**, separating conjunction and **1s**. In Section 2.1.1 we have shown how **1s** can be expressed with either \hookrightarrow^+ or \hookrightarrow^* . \square

Proof of (IV). It is shown in [53] that $\text{SL}(\exists, \neg*)$ restricted to two quantified variables is undecidable already on closed formulae. The translation provided in Section 3.2.1 assumes that distinct quantifications involve distinct variables. In order to translate $\text{SL}(\exists, \neg*)$ restricted to two quantified variables, it is necessary to give up that assumption and to update the definition of $\tau_{X,Y}(\varphi)$. Actually, only the clause for formulae of the form $\forall z \psi$ requires a change (where now z belongs to a set Y of two variables). Here is the new value for $\tau_{X,Y}(\forall z \psi)$ (that updates the definition given in Figure 3.7):

$$((z \hookrightarrow _ \wedge \mathbf{size} = 1) \vee \mathbf{emp}) * (\neg z \hookrightarrow _ \wedge (z \hookrightarrow _ \wedge \mathbf{size} = 1) \neg* (\text{Store}_Y(X) \Rightarrow \tau_{X,Y}(\psi))).$$

In particular, this formula uses the operator $*$ in order to remove the current assignment of \mathbf{z} (if it exists), and then apply the same translation given in Figure 3.7. The proof of Lemma 3.11 and Lemma 3.17 can be updated accordingly. As $\text{card}(\mathbb{Y}) = 2$, the formula $\tau_{\mathbf{X}, \mathbb{Y}}(\varphi)$ witnesses at most four variables (i.e. the cardinality of $\mathbb{Y} = \mathbb{Y} \cup \bar{\mathbb{Y}}$). \square

3.3.4 Modal separation logic is non RE.

Corollary 3.19(III) proves that MSL (Section 2.3.2) admits non RE satisfiability and validity problems by Proposition 2.20. However, Theorem 3.18 allows us to refine this result and show that it holds even when emp and both the modalities \Diamond^{-1} and $\langle \neq \rangle$ are dropped from the logic.

Theorem 3.20. MSL without \Diamond^{-1} , $\langle \neq \rangle$ and emp has non RE satisfiability and validity problems.

We refer the reader to Section 2.3.2 for the definition of the modal separation logic MSL, the definition of Kripke-style finite function and the definition of nominals. The specific fragment that we prove non RE is given by the grammar below:

$$\varphi := \top \mid p \mid \varphi \wedge \varphi \mid \neg \varphi \mid \varphi * \varphi \mid \varphi \multimap \psi \mid \Diamond \varphi.$$

First of all, notice that we can retrieve the formula emp , and define it as follows:

$$\text{emp} \stackrel{\text{def}}{=} \neg \Diamond \top \wedge ((\Diamond \neg \Diamond \top) \multimap \neg \Diamond \Diamond \top).$$

The correctness of this and every other formula introduced in this section is shown in Appendix A. By relying on emp , the logic can express the subtraction operator \multimap and the size formulae $\text{size} \geq \beta$ and $\text{size} = \beta$ (all defined as in Section 2.1.1).

Let φ be a formula in $\text{SL}(*, \multimap, \hookrightarrow^2, \hookrightarrow^3)$, written with variables from \mathbf{X} . Without loss of generality we assume $\text{AP} = \text{VAR}$ and $\mathcal{W} = \text{LOC}$. We define a translation $\tau_{\mathbf{X}}(\varphi)$ in the above fragment of MSL, that can be shown correct for the class of pointed finite functions encoding memory states, in the following sense.

Definition 3.21 (MSL - Memory state encoding.). A pointed finite function $(\mathcal{K}, \mathbf{w})$, where $\mathcal{K} = (\text{LOC}, R, \mathcal{V})$, is a \mathbf{X} -encoding of a memory state (s, h) whenever $R = h$ and moreover

1. every $\mathbf{x} \in \mathbf{X}$ is a nominal that corresponds to $s(\mathbf{x})$, i.e. $\mathcal{V}(\mathbf{x}) = \{s(\mathbf{x})\}$,
2. the current world \mathbf{w} is a spy, i.e. it is a nominal for a fixed propositional symbol $\text{spy} \notin \mathbf{X}$, it does not satisfy any propositional symbol in \mathbf{X} , and it does not belong to a pair in R .

Before defining the translation, let us derive formulae that capture the two properties (1) and (2) of Definition 3.21. Let $(\mathcal{K}, \mathbf{w})$ be a pointed finite function, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. First of all, we define the formula $\text{uniq}(p)$, stating that the current world \mathbf{w} is the only one satisfying $p \in \text{AP}$:

$$\text{uniq}(p) \stackrel{\text{def}}{=} \top * (p \wedge \text{emp} \wedge ((\text{size} = 1 \wedge \Diamond p) \multimap \Diamond \Diamond \top)).$$

This formula asks to consider the empty Kripke-style finite function $\mathcal{K}' = (\mathcal{W}, \emptyset, \mathcal{V}) \subseteq \mathcal{K}$. On this structure, it states that $\mathbf{w} \in \mathcal{V}(p)$ holds true (thus, it also holds for \mathcal{K}), and that whenever we add a Kripke-style finite function $\mathcal{K}'' = (\mathcal{W}, \{(\mathbf{w}, \mathbf{w}')\}, \mathcal{V})$, if $\mathbf{w}' \in \mathcal{V}(p)$ then $\mathcal{K}' + \mathcal{K}''$ witnesses a path of length two going from \mathbf{w} to a world \mathbf{w}' . As the accessibility relation of $\mathcal{K}' + \mathcal{K}''$ only has one arrow going from \mathbf{w} to \mathbf{w}' , this is only possible if $\mathbf{w}' = \mathbf{w}$. Therefore, \mathbf{w} is the only world satisfying p . By relying on $\text{uniq}(p)$, we define the formula $\text{nom}(p)$ stating that p is a nominal:

$$\text{nom}(p) \stackrel{\text{def}}{=} \top * (\text{emp} \wedge ((\Diamond \text{uniq}(p)) \multimap \top)).$$

$$\begin{array}{lll}
\tau_X(\top) & \stackrel{\text{def}}{=} \top, & \tau_X(\varphi \wedge \psi) \stackrel{\text{def}}{=} \tau_X(\varphi) \wedge \tau_X(\psi), \\
\tau_X(\text{emp}) & \stackrel{\text{def}}{=} \text{emp}, & \tau_X(\neg\varphi) \stackrel{\text{def}}{=} \neg\tau_X(\varphi), \\
\tau_X(x = y) & \stackrel{\text{def}}{=} \text{size} = 1 \multimap \Diamond(x \wedge y), & \tau_X(\varphi * \psi) \stackrel{\text{def}}{=} \tau_X(\varphi) * \tau_X(\psi), \\
\tau_X(x \hookrightarrow y) & \stackrel{\text{def}}{=} \text{size} = 1 \multimap \Diamond(x \wedge \Diamond y), & \tau_X(\varphi \multimap \psi) \stackrel{\text{def}}{=} (\text{is_a_spy}_X \wedge \tau_X(\varphi)) \multimap \tau_X(\psi), \\
\tau_X(x \hookrightarrow^2 y) & \stackrel{\text{def}}{=} \neg\tau_X(x = y) \wedge \neg\tau_X(x \hookrightarrow y) \wedge (\text{size} = 1 \multimap \Diamond(x \wedge \Diamond\Diamond y)), \\
\tau_X(x \hookrightarrow^3 y) & \stackrel{\text{def}}{=} \neg\tau_X(x = y) \wedge \neg\tau_X(x \hookrightarrow y) \wedge \neg\tau_X(x \hookrightarrow^2 y) \wedge (\text{size} = 1 \multimap \Diamond(x \wedge \Diamond\Diamond\Diamond y)).
\end{array}$$

Figure 3.8: Translation from $\text{SL}(*, \multimap, \hookrightarrow^2, \hookrightarrow^3)$ to a fragment of MSL.

Again, this formula asks to consider the empty function $\mathcal{K}' = (\mathcal{W}, \emptyset, \mathcal{V}) \subseteq \mathcal{K}$. On this structure, it states that it is possible to add a Kripke-style finite function $\mathcal{K}'' = (\mathcal{W}, \{(w, w')\}, \mathcal{V})$ such that $(\mathcal{K}'', w') \models \text{uniq}(p)$. This construction can be satisfied if and only if p is a nominal.

Let us now characterise the fact that w is a spy. The following formula does the job:

$$\text{is_a_spy}_X \stackrel{\text{def}}{=} \text{uniq}(\text{spy}) \wedge (\bigwedge_{x \in X} \neg x) \wedge \neg \Diamond \top \wedge \neg ((\text{size} = 1 \wedge \Diamond \neg \text{spy}) \multimap \Diamond \Diamond \text{spy}).$$

The first two conjuncts of is_a_spy_X capture the fact that w must be a nominal for spy and that it does not satisfy any symbol from X . Instead, the last two conjuncts state that $R(w) = \emptyset$ and $R^{-1}(w) = \emptyset$, as required by property (2). The first equality is directly captured by the subformula $\neg \Diamond \top$, whereas the subformula $\neg ((\text{size} = 1 \wedge \Diamond \neg \text{spy}) \multimap \Diamond \Diamond \text{spy})$ captures $R^{-1}(w) = \emptyset$ under the hypothesis that both $\text{uniq}(\text{spy})$ and $\neg \Diamond \top$ hold.

Being able to express nominals and spies allows us to define the translation $\tau_X(\varphi)$ as shown in Figure 3.8. We highlight two points of this translation. First, the formula $\tau_X(\varphi \multimap \psi)$ constraints the magic wand to only consider relations satisfying is_a_spy_X , so that $\tau_X(\varphi)$ and $\tau_X(\psi)$ are evaluated on pointed finite functions that satisfy the conditions in Definition 3.21. Second, since is_a_spy_X insures that the current world does not belong to a pair in R , we can combine the modality \Diamond and the subtraction \multimap in order to simulate the $\langle \neq \rangle$ modality. This “trick” is used to translate all the atomic predicate between program variables. For instance, let us consider $\tau_X(x \hookrightarrow y)$. This formula states that adding (w, w_x) , where w is the current world and w_x corresponds to the nominal x , to the accessibility relation R leads to a path of length two going from w to the world w_y corresponding to the nominal y . This is only possible if (w_x, w_y) belongs to R , as required by the predicate $x \hookrightarrow y$. The translation is shown to be correct below.

Lemma 3.22. Let (s, h) be a memory state and let φ be a formula in $\text{SL}(*, \multimap, \hookrightarrow^2, \hookrightarrow^3)$, with variables from $X \subseteq_{\text{fin}} \text{VAR} \setminus \{\text{spy}\}$. Let (\mathcal{K}, w) be a pointed finite function that is an X -encoding of (s, h) . We have, $(s, h) \models \varphi$ if and only if $(\mathcal{K}, w) \models \tau_X(\varphi)$.

This lemma states that the translation is correct and, similarly to Lemma 3.11, its proof is by structural induction on φ (see Appendix A). Afterwards, to prove Theorem 3.20 we simply use $\text{nom}(p)$ and is_a_spy to characterise the set of pointed finite functions considered in Lemma 3.22. This is formalised in the lemma below, while its proof is given in Appendix A.

Lemma 3.23. Let φ be a formula in $\text{SL}(*, \multimap, \hookrightarrow^2, \hookrightarrow^3)$, with variables from $X \subseteq_{\text{fin}} \text{VAR} \setminus \{\text{spy}\}$.

- (I) φ and $\text{is_a_spy}_X \wedge \bigwedge_{x \in X} \text{nom}(x) \wedge \tau_X(\varphi)$ are equisatisfiable.
- (II) φ and $\text{is_a_spy}_X \wedge \bigwedge_{x \in X} \text{nom}(x) \Rightarrow \tau_X(\varphi)$ are equivalent.

Intensionality and Reachability Leads to Non-elementary Logics

Contents

4.1	The Hardness of Reachability and Submodel Reasoning	77
4.1.1	ALT: An Auxiliary Logic on Trees.	77
4.1.2	Relating ALT and the separation logic $SL([\exists]_1, *, \mathbf{x} \hookrightarrow _, \hookrightarrow^+)$	80
4.2	On the Expressive Power of ALT	81
4.2.1	Towards TOWER-hardness: how to encode finite words in ALT.	82
4.2.2	Intermezzo: inexpressibility results via Ehrenfeucht-Fraïssé games.	88
4.3	The Complexity of ALT	97
4.3.1	Propositional Interval Temporal Logic.	97
4.3.2	PITL on marked words.	98
4.3.3	Reducing PITL to ALT.	101
4.4	Revisiting TOWER-hard Logics with ALT	103
4.4.1	From ALT to $SL(*, \neg*, \mathbf{1s})$ with bounded magic wand.	104
4.4.2	From ALT to Quantified Computation Tree Logic.	105
4.4.3	From ALT to Modal Separation Logic.	112

In this chapter

The proof that the logic $\text{SL}(*, *, \leftrightarrow^*)$ has a non RE satisfiability problem (Chapter 3) raises new questions on the complexity of other fragments of $\text{SL}(\exists, *, \neg)$ featuring reachability predicates. The aim of this chapter is to study the logic $\text{SL}([\exists]_1, *, \mathbf{x} \leftrightarrow _, \leftrightarrow^+)$, which sits between the PSPACE-complete $\text{SL}([\exists]_1, *, \mathbf{x} \leftrightarrow _)$ and the TOWER-complete $\text{SL}([\exists]_2, *)$. This logic features one quantified variable name, the separating conjunction $*$, the predicate alloc $\mathbf{x} \leftrightarrow _$ and the reachability predicate $\mathbf{x} \leftrightarrow^+ \mathbf{y}$.

As done in Chapter 3, instead of studying directly this logic we take a detour and only consider its significant features: its ability to reason on submodels, and the way it can express reachability conditions. We introduce the auxiliary logic ALT which corresponds to the minimal fragment of $\text{SL}([\exists]_1, *, \mathbf{x} \leftrightarrow _, \leftrightarrow^+)$ having these features. After studying its expressive power, we show that ALT admits a TOWER-complete satisfiability problem. Despite showing that the satisfiability for $\text{SL}([\exists]_1, *, \mathbf{x} \leftrightarrow _, \leftrightarrow^+)$ is non-elementary, our efforts are rewarded as ALT reveals to be an interesting theoretical instrument to prove TOWER-hardness of logics interpreted on trees. We prove that ALT is captured by non-trivial fragments of four logics that were independently found to be TOWER-hard: first-order separation logic with bounded separating implication [22], quantified computation tree logic (QCTL) [99], modal logic of heaps (MLH) [52] and modal separation logic (MSL) [54].

Here is a roadmap of the chapter.

Section 4.1. We introduce the Auxiliary Logic on Trees (ALT), a modal logic interpreted on finite forests that features reachability predicates together with the universal modality $\langle U \rangle$ [80], the sabotage modality \blacklozenge from sabotage modal logic [4] and its Kleene closure \blacklozenge^* . We show that ALT is a fragment of $\text{SL}([\exists]_1, *, \mathbf{x} \leftrightarrow _, \leftrightarrow^+)$, and thus decidable in TOWER.

Section 4.2. We analyse the expressive power of ALT. First of all, we introduce an encoding of finite words into finite forests (Definition 4.4), and show that the set of forests encoding finite words is characterisable in ALT (Lemma 4.10). This result provides a first step in the proof of TOWER-hardness of ALT. Second, we explore inexpressibility results for ALT by adapting the notion of Ehrenfeucht-Fraïssé games for first-order logic.

Section 4.3. We prove the main result of the chapter.

Theorem 4.28. The satisfiability problem of ALT is TOWER-complete.

The TOWER-hardness is by reduction from Propositional Interval Temporal Logic (PITL) under locality principle, known to be TOWER-complete [112, 128]. This logic is interpreted on finite words, and features a composition operator $|$ that splits the word into a prefix and a suffix overlapping in exactly one position. In view of the inexpressibility results obtained in Section 4.2, this operation cannot be directly simulated in ALT. To circumvent this problem, we introduce an alternative and computationally equivalent semantics for PITL, where the operator $|$ marks a position in the word, instead of splitting it into two subwords. The satisfiability problem of PITL under this new semantics is reduced to the satisfiability problem of ALT, leading to Theorem 4.28.

Section 4.4. We rely on the TOWER-hardness of ALT to refine the analysis on the computational complexity of separation logic, QCTL, MSL and MLH. We start with the first-order

separation logic $\text{SL}(\exists, *, \neg[n])$ featuring the suite of bounded separating implications $\neg[n]$ ($n \in \mathbb{N}$). Roughly speaking, $\neg[n]$ restricts the standard separating implication $\neg*$ so that only heaps with at most n memory cells can be added to the current one. $\text{SL}(\exists, *, \neg[n])$ is shown to be TOWER-complete in [22]. We show that TOWER-hardness already holds for its quantifier-free fragment $\text{SL}(*, \neg[1], \text{ls})$, where we notice that the connectives $\neg[n]$ are restricted to $n = 1$.

Theorem 4.29. Satisfiability of the two-variable fragment of $\text{SL}(*, \neg[1], \text{ls})$ is TOWER-c.

Afterwards, we move to Quantified Computation Tree Logic (QCTL), an extension of Computation Tree Logic featuring second-order propositional quantification. When interpreted on (either finite or infinite) trees, QCTL is known to be TOWER-complete [99]. ALT allows us to sharpen the TOWER-hardness analysis and show two fragments of QCTL that are already non-elementary.

Theorem 4.41. The satisfiability problems of $\text{QCTL}^t(\text{EU}^0)$ and $\text{QCTL}^t(\text{EF}^1)$ are TOWER-c.

Here, $\text{QCTL}^t(\text{EU}^0)$ stands for the fragment of QCTL restricted to the temporal modality exists-until, which cannot be nested. Instead, $\text{QCTL}^t(\text{EF}^1)$ is the fragment of QCTL restricted to the temporal modality exists-finally, which can be nested at most once.

Lastly, we consider the modal separation logics MSL and MLH introduced in Section 2.3.2. Thanks to ALT, we are able to find a common syntactical fragment of these two logics that is already TOWER-hard (in the following theorem $\blacklozenge_{\text{ML}}$ and $\blacklozenge_{\text{ML}}^*$ stand for the operators \blacklozenge and \blacklozenge^* of ALT, internalised in MSL and MLH by relying on the separating conjunction).

Theorem 4.45. The fragment of MLH and MSL with the $*$ (alternatively, $\blacklozenge_{\text{ML}}$ and $\blacklozenge_{\text{ML}}^*$), \top , Boolean connectives, \Diamond and $\langle U \rangle$ modalities, and has a TOWER-complete satisfiability problem.

As a general remark, all the reductions from the satisfiability problem of ALT to the satisfiability problem of the logics considered in this section are achieved via semantically faithful translations. Not only these reductions are quite straightforward, as the burden of proving TOWER-hardness is left to Theorem 4.28, but they show that all these logics are non-elementary as they reason on reachability and submodels in the same way.

4.1 THE HARDNESS OF REACHABILITY AND SUBMODEL REASONING

In the previous chapter, we saw how adding reachability predicates to $\text{SL}(*, \neg *)$ leads to logics with non RE satisfiability problems. Due to this unsatisfactory result, in this chapter we investigate further the effects of adding these predicates to other fragments of first-order separation logic. In Chapter 3 we identified how the separating implication can be used to update the memory state in a way that simulates the first-order quantification. However, if we restrict the use of the separating implication so that it can be used only to define the alloc predicate $(\cdot) \hookrightarrow _$, we know that the fragment of first-order logic featuring one quantified variable, i.e. $\text{SL}([\exists]_1, *, \mathbf{x} \hookrightarrow _)$, admits a PSPACE-complete satisfiability (and validity) problem [55]. In contrast, considering two quantified variables, i.e. the logic $\text{SL}([\exists]_2, *)$, makes the problem already non-elementary decidable [53]. As we know that $\text{SL}([\exists]_2, *)$ can express reachability predicates (Section 2.1.1), our hope is to show that enriching $\text{SL}([\exists]_1, *, \mathbf{x} \hookrightarrow _)$ with the reachability predicate \hookrightarrow^+ leads to a logic that is computationally less demanding than $\text{SL}([\exists]_2, *)$. As we have done in the previous chapter, to study the problem we take a detour and only consider the significant features of this logic: its ability to reason on submodels, and the way it can express reachability conditions. We also drop the memory states and consider the more classical frameworks of trees and forests. This makes our research agenda shifts a bit, so that the main question becomes

“Can logics featuring submodel reasoning and reachability predicates admit elementary satisfiability problems when interpreted on trees?”

To formalise and tackle this question, we take these features and formally exhibit them through an Auxiliary Logic on Trees (ALT). This logic, essentially deals with reachability of a fixed (target) node inside a forest, from a current node that can be updated with the somewhere modality $\langle U \rangle$. Moreover, the logic features two modalities that are subsumed by the separating conjunction in order to reason on submodels. Unfortunately, we show that this very restricted fragment of $\text{SL}([\exists]_2, *, \neg *)$ already admits a TOWER-complete satisfiability problem, answering negatively to our question. The proof is by reduction from the satisfiability problem of Propositional Interval Temporal Logic (PITL) under locality principle [112], for which we define an equivalent semantics that better suits the expressive power of ALT. Despite this negative result, ALT allows us to study other logics interpreted on trees and (re)prove their TOWER-hardness. Indeed, in the last part of this chapter we show that ALT is captured by four logics that were independently found to be TOWER-hard: the first-order separation logic with bounded separating implication from [22], quantified computation tree logic (QCTL) [99], modal logic of heaps (MLH) [52] and modal separation logic (MSL) [54]. In this context, beside exposing that all these logics admit TOWER-hard satisfiability problem because of the way they reason about reachability and submodels, we discover interesting sublogics that are still TOWER-complete:

- $\text{SL}(*, \neg^{[1]}, \mathbf{1s})$, where $\varphi \neg^{[1]} \psi$ is the syntactical restriction to the separating implication corresponding to the formula $(\varphi \wedge \neg \mathbf{size} \geq 2) \multimap \psi$ (Theorem 4.29),
- QCTL^t restricted to $E(\varphi U \psi)$ modalities, where φ, ψ are Boolean combinations of atomic propositions, or to the EF modality, which can be nested at most once (Theorem 4.41),
- the common fragment of MLH and MSL having Boolean connectives and the modalities \Diamond , $\langle U \rangle$ and $*$ (Theorem 4.45). Notice that this logic does not have propositional symbols.

4.1.1 ALT: An Auxiliary Logic on Trees.

We introduce an *Auxiliary Logic on Trees* (ALT). Its formulae φ are from the grammar:

$\pi :=$	\top	(true)	$\varphi :=$	π	(atomic formulae)
	Hit	(hit predicate)		$\varphi \wedge \varphi$	(Boolean connectives)
	Miss	(miss predicate)		$\diamond \varphi$	(sabotage modality)
				$\diamond^* \varphi$	(repeated sabotage modality)
				$\langle U \rangle \varphi$	(somewhere modality)

The symbol $\langle U \rangle$, already introduced in Section 2.3.2, is borrowed from V. Goranko and S. Passy paper on modal logic with universal modality [80]. Readers who are familiar with sabotage modal logics will recognise in \diamond the sabotage modality [4], and in \diamond^* its Kleene closure (i.e. the operator \diamond applied an arbitrary number of times). Similarly to the separating conjunction of separation logic, these two operators modify the model during the evaluation of a formula, making ALT a *relation-changing* modal logic (following the terminology used in [3]). However, contrary to most modal logics, ALT does not feature classical propositional symbols. Instead, this logic only features two interpreted atomic propositions Hit and Miss. Roughly speaking, Hit stands for “the target node is reachable” whereas Miss stands for “the target node is not reachable”. The formal definitions will be given soon in order to clarify these two sentences.

Let \mathcal{N} be a countably infinite set of *nodes*. We consider the class of models of finite forests.

Definition 4.1 (Forest). A (finite) forest $\mathcal{F} : \mathcal{N} \rightarrow_{\text{fin}} \mathcal{N}$ is a partial function that has finite domain and is acyclic, i.e. $\mathcal{F}^\delta(n) \neq n$ for all $n \in \text{dom}(\mathcal{F})$ and $\delta \geq 1$. Pairs in \mathcal{F} are called *edges*.

Albeit non-standard, our definition of finite forests over an infinite set of nodes simplifies the forthcoming definitions and makes the connections with separation logic more direct, as forests can be seen as acyclic heaps. Besides, in Section 4.3 we show how restricting \mathcal{N} to a finite set does not change the expressive power nor the complexity of ALT. We recall the standard notions of ancestors and parent of a node.

Definition 4.2 (Ancestors and Parents). Let n, n' be two nodes, and let \mathcal{F} be a forest. n' is a \mathcal{F} -ancestor of n if there is a path in the forest going from n to n' , i.e. $\mathcal{F}^\delta(n) = n'$ for some $\delta \geq 1$. If $\delta = 1$ then n' is the \mathcal{F} -parent of n .

Notice that, with this classification, \mathcal{F} encodes the parent relation. We drop the prefix \mathcal{F} - from \mathcal{F} -ancestor and \mathcal{F} -parent when the forest is clear from the context. As usual, if n' is an ancestor of n , then we can alternatively say that n is a *descendant* of n' . Similarly, n is a *child* of n' whenever n' is the parent of n . Given two forests $\mathcal{F}, \mathcal{F}'$, we say that \mathcal{F}' is a *subforest* of \mathcal{F} whenever $\mathcal{F}' \subseteq \mathcal{F}$ holds (as usual, we see functions as binary relations). Figure 4.1 intuitively represents two forests, the one on the left being a subforest of the one on the right. As done for heaps, nodes are denoted by small boxes (\square), and arrows represent the forest.

Semantics. ALT is interpreted on *pointed forests* (\mathcal{F}, t, n) , where \mathcal{F} is a forest and $t, n \in \mathcal{N}$ are two nodes. The node t is called the *target node*. The node n is the *current (evaluation) node*. The satisfaction relation \models for the formulae of ALT is given in Figure 4.2, omitting the standard clauses for \top and Boolean connectives. The semantics of Hit and Miss is pretty straightforward. Hit holds if there is a path in the forest going from the current node to the target node. Instead, Miss holds if the current node is in the domain of the forest, but such a path does not exist. Given a pointed forest (\mathcal{F}, t, n) , n is called a *hit node* whenever $(\mathcal{F}, t, n) \models \text{Hit}$. Instead, if $(\mathcal{F}, t, n) \models \text{Miss}$ then n is a *miss node*. As a visual aid, the hit nodes of the forest in

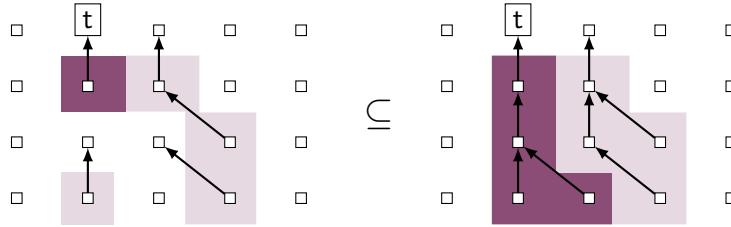


Figure 4.1: Subforest relation.

-
- $(\mathcal{F}, t, n) \models \text{Hit}$ iff n is a \mathcal{F} -descendant of t ,
 $(\mathcal{F}, t, n) \models \text{Miss}$ iff $n \in \text{dom}(\mathcal{F})$ and n is not a \mathcal{F} -descendant of t ,
 $(\mathcal{F}, t, n) \models \blacklozenge \varphi$ iff there is \mathcal{F}' such that $\mathcal{F}' \subseteq \mathcal{F}$, $\text{card}(\mathcal{F}') + 1 = \text{card}(\mathcal{F})$ and $(\mathcal{F}', t, n) \models \varphi$,
 $(\mathcal{F}, t, n) \models \blacklozenge^* \varphi$ iff there is \mathcal{F}' such that $\mathcal{F}' \subseteq \mathcal{F}$ and $(\mathcal{F}', t, n) \models \varphi$,
 $(\mathcal{F}, t, n) \models \langle U \rangle \varphi$ iff there is $n' \in \mathcal{N}$ such that $(\mathcal{F}, t, n') \models \varphi$.

Figure 4.2: Satisfaction relation for ALT, with respect to a pointed forest state (\mathcal{F}, t, n) .

Figure 4.1 are the ones in the darker area, whereas the ones in the lighter (not white) area are miss nodes. It is worth noting that **Miss** is not exactly the negation of **Hit**, as it requires the current evaluation node to be in the domain of the forest. On the other hand, let us define the formula $\text{inDom} \stackrel{\text{def}}{=} \text{Hit} \vee \text{Miss}$, which is satisfied by (\mathcal{F}, t, n) if and only if $n \in \text{dom}(\mathcal{F})$. Any two of the three formulae **Hit**, **Miss** and **inDom** suffice in order to define the third one. In particular:

$$\text{Hit} \equiv \text{inDom} \wedge \neg \text{Miss},$$

$$\text{Miss} \equiv \text{inDom} \wedge \neg \text{Hit}.$$

This tells us that, in order to relate separation logic with ALT, we can consider a logic featuring reachability predicates (to capture **Hit**) and the alloc predicate (to capture **inDom**).

Let us continue the analysis on the features of ALT. As stated before, the semantics given to $\langle U \rangle \varphi$ is the one of the existential modality *somewhere* [80], stating that there is a way to change the current evaluation node so that φ becomes true. As such, this operator can be seen as a restricted form of first-order quantification, where the reassignment only occurs on the current evaluation node. Its dual operator $[U] \varphi \stackrel{\text{def}}{=} \neg \langle U \rangle \neg \varphi$ is the universal modality *everywhere*, stating that φ holds on every node in \mathcal{N} . The semantics given to $\blacklozenge \varphi$ is the one of the *sabotage* modality from [4], which requires to find one edge of the forest that, when removed, makes the model satisfy φ . Its dual operator $\blacksquare \varphi \stackrel{\text{def}}{=} \neg \blacklozenge \neg \varphi$ states that φ holds on every subforest obtained from the current forest by removing just one edge. Lastly, the modality \blacklozenge^* , here called *repeated sabotage*, can be seen as the operator obtained by applying \blacklozenge an arbitrary number of times. Indeed, by inductively defining $\blacklozenge^k \varphi$ ($k \in \mathbb{N}$) as the formula φ for $k = 0$ and otherwise ($k \geq 1$) as $\blacklozenge \blacklozenge^{k-1} \varphi$, it is easy to see that

$$(\mathcal{F}, t, n) \models \blacklozenge^* \varphi \text{ if and only if } (\mathcal{F}, t, n) \models \blacklozenge^k \varphi \text{ for some } k \in \mathbb{N}.$$

Given a pointed forest (\mathcal{F}, t, n) , we write $\mathcal{F}[\text{Miss}]_t$ to denote the set of its miss nodes, i.e. $\mathcal{F}[\text{Miss}]_t \stackrel{\text{def}}{=} \{n' \in \mathcal{N} \mid (\mathcal{F}, t, n') \models \text{Miss}\}$. We omit the subscript t from $\mathcal{F}[\text{Miss}]_t$ when it is clear from the context. We augment the standard precedence rules of propositional logic so

that the modalities $\langle U \rangle$, \diamond and \diamond^* have the same precedence as the negation \neg . For instance, the formula $\langle U \rangle \text{Hit} \wedge \text{Miss}$ should be read as $(\langle U \rangle \text{Hit}) \wedge \text{Miss}$.

4.1.2 Relating ALT and the separation logic $\text{SL}([\exists]_1, *, x \hookrightarrow -, \hookrightarrow^+)$.

We start analysing ALT by showing its connections with separation logic. In particular, we consider the separation logic $\text{SL}([\exists]_1, *, x \hookrightarrow -, \hookrightarrow^+)$, with formulae φ from the following grammar:

$$\begin{array}{lll} \pi := \top & (\text{true}) & \varphi := \pi \\ | \quad \text{emp} & (\text{empty predicate}) & | \quad \varphi \wedge \varphi \quad | \quad \neg \varphi \\ | \quad x = y & (\text{equality predicate}) & | \quad \varphi * \varphi \\ | \quad x \hookrightarrow y & (\text{points-to predicate}) & | \quad \exists u \varphi \\ | \quad x \hookrightarrow - & (\text{alloc predicate}) & \\ | \quad x \hookrightarrow^+ y & (\text{reach-plus predicate}) & \end{array} \quad \begin{array}{ll} & (\text{atomic formulae}) \\ & (\text{Boolean connectives}) \\ & (\text{separating conjunction}) \\ & (\text{first-order quantification on } u) \end{array}$$

where $x, y, u \in \text{VAR}$. Additionally, every formula φ of this logic can only quantify over the variable name u , i.e. $\text{bv}(\varphi) \subseteq \{u\}$. The semantics of every element of $\text{SL}([\exists]_1, *, x \hookrightarrow -, \hookrightarrow^*)$ is standard, as defined in Chapter 2. Let us recall the definition of the alloc predicate $x \hookrightarrow -$ and the reach-plus predicate $x \hookrightarrow^+ y$, for a given memory state (s, h) :

$$\begin{aligned} (s, h) \models x \hookrightarrow - & \quad \text{iff} \quad s(x) \in \text{dom}(h), \quad (x \text{ corresponds to a memory cell of } h) \\ (s, h) \models x \hookrightarrow^+ y & \quad \text{iff} \quad (s(x), s(y)) \in h^+. \quad (\text{there is a non-empty path going from } s(x) \text{ to } s(y)) \end{aligned}$$

We know that $\text{SL}([\exists]_1, *, x \hookrightarrow -, \hookrightarrow^+)$ is a fragment of $\text{SL}([\exists]_2, *)$, as in Section 2.1.1 we have shown that both the alloc predicate and the reach-plus predicate can be expressed using only two quantified variables. Moreover, in that section we introduced the formula $\text{size} = 1$ stating that the heap has cardinality exactly one. It is defined as $\neg \text{emp} \wedge \neg(\neg \text{emp} * \neg \text{emp})$. This formula allows us to capture the sabotage operator \diamond of ALT.

We show that ALT is a fragment of $\text{SL}([\exists]_1, *, x \hookrightarrow -, \hookrightarrow^+)$. First, we notice that both sabotage and repeated sabotage operators can be expressed in separation logic. The sabotage operator can be defined as $\diamond_{\text{SL}} \varphi \stackrel{\text{def}}{=} (\text{size} = 1) * \varphi$, whereas the repeated sabotage operator can be defined as $\diamond^*_{\text{SL}} \varphi \stackrel{\text{def}}{=} \top * \varphi$. A quick semantical check shows that both formulae capture the analogous operators of ALT as follows:

$$\begin{aligned} (s, h) \models \diamond_{\text{SL}} \varphi & \quad \text{iff} \quad \text{there is a heap } h_1 \text{ s.t. } h_1 \subseteq h, \text{card}(h_1) + 1 = \text{card}(h) \text{ and } (s, h_1) \models \varphi. \\ (s, h) \models \diamond^*_{\text{SL}} \varphi & \quad \text{iff} \quad \text{there is a heap } h_1 \text{ s.t. } h_1 \subseteq h \text{ and } (s, h_1) \models \varphi. \end{aligned}$$

In order to translate a formula of ALT into a formula of $\text{SL}([\exists]_1, *, x \hookrightarrow -, \hookrightarrow^+)$, we fix a variable $x \in \text{VAR}$ that is syntactically different from the quantified variable u and that plays the role of the target node. Then, the translation $\tau_x(\varphi)$ of a formula φ in ALT is straightforward:

$$\begin{array}{lll} \tau_x(\text{Hit}) & \stackrel{\text{def}}{=} u \hookrightarrow^+ x, & \tau_x(\diamond \varphi) \stackrel{\text{def}}{=} \diamond_{\text{SL}} \tau_x(\varphi), & \tau_x(\top) \stackrel{\text{def}}{=} \top, \\ \tau_x(\text{Miss}) & \stackrel{\text{def}}{=} u \hookrightarrow - \wedge \neg \tau_x(\text{Hit}), & \tau_x(\diamond^* \varphi) \stackrel{\text{def}}{=} \diamond^*_{\text{SL}} \tau_x(\varphi), & \tau_x(\neg \varphi) \stackrel{\text{def}}{=} \neg \tau_x(\varphi). \\ \tau_x(\langle U \rangle \varphi) & \stackrel{\text{def}}{=} \exists u \tau_x(\varphi), & \tau_x(\varphi \wedge \psi) \stackrel{\text{def}}{=} \tau_x(\varphi) \wedge \tau_x(\psi), & \end{array}$$

The translation of Hit and Miss is as expected: $\tau_x(\text{Hit})$ states that there is a non-empty path from $s(u)$ to $s(x)$. This corresponds to the case where the current evaluation node is a hit node, i.e. a descendant of the target node. $\tau_x(\text{Miss})$ states that $s(u)$ is in the domain of the heap, which

does not witness a non-empty path going from $s(u)$ to $s(x)$. This corresponds to the case where the current evaluation node is a miss node. In every other case, the translation from ALT to separation logic is homomorphic, and interestingly enough the variable equality and the points-to predicate of $\text{SL}([\exists]_1, *, x \hookrightarrow _, \hookrightarrow^+)$ are not needed for the translation. Assuming (without loss of generality) that $\mathcal{N} = \text{LOC}$, we can show that for a given pointed forest (\mathcal{F}, t, n) and a store s such that $s(x) = t$ and $s(u) = n$, we have

$$(\mathcal{F}, t, n) \models \varphi \text{ if and only if } (s, \mathcal{F}) \models \tau_x(\varphi).$$

This statement can be proved with an easy structural induction on φ , which is left to the reader. Furthermore, in order to conclude that ALT is a fragment of $\text{SL}([\exists]_1, *, x \hookrightarrow _, \hookrightarrow^+)$ it is sufficient to show that this separation logic can characterise the class of models of ALT. As finite forests are isomorphic to the class of acyclic heaps, this can be done with the formula $\forall u \neg(u \hookrightarrow^+ u)$.

Proposition 4.3. ALT is a fragment of $\text{SL}([\exists]_1, *, x \hookrightarrow _, \hookrightarrow^+)$ restricted to two variable names.

Notice that the translation uses only two variables: the free variable x and the (possibly bound) variable u . This makes ALT a fragment of $\text{SL}([\exists]_2, *)$ on closed formulae, which is known to admit a TOWER-complete satisfiability problem [53]. Thus, the lemma above shows that the satisfiability problem of ALT is in TOWER. Moreover, from what we have seen in Section 2.2, we conclude that ALT is a fragment of WMSO † .

Of course, all these connections with other logics raise the question on whether ALT is needed in order to present the results of the next sections. As we will see, ALT comes with the benefit that various connections with other logics, as for example quantified computation tree logic, can be drawn very easily. Partially, this is due to the fact we consider forests instead of heaps, and that ALT does not feature predicates that are equivalent to $x \hookrightarrow y$ or $x = y$. From the point of expressiveness, from [1] we know that $\text{SL}([\exists]_2, *)$, and therefore ALT, is strictly less expressive than WMSO † . This also helps when reducing ALT to other logics.

4.2 ON THE EXPRESSIVE POWER OF ALT

It is certainly true that the two atomic propositions **Hit** and **Miss** make rather obscure what properties can be expressed in ALT. In order to become more familiar with the features of this logic, in this section we start playing with it. As we will soon find out, the ability to reason about submodels given by the combination of the two operators \blacklozenge and \blacklozenge^* greatly increases the expressive power of ALT. In particular, we show that ALT is able to characterise finite words. Encoding finite words in ALT is also the first step we need to show that this logic admits a TOWER-complete satisfiability problem. The proof of TOWER-hardness, addressed in Section 4.3, is by reduction from the satisfiability problem of Moszkowski's propositional interval temporal logic under locality condition (defined in Section 4.3.1). As we will see, this reduction is somewhat non-intuitive and can perhaps appear needlessly complicated. The reason for this is that we need to get around the fact that, given a pointed forest (\mathcal{F}, t, n) , ALT cannot deduce any property of the portion of the model corresponding to the set $\mathcal{F}[\text{Miss}]$, other than bounds on the size of this set and whether n belongs to it or not. This is shown formally at the end of this section, after providing a notion of Ehrenfeucht-Fraissé games based on the works in [36, 48].

4.2.1 Towards Tower-hardness: how to encode finite words in ALT.

As a first step, we define a correspondence between finite words and specific pointed forests. As usual, the set of *finite words* on a *finite alphabet* Σ is defined as the closure of Σ under Kleene star, i.e. Σ^* . To ease our modelling, we suppose $\Sigma \stackrel{\text{def}}{=} [1, n]$ to be the alphabet of natural numbers between 1 and n . Let $\mathbf{w} = a_1 \dots a_k$ be a k -symbols word in Σ^* . Let us explain how to encode \mathbf{w} as a finite forest. Every *symbol* a_j ($j \in [1, k]$), is encoded using a node n_j and $a_j + 1$ additional nodes that are children of n_j . So, for example the symbol 3 is represented by a node having four children. All the nodes in $\{n_j \mid j \in [1, k]\}$ are then connected in a path going from n_1 to the target node t , so that for every $j \in [1, k - 1]$ n_j is a child of n_{j+1} , and n_k is a child of t . Notice that, with the exception of n_1 , for every $j \in [2, k]$ this increases the number of children of n_j by one. Let us now formalise this encoding.

Definition 4.4 (Word encoding). Let $\mathbf{w} = a_1 \dots a_k$ be in Σ^* , where $\Sigma = [1, n]$ for some $n \geq 1$. We say that a pointed forest $(\mathcal{F}, t, \mathbf{n})$ encodes \mathbf{w} if and only if there is a tuple $\mathbb{M} = (n_1, \dots, n_k)$ of k nodes and tuple $\mathbb{C} = (N_1, \dots, N_k)$ of k sets of nodes such that

1. $\{n_1, \dots, n_k\}, N_1, \dots, N_k, \mathcal{F}[\text{Miss}]_t$ are pairwise disjoint sets, i.e. they do not share any node,
2. \mathbb{M} and \mathbb{C} are all the descendants of t , i.e. $(\mathcal{F}^{-1})^+(t) = \{n_1, \dots, n_k\} \cup \bigcup_{j \in [1, k]} N_j$,
3. n_k is the only child of t and for every $j \in [1, k - 1]$ $\mathcal{F}(n_j) = n_{j+1}$,
4. for every $j \in [1, k]$, $\text{card}(N_j) = a_j + 1$ and for every $n' \in N_j$, $\mathcal{F}(n') = n_j$.

Notice that given a pointed forest $(\mathcal{F}, t, \mathbf{n})$ encoding \mathbf{w} the tuples \mathbb{M} and \mathbb{C} are uniquely defined. With respect to the elements in Definition 4.4, the k nodes in \mathbb{M} are called *main nodes*, whereas the nodes in N_j ($j \in [1, k]$) are called *character nodes*. Main nodes and character nodes partition the set of \mathcal{F} -descendants of t . Thanks to the condition (3), main nodes form a path in the forest \mathcal{F} , going from n_1 to n_k . We call this path the *main path* of \mathcal{F} (notice that we do not include the target node t). The following proposition stresses four important properties of our encoding that follow directly from its definition.

Proposition 4.5. Let $(\mathcal{F}, t, \mathbf{n})$ be an encoding of $\mathbf{w} = a_1 \dots a_k$, with main nodes $\mathbb{M} = (n_1, \dots, n_k)$.

- (I) $n' \in \mathcal{N}$ is a main node if and only if it is a descendant of t and has at least one child.
- (II) n_1 is the only main node having the same number of descendants and children.
- (III) Given $j \in [2, k]$, n_j has exactly one child that is a main node.
- (IV) Given $j \in [1, k]$, n_j has exactly $a_j + 1$ children that are character nodes.

We say that a node $n \in \text{dom}(\mathcal{F})$ encodes the symbol $a \in \Sigma$ if it has exactly $a + 1$ children that are not in \mathbb{M} . Then, main nodes are the only ones encoding symbols, where n_j encodes a_j for every $j \in [1, k]$ (by property (IV)).

Example 4.6. Figure 4.3 shows a pointed forest encoding the word 1121. The main nodes of the encoding are $\mathbb{M} = (n_1, n_2, n_3, n_4)$, and its main path is given $\{(n_1, n_2), (n_2, n_3), (n_3, n_4)\}$. Supposing that the tuple of character nodes is $\mathbb{C} = (N_1, N_2, N_3, N_4)$, the set N_j ($j \in [1, 4]$) contains the children of n_j that are not in \mathbb{M} , so that $\text{card}(N_1) = \text{card}(N_2) = \text{card}(N_4) = 2$, whereas $\text{card}(N_3) = 3$. Albeit the forest depicted here does not have miss nodes, in general encodings of words can have an arbitrary number of them.

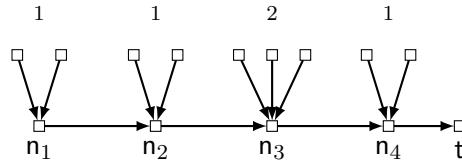


Figure 4.3: A forest encoding the word 1121.

Descendants and Children. We are now interested in characterising the class of pointed forests encoding finite words. In order to do so, we start by defining some easy formulae, which also serves as a way of familiarising with the logic. Looking at the properties in Proposition 4.5, we notice that they mainly rely on counting the number of descendants and children of a given node. Therefore, for now we focus on defining two formulae, $\#\text{desc} \geq \beta$ and $\#\text{child} \geq \beta$, that given a pointed forest (\mathcal{F}, t, n) bound from below the number of descendants and children of the current evaluation node n , provided that n is a descendant of the target node t .

Let (\mathcal{F}, t, n) be a pointed forest. Given $\beta \in \mathbb{N}$, we start by defining the formula $\text{size}(\text{Miss}) \geq \beta$ stating that \mathcal{F} contains at least β miss nodes, that is:

$$(\mathcal{F}, t, n) \models \text{size}(\text{Miss}) \geq \beta \text{ if and only if } \text{card}(\mathcal{F}[\text{Miss}]) \geq \beta.$$

This formula is inductively defined below:

$$\begin{aligned} \text{size}(\text{Miss}) \geq 0 &\stackrel{\text{def}}{=} \top, \\ \text{size}(\text{Miss}) \geq \beta + 1 &\stackrel{\text{def}}{=} \langle U \rangle (\text{Miss} \wedge \underbrace{\Diamond(\neg \text{inDom} \wedge \text{size}(\text{Miss}) \geq \beta)}_{\text{by excluding a miss node, at least other } \beta \text{ miss nodes can be found}}). \end{aligned}$$

Let us consider for a moment the definition of $\text{size}(\text{Miss}) \geq \beta + 1$. Informally, this formula is satisfied if it is possible to find a node in $\mathcal{F}[\text{Miss}]$ (as expressed by the “ $\langle U \rangle (\text{Miss} \wedge \dots)$ ” part of the formula), removing it from the model (as done by the “ $\Diamond(\neg \text{inDom} \dots)$ ” part), and then find other β elements of $\mathcal{F}[\text{Miss}]$. This formula essentially works because the set of miss nodes monotonically decreases when considering subforests, i.e. given $\mathcal{F}' \subseteq \mathcal{F}$ we have $\mathcal{F}'[\text{Miss}] \subseteq \mathcal{F}[\text{Miss}]$. Hence, finding a miss node in the subforest \mathcal{F}' implies finding a miss node in the original forest \mathcal{F} . This idea is generalisable to similar monotonous properties. Let us extend our notation and, given a formula φ of ALT and a pointed forest (\mathcal{F}, t, n) , write $\mathcal{F}[\varphi]_t$ for the set $\{n' \in \mathcal{N} \mid (\mathcal{F}, t, n') \models \varphi\}$. Moreover, given $\beta \in \mathbb{N}$ we inductively define the formula $\text{size}(\varphi) \geq \beta$ that bounds from below the amount of nodes satisfying φ , provided that φ satisfies some monotonic property formally defined below in Lemma 4.7. $\text{size}(\varphi) \geq \beta$ is defined by simply replacing Miss by φ in $\text{size}(\text{Miss}) \geq \beta$:

$$\begin{aligned} \text{size}(\varphi) \geq 0 &\stackrel{\text{def}}{=} \top, \\ \text{size}(\varphi) \geq \beta + 1 &\stackrel{\text{def}}{=} \langle U \rangle (\varphi \wedge \underbrace{\Diamond(\neg \text{inDom} \wedge \text{size}(\varphi) \geq \beta)}_{\text{by excluding a node in } \mathcal{F}[\varphi], \text{ at least other } \beta \text{ such nodes can be found}}). \end{aligned}$$

The formal semantics of $\text{size}(\varphi) \geq \beta$ is provided in the lemma below.

Lemma 4.7. Let (\mathcal{F}, t, n) be a pointed forest. Let φ be a formula with the following properties:

1. $\mathcal{F}[\varphi]_t \subseteq \text{dom}(\mathcal{F})$, i.e. the set of nodes satisfying φ is a subset of the domain of the forest,
2. for every $\mathcal{F}' \subseteq \mathcal{F}$, if $\text{dom}(\mathcal{F}) \setminus \text{dom}(\mathcal{F}') \subseteq \mathcal{F}[\varphi]_t$ then $\mathcal{F}[\varphi]_t \cap \text{dom}(\mathcal{F}') = \mathcal{F}'[\varphi]_t$.

Given $\beta \in \mathbb{N}$, we have $(\mathcal{F}, t, n) \models \text{size}(\varphi) \geq \beta$ if and only if $\text{card}(\mathcal{F}[\varphi]_t) \geq \beta$.

Before proving this lemma, let us look at the property (2) of φ . Consider a partition $\{S, T\}$ of the nodes in $\mathcal{F}[\varphi]_t$, and the subforest $\mathcal{F}' \subseteq \mathcal{F}$ such that $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus S$. Property (2) states that then $\mathcal{F}'[\varphi]_t = T$. Informally, this means that removing nodes in $\mathcal{F}[\varphi]_t$ does not change the set of nodes in the domain of the forest that still satisfy φ . This property, as well as property (1), holds for the case of $\varphi = \text{Miss}$, so that Lemma 4.7 implies the correctness of $\text{size}(\text{Miss}) \geq \beta$.

Proof. The proof is by induction on β , over the domain of natural numbers. The base case where $\beta = 0$ is direct, so let us consider the inductive case for $\beta = \beta' + 1$ where $\beta' \in \mathbb{N}$.

(\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \text{size}(\varphi) \geq \beta' + 1$, and therefore there is a node $n' \in N$ such that

$$\text{A. } (\mathcal{F}, t, n') \models \varphi, \quad \text{B. } (\mathcal{F}, t, n') \models \Diamond(\neg \text{inDom} \wedge \text{size}(\varphi) \geq \beta').$$

From (B), there is a forest $\mathcal{F}' \subseteq \mathcal{F}$ such that

$$\text{C. } \text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1, \quad \text{D. } (\mathcal{F}', t, n') \models \neg \text{inDom}, \quad \text{E. } (\mathcal{F}', t, n') \models \text{size}(\varphi) \geq \beta'.$$

First, let us prove that $\mathcal{F}[\varphi]_t = \mathcal{F}'[\varphi]_t \cup \{n'\}$ and $n' \notin \mathcal{F}'[\varphi]_t$. From (A) and the property (1) on φ , we have $n' \in \text{dom}(\mathcal{F})$. From (C) and (D), this means that $\text{dom}(\mathcal{F}) \setminus \text{dom}(\mathcal{F}') = \{n'\}$, which allows us to conclude that $\mathcal{F}[\varphi]_t \cap \text{dom}(\mathcal{F}') = \mathcal{F}'[\varphi]_t$, directly from the property (2) of φ . This implies that $\mathcal{F}[\varphi]_t = \mathcal{F}'[\varphi]_t \cup \{n'\}$ and $n' \notin \mathcal{F}'[\varphi]_t$. We now use this fact to show that the properties (1) and (2) of φ hold with respect to the forest \mathcal{F}' , so that we can then apply the induction hypothesis directly by (E). The property (1), i.e. $\mathcal{F}'[\varphi]_t \subseteq \text{dom}(\mathcal{F}')$, follows directly from $\mathcal{F}[\varphi]_t \cap \text{dom}(\mathcal{F}') = \mathcal{F}'[\varphi]_t$. For the property (2), let \mathcal{F}'' be a forest such that $\mathcal{F}'' \subseteq \mathcal{F}'$ and $\text{dom}(\mathcal{F}') \setminus \text{dom}(\mathcal{F}'') \subseteq \mathcal{F}'[\varphi]_t$. Let us prove that $\mathcal{F}'[\varphi]_t \cap \text{dom}(\mathcal{F}'') = \mathcal{F}''[\varphi]_t$. From $\mathcal{F}' \subseteq \mathcal{F}$ it holds that $\mathcal{F}'' \subseteq \mathcal{F}$. Moreover,

$$\begin{aligned} \text{dom}(\mathcal{F}) \setminus \text{dom}(\mathcal{F}'') &= (\text{dom}(\mathcal{F}') \cup \{n'\}) \setminus \text{dom}(\mathcal{F}'') && (\text{by } \text{dom}(\mathcal{F}) = \text{dom}(\mathcal{F}') \cup \{n'\}) \\ &= (\text{dom}(\mathcal{F}') \setminus \text{dom}(\mathcal{F}'')) \cup \{n'\} && (\text{by } n' \notin \text{dom}(\mathcal{F}') \text{ and } \mathcal{F}'' \subseteq \mathcal{F}') \\ &\subseteq \mathcal{F}'[\varphi]_t \cup \{n'\} && (\text{by } \text{dom}(\mathcal{F}') \setminus \text{dom}(\mathcal{F}'') \subseteq \mathcal{F}'[\varphi]_t) \\ &= \mathcal{F}[\varphi]_t && (\text{by } \mathcal{F}[\varphi]_t = \mathcal{F}'[\varphi]_t \cup \{n'\}) \end{aligned}$$

Therefore, by property (2) (w.r.t. \mathcal{F}), $\mathcal{F}[\varphi]_t \cap \text{dom}(\mathcal{F}'') = \mathcal{F}''[\varphi]_t$ holds, which is equivalent to $(\mathcal{F}'[\varphi]_t \cup \{n'\}) \cap \text{dom}(\mathcal{F}'') = \mathcal{F}''[\varphi]_t$. Lastly, from $n' \notin \text{dom}(\mathcal{F}')$ and $\mathcal{F}'' \subseteq \mathcal{F}'$, we conclude that $\mathcal{F}'[\varphi]_t \cap \text{dom}(\mathcal{F}'') = \mathcal{F}''[\varphi]_t$, completing the proof of property (2) w.r.t. \mathcal{F}' . This allows us to use the induction hypothesis and conclude from (E) that $\text{card}(\mathcal{F}'[\varphi]_t) \geq \beta'$. This is sufficient to also conclude that $\text{card}(\mathcal{F}[\varphi]_t) \geq \beta' + 1$, as we have already shown that $\mathcal{F}[\varphi]_t = \mathcal{F}'[\varphi]_t \cup \{n'\}$ and $n' \notin \mathcal{F}'[\varphi]_t$. We leave the proof of the other direction to the reader. \square

The formula $\text{size}(\varphi) \geq \beta$ is not only useful as it can be quickly instantiated to define various interesting formulae in ALT, but also because it shows a suitable way of reasoning in ALT. Roughly speaking, we often use the somewhere modality $\langle U \rangle$ to find a node in the domain of the forest that satisfies a certain property. Afterwards, we remove it with the sabotage operator \Diamond , in order to check if the resulting subforest satisfies a second property.

We can already make use of the formula $\text{size}(\varphi) \geq \beta$ in order to define a formula that checks whether the number of children of the target node is at least β . It is sufficient to notice that such a child can be characterised with the formula $\text{tchild} \stackrel{\text{def}}{=} \text{Hit} \wedge \neg \Diamond \text{Miss}$, and that this formula satisfies both the properties (1) and (2) of Lemma 4.7. This leads to the following result.

Lemma 4.8. $(\mathcal{F}, t, n) \models \text{size}(\text{tchild}) \geq \beta$ if and only if t has at least β children.

Proof. In order to prove this result it is sufficient to show that $(\mathcal{F}, t, n) \models \text{tchild}$ holds if and only if n is a child of t , and that tchild satisfies the properties (1) and (2) of Lemma 4.7. For

the correctness of tchild , simply notice that if the current evaluation node n is a child of the target node t , then the same holds in every subforest $\mathcal{F}' \subseteq \mathcal{F}$ such that $n \in \text{dom}(\mathcal{F}')$. Thus, (\mathcal{F}', t, n) cannot satisfy Miss . Otherwise, if n is a descendant of t but not one of its children, removing $(n, \mathcal{F}(n))$ from the forest makes n a miss node, hence tchild is not satisfied.

The property (1), i.e. $\mathcal{F}[\text{tchild}]_t \subseteq \text{dom}(\mathcal{F})$, holds from the tautology $\models \text{Hit} \Rightarrow \text{inDom}$. To prove the property (2), it is sufficient to see that the following (stronger) statement holds:

$$\text{for every } \mathcal{F}' \subseteq \mathcal{F}, \mathcal{F}[\text{tchild}]_t \cap \text{dom}(\mathcal{F}') = \mathcal{F}'[\text{tchild}]_t.$$

Showing this statement is straightforward, as a \mathcal{F} -child of t that is in the domain of \mathcal{F}' is by definition a \mathcal{F}' -child of t , and vice versa. \square

Let us now move to the definition of $\#\text{desc} \geq \beta$, the formula stating that the current evaluation node is a hit node with at least β descendants. It is defined as follows:

$$\#\text{desc} \geq \beta \stackrel{\text{def}}{=} \diamond^* (\underbrace{[U] \neg \text{Miss} \wedge \text{Hit}}_{\mathcal{F}[\text{Miss}] \text{ is empty}} \wedge \underbrace{\diamond (-\text{inDom} \wedge \text{size}(\text{Miss}) \geq \beta)}_{\text{removing } n \text{ lead to at least } \beta \text{ miss nodes}}).$$

The proof of correctness of this formula is given in Lemma 4.9. Intuitively, given a pointed forest (\mathcal{F}, t, n) where n is a descendant of t , this formula uses the fact that removing $(n, \mathcal{F}(n))$ from the forest \mathcal{F} makes all its descendant miss nodes. The repeated sabotage \diamond^* is used to remove the miss nodes before removing $(n, \mathcal{F}(n))$, so that then the formula $\text{size}(\text{Miss}) \geq \beta$ can be used to correctly count the descendants of n in \mathcal{F} .

Thanks to the formula $\#\text{desc} \geq \beta$ we are able to define the formula $\#\text{child} \geq \beta$ that checks the number of children of the current evaluation node n (assuming that n is a hit node):

$$\begin{aligned} \#\text{child} \geq 0 &\stackrel{\text{def}}{=} \text{Hit}, \\ \#\text{child} \geq \beta + 1 &\stackrel{\text{def}}{=} \#\text{desc} \geq \beta + 1 \wedge \underbrace{\blacksquare^\beta (\text{Hit} \Rightarrow \#\text{desc} \geq 1)}_{\text{whenever } \beta \text{ nodes of } \text{dom}(\mathcal{F}) \text{ are removed, if } n \text{ still reaches } t \text{ then it has at least one descendant}}. \end{aligned}$$

Informally, for a pointed forest (\mathcal{F}, t, n) , this formula express that n has $\beta \geq 1$ children by stating that the removal of $\beta - 1$ edges from \mathcal{F} cannot lead to a subforest where n has no descendants. The following lemma evaluates the correctness of $\#\text{desc} \geq \beta$ and $\#\text{child} \geq \beta$. Its proof can be found in Appendix B.

Lemma 4.9. Let (\mathcal{F}, t, n) be a pointed forest. Then,

- (I) $(\mathcal{F}, t, n) \models \#\text{desc} \geq \beta$ iff n has at least β descendants and it is a descendant of t .
- (II) $(\mathcal{F}, t, n) \models \#\text{child} \geq \beta$ iff n has at least β children and it is a descendant of t .

Given a syntactical element $S \in \{\text{size}(\varphi), \#\text{desc}, \#\text{child}\}$, we write $S = \beta$ for the formula $S \geq \beta \wedge \neg S \geq \beta + 1$. For instance, $\#\text{child} = \beta$ is the formula that states whether n has exactly β children and it is a descendant of t . We can now conclude the encoding of finite words.

Characterising words in ALT. We now move to the definition of the formula word_Σ that characterises the class of forests encoding words in Σ^* . Recall that we assume Σ to be the alphabet of natural numbers in $[1, n]$, for some $n \geq 1$. Let (\mathcal{F}, t, n) be a pointed forest encoding the word $w = a_1 \dots a_k$, and let $M = (n_1, \dots, n_k)$ be the set of its main nodes. Let us recall two of the properties of our encoding, expressed in Proposition 4.5, and introduce suitable formulae to express these properties. First, a node n encodes a symbol of w (i.e. it is a main node) if it is a hit node with at least one child (Property (I)). To better reflect this property, we write symb

for the formula $\#\text{child} \geq 1$, so that n encodes a symbol of w if and only if $(\mathcal{F}, t, n) \models \text{symb}$. Among the main nodes, n_1 is the only one having the same number of descendants and children (Property (II)). For this property, given $S \subseteq \Sigma$, we introduce the formula 1st_S that checks if the current evaluation node n corresponds to n_1 and encodes a symbol in S :

$$\text{1st}_S \stackrel{\text{def}}{=} \bigvee_{\beta \in S} (\#\text{desc} = \beta + 1 \wedge \#\text{child} = \beta + 1).$$

We are finally ready to define the formula word_Σ that characterise the class of forests that encode words in Σ^* , for $\Sigma = [1, n]$. It is defined as follows, and it is proved correct in Lemma 4.10,

The target node has no descendants, or has a descendant that encodes a symbol.

$$\begin{aligned} \text{word}_\Sigma &\stackrel{\text{def}}{=} \neg \text{size}(\text{tchild}) \geq 2 \wedge \overbrace{(\langle U \rangle \text{Hit} \Rightarrow \langle U \rangle \text{symb})} \\ &\quad \wedge [U](\text{symb} \Rightarrow \text{1st}_\Sigma \vee \underbrace{(\neg \text{1st}_{\{n+1\}} \wedge \blacklozenge \text{1st}_\Sigma)}). \end{aligned}$$

the current node encodes a symbol in $[1, n]$ and exactly one of its children encodes a symbol.

The first two conjuncts of the formula word_Σ are quite self-explanatory. First, the target node t has at most one child. Second, if w is the empty word then the forest does not contain hit nodes (alternatively, t does not have children), and otherwise there is a hit node encoding a symbol. The last conjunct is more complex, and subsumes the four properties in Proposition 4.5. Let n' be a node such that $(\mathcal{F}, t, n') \models \text{symb}$. From the property (I) this means that n' is a main node. If it is the first node in the main path, then from the property (II) it must have the same number of descendants and children, and it must have $a + 1$ children for some $a \in \Sigma$ (Property (IV)). Basically, n' must satisfy 1st_Σ . Otherwise, suppose that n' encodes a node in the main path that is different from the first one. From the property (III), exactly one of its children, say n'' , must encode a symbol, whereas the other children are $a + 1$ character nodes, for some $a \in \Sigma$ (again, from the property (IV)). This means that removing (n'', n') from \mathcal{F} makes the node n' be the first node in the main path, according to property (II). So, n' satisfies $\neg \text{1st}_{\{n+1\}} \wedge \blacklozenge \text{1st}_\Sigma$. We prove that word_Σ characterise the class of forests encoding words in Σ^* .

Lemma 4.10. A pointed forest (\mathcal{F}, t, n) is an encoding of a word in Σ^* iff $(\mathcal{F}, t, n) \models \text{word}_\Sigma$.

Proof. (\Rightarrow): Suppose (\mathcal{F}, t, n) be a pointed forest encoding the word $w = a_1 \dots a_k \in [1, n]^*$, where $n \geq 1$. Let $M = (n_1, \dots, n_k)$ and $C = (N_1, \dots, N_k)$ be the main nodes and character nodes of (\mathcal{F}, t, n) , respectively. Recapitulating Definition 4.4:

1. $\{n_1, \dots, n_k\}, N_1, \dots, N_k, \mathcal{F}[\text{Miss}]_t$ are pairwise disjoint sets, i.e. they do not share any node,
2. M and C are all the descendants of t , i.e. $(\mathcal{F}^{-1})^+(t) = \{n_1, \dots, n_k\} \cup \bigcup_{j \in [1, k]} N_j$,
3. n_k is the only child of t and for every $j \in [1, k - 1]$ $\mathcal{F}(n_j) = n_{j+1}$,
4. for every $j \in [1, k]$, $\text{card}(N_j) = a_j + 1$ and for every $n' \in N_j$, $\mathcal{F}(n') = n_j$.

Notice that (2) implies that if w is the empty word, then \mathcal{F} does not have hit nodes. If this is the case, then $(\mathcal{F}, t, n) \models [U] \neg \text{Hit}$, which implies that $(\mathcal{F}, t, n) \models \langle U \rangle \text{Hit} \Rightarrow \langle U \rangle \text{symb}$. Otherwise, the set of main nodes is non-empty, and by (3) and (4) we conclude that each main node is a descendant of t and has at least one child (as stated in Proposition 4.5). Again, this implies the satisfaction of $(\mathcal{F}, t, n) \models \langle U \rangle \text{Hit} \Rightarrow \langle U \rangle \text{symb}$. From (3) we have $(\mathcal{F}, t, n) \models \neg \text{size}(\text{tchild}) \geq 2$, leaving us with only the last conjunct of word_Σ being open. Let us consider a node n' such that $(\mathcal{F}, t, n') \models \text{symb}$. In particular, this implies that a main node exists and so w is not empty. By (3) and (4), n' is a main node and so there is $j \in [1, k]$ such that $n' = n_j$. If $j = 1$, we prove that $(\mathcal{F}, t, n') \models \text{1st}_\Sigma$. In this case, every child of n' is a character node from N_1 (and vice versa), which in turn does not have any children, so that n' has the same number of descendants

and children (as stated in Proposition 4.5). Moreover, (4) implies that $\text{card}(N_1) = a_1 + 1$. Thus, $(\mathcal{F}, t, n') \models \#\text{desc} = a_1 + 1 \wedge \#\text{child} = a_1 + 1$, i.e one of the disjuncts of 1st_Σ . Otherwise, consider the case where $j \neq 1$. Let us prove that $(\mathcal{F}, t, n') \models \neg 1\text{st}_{\{n+1\}} \wedge \blacklozenge 1\text{st}_\Sigma$. Exactly one child of n' is a main node (i.e. n_{j-1}), whereas all other children are character nodes. As n_{j-1} is a main node, it has at least one child. So, $(\mathcal{F}, t, n') \models \neg 1\text{st}_{\{n+1\}}$. Let $\mathcal{F}' \subseteq \mathcal{F}$ be the subforest such that $\mathcal{F}' = \mathcal{F} \setminus \{(n_{j-1}, n')\}$. On this subforest, all the \mathcal{F}' -children of n' are character nodes, more specifically (4) states that these children are the $a_j + 1$ nodes from N_j . As in the case of $j = 1$, this implies that $(\mathcal{F}', t, n') \models \#\text{desc} = a_j + 1 \wedge \#\text{child} = a_j + 1$, i.e a disjunct of 1st_Σ . Thus, $(\mathcal{F}, t, n') \models \blacklozenge 1\text{st}_\Sigma$.

(\Leftarrow): Conversely, suppose $(\mathcal{F}, t, n) \models \text{word}_\Sigma$. From the first two conjuncts of word_Σ we have:

B. t has at most one child (from $\neg \text{size}(\text{tchild}) \geq 2$),

C. If \mathcal{F} has a hit node, one descendant of t has a child (from $\langle U \rangle \text{Hit} \Rightarrow \langle U \rangle \text{symb}$).

Notice that if \mathcal{F} does not have descendants then it trivially encodes the empty word. So, let us assume that $\mathcal{F} \neq \emptyset$. From (B), t has exactly one child, say \bar{n} . Together with (C), this means that $(\mathcal{F}, t, \bar{n})$ must have a child (every other descendant of t is a descendant of \bar{n}). We define the following subsets of the descendants of t :

- $M \stackrel{\text{def}}{=} \{n' \in (\mathcal{F}^{-1})^+(t) \mid \mathcal{F}(n'') = n' \text{ for some } n'' \in N\}$, i.e. the *non-leaf* descendants of t ,
- for $n' \in M$, $N_{n'} \stackrel{\text{def}}{=} \{n'' \notin M \mid \mathcal{F}(n'') = n'\}$, i.e. the *leafs* descendants of t that are children of n' .

Notice that \bar{n} belongs to M . Besides, the nodes in M are the only ones that satisfy symb (as they have a child and are descendants of t). To prove that \mathcal{F} encodes a word in $[1, n]^+$ we show:

- I. for each node $n' \in M$ there is at most one node $n'' \in M$ such that $\mathcal{F}(n'') = n'$. This shows the existence of a main path in the tree, made by the elements in M ;
- II. for every $n' \in M$, $\text{card}(N_{n'}) \in [2, n+1]$. This shows that nodes of M encode symbols in $[1, n]$.

To prove (I) and (II), we use the fact that (\mathcal{F}, t, n) satisfies the last conjunct of word_Σ , i.e.

$$[U](\text{symb} \Rightarrow 1\text{st}_\Sigma \vee (\neg 1\text{st}_{\{n+1\}} \wedge \blacklozenge 1\text{st}_\Sigma)).$$

Let us consider $n' \in M$. As it satisfies symb , we have $(\mathcal{F}, t, n') \models 1\text{st}_\Sigma \vee (\neg 1\text{st}_{\{n+1\}} \wedge \blacklozenge 1\text{st}_\Sigma)$. If $(\mathcal{F}, t, n') \models 1\text{st}_\Sigma$, as $\Sigma = [1, n]$ we conclude that the number of children and descendants of n are equal, and take a value in $[2, n+1]$. This implies that is no $n'' \in M$ such that $\mathcal{F}(n'') = n'$ and $\text{card}(N_n) \in [2, n+1]$. In this case, both (I) and (II) are verified. Otherwise, let us suppose that $(\mathcal{F}, t, n') \models \neg 1\text{st}_{\{n+1\}} \wedge \blacklozenge 1\text{st}_\Sigma$. In order to show that both (I) and (II) hold (concluding the proof), we reason by contradiction. First, suppose *ad absurdum* that (I) does not hold and so there are two distinct nodes $n'', n''' \in M$ s.t. $\mathcal{F}(n'') = n' = \mathcal{F}(n''')$. As n'' and n''' are both in M , they both have at least one child. However, this implies that $(\mathcal{F}, t, n') \not\models \blacklozenge 1\text{st}_\Sigma$, leading to a contradiction. Indeed, $\blacklozenge 1\text{st}_\Sigma$ is satisfied if it is possible to remove a single edge from the forest \mathcal{F} , leading to a finite forest \mathcal{F}' , so that n' is a \mathcal{F}' -descendant of t and the number of its children coincide with the number of its descendants. So, (I) holds. Lastly, suppose *ad absurdum* that (II) does not hold, and so $\text{card}(N_{n'}) \notin [2, n+1]$. It is helpful to realise that the formula

$$\blacklozenge 1\text{st}_\Sigma \Rightarrow (\#\text{child} \geq 2 \wedge \#\text{child} \leq n+2),$$

is valid (recall that $\Sigma = [1, n]$). Indeed, consider a model $(\mathcal{F}_*, t_*, n_*)$ that satisfies $\blacklozenge 1\text{st}_\Sigma$. By definition, there is a edge $e \in \mathcal{F}_*$ such that $\mathcal{F}'_* \stackrel{\text{def}}{=} \mathcal{F}_* \setminus \{e\}$ enjoys $(\mathcal{F}'_*, t_*, n_*) \models 1\text{st}_\Sigma$. This implies $(\mathcal{F}'_*, t_*, n_*) \models \#\text{child} = \beta$ for some $\beta \in [2, n+1]$, which in turn means that in \mathcal{F}_* , n_* can only have between 2 and $n+2$ \mathcal{F}_* -children.

From this tautology we conclude that n' has between 2 and $n+2$ \mathcal{F} -children. If one of these children belongs to M , then of course $\text{card}(N_{n'}) \in [2, n+1]$, proving (II). If instead every child of

n' belongs to $N_{n'}$, then n' has the same number of descendants and children. Moreover, from the assumption $\text{card}(N_{n'}) \notin [2, n+1]$, we derive $\text{card}(N_{n'}) = n+2$. However, this is contradictory with the fact that $(\mathcal{F}, t, n') \models \neg \text{1st}_{\{n+1\}}$. Thus, (II) holds. \square

4.2.2 Intermezzo: inexpressibility results via Ehrenfeucht-Fraïssé games.

Now that we are more familiar with the logic, before moving to the TOWER-hardness proof of the satisfiability problem for ALT, it is helpful to see some of the properties that ALT cannot express. Notably, these properties give us some insight on what we should do (or rather, what we should not do) in order to build very expressive queries in ALT in a concise way, as we definitely need in order to reach TOWER-hardness. In particular, we show that the expressive power of ALT is very weak when it comes to expressing properties of miss nodes. On these nodes, ALT can essentially only state the properties captured by Boolean combinations of the formulae **Miss** and **size(Miss)** $\geq \beta$. Therefore, the expressive power of ALT is almost entirely concerned with hit nodes. On the other hand, inexpressibility results effectively reduce the set of forests that must be considered in order to solve the satisfiability problem. This in turn makes reductions from this problem to the satisfiability of other logics more immediate, as we show throughout Section 4.4. Readers that are eager to see the TOWER-hardness of the satisfiability problem for ALT can jump to Section 4.3 (page 97).

Various mathematical tools from model theory are suited to prove inexpressibility results, as for example compactness theorems, Löwenheim-Skolem theorem and Ehrenfeucht-Fraïssé games. However, when dealing with logics interpreted on finite structures, Ehrenfeucht-Fraïssé games are the only major tool available. This is the case for ALT. Without extensively discussing the other tools and why they fail on finite structures (a clear presentation is given in [102]), let us briefly recall the *compactness* theorem for a logic \mathcal{L} interpreted on the class of structures \mathcal{M} .

Theorem 4.11 (Compactness). Let S be a set of formulae in \mathcal{L} . S has a model from \mathcal{M} (i.e. S is consistent w.r.t. \mathcal{M}) if and only if every finite subset of S has a model from \mathcal{M} .

In the case that this theorem holds for ALT, it can be used to prove that a certain subclass C of pointed forests is not definable in the logic as follows. We first assume that C is characterised by a formula φ_C . We construct an infinite set of formulae S such that every finite subset of $S \cup \{\varphi_C\}$ is consistent, whereas the full set $S \cup \{\varphi_C\}$ is inconsistent. However, this contradicts the compactness theorem, and therefore C cannot be characterised in ALT.

Unfortunately, we cannot rely on this technique, as Theorem 4.11 does not hold for ALT. Indeed, consider the infinite set of formulae $S \stackrel{\text{def}}{=} \{\blacklozenge^k \top \mid k \in \mathbb{N}\}$. It is clear that every finite subset $T \subseteq_{\text{fin}} S$ is satisfied by every pointed forest (\mathcal{F}, t, n) such that $\text{card}(\mathcal{F}) \geq \max\{k \mid \blacklozenge^k \top \in T\}$. However, S can only be satisfied by an infinite forest, and is therefore inconsistent with respect to the class of pointed forests. This invalidates Theorem 4.11.

EF-games. As compactness fails, to prove inexpressibility results for ALT we adapt the notion of *Ehrenfeucht-Fraïssé games* (EF-games, in short) of first-order logic [102]. This has already been done for other relation-changing logics such as context logic for trees [36] and ambient logic [48]. EF-games are two players games. One player is called the *spoiler* and the other is called the *duplicator*. In the case of ALT, a game is played on a *state* that is represented by a triple $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), rk)$ made of two pointed forests $(\mathcal{F}_1, t_1, n_1)$ and $(\mathcal{F}_2, t_2, n_2)$, and a rank rk . The rank, to be formally defined below, roughly represents the numbers of turns in the

EF-Game played on the state $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), (m, s, k))$

if there is $\pi \in \{\text{Miss}, \text{Hit}\}$ such that $((\mathcal{F}_1, t_1, n_1) \models \pi \text{ iff } (\mathcal{F}_2, t_2, n_2) \models \pi)$ does not hold
then the spoiler wins,

else the spoiler chooses $i \in \{1, 2\}$ and plays on $(\mathcal{F}_i, t_i, n_i)$.

The duplicator replies on $(\mathcal{F}_j, t_j, n_j)$ where $j \in \{1, 2\} \setminus \{i\}$.

The spoiler **must** choose one of the following moves (otherwise the duplicator wins).

$\langle U \rangle$ move: if $m \geq 1$ then the spoiler **can** choose to play a $\langle U \rangle$ move. If he does so,

1. The spoiler selects a node $n'_i \in \mathcal{N}$.
2. The duplicator **must** select a node $n'_j \in \mathcal{N}$ (otherwise the spoiler wins).
3. The game continues on $((\mathcal{F}_1, t_1, n'_1), (\mathcal{F}_2, t_2, n'_2), (m-1, s, k))$.

\blacklozenge move: if $s \geq 1$ and $\text{dom}(\mathcal{F}_i) \neq \emptyset$ then the spoiler **can** choose to play a \blacklozenge move.

1. The spoiler selects a finite forest $\mathcal{F}'_i \subseteq \mathcal{F}_i$ such that $\text{card}(\mathcal{F}'_i) = \text{card}(\mathcal{F}_i) - 1$.
2. The duplicator **must** reply with a forest $\mathcal{F}'_j \subseteq \mathcal{F}_j$ such that $\text{card}(\mathcal{F}'_j) = \text{card}(\mathcal{F}_j) - 1$.
3. The game continues on $((\mathcal{F}'_1, t_1, n_1), (\mathcal{F}'_2, t_2, n_2), (m, s-1, k))$.

\blacklozenge^* move: if $k \geq 1$ then the spoiler **can** choose to play a \blacklozenge^* move.

1. The spoiler selects a finite forest $\mathcal{F}'_i \subseteq \mathcal{F}_i$.
2. The duplicator **must** reply with a finite forest $\mathcal{F}'_j \subseteq \mathcal{F}_j$.
3. The game continues on $((\mathcal{F}'_1, t_1, n_1), (\mathcal{F}'_2, t_2, n_2), (m, s, k-1))$.

Figure 4.4: Ehrenfeucht-Fraïssé games for ALT.

game. At each turn, the spoiler performs a *move* in one of the two pointed forests, which must be countered by the duplicator with a move on the other pointed forest. These moves are related to ALT, as they capture the semantics of the three modalities $\langle U \rangle$, \blacklozenge and \blacklozenge^* . The goal of the spoiler is to show that the two structures are different. The goal of the duplicator is to show that the two structures are similar. The notion of *being different* also traces back to the semantics of ALT: two pointed forests are different if and only if there is a formula of ALT that it is satisfied by only one of the two. The exact correspondence between the games and ALT is formalised with an adequacy result (Theorem 4.15, below). A player has a *winning strategy* if it can play in a way that guarantees it the victory, regardless what the other player does. We write $(\mathcal{F}_1, t_1, n_1) \sim_{rk} (\mathcal{F}_2, t_2, n_2)$ whenever the duplicator has a winning strategy for the game $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), rk)$. As we will see, our games are determined: if the duplicator does not have a winning strategy then spoiler has one, and vice versa. Hence, we write $(\mathcal{F}_1, t_1, n_1) \not\sim_{rk} (\mathcal{F}_2, t_2, n_2)$ to state that the spoiler has a winning strategy. Albeit determinacy holds directly from Zermelo's Theorem [145] (or Martin's Theorem [109]), for the simple games of ALT we prefer to derive it as a self-contained result (Lemma 4.14).

In order to introduce the games, we need to define the rank of a formula φ in ALT.

Definition 4.12 (Rank). The *rank* of φ a triple $(m, s, k) \in \mathbb{N}^3$ where the *modal rank* m is the greatest nesting depth of the modal operator $\langle U \rangle$ in φ , whereas the *sabotage rank* s (resp. *repeated sabotage rank* k) is the greatest nesting depth of the operator \blacklozenge (resp. \blacklozenge^*) in φ .

We write ALT_{rk} for the set of formulae with rank $rk \in \mathbb{N}^3$. We define the rank order $<_{rk}$ on \mathbb{N}^3 .

Definition 4.13 (Rank order). The *rank order* $<_{rk} \subseteq \mathbb{N}^3 \times \mathbb{N}^3$ is the relation defined as

$$(m, s, k) <_{rk} (m', s', k') \text{ iff } m \leq m', s \leq s', k \leq k' \text{ and } (m < m' \text{ or } s < s' \text{ or } k < k').$$

Notice that $<_{rk}$ is a well-founded strict order.

The EF-games for **ALT** are played with respect to a rank $rk \in \mathbb{N}^3$, and they are formally defined in Figure 4.4. As we can see, at the beginning of the turn, we check whether the two atomic formulae **Hit** and **Miss** are satisfied by only one of the two pointed forests $(\mathcal{F}_1, t_1, n_1)$ and $(\mathcal{F}_2, t_2, n_2)$. If this is the case, the spoiler wins. Otherwise, it must choose one move, among three possibilities which essentially capture the semantics of the modalities of **ALT**. The determinacy of the EF-games can be easily proven by induction on the rank of the game.

Lemma 4.14. For every state of the game, one of the two players has a winning strategy.

Proof. The proof is by induction on the rank rk of the game, with respect to the order $<_{rk}$.

base case: $rk = (0, 0, 0)$. Let us consider a games state $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), rk)$. If there is $\pi \in \{\text{Miss}, \text{Hit}\}$ such that $((\mathcal{F}_1, t_1, n_1) \models \pi \text{ iff } (\mathcal{F}_2, t_2, n_2) \models \pi)$ does not hold, then the spoiler wins (1st and 2nd line of Figure 4.4). Otherwise, since $rk = (0, 0, 0)$ the spoiler cannot perform any move and the duplicator wins (5th line in Figure 4.4).

induction step: $rk \neq (0, 0, 0)$. Let us consider a state $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), rk)$. Again, if there is $\pi \in \{\text{Miss}, \text{Hit}\}$ such that $((\mathcal{F}_1, t_1, n_1) \models \pi \text{ iff } (\mathcal{F}_2, t_2, n_2) \models \pi)$ does not hold, then the spoiler wins (1st and 2nd line of Figure 4.4). Otherwise, the spoiler can perform a move according to Figure 4.4, to which a move for the duplicator follows. Given a move of the spoiler, let S be the set of games states that can be reached following a possible answer from the duplicator. Each of these states has rank $rk' <_{rk} rk$. By induction hypothesis, either the spoiler or the duplicator has a winning strategy for each state in S . If the duplicator has a winning strategy for one of the states in S , then it can answer the move done by the spoiler so that the game continue on that state, ensuring a victory. Otherwise, no matter what is the answer of the duplicator, the spoiler has a winning strategy after selecting that particular move. This means that, for every move of the spoiler, the game proceeds in a state for which one of the two players has a winning strategy. If for every move of the spoiler the game proceeds in a state for which the duplicator has a winning strategy, then the duplicator has a winning strategy for $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), rk)$. If instead there is a move of the spoiler that makes the game proceed in a state for which the spoiler has a winning strategy, then it is sufficient for the spoiler to perform that move in order to produce a winning strategy for $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), rk)$. \square

We now aim at connecting the EF-games with **ALT** by proving that they are adequate with respect to the satisfaction relation \models of **ALT**. In particular, we want to show the following result.

Theorem 4.15. Let $(\mathcal{F}_1, t_1, n_1)$ and $(\mathcal{F}_2, t_2, n_2)$ be two pointed forests. Let $rk \in \mathbb{N}^3$.

- (I) If $(\mathcal{F}_1, t_1, n_1) \models \varphi$ and $(\mathcal{F}_2, t_2, n_2) \not\models \varphi$ for some φ in ALT_{rk} , then $(\mathcal{F}_1, t_1, n_1) \not\sim_{rk} (\mathcal{F}_2, t_2, n_2)$.
- (II) If for every φ in ALT_{rk} $((\mathcal{F}_1, t_1, n_1) \models \varphi \text{ iff } (\mathcal{F}_2, t_2, n_2) \models \varphi)$, then $(\mathcal{F}_1, t_1, n_1) \sim_{rk} (\mathcal{F}_2, t_2, n_2)$.

In order to show this theorem, which as we will see allows us to prove inexpressibility results for **ALT**, we need to state some of the properties of ranks. The first property is that ALT_{rk} is a finite set of formulae, up to logical equivalence.

Lemma 4.16. For each rank $rk \in \mathbb{N}^3$, ALT_{rk} is finite up to logical equivalence.

This result is rather standard, and similar ones can be found in [102, 36, 48]. Its proof (by induction on the rank rk) is left in Appendix B. Lemma 4.16 implies that given a rank rk , every pointed forest $(\mathcal{F}, \mathbf{t}, \mathbf{n})$ has a (finite) *characteristic formula* $\Gamma_{\text{rk}}(\mathcal{F}, \mathbf{t}, \mathbf{n}) \in \text{ALT}_{\text{rk}}$ that is logically equivalent to the infinite conjunction $\bigwedge \{\varphi \in \text{ALT}_{\text{rk}} \mid (\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \varphi\}$. Moreover, the formula $\Gamma_{\text{rk}}(\mathcal{F}, \mathbf{t}, \mathbf{n})$ enjoys the following properties.

Lemma 4.17. Let $(\mathcal{F}, \mathbf{t}, \mathbf{n})$ be a pointed forest and let $\text{rk} \in \mathbb{N}^3$. (I) $(\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \Gamma_{\text{rk}}(\mathcal{F}, \mathbf{t}, \mathbf{n})$, and (II) given a second pointed forest $(\mathcal{F}', \mathbf{t}', \mathbf{n}')$, $(\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \Gamma_{\text{rk}}(\mathcal{F}', \mathbf{t}', \mathbf{n}')$ iff $(\mathcal{F}', \mathbf{t}', \mathbf{n}') \models \Gamma_{\text{rk}}(\mathcal{F}, \mathbf{t}, \mathbf{n})$.

Proof. The statement (I) follows directly by the definition of characteristic formula. For the statement (II), by symmetry we just need to show one direction. Assume $(\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \Gamma_{\text{rk}}(\mathcal{F}', \mathbf{t}', \mathbf{n}')$. Let $\psi \in \text{ALT}_{\text{rk}}$ and suppose $(\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \psi$. To prove the result it is sufficient to show that $(\mathcal{F}', \mathbf{t}', \mathbf{n}') \models \psi$. *Ad absurdum*, suppose that $(\mathcal{F}', \mathbf{t}', \mathbf{n}') \not\models \psi$. Then by definition $(\mathcal{F}', \mathbf{t}', \mathbf{n}') \models \neg\psi$, and by definition of rank, we have that $\neg\psi \in \text{ALT}_{\text{rk}}$. Therefore, from the equivalence

$$\Gamma_{(m,s,k)}(\mathcal{F}', \mathbf{t}', \mathbf{n}') \stackrel{\text{by def}}{\equiv} \bigwedge \{\varphi \in \text{ALT}_{m,s,k} \mid (\mathcal{F}', \mathbf{t}', \mathbf{n}') \models \varphi\},$$

we conclude that $\models \Gamma_{(m,s,k)}(\mathcal{F}', \mathbf{t}', \mathbf{n}') \Rightarrow \neg\psi$. As $(\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \Gamma_{(m,s,k)}(\mathcal{F}', \mathbf{t}', \mathbf{n}')$, this implies $(\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \neg\psi$, in contradiction with the hypothesis $(\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \psi$. Hence, $(\mathcal{F}', \mathbf{t}', \mathbf{n}') \models \psi$. \square

Lemmata 4.16 and 4.17 allow us to prove Theorem 4.15 in a neat way. The statement (I) of Theorem 4.15, also called the *soundness* of the games, is proved by structural induction on φ . The *completeness* of the games, i.e. the statement (II) of Theorem 4.15, is proven by showing the contrapositive by induction on the rank and by cases on the first move that the spoiler makes in his winning strategy, which exists by determinacy (Lemma 4.14).

Proof of Theorem 4.15(I). The proof by structural induction on φ is similar to the one in [36].

base case: $\varphi \in \{\text{Hit}, \text{Miss}\}$. From the hypothesis $(\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}_1) \models \varphi$ and $(\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}_2) \not\models \varphi$ we conclude that spoiler wins the game (from the 1st and 2nd line of Figure 4.4).

For the induction steps, we omit the straightforward cases of Boolean connectives, and focus on the three cases $\varphi = \langle U \rangle \psi$, $\varphi = \blacklozenge \psi$ and $\varphi = \blacklozenge^* \psi$.

induction step: $\varphi = \langle U \rangle \psi$. By hypothesis $(\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}_1) \models \langle U \rangle \psi$ and $(\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}_2) \not\models \langle U \rangle \psi$. Then there is $\mathbf{n}'_1 \in \mathcal{N}$ such that $(\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}'_1) \models \psi$. Moreover, by definition the modal rank of $\langle U \rangle \psi$ is at least 1 and therefore the spoiler can play a $\langle U \rangle$ move. Suppose that the spoiler selects the structure $(\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}_1)$ and chooses exactly \mathbf{n}'_1 . According to the game, the duplicator must choose a $\mathbf{n}'_2 \in \mathcal{N}$. Since $(\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}_2) \not\models \langle U \rangle \psi$ it holds that $(\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}'_2) \not\models \psi$. By induction hypothesis, the spoiler has a winning strategy for $((\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}'_1), (\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}'_2), (m-1, s, k))$. Hence, by choosing \mathbf{n}'_1 the spoiler built a winning strategy for $((\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}_1), (\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}_2), (m, s, k))$.

induction step: $\varphi = \blacklozenge \psi$. By hypothesis $(\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}_1) \models \blacklozenge \psi$ and $(\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}_2) \not\models \blacklozenge \psi$. There is a finite forest $\mathcal{F}'_1 \subseteq \mathcal{F}_1$ such that $\text{card}(\mathcal{F}'_1) = \text{card}(\mathcal{F}_1) - 1$ and $(\mathcal{F}'_1, \mathbf{t}_1, \mathbf{n}_1) \models \psi$. Hence, $\text{dom}(\mathcal{F}_1) \neq \emptyset$ and moreover by definition the sabotage rank of $\blacklozenge \psi$ is at least 1. Therefore, the spoiler can play a \blacklozenge move. Suppose that the spoiler selects the structure $(\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}_1)$ and chooses exactly \mathcal{F}'_1 . According to the game, the duplicator must choose a finite forest $\mathcal{F}'_2 \subseteq \mathcal{F}_2$ such that $\text{card}(\mathcal{F}'_2) = \text{card}(\mathcal{F}_2) - 1$. Since $(\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}_2) \not\models \blacklozenge \psi$, it holds that $(\mathcal{F}'_2, \mathbf{t}_2, \mathbf{n}_2) \not\models \psi$. By induction hypothesis, the spoiler has a winning strategy for the game $((\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}'_1), (\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}'_2), (m, s-1, k))$. Hence, by choosing \mathcal{F}'_1 the spoiler built a winning strategy for $((\mathcal{F}_1, \mathbf{t}_1, \mathbf{n}_1), (\mathcal{F}_2, \mathbf{t}_2, \mathbf{n}_2), (m, s, k))$.

induction step: $\varphi = \diamond^* \psi$. By hypothesis $(\mathcal{F}_1, t_1, n_1) \models \diamond^* \psi$ and $(\mathcal{F}_2, t_2, n_2) \not\models \diamond^* \psi$. There is a finite forest $\mathcal{F}'_1 \subseteq \mathcal{F}_1$ such that $(\mathcal{F}'_1, t_1, n_1) \models \psi$. Moreover, by definition the repeated sabotage rank of $\diamond^* \psi$ is at least 1. Therefore, the spoiler can play a \diamond^* move. Suppose that the spoiler then select the structure $(\mathcal{F}_1, t_1, n_1)$ and chooses exactly \mathcal{F}'_1 . According to the game, the duplicator must choose a finite forest $\mathcal{F}'_2 \subseteq \mathcal{F}_2$. Since $(\mathcal{F}_2, t_2, n_2) \not\models \diamond^* \psi$, it holds that $(\mathcal{F}'_2, t_2, n_2) \not\models \psi$. By induction hypothesis, the spoiler has a winning strategy for the game $((\mathcal{F}_1, t_1, n'_1), (\mathcal{F}_2, t_2, n'_2), (m, s, k - 1))$. Hence, by choosing \mathcal{F}'_1 the spoiler built a winning strategy for $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), (m, s, k))$. \square

Proof of Theorem 4.15(II). We follow again the schema of the proof in [36]. We consider the contrapositive statement and thus prove that if $(\mathcal{F}_1, t_1, n_1) \sim_{rk} (\mathcal{F}_2, t_2, n_2)$ does not hold, then there is φ in ALT_{rk} s.t. $((\mathcal{F}_1, t_1, n_1) \models \varphi \text{ iff } (\mathcal{F}_2, t_2, n_2) \models \varphi)$ does not hold. Since the games are determined (Lemma 4.14) and ALT is closed under negation, we can alternatively show that

If $(\mathcal{F}_1, t_1, n_1) \not\sim_{rk} (\mathcal{F}_2, t_2, n_2)$ then there is φ in ALT_{rk} s.t. $(\mathcal{F}_1, t_1, n_1) \models \varphi$ and $(\mathcal{F}_2, t_2, n_2) \not\models \varphi$.

As already stated, the result is shown by induction on the rank rk , with respect to the order $<_{rk}$, and by cases on the first move that the spoiler makes in his winning strategy for the game $((\mathcal{F}_1, t_1, n_1), (\mathcal{F}_2, t_2, n_2), rk)$. Below, we reserve the symbol φ for the formula that distinguishes the two models, as in the statement above (i.e. $(\mathcal{F}_1, t_1, n_1) \models \varphi$ and $(\mathcal{F}_2, t_2, n_2) \not\models \varphi$).

base case: $rk = (0, 0, 0)$. Since spoiler has a winning strategy, in particular it wins the game of rank $(0, 0, 0)$. By definition of the game, spoiler does not play any move, and from the 1st and 2nd line of Figure 4.4 one of the following must hold:

- $(\mathcal{F}_1, t_1, n_1) \models \text{Hit}$ and $(\mathcal{F}_2, t_2, n_2) \not\models \text{Hit}$. Hence, $\varphi = \text{Hit}$.
- $(\mathcal{F}_1, t_1, n_1) \not\models \text{Hit}$ and $(\mathcal{F}_2, t_2, n_2) \models \text{Hit}$. Hence, $\varphi = \neg\text{Hit}$.
- $(\mathcal{F}_1, t_1, n_1) \models \text{Miss}$ and $(\mathcal{F}_2, t_2, n_2) \not\models \text{Miss}$. Hence, $\varphi = \text{Miss}$.
- $(\mathcal{F}_1, t_1, n_1) \not\models \text{Miss}$ and $(\mathcal{F}_2, t_2, n_2) \models \text{Miss}$. Hence, $\varphi = \neg\text{Miss}$.

This case also holds for games on arbitrary rank (m, s, k) where the spoiler wins simply from the conditions of the game that are imposed at the beginning of each round (1st and 2nd line of Figure 4.4), before playing any move.

In the induction steps, let us assume $rk = (m, s, k)$.

induction step: the spoiler plays a $\langle U \rangle$ move. Suppose that, by following its strategy, the spoiler chooses $(\mathcal{F}_1, t_1, n_1)$ and plays a $\langle U \rangle$ move. Notice that this implies $m \geq 1$. Let $n'_1 \in \mathcal{N}$ be the node selected by the spoiler. By Lemma 4.17(I), $(\mathcal{F}_1, t_1, n'_1) \models \Gamma_{(m-1, s, k)}(\mathcal{F}_1, t_1, n'_1)$. Let φ be defined as the formula $\langle U \rangle \Gamma_{(m-1, s, k)}(\mathcal{F}_1, t_1, n'_1)$. By definition, $\varphi \in \text{ALT}_{rk}$ and this formula is satisfied by $(\mathcal{F}_1, t_1, n_1)$. *Ad absurdum*, suppose that $(\mathcal{F}_2, t_2, n_2) \models \varphi$. Thus, there is n'_2 such that $(\mathcal{F}_2, t_2, n'_2) \models \Gamma_{(m-1, s, k)}(\mathcal{F}_1, t_1, n'_1)$. By Lemma 4.17(II) together with the definition of characteristic formula, there is no formula in $\text{ALT}_{(m-1, s, k)}$ that can discriminate between $(\mathcal{F}_1, t_1, n'_1)$ and $(\mathcal{F}_2, t_2, n'_2)$. As our games are determined (Lemma 4.14), by induction hypothesis the duplicator has a winning strategy for the game $((\mathcal{F}_1, t_1, n'_1), (\mathcal{F}_2, t_2, n'_2), (m - 1, s, k))$. However, this is contradictory as by hypothesis the spoiler has a winning strategy and the move it played is part of this strategy. Therefore, $(\mathcal{F}_1, t_1, n_1) \models \varphi$ and $(\mathcal{F}_2, t_2, n_2) \not\models \varphi$.

The proof is analogous for the case where the spoiler chooses $(\mathcal{F}_2, t_2, n_2)$ and a $n'_2 \in \mathcal{N}$. In this case we obtain $(\mathcal{F}_1, t_1, n_1) \not\models \psi$ and $(\mathcal{F}_2, t_2, n_2) \models \psi$ where $\psi = \langle U \rangle \Gamma_{(m-1, s, k)}(\mathcal{F}_2, t_2, n'_2)$. Thus, defining the formula φ as $\neg\psi$ proves the result.

induction step: the spoiler plays a ♦ move. This case is similar to the previous one. Suppose that, by following its strategy, the spoiler chooses $(\mathcal{F}_1, t_1, n_1)$ and plays a ♦ move. Notice that this implies $s \geq 1$ and $\text{dom}(\mathcal{F}_1) \neq \emptyset$. Let \mathcal{F}'_1 be the finite forest chosen by the spoiler. We have $\mathcal{F}'_1 \subseteq \mathcal{F}_1$ and $\text{card}(\mathcal{F}'_1) = \text{card}(\mathcal{F}_1) - 1$. By Lemma 4.17(I), $(\mathcal{F}'_1, t_1, n_1) \models \Gamma_{(m,s-1,k)}(\mathcal{F}'_1, t_1, n_1)$. Let φ be defined as the formula $\blacklozenge \Gamma_{(m,s-1,k)}(\mathcal{F}'_1, t_1, n_1)$. By definition, $\varphi \in \text{ALT}_{rk}$ and this formula is satisfied by $(\mathcal{F}_1, t_1, n_1)$. *Ad absurdum*, suppose that $(\mathcal{F}_2, t_2, n_2) \models \varphi$. There is \mathcal{F}'_2 such that $\mathcal{F}'_2 \subseteq \mathcal{F}_2$, $\text{card}(\mathcal{F}'_2) = \text{dom}(\mathcal{F}_2) - 1$ and $(\mathcal{F}'_2, t_2, n_2) \models \Gamma_{(m,s-1,k)}(\mathcal{F}'_1, t_1, n_1)$. By Lemma 4.17(II) together with the definition of characteristic formula, there is no formula in $\text{ALT}_{(m,s-1,k)}$ that can discriminate between $(\mathcal{F}'_1, t_1, n_1)$ and $(\mathcal{F}'_2, t_2, n_2)$. As our games are determined, by induction hypothesis this implies that the duplicator has a winning strategy for the game $((\mathcal{F}'_1, t_1, n_1), (\mathcal{F}'_2, t_2, n_2), (m, s-1, k))$. However, this is contradictory as by hypothesis the spoiler has a winning strategy and the move it played is part of this strategy. Therefore, $(\mathcal{F}_1, t_1, n_1) \models \varphi$ and $(\mathcal{F}_2, t_2, n_2) \not\models \varphi$.

Again, the proof is analogous for the case where the spoiler chooses $(\mathcal{F}_2, t_2, n_2)$ and a finite tree $\mathcal{F}'_2 \subseteq \mathcal{F}_2$ such that $\text{card}(\mathcal{F}'_2) = \text{card}(\mathcal{F}_2) - 1$. In this case we obtain $(\mathcal{F}_1, t_1, n_1) \not\models \psi$ and $(\mathcal{F}_2, t_2, n_2) \models \psi$ where $\psi = \blacklozenge \Gamma_{(m,s-1,k)}(\mathcal{F}'_2, t_2, n_2)$. Hence, $\varphi \stackrel{\text{def}}{=} \neg \psi$ proves the result.

induction step: the spoiler plays a ♦* move. This case is similar to the last two cases. Suppose that, by following its strategy, the spoiler chooses $(\mathcal{F}_1, t_1, n_1)$ and plays a ♦* move. This implies $k \geq 1$. Let \mathcal{F}'_1 be the finite forest chosen by the spoiler. Thus, $\mathcal{F}'_1 \subseteq \mathcal{F}_1$. By Lemma 4.17(I), we have that $(\mathcal{F}'_1, t_1, n_1) \models \Gamma_{(m,s,k-1)}(\mathcal{F}'_1, t_1, n_1)$. Let φ be the formula defined as $\blacklozenge^* \Gamma_{(m,s,k-1)}(\mathcal{F}'_1, t_1, n_1)$. By definition, $\varphi \in \text{ALT}_{rk}$ and this formula is satisfied by $(\mathcal{F}_1, t_1, n_1)$. *Ad absurdum*, suppose that $(\mathcal{F}_2, t_2, n_2) \models \varphi$. Then there is \mathcal{F}'_2 such that $\mathcal{F}'_2 \subseteq \mathcal{F}_2$ and $(\mathcal{F}'_2, t_2, n_2) \models \Gamma_{(m,s,k-1)}(\mathcal{F}'_1, t_1, n_1)$. By Lemma 4.17(II) together with the definition of characteristic formula, there is no formula in $\text{ALT}_{(m,s,k-1)}$ that can discriminate between $(\mathcal{F}'_1, t_1, n_1)$ and $(\mathcal{F}'_2, t_2, n_2)$. As our games are determined, by induction hypothesis this implies that the duplicator has a winning strategy for the game $((\mathcal{F}'_1, t_1, n_1), (\mathcal{F}'_2, t_2, n_2), (m, s, k-1))$. However, this is contradictory as we assumed that the spoiler has a winning strategy and the move it played is part of this strategy. Therefore, $(\mathcal{F}_1, t_1, n_1) \models \varphi$ and $(\mathcal{F}_2, t_2, n_2) \not\models \varphi$.

The proof is analogous for the case where the spoiler chooses $(\mathcal{F}_2, t_2, n_2)$ and a forest $\mathcal{F}'_2 \subseteq \mathcal{F}_2$. In this case we have $(\mathcal{F}_1, t_1, n_1) \not\models \psi$ and $(\mathcal{F}_2, t_2, n_2) \models \psi$, where $\psi = \blacklozenge^* \Gamma_{(m,s,k-1)}(\mathcal{F}'_2, t_2, n_2)$. Hence, $\varphi \stackrel{\text{def}}{=} \neg \psi$ proves the result. \square

Using the EF-games for ALT. We start using the EF-games for ALT to derive three easy inexpressibility results. Notably, these results are later helpful as they reduce the set of pointed forests needed in order to conclude that a formula φ of ALT is satisfiable.

Lemma 4.18. Let φ be a formula.

- (I) φ is satisfiable iff it is satisfiable by a pointed forest (\mathcal{F}, t, n) where $t \notin \text{dom}(\mathcal{F})$.
- (II) Given a forest \mathcal{F} and nodes $t \in \mathcal{N}$ and $n, n' \notin \text{dom}(\mathcal{F})$, $(\mathcal{F}, t, n) \models \varphi$ iff $(\mathcal{F}, t, n') \models \varphi$.
- (III) If $(\mathcal{F}_1, t_1, n_1) \sim_{rk} (\mathcal{F}_2, t_2, n_2)$ then the duplicator has a winning strategy where it always replies to $\langle U \rangle$ moves by selecting nodes in $\text{dom}(\mathcal{F}_i) \cup \text{ran}(\mathcal{F}_i)$, for some $i \in \{1, 2\}$.

As these results are quite straightforward, we just sketch their proof so that we can focus on how the EF-games are used without getting lost in technical details. We will see a full proof that uses the games later, with Lemma 4.19.

Proof (sketch). Consider the left-to-right direction of (I) (the other direction is obvious), and so let (\mathcal{F}, t, n) be a pointed forest such that $(\mathcal{F}, t, n) \models \varphi$. We modify (\mathcal{F}, t, n) so that the target node is not in the domain of the forest. In particular, we consider a node $t' \notin \text{dom}(\mathcal{F}) \cup \text{ran}(\mathcal{F})$ and define the forest $\mathcal{F}'(n') \stackrel{\text{def}}{=} \text{if } \mathcal{F}(n') = t \text{ then } t' \text{ else } \mathcal{F}(n')$. Notice that $t' \notin \text{dom}(\mathcal{F}')$. We show that for every $\text{rk} \in \mathbb{N}^3$, $(\mathcal{F}, t, n) \sim_{\text{rk}} (\mathcal{F}', t', n)$ with an easy induction on rk , leading to (I) directly by Theorem 4.15. The proof of (II) is even simpler, as we just need to prove that for all $\text{rk} \in \mathbb{N}^3$, $(\mathcal{F}, t, n) \sim_{\text{rk}} (\mathcal{F}, t, n')$, again by induction on the rank. (III) is a consequence of (II). \square

Interestingly enough, Lemma 4.18(III) fundamentally implies that changing the definition of the set of nodes \mathcal{N} to be finite, instead of infinite as we do throughout this work, does not change the expressive power nor the complexity of ALT.

The proof of Lemma 4.18(I) shows us how the games can be used in order to conclude an inexpressibility result. In general, we consider a property that we want to show to be not expressible in the logic, as for example the fact that the target node is in the domain of the forest (as in Lemma 4.18(I)). Then, for every rank $\text{rk} \in \mathbb{N}^3$, we construct two pointed forests $(\mathcal{F}_1, t_1, n_1)$ and $(\mathcal{F}_2, t_2, n_2)$ such that only one of the two has the wanted property. We show that $(\mathcal{F}_1, t_1, n_1) \sim_{\text{rk}} (\mathcal{F}_2, t_2, n_2)$, which allows us to conclude that the property cannot be expressed, by Theorem 4.15. When this property is very simple, as it is the case for Lemma 4.18(I), it is possible to construct a single pair of finite forests $(\mathcal{F}_1, t_1, n_1)$ and $(\mathcal{F}_2, t_2, n_2)$ so that for every $\text{rk} \in \mathbb{N}^3$ we can prove $(\mathcal{F}_1, t_1, n_1) \sim_{\text{rk}} (\mathcal{F}_2, t_2, n_2)$.

Let (\mathcal{F}, t, n) be a pointed forest. We now show that ALT has a very limited expressive power with respect to the miss nodes. In particular, it can only check whether the current evaluation node n is a member of $\mathcal{F}[\text{Miss}]$ (with the formula **Miss**), and for the size of $\mathcal{F}[\text{Miss}]$ (with the formula **size(Miss) $\geq \beta$**). We formalise this inexpressibility result with the following lemma.

Lemma 4.19. Let $\text{rk} = (m, s, k)$. Let $\mathcal{F}_1, \mathcal{F}_2$ be two forests, and $n_1, n_2, t \in \mathcal{N}$. Suppose that

1. (\mathcal{F}_1, t, n_1) and (\mathcal{F}_2, t, n_2) agree on the set of descendants of t , i.e. for every \mathcal{F}_1 -descendant or \mathcal{F}_2 -descendant n of t , $\mathcal{F}_1(n) = \mathcal{F}_2(n)$, and if n_1 or n_2 are descendants of t , then $n_1 = n_2$,
2. $n_1 \in \mathcal{F}_1[\text{Miss}]_t$ if and only if $n_2 \in \mathcal{F}_2[\text{Miss}]_t$,
3. $\min(\text{card}(\mathcal{F}_1[\text{Miss}]_t), m + s + k) = \min(\text{card}(\mathcal{F}_2[\text{Miss}]_t), m + s + k)$.

Then $(\mathcal{F}_1, t, n_1) \sim_{\text{rk}} (\mathcal{F}_2, t, n_2)$.

Before proving this result, let us informally explain how it shows that ALT can only express the two aforementioned properties of $\mathcal{F}[\text{Miss}]$. For example, let us suppose (ad absurdum) that there is a formula φ that characterises the set of pointed forests having a miss node with at least two children. Let us consider a rank $\text{rk} = (m, s, k)$ and a pointed forest (\mathcal{F}_1, t, n) that satisfies the formula φ . We consider the subforest $\mathcal{F} \subseteq \mathcal{F}_1$ whose domain corresponds to the set of \mathcal{F}_1 -descendants of t . We extend \mathcal{F} to a forest \mathcal{F}_2 by (re)defining it on the nodes in $\mathcal{F}_1[\text{Miss}]_t$ so that $\mathcal{F}_2[\text{Miss}]_t = \mathcal{F}_1[\text{Miss}]_t$ and none of these nodes has more than one \mathcal{F}_2 -child (this construction can always be done). Notice that \mathcal{F}_2 is defined in a way that (\mathcal{F}_1, t, n) and (\mathcal{F}_2, t, n) satisfy the three properties (1), (2) and (3). We apply Lemma 4.19 to conclude $(\mathcal{F}_1, t, n) \sim_{\text{rk}} (\mathcal{F}_2, t, n)$, which in turn shows that $(\mathcal{F}_2, t, n) \models \varphi$ by Theorem 4.15. However, (\mathcal{F}_2, t, n) is defined so that every node in $\mathcal{F}_2[\text{Miss}]_t$ has at most one child. Thus, φ cannot characterise the set of models having a miss node with at least two children.

As we discuss in the next section, the inexpressibility result shown in Lemma 4.19 plays a central role in the development of the reduction that leads to the TOWER-hardness of the satisfiability problem for ALT. In particular, most of the difficulties of this reduction stem from

the fact that we need to get around the limited expressiveness that ALT has with respect to miss nodes. We conclude this section on the expressive power of ALT with the proof of Lemma 4.19. Readers that are eager to see the TOWER-hardness of this logic can skip to page 97.

Proof of Lemma 4.19. The proof is by induction on the rank rk , with respect to the strict order $<_{\text{rk}}$ and by cases on the move made by the spoiler in the game. As the statement is symmetrical with respect to the two pointed forests (\mathcal{F}_1, t, n_1) and (\mathcal{F}_2, t, n_2) , we assume w.l.o.g. that the spoiler chooses and plays on the structure (\mathcal{F}_1, t, n_1) and hence define below the strategy of duplicator on (\mathcal{F}_2, t, n_2) . The strategy of the duplicator for the cases where it must reply on (\mathcal{F}_1, t, n_1) can be described from the one below by simply swapping the two structures. Below, the indices (1), (2) and (3) refer to the homonymous properties in the statement of the lemma. We refer to them as *hypothesis* whenever we consider (\mathcal{F}_1, t, n_1) and (\mathcal{F}_2, t, n_2) . Instead, we call them *properties* when we are proving them for subforests of (\mathcal{F}_1, t, n_1) and (\mathcal{F}_2, t, n_2) .

base case: $\text{rk} = (0, 0, 0)$. The hypothesis (1) and (2) imply that for every $\pi \in \{\text{Miss}, \text{Hit}\}$, the double implication $((\mathcal{F}_1, t, n_1) \models \pi \text{ iff } (\mathcal{F}_2, t, n_2) \models \pi)$ holds. Since the spoiler cannot play any move, the duplicator wins the game.

For the induction step, we assume $\text{rk} = (m, s, k) \neq (0, 0, 0)$ and that the lemma holds for every $\text{rk}' <_{\text{rk}} \text{rk}$. We divide the proof following the move of the spoiler.

induction step: the spoiler plays a $\langle U \rangle$ move. This implies $m \geq 1$. Let $n'_1 \in \mathcal{N}$ be the node chosen by the spoiler. Let us consider the following procedure for the duplicator:

```

if  $n'_1$  is a hit node of  $(\mathcal{F}_1, t, n_1)$  then the duplicator selects  $n'_1$ 
else if  $n'_1 \in \mathcal{F}_1[\text{Miss}]_t$  then the duplicator selects a node  $n'_2 \in \mathcal{F}_2[\text{Miss}]_t$ 
else the duplicator selects a node  $n'_2 \notin \text{dom}(\mathcal{F}_2)$ .
```

Notice the this procedure is well-defined. In particular, if $n'_1 \in \mathcal{F}_1[\text{Miss}]_t$ then from $m \geq 1$ and the hypothesis (3), we conclude that $\text{card}(\mathcal{F}_2[\text{Miss}]_t) \geq 1$, so that the the duplicator can effectively select a node $n'_2 \in \mathcal{F}_2[\text{Miss}]_t$. Moreover, the hypothesis (1) insures that if n'_1 is a hit node of (\mathcal{F}_1, t, n_1) then it is also a hit node of (\mathcal{F}_2, t, n_2) . Lastly, if the duplicator select a node $n'_2 \notin \text{dom}(\mathcal{F}_2)$, it means that n'_1 is not a hit or miss node, hence $n'_1 \notin \text{dom}(\mathcal{F}_1)$. The EF-game continues on the state $((\mathcal{F}_1, t, n'_1), (\mathcal{F}_2, t, n'_2), (m-1, s, k))$. By definition of n'_2 , we can check that (\mathcal{F}_1, t, n'_1) and (\mathcal{F}_2, t, n'_2) satisfy the three properties (1), (2) and (3), w.r.t. the rank $(m-1, s, k)$. By induction hypothesis, we conclude $(\mathcal{F}_1, t, n'_1) \sim_{(m-1, s, k)} (\mathcal{F}_2, t, n'_2)$. This implies that, by relying on the procedure above, the duplicator can build a winning strategy for the game $((\mathcal{F}_1, t, n_1), (\mathcal{F}_2, t, n_2), \text{rk})$.

induction step: the spoiler plays a \blacklozenge move. This implies $s \geq 1$. let $\mathcal{F}'_1 \subseteq \mathcal{F}_1$ be the sub-forest chosen by the spoiler. We have $\text{card}(\mathcal{F}'_1) = \text{card}(\mathcal{F}_1) - 1$. Let \bar{n} be the only node in $\text{dom}(\mathcal{F}_1) \setminus \text{dom}(\mathcal{F}'_1)$. Let us consider the following procedure for the duplicator:

```

if  $\bar{n}$  is a hit node of  $(\mathcal{F}_1, t, n_1)$  then the duplicator selects the forest  $\mathcal{F}_2 \setminus \{(\bar{n}, \mathcal{F}_2(\bar{n}))\}$ 
else the duplicator selects a forest  $\mathcal{F}_2 \setminus \{(n', \mathcal{F}_2(n'))\}$ 
    where  $n' \in \mathcal{F}_2[\text{Miss}]_t$ , and  $n' = n_1 \Leftrightarrow \bar{n} = n_2$ .
```

Notice that this procedure is well-defined. In particular, if \bar{n} is a hit node of (\mathcal{F}_1, t, n_1) , from the hypothesis (1) \bar{n} is a hit node of (\mathcal{F}_2, t, n_2) , and so $(\bar{n}, \mathcal{F}_2(\bar{n}))$ is defined. Moreover, if \bar{n} is not a hit node, then from $\bar{n} \in \text{dom}(\mathcal{F}_1)$ we conclude that it is a miss node. By $s \geq 1$ and thanks to the hypothesis (3), $\text{card}(\mathcal{F}_2[\text{Miss}]_t) \geq 1$. With the hypothesis (2), this implies that the duplicator can effectively select the forest in the else branch of the procedure. Let \mathcal{F}'_2 be the forest selected by the duplicator, using the procedure above. We show that

(\mathcal{F}'_1, t, n_1) and (\mathcal{F}'_2, t, n_2) satisfy the properties (1), (2) and (3) w.r.t. the rank $(m, s - 1, k)$. We divide the proof into two cases, depending on whether or not $\bar{n} \in \mathcal{F}_1[\text{Miss}]_t$.

case: $\bar{n} \in \mathcal{F}_1[\text{Miss}]_t$. Let n' be some node such that $\{n'\} = \text{dom}(\mathcal{F}_2) \setminus \text{dom}(\mathcal{F}'_2)$. From the definition of the procedure above, $n' \in \mathcal{F}_2[\text{Miss}]_t$, and $n' = n_1$ if and only if $\bar{n} = n_2$. This implies the satisfaction of the property (2). Moreover, the property (1) is also satisfied. Indeed, since \bar{n} is a miss node, every \mathcal{F}'_1 -descendant of t is also a \mathcal{F}_1 -descendant of t , and vice versa. Similarly, as n' is a miss node, every \mathcal{F}'_2 -descendant of t is also a \mathcal{F}_2 -descendant of t , and vice versa. Thus, the property (1) is implied by the hypothesis (1). In order to conclude this case, we prove the satisfaction of property (3). First, since \bar{n} is a miss node, $\mathcal{F}'_1[\text{Miss}]_t \cup \{\bar{n}\} = \mathcal{F}_1[\text{Miss}]_t$. Similarly, $\mathcal{F}'_2[\text{Miss}]_t \cup \{n'\} = \mathcal{F}_2[\text{Miss}]_t$. As $\bar{n} \notin \text{dom}(\mathcal{F}'_1)$ and $n' \notin \text{dom}(\mathcal{F}'_2)$, we conclude that $\text{card}(\mathcal{F}'_1[\text{Miss}]_t) + 1 = \text{card}(\mathcal{F}_1[\text{Miss}]_t)$ and $\text{card}(\mathcal{F}'_2[\text{Miss}]_t) + 1 = \text{card}(\mathcal{F}_2[\text{Miss}]_t)$.

Thanks to the hypothesis (3), i.e.

$$\min(\text{card}(\mathcal{F}_1[\text{Miss}]_t), m + s + k) = \min(\text{card}(\mathcal{F}_2[\text{Miss}]_t), m + s + k),$$

we show property (3) with the following equivalences:

$$\begin{aligned} \min(\text{card}(\mathcal{F}'_1[\text{Miss}]_t), m + (s - 1) + k) &= \min(\text{card}(\mathcal{F}_1[\text{Miss}]_t) + 1, m + s + k) - 1 \\ &= \min(\text{card}(\mathcal{F}_1[\text{Miss}]_t), m + s + k) - 1 \\ &= \min(\text{card}(\mathcal{F}_2[\text{Miss}]_t), m + s + k) - 1 \\ &= \min(\text{card}(\mathcal{F}'_2[\text{Miss}]_t) + 1, m + s + k) - 1 \\ &= \min(\text{card}(\mathcal{F}'_2[\text{Miss}]_t), m + (s - 1) + k). \end{aligned}$$

Here, we use the equivalence $\min(x+1, y+1) = \min(x, y)+1$ (x, y arbitrary numbers).

case: $\bar{n} \notin \mathcal{F}_1[\text{Miss}]_t$. This implies that \bar{n} is a hit node of (\mathcal{F}_1, t, n_1) , and from the procedure followed by the duplicator, $\mathcal{F}'_2 = \mathcal{F}_2 \setminus \{(\bar{n}, \mathcal{F}_2(\bar{n}))\}$. First of, since \bar{n} is a hit node and $\text{dom}(\mathcal{F}_1) \setminus \text{dom}(\mathcal{F}'_1) = \{\bar{n}\}$, we can show that

$$\mathcal{F}'_1[\text{Miss}]_t = \mathcal{F}_1[\text{Miss}]_t \cup \{n' \in \mathcal{N} \mid n' \text{ is a } \mathcal{F}_1\text{-descendant of } \bar{n}\}.$$

Indeed, when $(\bar{n}, \mathcal{F}_2(\bar{n}))$ is removed from \mathcal{F}_1 , all its descendants become miss nodes, whereas every other hit node of \mathcal{F}_1 (\bar{n} excluded) is still a hit node of \mathcal{F}'_1 . The same holds true for \mathcal{F}_2 , so that the following equality holds:

$$\mathcal{F}'_2[\text{Miss}]_t = \mathcal{F}_2[\text{Miss}]_t \cup \{n' \in \mathcal{N} \mid n' \text{ is a } \mathcal{F}_2\text{-descendant of } \bar{n}\}.$$

By hypothesis (1), removing $(\bar{n}, \mathcal{F}_2(\bar{n}))$ from both \mathcal{F}_1 and \mathcal{F}_2 leads to two pointed forests that agree on the set of descendants of t . Thus, property (1) is satisfied. Moreover, again from the hypothesis (1), the set of \mathcal{F}_1 -descendants of \bar{n} is also the set of \mathcal{F}_2 -descendants of \bar{n} , i.e.

$$\{n' \in \mathcal{N} \mid n' \text{ is a } \mathcal{F}_1\text{-descendant of } \bar{n}\} = \{n' \in \mathcal{N} \mid n' \text{ is a } \mathcal{F}_2\text{-descendant of } \bar{n}\}.$$

This implies two things. First, from the characterisation of $\mathcal{F}'_1[\text{Miss}]_t$ and $\mathcal{F}'_2[\text{Miss}]_t$ (above), together with the hypothesis (2), (\mathcal{F}'_1, t, n_1) and (\mathcal{F}'_2, t, n_2) satisfy property (2). Second, thanks to the hypothesis (3), i.e.

$$\min(\text{card}(\mathcal{F}_1[\text{Miss}]_t), m + s + k) = \min(\text{card}(\mathcal{F}_2[\text{Miss}]_t), m + s + k),$$

we show property (3) with the following equivalences, where $\beta = m + (s - 1) + k$:

$$\begin{aligned} \min(\text{card}(\mathcal{F}'_1[\text{Miss}]_t), \beta) &= \min(\text{card}(\mathcal{F}_1[\text{Miss}]_t) + \text{card}((\mathcal{F}_1^{-1})^+(\bar{n})), \beta) \\ &= \min(\text{card}(\mathcal{F}_2[\text{Miss}]_t) + \text{card}((\mathcal{F}_2^{-1})^+(\bar{n})), \beta) \\ &= \min(\text{card}(\mathcal{F}'_2[\text{Miss}]_t), \beta) \end{aligned}$$

where, given $j \in \{1, 2\}$, $(\mathcal{F}_j^{-1})^+(\bar{n})$ is the set of \mathcal{F}_j -descendants of \bar{n} .

In both cases, since we have shown that (\mathcal{F}'_1, t, n_1) and (\mathcal{F}'_2, t, n_2) satisfy the properties (1), (2) and (3) w.r.t. the rank $(m, s - 1, k)$, we can apply the induction hypothesis and conclude that $(\mathcal{F}'_1, t, n_1) \sim_{(m, s - 1, k)} (\mathcal{F}'_2, t, n_2)$. This implies that, by relying on the procedure above, the duplicator can build a winning strategy for the game $((\mathcal{F}_1, t, n_1), (\mathcal{F}_2, t, n_2), rk)$.

induction case: the spoiler plays a ♦* move. This implies $k \geq 1$. Let $\mathcal{F}'_1 \subseteq \mathcal{F}_1$ be the forest chosen by the spoiler. Let us partition \mathcal{F}'_1 into the two subforests H and M_1 s.t.

$$H \stackrel{\text{def}}{=} \{(n, \mathcal{F}_1(n)) \in \mathcal{F}'_1 \mid n \text{ is a } \mathcal{F}_1\text{-descendant of } t\},$$

$$M_1 \stackrel{\text{def}}{=} \{(n, \mathcal{F}_1(n)) \in \mathcal{F}'_1 \mid n \in \mathcal{F}_1[\text{Miss}]_t\}.$$

By hypothesis (1), H is a subforest of \mathcal{F}_2 . Let us consider a subforest M_2 of \mathcal{F}_2 such that

- A. M_2 contains only miss nodes of \mathcal{F}_2 , i.e. $\text{dom}(M_2) \subseteq \mathcal{F}_2[\text{Miss}]_t$,
- B. $n_2 \in M_2$ if and only if $n_1 \in M_1$,
- C. $\text{card}(M_2) = \min(M_1, m + s + (k - 1))$.

From the hypothesis (2) and (3), the subforest M_2 can always be defined. Moreover, M_2 is disjoint from H . Let us show that the duplicator has a winning strategy in which it replies to \mathcal{F}'_1 with the subforest $\mathcal{F}'_2 \stackrel{\text{def}}{=} H \cup M_2$ of \mathcal{F}_2 . We show that (\mathcal{F}'_1, t, n_1) and (\mathcal{F}'_2, t, n_2) satisfy the properties (1), (2) and (3) w.r.t. the rank $(m, s, k - 1)$. Property (1) holds directly from the definition of H together with hypothesis (1). For the properties (2) and (3), we first notice that $\mathcal{F}'_1[\text{Miss}]_t = H[\text{Miss}]_t \cup \text{dom}(M_2)$ and that $\mathcal{F}'_2[\text{Miss}]_t = H[\text{Miss}]_t \cup \text{dom}(M_1)$. Then, property (2) stems from (B), whereas property (3) stems from (C). This allows us to apply the induction hypothesis to conclude that $(\mathcal{F}'_1, t, n_1) \sim_{(m, s, k - 1)} (\mathcal{F}'_2, t, n_2)$. Therefore, the duplicator can build a winning strategy for the game $((\mathcal{F}_1, t, n_1), (\mathcal{F}_2, t, n_2), rk)$. \square

4.3 THE COMPLEXITY OF ALT

We are now ready to show that the satisfiability problem for ALT is TOWER-complete. The hardness proof is by reduction from the satisfiability problem of Propositional interval temporal logic under locality principle [112, 82], whereas the upper bound holds directly from Proposition 4.3.

4.3.1 Propositional Interval Temporal Logic.

Propositional Interval Temporal Logic (PITL) is a logic that was introduced by B. Moszkowski in [112] for the verification of hardware components. It is interpreted on non-empty finite words over a finite alphabet of unary symbols Σ . Its formulae φ are from the grammar below ($a \in \Sigma$):

$$\begin{array}{lll} \pi := & \top & (\text{true}) & \varphi := & \pi & (\text{atomic formulae}) \\ | & 1 & (\text{single predicate}) & | & \varphi \wedge \varphi & (\text{Boolean connectives}) \\ | & a & (\text{head predicate}) & | & \varphi \mathbin{|} \varphi & (\text{composition operator}) \end{array}$$

The satisfaction relation \models for the formulae of PITL is defined in Figure 4.5, with respect to a non-empty word $a_1 \dots a_k \in \Sigma^+$. Standard cases for \top and Boolean connectives are omitted. The interpretation considered here is often called the *locality principle* interpretation of PITL. This name highlights the fact that the satisfaction of the predicate a only depends on the first symbol (i.e. the head) of the word. The main feature of this logic is its *composition operator* $|$. Intuitively, $\varphi \mathbin{|} \psi$ is satisfied by words that can be “chopped” into a prefix and a suffix, so that

$$\begin{aligned}
 \mathbf{a}_1 \dots \mathbf{a}_k \models 1 &\quad \text{iff } k = 1 \text{ (i.e. the word } \mathbf{a}_1 \dots \mathbf{a}_k \text{ is a symbol of } \Sigma\text{),} \\
 \mathbf{a}_1 \dots \mathbf{a}_k \models \mathbf{a} &\quad \text{iff } \mathbf{a}_1 = \mathbf{a} \text{ (i.e. the word is headed by the symbol } \mathbf{a}\text{),} \\
 \mathbf{a}_1 \dots \mathbf{a}_k \models \varphi \mid \psi &\quad \text{iff there is } j \in [1, k] \text{ such that } \mathbf{a}_1 \dots \mathbf{a}_j \models \varphi \text{ and } \mathbf{a}_j \dots \mathbf{a}_k \models \psi.
 \end{aligned}$$

Figure 4.5: Satisfaction relation for PITL, under locality principle.

the prefix satisfies φ and the suffix satisfies ψ . It is important to notice that these prefix and suffix overlap: the last symbol of the prefix is the first symbol of the suffix.

The satisfiability problem of PITL under locality principle is TOWER-complete. The fact that it is non-elementary decidable was proven by B. Moszkowski [112], by reduction from the non-emptiness problem of star-free regular languages previously studied by A. R. Meyer and L. J. Stockmeyer [111]. TOWER-completeness is then established from [128].

Proposition 4.20 (From [112, 128]). The satisfiability problem of PITL is TOWER-complete.

As we have already shown that finite words can be encoded in ALT (Section 4.2), a promising route to prove that the satisfiability problem of ALT is TOWER-hard is by reduction from the satisfiability problem of PITL. However, because of the limited expressive power that ALT has on miss nodes (see Lemma 4.19) we already know that the composition operator $|$ cannot be easily translated. Let us consider a pointed forest $(\mathcal{F}, \mathbf{n}, \mathbf{t})$ encoding a non-empty word $\mathbf{w} = \mathbf{a}_1 \dots \mathbf{a}_k$. Moreover, let us assume that the set of main nodes of this encoding is $\mathbb{M} = (\mathbf{n}_1, \dots, \mathbf{n}_k)$. Chopping \mathbf{w} into two pieces means splitting in some way the main path $\mathbf{n}_1, \dots, \mathbf{n}_k$ of $(\mathcal{F}, \mathbf{t}, \mathbf{n})$ to then check that the word encoded by $\mathbf{n}_1, \dots, \mathbf{n}_i$ satisfies a certain formula φ , whereas the one encoded by $\mathbf{n}_i, \dots, \mathbf{n}_k$ satisfies a formula ψ . Neglecting the fact that the two structures share the node \mathbf{n}_i , the main problem in doing this is that after the split the nodes $\mathbf{n}_1, \dots, \mathbf{n}_i$ stop being descendants of the target node, and so they become miss nodes. As a consequence of Lemma 4.19, we know that ALT cannot check in any way what is the word encoded by these nodes. Trivial translations from PITL to ALT seem therefore impossible.

4.3.2 PITL on marked words.

To solve the issue of capturing the composition operator of PITL in ALT, we consider an alternative interpretation of PITL where, instead of chopping a word, the operator $|$ marks the symbol where the cut should have taken place. As we will see, the alternative interpretation is equivalent to the one under locality principle given above, so that the TOWER-completeness result of Proposition 4.20 still hold. We start by introducing the notions of marking of a symbol, an alphabet and a word, as well as a notion of decomposition for marked words. For simplicity, throughout the section we fix a (non-empty) finite alphabet Σ .

Definition 4.21 (Markings). Let $\bar{\Sigma}$ be an alphabet disjoint from Σ , such that $\text{card}(\bar{\Sigma}) = \text{card}(\Sigma)$. A *marking* for Σ is a bijection $\bar{(\cdot)} : \Sigma \rightarrow \bar{\Sigma}$, relating a symbol $\mathbf{a} \in \Sigma$ to its *marked variant* $\bar{\mathbf{a}} \in \bar{\Sigma}$.

We fix Σ_\bullet to be the alphabet $\Sigma \cup \bar{\Sigma}$. A word of Σ_\bullet is *marked* if it has some symbol from $\bar{\Sigma}$.

Definition 4.22 (Marked word decomposition). Given a marked word $\mathbf{w} \in \Sigma_\bullet^+$, we write $\Delta(\mathbf{w})$ for the *decomposition* $(\mathbf{w}', \bar{\mathbf{a}}, \mathbf{w}'')$ where $\mathbf{w}' \in \Sigma^*$ is not marked, $\bar{\mathbf{a}} \in \bar{\Sigma}$ is marked, and $\mathbf{w} = \mathbf{w}' \bar{\mathbf{a}} \mathbf{w}''$.

-
- $\mathfrak{w} \models 1$ iff $\Delta(\mathfrak{w}) = (\mathfrak{w}', \bar{a}, \mathfrak{w}'')$ and $\mathfrak{w}' = \epsilon$ (i.e. \mathfrak{w} is headed by a marked symbol),
 $\mathfrak{w} \models a$ iff \mathfrak{w} is headed by the symbol a or the symbol \bar{a} ,
 $\mathfrak{w} \models \varphi \mid \psi$ iff $\Delta(\mathfrak{w}) = (\mathfrak{w}', \bar{a}, \mathfrak{w}'')$ and there is a symbol $b \in \Sigma$ such that
 (a) $\mathfrak{w}' = \epsilon, b = a$ and $\bar{a} \mathfrak{w}'' \models_{\bullet} \varphi \wedge \psi$
 or (b) $\mathfrak{w}' = b \mathfrak{w}_2$ and $\bar{b} \mathfrak{w}_2 \bar{a} \mathfrak{w}'' \models_{\bullet} \varphi$ and $b \mathfrak{w}_2 \bar{a} \mathfrak{w}'' \models_{\bullet} \psi$, for some $\mathfrak{w}_2 \in \Sigma^*$
 or (c) $\mathfrak{w}' \neq \epsilon$ and $b = a$ and $\mathfrak{w}' \bar{a} \mathfrak{w}'' \models_{\bullet} \varphi$ and $\bar{a} \mathfrak{w}'' \models_{\bullet} \psi$
 or (d) $\mathfrak{w}' = \mathfrak{w}_1 b \mathfrak{w}_2$ and $\mathfrak{w}_1 \bar{b} \mathfrak{w}_2 \bar{a} \mathfrak{w}'' \models_{\bullet} \varphi$ and $b \mathfrak{w}_2 \bar{a} \mathfrak{w}'' \models_{\bullet} \psi$,
 for some $\mathfrak{w}_1 \in \Sigma^+$ and $\mathfrak{w}_2 \in \Sigma^*$.

Figure 4.6: Satisfaction relation for PITL on marked words.

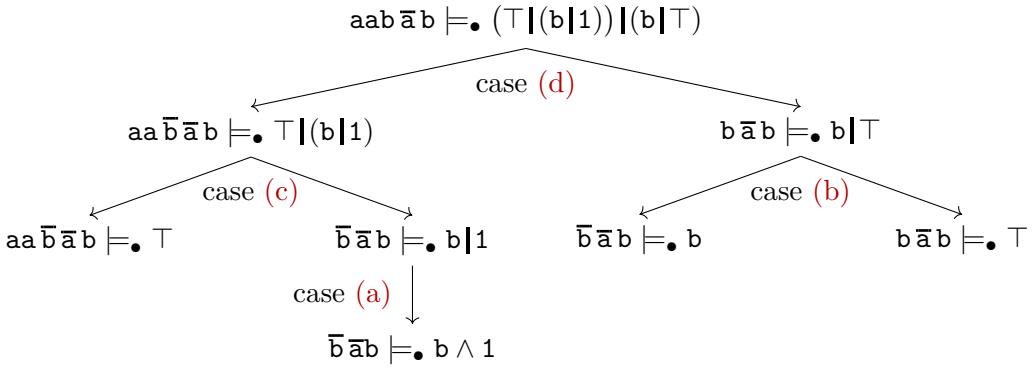


Figure 4.7: Example of the satisfaction of a formula on marked words.

Notice that the decomposition $\Delta(\mathfrak{w}) = (\mathfrak{w}', \bar{a}, \mathfrak{w}'')$ of a marked word \mathfrak{w} is uniquely defined, as the word $\mathfrak{w}' \bar{a}$ is the (only) prefix of \mathfrak{w} ending with its first marked symbol. As we will see, the notion of satisfiability we are about to define only depends on these prefixes.

We interpret PITL on marked words. Given a marked word $\mathfrak{w} \in \Sigma_{\bullet}^+$, the new satisfaction relation \models_{\bullet} for the formulae of PITL is given in Figure 4.6, again omitting standard cases for \top and Boolean connectives. The semantics of the predicates 1 and a is quite simple, and reflects the fact that the satisfaction of a formula depends on the only prefix of a marked word \mathfrak{w} that ends with its first marked symbol. For the predicate 1 to be satisfied, the word \mathfrak{w} must begin with a marked symbol, so in the decomposition $\Delta(\mathfrak{w}) = (\mathfrak{w}', \bar{a}, \mathfrak{w}'')$ the word \mathfrak{w}' is the *empty word* ϵ . For the predicate a , we simply check if \mathfrak{w} is headed by the symbol a or its marked variant \bar{a} . The definition of $\varphi \mid \psi$ is more involved. Let us consider the prefix $a_1 \dots a_{k-1} \bar{a}_k$ of \mathfrak{w} that ends with the first marked symbol. In order for $\mathfrak{w} \models_{\bullet} \varphi \mid \psi$ to hold, we must find a position $j \in [1, k]$ inside this prefix so that φ is satisfied by the word obtained from \mathfrak{w} by marking the j -th symbol (if it is not already marked), whereas ψ is satisfied by the suffix of \mathfrak{w} starting in j . In the formal definition given in Figure 4.6, this idea is split into four cases cases (a)–(d), depending on truthiness of $j = 1$ and $j = k$. For example, (a) correspond to the case where $j = k = 1$. This split is done as it better reflects the encoding of PITL in ALT.

Example 4.23. Consider the alphabets $\Sigma = \{a, b\}$ and $\bar{\Sigma} = \{\bar{a}, \bar{b}\}$. The schema in Figure 4.7 certifies that the marked word $aab\bar{a}b$ satisfies the formula $(T|(b|1))|(b|T)$. At each step we

highlight which of the four cases in the definition of $\varphi|\psi$ is used. For instance, let us pick the first step of the schema above, in which the case (d) in the definition of $\varphi|\psi$ is used. According to this case $aab\bar{a}b \models_{\bullet} (\top|(b|1))|(b|\top)$ holds as the word obtained by marking the third symbol, i.e. $a\bar{a}\bar{a}b$, satisfies $\top|(b|1)$, whereas the suffix of the word starting on this symbol, i.e. $b\bar{a}b$, satisfies $b|\top$. As we will show in a moment, marking symbols and considering suffixes of words are operations that can be simulated in ALT. Let us consider the decomposition $\Delta(aab\bar{a}b) = (aab, \bar{a}, b)$. One can check that the word $aaba$, obtained by concatenating the prefix aab of the decomposition with the non-marked symbol that corresponds to \bar{a} in the decomposition, satisfies the formula $(\top|(b|1))|(b|\top)$ in the standard semantics of PITL. Lemma 4.24 (below) shows that this is always the case.

The semantics on marked words is related to the standard semantics of PITL as follows.

Lemma 4.24. Let $w' \in \Sigma^*$, $a \in \Sigma$ and $w'' \in \Sigma_{\bullet}^*$. Let φ be a formula in PITL. We have,

$$w'a \models \varphi \text{ if and only if } w'\bar{a}w'' \models_{\bullet} \varphi.$$

Proof. The proof is by structural induction on φ (with the natural induction hypothesis stating that the lemma holds for strict subformulae of φ). Let us write w for $w'a$, and \bar{w} for the marked word $w'\bar{a}w''$. The base case with the atomic formulae 1 and $b \in \Sigma$ is by easy verification.

base case: $\varphi = 1$. The following equivalences show the result:

$$\begin{aligned} w \models 1 &\text{ if and only if } w' = \epsilon && (\text{by definition of } \models \text{ and } w = w'a) \\ &\text{if and only if } \bar{w} = \bar{a}w'' && (\text{from } \bar{w} = w'\bar{a}w'') \\ &\text{if and only if } \bar{w} \models_{\bullet} 1. && (\text{by definition of } \models_{\bullet}) \end{aligned}$$

base case: $\varphi = b$, where $b \in \Sigma$. The following double implication shows the result:

$$\begin{aligned} w \models b &\text{ if and only if } w'a \text{ is headed by } b && (\text{by definition of } \models \text{ and } w = w'a) \\ &\text{if and only if } \bar{w} \text{ is headed by } b \text{ or } \bar{b} && (\text{from } \bar{w} = w'\bar{a}w'') \\ &\text{if and only if } \bar{w} \models_{\bullet} b. && (\text{by definition of } \models_{\bullet}) \end{aligned}$$

The cases for Boolean connectives are obvious. We prove the result for the composition operator.

induction step: $\varphi = \varphi_1|\varphi_2$. Again, the result holds following a series of double implications:

$$\begin{aligned} w \models \varphi_1|\varphi_2 &\Leftrightarrow \text{there are } b \in \Sigma \text{ and } w_1, w_2 \in \Sigma^* \text{ s.t. } w = w_1bw_2, w_1b \models \varphi_1 \text{ and } bw_2 \models \varphi_2 \\ &\quad (\text{by definition of } \models) \\ &\Leftrightarrow \text{there are } b \in \Sigma \text{ and } w_1, w_2 \in \Sigma^* \text{ s.t. } w = w_1bw_2 \text{ and} \\ &\quad \begin{aligned} &(a) \quad w_1 = \epsilon, w_2 = \epsilon, b \models \varphi_1 \text{ and } b \models \varphi_2, && (\text{in this case, } b = a) \\ &\text{or } (b) \quad w_1 = \epsilon, w_2 \neq \epsilon, b \models \varphi_1 \text{ and } bw_2 \models \varphi_2, && (\text{in this case, } \exists w'_2 \ w_2 = w'_2a) \\ &\text{or } (c) \quad w_1 \neq \epsilon, w_2 = \epsilon, w_1b \models \varphi_1 \text{ and } b \models \varphi_2, && (\text{in this case, } b = a, w_1 = w) \\ &\text{or } (d) \quad w_1 \neq \epsilon, w_2 \neq \epsilon, w_1b \models \varphi_1 \text{ and } bw_2 \models \varphi_2. && (\text{in this case, } \exists w'_2 \ w_2 = w'_2a) \end{aligned} \\ &\quad (\text{by case distinction, on the truthiness of } w_1 = \epsilon \text{ and } w_2 = \epsilon) \\ &\Leftrightarrow \text{there are } b \in \Sigma \text{ and } w_1, w_2 \in \Sigma^* \text{ s.t. } w = w_1bw_2 \text{ and} \end{aligned}$$

- (a) $w_1 = \epsilon, w_2 = \epsilon, b = a, \bar{b}w'' \models_{\bullet} \varphi_1$ and $\bar{b}w'' \models_{\bullet} \varphi_2$,
- or (b) $w_1 = \epsilon, \exists w'_2 \in \Sigma^* \text{ s.t. } w_2 = w'_2 a, \bar{b}w_2 \bar{a}w'' \models_{\bullet} \varphi_1$ and $bw_2 \bar{a}w'' \models_{\bullet} \varphi_2$,
- or (c) $w_1 \neq \epsilon, w_2 = \epsilon, b = a, w' \bar{b}w'' \models_{\bullet} \varphi_1$ and $\bar{b}w'' \models_{\bullet} \varphi_2$,
- or (d) $w_1 \neq \epsilon, \exists w'_2 \in \Sigma^* \text{ s.t. } w_2 = w'_2 a, w_1 \bar{b}w'_2 \bar{a}w'' \models_{\bullet} \varphi_1$ and $bw'_2 \bar{a}w'' \models_{\bullet} \varphi_2$.

(by induction hypothesis, on all four cases)

\Leftrightarrow there is a symbol $b \in \Sigma$ such that

- (a) $w' = \epsilon, b = a$ and $\bar{a}w'' \models_{\bullet} \varphi \wedge \psi$
- or (b) $w' = bw_2$ and $\bar{b}w_2 \bar{a}w'' \models_{\bullet} \varphi$ and $bw_2 \bar{a}w'' \models_{\bullet} \psi$, for some $w_2 \in \Sigma^*$
- or (c) $w' \neq \epsilon$ and $b = a$ and $w' \bar{a}w'' \models_{\bullet} \varphi$ and $\bar{a}w'' \models_{\bullet} \psi$
- or (d) $w' = w_1 bw_2$ and $w_1 \bar{b}w_2 \bar{a}w'' \models_{\bullet} \varphi$ and $bw_2 \bar{a}w'' \models_{\bullet} \psi$,
for some $w_1 \in \Sigma^+$ and $w_2 \in \Sigma^*$,

(by easy manipulation of the formula)

$\Leftrightarrow \varphi \models_{\bullet} \varphi_1 \mid \varphi_2$. (by definition of \models_{\bullet})

□

4.3.3 Reducing PITL to ALT.

The alternative interpretation of PITL allows us to reduce the satisfiability problem of PITL to the satisfiability problem of ALT in a rather neat way. Once again, let us consider the alphabets $\Sigma, \bar{\Sigma}$ and $\Sigma_{\bullet} = \Sigma \cup \bar{\Sigma}$ of the previous section, and let us assume $\Sigma = [1, n]$ for some natural number $n \geq 1$. We consider the bijection $f : \Sigma_{\bullet} \rightarrow [1, 2n]$ defined as $f(a) \stackrel{\text{def}}{=} 2a$ for every symbol $a \in \Sigma$, and defined as $f(\bar{a}) \stackrel{\text{def}}{=} 2a - 1$ for every marked symbol $\bar{a} \in \bar{\Sigma}$. We write $f(a_1 \dots a_k)$ to denote the word $f(a_1) \dots f(a_k)$. Based on these definitions, f maps Σ_{\bullet} into the alphabet $[1, 2n]$, whose words can be encoded into trees (as in Section 4.2.1). In these trees, each symbol $a \in \Sigma$ corresponds to a main node having $2a + 1$ children that are character nodes (recall that we use $a + 1$ children to encode the symbol a). Similarly, each marked symbol $\bar{a} \in \bar{\Sigma}$ corresponds to a main node having $2a$ children that are character nodes. Let us consider a node n encoding a symbol in Σ . Because of the above distribution of non-marked and marked symbols, removing exactly one child of n that is a character node is equivalent to marking the symbol that n encodes. Let us now see how to capture this encoding in ALT.

Let us fix a pointed forest (\mathcal{F}, t, n) , which we suppose encodes a marked word $w \in \Sigma_{\bullet}$. We can check if the current node n encodes a marked symbol from $\bar{\Sigma}$ with the following formula:

$$\text{mark}_{\Sigma} \stackrel{\text{def}}{=} \bigvee_{a \in \Sigma} ((\#\text{child} = 2a \wedge \text{1st}_{[1,2n]}) \vee (\#\text{child} = 2a + 1 \wedge \neg \text{1st}_{[1,2n]}))$$

As already said, marked symbols correspond to nodes with $2a$ children that are character nodes, for some $a \in \Sigma$. In order to capture this notion, the formula $\bar{\Sigma}$ distinguishes the case where the current node n is the first node in the main path (whose only children are character nodes) from the case where n is not the first node in the main path (hence, it has one child in the main path).

As already stated, $w \models_{\bullet} \varphi$ examines the prefix of w that ends with the first marked symbol. To correctly reduce PITL to ALT we need to be able to find the part of (\mathcal{F}, t, n) that corresponds to this prefix. We can do so by noticing that this part is the only subtree whose root encodes a marked symbol and it is a \mathcal{F} -descendant of t as well as of every other node encoding marked symbols. This characterisation requires us to track the number of nodes encoding marked symbols in (\mathcal{F}, t, n) . To do so, first define a formula $\text{marks}_{\Sigma} \geq \beta$ stating that the forest has at least $\beta \in \mathbb{N}$ nodes encoding marked symbols. Luckily, we can rely on the formula $\text{size}(\varphi) \geq \beta$ defined in Section 4.2.1, and define $\text{marks}_{\Sigma} \geq \beta$ simply as $\text{size}(\text{mark}_{\Sigma}) \geq \beta$. Unfortunately, we cannot rely

$$\begin{aligned}
\tau_\beta(\top) &\stackrel{\text{def}}{=} \top, \\
\tau_\beta(1) &\stackrel{\text{def}}{=} \langle U \rangle (1st_{[1,2n]} \wedge \text{mark}_\Sigma), \\
\tau_\beta(a) &\stackrel{\text{def}}{=} \langle U \rangle 1st_{[2a-1,2a]}, \\
\tau_\beta(\neg\psi) &\stackrel{\text{def}}{=} \neg\tau_\beta(\psi), \\
\tau_\beta(\psi_1 \wedge \psi_2) &\stackrel{\text{def}}{=} \tau_\beta(\psi_1) \wedge \tau_\beta(\psi_2), \\
\tau_\beta(\psi_1 | \psi_2) &\stackrel{\text{def}}{=} \langle U \rangle (\text{symb} \wedge ((1st_{[1,2n]} \wedge \text{mark}_\Sigma \wedge \tau_\beta(\psi_1) \wedge \tau_\beta(\psi_2)) \\
&\quad \vee (1st_{[1,2n]} \wedge \neg \text{mark}_\Sigma \wedge \diamond(\text{mark}_\Sigma \wedge \tau_{\beta+1}(\psi_1)) \wedge \tau_\beta(\psi_2)) \\
&\quad \vee (\neg 1st_{[1,2n]} \wedge \text{mark}_\Sigma \wedge \#\text{markAnc}_\Sigma \geq \beta - 1 \wedge \tau_\beta(\psi_1) \wedge \diamond(1st_{[1,2n]} \wedge \tau_\beta(\psi_2))) \\
&\quad \vee (\neg 1st_{[1,2n]} \wedge \neg \text{mark}_\Sigma \wedge \#\text{markAnc}_\Sigma \geq \beta \wedge \diamond(\text{mark}_\Sigma \wedge \tau_{\beta+1}(\psi_1)) \\
&\quad \quad \wedge \diamond(1st_{[1,2n]} \wedge \tau_\beta(\psi_2))))).
\end{aligned}$$

Figure 4.8: Translation from PITL to ALT.

exactly on Lemma 4.7 to prove that this formula is correct, as the formula mark_Σ does not enjoy the property (2) required by this lemma. Similarly, we introduce the formula $\#\text{markAnc}_\Sigma \geq \beta$ which states that the current evaluation node encodes a symbol and has at least β ancestors that encode marked symbols. It is defined as follows:

$$\#\text{markAnc}_\Sigma \geq \beta \stackrel{\text{def}}{=} \text{symb} \wedge \diamond(\neg \text{inDom} \wedge \text{marks}_\Sigma \geq \beta).$$

The following lemma assures that the three formulae mark_Σ , $\text{marks}_\Sigma \geq \beta$ and $\#\text{markAnc}_\Sigma \geq \beta$ are correct, and highlights their semantics. The proof is given in Appendix B.

Lemma 4.25. Let $w \in \Sigma_\bullet^+$ and let (\mathcal{F}, t, n) be a pointed forest encoding the word $f(w) \in [1, 2n]^+$.

- (I) $(\mathcal{F}, t, n) \models \text{mark}_\Sigma$ iff n encodes a marked symbol of Σ_\bullet .
- (II) $(\mathcal{F}, t, n) \models \text{marks}_\Sigma \geq \beta$ iff \mathcal{F} contains at least β nodes encoding marked symbols of Σ_\bullet .
- (III) $(\mathcal{F}, t, n) \models \#\text{markAnc}_\Sigma \geq \beta$ iff n has at least β ancestors encoding marked symbols of Σ_\bullet .

At last, we are ready to translate formulae of PITL into formulae of ALT. Given a formula φ in PITL having symbols from $\Sigma = [1, n]$, we introduce its translation $\tau_\beta(\varphi)$ in ALT, where the index β is a positive natural number that we use to track the number of nodes encoding marked symbols. The translation is defined in Figure 4.8. It is homomorphic for \top and Boolean connectives. For the two predicates 1 and a , the translation faithfully represents the relation \models_\bullet . In the case of 1 , it requires the first node in the main path to correspond to a marked node. Instead, for the predicate a , it checks whether this node encodes the symbols $2a - 1$ or $2a$ which, by definition of f , correspond to $\bar{a} \in \bar{\Sigma}$ and $a \in \Sigma$, respectively. Lastly, the formula $\tau_\beta(\varphi | \psi)$ follows very closely the definition of the relation \models_\bullet : after the prefix " $\langle U \rangle (\text{symb} \wedge \dots)$ ", the formula split into four disjuncts, one for each of the cases in the definition of $\varphi | \psi$. For instance, let us consider a word w such that $\Delta(w) = (w', \bar{a}, w'')$. The second disjunct of $\tau_\beta(\varphi | \psi)$ encodes the case (b) in the definition of $w \models_\bullet \varphi | \psi$, as schematised below:

PITL	there is $b \in \Sigma \dots \exists w_2 \in \Sigma^* \text{ s.t. } w' = bw_2 \text{ and } \bar{b}w_2\bar{a}w'' \models \varphi \text{ and } bw_2\bar{a}w'' \models \psi$
ALT	$\langle U \rangle (\text{symb} \dots 1st_{[1,2n]} \wedge \neg \text{mark}_\Sigma \wedge \diamond(\text{mark}_\Sigma \wedge \tau_{\beta+1}(\varphi)) \wedge \tau_\beta(\psi))$

The lemma below ensures that the translation matches the semantics of the formula in PITL under the interpretation on marked words. Its proof, by structural induction on the formula φ of PITL, is quite long and thus given in Appendix B.

Lemma 4.26. Let $w \in \Sigma_\bullet^+$ with a marked word with $\beta \geq 1$ marked symbols. Let (\mathcal{F}, t, n) be an encoding of $f(w)$. For every φ in PITL, $w \models_\bullet \varphi$ if and only if $(\mathcal{F}, t, n) \models \tau_\beta(\varphi)$.

The reduction from the satisfiability problem of PITL on standard semantics follows as we are able to characterise the set of pointed forests encoding words in $\Sigma^* \bar{\Sigma}$ (first three conjuncts in the formula in the lemma below). To conclude, we simply apply Lemma 4.24 and Lemma 4.26.

Lemma 4.27. Every φ in PITL written with symbols from $\Sigma = [1, n]$ is satisfiable under the standard interpretation of PITL if and only if the following formula in ALT is satisfiable

$$\underbrace{\text{word}_{[1,2n]} \wedge \langle U \rangle \text{Hit} \wedge [U] (\text{mark}_\Sigma \Leftrightarrow \text{Hit} \wedge \neg \blacklozenge(\text{Miss}))}_{\text{The forest encodes a non-empty word. The only child of the target node is the only node encoding a marked symbol.}} \wedge \tau_1(\varphi).$$

Proof. (\Rightarrow): Suppose that φ is satisfiable, and let $w = a_1 \dots a_k \in \Sigma^+$ be a word satisfying it. By Lemma 4.24, the marked word $\bar{w} = a_1 \dots \bar{a}_k \in \Sigma^* \bar{\Sigma}$ satisfies φ with respect to the satisfaction relation \models_\bullet . Notice that \bar{w} contains only one marked symbol. Let (\mathcal{F}, t, n) be a pointed forest encoding $f(\bar{w})$. By Lemma 4.26, $(\mathcal{F}, t, n) \models \tau_1(\varphi)$. Moreover, as w is not empty we derive that (\mathcal{F}, t, n) satisfies $\langle U \rangle \text{Hit}$, and by Lemma 4.10 it satisfies $\text{word}_{[1,2n]}$. As shown in Lemma 4.8, the formula $\text{Hit} \wedge \neg \blacklozenge(\text{Miss})$ is only satisfied when the current evaluation node corresponds to a child of t . Instead, the formula mark_Σ is only satisfied if the current node encodes a marked symbol (Lemma 4.25(I)). We then conclude that (\mathcal{F}, t, n) also satisfies $[U] (\text{mark}_\Sigma \Leftrightarrow (\text{Hit} \wedge \neg \blacklozenge(\text{Miss})))$. Indeed, \bar{a}_k is the only marked symbol of \bar{w} and, by definition of encoding (Definition 4.4), it is encoded by the only \mathcal{F} -child of t .

(\Leftarrow): Suppose $\text{word}_{[1,2n]} \wedge \langle U \rangle \text{Hit} \wedge [U] (\text{mark}_\Sigma \Leftrightarrow (\text{Hit} \wedge \neg \blacklozenge(\text{Miss}))) \wedge \tau_1(\varphi)$ satisfiable, and let (\mathcal{F}, t, n) be a pointed forest satisfying it. From the satisfaction of $\text{word}_{[1,2n]}$ and $\langle U \rangle \text{Hit}$, by Lemma 4.10, (\mathcal{F}, t, n) is an encoding of a non-empty word in $[1, 2n]^+$. Let $b_1 \dots b_k$ be this word and let n_k be the node corresponding to b_k . By definition of encoding, n_k is the only child of t . Thus, from $(\mathcal{F}, t, n) \models [U] (\text{mark}_\Sigma \Leftrightarrow (\text{Hit} \wedge \neg \blacklozenge(\text{Miss})))$, together with Lemma 4.25(I) and Lemma 4.8, we conclude that n_k is the only node of $\text{dom}(\mathcal{F})$ encoding a marked symbol. This means that $b_1 \dots b_k$ is of the form $a_1 \dots \bar{a}_k \in \Sigma^* \bar{\Sigma}$. From $(\mathcal{F}, t, n) \models \tau_1(\varphi)$ and by Lemma 4.26 $a_1 \dots \bar{a}_k \models_\bullet \varphi$. Lastly, $a_1 \dots a_k \models \varphi$ by Lemma 4.24. \square

Because of the four disjuncts appearing in the formula $\tau_\beta(\varphi \mid \psi)$, the translation is exponential in the number of symbols used to write the PITL formula. Since the satisfiability problem of PITL is TOWER-hard (Proposition 4.20), an elementary translation is all we need in order to conclude that the satisfiability problem of ALT is also TOWER-hard, directly by Lemma 4.27. Decidability in TOWER stems from Proposition 4.3.

Theorem 4.28. The satisfiability problem of ALT is TOWER-complete.

4.4 REVISITING TOWER-HARD LOGICS WITH ALT

Strong of Theorem 4.28, we now display the usefulness of ALT as a tool for proving the TOWER-hardness of logics interpreted on tree-like structures. We start by revisiting the connections

between ALT and separation logic, and show that even when the separating implication is heavily restricted, the logic $\text{SL}(*, \neg*, \text{ls})$ studied in Chapter 3 still admits a non-elementary satisfiability problem. Afterwards, we provide semantically faithful reductions from ALT to three logics whose satisfiability problems were independently found to be TOWER-complete: quantified computation tree logic on trees [99], modal logic of heaps [52] and modal separation logic [54]. Thanks to the simplicity of ALT, our reductions only use strict fragments of these formalisms, allowing us to refine their non-elementary boundaries. In order to keep the presentation light, the proofs (all quite simple) of the results of this section are almost exclusively given in Appendix B.

4.4.1 From ALT to $\text{SL}(*, \neg*, \text{ls})$ with bounded magic wand.

In [22] the authors show that $\text{SL}(\exists, *, \neg*)$ becomes TOWER-complete when the separating implication is restricted so that the formula on the left side only admits small-models. In particular, given $n \in \mathbb{N}$, they introduce the *bounded magic wand* $\varphi \neg[n] \psi$ defined as $(\varphi \wedge \neg\text{size} \geq n+1) \rightarrow \psi$, and show that $\text{SL}(\exists, *, \neg[n])$ admits a TOWER-complete satisfiability problem. Since we have shown that ALT is a fragment of the separation logic $\text{SL}([\exists]_1, *, \mathbf{x} \hookrightarrow _, \hookrightarrow^+)$, and from Chapter 3 we know that the separating implication can be used to mimic first-order quantifications, it is quite natural to ask ourselves whether ALT can be used to refine the TOWER-hardness of $\text{SL}(\exists, *, \neg[n])$. In this section we answer this question by showing the following result.

Theorem 4.29. Satisfiability of the two-variable fragment of $\text{SL}(*, \neg[1], \text{ls})$ is TOWER-c.

The grammar of the formulae φ in $\text{SL}(*, \neg[1], \text{ls})$ is given below:

$$\varphi ::= \top \mid \text{emp} \mid \mathbf{x} = \mathbf{y} \mid \mathbf{x} \hookrightarrow \mathbf{y} \mid \text{ls}(\mathbf{x}, \mathbf{y}) \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi * \varphi \mid \varphi \neg[1] \varphi.$$

Notice that the operator $\varphi \neg[n] \psi$ is restricted to $n = 1$, which leads to the following semantics:

$(s, h) \models \varphi \neg[1] \psi$ iff for every heap h' , if $h' \perp h$, $\text{card}(h') \leq 1$ and $(s, h) \models \varphi$, then $(s, h + h') \models \psi$.

Besides, the logic can express $\mathbf{x} \hookrightarrow^* \mathbf{y}$ and $\text{size} = \beta$ as defined in Section 2.1.1. The alloc formula $\mathbf{x} \hookrightarrow _$ is equivalent to $\mathbf{x} \hookrightarrow \mathbf{x} \neg[1] \perp$. We define the *bounded subtraction* $\neg[1]$ as the right dual of the bounded magic wand, i.e. $\varphi \neg[1] \psi \stackrel{\text{def}}{=} \neg(\varphi \neg[1] \neg\psi)$.

Let us discuss how to encode a pointed forest $(\mathcal{F}, \mathbf{t}, \mathbf{n})$ as a memory state (s, h) . Without loss of generality, we assume $\mathcal{N} = \text{LOC}$. As we have done in Chapter 3, we use the location assigned to a fixed variable \mathbf{x} in order to mimic the first-order quantification of $\langle U \rangle \varphi$. So, in the encoding, the location $h(s(\mathbf{x}))$ corresponds to the current node \mathbf{n} and, in order to mimic the quantification correctly, it must be different from $s(\mathbf{x})$. In order to encode the target node we rely on the location assigned to a second program variable \mathbf{y} , and require that both $s(\mathbf{y}) = \mathbf{t} \neq s(\mathbf{x})$ and $s(\mathbf{y}) \notin \text{dom}(h)$ hold. Notice that this last condition is without loss of generality, as we can assume that $\mathbf{t} \notin \text{dom}(\mathcal{F})$ by Lemma 4.18(I). The encoding is formalised as follows.

Definition 4.30 (Forests as heaps). (s, h) is an (\mathbf{x}, \mathbf{y}) -encoding of $(\mathcal{F}, \mathbf{t}, \mathbf{n})$, where $\mathbf{x}, \mathbf{y} \in \text{VAR}$, iff

1. $h = \mathcal{F} + \{s(\mathbf{x}) \mapsto \mathbf{n}\}$ (seeing \mathcal{F} as a heap),
2. $\mathbf{n} \neq s(\mathbf{x})$,
3. $s(\mathbf{y}) = \mathbf{t} \notin \text{dom}(h)$.

With respect to memory states that are (\mathbf{x}, \mathbf{y}) -encodings of some pointed forest, given a formula φ in ALT we translate it into a formula $\tau_{\mathbf{x}, \mathbf{y}}(\varphi)$ in $\text{SL}(*, \neg[1], \text{ls})$ following the definition in Figure 4.9. The figure omits the cases for \top and Boolean connectives, which are defined homomorphically (e.g. $\tau_{\mathbf{x}, \mathbf{y}}(\neg\varphi) \stackrel{\text{def}}{=} \neg\tau_{\mathbf{x}, \mathbf{y}}(\varphi)$, as in the translation τ_β from PITL to ALT). The

$$\begin{aligned}
\tau_{x,y}(\text{Hit}) &\stackrel{\text{def}}{=} x \hookrightarrow^* y \wedge \neg x \hookrightarrow y, \\
\tau_{x,y}(\text{Miss}) &\stackrel{\text{def}}{=} \neg \tau_{x,y}(\text{Hit}) \wedge \neg x \hookrightarrow y \wedge (\text{size} = 1 \dashv [1] \neg(\top * (\text{ls}(x, y) \wedge \text{size} = 2))), \\
\tau_{x,y}(\blacklozenge \varphi) &\stackrel{\text{def}}{=} \blacklozenge_{\text{SL}}(x \hookrightarrow _ \wedge \tau_{x,y}(\varphi)), \\
\tau_{x,y}(\blacklozenge^* \varphi) &\stackrel{\text{def}}{=} \blacklozenge_{\text{SL}}^*(x \hookrightarrow _ \wedge \tau_{x,y}(\varphi)), \\
\tau_{x,y}(\langle U \rangle \varphi) &\stackrel{\text{def}}{=} (\text{size} = 1 \wedge x \hookrightarrow _) * (\text{size} = 1 \dashv [1] (x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \tau_{x,y}(\varphi))).
\end{aligned}$$

Figure 4.9: Translation from ALT to $\text{SL}(*, \dashv, \text{ls})$ with bounded magic wand.

translation of \blacklozenge and \blacklozenge^* uses the analogous formulae $\blacklozenge_{\text{SL}} \varphi \stackrel{\text{def}}{=} \text{size} = 1 * \varphi$ and $\blacklozenge_{\text{SL}}^* \varphi \stackrel{\text{def}}{=} \top * \varphi$, already introduced in Section 4.4.1, while taking care that the location assigned to x is not discharged from the domain of the heap. The translation of $\langle U \rangle \varphi$ is very close to the translation of the first-order quantification performed in Chapter 3: the pair $s(x) \mapsto h(s(x))$ in the heap h is replaced with some $s(x) \mapsto \ell'$, leading to the satisfaction of $\tau_{x,y}(\varphi)$. While the translation of **Hit** is also quite straightforward, the translation of **Miss** requires some work. In particular, as in Chapter 3, this predicate must be checked on $h(s(x))$ and should hold only if this location is in the domain of the heap but does not reach $s(y)$ in at least one step. The formula $\tau_{x,y}(\text{Miss})$ achieves this by stating that it is not possible to add one arrow to the heap in order to construct a path of length two going from $s(x)$ to $s(y)$. Indeed, under the hypothesis that $h(s(x))$ is not in the domain of the heap, such a path can always be constructed by adding $\{h(s(x)) \mapsto s(y)\}$ to the heap. So, $h(s(x))$ must be in $\text{dom}(h)$ which, together with $\neg \tau_{x,y}(\text{Hit})$, effectively captures the semantics of **Miss**. The correctness of the translation is formalised below.

Lemma 4.31. Let (s, h) be an (x, y) -encoding of a pointed forest (\mathcal{F}, t, n) . Let φ be a formula in ALT. We have, $(\mathcal{F}, t, n) \models \varphi$ if and only if $(s, h) \models \tau_{x,y}(\varphi)$.

The proof is by structural induction on φ . The formula $x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \neg y \hookrightarrow _$ characterises the set of memory states encoding pointed forests, which allows us to show the lemma below, concluding the reduction. For the proof of this last lemma we also rely on Lemma 4.18(I) in order to only consider pointed forests that admit an encoding. Lemma 4.32 implies Theorem 4.29.

Lemma 4.32. φ in ALT is satisfiable iff so is $x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \neg y \hookrightarrow _ \wedge \tau_{x,y}(\varphi)$ in $\text{SL}(*, \dashv, \text{ls})$.

Interestingly, Theorem 4.29 already holds when the bounded magic wand is limited to formulae of the form $\text{size} = 1 \dashv [1] \varphi$. Indeed, the alloc formula $x \hookrightarrow _$ can be substituted with the equivalent formula $x \hookrightarrow x \vee (\text{size} = 1 \dashv [1] \neg x \hookrightarrow x)$, so that the translation only uses the bounded magic wand in this way. This result has a role in the developments of Chapter 5.

Corollary 4.33. $\text{SL}(*, \dashv, \text{ls})$ where \dashv is restricted to $\text{size} = 1 \dashv [1] \varphi$ is TOWER-complete.

4.4.2 From ALT to Quantified Computation Tree Logic.

We now consider *Computation Tree Logic* (CTL), a well-known logic for branching time model checking [43, 42]. Among its extensions, in [99] the addition of propositional quantifiers is considered. The resulting logic, called *Quantified Computation Tree Logic* (QCTL) is undecidable on Kripke structures, and TOWER-complete on trees (QCTL^t). This non-elementary boundary has been recently refined in [8]: even when considering just one operator among *exists-next* EX

or *exists-finally* EF (the definitions are below), QCTL^t still admits a TOWER-complete satisfiability problem. Here, we reprove the result for EF by first tackling the TOWER-hardness of the logic with the *exists-until* $\text{E}(\varphi \cup \psi)$, to then show that this operator can be defined using EF . Differently from [8] and thanks to the properties of ALT , our reduction does not imbricate until operators, showing that this extension of CTL remains TOWER-hard even when $\text{E}(\varphi \cup \psi)$ is restricted so that φ and ψ are Boolean combinations of propositional symbols.

Let us start by recalling the syntax of QCTL, as defined in [99]. We use AP to denote the countable set of *propositional symbols* $\{p, q, \dots\}$. The formulae φ of QCTL are built from the following grammar (where $p \in \text{AP}$):

$\pi :=$	\top (<i>true</i>)	$\varphi :=$	π	(<i>atomic formulae</i>)
	p (<i>propositional symbol</i>)		$\varphi \wedge \varphi$ $\neg \varphi$	(<i>Boolean connectives</i>)
			$\text{EX } \varphi$	(<i>exists-next modality</i>)
			$\text{E}(\varphi \cup \varphi)$	(<i>exists-until modality</i>)
			$\text{A}(\varphi \cup \varphi)$	(<i>all-until modality</i>)
			$\exists p \varphi$	(<i>propositional quantification</i>)

QCTL is interpreted on standard Kripke structures [98].

Definition 4.34 (Kripke structure). A *Kripke structure* is a triple $(\mathcal{W}, R, \mathcal{V})$ where \mathcal{W} is a countable set of *worlds*, $R \subseteq \mathcal{W} \times \mathcal{W}$ is a left-total¹ accessibility relation and $\mathcal{V} : \text{AP} \rightarrow 2^{\mathcal{W}}$ is a *labelling function* which, given a propositional symbol p , returns the set of worlds satisfying p .

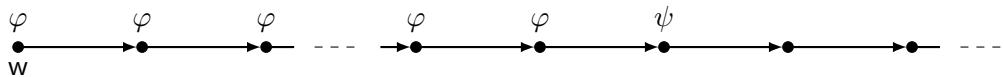
The satisfaction of the exists-until and all-until modalities depends on the paths in the structure.

Definition 4.35 (Path). Let $R \subseteq \mathcal{W} \times \mathcal{W}$ be a binary relation on worlds (possibly left-total). A *path* ρ starting in w is a (possibly finite) sequence of worlds (w_0, w_1, \dots) such that $w_0 = w$ and $(w_i, w_{i+1}) \in R$ for every two successive elements w_i, w_{i+1} of the sequence.

(*Maximal Path*) The path ρ is said to be *maximal* whenever it is not a strict prefix of any other path. We denote with $\Pi_R(w)$ the set of *maximal paths* starting in w .

Notice that if R is left-total then $\Pi_R(w)$ is the set of all infinite paths starting in w . Given a world $w \in \mathcal{W}$, we write $R(w)$ for the set $\{w' \in \mathcal{W} \mid (w, w') \in R\}$, i.e. the set of worlds that are *accessible* from w . Therefore $R^*(w)$ (where R^* is the Kleene closure of R) denotes the set of worlds reachable from w , i.e. the worlds belonging to a path in $\Pi_R(w)$.

Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ be a Kripke structure and consider $w \in \mathcal{W}$. The pair (\mathcal{K}, w) denotes a *pointed Kripke structure*, where the world w is the *current world*. Given (\mathcal{K}, w) , The satisfaction relation \models for formulae in QCTL is defined in Figure 4.10 (as usual, omitting standard cases for \top and Boolean connectives). The atomic formula p simply asks whether the propositional symbol p is satisfied by w . Using the exists-next modality $\text{EX } \varphi$, we can check whether φ holds in a world that is accessible from w . The two *temporal modalities* are more sophisticated. The formula $\text{E}(\varphi \cup \psi)$ checks whether there is a maximal path (w_0, w_1, \dots) starting in w for which there is a finite prefix (w_0, \dots, w_{j-1}) of worlds satisfying φ , followed by the world $w_j \in R(w_{j-1})$ that satisfies ψ . This path can be schematised as follows (arrows represent R , every “•” is a world).



¹left-total means that for each world $w \in \mathcal{W}$ there is $w' \in \mathcal{W}$ such that $(w, w') \in R$.

$$\begin{aligned}
(\mathcal{K}, w) \models p &\quad \text{iff } w \in \mathcal{V}(p), \\
(\mathcal{K}, w) \models \text{EX } \varphi &\quad \text{iff } \exists w' \in R(w) \text{ such that } (\mathcal{K}, w') \models \varphi, \\
(\mathcal{K}, w) \models E(\varphi \cup \psi) &\quad \text{iff there are } (w_0, w_1, \dots) \in \Pi_R(w) \text{ and } j \in \mathbb{N} \text{ such that} \\
&\quad (\mathcal{K}, w_j) \models \psi \text{ and for every } i < j, (\mathcal{K}, w_i) \models \varphi, \\
(\mathcal{K}, w) \models A(\varphi \cup \psi) &\quad \text{iff for all } (w_0, w_1, \dots) \in \Pi_R(w), \text{ there is } j \in \mathbb{N} \text{ such that} \\
&\quad (\mathcal{K}, w_j) \models \psi \text{ and for every } i < j, (\mathcal{K}, w_i) \models \varphi, \\
(\mathcal{K}, w) \models \exists p \varphi &\quad \text{iff there is } \mathcal{W}' \subseteq \mathcal{W} \text{ such that } (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W}']) \models \varphi.
\end{aligned}$$

Figure 4.10: Satisfaction relation for QCTL.

The formula $A(\varphi \cup \psi)$ asks the above property to hold for every maximal path, instead of at least one: given a maximal path (w_0, w_1, \dots) , there must be a finite prefix (w_0, \dots, w_{j-1}) of worlds satisfying φ , followed by the world $w_j \in R(w_{j-1})$ that satisfies ψ . Lastly, the propositional existential quantification $\exists p \varphi$ is quite similar to the second-order quantification of second-order logic. Essentially, this formula is satisfied if it is possible to update the satisfaction of the propositional symbol p to a new subset \mathcal{W}' of \mathcal{W} , so that then φ holds. In the formal definition given in Figure 4.10, this update is written as $\mathcal{V}[p \leftarrow \mathcal{W}']$. Similarly to the store update $s[u \leftarrow \ell']$ of the existential quantification of $SL(\exists, *, -)$, this notation stands for the function obtained from \mathcal{V} by changing the evaluation of p from $\mathcal{V}(p)$ to \mathcal{W}' .

The universal quantification $\forall p$ and the Boolean connectives \Rightarrow and \vee are defined as usual. So are the classical temporal operators of CTL, from [43]:

$$\begin{aligned}
\text{EF } \varphi &\stackrel{\text{def}}{=} E(\top \cup \varphi) && \text{(exists-finally)} \\
\text{AG } \varphi &\stackrel{\text{def}}{=} \neg \text{EF } \neg \varphi && \text{(all-generally)} \\
\text{AF } \varphi &\stackrel{\text{def}}{=} A(\top \cup \varphi) && \text{(all-finally)} \\
\text{EG } \varphi &\stackrel{\text{def}}{=} \neg \text{AF } \neg \varphi && \text{(exists-generally)} \\
E(\varphi M \psi) &\stackrel{\text{def}}{=} E(\varphi \cup \varphi \wedge \psi) && \text{(exists-strong-release)}
\end{aligned}$$

From [99], the satisfiability problem of QCTL is known to be undecidable on arbitrary Kripke structures, whereas it becomes TOWER-complete when the interpretation is restricted to the class of Kripke trees. We denote this restriction with QCTL^t .

Definition 4.36 (Kripke tree). A Kripke structure $(\mathcal{W}, R, \mathcal{V})$ is a *(finitely-branching) Kripke tree* if

1. R^{-1} is functional and acyclic,
2. for every world $w \in \mathcal{W}$, $R(w)$ is finite,
3. it has a *root*, i.e. $R^*(r) = \mathcal{W}$ for some $r \in \mathcal{W}$.

Given $w \in \mathcal{W}$, the worlds in $R^*(w) \setminus \{w\}$ are said to be *descendants* of w . As Kripke structures are left-total, Kripke trees can be seen as finitely-branching infinite trees. This correspondence leads to the satisfiability problem of QCTL being in TOWER by reduction to monadic second-order logic on trees [99].

From ALT to QCTL^t. In the following, we only consider Kripke trees instead of arbitrary Kripke structures (and write *pointed Kripke tree* instead of pointed Kripke structure), and we

aim at reducing the satisfiability problem of ALT to the satisfiability problem of QCTL^t. The semantics of the formula $\exists p \varphi$ should already give a good clue on how to perform such a reduction. Informally speaking, we can represent the nodes of a finite forest as the set of worlds satisfying a propositional symbol D . Then, for instance, the modality \blacklozenge^* can be encoded by using an existential $\exists E$ that changes the evaluation of a propositional symbol E so that it only holds on a subset of the worlds satisfying D (as in the semantics of the repeated sabotage modality). If we are able to check whether only one world satisfies D but not E , we can then also capture the semantics of the sabotage modality \blacklozenge . Similarly, the propositional quantification can be used to encode the universal modality $\langle U \rangle$, whereas for the reachability predicates **Hit** and **Miss** we can rely on the exists-until modality.

Let us discuss this encoding a little bit further. Let (\mathcal{F}, t, n) be a pointed forest that we want to encode as a pointed Kripke structure (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a Kripke tree. We use w to play the role of the target node t . To encode the forest \mathcal{F} and the current evaluation node n we use the worlds appearing in $R^*(w)$ and three propositional symbols: D , **end** and n . The intended use of D is to state which elements of $R^*(w)$ encode nodes in $\text{dom}(\mathcal{F})$. We need to be careful here, as $R^*(w)$ is an infinite set whereas $\text{dom}(\mathcal{F})$ is finite. We use the propositional symbol **end** to solve this inconsistency: we constraint \mathcal{K} to satisfy the formula $\text{AF}(\text{end})$ stating that every maximal path $(w_0, w_1, \dots) \in \Pi_R(w)$ has a finite prefix (w_0, \dots, w_{j-1}) ($j \in \mathbb{N}$) of worlds not satisfying **end**, whereas $w_j \in \mathcal{V}(\text{end})$. Then, a world in \mathcal{W} encodes an element in $\text{dom}(\mathcal{F})$ whenever it satisfies D and it belongs to one of these prefixes. We use the propositional symbol n to encode the current evaluation node. During the translation, we require n to be satisfied by exactly one descendant of w , so that the modality $\langle U \rangle$ roughly becomes a quantification over n . For technical reasons, we treat in a similar way the world w , which encodes the target node, and require it to be the only world (among the ones in $R^*(w)$) satisfying the auxiliary propositional symbol t . Lastly, we use an additional propositional symbol E in order to encode subforests and deal with the encoding of \blacklozenge and \blacklozenge^* (as already stated above). Notice that we can use the following formula from [99] to check if a formula φ holds in exactly one descendant of w :

$$\text{uniq}(\varphi) \stackrel{\text{def}}{=} \text{EF}(\varphi) \wedge \forall p (\text{EF}(\varphi \wedge p) \Rightarrow \text{AG}(\varphi \Rightarrow p)),$$

where $p \in \text{AP}$ is a propositional symbol that does not appear in φ .

Proposition 4.37 (From [99]). Let (\mathcal{K}, w) be a pointed Kripke structure, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a Kripke tree. $(\mathcal{K}, w) \models \text{uniq}(\varphi)$ iff there is exactly one $w' \in R^*(w)$ such that $(\mathcal{K}, w') \models \varphi$.

For the rest of the section, we fix a tuple $S \stackrel{\text{def}}{=} (\text{end}, n, t)$ of three different propositional symbols, and two (distinct) additional symbols D and E not in S . We also restrict ourselves to pointed forests (\mathcal{F}, t, n) such that $t \notin \text{dom}(\mathcal{F})$. From Lemma 4.18(I), this restriction is without loss of generality. We formally define the encoding of pointed forests into pointed Kripke trees.

Definition 4.38 (QCTL^t - Pointed forests encoding). A pointed Kripke tree $(\mathcal{K} = (\mathcal{W}, R, \mathcal{V}), w)$ is an (S, D) -encoding of (\mathcal{F}, t, n) if there is an injection $f : \mathcal{N} \rightarrow R^*(w)$ such that

1. $f(t) \stackrel{\text{def}}{=} w$ is the only world in $\text{ran}(f) \cap \mathcal{V}(t)$, and $f(n)$ is the only world in $\text{ran}(f) \cap \mathcal{V}(n)$,
2. for every $n' \in \text{dom}(\mathcal{F})$, it holds that $(f(\mathcal{F}(n')), f(n')) \in R$,
3. for every infinite path $(w_0, w_1, \dots) \in \Pi_R(w)$ there is $i \in \mathbb{N}$ such that
 - a. $w_i \in \mathcal{V}(\text{end})$ and for every $j \in [0, i-1]$ we have $w_j \notin \mathcal{V}(\text{end})$,
 - b. for every $j \in \mathbb{N}$, $(w_j \in \mathcal{V}(D))$ and $j < i$ if and only if there is $n' \in \text{dom}(\mathcal{F})$ $f(n') = w_j$.

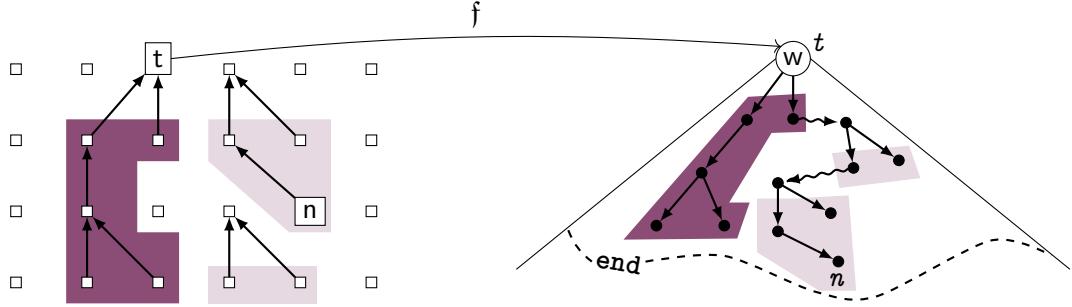


Figure 4.11: A pointed forest (left) and one of its encoding as a Kripke tree (right).

$$\begin{aligned}
 \tau_u(\text{Hit}) &\stackrel{\text{def}}{=} E(((u \vee t) \wedge \neg \text{end}) M (u \wedge n)), \\
 \tau_u(\text{Miss}) &\stackrel{\text{def}}{=} E(\neg \text{end} M (u \wedge n)) \wedge \neg \tau_u(\text{Hit}), \\
 \tau_u(\langle U \rangle \varphi) &\stackrel{\text{def}}{=} \exists n (\text{uniq}(n) \wedge \tau_u(\varphi)), \\
 \tau_u(\blacklozenge^* \varphi) &\stackrel{\text{def}}{=} \exists \bar{u} (AG(\bar{u} \Rightarrow u) \wedge \tau_{\bar{u}}(\varphi)), \\
 \tau_u(\blacklozenge \varphi) &\stackrel{\text{def}}{=} \exists \bar{u} (AG(\bar{u} \Rightarrow u) \wedge \text{uniq}(u \wedge \neg \bar{u}) \wedge E(\neg \text{end} M (u \wedge \neg \bar{u})) \wedge \tau_{\bar{u}}(\varphi)).
 \end{aligned}$$

Figure 4.12: Translation from ALT to QCTL.

We simply write *encoding* when (S, D) is clear from the context.

For instance, Figure 4.11 shows a possible encoding of a pointed forest into a pointed Kripke tree. Informally, the property (1) states that w encodes t and is the only world in $R^*(w)$ satisfying t . Similarly, the world $f(n)$ encoding n is the only world in $R^*(w)$ that satisfies n . The property (2) states that the forest must be correctly encoded in the Kripke structure. In particular, notice that the parent relation of the finite forest is inverted so that it becomes the child relation in the Kripke structure (as shown in Figure 4.11). As f is an injection, the encoding does not merge together subforests that are disconnected in \mathcal{F} . Lastly, the property (3) of f states that the elements in $\text{dom}(\mathcal{F})$ must be encoded by nodes in $R^*(w)$ that precede every world satisfying end . Moreover, among all the descendants of w preceding end , the worlds encoding $\text{dom}(\mathcal{F})$ are the only ones satisfying D . As $t \notin \text{dom}(\mathcal{F})$, t does not satisfy D . It is quite easy to see that every pointed forest (\mathcal{F}, t, n) such that $t \notin \text{dom}(\mathcal{F})$ admits an encoding.

We now formalise the translation of a formula in ALT into a formula in QCTL. During the translation, we alternate between D and E in order to keep track of the domain of the forest, following a \blacklozenge or \blacklozenge^* operator. To facilitate this alternation, we define $\overline{D} \stackrel{\text{def}}{=} E$ and $\overline{E} \stackrel{\text{def}}{=} D$. The translation $\tau_u(\varphi)$ in QCTL of a formula φ in ALT is parametrised by $u \in \{D, E\}$ and, implicitly, by S . It is homomorphic for T and Boolean connectives, and otherwise it is defined in Figure 4.12. Let (\mathcal{F}, t, n) be a pointed forest such that $t \notin \text{dom}(\mathcal{F})$ and let $((\mathcal{W}, R, \mathcal{V}), w)$ be one of its (S, u) -encodings. Let f be the injection certifying that the encoding hold (as in Definition 4.38). For instance, $\tau_u(\text{Hit})$ requires that there is a path (w, w_1, \dots, w_j) starting in $f(t) = w$ and whose worlds do not satisfy end and must satisfy u or t . Moreover, the last world w_j must satisfy u and n . From the property (1) of the definition of f , the only world satisfying t is w , which does not satisfy u . Moreover, n is only satisfied by $f(n)$. Lastly, from the property (3) of f , worlds satisfying u and preceding the ones that satisfy end must encode nodes in the domain of the

forest \mathcal{F} . Therefore, the path (w, w_1, \dots, w_j) corresponds to a path in the pointed forest, going from the current evaluation node n (which is encoded by the only world satisfying n) to the target node t . The correctness of the translation follows from the lemma below.

Lemma 4.39. Let (\mathcal{F}, t, n) be a pointed forest such that $t \notin \text{dom}(\mathcal{F})$, and let (\mathcal{K}, w) be a (S, u) -encoding of (\mathcal{F}, t, n) . Given a formula φ in ALT, $(\mathcal{F}, t, n) \models \varphi$ if and only if $(\mathcal{K}, w) \models \tau_u(\varphi)$.

In order to conclude the reduction we just need to characterise the set of pointed Kripke trees encoding pointed forests. This can be done with $\text{enc} \stackrel{\text{def}}{=} \neg D \wedge t \wedge \text{uniq}(t) \wedge \text{uniq}(n) \wedge \text{AF}(\text{end})$.

Lemma 4.40. A formula φ in ALT is satisfiable iff so is $\text{enc} \wedge \tau_D(\varphi)$ in QCTL t .

Notice that for the left-to-right direction we need to rely on Lemma 4.18(I) in order to produce a pointed forest (\mathcal{F}, t, n) satisfying φ and such that $t \notin \text{dom}(\mathcal{F})$. Only afterwards we can define an encoding in terms of pointed Kripke trees.

Tower-hard fragments of QCTL t . We now take a closer look to the translation. Given a temporal modality \mathcal{T} (e.g. EF) and $k \in \mathbb{N} \cup \{\omega\}$, we write QCTL $^t(\mathcal{T}^k)$ to denote the fragment of QCTL t restricted to formulae where the only temporal modality allowed is \mathcal{T} , which can be nested at most k times (ω stands for an arbitrary number of imbrications). For instance, QCTL $^t(\text{EF}^k)$ denotes the set of formulae restricted to the operator EF, which can be nested at most k times. This logic is shown to be k -NEXPTIME-hard in [8], which directly leads to the TOWER-hardness of QCTL $^t(\text{EF}^\omega)$ and QCTL $^t(\text{EU}^\omega)$. By analysing our translation it is easy to show that QCTL $^t(\text{EU}^0)$, i.e. QCTL restricted to the only modality E($\varphi \cup \psi$) where φ and ψ are Boolean combination of propositional symbols, and QCTL $^t(\text{EF}^1)$ are already TOWER-hard.

Theorem 4.41. The satisfiability problems of QCTL $^t(\text{EU}^0)$ and QCTL $^t(\text{EF}^1)$ are TOWER-c.

Let us first informally discuss this result. Let us fix a pointed Kripke tree $((\mathcal{W}, R, \mathcal{V}), w)$. First of all, an exists-until modality E($\varphi \cup \psi$) in QCTL $^t(\text{EU}^0)$ can be shown equivalent to the formula $\chi_{\text{EU}}(\varphi, \psi)$, in QCTL $^t(\text{EF}^1)$, defined below:

$$\chi_{\text{EU}}(\varphi, \psi) \stackrel{\text{def}}{=} \exists p (\text{AG}(\neg\varphi \wedge \neg\psi \Rightarrow p) \wedge \text{AG}(p \Rightarrow \text{AG}p) \wedge \text{EF}(\psi \wedge \neg p)),$$

where p is a propositional symbol that does not appear in φ or ψ . The idea behind this formula is quite simple. The formula $\chi_{\text{EU}}(\varphi, \psi)$ states that it is possible to change the evaluation of the symbol p so that, for every path starting from the current world w , p holds whenever $\neg\varphi \wedge \neg\psi$ holds (first conjunct of the formula), and if p holds in a world, then it holds on every world reachable from it (second conjunct of the formula). Lastly, the third conjunct states that it is possible to find a world $w' \in R^*(w)$ satisfying $\psi \wedge \neg p$. This means that the path going from w to w' cannot witness worlds satisfying $\neg\varphi \wedge \neg\psi$, which in turn implies that E($\varphi \cup \psi$) is satisfied. Thanks to $\chi_{\text{EU}}(\varphi, \psi)$, we just need to prove Theorem 4.41 for QCTL $^t(\text{EU}^0)$.

Clearly, the translation τ_u is defined so that the resulting formula is already in QCTL $^t(\text{EU}^0)$. However, we need to deal with the occurrence of AF(end) used inside the formula enc . Let us first consider the formula AG($\varphi \Rightarrow \text{AG}\psi$) which is satisfied by models where once φ is found to hold in a certain world w , then ψ is satisfied in every world of $R^*(w)$. Despite not being in QCTL $^t(\text{EU}^0)$, the formula AG($\varphi \Rightarrow \text{AG}\psi$) is equivalent to the formula $\chi_{\text{AGAG}}(\varphi, \psi)$ below:

$$\chi_{\text{AGAG}}(\varphi, \psi) \stackrel{\text{def}}{=} \forall p \forall q (\text{uniq}(p) \wedge \text{uniq}(q) \wedge \text{EF}(p \wedge \varphi) \wedge \text{EF}(q \wedge \neg\psi \Rightarrow \text{E}(\neg p \mathbin{M} q))),$$

where p and q do not appear in φ or ψ . This formula characterises $\text{AG}(\varphi \Rightarrow \text{AG } \psi)$ by stating that whenever we pick two worlds $w_1, w_2 \in R^*(w)$, if w_1 satisfies φ and w_2 does not satisfy ψ , then it is possible to find a path going from the current world w to w_2 that does not include w_1 .

Lastly, we define a formula $\chi_{\text{EG}}(\varphi)$ that only uses EF modalities and is equivalent to $\text{EG } \varphi$, so that then $\neg \chi_{\text{EG}}(\neg \varphi)$ is equivalent to $\text{AF } \varphi$:

$$\begin{aligned}\chi_{\text{EG}}(\varphi) \stackrel{\text{def}}{=} & \exists p (\neg p \wedge \text{AG}(\neg \varphi \Rightarrow p) \wedge \text{AG}(p \Rightarrow \text{AG } p) \wedge \\ & \forall q (\text{uniq}(q) \wedge \text{EF}(q \wedge \neg p) \Rightarrow \text{EF}(q \wedge \text{EF}(\neg q \wedge \neg p))),\end{aligned}$$

where p does not appear in φ . This formula is expressible in $\text{QCTL}^t(\text{EU}^0)$, as every subformula that is not in this fragment is an instance of $\text{AG}(\varphi \Rightarrow \text{AG } \psi)$. From the correctness of this formula, proved below, we conclude that $\text{AF}(\text{end})$ is expressible in $\text{QCTL}^t(\text{EU}^0)$, leading to Theorem 4.41. Differently from the formulae $\chi_{\text{EU}}(\varphi, \psi)$ and $\chi_{\text{AG AG}}(\varphi, \psi)$ (proved correct in Appendix B), understanding why $\chi_{\text{EG}}(\varphi)$ captures $\text{EG } \varphi$ is not immediate. Instead of giving just an informal explanation, we directly show the formal proof.

Proof of $\text{EG } \varphi \equiv \chi_{\text{EG}}(\varphi)$. Below, we consider a pointed Kripke tree (\mathcal{K}, w) where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.
 (\Rightarrow) : Suppose $(\mathcal{K}, w) \models \text{EG } \varphi$, and therefore that there is an infinite path $\rho \in \Pi_R(w)$ where for every $i \geq 0$ the i -th world w_i of the path ρ is such that $(\mathcal{K}, w_i) \models \varphi$. We write $\widehat{\mathcal{W}}$ for the set of worlds in ρ . Let us consider the model $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W} \setminus \widehat{\mathcal{W}}])$ obtained from \mathcal{K} by changing the evaluation of p to the set of worlds that are not in $\widehat{\mathcal{W}}$. Since w belongs to ρ , we have $(\mathcal{K}', w) \models \neg p$. Moreover, as every world in $\widehat{\mathcal{W}}$ satisfies φ whereas every world in $\mathcal{W} \setminus \widehat{\mathcal{W}}$ satisfies p , we conclude that $(\mathcal{K}', w) \models \text{AG}(\neg \varphi \Rightarrow p)$. Similarly, (\mathcal{K}', w) satisfies $\text{AG}(p \Rightarrow \text{AG } p)$. Indeed, let us consider a world $w' \in R^*(w)$ such that $w' \in \mathcal{V}(p)$, and show that for every $w'' \in R^*(w')$, $w'' \in \mathcal{V}(p)$ (as required by this formula). By definition, $w' \notin \widehat{\mathcal{W}}$. As \mathcal{K}' is a Kripke tree (in particular, it is an acyclic structure), every world w'' reachable from w' does not belong to ρ . So, by definition of \mathcal{K}' , $w'' \in \mathcal{V}(p)$. Lastly, let us focus on the subformula

$$\forall q (\text{nom}(q) \wedge \text{EF}(q \wedge \neg p) \Rightarrow \text{EF}(q \wedge \text{EF}(\neg q \wedge \neg p))).$$

We consider a Kripke tree $\mathcal{K}'' = (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W} \setminus \widehat{\mathcal{W}}][q \leftarrow \mathcal{W}''])$ obtained from \mathcal{K}' by updating the evaluation of q , and such that $(\mathcal{K}'', w) \models \text{nom}(q) \wedge \text{EF}(q \wedge \neg p)$. By definition of $\text{nom}(p)$, there is a world \widehat{w} such that $\mathcal{W}'' = \{\widehat{w}\}$. Together with $\text{EF}(q \wedge \neg p)$, this implies that \widehat{w} belongs to the path ρ . Let us say that $\widehat{w} = w_i$, i.e. \widehat{w} is the i -th world in ρ . Let us consider its successor w_{i+1} in the path. Clearly, $(\mathcal{K}'', w_{i+1}) \models \neg q \wedge \neg p$ and thus $(\mathcal{K}'', w_i) \models \text{EF}(\neg q \wedge \neg p)$, which in turn leads to $(\mathcal{K}'', w) \models \text{EF}(q \wedge \text{EF}(\neg q \wedge \neg p))$. From the semantics of the propositional quantification, $(\mathcal{K}', w) \models \forall q (\text{nom}(q) \wedge \text{EF}(q \wedge \neg p) \Rightarrow \text{EF}(q \wedge \text{EF}(\neg q \wedge \neg p)))$ and $(\mathcal{K}, w) \models \chi_{\text{EG}}(\varphi)$.

(\Leftarrow) : We take the contrapositive and show that if $(\mathcal{K}, w) \not\models \text{EG } \varphi$ then $(\mathcal{K}, w) \models \neg \chi_{\text{EG}}(\varphi)$. Notice that $\neg \chi_{\text{EG}}(\varphi)$ can be rewritten as

$$\forall p (\neg p \wedge \text{AG}(\neg \varphi \Rightarrow p) \wedge \text{AG}(p \Rightarrow \text{AG } p) \Rightarrow \exists q (\text{nom}(q) \wedge \text{EF}(q \wedge \neg p) \wedge \text{AG}(q \Rightarrow \text{AG}(q \vee p))))$$

Suppose that $(\mathcal{K}, w) \not\models \text{EG } \varphi$, and thus every path $(w_0, w_1, \dots) \in \Pi_R(w)$ must contain a world w_i ($i \geq 0$) s.t. $(\mathcal{K}, w_i) \models \neg \varphi$. Equivalently, one of the following holds:

A. $(\mathcal{K}, w) \models \neg \varphi$, or

B. for every path $(w_0, w_1, \dots) \in \Pi_R(w)$ there is $j \geq 0$ such that for every $i \leq j$ $(\mathcal{K}, w_i) \models \varphi$ whereas every $w' \in R(w_j)$ is such that $(\mathcal{K}, w') \models \neg \varphi$.

Let us now consider a Kripke tree $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W}'])$ obtained from \mathcal{K} by updating the evaluation of p with respect to a set \mathcal{W}' such that (\mathcal{K}', w) satisfies $\neg p \wedge \text{AG}(\neg \varphi \Rightarrow p) \wedge \text{AG}(p \Rightarrow \text{AG } p)$.

From the first two conjuncts we conclude that (A) does not hold, and so $(\mathcal{K}, w) \models \varphi$ and (B) hold. As Kripke trees are left-total, $\Pi_R(w) \neq \emptyset$ and so (B) and $\text{AG}(\neg\varphi \Rightarrow p) \wedge \text{AG}(p \Rightarrow \text{AG } p)$ imply

- C. there is a path $(w_0, w_1, \dots) \in \Pi_R(w)$ and a $j \geq 0$ such that $(\mathcal{K}, w_j) \models \varphi \wedge \neg p$ and for every $w' \in R^*(w_j) \setminus \{w_j\}$ it holds that $(\mathcal{K}, w') \models p$.

Let us consider the world w_j in (C) and define $\mathcal{K}'' = (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W}'][q \leftarrow \{w_j\}])$ to be the Kripke tree obtained by updating \mathcal{K}' so that q evaluates to $\{w_j\}$. By definition of \mathcal{K}'' and (C), $(\mathcal{K}'', w) \models \text{nom}(q) \wedge \text{EF}(q \wedge \neg p) \wedge \text{AG}(q \Rightarrow \text{AG}(q \vee p))$. By semantics of $\exists p$ we have

$$(\mathcal{K}', w) \models \neg p \wedge \text{AG}(\neg\varphi \Rightarrow p) \wedge \text{monotone}(p) \Rightarrow \exists q(\text{nom}(q) \wedge \text{EF}(q \wedge \neg p) \wedge \text{AG}(q \Rightarrow \text{AG}(q \vee p)))$$

Again from the semantics of $\exists p$, together with the definition of \mathcal{K}' , we get $(\mathcal{K}, w) \models \neg \chi_{\text{EG}}(\varphi)$. \square

4.4.3 From ALT to Modal Separation Logic.

In Section 2.3.2 we introduced the modal separation logics **MSL** and **MLH**. At their core, both logics can be seen as modal logics extended with separating connectives, hence mixing separation logic with temporal aspects as in quantified CTL. As we already showed how **ALT** is captured by these two latter logics, it is natural to ask ourselves if the same holds for **MLH** and **MSL**. In this section, we show that this is indeed the case and, as for the previous two sections, **ALT** allows us to refine the analysis on these logics. We refer the reader to Section 2.3.2 for the definitions of these logics, and consider here their fragments without the separating implication. Recall that **MLH** is a variant of **MSL** that does not feature propositional symbols, and both logics are interpreted on Kripke-style finite functions: a class of Kripke structures where the accessibility relation, instead of being left-total, is finite and weakly functional. The following diagram introduces a language (where $p \in \text{AP}$) having the operators from **MSL** and **MLH**, and summarise known and new results on the satisfiability problem of these logics:

$$\begin{array}{c} \text{MSL: TOWER-complete from [54].} & \text{MLH: TOWER-complete from [52].} \\ \overbrace{\varphi := p \mid \langle \neq \rangle \varphi \mid \top \mid \varphi \wedge \varphi \mid \neg \varphi \mid \Diamond \varphi \mid \varphi * \psi \mid \langle U \rangle \varphi \mid \Diamond^{-1} \varphi}^{\text{MSL/MLH: TOWER-hard by reduction from the satisfiability problem of ALT, proved here.}} \end{array}$$

As defined in Section 2.3.2, \Diamond is the standard alethic modality from modal logic, \Diamond^{-1} is its converse, and $\langle \neq \rangle$ is the elsewhere modality that generalises the somewhere modality $\langle U \rangle$ as $\langle U \rangle \varphi = \varphi \vee \langle \neq \rangle \varphi$. Given a pointed finite function (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a Kripke-style finite function and $w \in \mathcal{W}$, we recall the satisfaction relation \models for the fragment **MSL**/**MLH** of **MSL** and **MLH** we show to be TOWER-complete (omitting \top and Boolean connectives):

$$\begin{aligned} (\mathcal{K}, w) \models \Diamond \varphi &\stackrel{\text{by def}}{\iff} \text{there is } w' \in R(w) \text{ such that } (\mathcal{K}, w) \models \varphi, \\ (\mathcal{K}, w) \models \langle U \rangle \varphi &\stackrel{\text{by def}}{\iff} \text{there is } w' \in \mathcal{W} \text{ such that } (\mathcal{K}, w') \models \varphi, \\ (\mathcal{K}, w) \models \varphi * \psi &\stackrel{\text{by def}}{\iff} (\mathcal{K}_1, w) \models \varphi \text{ and } (\mathcal{K}_2, w) \models \psi \text{ for some } \mathcal{K}_1, \mathcal{K}_2 \text{ s.t. } \mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}. \end{aligned}$$

By looking at the diagram above, compared to the work in [54], **ALT** allows us to show that propositional symbols and the elsewhere modality can be removed from **MSL** without changing the complexity status of its satisfiability problem (notice that this logic features the somewhere modality). Similarly, **ALT** allows us to refine the analysis on the complexity of **MLH** done in [52] by showing that the \Diamond^{-1} modality is not needed in order to achieve non-elementary complexities.

$$\begin{aligned}
\tau(\text{Hit}) &\stackrel{\text{def}}{=} \blacklozenge_{\text{ML}}^*(\Diamond\top \wedge [\mathcal{U}](\Diamond\top \Rightarrow \Diamond\Diamond\top)), & \tau(\blacklozenge\varphi) &\stackrel{\text{def}}{=} \blacklozenge_{\text{ML}}(\tau(\varphi) \wedge \langle\mathcal{U}\rangle \text{selfloop}), \\
\tau(\text{Miss}) &\stackrel{\text{def}}{=} \Diamond\top \wedge \neg\tau(\text{Hit}), & \tau(\blacklozenge^*\varphi) &\stackrel{\text{def}}{=} \blacklozenge_{\text{ML}}^*(\tau(\varphi) \wedge \langle\mathcal{U}\rangle \text{selfloop}). \\
\tau(\langle\mathcal{U}\rangle\varphi) &\stackrel{\text{def}}{=} \langle\mathcal{U}\rangle(\neg\text{selfloop} \wedge \tau(\varphi)),
\end{aligned}$$

Figure 4.13: Translation from ALT to MSL/MLH .

From ALT to MSL/MLH . As \mathcal{W} and \mathcal{N} are both countably infinite sets, without loss of generality we assume $\mathcal{W} = \mathcal{N}$. Let (\mathcal{F}, t, n) be a pointed forest and let (\mathcal{K}, w) be a pointed finite function where $\mathcal{K} = (\mathcal{N}, R, \mathcal{V})$. As done in Section 4.1.2 in order to relate ALT to $\text{SL}([\exists]_2, *)$, we start by introducing the sabotage and repeated sabotage modalities in MSL/MLH . We define the formula $\text{size}=1 \stackrel{\text{def}}{=} \langle\mathcal{U}\rangle\Diamond\top \wedge \neg(\langle\mathcal{U}\rangle\Diamond\top * \langle\mathcal{U}\rangle\Diamond\top)$, that is satisfied whenever $\text{card}(R)=1$. Then, the modalities \blacklozenge and \blacklozenge^* are defined in MSL/MLH as $\blacklozenge_{\text{ML}}\varphi \stackrel{\text{def}}{=} (\text{size}=1) * \varphi$ and $\blacklozenge_{\text{ML}}^*\varphi \stackrel{\text{def}}{=} \top * \varphi$.

For the reduction, we use w to encode the current node n . Encoding t is not so immediate, as MSL/MLH does not have propositional symbols. A possible solution is to encode it as a self-loop, so that the formula Hit is translated to a query stating that w reaches the self-loop. So, we introduce the formula selfloop that is satisfied by (\mathcal{K}, w') if $(w', w') \in R$:

$$\text{selfloop} \stackrel{\text{def}}{=} \blacklozenge_{\text{ML}}^*(\Diamond\Diamond\top \wedge \neg\blacklozenge_{\text{ML}}\blacklozenge_{\text{ML}}\top).$$

Informally, this formula characterises a self-loop by stating that it is possible to find a structure $\mathcal{K}' \subseteq \mathcal{K}$ that has an accessibility relation of cardinality one and satisfies $\Diamond\Diamond\top$. Suppose for a moment that we are able to use this formula to characterise the class of every Kripke-style finite function having exactly one cycle, and where this cycle is a self-loop, say on a world w_t . On these finite functions, we use w_t to encode the target node t of a finite forest (\mathcal{F}, t, n) while being careful that the \blacklozenge and \blacklozenge^* operators of ALT are translated in such a way that the self-loop on w_t is preserved. Because of the specific treatment of w_t , it is convenient to assume that the current evaluation node n is encoded by a world different from w_t , which reflects on the translation of $\langle\mathcal{U}\rangle$. As it was the case for the translation to QCTL t , the admissibility of this assumption follows by Lemma 4.18. We formalise the encoding of a pointed forest as a pointed finite function.

Definition 4.42 (MSL/MLH - Pointed forest encoding). Let (\mathcal{F}, t, n) be a pointed forest such that $t \notin \text{dom}(\mathcal{F})$ and $n \neq t$. The pointed finite function $((\mathcal{N}, R, \mathcal{V}), n)$ is an *encoding* of (\mathcal{F}, t, n) if and only if for every $n', n'' \in \mathcal{N}$ we have $(n', n'') \in R \Leftrightarrow (\mathcal{F}(n') = n'' \text{ or } n' = n'' = t)$.

Notice how R is essentially defined from \mathcal{F} by adding the self-loop (t, t) . The translation $\tau(\varphi)$ in MLH of a formula φ in ALT is homomorphic for \top and Boolean connectives, and otherwise it is defined as in Figure 4.13. We highlight two points of this translation. First, $\tau(\text{Hit})$ essentially asks to find a submodel where every path reaches the self-loop and the current evaluation node is in one of these paths. Second, notice how the translation of \blacklozenge and \blacklozenge^* checks that the model is updated so that the self-loop is not lost, as required by our encoding. The following lemma (proved in Appendix B by structural induction on φ) shows the correctness of our translation.

Lemma 4.43. Let (\mathcal{F}, t, n) be a pointed model s.t. $n \neq t$ and $t \notin \text{dom}(\mathcal{F})$. Let (\mathcal{K}, n) be an encoding of (\mathcal{F}, t, n) . Given a formula φ in ALT, $(\mathcal{F}, t, n) \models \varphi$ iff $(\mathcal{K}, n) \models \tau(\varphi)$.

To conclude the reduction we show that we can characterise the class of models encoding pointed forests, i.e. the pointed finite functions with exactly one cycle, which is a self-loop that

does not involve the current evaluation node. We first define a formula that checks whether a Kripke-style finite function has at least one cycle:

$$\text{hascycles} \stackrel{\text{def}}{=} \lozenge_{\text{ML}}^* (\langle U \rangle \lozenge \top \wedge [U](\lozenge \top \Rightarrow \lozenge \lozenge \top)).$$

Informally, this formula characterises the presence of a cycle by stating that there is a structure $(W, R', V) \subseteq \mathcal{K}$ such that R' is not empty and every world w' that has a successor, i.e. $R'(w') \neq \emptyset$, also reaches a world in two steps $R'^2(w) \neq \emptyset$. The cyclicity of \mathcal{K} then follows from the fact that the accessibility relation is finite. Afterwards, the desired property of having exactly one cycle that is a self-loop can be defined by stating that there is a self-loop which, whenever removed, leads to an acyclic Kripke-style finite function. The following formula does the job:

$$\exists \text{selfloop} \stackrel{\text{def}}{=} \langle U \rangle (\text{selfloop} \wedge \neg \lozenge_{\text{ML}}(\Box \perp \wedge \text{hascycles})).$$

Lemma 4.44. Every formula φ in ALT is equisatisfiable with $\tau(\varphi) \wedge \exists \text{selfloop} \wedge \neg \text{selfloop}$.

In the proof of Lemma 4.44, both (I) and (II) of Lemma 4.18 are used in order to restrict ourselves to pointed forest (\mathcal{F}, t, n) s.t. $n \neq t$ and $t \notin \text{dom}(\mathcal{F})$. Afterwards, we apply Lemma 4.43.

Theorem 4.45. The fragment of MLH and MSL with the * (alternatively, \lozenge_{ML} and \lozenge_{ML}^*), \top , Boolean connectives, \lozenge and $\langle U \rangle$ modalities, and has a TOWER-complete satisfiability problem.

Deciding Robustness Properties in PSpace

Contents

5.1	Taming the Robustness Properties	119
5.1.1	The separation logic $\mathbf{SL}([\exists]_1, *, [-*, \hookrightarrow^{\dagger}]_{\mathcal{W}}^{\mathcal{S}})$	119
5.1.2	Reasoning in $\mathbf{SL}([\exists]_1, *, [-*, \hookrightarrow^{\dagger}]_{\mathcal{W}}^{\mathcal{S}})$	120
5.2	Towards Small Models: The Core Formulae Technique	122
5.2.1	The Core Formulae Technique.	122
5.2.2	Game Hopping.	124
5.3	A Family of Core Formulae Capturing the Fragment w	127
5.3.1	Step I: partitioning the heap.	127
5.3.2	Step II: the core formulae for w	129
5.3.3	Step III: indistinguishability relation, hops and $*$ -simulation.	131
5.3.4	Step IV: \exists -simulation.	141
5.4	Recap: How to Apply the Core Formulae Technique	144
5.5	A Family of Core Formulae Capturing the Fragment s	146
5.5.1	Step I: partitioning the heap.	146
5.5.2	Step II: the core formulae for s	152
5.5.3	Step III: $*$ -simulation.	170
5.5.4	Step IV: \exists -simulation.	216
5.6	Connecting the Two Families of Core Formulae	221
5.6.1	Small-heap property and PSPACE-completeness.	224
5.6.2	One last step: the $-*$ -simulation property.	230

In this chapter

Thanks to the knowledge gathered in Chapters 3 and 4, in this chapter we devise a separation logic, denoted by $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$, that can specify whether a formula is robust for the properties of acyclicity and garbage-freedom, while admitting a PSPACE-complete satisfiability (and entailment/validity) problem. This result is quite interesting, as the logic is a non-trivial syntactical extension of $\text{SL}([\exists]_1, *, -*)$ and $\text{SL}(*, \hookrightarrow^+)$, which are both PSPACE-complete [55, 56]. To establish the PSPACE upper bound, we rely on the *core formulae* technique already used to prove the decidability of $\text{SL}(*, \hookrightarrow^+)$ [56]. Due to the expressive power of our logic, applying this technique by adapting the presentation in [56] would lead to a monolithic proof of hard to check technical steps. We partially ease this issue by revisiting the core formulae technique, in order to improve its modularity. Despite our efforts, the proof still reveals to be technically involved.

Here is a roadmap of the chapter.

Section 5.1. We introduce the separation logic $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$. This logic features both the separating conjunction and implication, the reachability predicate \hookrightarrow^+ and a single quantified variable name u . To prevent the logic from being TOWER-hard following the results in Chapters 3 and 4, two syntactical conditions are imposed on the reachability predicates \hookrightarrow^+ :

- $x \hookrightarrow^+ y$ does not appear on the right side of the first $-*$ ancestor,
- if the variable x appearing in $x \hookrightarrow^+ y$ is syntactically equal to u , then so is y .

The first condition splits the grammar of the logic into two fragments shown below:

$$\begin{aligned} w &:= \top \mid \text{emp} \mid t = t' \mid t \hookrightarrow t' \mid w \wedge w \mid \neg w \mid w * w \mid s \multimap w \mid \exists u w \\ s &:= w \mid x \hookrightarrow^+ t \mid u \hookrightarrow^+ u \mid s \wedge s \mid \neg s \mid s * s \mid \exists u s \end{aligned}$$

We call *weak* the fragment of the logic generated from the non-terminal symbol w and *strong* the fragment generated from s . Roughly speaking, the strong fragment extends the weak fragment with reachability predicates. Besides, notice that the magic wand is restricted to the form $s \multimap w$. The section ends by showing that the robustness properties of acyclicity and garbage freedom can be characterised as entailment queries of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$.

Section 5.2. We discuss the core formulae technique used in order to prove that the satisfiability problem of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$ is decidable in PSPACE. The goal is to prove that the logic enjoys a polynomial small-heap property, defined as follows.

Definition 5.3 (Small-heap property). A separation logic \mathcal{L} is said to have the polynomial *small-heap property* if there is a polynomial $\mathcal{Q} : \mathbb{N} \rightarrow \mathbb{N}$ such that, for every formula φ in \mathcal{L} ,

φ is satisfiable if and only if φ is satisfied by a memory state (s, h) where $\text{card}(h) \leq \mathcal{Q}(|\varphi|)$.

In order to show this result, the core formulae technique requires the definition of a set of formulae (called core formulae) interpreted on memory states and capturing the expressive power of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$. In particular, every formula of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$ shall be equivalent to a Boolean combination of core formulae. The polynomial small-heap property is then easily derived in terms of such Boolean combinations.

The proof that each formula of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$ can be translated into a Boolean combination of core formulae is technical. To partially ease this issue, we profit from connections

between the core formulae technique and the notion of Ehrenfeucht-Fraïssé games for separation logic. We introduce the notion of game hopping, which helps with the modularity of the proof.

Section 5.3. We apply the core formulae technique to the fragment \mathcal{W} of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}}$), excluding for the moment the magic wand $s \multimap \mathcal{W}$. We introduce a suitable set of core formulae, and prove that their Boolean combinations capture the atomic formulae of \mathcal{W} . Afterwards, we show two fundamental properties of the core formulae: the $*$ -simulation property and \exists -simulation property. These two properties are key to show that each weak formula is equivalent to a Boolean combination of core formulae. Roughly speaking, the $*$ -simulation property (resp. \exists -simulation property) implies that formulae of the form $\varphi * \psi$ (resp. $\exists x \varphi$), where φ and ψ are Boolean combinations of core formulae, are themselves equivalent to Boolean combinations of core formulae. Together with the fact that every atomic formula of the weak fragment is equivalent to a Boolean combination of core formulae, these properties allow to perform a bottom-up translation of all $*$ -free formulae of \mathcal{W} into equivalent Boolean combinations of core formulae.

Section 5.4. In this short section, we recapitulate the content of Section 5.2 and Section 5.3, highlighting the steps needed in order to apply the core formulae technique to the weak fragment. This section should be seen as a guide to Section 5.5, where the exact same steps are applied in order to study the core formulae for the strong fragment.

Section 5.5. We introduce the core formulae for \mathcal{S} , show that they capture every atomic formula of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}}$), and prove that they satisfy the $*$ -simulation and \exists -simulation properties. Despite the long and technical arguments we need in order to prove the two simulation properties, the structure of their proof follows exactly the one employed for the weak fragment, analysed in Section 5.4.

Section 5.6. In order to translate every formula of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}}$) into a Boolean combination of core formulae, we miss a $*$ -simulation property that allows us to translate a formula $\varphi * \psi$, where φ (resp. ψ) is a Boolean combination of core formulae of the strong (resp. \mathcal{W}) fragment, into an equivalent Boolean combination of core formulae. Under the assumption that the $*$ -simulation property holds, the section starts by showing the following result.

Theorem 5.46. Every formula φ in $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}}$) is logically equivalent to a Boolean combination of core formulae from $\text{Core}[\mathcal{S}](\text{fv}(\varphi) \setminus \{u\}, |\varphi|_m)$.

Here, $|\varphi|_m$ is roughly the size of φ , and $\text{Core}[\mathcal{S}](X, \alpha)$ is the set of core formulae for the fragment \mathcal{S} , which is indexed by a set of variables X and an integer threshold $\alpha \geq 1$. From their definition, it is easy to see that the core formulae enjoy a polynomial small-heap property, which carries out to $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}}$) directly from Theorem 5.46 (see Corollary 5.48). This allows us to design an algorithm for the satisfiability problem of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}}$) that runs in PSPACE. As PSPACE-hardness is inherited from $\text{SL}(*, *)$, we prove the main result of the chapter.

Theorem 5.50. The satisfiability problem of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}}$) is PSPACE-complete.

Of course, this result holds under the assumption that the core formulae enjoy the $*$ -simulation property, which we show at the end of the section. As in the case of the other simulation properties, the proof reveals to be technically involved.

5.1 TAMING THE ROBUSTNESS PROPERTIES

We return to our original research agenda of finding a separation logic that can express whether a formula is robust for the properties of acyclicity and garbage-freedom (see the introduction of Part I). More precisely, we aim at defining a separation logic whose satisfiability, validity and entailment problems can be solved in PSPACE, and where the decision problems related to acyclicity and garbage-freedom can be expressed as queries of the entailment problem.

First of all, let us recall the definitions of acyclic and garbage-free memory state.

Definition 5.1 (Acyclicity and garbage-freedom). Let (s, h) be a memory state. Let $\mathbf{X} \subseteq_{\text{fin}} \text{VAR}$.

- (acyclicity) (s, h) is *acyclic* if for every $\ell \in \text{LOC}$ and for every $\delta \geq 1$, $h^\delta(\ell) \neq \ell$,
- (\mathbf{X} -garbage-freedom) (s, h) is \mathbf{X} -garbage-free whenever for every $\ell \in \text{dom}(h)$ there is $\delta \in \mathbb{N}$ and $\mathbf{x} \in \mathbf{X}$ such that $h^\delta(s(\mathbf{x})) = \ell$.

The definition of the decision problems related to these two properties is given in Figure 5.1, with respect to a formula φ written in an arbitrary (separation) logic \mathcal{L} interpreted on memory states. Both these decision problems are oriented towards program verification. The acyclicity property reveals to be quite useful to guarantee that a loop traversing any region of the memory by dereferentiation is bound to terminate [76], whereas garbage-freedom can be helpful to prove that a program does not leak memory. In this chapter, we analyse the decidability of these problems in a fragment of $\text{SL}(\exists, *, -*)$.

5.1.1 The separation logic $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+_\mathcal{W}])$.

In order to tame the decision problems for acyclicity and garbage-freedom into queries of entailment, we consider separation logics featuring reachability predicates, separating implication and one quantified variable. The results in Chapters 3 and 4 heavily limit the set of possible separation logics that enjoy these three features and admit an elementary satisfiability problem. In this chapter, we consider the fragment of $\text{SL}(\exists, *, -*)$, denoted by $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+_\mathcal{W}])$, having just one quantified variable name $\mathbf{u} \in \text{VAR}$ (“ \mathbf{u} ” stands for *unique*) and with formulae from the non-terminal s of the following grammar, where $\mathbf{t}, \mathbf{t}' \in \text{VAR}$ and $\mathbf{x} \in \text{VAR} \setminus \{\mathbf{u}\}$,

$w :=$	\top (<i>true</i>)	$s :=$	w (<i>weak fragment</i>)
	\mathbf{emp} (<i>empty</i>)		$\mathbf{x} \hookrightarrow^+ \mathbf{t}$ (<i>reach-plus</i>)
	$\mathbf{t} = \mathbf{t}'$ (<i>equality</i>)		$\mathbf{u} \hookrightarrow^+ \mathbf{u}$ (<i>unique reach-plus</i>)
	$\mathbf{t} \hookrightarrow \mathbf{t}'$ (<i>points-to</i>)		$s \wedge s$ (<i>strong conjunction</i>)
	$w \wedge w$ (<i>weak conjunction</i>)		$\neg s$ (<i>strong negation</i>)
	$\neg w$ (<i>weak negation</i>)		$s * s$ (<i>strong star</i>)
	$w * w$ (<i>weak star</i>)		$\exists \mathbf{u} s$ (<i>strong existential</i>)
	$s \multimap w$ (<i>strong-to-weak magic wand</i>)		
	$\exists \mathbf{u} w$ (<i>weak existential</i>)		

As done in the grammar, we generally write $\mathbf{x}, \mathbf{y}, \mathbf{z}, \dots$ for variables that are syntactically different from \mathbf{u} . Instead, we use $\mathbf{t}, \mathbf{t}', \dots$ for arbitrary variables in VAR . Analogously, we write $\mathbf{X}, \mathbf{Y}, \dots$ for (usually finite) sets of variables that do not contain \mathbf{u} , and $\mathbf{T}, \mathbf{T}', \dots$ for arbitrary sets of variables. The grammar of the logic is divided into two fragments, called *weak* and

acyclicity: *Input:* A formula φ in \mathcal{L} .

Question: Is every memory state satisfying φ acyclic?

garbage-freedom: *Input:* A formula φ in \mathcal{L} .

Question: Is every memory state satisfying φ $\text{fv}(\varphi)$ -garbage-free?

Figure 5.1: The decision problems for the properties of acyclicity and garbage freedom.

strong, which roughly regulate whether the separating implication and the reachability predicates can be used. The weak fragment, whose formulae are generated from the non-terminal w , does not directly features reachability predicates, but contains the separating implication $s \multimap w$, where s belongs to the strong fragment. The strong fragment, whose formulae are generated from the non-terminal s , extends the weak fragment with the reach-plus predicate defined in Section 2.1.1, but still, it can only use the separating implication $s \multimap w$, so that the right-hand side of the formula belongs to the weak fragment. Informally, a formula of the separation logic $\text{SL}([\exists]_1, *, \multimap, \hookrightarrow^+)$, featuring one quantified variable and unconstrained reach-plus predicates and magic wand, is in $\text{SL}([\exists]_1, *, [\multimap, \hookrightarrow^+]_{\mathcal{W}}^S)$ if every occurrence of the reach-plus $t \hookrightarrow^+ t'$ is constrained so that it satisfies the following two conditions:

1. it is not on the right side of its first \multimap ancestor (seeing the formula as a tree),
2. if t is syntactically equal to u , then so is t' .

For instance, given a formula φ generated from s and a formula ψ generated from w , the formula $u \hookrightarrow^+ x \multimap (\varphi \multimap \psi)$ only satisfies the first condition, the formula $\varphi \multimap (x \hookrightarrow^+ u \multimap \psi)$ satisfies both conditions, whereas the formula $\varphi \multimap (\psi \multimap u \hookrightarrow^+ u)$ only satisfies the second condition. Thanks to these two conditions, the computational complexity of $\text{SL}([\exists]_1, *, [\multimap, \hookrightarrow^+]_{\mathcal{W}}^S)$ cannot be traced back to the results in Chapters 3 and 4. More precisely, by disallowing the reach-plus predicate on positions corresponding to the right-hand side of a separating implication, the condition (1) forbid the definition of the bounded reachability predicate \hookrightarrow^3 in these positions. This breaks the undecidability result given in Chapter 3 for $\text{SL}(*, \multimap, \hookrightarrow^2, \hookrightarrow^3)$, which requires the bounded reachability predicates $x \hookrightarrow^2 y$ and $x \hookrightarrow^3 y$ to appear in both sides of the magic wand. The predicate $x \hookrightarrow^2 y$ can be still defined with the formula $\exists u (x \hookrightarrow u \wedge u \hookrightarrow y)$, which belongs to the weak fragment. One can ask why the restriction should be done on the right-side of the separating implication instead of the left-side. Unfortunately, taking $w \multimap s$ over $s \multimap w$ leads to TOWER-hardness directly from Section 4.4.1. Indeed, Corollary 4.33 shows us that taking $\text{size}=1 \multimap s$ is already enough to capture ALT. Similarly, the condition (2) prevents us from using the reachability query $u \hookrightarrow^+ x$, which would again lead to the internalisation of ALT, as shown in Section 4.1.2. Notably, $\text{SL}([\exists]_1, *, [\multimap, \hookrightarrow^+]_{\mathcal{W}}^S)$ still allows us to write $x \hookrightarrow^+ u$, which is essential in order to express the property of garbage-freedom.

5.1.2 Reasoning in $\text{SL}([\exists]_1, *, [\multimap, \hookrightarrow^+]_{\mathcal{W}}^S)$.

Despite its restrictions, $\text{SL}([\exists]_1, *, [\multimap, \hookrightarrow^+]_{\mathcal{W}}^S)$ is still a very expressive fragment of $\text{SL}(\exists, *, \multimap)$. Its syntax extends the one-variable fragment $\text{SL}([\exists]_1, *, *)$ from [55] and the logic $\text{SL}(*, \hookrightarrow^+)$ from [56], both admitting a PSPACE-complete satisfiability problem. Moreover, our logic can express all the auxiliary formulae introduced in Section 2.1.1, which we recall in Figure 5.2.

Formula:	Semantics w.r.t. (s, h)	Definition:	\mathcal{W}/\mathcal{S} :
$t \hookrightarrow _$	$s(t) \in \text{dom}(h)$	$t \hookrightarrow t \ast \varphi$	\mathcal{W}
$_ \hookrightarrow x$	$s(x) \in \text{ran}(h)$	$\exists u u \hookrightarrow x$	\mathcal{W}
$\text{size} \geq 0$	$\text{card}(h) \geq 0$	\top	\mathcal{W}
$\text{size} \geq 1$	$\text{card}(h) \geq 1$	$\neg \text{emp}$	\mathcal{W}
$\text{size} \geq \beta + 1$	$\text{card}(h) \geq \beta + 1$	$\neg \text{emp} * \text{size} \geq \beta$	\mathcal{W}
$x \hookrightarrow^* t$	$(s(x), s(t)) \in h^*$	$x = t \vee x \hookrightarrow^+ t$	\mathcal{S}
$\text{strict}(\varphi)$	$(s, h) \models \varphi$ and $\forall h' \subsetneq h, (s, h') \not\models \varphi$	$\varphi \wedge \neg(\neg \text{emp} * \varphi)$	\mathcal{W}
$t \mapsto t'$	$h = \{s(t) \mapsto s(t')\}$	$\text{strict}(t \hookrightarrow t')$	\mathcal{W}
$\text{ls}(x, t)$	$h^\delta(s(x)) = s(t)$ iff $\delta = \text{card}(h)$	$\text{strict}(x \hookrightarrow^* t)$	\mathcal{S}

Figure 5.2: Formulae from Section 2.1.1, in $\text{SL}([\exists]_1, *, [\ast, \hookrightarrow^{\mathcal{S}}]_{\mathcal{W}})$.

Example 5.2. The combination of one quantifier variable and reachability predicates, although not as powerful as in ALT, can express interesting shapes of a memory state (s, h) . For instance, we can state that $s(x)$ does not belong to a cycle but reaches a location that belongs to one. This pattern can be described with the intuitive formula $\exists u (x \hookrightarrow^+ u * u \hookrightarrow^+ u)$. The unique quantified variable is also helpful to state whether paths starting from two locations corresponding to program variables meet. For instance, consider $\exists u ((x \hookrightarrow^+ u * y \hookrightarrow^+ u) \wedge \neg u \hookrightarrow _)$. The memory state (s, h) satisfies this formula if it witnesses two distinct paths, one going from $s(x)$ to ℓ and one going from $s(y)$ to ℓ , where ℓ is a location not in $\text{dom}(h)$. Roughly speaking, we can even combine the two formulae just introduced in order to state that both $s(x)$ and $s(y)$ reach two locations ℓ and ℓ' belonging to the same cycle, but $\ell \neq \ell'$. The following formula does the job:

$$\exists u ((x \hookrightarrow^+ u * (y \hookrightarrow^+ u \wedge u \hookrightarrow^+ u \wedge \neg y \hookrightarrow^+ y)) \wedge \neg(x \hookrightarrow^+ u * y \hookrightarrow^+ u * u \hookrightarrow^+ u)).$$

From the expressivity results in [55] and [56], we know that the three formulae considered in this example cannot be expressed in neither $\text{SL}([\exists]_1, *, \ast)$ nor $\text{SL}(*, \hookrightarrow^+)$ (or any separation logic with a PSPACE-complete satisfiability problem that we know of).

Acyclicity and garbage-freedom, revised. Fundamentally, the decision problems for the properties of acyclicity and garbage-freedom can be expressed very easily in $\text{SL}([\exists]_1, *, [\ast, \hookrightarrow^{\mathcal{S}}]_{\mathcal{W}})$. Indeed, the set of acyclic memory states can be characterised with the formula $\forall u \neg u \hookrightarrow^+ u$. Given a set of variables $X \subseteq_{\text{fin}} \text{VAR}$, a memory state is X -garbage-free whenever it satisfies the formula $\forall u (u \hookrightarrow _ \Rightarrow \bigvee_{x \in X} x \hookrightarrow^* u)$. Both these formulae of $\text{SL}([\exists]_1, *, [\ast, \hookrightarrow^{\mathcal{S}}]_{\mathcal{W}})$ are a natural translation of the two properties expressed in Definition 5.1. This allows us to rephrase the decision problems in Figure 5.1 using two queries of entailment, as shown below:

acyclicity: *Input:* A formula φ in $\text{SL}([\exists]_1, *, [\ast, \hookrightarrow^{\mathcal{S}}]_{\mathcal{W}})$.
Question: Does $\varphi \models \forall u \neg u \hookrightarrow^+ u$ hold?

garbage-freedom: *Input:* A formula φ in $\text{SL}([\exists]_1, *, [\ast, \hookrightarrow^{\mathcal{S}}]_{\mathcal{W}})$ such that $u \notin \text{fv}(\varphi)$.
Question: Does $\varphi \models \forall u (u \hookrightarrow _ \Rightarrow \bigvee_{x \in \text{fv}(\varphi)} x \hookrightarrow^* u)$ hold?

Notice that the auxiliary condition “ $u \notin fv(\varphi)$ ” in the decision problem for garbage-freedom can be enforced without loss of generality, as we can always rename the free occurrences of u in φ with a different variable name.

The rest of the chapter is devoted to the satisfiability problem of $SL([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$, which we show to be PSPACE-complete. This implies that the entailment and validity problem of this logic are also PSPACE-complete, and that the decision problems related to acyclicity and garbage-freedom can be decided in PSPACE. Notably, $SL([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$ extends various separation logics, some of which are applied to program verification. Besides $SL([\exists]_1, *, -*)$ and $SL(*, \hookrightarrow^+)$, the logic extends the symbolic-heap fragment $SH(1s)$ (Section 2.3.1), and the logic obtained by closing $SH(1s)$ under Boolean connectives. The decidability of acyclicity and garbage-freedom provided for $SL([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$ can be transferred to all these logics.

5.2 TOWARDS SMALL MODELS: THE CORE FORMULAE TECHNIQUE

In this section we discuss the technique used to prove that $SL([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$ admit a satisfiability problem that can be solved in PSPACE. The general idea is to show a bound on the size of the smallest memory state that satisfies a formula in $SL([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$. From the notion of X-heap isomorphism, we know that the store does not pose a particular challenge (see Proposition 2.10), so we primarily aim at showing that the logic enjoys a polynomial *small-heap property*.

Definition 5.3 (Small-heap property). A separation logic \mathcal{L} is said to have the polynomial *small-heap property* if there is a polynomial $\mathcal{Q} : \mathbb{N} \rightarrow \mathbb{N}$ such that, for every formula φ in \mathcal{L} ,

φ is satisfiable if and only if φ is satisfied by a memory state (s, h) where $\text{card}(h) \leq \mathcal{Q}(|\varphi|)$.

We recall that $|\varphi|$ is the size of φ , i.e. the number of symbols needed to encode it (as a tree).

One peculiarity of $SL([\exists]_1, *, [-*, \hookrightarrow^+]_{\mathcal{W}}^S)$ is that the small-heap property must also take into account all the heaps that must be considered for the satisfaction of the separating implication. This problem, that is common to all the separation logics featuring the magic wand, was firstly solved by C. Calcagno, H. Yang and P. W. O’Hearn [33], and successively by E. Lozes [104], for the quantifier-free separation logic $SL(*, -*)$. Both papers essentially rely on suitable model abstractions in order to polynomially bound the size of the heaps that must be considered when dealing with the magic wand. The bound is then extended to arbitrary formulae, leading to a polynomial small-heap property for $SL(*, -*)$. The main difference between [33] and [104] is that the abstraction given by E. Lozes is defined entirely from formulae of the logic, allowing to characterise the expressiveness of $SL(*, -*)$ has a by-product.

After [104], the idea of relying on the logic itself to describe a model abstraction led to a quite successful methodology, which we call here the *core formulae technique*. This technique has been used to show the decidability of several separation logics, including:

$SL([\exists]_1, *, -*)$ [55], $SL(*, \hookrightarrow^+)$ [56], $\exists^* \forall^* SL(\exists, *, -*)$ [62], and $pnf\text{-}SL(\exists, *, -*)$ [62].

5.2.1 The Core Formulae Technique.

Let us introduce the core formula technique with a running example. To keep things as simple as possible, we consider the separation logic $SL(*)$, having formulae from the grammar:

$$\varphi := \top \mid \text{emp} \mid x = y \mid x \hookrightarrow y \mid \varphi \wedge \varphi \mid \neg \varphi \mid \varphi * \varphi.$$

Our aim is twofold. On one side, we want to appreciate the key components of the core formulae technique, in order to later apply it to $\text{SL}([\exists]_1, *, [\neg*, \hookrightarrow]^S_W)$. On the other side, we want to understand and address some of the limitations of this methodology.

A finite set of formulae to capture the logic. The crucial point of the technique is to define a set of *core formulae*. For instance, in the case of $\text{SL}(*)$, this set, denoted by $\text{Core}(X, \alpha)$, is indexed by a finite set of variables $X \subseteq_{\text{fin}} \text{VAR}$ and natural number $\alpha \geq 1$, and it is defined as:

$$\text{Core}(X, \alpha) \stackrel{\text{def}}{=} \{x = y, x \hookrightarrow y, \text{rem}_X \geq \beta \mid x, y \in X, \beta \in [0, \alpha]\}.$$

Here, $\text{rem}_X \geq \beta$ is a formula that is satisfied by a memory state (s, h) whenever the heap $h \setminus \{s(x) \mapsto s(y) \mid x, y \in X\}$, i.e. the heap obtained from h by discharging the memory cells that witness the satisfaction of the points-to predicates, contains at least β memory cells. It should be noted that $\text{rem}_X \geq \beta$ can be defined in $\text{SL}(*)$. Nonetheless, definability of the core formulae in the logic under analysis is not needed in order to study its computational complexity.

In general, we want the set of core formulae to satisfy three properties. First of all, the set of core formulae must be finite, as it is the case for the set $\text{Core}(X, \alpha)$. A second property is that the logic obtained by closing the core formulae under Boolean connectives should enjoy a small-heap property. One can show that this is the case for $\text{Core}(X, \alpha)$, and that each satisfiable Boolean combination of formulae from $\text{Core}(X, \alpha)$ can be satisfied by a memory state (s, h) such that $\text{card}(h) \leq \text{card}(X) + \alpha$. A third property is that Boolean combinations of core formulae exhaust the expressivity of the separation logic under analysis. With respect to $\text{SL}(*)$, this property can be formalised with the following proposition.

Proposition 5.4. For every formula φ in $\text{SL}(*)$ written with variables from X there is a Boolean combination ψ of formulae in $\text{Core}(X, |\varphi|)$ such that $\varphi \equiv \psi$.

By relying on the small-heap property of $\text{Core}(X, \alpha)$, this proposition leads to a small-heap property for $\text{SL}(*)$. When considering more complex logics, finding the right set of core formulae that satisfy Proposition 5.4 is quite challenging. To partially solve this issue, [104] shows a way of checking if this proposition holds by relying on a simulation argument, which helps us to systematically examine the core formulae we conjecture to be correct.

Indistinguishability relation and its simulation. As a preliminary result, [104] requires that Boolean combinations of core formulae capture every atomic formula of the logic. With respect to our example concerning $\text{SL}(*)$, the precise statement is given below.

Lemma 5.5. Let $X \subseteq_{\text{fin}} \text{VAR}$. Every formula among $\{\top, \text{emp}, x = y, x \hookrightarrow y \mid x, y \in X\}$ in $\text{SL}(*)$ is equivalent to a Boolean combination of formulae from $\text{Core}(X, 1)$.

Proving this lemma is quite straightforward: both the atomic formulae $x = y$ and $x \hookrightarrow y$ are already in $\text{Core}(X, 1)$, whereas $\top \equiv \text{rem}_X \geq 0$ and $\text{emp} \equiv \neg(\text{rem}_X \geq 1 \vee \bigvee_{x, y \in X} x \hookrightarrow y)$.

Once the “base case” of Lemma 5.5 is proved, [104] considers an *indistinguishability relation*, here denoted by $\approx_{X, \alpha}$, that relates memory states satisfying the same core formulae. Formally,

$$(s, h) \approx_{X, \alpha} (s', h') \text{ if and only if for every } \varphi \in \text{Core}(X, \alpha), (s, h) \models \varphi \text{ iff } (s', h') \models \varphi.$$

A first property of this indistinguishability relation is that it is an equivalence relation with finite-index, i.e. $\approx_{X,\alpha}$ has finitely many equivalence classes. This stems directly from the fact that $\text{Core}(X, \alpha)$ is finite. Its second key property is that $\approx_{X,\alpha}$ is a *simulation* with respect to the semantics of the separating conjunction. We call this the **-simulation property* of the core formulae. Its technical statement is formalised below.

Lemma 5.6 (*-simulation). Let $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq 1$. Consider $(s, h) \approx_{X,\alpha} (s', h')$. For every two heaps h_1 and h_2 and every $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$, if $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$ then there are two heaps h'_1 and h'_2 such that $h' = h'_1 + h'_2$, $(s, h_1) \approx_{X,\alpha_1} (s', h'_1)$ and $(s, h_2) \approx_{X,\alpha_2} (s', h'_2)$.

Consider two Boolean combinations φ, ψ of core formulae in $\text{Core}(X, \alpha_1)$ and $\text{Core}(X, \alpha_2)$, respectively. Fundamentally, this lemma states that whether two memory states (s, h) and (s', h') satisfy the same formulae in $\text{Core}(X, \alpha_1 + \alpha_2)$, then $(s, h) \models \varphi * \psi$ if and only if $(s', h') \models \varphi * \psi$. Thanks to this lemma, we can show that the formula $\varphi * \psi$ must be equivalent to a finite Boolean combination of core formulae from $\text{Core}(X, \alpha_1 + \alpha_2)$. The rough idea is that every equivalence class of $\approx_{X,\alpha_1+\alpha_2}$ can be characterised with a conjunction of literals built from formulae in $\text{Core}(X, \alpha_1 + \alpha_2)$. As $\approx_{X,\alpha_1+\alpha_2}$ has finitely many equivalence classes, this implies that $\varphi * \psi$ is equivalent to the (finite) disjunction of every conjunction corresponding to an equivalence class of a memory state satisfying $\varphi * \psi$. With a bottom-up argument and starting from the atomic formulae, Lemma 5.5 and Lemma 5.6 show that every formula is equivalent to a Boolean combination of core formulae, thus proving Proposition 5.4.

This approach has a drawback. Albeit giving us a good way of checking whether the core formulae capture the whole logic, the proof of Lemma 5.6 becomes quite technical and monolithic when considering complex logics. For instance, the proof of the *-simulation property for $\text{SL}(*, \hookrightarrow^\dagger)$ spawn a dozen pages of hard-to-check technical steps [56]. It is not reasonable to do the same for $\text{SL}([\exists]_1, *, [-], \hookrightarrow^\dagger)_{\mathcal{W}}$: we need a way to make the proof more modular.

5.2.2 Game Hopping.

In order to provide a modular proof of the *-simulation property, it is helpful to frame it within the standard tools of finite-model theory for first-order logic. In particular, the core formulae have a strong connections with the *Gaifman's locality Theorem* ([102], Theorem 4.22). Informally, this theorem proven by H. Gaifman in [73] states that every first-order formula is equivalent to a Boolean combination of *local formulae*. Skipping the technical definition of local formulae, we notice how Proposition 5.4 can be seen as an adaptation of Gaifman's locality Theorem to separation logic, where the core formulae enjoy the same property of the local formulae of first-order logic. This connection is quite revealing, the local formulae are connected to the notion of winning strategy for the duplicator on the Ehrenfeucht-Fraïssé games for first-order logic [102]. We already introduced the Ehrenfeucht-Fraïssé games in Section 4.2.2 in order to study the expressive power for ALT. We refer the reader to that section (or even better, to [102]), for an introduction on these types of games. When it comes to $\text{SL}(*)$, the EF-games can be defined on game states consisting of two memory states (s, h) and (s', h') and a *rank* (X, α) , where $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq 1$. As usual, the game is played by two players: the spoiler, who wants to prove that the two memory states can be told apart by a formula of the logic, and the duplicator, who instead wants to prove that they are indistinguishable. The EF-games for $\text{SL}(*)$ are defined in Figure 5.3. Let us write $(s, h) \sim_{X,\alpha} (s', h')$ whenever the duplicator has a winning strategy for the game $((s, h), (s', h'), (X, \alpha))$. We define the rank of a formula φ in $\text{SL}(*)$ as (X, α) ,

EF-Game played on the state $((s, h), (s', h'), (\mathbf{x}, \alpha))$.

if there is $\pi \in \{\top, \text{emp}, \mathbf{x} = \mathbf{y}, \mathbf{x} \rightarrowtail \mathbf{y} \mid \mathbf{x}, \mathbf{y} \in \mathbf{X}\}$ s.t. $((s, h) \models \pi \text{ iff } (s', h') \models \pi)$ does not hold
then the spoiler wins,
else if $\alpha = 1$ **then** the duplicator wins,
else ($\alpha \geq 2$) the spoiler chooses $(s^S, h^S) \in \{(s, h), (s', h')\}$.

The duplicator replies on the other memory state, say (s^D, h^D) . Afterwards,

1. The spoiler selects $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $\alpha = \alpha_1 + \alpha_2$.
2. The spoiler selects two heaps h_1^S and h_2^S such that $h^S = h_1^S + h_2^S$.
3. The duplicator selects two heaps h_1^D and h_2^D such that $h^D = h_1^D + h_2^D$.
4. The spoiler selects $j \in \{1, 2\}$.
5. The game continues on $((s^S, h_j^S), (s^D, h_j^D), (\mathbf{x}, \alpha_j))$.

Figure 5.3: Ehrenfeucht-Fraïssé games for $\text{SL}(*)$.

where \mathbf{X} is the set of variables appearing in φ whereas α is one plus the number of separating conjunctions in φ . The EF-games are sound and complete for $\text{SL}(*)$, as formalised below.

Proposition 5.7. Let (s, h) and (s', h') be two pointed forests. Let (\mathbf{x}, α) be a rank.

$$(s, h) \sim_{\mathbf{x}, \alpha} (s', h') \text{ iff for each formula } \varphi \text{ in } \text{SL}(*) \text{ of rank } (\mathbf{x}, \alpha), ((s, h) \models \varphi \text{ iff } (s', h') \models \varphi).$$

Instead of proving this proposition (which can be shown as we did for ALT, see Theorem 4.15), we are here interested in the connections between the EF-games and the core formulae. In particular, we show that the inclusion $\approx_{\mathbf{x}, \alpha} \subseteq \sim_{\mathbf{x}, \alpha}$ holds, i.e. if two memory states (s, h) and (s', h') satisfy the same core formulae from $\text{Core}(\mathbf{x}, \alpha)$, then the duplicator has a winning strategy for the game state $((s, h), (s', h'), (\mathbf{x}, \alpha))$.

Lemma 5.8. $\approx_{\mathbf{x}, \alpha} \subseteq \sim_{\mathbf{x}, \alpha}$

Proof. Suppose $(s, h) \approx_{\mathbf{x}, \alpha} (s', h')$. The proof is by induction on α .

base case: $\alpha = 1$. By Lemma 5.5, the two memory states satisfy the same atomic formulae.

With respect to the description of the games given in Figure 5.3, duplicator wins (line 3).

inductive step: $\alpha \geq 2$. Again, two memory states satisfy the same atomic formulae. So, with respect to the description of the games given in Figure 5.3, the spoiler must select $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $\alpha = \alpha_1 + \alpha_2$, as well as a memory state $(s^S, h^S) \in \{(s, h), (s', h')\}$ and two heaps h_1^S and h_2^S such that $h^S = h_1^S + h_2^S$. Duplicator replies in the other memory state, say (s^D, h^D) . From the $*$ -simulation lemma (Lemma 5.6), there are two heaps h_1^D and h_2^D such that $h^D = h_1^D + h_2^D$, $(s^S, h_1^S) \approx_{\mathbf{x}, \alpha_1} (s^D, h_1^D)$ and $(s^S, h_2^S) \approx_{\mathbf{x}, \alpha_2} (s^D, h_2^D)$. By induction hypothesis $(s^S, h_1^S) \sim_{\mathbf{x}, \alpha_1} (s^D, h_1^D)$ and $(s^S, h_2^S) \sim_{\mathbf{x}, \alpha_2} (s^D, h_2^D)$. Thus, selecting h_1^D and h_2^D leads to a winning strategy for the duplicator in the original game. \square

Relating the core formulae technique with Ehrenfeucht-Fraïssé games allows us to transfer more easily tools from various games on indistinguishability relations to the context of separation logic, which in turn helps us finding a more modular proof of Lemma 5.6 (and similar simulation properties). First of all, let us revisit the statement of Lemma 5.6. We introduce a binary relation $\leftrightarrow_{\mathbf{x}, \alpha}$, called *hop relation*, that stresses the property underlying the $*$ -simulation:

$(s, h) \leftrightarrow_{\mathbf{x}, \alpha} (s', h')$ iff for every two heaps h_1 and h_2 and every $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$,
if $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$ then there are two heaps h'_1 and h'_2
such that $h' = h'_1 + h'_2$, $(s, h_1) \approx_{\mathbf{x}, \alpha_1} (s', h'_1)$ and $(s, h_2) \approx_{\mathbf{x}, \alpha_2} (s', h'_2)$.

The hop relation allows us to succinctly rephrase Lemma 5.6 as the inclusion $\approx_{\mathbf{x}, \alpha} \subseteq \leftrightarrow_{\mathbf{x}, \alpha}$. In order to prove this inclusion in a modular way, given two memory states $(s, h) \approx_{\mathbf{x}, \alpha} (s', h')$, we build a *chain of hops* as the one schematised below:

$$(s, h) = (s_1, h_1) \leftrightarrow_{\mathbf{x}, \alpha} (s_2, h_2) \leftrightarrow_{\mathbf{x}, \alpha} \dots \leftrightarrow_{\mathbf{x}, \alpha} (s_{k-1}, h_{k-1}) \leftrightarrow_{\mathbf{x}, \alpha} (s_k, h_k) = (s', h').$$

At each hop $(s_j, h_j) \leftrightarrow_{\mathbf{x}, \alpha} (s_{j+1}, h_{j+1})$, where $j \in [1, k-1]$, the memory state (s_{j+1}, h_{j+1}) is constructed by updating (s_j, h_j) . The idea is that this update should be quite small and localised, so that it is easy to check that the two memory states are in the hop relation, or analogously that the duplicator wins the underlying EF-game on $((s_j, h_j), (s_{j+1}, h_{j+1}), (\mathbf{x}, \alpha))$. Each hop can be treated separately from the others, making the whole proof modular. Thanks to the chain of hops, we are able to conclude that $(s, h) \leftrightarrow_{\mathbf{x}, \alpha} (s', h')$ holds (thus, providing a proof of Lemma 5.6) from the transitivity of the hop relation, which we now show.

Lemma 5.9. $\leftrightarrow_{\mathbf{x}, \alpha}$ is reflexive and transitive.

Proof. Reflexivity is obvious. For transitivity, consider three memory states (s, h) , (s', h') and (s'', h'') . Suppose $(s, h) \leftrightarrow_{\mathbf{x}, \alpha} (s', h')$ and $(s', h') \leftrightarrow_{\mathbf{x}, \alpha} (s'', h'')$, and consider two heaps h_1 and h_2 and two natural numbers $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. From $(s, h) \leftrightarrow_{\mathbf{x}, \alpha} (s', h')$ there are two heaps h'_1 and h'_2 such that $h' = h'_1 + h'_2$, $(s, h_1) \approx_{\mathbf{x}, \alpha_1} (s', h'_1)$ and $(s, h_2) \approx_{\mathbf{x}, \alpha_2} (s', h'_2)$. From $(s', h') \leftrightarrow_{\mathbf{x}, \alpha} (s'', h'')$ there are h''_1 and h''_2 such that $h'' = h''_1 + h''_2$, $(s', h'_1) \approx_{\mathbf{x}, \alpha_1} (s'', h''_1)$ and $(s', h'_2) \approx_{\mathbf{x}, \alpha_2} (s'', h''_2)$. As $\approx_{\mathbf{x}, \alpha}$ is an equivalence relation, we conclude that $(s, h_1) \approx_{\mathbf{x}, \alpha_1} (s'', h''_1)$ and $(s, h_2) \approx_{\mathbf{x}, \alpha_2} (s'', h''_2)$. Thus, $(s, h) \leftrightarrow_{\mathbf{x}, \alpha} (s'', h'')$. \square

Due to the lack of a better terminology, we call the process of building a chain of hops *game hopping*. This term, as well as the general idea presented here, is borrowed from the homonymous technique in computational security [16]. In this framework, an attacker has an unknown probability of success against an environment. In order to compute this probability, a game hopping proof slowly changes the environment with a chain of updates that are shown admissible thanks to some underlying invariant. The process continues until we reach an environment where the probability of success can be easily computed. As an analogy, in our case the environments are given by the memory states and the attacker is the spoiler, which we want to prove having zero probability of winning the underlying EF-game. Our invariants are given by the equisatisfaction of the core formulae at each step of the chain of hops.

In our setting, a last question about game hopping is how to identify the updates we want to carry out in the chain. What we found to work well in practice is to define the core formulae starting from a partition on the heap, so that the satisfaction of each core formula is governed by exactly one part of the partition. We already applied this idea to the core formulae $\mathbf{Core}(\mathbf{x}, \alpha)$ of $\mathbf{SL}(*)$: given a memory state (s, h) , every $(\ell, \ell') \in h$ is involved in the satisfaction of either the core formula $\mathbf{x} \rightarrow y$, for some $\mathbf{x}, y \in \mathbf{X}$, or the core formula $\mathbf{rem}_{\mathbf{x}} \geq \beta$, but not both. Then, at each hop we only modify locations belonging to one element of the partition, leaving the other ones untouched. The benefits of this approach can be better appreciated in the next section, where we deal with the core formulae for the weak fragment of $\mathbf{SL}([\exists]_1, *, [\neg], \rightarrow^{\dagger})_{\mathcal{W}}$.

5.3 A FAMILY OF CORE FORMULAE CAPTURING THE FRAGMENT \mathcal{W}

We start adapting the core formulae technique to the weak fragment of $\text{SL}([\exists]_1, *, [\neg, \rightarrow^\dagger]_{\mathcal{W}}^S)$. Let us recall the grammar of this sublogic:

$$\mathcal{W} := \top \mid \text{emp} \mid t = t' \mid t \hookrightarrow t' \mid w \wedge w \mid \neg w \mid w * w \mid s * w \mid \exists u w$$

For the time being, we disregard the separating implication $s * w$, as it requires some analysis on the strong fragment, which will be carried out throughout Section 5.5 and Section 5.6. When the magic wand is dropped from the logic, \mathcal{W} becomes a fragment of $\text{SL}([\exists]_1, *, \neg)$, whose satisfiability problem has been proven PSPACE-complete using the core formulae technique in [55]. The family of core formulae we consider in this section can be shown to be equiexpressive to the ones in [55], while being better suited for game hopping.

We stress once more the goal of this section. Following Section 5.2, we want to define a set of core formulae whose Boolean combinations capture the expressive power of the weak fragment (excluding the operator $s * w$). As shown in [104], in order to show that the core formulae we define enjoy this property it is sufficient to prove that they capture the atomic formulae of the weak fragment and that they satisfy the $*$ -simulation property (see e.g. Lemma 5.6) and an analogous \exists -simulation property. Broadly speaking, these two properties imply that formulae of the form $\varphi * \psi$ or $\exists x \varphi$, where φ and ψ are Boolean combinations of core formulae, are themselves equivalent to Boolean combinations of core formulae. With a bottom-up argument starting from the base case of atomic formulae, this implies that every $*$ -free formula of the weak fragment can be translated into a Boolean combination of core formulae. To prove the $*$ -simulation property, we rely on the game hopping strategy sketched in Section 5.2.2. Our proof technique naturally divides the section in four steps. First (step I), we introduce a family of disjoint sets that induce a partition of the domain of the heap. These sets are used (step II) to define the core formulae, which are shown to capture the atomic formulae of the weak fragment. Afterwards, we prove (step III) that the core formulae satisfy the $*$ -simulation property and (step IV) the \exists -simulation property. This division serves as a roadmap for the much more involved Section 5.5, in which we introduce the core formulae for the strong fragment.

5.3.1 Step I: partitioning the heap.

We define the core formulae starting from a partition of the heap, which is in turn defined from a set of syntactical terms that correspond to specific locations of the heap. For the whole section, we fix $X \subseteq_{\text{fin}} \text{VAR} \setminus \{u\}$ to be a finite set of program variables not including the unique quantified variable name u , which we treat separately.

Definition 5.10 (Next-point variables and terms). We write $\text{NV}[\mathcal{W}]^X$ for the set of *next-point variables* $\{n(x) \mid x \in X\}$, $n(x)$ being a syntactical object. $T[\mathcal{W}]^X$ stands for set of *terms* $X \cup \text{NV}[\mathcal{W}]^X$. Given a memory state (s, h) , the evaluation $\llbracket . \rrbracket_{s,h}^X$ of a term is defined as $\llbracket x \rrbracket_{s,h}^X \stackrel{\text{def}}{=} s(x)$ for $x \in X$. If $s(x) \in \text{dom}(h)$ then $\llbracket n(x) \rrbracket_{s,h}^X \stackrel{\text{def}}{=} h(s(x))$, otherwise $\llbracket n(x) \rrbracket_{s,h}^X$ is not defined.

Intuitively, the next-point variable $n(x)$ corresponds to the location pointed by $s(x)$, if any. We already saw a similar concept in Section 3.2. The evaluation $\llbracket . \rrbracket_{s,h}^X$ leads to labelled locations.

Definition 5.11 (Labelled locations). Given a memory state (s, h) , we write $\text{Lab}[\mathcal{W}]_{s,h}^X$ for the set of locations $\{\llbracket t \rrbracket_{s,h}^X \mid t \in T[\mathcal{W}]^X\}$. The locations in $\text{Lab}[\mathcal{W}]_{s,h}^X$ are said to be *labelled*.

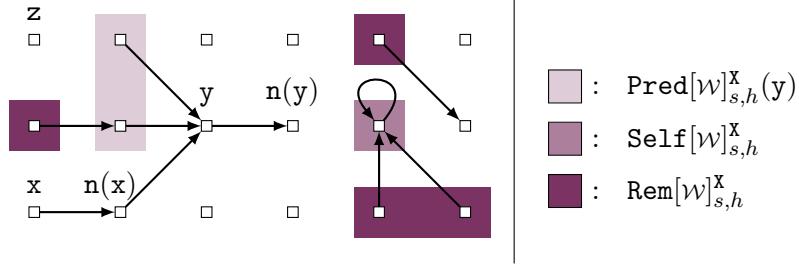


Figure 5.4: A memory state. The partition of the heap is highlighted.

Locations that are not in $\text{Lab}[w]_{s,h}^X$ are said to be *unlabelled*. Intuitively, the weak fragment can express different properties about labelled locations. For instance, we can check if two next-point variables $n(x)$ and $n(y)$ correspond to the same location with the formula $\exists u (x \hookrightarrow u \wedge y \hookrightarrow u)$. As such, every subheap $\{\ell \mapsto \ell'\} \subseteq h$, where ℓ is a labelled location, correspond to one part of the partition of h we are about to define. As h is functional, this partition can be equivalently described in terms of $\text{dom}(h)$. Excluding labelled locations, its parts are defined below.

Definition 5.12 (Predecessors, self-loops and the remainder). Let (s, h) be a memory state. We define the following subsets of $\text{dom}(h)$:

Predecessors. $\text{Pred}[w]_{s,h}^X(x) \stackrel{\text{def}}{=} \{\ell' \in \text{dom}(h) \mid h(\ell') = s(x) \text{ and } \ell' \notin \text{Lab}[w]_{s,h}^X\}$, where $x \in X$.

Informally, $\text{Pred}[w]_{s,h}^X(x)$ is the set of all unlabelled *predecessors* of $s(x)$.

Self-loops. $\text{Self}[w]_{s,h}^X \stackrel{\text{def}}{=} \{\ell \in \text{dom}(h) \mid h(\ell) = \ell \text{ and } \ell \notin \text{Lab}[w]_{s,h}^X\}$.

Informally, $\text{Self}[w]_{s,h}^X$ contains the set of unlabelled *self-loops*.

Remainder. $\text{Rem}[w]_{s,h}^X \stackrel{\text{def}}{=} \text{dom}(h) \setminus (\text{Lab}[w]_{s,h}^X \cup \text{Self}[w]_{s,h}^X \cup \bigcup_{x \in X} \text{Pred}[w]_{s,h}^X(x))$.

Informally, $\text{Rem}[w]_{s,h}^X$ is the set of unlabelled locations that are neither self-loop nor predecessors of variables in X .

Figure 5.4 highlights these sets on a memory state (s, h) . As we can see, all the locations in the domain of the heap are either labelled locations, or they belong to one of these sets. Indeed, directly from Definition 5.12, it is quite clear that these three types of sets, together with the memory cells of h that are labelled locations, uniquely define a partition of $\text{dom}(h)$ (and of h). This property is formalised in the following proposition, whose proof is left to the reader.

Proposition 5.13. Let (s, h) be a memory state. The set of all the non-empty sets among $\text{dom}(h) \cap \text{Lab}[w]_{s,h}^X$, $\text{Self}[w]_{s,h}^X$, $\text{Rem}[w]_{s,h}^X$ and all $\text{Pred}[w]_{s,h}^X(x)$ ($x \in X$), is a partition of $\text{dom}(h)$.

Due to the $*$ -simulation property we aim to establish, it is quite important to understand how this partition evolves when considering subheaps. For instance, given a location $\ell \in \text{Pred}[w]_{s,h}^X(x)$, it is quite easy to see that in every subheap $h' \subseteq h$ where $\ell \in \text{dom}(h')$, we find that ℓ belongs to $\text{Pred}[w]_{s,h'}^X(x)$. Similar properties can be stated for $\text{Self}[w]_{s,h}^X$ and $\text{Rem}[w]_{s,h}^X$. In general, the converse does not hold: given a location $\ell \in \text{Pred}[w]_{s,h}^X(x)$, it can be that $\ell \notin \text{Pred}[w]_{s,h}^X(x)$. This is typically the case when ℓ corresponds to a next-point variable with respect to (s, h) , but it becomes an unlabelled location when considering (s, h') . The following technical lemma shows all these relationships between partitions.

Lemma 5.14. Let (s, h) be a memory state. Consider a subheap $h' \subseteq h$ and $\ell \in \text{LOC}$.

- (I) for every term $t \in T[\mathcal{W}]^X$, if $\llbracket t \rrbracket_{s,h'}^X$ is defined then $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$.
- (II) $\text{Pred}[\mathcal{W}]_{s,h'}^X(x) = (\text{Pred}[\mathcal{W}]_{s,h}^X(x) \cap \text{dom}(h')) \cup \{\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'}^X \mid h'(\ell) = s(x)\}$.
- (III) $\text{Self}[\mathcal{W}]_{s,h'}^X = (\text{Self}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h')) \cup \{\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'}^X \mid h'(\ell) = \ell\}$.
- (IV) $\text{Rem}[\mathcal{W}]_{s,h'}^X = (\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h')) \cup \{\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'}^X \mid \ell \in \text{dom}(h'), h'(\ell) \neq \ell \text{ and } \forall x \in X, h'(\ell) \neq s(x)\}$.

The proofs of these four statements are all quite simple and follow from Definition 5.12. Below, we show the proof of the first two statements, leaving the other two to the reader.

Proof of (I). The proof is obvious when t is a variable in X . So, let us assume $t = n(x) \in NV[\mathcal{W}]^X$. If $\llbracket n(x) \rrbracket_{s,h'}^X$ is defined, then $\llbracket n(x) \rrbracket_{s,h'}^X \stackrel{\text{by def}}{=} h'(s(x))$. From $h' \subseteq h$ we have $h(s(x)) = h'(s(x))$. Therefore, by $\llbracket n(x) \rrbracket_{s,h}^X \stackrel{\text{by def}}{=} h(s(x))$ we derive $\llbracket t \rrbracket_{s,h'}^X = \llbracket t \rrbracket_{s,h}^X$. \square

Proof of (II). (\Rightarrow): Let us consider a location $\ell \in \text{Pred}[\mathcal{W}]_{s,h'}^X(x)$, and thus $\ell \notin \text{Lab}[\mathcal{W}]_{s,h'}^X$ and $h'(\ell) = s(x)$. From $h' \subseteq h$ we derive that $h(\ell) = s(x)$. In the case that $\ell \notin \text{Lab}[\mathcal{W}]_{s,h}^X$, then we derive $\ell \in \text{Pred}[\mathcal{W}]_{s,h}^X(x) \cap \text{dom}(h')$. Otherwise, $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'}^X$ and $h'(\ell) = s(x)$. (\Leftarrow): Clearly, if $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'}^X$ and $h'(\ell) = s(x)$ then $\ell \notin \text{Lab}[\mathcal{W}]_{s,h'}^X$, and so we derive $\ell \in \text{Pred}[\mathcal{W}]_{s,h'}^X(x)$. Otherwise, let us consider the case were $\ell \in \text{Self}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h')$. We have $\ell \notin \text{Lab}[\mathcal{W}]_{s,h}^X$, $h(\ell) = s(x)$ and $\ell \in \text{dom}(h')$. From $h' \subseteq h$ we derive $h'(\ell) = s(x)$. From Lemma 5.14(I) we derive $\ell \notin \text{Lab}[\mathcal{W}]_{s,h'}^X$. Thus, $\ell \in \text{Pred}[\mathcal{W}]_{s,h'}^X(x)$. \square

5.3.2 Step II: the core formulae for \mathcal{W} .

We use the partition as a base to define the core formulae. Each of these formulae describes a feature of one of the sets in the partition, and its satisfaction does not depend on the properties of the other sets. For instance, let us consider a predicate $\text{self}_X^{\mathcal{W}} \geq 3$ stating that a memory state (s, h) contains at least 3 unlabelled self-loops. Clearly, $\text{self}_X^{\mathcal{W}} \geq 3$ depends on the cardinality of $\text{Self}[\mathcal{W}]_{s,h}^X$, but it is completely independent from the locations in other sets of the partition. As done for the core formulae of $\text{SL}(\ast)$, given as an example during Section 5.2.1, the core formulae $\text{Core}[\mathcal{W}](X, \alpha)$ for the weak fragment are parametric on X and a natural number $\alpha \geq 1$. Here, α is a quantity that roughly expresses upper bounds on the capabilities of a formula φ to check the sizes of the sets of the partition. As we will see in Section 5.6, this bound is connected with the number of separating conjunctions in φ , and ultimately to its size. $\text{Core}[\mathcal{W}](X, \alpha)$ is divided into two sets, a *skeleton set* $\text{Sk}[\mathcal{W}](X, \alpha)$ expressing structural properties that do not depend on the assignment of u , and an *observed set* $\text{Obs}[\mathcal{W}](X)$ of relationships between the memory state and the location currently assigned to u . The skeleton set is defined below, and the semantics of its formulae is given in Figure 5.5. We recall that X does not contain u .

$$\text{Sk}[\mathcal{W}](X, \alpha) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} t_1 = t_2, \quad t_1 \hookrightarrow _, \quad t_1 \hookrightarrow x, \quad t_1 \hookrightarrow t_1, \\ \text{pred}_X^{\mathcal{W}}(x) \geq \beta, \quad \text{self}_X^{\mathcal{W}} \geq \beta, \quad \text{rem}_X^{\mathcal{W}} \geq \beta \end{array} \middle| \begin{array}{l} x \in X, \quad \beta \in [1, \alpha], \\ t_1, t_2 \in T[\mathcal{W}]^X \end{array} \right\}.$$

As we can see, the skeleton set contains equality and points-to relation between terms, and formulae that state lower-bounds on the cardinality of the sets $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$, $\text{Self}[\mathcal{W}]_{s,h}^X$ and $\text{Rem}[\mathcal{W}]_{s,h}^X$. In particular, it should be noted that $(s, h) \models t = t$ if and only if $\llbracket t \rrbracket_{s,h}^X$ is defined. Notice that these formulae are syntactically defined so that $n(x) \hookrightarrow n(y)$ is not a core formula. In particular, the only points-to relation between next-point variables is $n(x) \hookrightarrow n(x)$. The reason for this syntactical restriction is quite simple: $n(x) \hookrightarrow n(x)$ is definable in the weak fragment as $\exists u (x \hookrightarrow u \wedge u \hookrightarrow u)$, whereas $n(x) \hookrightarrow n(y)$ requires two quantified variables to be defined. With

$(s, h) \models t_1 = t_2$	iff $\llbracket t_1 \rrbracket_{s,h}^X$ and $\llbracket t_2 \rrbracket_{s,h}^X$ are defined and $\llbracket t_1 \rrbracket_{s,h}^X = \llbracket t_2 \rrbracket_{s,h}^X$,
$(s, h) \models t_1 \hookrightarrow __$	iff $\llbracket t_1 \rrbracket_{s,h}^X$ is defined and $\llbracket t_1 \rrbracket_{s,h}^X \in \text{dom}(h)$,
$(s, h) \models t_1 \hookrightarrow x$	iff $\llbracket t_1 \rrbracket_{s,h}^X$ is defined and $h(\llbracket t_1 \rrbracket_{s,h}^X) = s(x)$,
$(s, h) \models t_1 \hookrightarrow t_1$	iff $\llbracket t_1 \rrbracket_{s,h}^X$ is defined and $h(\llbracket t_1 \rrbracket_{s,h}^X) = \llbracket t_1 \rrbracket_{s,h}^X$,
$(s, h) \models \text{pred}_X^W(x) \geq \beta$	iff $\text{card}(\text{Pred}[W]_{s,h}^X(x)) \geq \beta$,
$(s, h) \models \text{self}_X^W \geq \beta$	iff $\text{card}(\text{Self}[W]_{s,h}^X) \geq \beta$,
$(s, h) \models \text{rem}_X^W \geq \beta$	iff $\text{card}(\text{Rem}[W]_{s,h}^X) \geq \beta$.

Figure 5.5: Semantics of the formulae in $\text{Sk}[W](X, \alpha)$, with respect to a memory state (s, h) .

$(s, h) \models u = t$	iff $\llbracket t \rrbracket_{s,h}^X$ is defined and $s(u) = \llbracket t \rrbracket_{s,h}^X$,
$(s, h) \models u \in \text{pred}_X^W(x)$	iff $s(u) \in \text{Pred}[W]_{s,h}^X(x)$,
$(s, h) \models u \in \text{self}_X^W$	iff $s(u) \in \text{Self}[W]_{s,h}^X$,
$(s, h) \models u \in \text{rem}_X^W$	iff $s(u) \in \text{Rem}[W]_{s,h}^X$.

Figure 5.6: Semantics of the formulae in $\text{Obs}[W](X)$, with respect to a memory state (s, h) .

respect to the three core formulae $\text{pred}_X^W(x) \geq \beta$, $\text{self}_X^W \geq \beta$ and $\text{rem}_X^W \geq \beta$, it is important to notice that β is bounded by α . Because of this, the core formulae cannot distinguish two memory states exceeding α locations in the three sets corresponding to these core formulae. As we will see in Section 5.6, this property is essential in order to conclude that $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^\#]^S_W)$ enjoys a polynomial small-heap property. Let us now move to the observed set:

$$\text{Obs}[W](X) \stackrel{\text{def}}{=} \left\{ u = t, u \in \text{pred}_X^W(x), u \in \text{self}_X^W, u \in \text{rem}_X^W \mid x \in X \text{ and } t \in T[W]^X \right\}.$$

The semantics of these formulae is given in Figure 5.6, and it is quite self-explanatory. The formula $u = t$ checks whether $s(u)$ is a labelled location. The formulae $u \in \text{pred}_X^W(x)$, $u \in \text{self}_X^W$ and $u \in \text{rem}_X^W$ check whether the location currently assigned to u is in the set $\text{Pred}[W]_{s,h}^X(x)$, in the set $\text{Self}[W]_{s,h}^X$ or in the set $\text{Rem}[W]_{s,h}^X$, respectively.

As described in the previous section (Lemma 5.5), a first key property the core formulae must satisfy is that their Boolean combinations must be able to express the atomic formulae of separation logic. For the weak fragment, this can be formalised as follows.

Lemma 5.15. Every atomic formula of the weak fragment written with variables from $X \cup \{u\}$ is equivalent to a Boolean combination of formulae from $\text{Core}[W](X, 1)$.

Proof (sketch). The proof is straightforward. Here, we simply give the definition of the atomic formulae in term of Boolean combinations of core formulae from $\text{Core}[W](X, 1)$, leaving the proof of their correctness to the reader. The atomic formulae $x = y$, $u = x$ and $x \hookrightarrow y$, where $x, y \in X$, are already core formulae. Otherwise,

$$x = u \equiv u = x, \quad x \hookrightarrow u \equiv u = n(x), \quad \top \equiv \text{rem}_X^W \geq 1 \vee \neg \text{rem}_X^W \geq 1,$$

$$u \hookrightarrow x \equiv u \in \text{pred}_X^W(x) \vee \bigvee_{t \in T[W]^X} (u = t \wedge t \hookrightarrow x),$$

$$\begin{aligned} u \hookrightarrow u &\equiv u \in \text{self}_x^{\mathcal{W}} \vee \bigvee_{t \in T[\mathcal{W}]^x} (u = t \wedge t \hookrightarrow t), \\ \text{emp} &\equiv \neg(\text{pred}_x^{\mathcal{W}}(x) \geq 1 \vee \text{self}_x^{\mathcal{W}} \geq 1 \vee \text{rem}_x^{\mathcal{W}} \geq 1 \vee \bigvee_{t \in T[\mathcal{W}]^x} t \hookrightarrow _). \end{aligned} \quad \square$$

5.3.3 Step III: indistinguishability relation, hops and $*$ -simulation.

After showing that the core formulae capture the atomic formulae of the logic, we show that they satisfy the $*$ -simulation property. As we saw in the previous section, this property is stated using an indistinguishability relation on memory state that is governed by the satisfaction of the core formulae. For the weak fragment, this relation is defined as follows.

Definition 5.16 (\mathcal{W} -indistinguishable memory states). We write $\approx_{x,\alpha}^{\mathcal{W}}$ for the equivalence relation on memory states characterised as:

$$(s, h) \approx_{x,\alpha}^{\mathcal{W}} (s', h') \text{ if and only if for every } \varphi \in \text{Core}[\mathcal{W}](x, \alpha), (s, h) \models \varphi \text{ iff } (s', h') \models \varphi.$$

The $*$ -simulation property corresponds to showing the inclusion $\approx_{x,\alpha}^{\mathcal{W}} \subseteq \leftrightarrow_{x,\alpha}^{\mathcal{W}}$, where $\leftrightarrow_{x,\alpha}^{\mathcal{W}}$ is the following hop relation.

Definition 5.17 (\mathcal{W} -hop relation). We write $\leftrightarrow_{x,\alpha}^{\mathcal{W}}$ for the relation on memory states such that

$$\begin{aligned} (s, h) \leftrightarrow_{x,\alpha}^{\mathcal{W}} (s', h') &\text{ iff for every two heaps } h_1 \text{ and } h_2 \text{ and every } \alpha_1 \geq 1 \text{ and } \alpha_2 \geq 1, \\ &\text{if } h = h_1 + h_2 \text{ and } \alpha = \alpha_1 + \alpha_2 \text{ then there are two heaps } h'_1 \text{ and } h'_2 \\ &\text{such that } h' = h'_1 + h'_2, (s, h_1) \approx_{x,\alpha_1}^{\mathcal{W}} (s', h'_1) \text{ and } (s, h_2) \approx_{x,\alpha_2}^{\mathcal{W}} (s', h'_2). \end{aligned}$$

The hop relation $\leftrightarrow_{x,\alpha}^{\mathcal{W}}$ is both reflexive and transitive (see Lemma 5.9).

As described in the previous section, in order to prove the inclusion $\approx_{x,\alpha}^{\mathcal{W}} \subseteq \leftrightarrow_{x,\alpha}^{\mathcal{W}}$, given two memory states (s, h) and (s', h') such that $(s, h) \approx_{x,\alpha}^{\mathcal{W}} (s', h')$, we aim at building a chain of hops

$$(s, h) = (s_1, h_1) \leftrightarrow_{x,\alpha}^{\mathcal{W}} (s_2, h_2) \leftrightarrow_{x,\alpha}^{\mathcal{W}} \dots \leftrightarrow_{x,\alpha}^{\mathcal{W}} (s_{k-1}, h_{k-1}) \leftrightarrow_{x,\alpha}^{\mathcal{W}} (s_k, h_k) = (s', h').$$

Each hop $(s_j, h_j) \leftrightarrow_{x,\alpha}^{\mathcal{W}} (s_{j+1}, h_{j+1})$ in this chain corresponds to one of four intermediate results, which we divide into two lemmata (Lemmata 5.18 and 5.19, below). The hops are then put together in the proof of the $*$ -simulation property formalised in Lemma 5.20. The first lemma involves the “base case” of memory states that are in the indistinguishability relation $\approx_{x,\alpha}^{\mathcal{W}}$ for every $\alpha \geq 1$. Notably, these memory states agree on the cardinality of their predecessor sets, self-loop set and remainder set.

Lemma 5.18. For every $\alpha \geq 1$, $\left(\bigcap_{\alpha' \geq 1} \approx_{x,\alpha'}^{\mathcal{W}} \right) \subseteq \leftrightarrow_{x,\alpha}^{\mathcal{W}}$.

Proof. Let (s, h) and (s', h') be two memory states such that $((s, h), (s', h')) \in \left(\bigcap_{\alpha' \geq 1} \approx_{x,\alpha'}^{\mathcal{W}} \right)$. Let us consider a bijection $f : \text{LOC} \rightarrow \text{LOC}$ such that

- 1_f. $f(s(u)) = s'(u)$ and for every $t \in T[\mathcal{W}]^x$, if $\llbracket t \rrbracket_{s,h}^x$ is defined then $f(\llbracket t \rrbracket_{s,h}^x) = \llbracket t \rrbracket_{s',h'}^x$,
- 2_f. for every $x \in X$, $\text{Pred}[\mathcal{W}]_{s',h'}^x(x) = f(\text{Pred}[\mathcal{W}]_{s,h}^x(x))$,
- 3_f. $\text{Self}[\mathcal{W}]_{s',h'}^x = f(\text{Self}[\mathcal{W}]_{s,h}^x)$,
- 4_f. $\text{Rem}[\mathcal{W}]_{s',h'}^x = f(\text{Rem}[\mathcal{W}]_{s,h}^x)$,

where we recall that given a set of locations L , $f(L) \stackrel{\text{by def}}{=} \{f(\ell) \mid \ell \in L\}$.

The existence of the bijection \mathfrak{f} stems directly from $((s, h), (s', h')) \in (\bigcap_{\alpha' \geq 1} \approx_{X, \alpha'}^W)$ together with Proposition 5.13. Indeed, the constraint (1_f) holds as the two memory states satisfy the same core formulae from

$$\{\mathbf{t}_1 = \mathbf{t}_2, \quad \mathbf{t}_1 \hookrightarrow _, \quad \mathbf{t}_1 \hookrightarrow X, \quad \mathbf{t}_1 \hookrightarrow \mathbf{t}_1 \mid \mathbf{t}_1, \mathbf{t}_2 \in T[W]^X\} \cup \text{Obs}[W](X).$$

Instead, the other three constraints follow as the membership in $(\bigcap_{\alpha' \geq 1} \approx_{X, \alpha'}^W)$ implies that for all $X \in X$ $\text{card}(\text{Pred}[W]_{s, h}^X(X)) = \text{card}(\text{Pred}[W]_{s', h'}^X(X))$ $\text{card}(\text{Self}[W]_{s, h}^X) = \text{card}(\text{Self}[W]_{s', h'}^X)$ and $\text{card}(\text{Rem}[W]_{s, h}^X) = \text{card}(\text{Rem}[W]_{s', h'}^X)$. Let us show that $(s, h) \leftrightarrow_{X, \alpha}^W (s', h')$.

Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Notice that this requires α to be at least two, otherwise the lemma trivially holds. Consider the heaps h'_1 and h'_2 defined as follows:

$$h'_1 = \{(\ell, \ell') \in h' \mid \mathfrak{f}^{-1}(\ell) \in \text{dom}(h_1)\}, \quad h'_2 = \{(\ell, \ell') \in h' \mid \mathfrak{f}^{-1}(\ell) \in \text{dom}(h_2)\}.$$

Notice that, since \mathfrak{f} is a bijection and h_1 and h_2 are disjoint, we have $h' = h'_1 + h'_2$. Let us discuss the following properties (A)–(D) of h'_j (where $j \in \{1, 2\}$):

- A. (a) for every $\mathbf{t} \in T[W]^X$, $\llbracket \mathbf{t} \rrbracket_{s, h_j}^X$ is defined iff so is $\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X$. If defined, $\mathfrak{f}(\llbracket \mathbf{t} \rrbracket_{s, h_j}^X) = \llbracket \mathbf{t} \rrbracket_{s', h'_j}^X$.
- (b) $\llbracket \mathbf{t} \rrbracket_{s, h_j}^X \in \text{dom}(h_j)$ if and only if $\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X \in \text{dom}(h'_j)$.
- (c) Given $\mathbf{t}' \in X \cup \{\mathbf{t}\}$, we have $h_j(\llbracket \mathbf{t} \rrbracket_{s, h_j}^X) = \llbracket \mathbf{t}' \rrbracket_{s, h_j}^X$ if and only if $h'_j(\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X) = \llbracket \mathbf{t}' \rrbracket_{s', h'_j}^X$.
- (d) $\ell \in \text{Lab}[W]_{s', h'}^X \setminus \text{Lab}[W]_{s', h'_j}^X$ if and only if $\mathfrak{f}^{-1}(\ell) \in \text{Lab}[W]_{s, h}^X \setminus \text{Lab}[W]_{s, h_j}^X$.

These four statements follow primarily from Lemma 5.14(I) and $(s, h) \approx_{X, 1}^W (s', h')$.

In the following, we show the left-to-right direction of each of these statements. The right-to-left direction follows analogously, by relying on the fact that \mathfrak{f} is bijective.

Proof of (a). Obvious for $\mathbf{t} \in X$, so suppose $\mathbf{t} = n(x) \in NV[W]^X$.

(\Rightarrow): Suppose $\llbracket n(x) \rrbracket_{s, h_j}^X$ to be defined, i.e. $\llbracket n(x) \rrbracket_{s, h_j}^X \stackrel{\text{by def}}{=} h_j(s(x))$. From $h_j \subseteq h$ and Lemma 5.14(I), $\llbracket n(x) \rrbracket_{s, h}^X = \llbracket n(x) \rrbracket_{s, h_j}^X$. Since (s, h) and (s', h') equisatisfy the formula $x \hookrightarrow _$, we conclude that $\llbracket n(x) \rrbracket_{s', h'}^X$ is also defined. From the property (1_f) of \mathfrak{f} , $\mathfrak{f}(s(x)) = s'(x)$. So, $(s(x), h'(s'(x))) \in \text{dom}(h'_j)$ by definition of h'_j . By Lemma 5.14(I), $\llbracket n(x) \rrbracket_{s', h'}^X = \llbracket n(x) \rrbracket_{s', h'_j}^X$. From the property (1_f) of \mathfrak{f} , $\mathfrak{f}(\llbracket n(x) \rrbracket_{s, h_j}^X) = \llbracket n(x) \rrbracket_{s', h'_j}^X$.

Proof of (b). (\Rightarrow): Suppose $\llbracket \mathbf{t} \rrbracket_{s, h_j}^X \in \text{dom}(h_j)$. From (a), $\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X = \mathfrak{f}(\llbracket \mathbf{t} \rrbracket_{s, h_j}^X)$. Therefore, by definition of h'_j , it is sufficient to show that $\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X \in \text{dom}(h')$. By $h_j \subseteq h$ and Lemma 5.14(I) we derive that $\llbracket \mathbf{t} \rrbracket_{s, h_j}^X = \llbracket \mathbf{t} \rrbracket_{s, h}^X \in \text{dom}(h)$. Since (s, h) and (s', h') equisatisfy the core formula $\mathbf{t} \hookrightarrow _$, we have $\llbracket \mathbf{t} \rrbracket_{s', h'}^X \in \text{dom}(h')$. By Lemma 5.14(I), $\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X = \llbracket \mathbf{t} \rrbracket_{s', h'}^X$ and so $\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X \in \text{dom}(h')$.

Proof of (c). (\Rightarrow): $h_j(\llbracket \mathbf{t} \rrbracket_{s, h_j}^X) = \llbracket \mathbf{t}' \rrbracket_{s, h_j}^X$ and so by (a) we have $\mathfrak{f}(\llbracket \mathbf{t} \rrbracket_{s, h_j}^X) = \llbracket \mathbf{t} \rrbracket_{s', h'_j}^X$ and $\mathfrak{f}(\llbracket \mathbf{t}' \rrbracket_{s, h_j}^X) = \llbracket \mathbf{t}' \rrbracket_{s', h'_j}^X$. By definition of h'_j , showing $h'(\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X) = \llbracket \mathbf{t}' \rrbracket_{s', h'_j}^X$ is sufficient. By $h_j \subseteq h$ we derive $h(\llbracket \mathbf{t} \rrbracket_{s, h_j}^X) = \llbracket \mathbf{t}' \rrbracket_{s, h_j}^X$, whereas by Lemma 5.14(I), $\llbracket \mathbf{t} \rrbracket_{s, h_j}^X = \llbracket \mathbf{t} \rrbracket_{s, h}^X$ and $\llbracket \mathbf{t}' \rrbracket_{s, h_j}^X = \llbracket \mathbf{t}' \rrbracket_{s, h}^X$. Since (s, h) and (s', h') equisatisfy the core formula $\mathbf{t} \hookrightarrow \mathbf{t}'$, where $\mathbf{t}' \in X \cup \{\mathbf{t}\}$, we conclude that $h'(\llbracket \mathbf{t} \rrbracket_{s', h'}^X) = \llbracket \mathbf{t}' \rrbracket_{s', h'}^X$. By Lemma 5.14(I), $\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X = \llbracket \mathbf{t} \rrbracket_{s', h'}^X$ and $\llbracket \mathbf{t}' \rrbracket_{s', h'_j}^X = \llbracket \mathbf{t}' \rrbracket_{s', h'}^X$. Thus, $h'(\llbracket \mathbf{t} \rrbracket_{s', h'_j}^X) = \llbracket \mathbf{t}' \rrbracket_{s', h'_j}^X$.

Proof of (d). (\Rightarrow): Assume $\ell \in \text{Lab}[W]_{s', h'}^X \setminus \text{Lab}[W]_{s', h'_j}^X$. As $\ell \in \text{Lab}[W]_{s', h'}^X$ there is a term $\mathbf{t} \in T[W]^X$ such that $\llbracket \mathbf{t} \rrbracket_{s', h'}^X = \ell$. Since (s, h) and (s', h') equisatisfy the core formula $\mathbf{t} = \mathbf{t}$, $\llbracket \mathbf{t} \rrbracket_{s, h}^X$ is defined, and thus $\mathfrak{f}(\llbracket \mathbf{t} \rrbracket_{s, h}^X) = \ell$ holds from the property (1_f) of \mathfrak{f} . So, $\mathfrak{f}^{-1}(\ell) \in \text{Lab}[W]_{s, h}^X$. Ad absurdum, suppose $\mathfrak{f}^{-1}(\ell) \in \text{Lab}[W]_{s, h_j}^X$. From (a) we have

$\ell \in \text{Lab}[\mathcal{W}]_{s',h'_j}^X$, contradicting $\ell \in \text{Lab}[\mathcal{W}]_{s',h'}^X \setminus \text{Lab}[\mathcal{W}]_{s',h'_j}^X$. Thus, $f^{-1}(\ell) \notin \text{Lab}[\mathcal{W}]_{s,h}^X$ and therefore $f^{-1}(\ell) \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$.

- B. For every $x \in X$, $\text{Pred}[\mathcal{W}]_{s',h'_j}^X(x) = f(\text{Pred}[\mathcal{W}]_{s,h_j}^X(x))$.

Given a location $\ell \in \text{LOC}$, we prove the following two equivalences:

- d. $\ell \in \text{Pred}[\mathcal{W}]_{s',h'}^X(x) \cap \text{dom}(h'_j)$ if and only if $f^{-1}(\ell) \in \text{Pred}[\mathcal{W}]_{s,h}^X(x) \cap \text{dom}(h_j)$,
- e. $\ell \in \text{Lab}[\mathcal{W}]_{s',h'}^X \setminus \text{Lab}[\mathcal{W}]_{s',h'_j}^X$ and $h'_j(\ell) = s'(x)$ if and only if
 $f^{-1}(\ell) \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$ and $h_j(f^{-1}(\ell)) = s(x)$.

So that (B) follows directly from Lemma 5.14(II) and the fact that f is bijective.

Proof of (d). Both direction hold directly by definition of h'_j and from the property (2_f) of f .

Proof of (e). (\Rightarrow): Assume $\ell \in \text{Lab}[\mathcal{W}]_{s',h'}^X \setminus \text{Lab}[\mathcal{W}]_{s',h'_j}^X$ and $h'_j(\ell) = s'(x)$. From (A)(d),

$f^{-1}(\ell) \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$. From $h'_j(\ell) = s'(x)$ and by definition of h'_j , we have $h'(\ell) = s'(x)$ and $f^{-1}(\ell) \in \text{dom}(h_j)$. As (s, h) and (s', h') equisatisfy the core formula $t \hookrightarrow x$, we have that $h(f^{-1}(\ell)) = s(x)$. By $f^{-1}(\ell) \in \text{dom}(h_j)$, $h_j(f^{-1}(\ell)) = s(x)$.

(\Leftarrow): Analogous to the other direction, by relying on the bijectivity of f .

- C. $\text{Self}[\mathcal{W}]_{s',h'_j}^X = f(\text{Self}[\mathcal{W}]_{s,h_j}^X)$.

Given a location $\ell \in \text{LOC}$, we prove the following two equivalences:

- f. $\ell \in \text{Self}[\mathcal{W}]_{s',h'}^X \cap \text{dom}(h'_j)$ if and only if $f^{-1}(\ell) \in \text{Self}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j)$,
- g. $\ell \in \text{Lab}[\mathcal{W}]_{s',h'}^X \setminus \text{Lab}[\mathcal{W}]_{s',h'_j}^X$ and $h'_j(\ell) = \ell$ if and only if
 $f^{-1}(\ell) \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$ and $h_j(f^{-1}(\ell)) = f^{-1}(\ell)$.

So that (C) follows directly from Lemma 5.14(III) and the fact that f is bijective.

Proof of (f). Both directions hold directly by definition of h'_j and from the property (3_f) of f .

Proof of (g). (\Rightarrow): Assume $\ell \in \text{Lab}[\mathcal{W}]_{s',h'}^X \setminus \text{Lab}[\mathcal{W}]_{s',h'_j}^X$ and $h'_j(\ell) = \ell$. From (A)(d),

$f^{-1}(\ell) \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$. From $h'_j(\ell) = \ell$ and by definition of h'_j we have $h'(\ell) = \ell$ and $f^{-1}(\ell) \in \text{dom}(h_j)$. Since (s, h) and (s', h') equisatisfy the core formula $t \hookrightarrow t$, we conclude that $h(f^{-1}(\ell)) = f^{-1}(\ell)$. By $f^{-1}(\ell) \in \text{dom}(h_j)$, $h_j(f^{-1}(\ell)) = f^{-1}(\ell)$.

(\Leftarrow): Analogous to the other direction, by relying on the bijectivity of f .

- D. $\text{Rem}[\mathcal{W}]_{s',h'_j}^X = f(\text{Rem}[\mathcal{W}]_{s,h_j}^X)$.

Given a location $\ell \in \text{LOC}$ and $\ell' = f^{-1}(\ell)$, we prove the following two equivalences:

- h. $\ell \in \text{Rem}[\mathcal{W}]_{s',h'}^X \cap \text{dom}(h'_j)$ if and only if $\ell' \in \text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j)$,
- i. $\ell \in \text{Lab}[\mathcal{W}]_{s',h'}^X \setminus \text{Lab}[\mathcal{W}]_{s',h'_j}^X$, $\ell \in \text{dom}(h'_j)$, $h'_j(\ell) \neq \ell$ and $\forall x \in X$ $h'_j(\ell) \neq s'(x)$ iff
 $\ell' \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$, $\ell' \in \text{dom}(h_j)$, $h_j(\ell') \neq \ell'$ and $\forall x \in X$ $h_j(\ell') \neq s(x)$.

So that (D) follows directly from Lemma 5.14(IV) and the fact that f is bijective.

Proof of (h). Both directions hold directly from the definition of h'_j and the property (4_f) of f .

Proof of (i). (\Rightarrow): Assume $\ell \in \text{Lab}[\mathcal{W}]_{s',h'}^X \setminus \text{Lab}[\mathcal{W}]_{s',h'_j}^X$, $\ell \in \text{dom}(h'_j)$, $h'_j(\ell) \neq \ell$ and for

every $x \in X$, $h'_j(\ell) \neq s'(x)$. From (A)(d), $f^{-1}(\ell) = \ell' \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$. From the definition of h'_j we have $f^{-1}(\ell) \in \text{dom}(h_j)$. *Ad absurdum*, suppose $h_j(\ell') = \ell'$. Then, by (C)(g) we derive $h'_j(\ell) = \ell$, in contradiction with the hypothesis. Thus, $h_j(\ell') \neq \ell'$. Similarly, *ad absurdum* suppose that there is $x \in X$ such that $h_j(\ell') = s(x)$. Then, by (B)(g) we derive $h'_j(\ell) = s'(x)$, in contradiction with the hypothesis. Thus, for every $x \in X$, $h_j(\ell') \neq s(x)$.

(\Leftarrow): Analogous to the other direction, by relying on the bijectivity of f .

Thanks to the properties (A)–(D), proving $(s, h_j) \approx_{\mathbf{x}, \alpha_j}^{\mathcal{W}} (s', h'_j)$, for $j \in \{1, 2\}$, is straightforward. Consider a core formula φ in $\text{Core}[\mathcal{W}](\mathbf{x}, \alpha_j)$. Then, $(s, h_j) \models \varphi$ iff $(s', h'_j) \models \varphi$, as shown below:

case: $\varphi = t_1 = t_2$. Follows directly from (A)(a) and the fact that \mathbf{f} is a bijection.

case: $\varphi = t \hookrightarrow __$. Follows directly from (A)(b).

case: $\varphi = t \hookrightarrow x$ or $\varphi = t \hookrightarrow t$. Follows directly from (A)(c).

case: $\varphi = \text{pred}_{\mathbf{x}}^{\mathcal{W}}(x) \geq \beta$. Follows from (B) and the bijectivity of \mathbf{f} , which imply that

$$\text{Pred}[\mathcal{W}]_{s, h_j}^{\mathbf{x}}(x) \text{ and } \text{Pred}[\mathcal{W}]_{s', h'_j}^{\mathbf{x}}(x) \text{ have the same cardinality.}$$

case: $\varphi = \text{self}_{\mathbf{x}}^{\mathcal{W}} \geq \beta$. Follows directly from (C) and the bijectivity of \mathbf{f} .

case: $\varphi = \text{rem}_{\mathbf{x}}^{\mathcal{W}} \geq \beta$. Follows directly from (D) and the bijectivity of \mathbf{f} .

case: $\varphi = u = t$. Follows directly from (A)(a) and since $\mathbf{f}(s(u)) = s'(u)$ (property (1_f) of \mathbf{f}).

case: $\varphi = u \in \text{pred}_{\mathbf{x}}^{\mathcal{W}}(x)$. Follows directly from (B) and $\mathbf{f}(s(u)) = s'(u)$.

case: $\varphi = u \in \text{self}_{\mathbf{x}}^{\mathcal{W}}$. Follows directly from (C) and $\mathbf{f}(s(u)) = s'(u)$.

case: $\varphi = u \in \text{rem}_{\mathbf{x}}^{\mathcal{W}}$. Follows directly from (D) and $\mathbf{f}(s(u)) = s'(u)$.

Therefore, $(s, h) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s', h')$. □

In the game hopping proof of the $*$ -simulation property, the lemma we just proved is used as a base case to treat the last hop of the chain of hops, i.e. $(s_{k-1}, h_{k-1}) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s_k, h_k) = (s', h')$ in the example above. Referring to this example, all the other hops connecting (s, h) to (s_{k-1}, h_{k-1}) are taken care of by three intermediate results, one for each type of sets between predecessor sets, self-loop sets, or remainder sets. The idea is that in every intermediate hop $(s_j, h_j) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s_{j+1}, h_{j+1})$, the memory state (s_{j+1}, h_{j+1}) is obtained from (s_j, h_j) by slightly updating one of these sets. For instance, we could decide to modify the locations in the set $\text{Rem}[\mathcal{W}]_{s_j, h_j}^{\mathbf{x}}$, while being careful that the resulting memory state satisfies the same core formulae of (s_j, h_j) . This idea is formalised in the following lemma.

Lemma 5.19. $(s, h) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s, h')$ holds if $(s, h) \approx_{\mathbf{x}, \alpha}^{\mathcal{W}} (s, h')$ and one of the following holds:

- (I) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[\mathcal{W}]_{s, h}^{\mathbf{x}}\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{W}]_{s, h'}^{\mathbf{x}}\}$,
- (II) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Self}[\mathcal{W}]_{s, h}^{\mathbf{x}}\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Self}[\mathcal{W}]_{s, h'}^{\mathbf{x}}\}$,
- (III) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[\mathcal{W}]_{s, h}^{\mathbf{x}}(x)\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[\mathcal{W}]_{s, h'}^{\mathbf{x}}(x)\}$, for some $x \in \mathbf{X}$.

Informally, in Lemma 5.19 the heap h' is obtained by h by modifying the memory cells corresponding to the sets (I) $\text{Rem}[\mathcal{W}]_{s, h}^{\mathbf{x}}$, (II) $\text{Self}[\mathcal{W}]_{s, h}^{\mathbf{x}}$, or (III) $\text{Pred}[\mathcal{W}]_{s, h}^{\mathbf{x}}(x)$ (for some $x \in \mathbf{X}$). We ask this modification to be invariant with respect to the satisfaction of the core formulae, i.e. $(s, h) \approx_{\mathbf{x}, \alpha}^{\mathcal{W}} (s, h')$. The proofs of the three statements of Lemma 5.19 are all very similar, and they follow quite closely the proof of Lemma 5.18. In the following, we present the proof of Lemma 5.19(I). The proofs of the other two statements are given in Appendix C.

Proof of (I). Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Notice that this requires α to be at least two, otherwise the lemma trivially holds. We partition the set $\text{Rem}[\mathcal{W}]_{s, h'}^{\mathbf{x}}$ into two sets S_1 and S_2 , using the following case analysis:

```

if card( $\text{Rem}[\mathcal{W}]_{s, h}^{\mathbf{x}} \cap \text{dom}(h_1)$ ) <  $\alpha_1$  then
  let  $S_1$  be a set of  $\text{card}(\text{Rem}[\mathcal{W}]_{s, h}^{\mathbf{x}} \cap \text{dom}(h_1))$  locations in  $\text{Rem}[\mathcal{W}]_{s, h'}^{\mathbf{x}}$ 
  such that  $s(u) \in S_1$  if and only if  $s(u) \in \text{Rem}[\mathcal{W}]_{s, h}^{\mathbf{x}} \cap \text{dom}(h_1)$ .

```

```

 $S_2 \leftarrow \text{Rem}[\mathcal{W}]_{s,h'}^X \setminus S_1.$ 
else if  $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2)) < \alpha_2$  then
  let  $S_2$  be a set of  $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2))$  locations in  $\text{Rem}[\mathcal{W}]_{s,h'}^X$ 
    such that  $s(u) \in S_2$  if and only if  $s(u) \in \text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2)$ .
   $S_1 \leftarrow \text{Rem}[\mathcal{W}]_{s,h'}^X \setminus S_2.$ 
else (i.e.  $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_1)) \geq \alpha_1$  and  $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2)) \geq \alpha_2$ )
  let  $S_1$  be a set of  $\alpha_1$  locations in  $\text{Rem}[\mathcal{W}]_{s,h'}^X$ 
    such that  $s(u) \in S_1$  if and only if  $s(u) \in \text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_1)$ .
   $S_2 \leftarrow \text{Rem}[\mathcal{W}]_{s,h'}^X \setminus S_1.$ 

```

Notice that S_1 and S_2 are always well-defined, since both (s, h) and (s, h') satisfy the same formulae among $u \in \text{rem}_X^{\mathcal{W}}$ and $\text{rem}_X^{\mathcal{W}} \geq \beta$, for every $\beta \in [1, \alpha]$. Indeed, thanks to the formula $u \in \text{rem}_X^{\mathcal{W}}$, if $s(u) \in \text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j)$ (where $j \in \{1, 2\}$) then $s(u) \in \text{Rem}[\mathcal{W}]_{s,h'}^X$ and so $s(u)$ can be selected when building S_j . From the formulae of the form $\text{rem}_X^{\mathcal{W}} \geq \beta$, if $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j)) < \alpha_j$ then, as $\alpha_j < \alpha$ we conclude that $\text{Rem}[\mathcal{W}]_{s,h'}^X$ contains at least $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j))$ locations, allowing us to correctly define S_j in the first two cases above. If instead $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_1)) \geq \alpha_1$ and $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2)) \geq \alpha_2$, then we conclude that both $\text{Rem}[\mathcal{W}]_{s,h}^X$ and $\text{Rem}[\mathcal{W}]_{s,h'}^X$ contain at least $\alpha > \alpha_1$ locations. Again, this allows us to correctly define S_1 in the last of the cases above. S_1 and S_2 enjoy the following properties, given with respect to $j \in \{1, 2\}$.

1. $s(u) \in S_j$ if and only if $s(u) \in \text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j)$,
2. $\min(\alpha_j, \text{card}(S_j)) = \min(\alpha_j, \text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j)))$.

Proof of (1). From the equisatisfaction of $u \in \text{rem}_X^{\mathcal{W}}$, $s(u) \in \text{Rem}[\mathcal{W}]_{s,h}^X$ iff $s(u) \in \text{Rem}[\mathcal{W}]_{s,h'}^X$.

Then, the property follows from the definition of S_1 and S_2 .

Proof of (2). From the equisatisfaction of $\text{rem}_X^{\mathcal{W}} \geq \beta$, for every $\beta \in [1, \alpha]$, it holds that

$$\min(\alpha, \text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X)) = \min(\alpha, \text{card}(\text{Rem}[\mathcal{W}]_{s,h'}^X)).$$

First, suppose $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X) = \text{card}(\text{Rem}[\mathcal{W}]_{s,h'}^X) < \alpha$. From $\alpha_1 + \alpha_2 = \alpha$, the third case in the definition of S_1 and S_2 cannot hold. The other two cases lead to $\text{card}(S_j) = \text{card}(\text{Rem}[\mathcal{W}]_{s,h_j}^X)$, for both $j \in \{1, 2\}$. Instead, suppose that both $\text{Rem}[\mathcal{W}]_{s,h}^X$ and $\text{Rem}[\mathcal{W}]_{s,h'}^X$ have at least α elements. We distinguish three cases:

- Suppose $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_1)) < \alpha_1$. As $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X) \geq \alpha$ and $\alpha = \alpha_1 + \alpha_2$, we conclude that $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2)) \geq \alpha_2$. The first case in the definition of S_1 and S_2 applies, so that $\text{card}(S_1) = \text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_1))$. Thus, $\text{card}(S_1) < \alpha_1$. From $\text{card}(\text{Rem}[\mathcal{W}]_{s,h'}^X) \geq \alpha$ and $\alpha = \alpha_1 + \alpha_2$, $\text{card}(S_2) \geq \alpha_2$.
- Suppose $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2)) < \alpha_2$. As $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X) \geq \alpha$ and $\alpha = \alpha_1 + \alpha_2$, we conclude that $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_1)) \geq \alpha_1$. The second case in the definition of S_1 and S_2 applies, so that $\text{card}(S_2) = \text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2))$. Thus, $\text{card}(S_2) < \alpha_2$. From $\text{card}(\text{Rem}[\mathcal{W}]_{s,h'}^X) \geq \alpha$ and $\alpha = \alpha_1 + \alpha_2$, $\text{card}(S_1) \geq \alpha_1$.
- Suppose $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_1)) \geq \alpha_1$ and $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_2)) \geq \alpha_2$. Then, the third case in the definition of S_1 and S_2 applies, so that $\text{card}(S_1) = \alpha_1$. Lastly, from $\text{card}(\text{Rem}[\mathcal{W}]_{s,h'}^X) \geq \alpha$ and $\alpha = \alpha_1 + \alpha_2$, $\text{card}(S_2) \geq \alpha_2$.

We rely on S_1 and S_2 in order to define the heaps h'_1 and h'_2 such that $(s, h_1) \approx_{X,\alpha_1}^W (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^W (s, h'_2)$ (as required by the hop relation $\leftrightarrow_{X,\alpha}^W$). First, let us define the two heaps $\widehat{h}_1 \stackrel{\text{def}}{=} h_1 \setminus \{(\ell, \ell') \in h_1 \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h}^X\}$ and $\widehat{h}_2 \stackrel{\text{def}}{=} h_2 \setminus \{(\ell, \ell') \in h_2 \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h}^X\}$, obtained from h_1 and h_2 by removing the locations in $\text{Rem}[\mathcal{W}]_{s,h}^X$ from their domain. Therefore, from $h = h_1 + h_2$ we conclude that:

$$h = \widehat{h}_1 + \widehat{h}_2 + \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h}^X\}.$$

Thus, from the hypothesis $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h}^X\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h'}^X\}$ we derive $\widehat{h}_1 + \widehat{h}_2 = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h'}^X\}$. We define the heaps h'_1 and h'_2 as:

$$h'_1 \stackrel{\text{def}}{=} \widehat{h}_1 + \{(\ell, \ell') \in h' \mid \ell \in S_1\}, \quad h'_2 \stackrel{\text{def}}{=} \widehat{h}_2 + \{(\ell, \ell') \in h' \mid \ell \in S_2\}.$$

As $\{(\ell, \ell') \in h' \mid \ell \in S_1\} + \{(\ell, \ell') \in h' \mid \ell \in S_2\} = \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h'}^X\}$ by definition of S_1 and S_2 , the two heaps h'_1 and h'_2 are well-defined, they are disjoint, and $h' = h'_1 + h'_2$. Moreover, $S_1 = \text{Rem}[\mathcal{W}]_{s,h'}^X \cap \text{dom}(h'_1)$ and $S_2 = \text{Rem}[\mathcal{W}]_{s,h'}^X \cap \text{dom}(h'_2)$.

We now prove four properties of h'_1 and h'_2 that are analogous to the properties (A)–(D) in the proof of Lemma 5.18. Let $j \in \{1, 2\}$.

- A. (a) for every $t \in T[\mathcal{W}]^X$, $\llbracket t \rrbracket_{s,h_j}^X$ is defined iff so is $\llbracket t \rrbracket_{s,h'_j}^X$. If defined, $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h'_j}^X$.
- (b) $\llbracket t \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$ if and only if $\llbracket t \rrbracket_{s,h'_j}^X \in \text{dom}(h'_j)$.
- (c) Given $t' \in X \cup \{t\}$, we have $h_j(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t' \rrbracket_{s,h_j}^X$ if and only if $h'_j(\llbracket t \rrbracket_{s,h'_j}^X) = \llbracket t' \rrbracket_{s,h'_j}^X$.
- (d) $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$ if and only if $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X$.

These four statements follow primarily from the fact that \widehat{h}_j is a subheap of both h_j and h'_j . In the following we show the left-to-right direction for each of these statements. The right-to-left direction follows analogously.

Proof of (a). Obvious for $t \in X$, so suppose $t = n(x) \in NV[\mathcal{W}]^X$.

(\Rightarrow): Suppose $\llbracket n(x) \rrbracket_{s,h_j}^X$ to be defined, so $\llbracket n(x) \rrbracket_{s,h_j}^X \stackrel{\text{by def}}{=} h_j(s(x))$. As $s(x) \in \text{Lab}[\mathcal{W}]_{s,h}^X$, $s(x) \notin \text{Rem}[\mathcal{W}]_{s,h}^X$, and therefore $s(x) \in \text{dom}(\widehat{h}_j)$ and $\widehat{h}_j(s(x)) = h_j(s(x))$. By definition of h'_j , $\widehat{h}_j(s(x)) = h'_j(s(x))$. Thus, $\llbracket n(x) \rrbracket_{s,h_j}^X = \llbracket n(x) \rrbracket_{s,h'_j}^X$.

Proof of (b). (\Rightarrow): Suppose $\llbracket t \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$. By Lemma 5.14(I), $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h}^X$ and thus $\llbracket t \rrbracket_{s,h_j}^X \notin \text{Rem}[\mathcal{W}]_{s,h}^X$. Therefore, $\llbracket t \rrbracket_{s,h_j}^X \in \text{dom}(\widehat{h}_j)$. From (a), $\llbracket t \rrbracket_{s,h'_j}^X = \llbracket t \rrbracket_{s,h_j}^X$, which in turn implies $\llbracket t \rrbracket_{s,h'_j}^X \in \text{dom}(\widehat{h}_j) \subseteq \text{dom}(h'_j)$.

Proof of (c). (\Rightarrow): $h_j(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t' \rrbracket_{s,h_j}^X$ and therefore by (a) we have $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h'_j}^X$ and $\llbracket t' \rrbracket_{s,h_j}^X = \llbracket t' \rrbracket_{s,h'_j}^X$. As done in the proof of (b), we conclude that $\llbracket t \rrbracket_{s,h_j}^X \in \text{dom}(\widehat{h}_j)$. By $\widehat{h}_j \subseteq h_j$, $\widehat{h}_j(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t' \rrbracket_{s,h_j}^X$. By $\widehat{h}_j \subseteq h'_j$, $h'_j(\llbracket t \rrbracket_{s,h'_j}^X) = \llbracket t' \rrbracket_{s,h'_j}^X$.

Proof of (d). (\Rightarrow): Assume $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$. By definition, it cannot be that ℓ is assigned to a program variable in X , as otherwise $\ell \in \text{Lab}[\mathcal{W}]_{s,h_j}^X$. So, there is a next-point variable $n(x)$ such that $\llbracket n(x) \rrbracket_{s,h}^X = \ell$. From $s(x) \in \text{Lab}[\mathcal{W}]_{s,h}^X$, we derive that $s(x) \notin \text{Rem}[\mathcal{W}]_{s,h}^X$ and therefore $s(x) \in \text{dom}(h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h}^X\})$. From $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h}^X\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{W}]_{s,h'}^X\}$, we conclude that $h'(s(x)) = \ell$ and so $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X$. *Ad absurdum*, suppose $\ell \in \text{Lab}[\mathcal{W}]_{s,h'_j}^X$. From (a) we have $\ell \in \text{Lab}[\mathcal{W}]_{s,h_j}^X$, contradicting $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$. Thus, $\ell \notin \text{Lab}[\mathcal{W}]_{s,h'_j}^X$ and therefore $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X$.

B. For every $x \in X$, $\text{Pred}[\mathcal{W}]_{s,h'_j}^X(x) = \text{Pred}[\mathcal{W}]_{s,h_j}^X(x)$.

We show left-to-right direction. Thanks to \widehat{h}_j , the right-to-left direction is analogous.

Proof of (B). (\Rightarrow): Suppose $\ell \in \text{Pred}[\mathcal{W}]_{s,h'_j}^X(x)$. By definition, $\ell \notin \text{Lab}[\mathcal{W}]_{s,h'_j}^X$ and $h'_j(\ell) = s(x)$. From (a), $\ell \notin \text{Lab}[\mathcal{W}]_{s,h_j}^X$. From $h'_j \subseteq h'$, $h'(\ell) = s(x)$ and therefore it cannot be that ℓ belongs to $\text{Rem}[\mathcal{W}]_{s,h'}^X$. By definition of h'_j , $\ell \in \text{dom}(\widehat{h}_j)$. From $\widehat{h}_j \subseteq h'_j$, $\widehat{h}_j(\ell) = s(x)$. From $\widehat{h}_j \subseteq h_j$, $h_j(\ell) = s(x)$. Together with $\ell \notin \text{Lab}[\mathcal{W}]_{s,h_j}^X$, this implies $\ell \in \text{Pred}[\mathcal{W}]_{s,h_j}^X(x)$,

C. $\text{Self}[\mathcal{W}]_{s,h'_j}^X = \text{Self}[\mathcal{W}]_{s,h_j}^X$.

We show left-to-right direction. Thanks to \widehat{h}_j , the right-to-left direction is analogous.

Proof of (C). (\Rightarrow): Suppose $\ell \in \text{Self}[\mathcal{W}]_{s,h'_j}^X$. By definition, $\ell \notin \text{Lab}[\mathcal{W}]_{s,h'_j}^X$ and $h'_j(\ell) = \ell$.

From (a), $\ell \notin \text{Lab}[\mathcal{W}]_{s,h_j}^X$. From $h'_j \subseteq h'$, $h'(\ell) = \ell$ and therefore it cannot be that ℓ belongs to $\text{Rem}[\mathcal{W}]_{s,h'}^X$. By definition of h'_j , $\ell \in \text{dom}(\widehat{h}_j)$. From $\widehat{h}_j \subseteq h'_j$, $\widehat{h}_j(\ell) = \ell$.

From $\widehat{h}_j \subseteq h_j$, $h_j(\ell) = \ell$. Together with $\ell \notin \text{Lab}[\mathcal{W}]_{s,h_j}^X$, this implies $\ell \in \text{Self}[\mathcal{W}]_{s,h_j}^X$,

D. $\min(\alpha_j, \text{card}(\text{Rem}[\mathcal{W}]_{s,h'_j}^X)) = \min(\alpha_j, \text{card}(\text{Rem}[\mathcal{W}]_{s,h_j}^X))$.

Proof of (D). From Lemma 5.14(IV) we have:

$$\begin{aligned} \text{Rem}[\mathcal{W}]_{s,h'_j}^X &= (\text{Rem}[\mathcal{W}]_{s,h'}^X \cap \text{dom}(h'_j)) \cup \\ &\quad \{\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X \mid \ell \in \text{dom}(h'_j), h'_j(\ell) \neq \ell \text{ and } \forall x \in X, h'_j(\ell) \neq s(x)\}, \\ \text{Rem}[\mathcal{W}]_{s,h_j}^X &= (\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j)) \cup \\ &\quad \{\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X \mid \ell \in \text{dom}(h_j), h_j(\ell) \neq \ell \text{ and } \forall x \in X, h_j(\ell) \neq s(x)\}. \end{aligned}$$

By definition of h'_j , $\text{Rem}[\mathcal{W}]_{s,h'}^X \cap \text{dom}(h'_j) = S_j$. By (2),

$$\min(\alpha_j, \text{card}(\text{Rem}[\mathcal{W}]_{s,h'}^X \cap \text{dom}(h'_j))) = \min(\alpha_j, \text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j))).$$

Thus, in order to prove (D) we only need to show that the two sets

$$\{\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X \mid \ell \in \text{dom}(h'_j), h'_j(\ell) \neq \ell \text{ and } \forall x \in X, h'_j(\ell) \neq s(x)\}$$

$$\text{and } \{\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X \mid \ell \in \text{dom}(h_j), h_j(\ell) \neq \ell \text{ and } \forall x \in X, h_j(\ell) \neq s(x)\}$$

are equivalent. This amounts to showing that, given a location $\ell \in \text{LOC}$,

e. $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X, \ell \in \text{dom}(h'_j), h'_j(\ell) \neq \ell \text{ and } \forall x \in X, h'_j(\ell) \neq s(x)$ iff $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X, \ell \in \text{dom}(h_j), h_j(\ell) \neq \ell \text{ and } \forall x \in X, h_j(\ell) \neq s(x)$.

Proof of (e). (\Rightarrow): Suppose $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X$ such that $\ell \in \text{dom}(h'_j)$, $h'_j(\ell) \neq \ell$ and for every $x \in X$, $h'_j(\ell) \neq s(x)$. From (A)(d), we derive $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$. From $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X$, we derive $\ell \notin \text{Rem}[\mathcal{W}]_{s,h'}^X$. So, by $\ell \in \text{dom}(h'_j)$ and definition of h'_j , we conclude that $\ell \in \text{dom}(\widehat{h}_j)$. Moreover, $\widehat{h}_j(\ell) \neq \ell$ and for every $x \in X$, $\widehat{h}_j(\ell) \neq s(x)$. By $\widehat{h}_j \subseteq h_j$, we derive that $\ell \in \text{dom}(h_j)$, $h_j(\ell) \neq \ell$ and for every $x \in X$, $h_j(\ell) \neq s(x)$. (\Leftarrow): Analogous to the other direction, again using (A)(d) and the definition of \widehat{h}_j .

Thanks to the properties (A)–(D), proving $(s, h_j) \approx_{X,\alpha_j}^{\mathcal{W}} (s', h'_j)$, for $j \in \{1, 2\}$, is straightforward. Consider a core formula φ in $\text{Core}[\mathcal{W}](X, \alpha_j)$. Then, $(s, h_j) \models \varphi$ iff $(s, h'_j) \models \varphi$, as shown below:

case: $\varphi = t_1 = t_2$. Follows directly from (A)(a).

case: $\varphi = t \hookrightarrow __$. Follows directly from (A)(b).

- case:** $\varphi = t \hookrightarrow x$ or $\varphi = t \hookrightarrow t$. Follows directly from (A)(c).
- case:** $\varphi = \text{pred}_x^W(x) \geq \beta$. Follows directly from (B).
- case:** $\varphi = \text{self}_x^W \geq \beta$. Follows directly from (C).
- case:** $\varphi = \text{rem}_x^W \geq \beta$. Follows directly from (D).
- case:** $\varphi = u = t$. Follows directly from (A)(a), since (s, h_j) and (s, h_j) share the same store.
- case:** $\varphi = u \in \text{pred}_x^W(x)$. Follows directly from (B).
- case:** $\varphi = u \in \text{self}_x^W$. Follows directly from (C).
- case:** $\varphi = u \in \text{rem}_x^W$. Since $S_j = \text{Rem}[W]_{s,h}^X \cap \text{dom}(h'_j)$, it follows from (1) and (D)(e). Therefore, $(s, h) \leftrightarrow_{X,\alpha}^W (s, h')$. \square

Strong of Lemma 5.19, we are ready to prove the $*$ -simulation property for the weak fragment.

Lemma 5.20 (w : $*$ -simulation). $\approx_{X,\alpha}^W \subseteq \leftrightarrow_{X,\alpha}^W$.

Proof. Let us consider (s, h) and (s', h') such that $(s, h) \approx_{X,\alpha}^W (s', h')$. We build a chain of hops as the one below, leading to the result by transitivity of $\leftrightarrow_{X,\alpha}^W$ and Lemma 5.18,

$$(s, h) = (s_1, h_1) \leftrightarrow_{X,\alpha}^W (s_2, h_2) \leftrightarrow_{X,\alpha}^W \dots \leftrightarrow_{X,\alpha}^W (s_{k-1}, h_{k-1}) \leftrightarrow_{X,\alpha}^W (s_k, h_k) = (s', h').$$

The proof is by induction on the cardinality of the set $[(s, h) \#_X (s', h')]$ defined as follows:

$$\left\{ (S, T) \in \left\{ \begin{array}{l} (\text{Rem}[W]_{s,h}^X, \text{Rem}[W]_{s',h'}^X), (\text{Self}[W]_{s,h}^X, \text{Self}[W]_{s',h'}^X) \\ (\text{Pred}[W]_{s,h}^X(x), \text{Pred}[W]_{s',h'}^X(x)) \end{array} \middle| x \in X \right\} \mid \text{card}(S) \neq \text{card}(T) \right\}$$

Intuitively, this set contains pairs of predecessors sets, self-loops sets, or remainder sets that have different cardinalities in the two memory states. We build the chain of hops so that for every intermediate memory state in the chain is obtained from the previous one by modifying the heap in a way that strictly reduces the number of these pairs, always with respect to the last memory state of the chain, i.e. (s', h') .

base case: $[(s, h) \#_X (s', h')] = 0$. Follows by Lemma 5.18, as $((s, h), (s', h')) \in (\bigcap_{\alpha' \geq 1} \approx_{X,\alpha'}^W)$.

induction step: $[(s, h) \#_X (s', h')] > 0$. Let $(S, T) \in [(s, h) \#_X (s', h')]$. We split the proof in three cases, all of them quite similar, dealing with the different types of sets S and T .

case $(S, T) = (\text{Rem}[W]_{s,h}^X, \text{Rem}[W]_{s',h'}^X)$. Let us assume that $\text{card}(S) > \text{card}(T)$. Notice that this assumption is without loss of generality: in the case where $\text{card}(S) < \text{card}(T)$, it is sufficient to swap (s, h) and (s', h') in the proof, and apply the construction we now show to produce a chain of hops going from (s', h') to (s, h) , i.e.

$$(s', h') = (s_1, h_1) \leftrightarrow_{X,\alpha}^W (s_2, h_2) \leftrightarrow_{X,\alpha}^W \dots \leftrightarrow_{X,\alpha}^W (s_{k-1}, h_{k-1}) \leftrightarrow_{X,\alpha}^W (s_k, h_k) = (s, h).$$

So, assuming $\text{card}(S) > \text{card}(T)$, consider the heap h'' obtained from h by removing from its domain $\text{card}(S) - \text{card}(T)$ locations in $\text{Rem}[W]_{s,h}^X$ and different from $s(u)$. Formally, $h'' \subseteq h$ and there is a set $Q \subseteq \text{Rem}[W]_{s,h}^X$ such that $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, $\text{dom}(h'') = \text{dom}(h) \setminus Q$ and $s(u) \notin Q$. Notice that a heap h'' satisfying these conditions exists. In particular, as $\text{card}(S) \neq \text{card}(T)$ and $(s, h) \approx_{X,\alpha}^W (s', h')$, both memory states must satisfy $\text{rem}_x^W \geq \alpha$, where α is assumed strictly positive. So, both S and T have at least $\alpha \geq 1$ elements, which allows us to keep $s(u)$ in the domain of h'' , in the case it belongs to $\text{dom}(h)$. We derive four properties of (s, h) and (s, h'') :

- for every $t \in T[W]^X$, $\llbracket t \rrbracket_{s,h}^X$ is defined iff so is $\llbracket t \rrbracket_{s,h''}^X$. When defined, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h''}^X$, and if $\llbracket t \rrbracket_{s,h}^X \in \text{dom}(h)$, then $h''(\llbracket t \rrbracket_{s,h''}^X) = h(\llbracket t \rrbracket_{s,h}^X)$.

Proof. Let us first show that for every $t \in T[\mathcal{W}]^X$, $\llbracket t \rrbracket_{s,h}^X$ is defined iff so is $\llbracket t \rrbracket_{s,h''}^X$.

This statement is obvious for $t \in X$, so suppose $t = n(x) \in NV[\mathcal{W}]^X$. The right-to-left direction follows directly from Lemma 5.14(I). For the left-to-right direction, suppose $\llbracket n(x) \rrbracket_{s,h}^X$ to be defined, i.e. $\llbracket n(x) \rrbracket_{s,h}^X \stackrel{\text{by def}}{=} h(s(x))$. Since $s(x) \in Lab[\mathcal{W}]_{s,h}^X$, it cannot be that $s(x) \in Rem[\mathcal{W}]_{s,h}^X$, which allows us to conclude that $h(s(x)) = h''(s(x))$. From $\llbracket n(x) \rrbracket_{s,h''}^X \stackrel{\text{by def}}{=} h''(s(x))$ we derive $\llbracket n(x) \rrbracket_{s,h''}^X = \llbracket n(x) \rrbracket_{s,h}^X$. Notice that this proves the second statement, i.e. $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h''}^X$, for the case of next-point variables (the case of program variables being obvious). Let us now show that if $\llbracket t \rrbracket_{s,h}^X \in \text{dom}(h)$, then $h''(\llbracket t \rrbracket_{s,h}^X) = h(\llbracket t \rrbracket_{s,h}^X)$. Suppose $\llbracket t \rrbracket_{s,h}^X \in \text{dom}(h)$. Since $\llbracket t \rrbracket_{s,h}^X \in Lab[\mathcal{W}]_{s,h}^X$, we conclude that $\llbracket t \rrbracket_{s,h}^X \notin Rem[\mathcal{W}]_{s,h}^X$. By definition of h'' , $\llbracket t \rrbracket_{s,h}^X \in \text{dom}(h'')$. From $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h''}^X$ and $h'' \subseteq h$ we conclude: $h''(\llbracket t \rrbracket_{s,h''}^X) = h(\llbracket t \rrbracket_{s,h}^X)$.

2. for every $x \in X$, $\text{Pred}[\mathcal{W}]_{s,h''}^X(x) = \text{Pred}[\mathcal{W}]_{s,h}^X(x)$.

Proof. Directly from the property (1) above and Lemma 5.14(II). Indeed, (1) implies $Lab[\mathcal{W}]_{s,h}^X = Lab[\mathcal{W}]_{s,h''}^X$, so that the equivalence in Lemma 5.14(II) becomes $\text{Pred}[\mathcal{W}]_{s,h''}^X(x) = \text{Pred}[\mathcal{W}]_{s,h}^X(x) \cap \text{dom}(h'')$. Lastly, by definition of h'' , $\text{Pred}[\mathcal{W}]_{s,h}^X(x) \cap \text{dom}(h'') = \text{Pred}[\mathcal{W}]_{s,h}^X(x)$.

3. $\text{Self}[\mathcal{W}]_{s,h''}^X = \text{Self}[\mathcal{W}]_{s,h}^X$.

Proof. Directly from the property (1) and Lemma 5.14(III).

4. $Rem[\mathcal{W}]_{s,h''}^X \subseteq Rem[\mathcal{W}]_{s,h}^X$.

Proof. Directly from the property (1) and Lemma 5.14(IV).

Thanks to these four properties, we conclude that $Q \cup Rem[\mathcal{W}]_{s,h''}^X = Rem[\mathcal{W}]_{s,h}^X$, and so from $\text{card}(Q) = \text{card}(S) - \text{card}(T)$ we conclude that $\text{card}(Rem[\mathcal{W}]_{s,h''}^X) = \text{card}(T)$. We now show that $(s, h) \approx_{X,\alpha}^W (s, h'')$, $(s, h) \leftrightarrow_{X,\alpha}^W (s, h'')$, and $(s, h'') \leftrightarrow_{X,\alpha}^W (s', h')$.

Proof of $(s, h) \approx_{X,\alpha}^W (s, h'')$. Thanks to the property (1), (s, h) and (s, h'') satisfy the same core formulae of the form $t_1 = t_2$, $t_1 \hookrightarrow _$, $t_1 \hookrightarrow x$, $t_1 \hookrightarrow t_1$ and $u = t_1$. Thanks to the property (2), given $x \in X$, (s, h) and (s, h'') satisfy the same core formulae of the form $\text{pred}_X^W(x) \geq \beta$, for every $\beta \in [1, \alpha]$, and they equisatisfy the core formula $u \in \text{pred}_X^W(x)$. Thanks to the property (3), (s, h) and (s, h'') satisfy the same core formulae of the form $\text{self}_X^W \geq \beta$, for every $\beta \in [1, \alpha]$, and they equisatisfy the core formula $u \in \text{self}_X^W$. From $\text{card}(Rem[\mathcal{W}]_{s,h''}^X) = \text{card}(T) \geq \alpha$ and $\text{card}(S) \geq \alpha$, we conclude that (s, h) and (s, h'') satisfy the same core formulae of the form $\text{rem}_X^W \geq \beta$, for every $\beta \in [1, \alpha]$. Since h'' is constructed so that $s(u)$ is kept in $\text{dom}(h'')$ if it belongs to $Rem[\mathcal{W}]_{s,h}^X$, we also conclude that the two memory states (s, h) and (s, h'') equisatisfy the formula $u \in \text{rem}_X^W$.

Proof of $(s, h) \leftrightarrow_{X,\alpha}^W (s, h'')$. Directly from the properties (1)–(4), we conclude that $h \setminus \{(\ell, \ell') \in h \mid \ell \in Rem[\mathcal{W}]_{s,h}^X\} = h'' \setminus \{(\ell, \ell') \in h'' \mid \ell \in Rem[\mathcal{W}]_{s,h''}^X\}$. Thanks to $(s, h) \approx_{X,\alpha}^W (s, h'')$, we apply Lemma 5.19(I) and derive that $(s, h) \leftrightarrow_{X,\alpha}^W (s, h'')$.

Proof of $(s, h'') \leftrightarrow_{X,\alpha}^W (s', h')$. As $\approx_{X,\alpha}^W$ is an equivalence relation, $(s, h) \approx_{X,\alpha}^W (s, h'')$ allows us to derive that $(s, h'') \approx_{X,\alpha}^W (s', h')$. From the properties (1)–(4), together with $\text{card}(Rem[\mathcal{W}]_{s,h''}^X) = \text{card}(T)$, we derive $[(s, h'') \#_X (s', h')] < [(s, h) \#_X (s', h')]$. By induction hypothesis, $(s, h'') \leftrightarrow_{X,\alpha}^W (s', h')$.

From $(s, h) \leftrightarrow_{X,\alpha}^W (s, h'')$, $(s, h'') \leftrightarrow_{X,\alpha}^W (s', h')$ and by transitivity of the hop relation $\leftrightarrow_{X,\alpha}^W$ we conclude: $(s, h) \leftrightarrow_{X,\alpha}^W (s', h')$.

case $(S, T) = (\text{Self}[\mathcal{W}]_{s,h}^X, \text{Self}[\mathcal{W}]_{s',h'}^X)$. As in the previous case, without loss of generality we can assume $\text{card}(S) > \text{card}(T)$. We consider the heap h'' obtained from h by removing from its domain $\text{card}(S) - \text{card}(T)$ locations in $\text{Self}[\mathcal{W}]_{s,h}^X$, all different from $s(u)$. Formally, $h'' \subseteq h$ and there is a set $Q \subseteq \text{Self}[\mathcal{W}]_{s,h}^X$ s.t. $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, $\text{dom}(h'') = \text{dom}(h) \setminus Q$ and $s(u) \notin Q$. Notice that a heap h'' satisfying these conditions exists. In particular, as $\text{card}(S) \neq \text{card}(T)$ and $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s', h')$, both memory states must satisfy $\text{self}_X^{\mathcal{W}} \geq \alpha$, where α is assumed strictly positive. So, both S and T have at least $\alpha \geq 1$ elements, which allows us to keep $s(u)$ in the domain of h'' , in the case it belongs to $\text{dom}(h)$. (s, h) and (s, h'') enjoy the following four properties:

1. for every $t \in T[\mathcal{W}]^X$, $\llbracket t \rrbracket_{s,h}^X$ is defined iff so is $\llbracket t \rrbracket_{s,h''}^X$. When defined, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h''}^X$, and if $\llbracket t \rrbracket_{s,h}^X \in \text{dom}(h)$, then $h''(\llbracket t \rrbracket_{s,h}^X) = h(\llbracket t \rrbracket_{s,h}^X)$,
2. for every $x \in X$, $\text{Pred}[\mathcal{W}]_{s,h''}^X(x) = \text{Pred}[\mathcal{W}]_{s,h}^X(x)$,
3. $\text{Rem}[\mathcal{W}]_{s,h''}^X = \text{Rem}[\mathcal{W}]_{s,h}^X$,
4. $\text{Self}[\mathcal{W}]_{s,h''}^X \subseteq \text{Self}[\mathcal{W}]_{s,h}^X$.

The first property is proven as the analogous property in the previous case of the proof, whereas the other three properties follow by Lemma 5.14. Thanks to (1)–(4), we derive that $Q \cup \text{Self}[\mathcal{W}]_{s,h''}^X = \text{Self}[\mathcal{W}]_{s,h}^X$, and so, by $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, we conclude that $\text{card}(\text{Self}[\mathcal{W}]_{s,h''}^X) = \text{card}(T)$. As done in the previous step, we show that $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s, h'')$, $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s, h'')$, and $(s, h'') \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s', h')$. By transitivity of the hop relation $\leftrightarrow_{X,\alpha}^{\mathcal{W}}$, the last two relationships imply $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s', h')$.

Proof of $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s, h'')$. From (1), (s, h) and (s, h'') satisfy the same core formulae of the form $t_1 = t_2$, $t_1 \hookrightarrow _$, $t_1 \hookrightarrow x$, $t_1 \hookrightarrow t_1$ and $u = t_1$. From (2), given $x \in X$ and $\beta \in [1, \alpha]$, (s, h) and (s, h'') equisatisfy the core formula $\text{pred}_X^{\mathcal{W}}(x) \geq \beta$ and they equisatisfy the core formula $u \in \text{pred}_X^{\mathcal{W}}(x)$. From (3), (s, h) and (s, h'') satisfy the same core formulae of the form $\text{rem}_X^{\mathcal{W}} \geq \beta$, for all $\beta \in [1, \alpha]$, and they equisatisfy the core formula $u \in \text{rem}_X^{\mathcal{W}}$. By $\text{card}(\text{Self}[\mathcal{W}]_{s,h''}^X) = \text{card}(T) \geq \alpha$ and $\text{card}(S) \geq \alpha$, we conclude that (s, h) and (s, h'') satisfy the same core formulae of the form $\text{self}_X^{\mathcal{W}} \geq \beta$, for every $\beta \in [1, \alpha]$. Since h'' is constructed so that $s(u)$ is kept in $\text{dom}(h'')$ if it belongs to $\text{Self}[\mathcal{W}]_{s,h}^X$, we also conclude that the two memory states (s, h) and (s, h'') equisatisfy the formula $u \in \text{self}_X^{\mathcal{W}}$.

Proof of $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s, h'')$. Directly from the properties (1)–(4), we conclude that $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Self}[\mathcal{W}]_{s,h}^X\} = h'' \setminus \{(\ell, \ell') \in h'' \mid \ell \in \text{Self}[\mathcal{W}]_{s,h''}^X\}$. Thanks to $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s, h'')$, we apply Lemma 5.19(II) and derive that $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s, h'')$.

Proof of $(s, h'') \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s', h')$. As $\approx_{X,\alpha}^{\mathcal{W}}$ is an equivalence relation, $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s, h'')$ allows us to derive that $(s, h'') \approx_{X,\alpha}^{\mathcal{W}} (s', h')$. From the equivalences (1)–(4), and $\text{card}(\text{Self}[\mathcal{W}]_{s,h''}^X) = \text{card}(T)$, we derive $[(s, h'') \#_X (s', h')] < [(s, h) \#_X (s', h')]$. By induction hypothesis, $(s, h'') \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s', h')$.

case $(S, T) = (\text{Pred}[\mathcal{W}]_{s,h}^X(x), \text{Pred}[\mathcal{W}]_{s',h'}^X(x))$, for some $x \in X$. As previously done, without loss of generality we assume $\text{card}(S) > \text{card}(T)$. Consider the heap h'' obtained from h by removing from its domain $\text{card}(S) - \text{card}(T)$ locations in $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$ and different from $s(u)$. Formally, $h'' \subseteq h$ and there is a set $Q \subseteq \text{Pred}[\mathcal{W}]_{s,h}^X(x)$ such that $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, $\text{dom}(h'') = \text{dom}(h) \setminus Q$ and $s(u) \notin Q$. Notice that a heap h'' satisfying these conditions exists. In particular, as $\text{card}(S) \neq \text{card}(T)$ and $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s', h')$, both memory states must satisfy $\text{pred}_X^{\mathcal{W}}(x) \geq \alpha$, where α is assumed strictly positive. So, both S and T have at least $\alpha \geq 1$ elements, which

allows us to keep $s(u)$ in the domain of h'' , in the case it belongs to $\text{dom}(h)$. (s, h) and (s, h'') enjoy the following five properties:

1. for every $t \in T[\mathcal{W}]^X$, $\llbracket t \rrbracket_{s,h}^X$ is defined iff so is $\llbracket t \rrbracket_{s,h''}^X$. When defined, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h''}^X$, and if $\llbracket t \rrbracket_{s,h}^X \in \text{dom}(h)$, then $h''(\llbracket t \rrbracket_{s,h''}^X) = h(\llbracket t \rrbracket_{s,h}^X)$,
2. for every $y \in X$, if $s(x) \neq s(y)$ then $\text{Pred}[\mathcal{W}]_{s,h''}^X(y) = \text{Pred}[\mathcal{W}]_{s,h}^X(y)$,
3. $\text{Self}[\mathcal{W}]_{s,h''}^X = \text{Self}[\mathcal{W}]_{s,h}^X$,
4. $\text{Rem}[\mathcal{W}]_{s,h''}^X = \text{Rem}[\mathcal{W}]_{s,h}^X$,
5. $\text{Pred}[\mathcal{W}]_{s,h''}^X(x) \subseteq \text{Pred}[\mathcal{W}]_{s,h}^X(x)$.

Again, we omit the proofs of these properties, which are analogous to the properties in the previous two cases of the proof. Thanks to these properties, we conclude that $Q \cup \text{Pred}[\mathcal{W}]_{s,h''}^X(x) = \text{Pred}[\mathcal{W}]_{s,h}^X(x)$, and so from $\text{card}(Q) = \text{card}(S) - \text{card}(T)$ we conclude that $\text{card}(\text{Pred}[\mathcal{W}]_{s,h''}^X(x)) = \text{card}(T)$. We show that $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s, h'')$, $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s, h'')$, and $(s, h'') \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s', h')$. By transitivity of the hop relation $\leftrightarrow_{X,\alpha}^{\mathcal{W}}$, the last two relationships imply $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s', h')$, concluding the proof.

Proof of $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s, h'')$. Thanks to the property (1), (s, h) and (s, h'') satisfy the same core formulae of the form $t_1 = t_2$, $t_1 \hookrightarrow \perp$, $t_1 \hookrightarrow x$, $t_1 \hookrightarrow t_1$ and $u = t_1$. Thanks to the property (3), (s, h) and (s, h'') satisfy the same core formulae of the form $\text{self}_X^{\mathcal{W}} \geq \beta$, for every $\beta \in [1, \alpha]$, and they equisatisfy the core formula $u \in \text{self}_X^{\mathcal{W}}$. Thanks to the property (4), (s, h) and (s, h'') satisfy the same core formulae of the form $\text{rem}_X^{\mathcal{W}} \geq \beta$, for every $\beta \in [1, \alpha]$, and they equisatisfy the core formula $u \in \text{rem}_X^{\mathcal{W}}$. Given a variable $y \in X$ such that $s(y) \neq s(x)$, the property (2) insures that (s, h) and (s, h'') satisfy the same core formulae of the form $\text{pred}_X^{\mathcal{W}}(y) \geq \beta$, for every $\beta \in [1, \alpha]$, and they equisatisfy the formula $u \in \text{self}_X^{\mathcal{W}}(y)$. Consider a variable $y \in X$ such that $s(y) = s(x)$. By definition, $\text{Pred}[\mathcal{W}]_{s,h''}^X(y) = \text{Pred}[\mathcal{W}]_{s,h}^X(x)$ and $\text{Pred}[\mathcal{W}]_{s,h}^X(y) = \text{Pred}[\mathcal{W}]_{s,h}^X(x)$. So, from $\text{card}(\text{Pred}[\mathcal{W}]_{s,h''}^X) = \text{card}(T) \geq \alpha$ and $\text{card}(S) \geq \alpha$, we conclude that both $\text{card}(\text{Pred}[\mathcal{W}]_{s,h''}^X(y)) \geq \alpha$ and $\text{card}(\text{Pred}[\mathcal{W}]_{s,h}^X(y)) \geq \alpha$ holds. So, (s, h) and (s, h'') satisfy the same core formulae of the form $\text{pred}_X^{\mathcal{W}}(y) \geq \beta$, for all $\beta \in [1, \alpha]$. Lastly, as h'' is constructed so that $s(u)$ is kept in $\text{dom}(h'')$ if it belongs to $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$, we conclude that the two memory states (s, h) and (s, h'') equisatisfy the formula $u \in \text{pred}_X^{\mathcal{W}}(y)$.

Proof of $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s, h'')$. Directly from the properties (1)–(4), we conclude that $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[\mathcal{W}]_{s,h}^X(x)\} = h'' \setminus \{(\ell, \ell') \in h'' \mid \ell \in \text{Pred}[\mathcal{W}]_{s,h''}^X(x)\}$ holds. As $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s, h'')$, we apply Lemma 5.19(III) to derive $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s, h'')$.

Proof of $(s, h'') \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s', h')$. As $\approx_{X,\alpha}^{\mathcal{W}}$ is an equivalence relation, $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s, h'')$ allows us to derive $(s, h'') \approx_{X,\alpha}^{\mathcal{W}} (s', h')$. From the properties (1)–(5), together with $\text{card}(\text{Pred}[\mathcal{W}]_{s,h''}^X(x)) = \text{card}(T)$, we derive $[(s, h'') \#_X (s', h')] < [(s, h) \#_X (s', h')]$. By induction hypothesis, $(s, h'') \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s', h')$. \square

5.3.4 Step IV: \exists -simulation.

As the weak fragment involves first-order quantification, in order for the core formulae to characterise the expressive power of the logic they need to enjoy a *\exists -simulation property*. Similarly to the $*$ -simulation stated in Lemma 5.6, this property can be formalised by looking at the semantics of the existential quantification over the unique variable name u . Recall that, in a

formula of the form $\exists u \varphi$, the existential quantification essentially requires to update the location assigned to u so that the formula φ is satisfied. The \exists -simulation property states that, given $(s, h) \approx_{X,\alpha}^W (s', h')$, whenever we assign a location ℓ_1 to u through s , it is possible to find a location ℓ_2 such that $(s[u \leftarrow \ell_1], h) \approx_{X,\alpha}^W (s'[u \leftarrow \ell_2], h')$. Albeit this is enough to conclude that the core formulae effectively capture the expressiveness of the first-order quantification, in order to show that $SL([\exists]_1, *, [-*, \rightarrow^\dagger]^S_W)$ enjoys a polynomial small-heap property we want to restrict the space of locations that must be considered when selecting the location ℓ_2 . In order to do so, in the following and for the rest of the chapter we assume LOC to be the set of natural numbers.

Assumption 5.21. $LOC = \mathbb{N}$.

As the two sets are isomorphic this assumption is without loss of generality, and allows us to use arithmetic constraints directly on locations. We define the maximum value of a memory state.

Definition 5.22 (Maximum value). Consider a memory state (s, h) , and let $Y \subseteq VAR$. We write $\text{maxval}_Y(s, h)$ for the location $\max(\text{dom}(h) \cup \text{ran}(h) \cup s(Y))$, i.e. the *maximum value* among the locations assigned to a variable in Y or appearing in either the domain or range of the heap h .

The notion of maximum value allows us to restrict the possible choices for the location ℓ_2 in the \exists -simulation property to a location that is at most $\text{maxval}_X(s', h') + 1$. Taking into account this additional constraint, the \exists -simulation property is formalised as follows.

Lemma 5.23 (W : \exists -simulation). Suppose $(s, h) \approx_{X,\alpha}^W (s', h')$. For every location $\ell_1 \in LOC$ there is a location $\ell_2 \leq \text{maxval}_X(s', h') + 1$ such that $(s[u \leftarrow \ell_1], h) \approx_{X,\alpha}^W (s'[u \leftarrow \ell_2], h')$.

Proof. First of, we notice that the definition of predecessors sets, self-loops sets and remainder sets does not depend on the location assigned to the variable $u \notin X$. More precisely, for every memory state (\hat{s}, \hat{h}) and location $\hat{\ell}$ the following equivalences hold (where $x \in X$):

$$\begin{array}{lll} \text{Lab}[W]_{s,\hat{h}}^X & = & \text{Lab}[W]_{\hat{s}[u \leftarrow \hat{\ell}],\hat{h}}^X, \\ \text{Self}[W]_{s,\hat{h}}^X & = & \text{Self}[W]_{\hat{s}[u \leftarrow \hat{\ell}],\hat{h}}^X, \end{array} \quad \begin{array}{lll} \text{Pred}[W]_{s,\hat{h}}^X(x) & = & \text{Pred}[W]_{\hat{s}[u \leftarrow \hat{\ell}],\hat{h}}^X(x), \\ \text{Rem}[W]_{s,\hat{h}}^X & = & \text{Rem}[W]_{\hat{s}[u \leftarrow \hat{\ell}],\hat{h}}^X. \end{array}$$

We denote these equivalences by (Inv-u). Directly from them, we notice that for every core formula φ in $\text{Sk}[W](X, \alpha)$ and $\ell_1, \ell_2 \in LOC$, we have

$$\begin{aligned} (s[u \leftarrow \ell_1], h) \models \varphi, \quad & \text{iff } (s, h) \models \varphi, & (\text{by (Inv-u)}) \\ & \text{iff } (s', h') \models \varphi, & (\text{by } (s, h) \approx_{X,\alpha}^W (s', h')) \\ & \text{iff } (s'[u \leftarrow \ell_2], h') \models \varphi. & (\text{by (Inv-u)}) \end{aligned}$$

Therefore, in order to prove the result it is sufficient to show that for every $\ell_1 \in LOC$ there is $\ell_2 \leq \text{maxval}_X(s', h') + 1$ such that the memory states $(s[u \leftarrow \ell_1], h)$ and $(s'[u \leftarrow \ell_2], h')$ agree on the satisfaction of every core formula in $\text{Obs}[W](X)$. The choice for ℓ_2 depends on whether ℓ_1 belongs to the set of labelled locations, a predecessor set, the self-loop set, the remainder set, or it does not belong to any of these sets:

case: $\ell_1 \in \text{Lab}[W]_{s,h}^X$. Let $t \in T[W]^X$ be such that $\llbracket t \rrbracket_{s,h}^X = \ell_1$. By $(s, h) \approx_{X,\alpha}^W (s', h')$, $\llbracket t \rrbracket_{s',h'}^X$ is defined. Consider $\ell_2 = \llbracket t \rrbracket_{s',h'}^X$. Notice that if $t = x$ (syntactically) holds for some $x \in X$, then $\ell_2 \in s(X)$. Otherwise, $t = n(x)$ (for $x \in X$) and so $\ell_2 \in \text{ran}(h)$. Therefore, by definition of maximum value, $\ell_2 \leq \text{maxval}_X(s', h') + 1$. We show that the two memory states $(s[u \leftarrow \ell_1], h)$ and $(s'[u \leftarrow \ell_2], h')$ satisfy the same core formulae from $\text{Obs}[W](X)$. Given a core formula φ in $\{u \in \text{pred}_X^W(x), u \in \text{self}_X^W, u \in \text{rem}_X^W \mid x \in X\}$, we conclude that

$(s[u \leftarrow \ell_1], h) \not\models \varphi$ and $(s'[u \leftarrow \ell_2], h') \not\models \varphi$. Indeed, φ can be satisfied only if the location assigned to u is unlabelled. This is not the case here, as $\ell_1 \in \text{Lab}[\mathcal{W}]_{s,h}^X = \text{Lab}[\mathcal{W}]_{s[u \leftarrow \ell_1],h}^X$ and $\ell_2 \in \text{Lab}[\mathcal{W}]_{s',h'}^X = \text{Lab}[\mathcal{W}]_{s'[u \leftarrow \ell_2],h'}^X$ by (Inv-u). Now, let us consider a core formula of the form $u = t'$, where $t' \in T[\mathcal{W}]^X$. We have

$$\begin{aligned}
& (s[u \leftarrow \ell_1], h) \models u = t', \\
\Leftrightarrow & \ell_1 = \llbracket t' \rrbracket_{s[u \leftarrow \ell_1],h}^X, && (\text{by definition of } \models) \\
\Leftrightarrow & \ell_1 = \llbracket t' \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h}^X, && (\text{by hypothesis } \ell_1 = \llbracket t \rrbracket_{s,h}^X \text{ and } \llbracket t' \rrbracket_{s[u \leftarrow \ell_1],h}^X = \llbracket t' \rrbracket_{s,h}^X) \\
\Leftrightarrow & \ell_2 = \llbracket t' \rrbracket_{s',h'}^X = \llbracket t \rrbracket_{s',h'}^X, && (\text{from } \ell_2 = \llbracket t \rrbracket_{s',h'}^X \text{ and } (s, h) \approx_{X,\alpha}^{\mathcal{W}} (s', h')) \\
\Leftrightarrow & \ell_2 = \llbracket t' \rrbracket_{s'[u \leftarrow \ell_2],h'}^X, \\
& && (\text{from } \llbracket t' \rrbracket_{s'[u \leftarrow \ell_1],h'}^X = \llbracket t' \rrbracket_{s',h'}^X. \text{ The right-to-left direction also uses } \ell_2 = \llbracket t' \rrbracket_{s',h'}^X) \\
\Leftrightarrow & (s'[u \leftarrow \ell_2], h') \models u = t'. && (\text{by definition of } \models)
\end{aligned}$$

case: $\ell_1 \in \text{Rem}[\mathcal{W}]_{s,h}^X$. In this case, $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X) \geq 1$ and so $(s, h) \models \text{rem}_X^{\mathcal{W}} \geq 1$. From the hypothesis $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s', h')$, this implies $\text{card}(\text{Rem}[\mathcal{W}]_{s',h'}^X) \geq 1$. Consider $\ell_2 \in \text{Rem}[\mathcal{W}]_{s',h'}^X$. As $\ell_2 \in \text{dom}(h)$, we have $\ell_2 \leq \text{maxval}_X(s', h') + 1$. From (Inv-u) (fourth equivalence), both the memory states $(s[u \leftarrow \ell_1], h)$ and $(s'[u \leftarrow \ell_2], h')$ satisfy the formula $u \in \text{rem}_X^{\mathcal{W}}$. Moreover, $(s[u \leftarrow \ell_1], h) \not\models \varphi$ and $(s'[u \leftarrow \ell_2], h') \not\models \varphi$ hold for every core formula φ in the set $\{u = t, u \in \text{pred}_X^{\mathcal{W}}(x), u \in \text{self}_X^{\mathcal{W}} \mid x \in X \text{ and } t \in T[\mathcal{W}]^X\}$.

case: $\ell_1 \in \text{Self}[\mathcal{W}]_{s,h}^X$. In this case, $\text{card}(\text{Self}[\mathcal{W}]_{s,h}^X) \geq 1$ and so $(s, h) \models \text{self}_X^{\mathcal{W}} \geq 1$. From the hypothesis $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s', h')$, this means $\text{card}(\text{Self}[\mathcal{W}]_{s',h'}^X) \geq 1$. Consider $\ell_2 \in \text{Self}[\mathcal{W}]_{s',h'}^X$. As $\ell_2 \in \text{dom}(h)$, we have $\ell_2 \leq \text{maxval}_X(s', h') + 1$. Similarly to the previous cases, we can show that the two memory states $(s[u \leftarrow \ell_1], h)$ and $(s'[u \leftarrow \ell_2], h')$ satisfy the same core formulae from $\text{Obs}[\mathcal{W}](X)$. More precisely, the two memory states $(s[u \leftarrow \ell_1], h)$ and $(s'[u \leftarrow \ell_2], h')$ only satisfy the core formula $u \in \text{self}_X^{\mathcal{W}}$.

case: $\ell_1 \in \text{Pred}[\mathcal{W}]_{s,h}^X(x)$, **for some** $x \in X$. In this case, $\text{card}(\text{Pred}[\mathcal{W}]_{s,h}^X(x)) \geq 1$ and so (s, h) satisfies $\text{pred}_X^{\mathcal{W}}(x) \geq 1$. From $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s', h')$, this implies $\text{card}(\text{Pred}[\mathcal{W}]_{s',h'}^X(x)) \geq 1$. Consider $\ell_2 \in \text{Pred}[\mathcal{W}]_{s',h'}^X(x)$. As $\ell_2 \in \text{dom}(h)$, it holds that $\ell_2 \leq \text{maxval}_X(s', h') + 1$. Both $(s[u \leftarrow \ell_1], h) \not\models \varphi$ and $(s'[u \leftarrow \ell_2], h') \not\models \varphi$ hold for every formula φ in the set

$$\{u = t, u \in \text{self}_X^{\mathcal{W}}, u \in \text{rem}_X^{\mathcal{W}} \mid x \in X \text{ and } t \in T[\mathcal{W}]^X\}.$$

Let us consider the a core formula $u \in \text{pred}_X^{\mathcal{W}}(y)$, for $y \in X$. We have,

$$\begin{aligned}
& (s[u \leftarrow \ell_1], h) \models u \in \text{pred}_X^{\mathcal{W}}(y), \\
\Leftrightarrow & \ell_1 \in \text{Pred}[\mathcal{W}]_{s[u \leftarrow \ell_1],h}^X(y), && (\text{by definition of } \models) \\
\Leftrightarrow & \ell_1 \in \text{Pred}[\mathcal{W}]_{s,h}^X(x) \text{ and } s(x) = s(y), \\
& && (\text{by hypothesis } \ell_1 \in \text{Pred}[\mathcal{W}]_{s,h}^X(x) \text{ and } \text{Pred}[\mathcal{W}]_{s,h}^X(y) = \text{Pred}[\mathcal{W}]_{s[u \leftarrow \ell_1],h}^X(y), \text{ as } u \notin X) \\
\Leftrightarrow & \ell_2 \in \text{Pred}[\mathcal{W}]_{s',h'}^X(x) \text{ and } s'(x) = s'(y), \\
& && (\text{from } \ell_2 \in \text{Pred}[\mathcal{W}]_{s',h'}^X(x) \text{ and } (s, h) \approx_{X,\alpha}^{\mathcal{W}} (s', h')) \\
\Leftrightarrow & \ell_2 \in \text{Pred}[\mathcal{W}]_{s'[u \leftarrow \ell_2],h'}^X(y), \\
& && (\text{from } \text{Pred}[\mathcal{W}]_{s',h'}^X(y) = \text{Pred}[\mathcal{W}]_{s'[u \leftarrow \ell_1],h'}^X(y), \text{ as } u \notin X) \\
& && (\text{The right-to-left direction also uses } \ell_2 \in \text{Pred}[\mathcal{W}]_{s',h'}^X(x)) \\
\Leftrightarrow & (s'[u \leftarrow \ell_2], h') \models u \in \text{pred}_X^{\mathcal{W}}(y). && (\text{by definition of } \models)
\end{aligned}$$

case: $\ell_1 \notin \text{dom}(h) \cup \text{Lab}[\mathcal{W}]_{s,h}^X$. In this case, ℓ_1 is an unlabelled location that does not belong to $\text{Rem}[\mathcal{W}]_{s,h}^X$, $\text{Self}[\mathcal{W}]_{s,h}^X$ nor $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$ (for any $x \in X$). Let $\ell_2 = \text{maxval}_X(s', h') + 1$. By definition of $\text{maxval}_X(s', h')$, we have $\ell_2 \notin \text{dom}(h')$ and $\ell_2 \notin \text{Lab}[\mathcal{W}]_{s',h'}^X \subseteq \text{ran}(h') \cup s'(X)$. Thus, ℓ_2 is an unlabelled location that does not belong to neither $\text{Rem}[\mathcal{W}]_{s',h'}^X$, $\text{Self}[\mathcal{W}]_{s',h'}^X$ nor $\text{Pred}[\mathcal{W}]_{s',h'}^X(x)$ (for any $x \in X$). By (Inv-u) and from the definition of the core formulae, for every φ in $\text{Obs}[\mathcal{W}](X)$ we have $(s[u \leftarrow \ell_1], h) \not\models \varphi$ and $(s'[u \leftarrow \ell_2], h') \not\models \varphi$. \square

5.4 RECAP: HOW TO APPLY THE CORE FORMULAE TECHNIQUE

The proof of the \exists -simulation property ends the analysis of the weak fragment (with the exception of the magic wand $w \rightarrowtail s$, which is studied in Section 5.6). In the next section, we reformulate this analysis in the context of the strong fragment. While the key steps are exactly the same, the definition of the core formulae and the proofs needed in order to establish the \rightarrowtail -simulation property reveal to be much more challenging. Thus, before moving to the strong fragment, we recapitulate the key component of the analysis performed on the weak fragment.

Recall that the goal of Section 5.3 is to define a set of core formulae whose Boolean combinations capture the expressive power of the weak fragment (excluding the separating implication). From [104], the core formulae should capture the atomic formulae of the weak fragment, and satisfy the \rightarrowtail -simulation and \exists -simulation properties. With this in mind, we proceed as follows.

Step I and II. First of all, we focus on the definition of core formulae. We start (step I) by considering a family of disjoint sets of locations that partition the domain on the heap. In the case of the weak fragment, given a memory state (s, h) , this family is made of the set of labelled locations $\text{Lab}[\mathcal{W}]_{s,h}^X$, the predecessor sets $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$, the self-loops set $\text{Self}[\mathcal{W}]_{s,h}^X$ and the remainder set $\text{Rem}[\mathcal{W}]_{s,h}^X$. The following result is established.

Proposition 5.13. Let (s, h) be a memory state. The set of all the non-empty sets among $\text{dom}(h) \cap \text{Lab}[\mathcal{W}]_{s,h}^X$, $\text{Self}[\mathcal{W}]_{s,h}^X$, $\text{Rem}[\mathcal{W}]_{s,h}^X$ and all $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$ ($x \in X$), is a partition of $\text{dom}(h)$.

The core formulae are defined (step II) following this family of sets, so that the satisfiability of each core formula only depends on the locations in a single set. For instance, the core formula $\text{self}_X^{\mathcal{W}} \geq \beta$ only depends on the cardinality of the set $\text{Self}[\mathcal{W}]_{s,h}^X$. This is done to simplify the game hopping strategy used in order to prove the \rightarrowtail -simulation property.

We show that Boolean combinations of core formulae capture the atomic formulae of the logic. In the context of the weak fragment, this is established in Lemma 5.15, recalled below.

Lemma 5.15. Every atomic formula of the weak fragment written with variables from $X \cup \{u\}$ is equivalent to a Boolean combination of formulae from $\text{Core}[\mathcal{W}](X, 1)$.

Step III. The definition of core formulae naturally leads to an indistinguishability relation on memory states, where two memory states are in the relation if and only if they satisfy the same core formulae, up to certain thresholds. For the weak fragment, this relation is denoted by $\approx_{X,\alpha}^{\mathcal{W}}$, where the threshold $\alpha \geq 1$ is an upper bound on the natural number β appearing in the core formulae $\text{self}_X^{\mathcal{W}} \geq \beta$, $\text{pred}_X^{\mathcal{W}}(x) \geq \beta$ and $\text{rem}_X^{\mathcal{W}} \geq \beta$.

We show that the core formulae enjoy the \rightarrowtail -simulation property. Given $(s, h) \approx_{X,\alpha}^{\mathcal{W}} (s', h')$, this property states that for every way of partitioning h into two heaps h_1 and h_2 , and dividing α into $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ (so, $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$), there is a way of partitioning h' into

h'_1 and h'_2 such that $(s, h_1) \approx_{\mathbf{x}, \alpha_1}^{\mathcal{W}} (s', h'_2)$ and $(s, h_2) \approx_{\mathbf{x}, \alpha_2}^{\mathcal{W}} (s', h'_2)$. To prove the $*$ -simulation property, we rely on game hopping. We introduce the \mathcal{W} -hop relation, recalled below, which allows us to rephrase the $*$ -simulation property as the inclusion $\approx_{\mathbf{x}, \alpha}^{\mathcal{W}} \subseteq \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}}$.

Definition 5.17 (\mathcal{W} -hop relation). We write $\leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}}$ for the relation on memory states such that

$$(s, h) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s', h') \text{ iff } \begin{aligned} &\text{for every two heaps } h_1 \text{ and } h_2 \text{ and every } \alpha_1 \geq 1 \text{ and } \alpha_2 \geq 1, \\ &\text{if } h = h_1 + h_2 \text{ and } \alpha = \alpha_1 + \alpha_2 \text{ then there are two heaps } h'_1 \text{ and } h'_2 \\ &\text{such that } h' = h'_1 + h'_2, (s, h_1) \approx_{\mathbf{x}, \alpha_1}^{\mathcal{W}} (s', h'_1) \text{ and } (s, h_2) \approx_{\mathbf{x}, \alpha_2}^{\mathcal{W}} (s', h'_2). \end{aligned}$$

With the aim of building a chain of hops as explained in Section 5.2, we start by proving that the inclusion $\approx_{\mathbf{x}, \alpha}^{\mathcal{W}} \subseteq \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}}$ holds for specific memory states (s, h) and (s', h') that are very similar. First of all, we restrict ourselves to memory states that are in the relation $\approx_{\mathbf{x}, \alpha'}^{\mathcal{W}}$ for every $\alpha' \geq 1$. This is done in Lemma 5.18, recalled below.

Lemma 5.18. For every $\alpha \geq 1$, $(\bigcap_{\alpha' \geq 1} \approx_{\mathbf{x}, \alpha'}^{\mathcal{W}}) \subseteq \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}}$.

Notice that $((s, h), (s', h')) \in \bigcap_{\alpha' \geq 1} \approx_{\mathbf{x}, \alpha'}^{\mathcal{W}}$ implies that the two memory states (s, h) and (s', h') agree on the set of labelled locations, as well as on the cardinality of all sets defined in Step I. This facilitates the proof of $(s, h) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s', h')$, as the two memory states are essentially equivalent when it comes to the $*$ -simulation property.

After the “base case” for equivalent memory states considered in Lemma 5.18, we consider $(s, h) \approx_{\mathbf{x}, \alpha}^{\mathcal{W}} (s', h')$ where $s' = s$ and h' is obtained from h by modifying the locations of exactly one of the sets introduced in step I. Thanks to the disjointness of these sets, this local update simplifies the proof of $(s, h) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s', h')$, discussed in Lemma 5.19.

Lemma 5.19. $(s, h) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s, h')$ holds if $(s, h) \approx_{\mathbf{x}, \alpha}^{\mathcal{W}} (s, h')$ and one of the following holds:

- (I) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \mathsf{Rem}[\mathcal{W}]_{s, h}^{\mathbf{x}}\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \mathsf{Rem}[\mathcal{W}]_{s, h'}^{\mathbf{x}}\}$,
- (II) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \mathsf{Self}[\mathcal{W}]_{s, h}^{\mathbf{x}}\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \mathsf{Self}[\mathcal{W}]_{s, h'}^{\mathbf{x}}\}$,
- (III) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \mathsf{Pred}[\mathcal{W}]_{s, h}^{\mathbf{x}}(\mathbf{x})\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \mathsf{Pred}[\mathcal{W}]_{s, h'}^{\mathbf{x}}(\mathbf{x})\}$, for some $\mathbf{x} \in \mathbf{X}$.

After Lemma 5.19 is established, we are ready to prove the $*$ -simulation property $\approx_{\mathbf{x}, \alpha}^{\mathcal{W}} \subseteq \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}}$ by building a chain of hops as the one schematised below:

$$(s, h) = (s_1, h_1) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s_2, h_2) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} \dots \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s_{k-1}, h_{k-1}) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s_k, h_k) = (s', h').$$

This is done inductively in Lemma 5.20. At each hop $j \in [1, k-2]$, the memory state (s_{j+1}, h_{j+1}) is constructed by locally updating (s_j, h_j) so that we rely on Lemma 5.19 to conclude that $(s_j, h_j) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s_{j+1}, h_{j+1})$. At each hop we make (s_j, h_j) closer and closer to (s', h') , until we reach a memory state (s_{k-1}, h_{k-1}) for which we can derive $(s_{k-1}, h_{k-1}) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s', h')$ directly by Lemma 5.18. Then, $(s, h) \leftrightarrow_{\mathbf{x}, \alpha}^{\mathcal{W}} (s', h')$ follows by transitivity of the hop relation (see Lemma 5.9), concluding the proof of the $*$ -simulation property.

In the next section, most of our efforts are spent deriving a $*$ -simulation property for the core formulae of the strong fragment. Even though the complexity of these formulae severely complicates the technical steps required to show this result, the proof strategy we use is exactly the one described here.

Step IV. Lastly, we show that the core formulae enjoy the following \exists -simulation property.

Lemma 5.23 ($w:\exists$ -simulation). Suppose $(s, h) \approx_{X,\alpha}^W (s', h')$. For every location $\ell_1 \in \text{LOC}$ there is a location $\ell_2 \leq \text{maxval}_X(s', h') + 1$ such that $(s[u \leftarrow \ell_1], h) \approx_{X,\alpha}^W (s'[u \leftarrow \ell_2], h')$.

This property essentially states that every new assignment $u \leftarrow \ell_1$ performed on (s, h) can be simulated in (s', h') , with respect to the indistinguishability relation $\approx_{X,\alpha}^W$. The proof is by cases on the membership of ℓ_1 to the sets defined in step I. To facilitate this case analysis, the core formulae introduced in step II are divided in two sets $\text{Sk}[w](X, \alpha)$ and $\text{Obs}[w](X)$. The satisfaction of formulae in $\text{Sk}[w](X, \alpha)$ does not depend on the location assigned to u , so that only the core formulae in $\text{Obs}[w](X)$ need to be considered.

5.5 A FAMILY OF CORE FORMULAE CAPTURING THE FRAGMENT s

In this section, we extend the analysis carried out in Section 5.3 and summarised in Section 5.4 to the context of the fragment s . Our goal is to define a set of core formulae whose Boolean combinations capture the expressive power of strong fragment, whose syntax is recalled below:

$$s := w \mid x \hookrightarrow^+ t \mid u \hookrightarrow^+ u \mid s \wedge s \mid \neg \varphi \mid s * s \mid \exists u s.$$

The strong fragment can be seen as an extension of the weak fragment featuring the reachability predicates $x \hookrightarrow^+ t$ and $u \hookrightarrow^+ u$. This addition complicates the definition of the core formulae, which reflects on the analysis needed to prove the $*$ -simulation and \exists -simulation properties.

5.5.1 Step I: partitioning the heap.

As done in Section 5.3, we first aim at defining a partition of the heap, which depends on syntactical terms that correspond to specific locations of a memory state. These terms are more advanced than the next-point variables introduced for the weak fragment, and they deserve a more gentle introduction. For the whole section, we let $X \subseteq_{\text{fin}} \text{VAR} \setminus \{u\}$ be a finite set of program variables not including the unique quantified variable name u .

End-point variables. Consider a memory state (s, h) . As analysed during Example 5.2, the strong formula $\exists u (x \hookrightarrow^+ u * u \hookrightarrow^+ u)$, states that $s(x)$ does not belong to a cycle but reaches one (for instance, see Figure 5.7). More precisely, in the formula this is accomplished by stating that it is possible to assign to u the first location reachable from $s(x)$ that belongs to a cycle. Formally, a location ℓ *belongs to a cycle* in h whenever there is $\delta \geq 1$ such that $h^\delta(\ell) = \ell$. Similarly, we can force u to correspond to a location that is not in the domain of the heap and it is reached by $s(x)$ in at least one step. This corresponds to the formula $\exists u (x \hookrightarrow^+ u \wedge \neg u \hookrightarrow_-)$. These two examples lead to the introduction of *end-point variables*: syntactical objects of the form $e(x)$ whose set is defined as $\text{EV}[s]^X \stackrel{\text{def}}{=} \{e(x) \mid x \in X\}$. An end-point variable $e(x)$ is intended to correspond exactly to a location among the ones we just described. More precisely,

1. if $s(x)$ does not belong to a cycle but reaches one, then $e(x)$ corresponds to the first location reachable from $s(x)$ that belongs to that cycle,
2. otherwise, if $s(x) \in \text{dom}(h)$ does not reach a cycle, then $e(x)$ corresponds to the only location reachable from $s(x)$ that does not belong to the domain of h .

Formally, the semantics of $e(x)$ is given by extending the evaluation $\llbracket \cdot \rrbracket_{s,h}^X$ as follows:

$$\llbracket e(x) \rrbracket_{s,h}^X = \ell \stackrel{\text{def}}{\iff} \text{there is } \delta \geq 1 \text{ s.t. } h^\delta(s(x)) = \ell \text{ and if } \ell \in \text{dom}(h) \text{ then } \ell \text{ belongs to a cycle.}$$

Moreover, $h^{\delta-1}(s(x))$ does not belong to a cycle.

We highlight that $\llbracket e(x) \rrbracket_{s,h}^X$ is not defined if $s(x) \notin \text{dom}(h)$, nor if $s(x)$ belongs to a cycle. Otherwise, $\llbracket e(x) \rrbracket_{s,h}^X$ is uniquely defined.

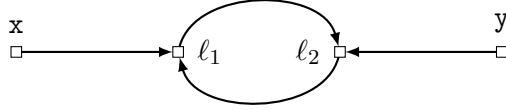
Meet-point variables. In Example 5.2, we have seen that $\exists u ((x \hookrightarrow^+ u * y \hookrightarrow^+ u) \wedge \neg u \hookrightarrow_-)$ is satisfied whenever a memory state (s, h) witnesses two disjoint non-empty paths, one going from $s(x)$ to ℓ and one going from $s(y)$ to ℓ , where ℓ is a location not in the domain of h . Here, notice that the disjointness of the two paths refers to the arrows in the heap. This pattern contributes to the definition of the *meet-point variables* $MV[s]^X \stackrel{\text{def}}{=} \{m(x, y) \mid x, y \in X\}$. A meet-point variable $m(x, y) \in MV[s]^X$ is evaluated through $\llbracket \cdot \rrbracket_{s,h}^X$ as follows:

$$\llbracket m(x, y) \rrbracket_{s,h}^X = \ell \stackrel{\text{def}}{\iff} \text{there are } \delta_1, \delta_2 \geq 1 \text{ such that } h^{\delta_1}(s(x)) = h^{\delta_2}(s(y)) = \ell \text{ and}$$

for all $\delta'_1 \in [0, \delta_1], \delta'_2 \in [0, \delta_2]$, if $\delta'_1 + \delta'_2 < \delta_1 + \delta_2$ then $h^{\delta'_1}(s(x)) \neq h^{\delta'_2}(s(y))$.

Moreover, ℓ does not belong to a cycle of h .

Informally, when $\llbracket m(x, y) \rrbracket_{s,h}^X$ is defined, it is the only location ℓ for which there are two disjoint non-empty paths, one going from $s(x)$ to ℓ and one going from $s(y)$ to ℓ . In order for $\llbracket m(x, y) \rrbracket_{s,h}^X$ to be uniquely defined, we require that it does not correspond to a location belonging to a cycle, as formalised in the last statement of its characterisation. Indeed, in the following memory state there are two ways to construct disjoint paths going from $s(x)$ and $s(y)$ to a common location ℓ , one where $\ell = \ell_1$ and the other where $\ell = \ell_2$.



In this case, $\llbracket m(x, y) \rrbracket_{s,h}^X$ is not defined. However, we notice that the locations ℓ_1 and ℓ_2 are still interesting, as they correspond to the end-point variables $e(x)$ and $e(y)$, respectively. Moreover, if $\{\ell_1 \mapsto \ell_2\}$ is removed from the heap then ℓ_1 becomes the location corresponding to $m(x, y)$. Symmetrically, removing $\{\ell_2 \mapsto \ell_1\}$ leads to $m(x, y)$ being evaluated as ℓ_2 . One can notice that there is only one case where $m(x, y)$ and $e(x)$ evaluate to the same location. That is, when (s, h) satisfies the formula $\exists u ((x \hookrightarrow^+ u * y \hookrightarrow^+ u) \wedge \neg u \hookrightarrow_-)$ discussed above.

Labelled locations. We extend the notion of *terms* introduced in the previous section to the elements belonging to the set $T[s]^X \stackrel{\text{def}}{=} X \cup EV[s]^X \cup MV[s]^X$. Similarly, we call *labelled* the locations corresponding to terms of $T[s]^X$, and write $\text{Lab}[s]_{s,h}^X$ for their set.

Example 5.24. Figure 5.7 highlights the labelled locations of a memory state, say (s, h) , with their respective terms. Notice that no location corresponds to the term $m(v, w)$, since there is a path going from $s(w)$ to $s(v)$. Similarly, $\llbracket m(x, z) \rrbracket_{s,h}^X$ and $\llbracket m(y, z) \rrbracket_{s,h}^X$ are not defined, as the two disjoint paths starting from $s(x)$ (or $s(y)$) and $s(z)$ meet at a location that belongs to a cycle, i.e. the location corresponding to the end-point variables $e(x)$, $e(y)$ and $e(z)$.

Interestingly enough, despite the fact that $MV[s]^X$ contains $\text{card}(X)^2$ meet-point variables, we show that $\text{Lab}[s]_{s,h}^X$ contains at most $\text{card}(X)$ distinct locations that correspond to meet-point or

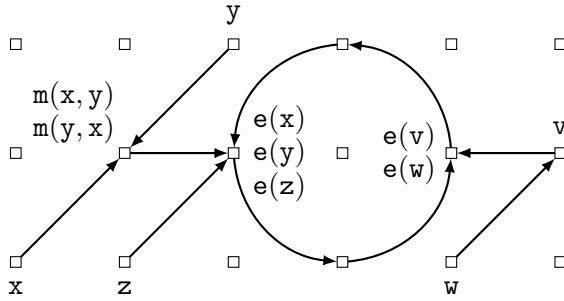


Figure 5.7: A memory state. Labelled locations are highlighted.

end-point variables. That is, $\text{card}(\text{Lab}[s]_{s,h}^X \setminus s(X)) \leq \text{card}(X)$. Together with $s(X) \subseteq \text{Lab}[s]_{s,h}^X$, this implies that $\text{card}(\text{Lab}[s]_{s,h}^X) \leq 2 \times \text{card}(X)$. Below, we prove this result, which helps us familiarise with the concepts of meet-point and end-point variables.

Lemma 5.25. $\text{card}(\text{Lab}[s]_{s,h}^X \setminus s(X)) \leq \text{card}(X)$.

The proof of this lemma revisits standard arguments relating the number of internal nodes appearing in a forest with the number of its leafs. Roughly speaking, the reachability relation between locations corresponding to program variables and meet-points variables can be represented as a forest where every location corresponding to a meet-point variable has at least 2 children. In each tree of the forest, if n locations correspond to meet-point variables then there are at least $n + 1$ leaves, all of them corresponding to program variables in $Y \subseteq X$. Lastly, the root of each tree reaches exactly one location that corresponds to an end-point variable written using variables from Y . This leads to Lemma 5.25. The formal proof is given below.

Proof. Suppose $\ell = \llbracket m(x,y) \rrbracket_{s,h}^X$ and $\ell' = \llbracket e(x) \rrbracket_{s,h}^X$. We recall that, from the semantics of meet-point variables, $\ell = \llbracket m(x,y) \rrbracket_{s,h}^X$ implies that there are two disjoint non-empty paths, one going from $s(x)$ to ℓ and one going from $s(y)$ to ℓ . Moreover, ℓ does not belong to a cycle of h . Instead, from the semantics of end-point variables, $\ell' = \llbracket e(x) \rrbracket_{s,h}^X$ implies that $s(x)$ does not belong to a cycle and there is a non-empty path going from $s(x)$ to ℓ' . Moreover, either ℓ' does not belong to $\text{dom}(h)$ or it is the first location reachable from $s(x)$ that belongs to a cycle in h . Obviously, this means that there cannot be a non-empty path in h going from ℓ' to ℓ , as it would entail that ℓ belongs to a cycle. On the contrary, h witnesses a (possibly empty) path going from ℓ to ℓ' . Moreover, $\ell' = \llbracket e(y) \rrbracket_{s,h}^X$. Indeed, h witnesses a non-empty path going from $s(y)$ to ℓ , and a path from ℓ to ℓ' . If ℓ' does not belong to $\text{dom}(h)$, then clearly $\ell' = \llbracket e(y) \rrbracket_{s,h}^X$. Otherwise, as $\ell' = \llbracket e(x) \rrbracket_{s,h}^X$, we conclude that ℓ' is the first location reachable from $s(x)$ that belongs to a cycle. Since $\ell = \llbracket m(x,y) \rrbracket_{s,h}^X$ does not belong to a cycle, we conclude that the path in h going from ℓ to ℓ' is non-empty, which implies that ℓ' is also the first location reachable from $s(y)$ that belongs to a cycle. Recapitulating,

- A. if $\llbracket m(x,y) \rrbracket_{s,h}^X$ is defined, then $\llbracket e(x) \rrbracket_{s,h}^X = \llbracket e(y) \rrbracket_{s,h}^X$.

As every location corresponding to a meet-point variable necessarily reaches a location corresponding to an end-point variable, the lemma trivially holds if no end-point variable is defined. Otherwise, let us consider the set $S = \{\ell_1, \dots, \ell_k\}$ of all locations corresponding to at least one end-point variable, where $k \geq 1$. Furthermore, given $i \in [1, k]$, we write $E(\ell_i)$ for the set of variables in X that are used to write the end-point variable corresponding to ℓ_i , that

is $E(\ell_j) \stackrel{\text{def}}{=} \{x \in X \mid [\![e(x)]\!]_{s,h}^X = \ell_j\}$. For every distinct $i, j \in [1, k]$, $E(\ell_i)$ and $E(\ell_j)$ are disjoint. Moreover, $\bigcup_{j \in [1, k]} E(\ell_j) \subseteq X$, which implies

$$\text{B. } k \leq \sum_{j \in [1, k]} \text{card}(E(\ell_j)) \leq \text{card}(X).$$

We write $M(\ell_j)$ for the set $\{\ell \in \text{LOC} \mid \text{there are } x, y \in E(\ell_j), [\![m(x, y)]\!]_{s,h}^X = \ell\}$. Given $i, j \in [1, k]$, $x \in E(\ell_i)$ and $y \in E(\ell_j)$, if $[\![m(x, y)]\!]_{s,h}^X$ is defined then, by (A), $\ell_i = [\![e(x)]\!]_{s,h}^X = [\![e(y)]\!]_{s,h}^X = \ell_j$, and thus $i = j$. This allows us to manipulate $\text{Lab}[S]_{s,h}^X \setminus s(x)$ as follows:

$$\begin{aligned} \text{Lab}[S]_{s,h}^X \setminus s(x) &= \{\ell_1, \dots, \ell_k\} \cup \{\ell \in \text{Lab}[S]_{s,h}^X \mid \text{there are } x, y \in X, [\![m(x, y)]\!]_{s,h}^X = \ell\} \\ &= \{\ell_1, \dots, \ell_k\} \cup \bigcup_{j \in [1, k]} M(\ell_j). \end{aligned}$$

Therefore, $\text{card}(\text{Lab}[S]_{s,h}^X \setminus s(x)) \leq k + \sum_{j \in [1, k]} \text{card}(M(\ell_j))$. In order to conclude the proof it is sufficient to show that for every $j \in [1, k]$, $\text{card}(M(\ell_j)) < \text{card}(E(\ell_j))$. Indeed, by (B), this implies

$$\text{card}(\text{Lab}[S]_{s,h}^X \setminus s(x)) \leq k + \sum_{j \in [1, k]} (\text{card}(E(\ell_j)) - 1) \leq \sum_{j \in [1, k]} \text{card}(E(\ell_j)) \leq s(X).$$

Let $j \in [1, k]$. As $E(\ell_j) \neq \emptyset$, if $M(\ell_j) = \emptyset$ then obviously $\text{card}(M(\ell_j)) < \text{card}(E(\ell_j))$. Therefore, let us assume $M(\ell_j) \neq \emptyset$. We extend the definition of $E(\cdot)$ and $M(\cdot)$ to locations in $M(\ell_j)$. Given $\ell \in M(\ell_j)$, we write $E(\cdot)$ for the set of locations $x \in E(\ell_j)$ such that h witnesses a non-empty path going from $s(x)$ to ℓ . We write $M(\ell)$ for the set of locations $\ell' \in M(\ell_j)$ such that h witnesses a (possibly empty) path going from ℓ' to ℓ . Notice that $M(\ell) \subseteq M(\ell_j)$ and $E(\ell) \subseteq E(\ell_j)$. Since ℓ corresponds to a meet-point variable, the following properties are satisfied:

- C. For every $\ell' \in M(\ell)$, if $\ell' \neq \ell$ then $\ell \notin M(\ell')$.
- D. For every $\ell' \in M(\ell)$, $M(\ell') \subseteq M(\ell)$.
- E. $\ell', \ell'' \in M(\ell)$, if $\ell' \notin M(\ell'')$ and $\ell'' \notin M(\ell')$, then $M(\ell') \cap M(\ell'') = \emptyset$ and $E(\ell') \cap E(\ell'') = \emptyset$.
- F. For every $\ell' \in M(\ell)$, if $\ell' \neq \ell$ then $E(\ell') \subset E(\ell)$.

The properties (C)–(E) hold directly from the fact that ℓ does not belong to a cycle. In particular, (C) states that if h witnesses a non-empty path going from $\ell' \in M(\ell)$ to ℓ , then h does not witness a path going from ℓ to ℓ' . (D) states that if a location corresponding to a meet-point variable reaches $\ell' \in M(\ell)$, then it also reaches also ℓ . (E) states that, given two locations $\ell', \ell'' \in M(\ell)$ that do not reach each other, for every location ℓ''' that corresponds to either a variable or a meet-point variable, if ℓ''' reaches ℓ' then ℓ''' does not reach ℓ'' , and vice versa. The property (F) is slightly different, and holds directly from the fact that, for all $\ell \in M(\ell')$, h witnesses disjoint non-empty paths going from (at least) two locations assigned to program variable to ℓ . Therefore, given $\ell' \in M(\ell) \setminus \{\ell\}$, the locations in $s(X)$ that reach ℓ are more than the locations in $s(X)$ that reach ℓ' . This implies (F). Given $\ell \in M(\ell_j) \cup \{\ell_j\}$, we prove that $\text{card}(M(\ell_j)) < \text{card}(E(\ell_j))$ by induction on $M(\ell) \setminus \{\ell\}$, and with induction hypothesis:

$$\text{for every } \ell' \in M(\ell) \setminus \{\ell\}, \text{card}(M(\ell')) < \text{card}(E(\ell')).$$

Notice that, from (C) and (D), for every $\ell' \in M(\ell)$, if $\ell' \neq \ell$ then $M(\ell') \subset M(\ell)$. The induction is well-founded, with base case $M(\ell) \setminus \{\ell\} = \emptyset$.

base case: $M(\ell) \setminus \{\ell\} = \emptyset$. As we assumed $M(\ell_j) \neq \emptyset$, ℓ is a meet-point variable (and it can be that $\ell = \ell_j$). Let $x, y \in \text{VAR}$ such that $[\![m(x, y)]\!]_{s,h}^X = \ell$. By definition, h witnesses two disjoint non-empty paths, one going from $s(x)$ to ℓ , and one going from $s(y)$ to ℓ . By definition of $E(\cdot)$, $\{x, y\} \subseteq E(\ell)$. Thus, $\text{card}(M(\ell)) \leq 1 < 2 \leq \text{card}(E(\ell))$.

induction step: $M(\ell) \setminus \{\ell\} = \{\ell'_1, \dots, \ell'_n\}$. By (E), together with the fact that for all $i \in [1, n]$, $\ell'_i \in M(\ell'_i)$, there is a non-empty subset $\{\ell''_1, \dots, \ell''_m\} \subseteq \{\ell'_1, \dots, \ell'_n\}$ such that

$$\text{G. for every } i, i' \in [1, m], M(\ell''_i) \cap M(\ell''_{i'}) = \emptyset \text{ and } E(\ell''_i) \cap E(\ell''_{i'}) = \emptyset,$$

$$\text{H. } \{\ell'_1, \dots, \ell'_n\} = \bigcup_{i \in [1, m]} M(\ell''_i).$$

By induction hypothesis, for every $i \in [1, m]$, $\text{card}(M(\ell''_i)) < \text{card}(E(\ell''_i))$. Therefore,

$$n = \sum_{i \in [1, m]} \text{card}(M(\ell''_i)) \leq (\sum_{i \in [1, m]} \text{card}(E(\ell''_i))) - m, \quad (\text{I})$$

where the first equivalence holds by (H). The proof splits in the following three cases:

case: ℓ does not correspond to a meet-point. In this case, $\ell = \ell_j$ and $\ell_j \notin M(\ell_j)$.

Therefore, we conclude that $M(\ell_j) = \{\ell'_1, \dots, \ell'_n\}$ and $\bigcup_{i \in [1, m]} E(\ell''_i) \subseteq E(\ell_j)$. From (G) and (I), $\text{card}(M(\ell_j)) = n < (\sum_{i \in [1, m]} \text{card}(E(\ell''_i))) < \text{card}(E(\ell_j))$.

case: ℓ corresponds to a meet-point variable, and $m = 1$. In this case, we have that

$M(\ell) = \{\ell', \dots, \ell'_n, \ell\}$ and thus $\text{card}(M(\ell)) = n + 1$. Since $m = 1$, by (I), we have $\text{card}(M(\ell)) \leq \text{card}(E(\ell'_1))$. By (F), $\text{card}(E(\ell'_1)) < \text{card}(E(\ell))$. So, $\text{card}(M(\ell)) < \text{card}(E(\ell))$.

case: ℓ corresponds to a meet-point variable, and $m \geq 2$. As in the previous case,

we have $M(\ell) = \{\ell'_1, \dots, \ell'_n, \ell\}$ and thus $\text{card}(M(\ell)) = n + 1$. Since $m \geq 2$, by (I), $\text{card}(M(\ell)) < \sum_{i \in [1, m]} \text{card}(E(\ell''_i))$. By (G) and (F), $\sum_{i \in [1, m]} \text{card}(E(\ell''_i)) \leq \text{card}(E(\ell))$. Thus, $\text{card}(M(\ell)) < \text{card}(E(\ell))$. \square

Further properties of labelled locations are required in order to introduce the partition of the heap. First of all, we notice that every labelled location that belongs to the domain of the heap reaches, in at least one step, a labelled location.

Lemma 5.26. Let (s, h) be a memory state and consider a labelled location $\ell \in \text{Lab}[s]_{s,h}^X$ such that $\ell \in \text{dom}(h)$. h witnesses a non-empty path going from ℓ to a labelled location.

Proof. If ℓ belongs to a cycle, then the lemma is trivially satisfied. So, let us assume that ℓ does not belong to a cycle. As $\ell \in \text{dom}(h)$, it cannot be that ℓ corresponds to an end-point variable, and therefore it corresponds to either a variable or a meet-point variable. Let us analyse the first case by supposing that there is $x \in X$ such that $s(x) = \ell$. As ℓ does not belong to a cycle and it is in the domain of the heap, $\llbracket e(x) \rrbracket_{s,h}^X$ is defined. From its definition, $\llbracket e(x) \rrbracket_{s,h}^X$ is reached by ℓ in at least one step. Now, let us suppose that there are $x, y \in X$ such that $\llbracket m(x, y) \rrbracket_{s,h}^X = \ell$. By definition, there is a non-empty path from $s(x)$ to ℓ , which implies that $s(x) \in \text{dom}(h)$. Again, $s(x)$ cannot belong to a cycle, and therefore $\llbracket e(x) \rrbracket_{s,h}^X$ is defined. From the functionality of h , we know that there must be either a path from $\llbracket e(x) \rrbracket_{s,h}^X$ to ℓ or vice versa. However, $\llbracket e(x) \rrbracket_{s,h}^X$ either belongs to a cycle or does not belong to $\text{dom}(h)$. Since ℓ is in $\text{dom}(h)$ and does not belong to a cycle, there is a non-empty path from ℓ to $\llbracket e(x) \rrbracket_{s,h}^X$. \square

Following up on Lemma 5.26, among the labelled locations that are reachable, in at least one step, from the labelled location $\ell \in \text{dom}(h)$, we are particularly interested in the one that is *closest* to ℓ . We call this the location *seen by* ℓ . Its formal definition is given below.

Definition 5.27 (Seen by). Let (s, h) be a memory state and let $\ell \in \text{Lab}[s]_{s,h}^X \cap \text{dom}(h)$. A labelled location ℓ' is *seen by* ℓ if there is $\delta \geq 1$ such that (1) $h^\delta(\ell) = \ell'$ and (2) for every $\delta' \in [1, \delta - 1]$, $h^{\delta'}(\ell)$ is not a labelled location. We write $\text{sby}_{s,h}^X(\ell)$ to denote ℓ' .

Given $t \in T[s]^X$ such that $\llbracket t \rrbracket_{s,h}^X$ is defined and in the domain of h , we often write $\text{sby}_{s,h}^X(t)$ as a shortcut for $\text{sby}_{s,h}^X(\llbracket t \rrbracket_{s,h}^X)$. The existence of a seen by location stems from Lemma 5.26, whereas its uniqueness follows directly from the property (2) of its definition.

Proposition 5.28. Let (s, h) be a memory state and let $\ell \in \text{Lab}[s]_{s,h}^X \cap \text{dom}(h)$. There is a unique location ℓ' such that $\text{sby}_{s,h}^X(\ell) = \ell'$.

The partition. We introduce the partition of the heap. In order to simplify the presentation, for the whole chapter, given an arbitrary set S we write $[S]^\flat$ for the set of all the locations needed in order to construct S . For instance, $[S]^\flat = \bigcup_{L \in S} L$ if S is a set of sets of locations, and $[S]^\flat = \bigcup_{(L_1, L_2) \in S} (L_1 \cup L_2)$ if S is a set of pairs of sets of locations, i.e. a subset of $2^{\text{LOC} \times \text{LOC}}$.

Definition 5.29 (Predecessors, paths, cycles and the remainder). Let (s, h) be a memory state and consider a natural number $\alpha \geq 1$. We define the following subsets of $\text{dom}(h)$:

Predecessors. Given $\ell \in s(\mathbf{x})$, $\text{Pred}[s]_{s,h}^{\mathbf{x}}(\ell)$ is the set of *predecessors* of ℓ that are not reached by any location corresponding to program variables. Formally,

$$\text{Pred}[s]_{s,h}^{\mathbf{x}}(\ell) \stackrel{\text{def}}{=} \{\ell' \in \text{dom}(h) \mid h(\ell') = \ell \text{ and for all } y \in \mathbf{x} \text{ and } \delta \geq 0, h^\delta(s(y)) \neq \ell'\}.$$

We notice that ℓ is assigned to a program variable. Given $\mathbf{x} \in \mathbf{X}$, we often write $\text{Pred}[s]_{s,h}^{\mathbf{x}}(\mathbf{x})$ as a shortcut for $\text{Pred}[s]_{s,h}^{\mathbf{x}}(s(\mathbf{x}))$.

Paths. Given a labelled location $\ell \in \text{Lab}[s]_{s,h}^{\mathbf{x}}$, $\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell)$ is the set of memory cells that are reachable from ℓ without passing through the location seen by ℓ . Formally,

$$\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell) \stackrel{\text{def}}{=} \{\ell' \in \text{dom}(h) \mid \exists \delta \geq 0 \ h^\delta(\ell) = \ell' \text{ and for all } \delta' \in [1, \delta], h^{\delta'}(\ell) \neq \text{sby}_{s,h}^{\mathbf{x}}(\ell)\}.$$

Given $\mathbf{t} \in T[s]^{\mathbf{x}}$, we often write $\text{Path}[s]_{s,h}^{\mathbf{x}}(\mathbf{t})$ as a shortcut for $\text{Path}[s]_{s,h}^{\mathbf{x}}([\mathbf{t}]_{s,h}^{\mathbf{x}})$.

Bounded cycles. Given $\beta \in [1, \alpha]$, $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$ is the set of *unlabelled cycles* of length β . A cycle is said to be *unlabelled* if it does not involve labelled locations. Formally,

$$\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta) \stackrel{\text{def}}{=} \{\{\ell_0, \dots, \ell_{\beta-1}\} \subseteq \text{dom}(h) \mid \forall j \in [0, \beta-1], h(\ell_j) = \ell_{(j+1 \bmod \beta)} \notin \text{Lab}[s]_{s,h}^{\mathbf{x}}\}.$$

Unbounded cycles. $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x}, \alpha}$ is the set of *unbounded and unlabelled cycles*, i.e. unlabelled cycles of length strictly greater than α . Formally,

$$\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x}, \alpha} \stackrel{\text{def}}{=} \left\{ \{\ell_0, \dots, \ell_\gamma\} \subseteq \text{dom}(h) \mid \begin{array}{l} \gamma > \alpha \text{ and for every } j \in [0, \gamma-1], \\ h(\ell_i) = \ell_{(i+1 \bmod \gamma)} \notin \text{Lab}[s]_{s,h}^{\mathbf{x}} \end{array} \right\}.$$

Remainder. $\text{Rem}[s]_{s,h}^{\mathbf{x}, \alpha}$ is the set of memory cells that do not belong to any of the sets above, i.e.

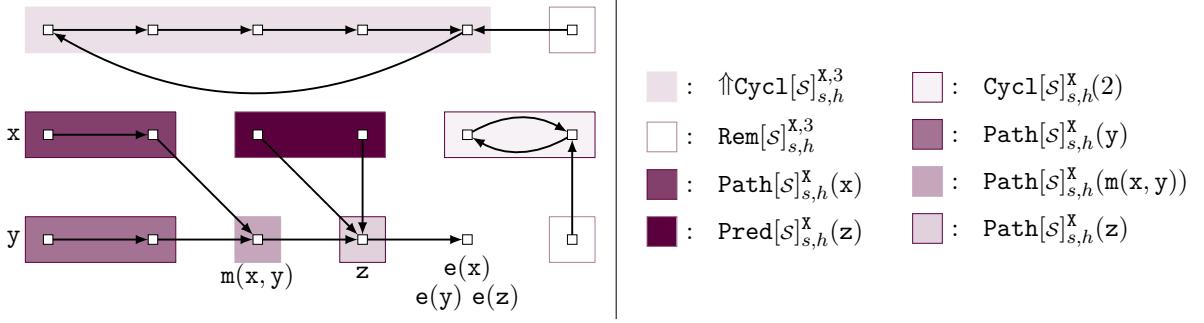
$$\text{dom}(h) \setminus \left([\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x}, \alpha}]^\flat \cup \bigcup_{\mathbf{x} \in \mathbf{X}} \text{Pred}[s]_{s,h}^{\mathbf{x}}(\mathbf{x}) \cup \bigcup_{\ell \in \text{Lab}[s]_{s,h}^{\mathbf{x}}} \text{Path}[s]_{s,h}^{\mathbf{x}}(\ell) \cup \bigcup_{\beta \in [1, \alpha]} [\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)]^\flat \right).$$

Very often in the proofs of this section, we reason on subsets of the domain of a heap h instead of looking directly at h . In particular, we often rely on sets in order to *describe* paths in the heap. To be completely formal, we define what we mean by describing a path using a set.

Definition 5.30 (Sets describing paths.). A set of locations L *describes* a path in a heap h , going from a location ℓ_1 to a location ℓ_2 , if the heap $h' \stackrel{\text{def}}{=} \{(\ell, \ell') \in h \mid \ell \in L\}$ witnesses a path from ℓ_1 to ℓ_2 . L is *minimal* if every strict subset of L does not describe a path from ℓ_1 to ℓ_2 . Alike, L describes a cycle if h' is cyclic. It is minimal if its strict subsets do not describe a cycle.

Using this terminology, every set in $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$ and $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x}, \alpha}$ describes a cycle and it is minimal. Whenever non-empty, $\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell)$ is a minimal set describing a non-empty path in h , going from ℓ to $\text{sby}_{s,h}^{\mathbf{x}}(\ell)$. This means that $\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell)$ is empty whenever $\ell \notin \text{dom}(h)$. Otherwise, $\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell)$ must contain ℓ . Moreover, this set contains $\text{sby}_{s,h}^{\mathbf{x}}(\ell)$ if and only if $\text{sby}_{s,h}^{\mathbf{x}}(\ell) = \ell$. In this case, $\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell)$ is a minimal set describing a cycle. Moreover, in accordance with the definition of $\text{sby}_{s,h}^{\mathbf{x}}(\ell)$, no location other than ℓ belongs to both $\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell)$ and $\text{Lab}[s]_{s,h}^{\mathbf{x}}$, i.e. whenever non-empty, the only labelled location in $\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell)$ is ℓ .

Figure 5.8 shows a memory state and highlights the sets in Definition 5.29. It is easy to see that these sets define a partition of $\text{dom}(h)$ (equivalently, of h). The proof is left to the reader.

Figure 5.8: A memory state (s, h) . The partition of the heap is highlighted ($\alpha = 3$).

$$\begin{aligned}
 (s, h) \models t_1 = t_2 &\quad \text{iff } \llbracket t_1 \rrbracket_{s,h}^X \text{ and } \llbracket t_2 \rrbracket_{s,h}^X \text{ are defined and } \llbracket t_1 \rrbracket_{s,h}^X = \llbracket t_2 \rrbracket_{s,h}^X, \\
 (s, h) \models \text{sees}_X(t_1, t_2) \geq \beta &\quad \text{iff } \text{card}(\text{Path}[S]_{s,h}^X(t_1)) \geq \beta \text{ and } \llbracket t_2 \rrbracket_{s,h}^X = \text{sby}_{s,h}^X(t_1), \\
 (s, h) \models \text{loop}_X^S(\beta_1) \geq \beta_2 &\quad \text{iff } \text{card}(\text{Cycl}[S]_{s,h}^X(\beta_1)) \geq \beta_2, \\
 (s, h) \models \text{↑loop}_{X,\alpha}^S \geq \beta &\quad \text{iff } \text{card}(\text{↑Cycl}[S]_{s,h}^{X,\alpha}) \geq \beta, \\
 (s, h) \models \text{pred}_X^S(x) \geq \beta &\quad \text{iff } \text{card}(\text{Pred}[S]_{s,h}^X(x)) \geq \beta, \\
 (s, h) \models \text{rem}_{X,\alpha}^S \geq \beta &\quad \text{iff } \text{card}(\text{Rem}[S]_{s,h}^{X,\alpha}) \geq \beta.
 \end{aligned}$$

Figure 5.9: Semantics of the formulae in $\text{Sk}[S](X, \alpha)$, with respect to a memory state (s, h) .

Proposition 5.31. Let $\alpha \geq 1$. Given (s, h) , let S be the set of all non-empty sets in

$$\left\{ \text{Pred}[S]_{s,h}^X(x), \text{Path}[S]_{s,h}^X(\ell), [\text{Cycl}[S]_{s,h}^X(\beta)]^\flat, [\text{↑Cycl}[S]_{s,h}^{X,\alpha}]^\flat, \text{Rem}[S]_{s,h}^{X,\alpha} \mid x \in X, \ell \in \text{Lab}[S]_{s,h}^X, \beta \in [1, \alpha] \right\}.$$

S forms a partition of $\text{dom}(h)$.

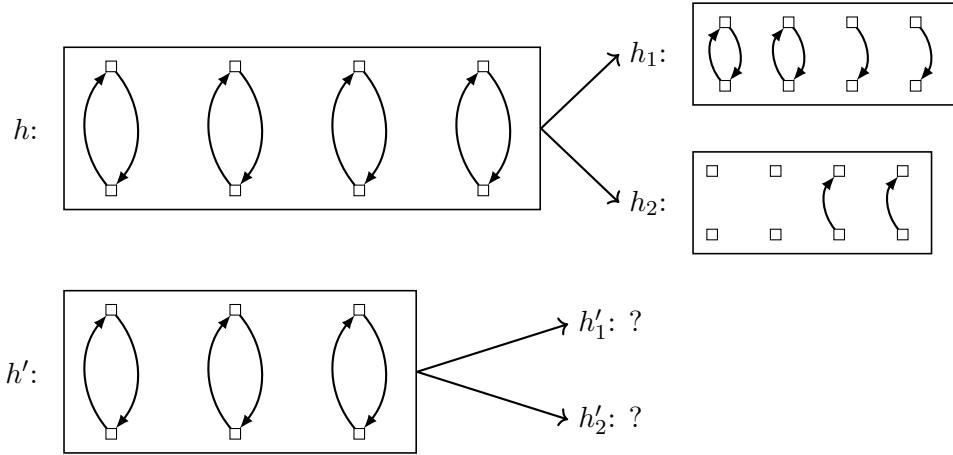
5.5.2 Step II: the core formulae for s .

Following the steps conducted for the weak fragment, the set of core formulae $\text{Core}[S](X, \alpha)$ is parametric on X and a natural number $\alpha \geq 1$, and it is split into a *skeleton set* $\text{Sk}[S](X, \alpha)$ and an *observed set* $\text{Obs}[S](X, \alpha)$. The syntax of the formulae in the skeleton set is defined below, whereas their semantics is given in Figure 5.9.

$$\text{Sk}[S](X, \alpha) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} t_1 = t_2, \text{sees}_X(t_1, t_2) \geq \beta^\ddagger, \\ \text{loop}_X^S(\beta) \geq \beta^\circlearrowleft, \text{↑loop}_{X,\alpha}^S \geq \beta^\circlearrowright, \\ \text{pred}_X^S(x) \geq \beta, \text{rem}_{X,\alpha}^S \geq \beta \end{array} \mid \begin{array}{l} \beta^\ddagger \in \left[1, \frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3)\right] \\ \beta^\circlearrowleft \in \left[1, \frac{1}{2}\alpha(\alpha+3)-1\right], \beta \in [1, \alpha] \\ x \in X, t_1, t_2 \in T[S]^X \end{array} \right\}.$$

The core formulae of the skeleton set follow closely the partition of the heap. For instance, the formula $\text{sees}_X(t_1, t_2) \geq \beta$ states that $\llbracket t_2 \rrbracket_{s,h}^X$ is seen by $\llbracket t_1 \rrbracket_{s,h}^X$, and that $\text{Path}[S]_{s,h}^X(t_1)$ describes a path in h of length at least β , going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $\llbracket t_2 \rrbracket_{s,h}^X$.

An interesting aspect of these core formulae lies in the upper bounds given to β , β^\ddagger , and β^\circlearrowleft . When we look at the core formulae of the weak fragment, the natural number β appearing in all the core formulae $\text{pred}_X^W(x) \geq \beta$, $\text{self}_X^W \geq \beta$ and $\text{rem}_X^W \geq \beta$ belongs to the interval $[1, \alpha]$. For the strong fragment, this simplicity is lost as soon as we want to satisfy the $*$ -simulation

Figure 5.10: The two heaps h and h' described in Example 5.32.

property. The following two examples explain why this is the case. In the first example, we show that the $*$ -simulation property fails with trivial upper bounds. The second example explains how the upper bound $\frac{1}{2}\alpha(\alpha+3)-1$ given to β^\circlearrowright , for the core formula $\uparrow\text{loop}_{X,\alpha}^S \geq \beta^\circlearrowright$, is derived.

Example 5.32. Let us restrict $\text{Sk}[S](X, \alpha)$ so that, in the formulae of the form $\text{loop}_X^S(\beta) \geq \beta^\circlearrowright$, $\uparrow\text{loop}_{X,\alpha}^S \geq \beta^\circlearrowright$ and $\text{rem}_{X,\alpha}^S \geq \beta$, the natural numbers β^\circlearrowright and β range in $[1, \alpha]$ (similarly to what it is done for the weak fragment). Consider two memory states (s, h) and (s', h') whose heaps are depicted in Figure 5.10. In particular, h is made of exactly four unlabelled cycles that belong to $\text{Cycl}[S]_{s,h}^X(2)$, whereas h' contains three such cycles. Let us assume that $\alpha = 3$, so that the two memory states satisfy the same core formulae in $\text{Sk}[S](X, \alpha)$ (with respect to the restricted upper bounds). Following the notion of $*$ -simulation (see Lemma 5.6), we partition h into the two heaps h_1 and h_2 represented in the figure, and consider the two positive natural numbers $\alpha_1 = 2$ and $\alpha_2 = 1$, so that $\alpha = \alpha_1 + \alpha_2$. Notice that h_1 contains two of the four unbounded cycles of h , whereas every other unbounded cycle is split so that one of its locations belongs to $\text{dom}(h_1)$, and the other belongs to $\text{dom}(h_2)$. In particular, following Definition 5.29, this means that $\text{card}(\text{Cycl}[S]_{s,h_1}^X(2)) = 2$ and $\text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha_1}) = 2$, whereas $\text{card}(\uparrow\text{Cycl}[S]_{s,h_2}^{X,\alpha_2}) = 0$ and $\text{card}(\text{Rem}[S]_{s,h_2}^{X,\alpha_2}) = 2$. Here, notice that $\alpha_2 = 1$ and so the cycles in h_2 of length 2 belongs to $\uparrow\text{Cycl}[S]_{s,h_2}^{X,\alpha_2}$. Among the (restricted) formulae in $\text{Sk}[S](X, \alpha_1)$, the memory state (s, h_1) satisfies the two core formulae $\text{loop}_X^S(2) \geq 2$ and $\text{rem}_{X,\alpha_1}^S \geq 2$. Among the formulae in $\text{Sk}[S](X, \alpha_2)$, the memory state (s, h_2) satisfies $\text{rem}_{X,\alpha_2}^S \geq 1$, and it does not satisfy $\uparrow\text{loop}_{X,\alpha_2}^S \geq 1$. Now, the $*$ -simulation property requires us to partition h' into two subheaps h'_1 and h'_2 such that, for every $j \in \{1, 2\}$, (s, h_j) and (s', h'_j) satisfy the same core formulae from $\text{Core}[S](X, \alpha_j)$. Unfortunately, it is quite easy to see that this cannot be done. Indeed, since the memory state (s', h'_1) must satisfy $\text{loop}_X^S(2) \geq 2$, two of the three unbounded cycles in h should belong to h'_1 . This leaves us with one unbounded cycle of two locations, which must be split between the two heaps h'_1 and h'_2 in order for (s', h'_2) to satisfy $\text{rem}_{X,\alpha_2}^S \geq 1$. However, this implies $\text{card}(\text{Rem}[S]_{s',h'_1}^{X,\alpha_1}) = 1$ and so, differently from (s, h_1) , $(s', h'_1) \not\models \text{rem}_{X,\alpha_1}^S \geq 2$.

Example 5.33. The previous example shows how simple upper bounds do not work for the core formulae in $\text{Sk}[S](X, \alpha)$. Intuitively, the reason for this is that locations belonging to a set of the partition can move to a different type of set when considering subheaps. For in-

stance, Example 5.32 shows that given $h_1 \subseteq h$, locations in $\text{Cycl}[s]_{s,h}^{\mathbb{X}}(2)$ can be found in the sets $\text{Cycl}[s]_{s,h_1}^{\mathbb{X}}(2)$ and $\text{Rem}[s]_{s,h_1}^{\mathbb{X},\alpha_1}$. In order to correctly define a family of core formulae that enjoys the $*$ -simulation property we need to take care of these dependencies between types of sets. This leads to the non-trivial upper bounds used in the definition of $\text{Sk}[s](\mathbb{X}, \alpha)$. As an example, let us informally describe how to derive the upper bound $\frac{1}{2}\alpha(\alpha + 3) - 1$ given to β^\circlearrowright , for the core formula $\uparrow\text{loop}_{\mathbb{X},\alpha}^S \geq \beta^\circlearrowright$. As required in the $*$ -simulation property, we consider a memory state (s, h) , two heaps h_1 and h_2 such that $h = h_1 + h_2$ and two natural numbers $\alpha_1, \alpha_2 \geq 1$ such that $\alpha = \alpha_1 + \alpha_2$. We study how the satisfaction of the core formulae changes when the heap h is split into h_1 and h_2 by looking at the possible partitions of these two subheaps. When considering the set $\uparrow\text{Cycl}[s]_{s,h}^{\mathbb{X},\alpha}$, for each set $L \in \uparrow\text{Cycl}[s]_{s,h}^{\mathbb{X},\alpha}$ we either have $L \subseteq \text{dom}(h_j)$, with $j \in \{1, 2\}$, or we have that L is divided into two non-empty sets $L_1 \subseteq \text{dom}(h_1)$ and $L_2 \subseteq \text{dom}(h_2)$. In the first case, L corresponds to an unbounded and unlabelled cycle inside h_i , and so by definition L belongs to $\uparrow\text{Cycl}[s]_{s,h_i}^{\mathbb{X},\alpha_i}$. In the second case, the cycle is split between h_1 and h_2 , causing L_1 to be a subset of $\text{Rem}[s]_{s,h_1}^{\mathbb{X},\alpha_1}$ and L_2 to be a subset of $\text{Rem}[s]_{s,h_2}^{\mathbb{X},\alpha_2}$. Every cycle in $\uparrow\text{Cycl}[s]_{s,h}^{\mathbb{X},\alpha}$ witnesses one among these two cases. Because of the semantics of the core formulae, this means that the locations in L affect either the satisfaction of the core formulae of the form $\uparrow\text{loop}_{\mathbb{X},\alpha_j}^S \geq \beta_j$, for exactly one $j \in \{1, 2\}$ (first case), or the satisfaction of the core formulae $\text{rem}_{\mathbb{X},\alpha_1}^S \geq \beta_1$ and $\text{rem}_{\mathbb{X},\alpha_2}^S \geq \beta_2$ (second case). Let us write $\mathcal{L}(\alpha)$ for the upper bound on β for the core formulae of the form $\uparrow\text{loop}_{\mathbb{X},\alpha}^S \geq \beta$. Similarly, we write $\mathcal{R}(\alpha)$ for the upper bound on β for the core formulae of the form $\text{rem}_{\mathbb{X},\alpha}^S \geq \beta$. We assume that $\mathcal{R}(\alpha) = \alpha$ (this equality can be shown analysing $\text{rem}_{\mathbb{X},\alpha}^S \geq \beta$ with similar arguments to the ones herein described for $\uparrow\text{loop}_{\mathbb{X},\alpha}^S \geq \beta^\circlearrowright$). To simulate the separating conjunct, $\mathcal{L}(\alpha)$ must be at least the sum of the upper bounds $\mathcal{L}(\alpha_1)$ and $\mathcal{L}(\alpha_2)$ (as required by the first case above), plus the upper bound $\mathcal{R}(\max(\alpha_1, \alpha_2))$ (as required by the second case). Therefore, we expect the following inequality to hold, for all $\alpha \geq 1$:

$$\mathcal{L}(\alpha) \geq \max_{\substack{\alpha_1, \alpha_2 \geq 1 \\ \alpha = \alpha_1 + \alpha_2}} (\mathcal{L}(\alpha_1) + \mathcal{L}(\alpha_2) + \mathcal{R}(\max(\alpha_1, \alpha_2))) + 1.$$

Here, the last addend 1 is introduced to handle the quantified variable u , which could belong to one of the unlabelled cycles. In particular, the inequality entails that $\mathcal{L}(\alpha)$ must be at least $\mathcal{L}(\alpha - 1) + \mathcal{L}(1) + \mathcal{R}(\alpha - 1) + 1$, which corresponds to the case where $\alpha_1 = \alpha - 1$ and $\alpha_2 = 1$ (or vice versa). Afterwards, by assuming $\mathcal{L}(1) \geq 1$, we can solve the recurrence system

$$\{\mathcal{L}(1) = 1, \quad \mathcal{L}(\alpha + 1) = \mathcal{L}(\alpha) + \mathcal{L}(1) + \mathcal{R}(\alpha) + 1, \quad \mathcal{R}(\alpha) = \alpha\},$$

in order to conclude that $\mathcal{L}(\alpha)$ must be at least $\frac{1}{2}\alpha(\alpha + 3) - 1$, i.e. the upper bound given to β^\circlearrowright in the core formulae of the form $\uparrow\text{loop}_{\mathbb{X},\alpha}^S \geq \beta^\circlearrowright$. Fundamentally, even though the recurrence system is obtained by considering the case where $\alpha_1 = \alpha - 1$ and $\alpha_2 = 1$, one can show that if $\mathcal{L}(\alpha) = \frac{1}{2}\alpha(\alpha + 3) - 1$ then the expression $\mathcal{L}(\alpha_1) + \mathcal{L}(\alpha_2) + \mathcal{R}(\max(\alpha_1, \alpha_2))$ is indeed maximal for these two values of α_1 and α_2 , so that also the original inequality on $\mathcal{L}(\alpha)$ is satisfied. Clearly, our analysis is incomplete: we now suspect that $\mathcal{L}(\alpha)$ must be at least $\frac{1}{2}\alpha(\alpha + 3) - 1$ in order for the $*$ -simulation property to hold, but we still do not know whether $\mathcal{L}(\alpha) = \frac{1}{2}\alpha(\alpha + 3) - 1$ is sufficient. This will be answered in due time, when proving the $*$ -simulation property. All the upper bounds in $\text{Sk}[s](\mathbb{X}, \alpha)$ are derived in a similar way: by studying how the locations in a set of the partition impact the satisfaction of the core formulae, once a heap is split into two subheaps. Further details about these upper bounds are given in Appendix C.

Let us now move to the core formulae in the observed set $\text{Obs}[s](\mathbb{X}, \alpha)$, defined below. Their

$(s, h) \models u = t$	iff $s(u) = \llbracket t \rrbracket_{s,h}^X,$
$(s, h) \models u \in \text{loop}_X^S(\beta)$	iff there is a set $L \in \text{Cycl}[s]_{s,h}^X(\beta)$ such that $s(u) \in L,$
$(s, h) \models u \in \uparrow\text{loop}_{X,\alpha}^S$	iff there is a set $L \in \uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}$ such that $s(u) \in L,$
$(s, h) \models u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$	iff there are $\delta_1 \geq \overleftarrow{\beta}$ and $\delta_2 \geq \overrightarrow{\beta}$ such that $\delta_1 + \delta_2 = \text{card}(\text{Path}[s]_{s,h}^X(t_1)),$ $h^{\delta_1}(\llbracket t_1 \rrbracket_{s,h}^X) = s(u)$ and $h^{\delta_2}(s(u)) = \llbracket t_2 \rrbracket_{s,h}^X,$
$(s, h) \models u \in \text{pred}_X^S(x)$	iff $s(u) \in \text{Pred}[s]_{s,h}^X(x),$
$(s, h) \models u \in \text{rem}_{X,\alpha}^S$	iff $s(u) \in \text{Rem}[s]_{s,h}^{X,\alpha}.$

Figure 5.11: Semantics of the formulae in $\text{Obs}[s](X, \alpha)$, with respect to a memory state $(s, h).$

semantics is given in Figure 5.11, with respect to a memory state $(s, h).$

$$\text{Obs}[s](X, \alpha) \stackrel{\text{def}}{=} \begin{cases} u = t_1, u \in \text{loop}_X^S(\beta), u \in \uparrow\text{loop}_{X,\alpha}^S, & \overleftarrow{\beta} \in \left[1, \frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1\right] \\ u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta}), & \overrightarrow{\beta} \in \left[1, \frac{1}{2}\alpha(\alpha+3)\right], \beta \in [1, \alpha] \\ u \in \text{pred}_X^S(x), u \in \text{rem}_{X,\alpha}^S & x \in X, t_1, t_2 \in T[s]^X \end{cases}.$$

Intuitively, the core formulae in the observed set deals with the membership of $s(u)$ to the sets of the partition. Particularly interesting is the formula $u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$ which not only states that $s(u)$ belongs to $\text{Path}[s]_{s,h}^X(t_1)$, but also that the path described by this set goes from $\llbracket t_1 \rrbracket_{s,h}^X$ to $\llbracket t_2 \rrbracket_{s,h}^X$ and can be split into two paths, one of length at least $\overleftarrow{\beta}$ and going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $s(u)$, and one of length at least $\overrightarrow{\beta}$ and going from $s(u)$ to $\llbracket t_2 \rrbracket_{s,h}^X$. Fundamentally, the upper bounds $\frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1$ and $\frac{1}{2}\alpha(\alpha+3)$ given to $\overleftarrow{\beta}$ and $\overrightarrow{\beta}$, respectively, sum to $\frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3)$, which is the upper bound given to β^\downarrow in the core formulae of the form $\text{sees}_X(t_1, t_2) \geq \beta^\downarrow$. As we will see, this property is essential in order to prove that the core formulae $\text{Core}[s](X, \alpha)$ enjoy the \exists -simulation property.

Following the outline of the weak fragment, we use the core formulae $\text{Core}[s](X, \alpha)$ in order to define an indistinguishability relation on memory states that is governed by their satisfaction.

Definition 5.34 (s -indistinguishable memory states). We write $\approx_{X,\alpha}^S$ for the equivalence relation on memory states characterised as:

$$(s, h) \approx_{X,\alpha}^S (s', h') \text{ if and only if for every } \varphi \in \text{Core}[s](X, \alpha), (s, h) \models \varphi \text{ iff } (s', h') \models \varphi.$$

As the strong fragment is a syntactical extension of the weak fragment, we expect the indistinguishability relation $\approx_{X,\alpha}^S$ to be a refinement of $\approx_{X,\alpha}^W$. This is indeed the case, as in the next lemma we show that every core formula in $\text{Core}[w](X, \alpha)$ corresponds to a Boolean combination of core formulae from $\text{Core}[s](X, \alpha)$. The proof of this result is quite long but nonetheless interesting, as it forces us to express the sets of the partition introduced for the weak fragment in terms of the ones introduced for the strong fragment. Readers that are eager to see the proof of the $*$ -simulation property for $\text{Core}[s](X, \alpha)$ can skip to page 170.

Lemma 5.35. Let $\alpha \geq 1$. Every core formula in $\text{Core}[\mathcal{W}](\mathbf{X}, \alpha)$ is equivalent to a Boolean combination of formulae in $\text{Core}[\mathcal{S}](\mathbf{X}, \alpha)$.

Proof. First, let us introduce some useful shortcuts using core formulae in $\text{Core}[\mathcal{S}](\mathbf{X}, \alpha)$:

Formula:	Definition:
$\text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') = \beta$	$\text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') \geq \beta \wedge \neg \text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') \geq \beta + 1$
$\mathbf{t} =_{\mathcal{S}} \mathbf{n}(\mathbf{x})$	$\text{sees}_{\mathbf{X}}(\mathbf{x}, \mathbf{t}) = 1$
$\mathbf{t} \hookrightarrow_{\mathcal{S}} \mathbf{x}$	$\text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{x}) = 1$
$\text{seeslen}_{\mathbf{X}}(\mathbf{t}) \geq \beta$	$\bigvee_{\mathbf{t}' \in \mathbf{T}[\mathcal{S}]^{\mathbf{X}}} \text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') \geq \beta$
$\text{alloc}_{\mathcal{S}}(\mathbf{t})$	$\text{seeslen}_{\mathbf{X}}(\mathbf{t}) \geq 1$

where $\mathbf{t}, \mathbf{t}' \in \mathbf{T}[\mathcal{S}]^{\mathbf{X}}$ and $\beta \in [1, \frac{1}{6}(\alpha + 1)(\alpha + 2)(\alpha + 3) - 1]$. Notice that all these formulae are Boolean combinations of core formulae from $\text{Core}[\mathcal{S}](\mathbf{X}, \alpha)$. In particular, the formula $\text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') = 1$ requires the core formulae $\text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') \geq 1$ and $\text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') \geq 2$ in order to be defined. As the upper bound given to β in the formulae of the form $\text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') \geq \beta$ is at least 4 (it is 4 when $\alpha = 1$), these formulae belong to $\text{Core}[\mathcal{S}](\mathbf{X}, \alpha)$, for all $\alpha \geq 1$. Let (s, h) be a memory state. One can check that the formulae above have the following semantics:

$$\begin{aligned} (s, h) \models \text{sees}_{\mathbf{X}}(\mathbf{t}, \mathbf{t}') = \beta &\quad \text{iff } \mathbf{sby}_{s,h}^{\mathbf{X}}(\mathbf{t}) = \llbracket \mathbf{t}' \rrbracket_{s,h}^{\mathbf{X}} \text{ and } \text{card}(\mathbf{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{t})) = \beta, \\ (s, h) \models \mathbf{t} =_{\mathcal{S}} \mathbf{n}(\mathbf{x}) &\quad \text{iff } h(s(\mathbf{x})) = \llbracket \mathbf{t} \rrbracket_{s,h}^{\mathbf{X}}, \\ (s, h) \models \mathbf{t} \hookrightarrow_{\mathcal{S}} \mathbf{x} &\quad \text{iff } h(\llbracket \mathbf{t} \rrbracket_{s,h}^{\mathbf{X}}) = s(\mathbf{x}), \\ (s, h) \models \text{seeslen}_{\mathbf{X}}(\mathbf{t}) \geq \beta &\quad \text{iff } \text{card}(\mathbf{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{t})) \geq \beta, \\ (s, h) \models \text{alloc}_{\mathcal{S}}(\mathbf{t}) &\quad \text{iff } \llbracket \mathbf{t} \rrbracket_{s,h}^{\mathbf{X}} \in \text{dom}(h). \end{aligned}$$

Given a formula φ in $\text{Core}[\mathcal{W}](\mathbf{X}, \alpha)$, we write $\tau(\varphi)$ for its equivalent Boolean combination of core formulae from $\text{Core}[\mathcal{S}](\mathbf{X}, \alpha)$ (to be defined below). Let us start the proof by characterising every formula φ in $\text{Core}[\mathcal{W}](\mathbf{X}, \alpha)$ that is an alloc, points-to or equality between terms.

case: $\varphi = \mathbf{x} \hookrightarrow _$. As $\mathbf{x} \in \mathbf{T}[\mathcal{S}]^{\mathbf{X}}$, $\tau(\mathbf{x} \hookrightarrow _) = \text{alloc}_{\mathcal{S}}(\mathbf{x})$.

case: $\varphi = \mathbf{n}(\mathbf{x}) \hookrightarrow _$. φ is equivalent to $\text{seeslen}_{\mathbf{X}}(\mathbf{x}) \geq 2 \vee \bigvee_{\mathbf{t} \in \mathbf{T}[\mathcal{S}]^{\mathbf{X}}} (\mathbf{t} =_{\mathcal{S}} \mathbf{n}(\mathbf{x}) \wedge \text{alloc}_{\mathcal{S}}(\mathbf{t}))$.

(\Rightarrow): Suppose $(s, h) \models \varphi$, and so $h(s(\mathbf{x})) \in \text{dom}(h)$. We divide the proof in two cases, depending on whether or not $\mathbf{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{x})$ contains at least two locations. If this is the case, then $(s, h) \models \tau(\varphi)$ holds directly from $(s, h) \models \text{seeslen}_{\mathbf{X}}(\mathbf{x}) \geq 2$. Otherwise, from $s(\mathbf{x}) \in \text{dom}(h)$ we conclude that $\text{card}(\mathbf{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{x})) = 1$, which in turn implies that $h(s(\mathbf{x})) = \mathbf{sby}_{s,h}^{\mathbf{X}}(\mathbf{x})$. Hence, there is a term $\mathbf{t} \in \mathbf{T}[\mathcal{S}]^{\mathbf{X}}$ such that $\llbracket \mathbf{t} \rrbracket_{s,h}^{\mathbf{X}} = h(s(\mathbf{x}))$. From $h(s(\mathbf{x})) \in \text{dom}(h)$, we have $(s, h) \models \mathbf{t} =_{\mathcal{S}} \mathbf{n}(\mathbf{x}) \wedge \text{alloc}_{\mathcal{S}}(\mathbf{t})$, and so $(s, h) \models \tau(\varphi)$.

(\Leftarrow): The proof of this direction is straightforward.

case: $\varphi = \mathbf{x} = \mathbf{y}$. In this case, φ belongs to $\text{Sk}[\mathcal{S}](\mathbf{X}, \alpha)$. So, $\tau(\varphi) = \varphi$.

case: $\varphi = \mathbf{x} = \mathbf{n}(\mathbf{y})$ or $\varphi = \mathbf{n}(\mathbf{y}) = \mathbf{x}$ or $\varphi = \mathbf{y} \hookrightarrow \mathbf{x}$. φ is equivalent to $\mathbf{x} =_{\mathcal{S}} \mathbf{n}(\mathbf{y})$.

case: $\varphi = \mathbf{n}(\mathbf{x}) = \mathbf{n}(\mathbf{y})$. φ is equivalent to the formula

$$(\mathbf{x} = \mathbf{y} \wedge \text{alloc}_{\mathcal{S}}(\mathbf{x})) \vee \bigvee_{\mathbf{t} \in \mathbf{T}[\mathcal{S}]^{\mathbf{X}}} (\mathbf{t} =_{\mathcal{S}} \mathbf{n}(\mathbf{x}) \wedge \mathbf{t} =_{\mathcal{S}} \mathbf{n}(\mathbf{y})).$$

(\Rightarrow): Suppose $(s, h) \models \mathbf{n}(\mathbf{x}) = \mathbf{n}(\mathbf{y})$ and so $h(s(\mathbf{x})) = h(s(\mathbf{y}))$. If $s(\mathbf{x}) = s(\mathbf{y})$, then we have $(s, h) \models \tau(\varphi)$ directly from $\mathbf{x} = \mathbf{y} \wedge \text{alloc}_{\mathcal{S}}(\mathbf{x})$. Otherwise, let us assume

that $s(x) \neq s(y)$. We argue that $\ell = h(s(x))$ is a labelled location. *Ad absurdum*, suppose that ℓ is not a labelled location. We then conclude that $\ell \in \text{dom}(h)$, as otherwise we have $\llbracket e(x) \rrbracket_{s,h}^X = \ell$. Because of this, we must have $\ell \in \text{Path}[s]_{s,h}^X(x)$, and with symmetrical arguments we derive $\ell \in \text{Path}[s]_{s,h}^X(x)$. However, this is contradictory as $s(x) \neq s(y)$ implies $\text{Path}[s]_{s,h}^X(x) \cap \text{Path}[s]_{s,h}^X(y) = \emptyset$ by Proposition 5.31. Therefore ℓ is a labelled location, and therefore there is a term $t \in T[s]^X$ such that $\llbracket t \rrbracket_{s,h}^X = \ell$. We conclude that $(s, h) \models t =_S n(x) \wedge t =_S n(y)$, and so $(s, h) \models \tau(\varphi)$.

(\Leftarrow): The proof of this direction is straightforward.

case: $\varphi = n(x) \hookrightarrow y$. φ is equivalent to $\text{sees}_X(x, y) = 2 \vee \bigvee_{t \in T[s]^X} (t =_S n(x) \wedge t \hookrightarrow_S y)$.

We omit the proof of $\varphi \equiv \tau(\varphi)$, which is quite similar to the one given for $\varphi = n(x) \hookrightarrow \dots$.

case: $\varphi = n(x) \hookrightarrow n(x)$. φ is equivalent to $x =_S n(x) \vee (e(x) =_S n(x) \wedge \text{sees}_X(e(x), e(x)) = 1)$.

(\Rightarrow): Suppose $(s, h) \models \varphi$, and so $h(s(x)) = h^2(s(x))$. If $s(x) = h(s(x))$ then (s, h) satisfies $x =_S n(x)$ and so $(s, h) \models \tau(\varphi)$. Otherwise, let us assume that $s(x) \neq h(s(x))$. In this case, given $\ell = h(s(x))$ we have $\{s(x) \mapsto \ell, \ell \mapsto \ell\} \subseteq h$. From the semantics of end-point variables, we conclude that $\ell = \llbracket e(x) \rrbracket_{s,h}^X$. So, $(s, h) \models e(x) =_S n(x) \wedge \text{sees}_X(e(x), e(x)) = 1$, which implies $(s, h) \models \tau(\varphi)$.

(\Leftarrow): The proof of this direction is straightforward.

case: $\varphi = u = x$. As $x \in T[s]^X$, φ belongs to $\text{Core}[s](X, \alpha)$ (for every $\alpha \geq 1$). Thus, $\tau(\varphi) = \varphi$.

case: $\varphi = u = n(x)$. φ is equivalent to the formula

$$\bigvee_{t \in T[s]^X} ((u = t \wedge t =_S n(x)) \vee (u \in \text{sees}_X(x, t) \geq (1, 1) \wedge \neg u \in \text{sees}_X(x, t) \geq (2, 1))).$$

Essentially, $\tau(u = n(x))$ is split depending on whether $s(u)$ is a labelled location with respect to the terms in $T[s]^X$. One can check that this formula is a Boolean combination of core formulae that belong to $\text{Core}[s](X, \alpha)$, for every $\alpha \geq 1$. In particular, this holds true for the subformula $u \in \text{sees}_X(x, t) \geq (2, 1)$, as the upper bounds given to β_1 and β_2 in the formulae of the form $u \in \text{sees}_X(t, t') \geq (\beta_1, \beta_2)$ are always at least 2. We sketch the left-to-right direction of this equivalence, and leave the right-to-left direction to the reader.

(\Rightarrow): Suppose $(s, h) \models \varphi$, and so $h(s(x)) = s(u)$. If there is a term $t \in T[s]^X$ such that $\llbracket t \rrbracket_{s,u}^X = s(x)$, then $(s, h) \models u = t \wedge t =_S n(x)$. Otherwise, as $s(x) \in \text{dom}(h)$, we have $s(u) \in \text{Path}[s]_{s,h}^X(x)$. Let $t \in T[s]^X$ such that $\llbracket t \rrbracket_{s,h}^X = \text{sby}_{s,h}^X(x)$. From $h(s(x)) = s(u)$, we have $(s, h) \models u \in \text{sees}_X(x, t) \geq (1, 1) \wedge \neg u \in \text{sees}_X(x, t) \geq (2, 1)$.

We introduce a second set of shortcuts:

Formula:	Definition:
$\text{var}_X(t)$	$\bigvee_{x \in X} t = x$
$\text{next}_X(t)$	$\neg \text{var}_X(t) \wedge \bigvee_{x \in X} t =_S n(x)$
$\text{unlab}(t)$	$t = t \wedge \neg (\text{var}_X(t) \vee \text{next}_X(t))$
$\text{unlab}(u)$	$\bigwedge_{x \in X} (u \neq x \wedge \neg \tau(u = n(x)))$
$\text{var.sby}(t)$	$\bigvee_{x \in X} \text{sees}_X(t, x) \geq 1$
$\text{alldiff}(T)$	$\bigwedge_{\substack{t, t' \in T \\ t \neq t'}} (t = t \wedge t \neq t')$

where $t \in T[s]^X$ and $T \subseteq T[s]^X$. The semantics of these formulae is given below:

- $(s, h) \models \text{var}_X(t)$ iff $\llbracket t \rrbracket_{s,h}^X$ is defined and belongs to $s(X)$,
- $(s, h) \models \text{next}_X(t)$ iff $\llbracket t \rrbracket_{s,h}^X$ is defined and belongs to $\text{Lab}[W]_{s,h}^X \setminus s(X)$,
- $(s, h) \models \text{unlab}(t)$ iff $\llbracket t \rrbracket_{s,h}^X$ is defined and does not belong to $\text{Lab}[W]_{s,h}^X$,
- $(s, h) \models \text{unlab}(u)$ iff $s(u) \notin \text{Lab}[W]_{s,h}^X$,
- $(s, h) \models \text{var.sby}(t)$ iff $\text{sby}_{s,h}^X(t)$ is defined and belongs to $s(X)$,
- $(s, h) \models \text{alldiff}(T)$ iff $\text{card}(\llbracket T \rrbracket_{s,h}^X) = \text{card}(T)$.

Notice that the formulae $\text{var}_X(t)$, $\text{next}_X(t)$ and $\text{unlab}(t)$ relate terms of the strong fragment with labelled locations of the weak fragment. In particular, given a memory state (s, h) , these three formulae ask $\llbracket t \rrbracket_{s,h}^X$ to be defined, and moreover

- $\text{var}_X(t)$ requires that $\llbracket t \rrbracket_{s,h}^X$ corresponds to a program variable,
- $\text{next}_X(t)$ requires that $\llbracket t \rrbracket_{s,h}^X$ corresponds to a next-point variable, but not to a variable,
- $\text{unlab}(t)$ requires that $\llbracket t \rrbracket_{s,h}^X$ is not a labelled location of $\text{Lab}[W]_{s,h}^X$.

When $\llbracket t \rrbracket_{s,h}^X$ is defined, (s, h) always satisfies exactly one among $\text{var}_X(t)$, $\text{next}_X(t)$ and $\text{unlab}(t)$. Before continuing the characterisation of the core formulae in $\text{Core}[W](X, \alpha)$, we prove an intermediate result that, as we will later see, shows which locations in $\text{Path}[s]_{s,h}^X(\ell)$ ($\ell \in \text{Lab}[s]_{s,h}^X$) can ever belong to $\text{Lab}[W]_{s,h}^X$.

($\text{Lab}^{S/W}$) Let (s, h) be a memory state and $\ell \in \text{Lab}[s]_{s,h}^X$. Suppose that there is $x \in X$ such that $\llbracket n(x) \rrbracket_{s,h}^X \in \text{Path}[s]_{s,h}^X(\ell)$ and $s(x) \notin \text{Path}[s]_{s,h}^X(\ell)$. Then, $\llbracket n(x) \rrbracket_{s,h}^X = \ell$.

Proof of ($\text{Lab}^{S/W}$). Ad absurdum, suppose $\llbracket n(x) \rrbracket_{s,h}^X \neq \ell$. Below, we show that, together with $\llbracket n(x) \rrbracket_{s,h}^X \in \text{Path}[s]_{s,h}^X(\ell)$ and $s(x) \notin \text{Path}[s]_{s,h}^X(\ell)$, this implies that $\llbracket n(x) \rrbracket_{s,h}^X$ is in $\text{Lab}[s]_{s,h}^X$. However, this is contradictory, as we know that ℓ is the only location in $\text{Path}[s]_{s,h}^X(\ell)$ that belongs to $\text{Lab}[s]_{s,h}^X$ (see Definition 5.29), and therefore $\llbracket n(x) \rrbracket_{s,h}^X = \ell$ must hold. For the proof that $\llbracket n(x) \rrbracket_{s,h}^X \in \text{Lab}[s]_{s,h}^X$, we start by noticing that $\llbracket n(x) \rrbracket_{s,h}^X \in \text{Path}[s]_{s,h}^X(\ell)$, and $\llbracket n(x) \rrbracket_{s,h}^X \neq \ell$ imply not only that $\ell \in \text{dom}(h)$, but also that $\llbracket n(x) \rrbracket_{s,h}^X$ is not the first location of the path described by $\text{Path}[s]_{s,h}^X(\ell)$. So, let ℓ' be the location in $\text{Path}[s]_{s,h}^X(\ell)$ such that $h(\ell') = \llbracket n(x) \rrbracket_{s,h}^X$. As $s(x) \notin \text{Path}[s]_{s,h}^X(\ell)$, $s(x)$ is different from ℓ' . Moreover, $h(s(x)) = \llbracket n(x) \rrbracket_{s,h}^X$ by definition of next-point variable. Recapitulating: $\{s(x) \mapsto \llbracket n(x) \rrbracket_{s,h}^X, \ell' \mapsto \llbracket n(x) \rrbracket_{s,h}^X\} \subseteq h$. The proof of $\llbracket n(x) \rrbracket_{s,h}^X \in \text{Lab}[s]_{s,h}^X$ is divided in the following cases:

$\llbracket n(x) \rrbracket_{s,h}^X$ belongs to a cycle. As h is functional, in this case either $s(x)$ or ℓ' does not belong to a cycle. If $s(x)$ does not belong to a cycle, then $\llbracket n(x) \rrbracket_{s,h}^X$ is the first location reachable from $s(x)$ that belongs to one and, by definition of end-point variable, $\llbracket n(x) \rrbracket_{s,h}^X = \llbracket e(x) \rrbracket_{s,h}^X$. Instead, if ℓ' does not belong to a cycle, then the same holds for ℓ and, since $\ell \in \text{dom}(h)$, we conclude that ℓ does not correspond to an end-point variable. Thus, there are $y, z \in X$ such that either $\llbracket y \rrbracket_{s,h}^X = \ell$ or $\llbracket m(y, z) \rrbracket_{s,h}^X = \ell$. In both cases, we conclude that $\llbracket n(x) \rrbracket_{s,h}^X$ is the first location reachable from $s(y)$ that belongs to a cycle. Thus, by definition of end-point variable, $\llbracket n(x) \rrbracket_{s,h}^X = \llbracket e(y) \rrbracket_{s,h}^X$.

$\llbracket n(x) \rrbracket_{s,h}^X$ does not belong to a cycle. In this case, neither $s(x)$ nor ℓ belong to a cycle. As $\ell \in \text{dom}(h)$, we conclude that ℓ does not correspond to an end-point variable, and thus there are $y, z \in X$ such that either $s(y) = \ell$ or $\llbracket m(y, z) \rrbracket_{s,h}^X = \ell$. In both cases, h witnesses two disjoint non-empty paths, one going from $s(x)$ to $\llbracket n(x) \rrbracket_{s,h}^X$, and one going from $s(y)$ to $\llbracket n(x) \rrbracket_{s,h}^X$. As $\llbracket n(x) \rrbracket_{s,h}^X$ does not belong to a cycle, by the definition of meet-point variables we derive $\llbracket n(x) \rrbracket_{s,h}^X = \llbracket m(y, x) \rrbracket_{s,h}^X$.

From $(\text{Lab}^{\mathcal{S}/\mathcal{W}})$ we derive the three following results:

- (I) If $(s, h) \models \text{unlab}(t)$ then $\text{Path}[s]_{s,h}^X(t)$ does not contain locations in $\text{Lab}[\mathcal{W}]_{s,h}^X$.

Proof of (I). Let $\ell = \llbracket t \rrbracket_{s,h}^X$. From $(s, h) \models \text{unlab}(t)$, ℓ is not assigned to a program variable in X . We know that ℓ is the only location in $\text{Path}[s]_{s,h}^X(\ell)$ that belongs to $\text{Lab}[s]_{s,h}^X$. Therefore, no location in $\text{Path}[s]_{s,h}^X(\ell)$ can be assigned to a program variable in X . *Ad absurdum*, suppose that there is $x \in X$ such that $\llbracket n(x) \rrbracket_{s,h}^X \in \text{Path}[s]_{s,h}^X(\ell)$. By $(\text{Lab}^{\mathcal{S}/\mathcal{W}})$ we conclude that $\llbracket n(x) \rrbracket_{s,h}^X = \ell$. However, this implies $(s, h) \not\models \text{unlab}(t)$, a contradiction. Thus, no location in $\text{Path}[s]_{s,h}^X(\ell)$ corresponds to a next-point variable in $\text{NV}[\mathcal{W}]^X$. So, $\text{Path}[s]_{s,h}^X(\llbracket t \rrbracket_{s,h}^X)$ does not contain locations in $\text{Lab}[\mathcal{W}]_{s,h}^X$.

- (II) If $(s, h) \models \text{next}_X(t)$ then $\text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$ does not contain locations in $\text{Lab}[\mathcal{W}]_{s,h}^X$.

Proof of (II). Let $\ell = \llbracket t \rrbracket_{s,h}^X$. From $(s, h) \models \text{next}_X(t)$, ℓ is not assigned to a program variable in X . As shown in the proof of (I) above, this implies that no location in $\text{Path}[s]_{s,h}^X(\ell)$ can be assigned to a program variable in X . Suppose that there is $x \in X$ such that $\llbracket n(x) \rrbracket_{s,h}^X \in \text{Path}[s]_{s,h}^X(\ell)$. By $(\text{Lab}^{\mathcal{S}/\mathcal{W}})$ we conclude that $\llbracket n(x) \rrbracket_{s,h}^X = \ell$. Thus, $\text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$ does not contain locations in $\text{Lab}[\mathcal{W}]_{s,h}^X$.

- (III) Suppose $(s, h) \models \text{var}_X(t)$. Let $S = \text{if } \llbracket t \rrbracket_{s,h}^X \in \text{dom}(h) \text{ then } \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X)\} \text{ else } \{\llbracket t \rrbracket_{s,h}^X\}$. Then, $\text{Path}[s]_{s,h}^X(t) \setminus S$ does not contain locations in $\text{Lab}[\mathcal{W}]_{s,h}^X$.

Proof of (III). *Ad absurdum*, suppose that there is a location ℓ that belongs to both $\text{Path}[s]_{s,h}^X(t) \setminus S$ and $\text{Lab}[\mathcal{W}]_{s,h}^X$. In particular, this means that $\text{Path}[s]_{s,h}^X(t)$ contains at least 3 distinct locations: $\llbracket t \rrbracket_{s,h}^X$, $h(\llbracket t \rrbracket_{s,h}^X)$ and ℓ . We know that $\llbracket t \rrbracket_{s,h}^X$ is the only location in $\text{Path}[s]_{s,h}^X(t)$ that belongs to a program variable in $X \subseteq T[s]^X$. This fact has two consequences. First, the location ℓ does not correspond to a program variable, which in turn implies $\ell = \llbracket n(x) \rrbracket_{s,h}^X$ for some $x \in X$. Second, as $h(\llbracket t \rrbracket_{s,h}^X) \neq \ell$, this means that $s(x) \notin \text{Path}[s]_{s,h}^X(t)$. However, by $(\text{Lab}^{\mathcal{S}/\mathcal{W}})$, this allows us to derive that $\ell = \llbracket t \rrbracket_{s,h}^X$, a contradiction. Thus, no location in $\text{Lab}[\mathcal{W}]_{s,h}^X$ belongs to $\text{Path}[s]_{s,h}^X(t) \setminus S$.

Let us now resume the proof of Lemma 5.35. It remains to show the characterisation of the formulae $\text{pred}_X^{\mathcal{W}}(x) \geq \beta$, $\text{self}_X^{\mathcal{W}} \geq \beta$ and $\text{rem}_X^{\mathcal{W}} \geq \beta$, as well as the formulae $u \in \text{pred}_X^{\mathcal{W}}(x)$, $u \in \text{self}_X^{\mathcal{W}}$ and $u \in \text{rem}_X^{\mathcal{W}}$. These core formulae need a careful analysis, since they have a different semantics with respect to their counterpart in $\text{Core}[s](X, \alpha)$. Indeed, for example the core formula $\text{pred}_X^{\mathcal{W}}(x) \geq \beta$ only counts the number of predecessors of the location corresponding to x that are not labelled locations, whereas the formula $\text{pred}_X^{\mathcal{S}}(x) \geq \beta$ is more precise and only count those predecessors that are not reached by a program variable (or, equivalently, are not reached by a labelled location). Therefore, $\text{pred}_X^{\mathcal{S}}(x) \geq \beta$ implies $\text{pred}_X^{\mathcal{W}}(x) \geq \beta$, or analogously $\text{Pred}[s]_{s,h}^X(x) \subseteq \text{Pred}[\mathcal{W}]_{s,h}^X(x)$. The inclusion in the other direction does not hold. So, before looking directly at the core formulae, we consider a memory state (s, h) , and define the sets $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$, $\text{Self}[\mathcal{W}]_{s,h}^X$ and $\text{Rem}[\mathcal{W}]_{s,h}^X$ in terms of the labelled locations and sets introduced for the strong fragment.

For $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$, we have the following equivalence:

$$\text{Pred}[\mathcal{W}]_{s,h}^X(x) = \text{Pred}[s]_{s,h}^X(x) \cup \left\{ \ell \in \text{dom}(h) \setminus \text{Lab}[\mathcal{W}]_{s,h}^X \mid \begin{array}{l} h(\ell) = s(x), \ell \in \text{Path}[s]_{s,h}^X(\ell') \\ \text{for some } \ell' \in \text{Lab}[s]_{s,h}^X \end{array} \right\} \quad (\dagger)$$

Indeed, every predecessor ℓ of $s(x)$ is either in $\text{Pred}[s]_{s,h}^X(x)$ or in $\text{Path}[s]_{s,h}^X(\ell')$, for some $\ell' \in \text{Lab}[s]_{s,h}^X$. In this latter case, ℓ belongs to $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$ only if it is not a labelled location in $\text{Lab}[\mathcal{W}]_{s,h}^X$, which leads to the equivalence (\dagger) . To characterise the formulae $\text{pred}_X^{\mathcal{W}}(x) \geq \beta$ and $u \in \text{pred}_X^{\mathcal{W}}(x)$ we follow this equivalence rather closely.

case: $\varphi = \text{pred}_X^W(x) \geq \beta$. φ is equivalent to the formula

$$\bigvee_{\substack{T \subseteq T[S]^X \\ \beta' \in [0, \alpha] \\ \beta' + \text{card}(T) \geq \beta}} \left(\text{pred}_X^S(x) \geq \beta' \wedge \text{alldiff}(T) \wedge \bigwedge_{t \in T} \left(\begin{array}{l} (\text{unlab}(t) \Rightarrow \text{sees}_X(t, x) \geq 1) \\ \wedge (\text{next}_X(t) \Rightarrow \text{sees}_X(t, x) \geq 2) \\ \wedge (\text{var}_X(t) \Rightarrow \text{sees}_X(t, x) \geq 3) \end{array} \right) \right),$$

where we define $\text{pred}_X^S(x) \geq 0$ as the tautology $\text{pred}_X^S(x) \geq 1 \vee \neg \text{pred}_X^S(x) \geq 1$.

Essentially, this formula relies on the fact that $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$ contains at least β locations if and only if there are two integers $\beta', \beta'' \geq 0$ such that $\beta' + \beta'' \geq \beta$ and

- a. $\text{Pred}[S]_{s,h}^X(x)$ contains at least β' locations,
- b. $\left\{ \ell \in \text{dom}(h) \setminus \text{Lab}[\mathcal{W}]_{s,h}^X \mid h(\ell) = s(x), \ell \in \text{Path}[S]_{s,h}^X(\ell') \right\}$ has at least β'' locations.

This double implication holds directly from the equivalence (†). Let us show that $\varphi \equiv \tau(\varphi)$.

(\Rightarrow): Suppose $(s, h) \models \varphi$. Following (†), there are $\beta', \beta'' \geq 0$ such that (a) and (b) hold. Moreover, as $\beta \in [1, \alpha]$, we can assume $\beta' \leq \alpha$. From (a), $(s, h) \models \text{pred}_X^S(x) \geq \beta'$. Given a term $t \in T[S]^X$ such that $\text{sby}_{s,h}^X(t) = s(x)$, we write ℓ_t for the location in $\text{Path}[S]_{s,h}^X(t)$ such that $h(\ell_t) = s(x)$. Informally, ℓ_t is the predecessor of $s(x)$ in the non-empty path going from $\llbracket t \rrbracket_{s,h}^X$ to $s(x)$. Let T be a minimal subset of $T[S]^X$ where, for every location ℓ belonging to the set described in (b), there is a term $t \in X$ such that $\ell = \ell_t$. Thus, given $t \in T$, ℓ_t is not a labelled location of $\text{Lab}[\mathcal{W}]_{s,h}^X$. Moreover, since T is minimal, no two terms in it correspond to the same location, and from (b) we conclude that $\text{card}(T) = \text{card}(\llbracket T \rrbracket_{s,h}^X) \geq \beta''$. This allows us to conclude that $(s, h) \models \text{alldiff}(T)$. Let us look more in depth at $\llbracket t \rrbracket_{s,h}^X$:

1. by definition of T , we know that $\text{sby}_{s,h}^X(t) = s(x)$, and so $(s, h) \models \text{sees}_X(t, x) \geq 1$. This implies that $(s, h) \models \text{unlab}(t) \Rightarrow \text{sees}_X(t, x) \geq 1$,
2. if $\llbracket t \rrbracket_{s,h}^X$ corresponds to a program variable, then $h(\llbracket t \rrbracket_{s,h}^X)$ corresponds to a next-point variable. Since ℓ_t is not a labelled location, we conclude that $\text{Path}[S]_{s,h}^X(t)$ contains at least 3 locations. Thus, $(s, h) \models \text{var}_X(t) \Rightarrow \text{sees}_X(t, x) \geq 3$,
3. if $\llbracket t \rrbracket_{s,h}^X$ corresponds to a next-point variable, then it must be different from ℓ_t , which is an unlabelled location, and therefore $\text{Path}[S]_{s,h}^X(t)$ contains at least 2 locations. Thus, $(s, h) \models \text{next}_X(t) \Rightarrow \text{sees}_X(t, x) \geq 2$.

Therefore, $(s, h) \models \tau(\varphi)$.

(\Leftarrow): Suppose $(s, h) \models \tau(\varphi)$, and thus (s, h) satisfies one disjunct of $\tau(\varphi)$. Let $T \subseteq T[S]^X$ and $\beta' \in [0, \alpha]$, where $\beta' + \text{card}(T) \geq \beta$, be the set of terms and integer that correspond to a satisfied disjunct. Since $(s, h) \models \text{pred}_X^S(x) \geq \beta'$, (a) holds. To conclude the proof, we show that (b) also holds, where $\beta'' = \text{card}(T)$. By $(s, h) \models \text{alldiff}(T)$ we have $\beta'' = \text{card}(\llbracket T \rrbracket_{s,h}^X)$, and therefore in order to show (b) it is sufficient, given a term $t \in T$, to show that there is a location $\ell_t \in \text{Path}[S]_{s,h}^X(t)$ that belongs to the set described in (b). Equivalently, we must show that $h(\ell_t) = s(x)$ and that $\ell_t \notin \text{Lab}[\mathcal{W}]_{s,h}^X$. First of all, from the satisfaction of

$$\begin{aligned} & (\text{unlab}(t) \Rightarrow \text{sees}_X(t, x) \geq 1) \wedge (\text{next}_X(t) \Rightarrow \text{sees}_X(t, x) \geq 2) \\ & \wedge (\text{var}_X(t) \Rightarrow \text{sees}_X(t, x) \geq 3) \end{aligned}$$

we realise that $(s, h) \models \text{sees}_X(t, x) \geq 1$. Indeed, $\llbracket t \rrbracket_{s,h}^X$ is defined thanks to the satisfaction of $\text{alldiff}(T)$, and therefore (s, h) satisfies exactly one formula among $\text{unlab}(t)$, $\text{next}_X(t)$ and $\text{var}_X(t)$. We conclude that $\text{sby}_{s,h}^X(t) = s(x)$, and so there is $\ell_t \in \text{Path}[S]_{s,h}^X(t)$

such that $h(\ell_t) = s(x)$. To conclude, let us show that the location ℓ_t does not belong to $\text{Lab}[\mathcal{W}]_{s,h}^X$. The proof is divided in three cases, following the satisfaction of the formulae $\text{unlab}(t)$, $\text{next}_X(t)$ and $\text{var}_X(t)$:

1. Suppose $(s, h) \models \text{unlab}(t)$. Then $\ell_t \notin \text{Lab}[\mathcal{W}]_{s,h}^X$ holds directly from (I).
2. Suppose $(s, h) \models \text{next}_X(t) \Rightarrow \text{sees}_X(t, x) \geq 2$ we conclude that $\text{Path}[s]_{s,h}^X(t)$ describes a minimal non-empty path of length at least 2, going from $\llbracket t \rrbracket_{s,h}^X$ to $s(x)$. By (II), all locations but $\llbracket t \rrbracket_{s,h}^X$ do not belong to $\text{Lab}[\mathcal{W}]_{s,h}^X$. As $h(\ell_t) = s(x)$ and the path has length at least 2, $\ell_t \neq \llbracket t \rrbracket_{s,h}^X$. Thus, $\ell_t \notin \text{Lab}[\mathcal{W}]_{s,h}^X$.
3. Suppose $(s, h) \models \text{var}_X(t)$. From $(s, h) \models \text{var}_X(t) \Rightarrow \text{sees}_X(t, x) \geq 3$ we conclude that $\text{Path}[s]_{s,h}^X(t)$ describes a minimal non-empty path of length at least 3, going from $\llbracket t \rrbracket_{s,h}^X$ to $s(x)$. By (III), all locations except $\llbracket t \rrbracket_{s,h}^X$ and $h(\llbracket t \rrbracket_{s,h}^X)$ do not belong to $\text{Lab}[\mathcal{W}]_{s,h}^X$. As $h(\ell_t) = s(x)$ and the path has length at least 3, $\llbracket t \rrbracket_{s,h}^X \neq \ell_t \neq h(\llbracket t \rrbracket_{s,h}^X)$. Thus, $\ell_t \notin \text{Lab}[\mathcal{W}]_{s,h}^X$.

case: $\varphi = u \in \text{pred}_X^{\mathcal{W}}(x)$. φ is equivalent to the formula

$$u \in \text{pred}_X^{\mathcal{S}}(x) \vee \left(\text{unlab}(u) \wedge \bigvee_{t \in T[s]^X} \left((u = t \wedge t \hookrightarrow_S x) \vee (u \in \text{sees}_X(t, x) \geq (1, 1) \wedge \neg u \in \text{sees}_X(t, x) \geq (1, 2)) \right) \right).$$

(\Rightarrow): Suppose $(s, h) \models u \in \text{pred}_X^{\mathcal{W}}(x)$, and thus, from (\dagger), either (1) $s(u) \in \text{Pred}[s]_{s,h}^X(x)$ or (2) $s(u) \in \text{dom}(h) \setminus \text{Lab}[\mathcal{W}]_{s,h}^X$, $h(s(u)) = s(x)$, $s(u) \in \text{Path}[s]_{s,h}^X(\ell')$ for some $\ell' \in \text{Lab}[s]_{s,h}^X$. If (1) holds then $(s, h) \models u \in \text{pred}_X^{\mathcal{S}}(x)$, i.e. the left disjunct of $\tau(\varphi)$. For the case (2), we show that (s, h) satisfies the right disjunct of $\tau(\varphi)$. As $s(u) \notin \text{Lab}[\mathcal{W}]_{s,h}^X$, $(s, h) \models \text{unlab}(u)$. Besides, consider a term $t \in T[s]^X$ such that $s(u) \in \text{Path}[s]_{s,h}^X(t)$. If $s(u) = \llbracket t \rrbracket_{s,h}^X$, then $(s, h) \models u = t \wedge t \hookrightarrow_S x$. Else, $h^\delta(\llbracket t \rrbracket_{s,h}^X) = s(u)$ for some $\delta \in [1, \text{card}(\text{Path}[s]_{s,h}^X(t)) - 1]$. By $h(s(u)) = s(x)$, we derive $(s, h) \models u \in \text{sees}_X(t, x) \geq (1, 1) \wedge \neg u \in \text{sees}_X(t, x) \geq (1, 2)$. This concludes the proof.

(\Leftarrow): Suppose $(s, h) \models \tau(\varphi)$. If (s, h) satisfies the left disjunct of $\tau(\varphi)$, we conclude that $s(u) \in \text{Pred}[s]_{s,h}^X(x) \subseteq \text{Pred}[\mathcal{W}]_{s,h}^X(x)$ (inclusion from (\dagger)), and $(s, h) \models u \in \text{pred}_X^{\mathcal{W}}(x)$. Otherwise, let us consider the case where (s, h) satisfies the right disjunct of $\tau(\varphi)$. We show that $s(u)$ belongs to the set

$$L = \left\{ \ell \in \text{dom}(h) \setminus \text{Lab}[\mathcal{W}]_{s,h}^X \mid h(\ell) = s(x), \ell \in \text{Path}[s]_{s,h}^X(\ell') \text{ for some } \ell' \in \text{Lab}[s]_{s,h}^X \right\}.$$

By looking at the right disjunct, we have $s(u) \notin \text{Lab}[\mathcal{W}]_{s,h}^X$. Besides, there is a term $t \in T[s]^X$ such that either $u = t \wedge t \hookrightarrow_S x$ or $u \in \text{sees}_X(t, x) \geq (1, 1) \wedge \neg u \in \text{sees}_X(t, x) \geq (1, 2)$ are satisfied by (s, h) . If $(s, h) \models u = t \wedge t \hookrightarrow_S x$ then $\llbracket t \rrbracket_{s,h}^X = s(u)$ and $h(\llbracket t \rrbracket_{s,h}^X) = s(x)$. This implies $s(u) \in \text{Path}[s]_{s,h}^X(t)$ and $h(s(u)) = s(x)$, which in turn shows that $s(u) \in L$. Otherwise, suppose that $(s, h) \models u \in \text{sees}_X(t, x) \geq (1, 1) \wedge \neg u \in \text{sees}_X(t, x) \geq (1, 2)$. From the left conjunct, there are $\delta_1, \delta_2 \geq 1$ such that $\delta_1 + \delta_2 = \text{card}(\text{Path}[s]_{s,h}^X(t))$, $h^{\delta_1}(\llbracket t \rrbracket_{s,h}^X) = s(u)$ and $h^{\delta_2}(s(u)) = s(x)$. Form the right conjunct, $\delta_2 = 1$. Again, we derive that $s(u) \in L$. By (\dagger), we conclude: $(s, h) \models u \in \text{pred}_X^{\mathcal{W}}$.

Let us now move to the set $\text{Self}[\mathcal{W}]_{s,h}^X$, for which we have the following equivalence:

$$\text{Self}[\mathcal{W}]_{s,h}^X = [\text{Cycl}[s]_{s,h}^X(1)]^b \cup \{\ell \in \text{Lab}[s]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h}^X \mid h(\ell) = \ell\} \quad (\ddagger)$$

Indeed, every self-loop is either in $\text{Cycl}[s]_{s,h}^X(1)$ or it corresponds to a location $\ell \in \text{Lab}[s]_{s,h}^X$ for which $h(\ell) = \ell$. In this latter case, ℓ belongs to $\text{Self}[\mathcal{W}]_{s,h}^X$ if and only if it is not a labelled location of $\text{Lab}[\mathcal{W}]_{s,h}^X$, which leads to the equivalence (\ddagger).

case: $\varphi = \text{self}_X^W \geq \beta$. φ is equivalent to the formula

$$\bigvee_{\substack{T \subseteq T[S]^X \\ \beta' \in [0, \alpha] \\ \beta' + \text{card}(T) \geq \beta}} \left(\text{loop}_X^S(1) \geq \beta' \wedge \text{alldiff}(T) \wedge \bigwedge_{t \in T} (\text{unlab}(t) \wedge \text{sees}_X(t, t) = 1) \right),$$

where we define $\text{loop}_X^S(1) \geq 0$ as the tautology $\text{loop}_X^S(1) \geq 1 \vee \neg \text{loop}_X^S(1) \geq 1$.

Essentially, this formula relies on the fact that $\text{Self}[W]_{s,h}^X$ contains at least β locations if and only if there are two integers $\beta', \beta'' \geq 0$ such that $\beta_1 + \beta_2 \geq \beta$ and

- c. $[\text{Cycl}[S]_{s,h}^X(1)]^b$ contains at least β' locations,
- d. $\{\ell \in \text{Lab}[S]_{s,h}^X \setminus \text{Lab}[W]_{s,h}^X \mid h(\ell) = \ell\}$ contains at least β'' locations.

This double implication holds directly from (‡). Let us show that $\varphi \equiv \tau(\varphi)$.

(\Rightarrow): Suppose $(s, h) \models \varphi$. Following (‡), there are $\beta', \beta'' \geq 0$ such that (c) and (d) hold. Moreover, as $\beta \in [1, \alpha]$, we can assume $\beta' \leq \alpha$. From (c), $(s, h) \models \text{loop}_X^S(1) \geq \beta'$. Let T a minimal subset of $T[S]^X$ where, for every location ℓ belonging to the set described in (d), there is a term $t \in T$ such that $\llbracket t \rrbracket_{s,h}^X = \ell$. As every location in (d) belongs to $\text{Lab}[S]_{s,h}^X$, the set T can be defined. From (c), $\text{card}(\llbracket t \rrbracket_{s,h}^X) \geq \beta''$ and from the minimality of X we conclude that $\text{card}(X) = \text{card}(\llbracket t \rrbracket_{s,h}^X)$. This implies that $(s, h) \models \text{alldiff}(X)$. Lastly, let us show that $(s, h) \models \bigwedge_{t \in T} (\text{unlab}(t) \wedge \text{sees}_X(t, t) = 1)$. By definition, every term $t \in X$ is such that $\llbracket t \rrbracket_{s,h}^X \notin \text{Lab}[W]_{s,h}^X$ and $h(\llbracket t \rrbracket_{s,h}^X) = \llbracket t \rrbracket_{s,h}^X$. As the latter implies $\text{sby}_{s,h}^X(t) = \llbracket t \rrbracket_{s,h}^X$ and $\text{card}(\text{Path}[S]_{s,h}^X(t)) = 1$, we conclude: $(s, h) \models \text{unlab}(t) \wedge \text{sees}_X(t, t) = 1$.

(\Leftarrow): Suppose $(s, h) \models \tau(\varphi)$, and thus (s, h) satisfies one disjunct of $\tau(\varphi)$. Let $T \subseteq T[S]^X$ and $\beta' \in [0, \alpha]$, where $\beta' + \text{card}(T) \geq \beta$, be the set of terms and the integer that correspond to a satisfied disjunct. (c) holds from $(s, h) \models \text{self}_X^W \geq \beta'$, and to conclude the proof we must simply show that (d) also holds, where $\beta'' = \text{card}(T)$. By $(s, h) \models \text{alldiff}(T)$ we have $\beta'' = \text{card}(\llbracket T \rrbracket_{s,h}^X)$, and therefore to show (d) it is sufficient, given $t \in T$, to show that $\llbracket t \rrbracket_{s,h}^X \notin \text{Lab}[W]_{s,h}^X$ and $h(\llbracket t \rrbracket_{s,h}^X) = \llbracket t \rrbracket_{s,h}^X$. The former holds as $(s, h) \models \text{unlab}(t)$, whereas the latter holds from $(s, h) \models \text{sees}_X(t, t) = 1$. Indeed, $h(\llbracket t \rrbracket_{s,h}^X) = \llbracket t \rrbracket_{s,h}^X$ is equivalent to $\text{sby}_{s,h}^X(t) = \llbracket t \rrbracket_{s,h}^X$ and $\text{card}(\text{Path}[S]_{s,h}^X(t)) = 1$.

case: $\varphi = u \in \text{self}_X^W$. φ is equivalent to the formula

$$u \in \text{loop}_X^S(1) \vee \left(\text{unlab}(u) \wedge \bigvee_{t \in T[S]^X} (u = t \wedge \text{sees}_X(t, t) = 1) \right).$$

(\Rightarrow): Suppose $(s, h) \models \varphi$, and thus, from (‡), we have either (1) $s(u) \in [\text{Cycl}[S]_{s,h}^X(1)]^b$, or (2) $s(u) \in \text{Lab}[S]_{s,h}^X \setminus \text{Lab}[W]_{s,h}^X$ and $h(s(u)) = s(u)$. If (1) then $(s, h) \models u \in \text{loop}_X^S(1)$. For the case (2), we show that (s, h) satisfies the right disjunct of $\tau(\varphi)$. As $s(u) \notin \text{Lab}[W]_{s,h}^X$, $(s, h) \models \text{unlab}(u)$. As $s(u) \in \text{Lab}[S]_{s,h}^X$, there is a term $t \in T[S]^X$ such that $s(u) = \llbracket t \rrbracket_{s,h}^X$. Thus, $h(\llbracket t \rrbracket_{s,h}^X) = \llbracket t \rrbracket_{s,h}^X$, which allows us to conclude that $(s, h) \models u = t \wedge \text{sees}_X(t, t) = 1$.

(\Leftarrow): Suppose $(s, h) \models \tau(\varphi)$. If (s, h) satisfies the left disjunct of $\tau(\varphi)$, we conclude that $s(u) \in [\text{Cycl}[S]_{s,h}^X(1)]^b \subseteq \text{Self}[W]_{s,h}^X$ (inclusion from (‡)), and $(s, h) \models u \in \text{self}_X^W$. Otherwise, let us consider the case where (s, h) satisfies the right disjunct of $\tau(\varphi)$, and so there is a term $t \in T[S]^X$ such that $(s, h) \models \text{unlab}(u) \wedge u = t \wedge \text{sees}_X(t, t) = 1$. Clearly, $s(u)$ belongs to the set $\{\ell \in \text{Lab}[S]_{s,h}^X \setminus \text{Lab}[W]_{s,h}^X \mid h(\ell) = \ell\}$. Indeed, the satisfaction of $\text{unlab}(u)$ implies $s(u) \notin \text{Lab}[W]_{s,h}^X$, whereas $(s, h) \models u = t$ means that $s(u) = \llbracket t \rrbracket_{s,h}^X \in \text{Lab}[S]_{s,h}^X$. Moreover, from the satisfaction of $\text{sees}_X(t, t) = 1$ we conclude that $h(s(u)) = s(u)$. Lastly, by (‡), $s(u) \in \text{Self}[W]_{s,h}^X$, and therefore $(s, h) \models u \in \text{self}_X^W$.

The characterisation of the two last formulae, i.e. $\text{rem}_X^W \geq \beta$ and $u \in \text{rem}_X^W$, is quite different. Let us first discuss the case of $u \in \text{rem}_X^W$. First, we define a Boolean combination of core formulae in $\text{Core}[\mathcal{S}](X, \alpha)$ that is semantically equivalent to $u \hookrightarrow \perp$. Similarly to $\text{alloc}_{\mathcal{S}}(t)$ defined at the beginning of the proof, we write $\text{alloc}_{\mathcal{S}}(u)$ for this formula, which is defined as

$$\begin{aligned} u \in \text{rem}_{X,\alpha}^{\mathcal{S}} \vee u \in \uparrow\text{loop}_{X,\alpha}^{\mathcal{S}} \vee \bigvee_{\beta \in [1,\alpha]} u \in \text{loop}_X^{\mathcal{S}}(\beta) \\ \vee \bigvee_{t,t' \in T[\mathcal{S}]^X} ((u = t \wedge \text{alloc}_{\mathcal{S}}(t)) \vee u \in \text{sees}_X(t, t') \geq (1, 1)) \end{aligned}$$

The correctness of $\text{alloc}_{\mathcal{S}}(u)$ stems directly from the fact that the sets in Definition 5.29 form a partition of the domain of the heap, according to Proposition 5.31.

case: $\varphi = u \in \text{rem}_X^W$. Thanks to the formula $\text{alloc}_{\mathcal{S}}(u)$, we can define $\tau(\varphi)$ as the formula

$$\text{alloc}_{\mathcal{S}}(u) \wedge \bigwedge_{x \in X} \neg \tau(u \in \text{pred}_X^W(x)) \wedge \neg \tau(u \in \text{self}_X^W) \wedge \bigwedge_{t \in T[W]^X} \neg \tau(u = t).$$

Essentially, this formula follows the definition of $\text{Rem}[W]_{s,h}^X$ given in Definition 5.12:

$$\text{Rem}[W]_{s,h}^X \stackrel{\text{by def}}{=} \text{dom}(h) \setminus \left(\text{Lab}[W]_{s,h}^X \cup \text{Sel}\text{f}[W]_{s,h}^X \cup \bigcup_{x \in X} \text{Pred}[W]_{s,h}^X(x) \right).$$

Indeed, $\varphi \equiv \tau(\varphi)$ follows directly from the double implications below:

$$\begin{aligned} s(u) \in \text{dom}(h) &\quad \text{iff } (s, h) \models \text{alloc}_{\mathcal{S}}(u), \\ s(u) \notin \text{Lab}[W]_{s,h}^X &\quad \text{iff } (s, h) \models \bigwedge_{t \in T[W]^X} \neg \tau(u = t), \\ s(u) \notin \bigcup_{x \in X} \text{Pred}[W]_{s,h}^X(x) &\quad \text{iff } (s, h) \models \bigwedge_{x \in X} \neg \tau(u \in \text{pred}_X^W), \\ s(u) \notin \text{Sel}\text{f}[W]_{s,h}^X &\quad \text{iff } (s, h) \models \neg \tau(u \in \text{self}_X^W). \end{aligned}$$

Lastly, we characterise the formula $\text{rem}_X^W \geq \beta$. To do so, we first study how the set $\text{Rem}[W]_{s,h}^X$ is described in terms of sets of the partition of the strong fragment. We have:

$$\text{Rem}[W]_{s,h}^X = \text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha} \cup \bigcup_{\beta \in [2,\alpha]} [\text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)]^b \cup [\uparrow\text{Cycl}[\mathcal{S}]_{s,h}^{X,\alpha}]^b \cup \bigcup_{\ell \in \text{Lab}[\mathcal{S}]_{s,h}^X} (\text{Path}[\mathcal{S}]_{s,h}^X(\ell) \cap \text{Rem}[W]_{s,h}^X) \quad (\star)$$

Notice that the sets in the right-hand side of this equivalence are all two by two disjoint (by Proposition 5.31). It is quite easy to see that if a location ℓ belongs to either $\text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}$ or to an unlabelled cycle of length greater than two, then $\ell \in \text{Rem}[W]_{s,h}^X$. So, the correctness of (\star) is based on the fact that $[\text{Cycl}[\mathcal{S}]_{s,h}^X(1)]^b \cap \text{Rem}[W]_{s,h}^X = \emptyset$ and that, together with paths sets of the form $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$, these sets cover $\text{dom}(h)$. In the equivalence (\star) , the subsets of the form $\text{Path}[\mathcal{S}]_{s,h}^X(\ell) \cap \text{Rem}[W]_{s,h}^X$ are quite unsatisfactory, as they do not tell us which locations of $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$ are in $\text{Rem}[W]_{s,h}^X$. We study these locations. We recall that, when non-empty, $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$ describes a path going from ℓ to the location $\text{sby}_{s,h}^X(\ell) \in \text{Lab}[\mathcal{S}]_{s,h}^X$. Moreover, ℓ is the only location in $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$ that belongs to $\text{Lab}[\mathcal{S}]_{s,h}^X$. If $h(\ell) = \ell$, then $\text{Path}[\mathcal{S}]_{s,h}^X(\ell) = \{\ell\}$ and, as ℓ points to itself, we have $\text{Path}[\mathcal{S}]_{s,h}^X(\ell) \cap \text{Rem}[W]_{s,h}^X = \emptyset$. Otherwise, suppose $h(\ell) \neq \ell$. From the three intermediate result (I), (II) and (III), we know that the only two locations of $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$ that can possibly belong to $\text{Lab}[\mathcal{S}]_{s,h}^X$ are ℓ and $h(\ell)$. As self-loops are excluded, from Proposition 5.13 we conclude that every other location of $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$ belongs to either $\text{Rem}[W]_{s,h}^X$ or $\text{Pred}[W]_{s,h}^X(x)$, for some $x \in X$. Moreover, as $s(x)$ is a labelled location, the only location that can ever belong to $\text{Pred}[W]_{s,h}^X(x)$ is the one that precedes $\text{sby}_{s,h}^X(\ell)$. Formally, let l_{pre} be the only location in $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$ such that $h(l') = \text{sby}_{s,h}^X(\ell)$. We have, $l_{pre} \in \text{Pred}[W]_{s,h}^X(x)$ if and only if $s(x) = \text{sby}_{s,h}^X(\ell)$. If $\text{sby}_{s,h}^X(\ell)$ does not correspond to a program variable, then $l_{pre} \in \text{Rem}[W]_{s,h}^X$. Recapitulating, every location in $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$ that is not ℓ , $h(\ell)$ and l_{pre} necessarily belongs to $\text{Rem}[W]_{s,h}^X$. Instead, the membership of ℓ , $h(\ell)$ and l_{pre} to $\text{Rem}[W]_{s,h}^Y$ depend on (I), (II) (III) and on whether or not $\text{sby}_{s,h}^X(\ell)$ corresponds to a program variable. This leads to six different cases, depicted in Figure 5.12 (where, for simplicity, we

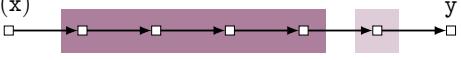
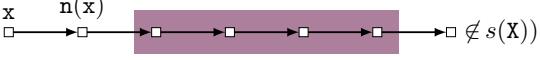
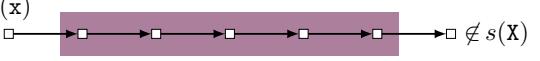
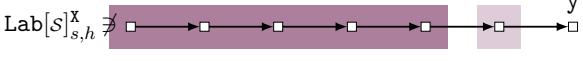
Pattern described by $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$:	$\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(\ell) \setminus \text{Rem}[\mathcal{W}]_{s,h}^X)$:
	3
	2
	2
	1
	1
	0
$\blacksquare : \text{Rem}[\mathcal{W}]_{s,h}^X$	$\square : \text{Pred}[\mathcal{W}]_{s,h}^X(y)$

Figure 5.12: Case analysis for the formula $\tau(\text{rem}_X^{\mathcal{W}} \geq \beta)$, assuming $\text{Path}[\mathcal{S}]_{s,h}^X(\ell) \cap \text{Rem}[\mathcal{W}]_{s,h}^X \neq \emptyset$.

assume $\text{Path}[\mathcal{S}]_{s,h}^X(\ell) \cap \text{Rem}[\mathcal{W}]_{s,h}^X \neq \emptyset$. Below, suppose that $\text{Path}[\mathcal{S}]_{s,h}^X(\ell) \neq \emptyset$, and that l_{pre} is the only location $\text{Path}[\mathcal{S}]_{s,h}^X(\ell)$ such that $h(\ell') = \text{sby}_{s,h}^X(\ell)$. Given a term $t \in T[\mathcal{S}]^X$ these six cases are formalised below, where \mathcal{R} is short for $\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X$.

- A. If $(s, h) \models \text{var}_X(t) \wedge \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X), l_{\text{pre}}\}$,
- B. if $(s, h) \models \text{next}_X(t) \wedge \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, l_{\text{pre}}\}$,
- C. if $(s, h) \models \text{var}_X(t) \wedge \neg \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X)\}$,
- D. if $(s, h) \models \text{next}_X(t) \wedge \neg \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$,
- E. if $(s, h) \models \text{unlab}(t) \wedge \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{l_{\text{pre}}\}$,
- F. if $(s, h) \models \text{unlab}(t) \wedge \neg \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[\mathcal{S}]_{s,h}^X(t)$.

The proofs of (A)–(F) are all very similar. We detail the one for the case (A) and leave the others to the reader.

Proof of (A). Suppose $(s, h) \models \text{var}_X(t) \wedge \text{var.sby}(t)$. If $\text{Path}[\mathcal{S}]_{s,h}^X(t) = \emptyset$, then (A) and $\text{Path}[\mathcal{S}]_{s,h}^X(t) \neq \emptyset$. So, $\llbracket t \rrbracket_{s,h}^X \in \text{dom}(h)$ and $\text{Path}[\mathcal{S}]_{s,h}^X(t)$ is a minimal set describing a non-empty path, going from $\llbracket t \rrbracket_{s,h}^X$ to $\text{sby}_{s,h}^X(t)$. From $(s, h) \models \text{var}_X(t)$ and by (III), we derive that $\llbracket t \rrbracket_{s,h}^X$ and $h(\llbracket t \rrbracket_{s,h}^X)$ are in $\text{Lab}[\mathcal{W}]_{s,h}^X$. Then, by Proposition 5.13, both $\llbracket t \rrbracket_{s,h}^X$ and $h(\llbracket t \rrbracket_{s,h}^X)$ do not belong to $\text{Rem}[\mathcal{W}]_{s,h}^X$. If $h(\llbracket t \rrbracket_{s,h}^X) = \text{sby}_{s,h}^X(t)$, then we conclude that $\text{Path}[\mathcal{S}]_{s,h}^X(t) = \{\llbracket t \rrbracket_{s,h}^X\}$. We conclude that (A) holds, as

$$\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X = \emptyset = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, l_{\text{pre}}\}.$$

Otherwise, let us assume that $h(\llbracket t \rrbracket_{s,h}^X) \neq \text{sby}_{s,h}^X(t)$, and therefore $\llbracket t \rrbracket_{s,h}^X$ and $h(\llbracket t \rrbracket_{s,h}^X)$ are two disjoint locations in $\text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \text{Rem}[\mathcal{W}]_{s,h}^X$. Let us consider the set

$$L_1 \stackrel{\text{def}}{=} \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X)\}.$$

Notice that $L_1 \cap \text{Rem}[\mathcal{W}]_{s,h}^X = \text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X$. If $L_1 = \emptyset$, then $\text{Path}[\mathcal{S}]_{s,h}^X(t) = \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X)\}$ and $\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X = \emptyset$, which verifies (A). Otherwise, suppose $L_1 \neq \emptyset$. This implies that $\text{Path}[\mathcal{S}]_{s,h}^X(t)$ contains at least three locations, and therefore it

$$\bigvee_{\substack{\tau \subseteq T[\mathcal{S}]^X \\ (\beta_t \in [1, \alpha])_{t \in \tau} \\ (\beta_j \in [0, \alpha])_{j \in [1, \alpha]} \\ \beta_{\uparrow}, \beta_R \in [0, \alpha] \\ \beta \leq \sum_{i \in \tau \cup [1, \alpha] \cup \{\uparrow, R\}} \beta_i}} \left(\begin{array}{l} \text{rem}_{X, \alpha}^{\mathcal{S}} \geq \beta_R \wedge \bigwedge_{j \in [1, \alpha]} \text{loop}_X^{\mathcal{S}}(j) \geq \beta_j \wedge \uparrow \text{loop}_{X, \alpha}^{\mathcal{S}} \geq \beta_{\uparrow} \wedge \text{alldiff}(\tau) \\ \wedge \bigwedge_{t \in \tau} \left(\begin{array}{l} \neg \text{sees}_X(t, t) = 1 \\ \wedge (\text{var}_X(t) \wedge \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 3) \\ \wedge (\text{next}_X(t) \wedge \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 2) \\ \wedge (\text{var}_X(t) \wedge \neg \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 2) \\ \wedge (\text{next}_X(t) \wedge \neg \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 1) \\ \wedge (\text{unlab}(t) \wedge \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 1) \\ \wedge (\text{unlab}(t) \wedge \neg \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t) \end{array} \right) \end{array} \right)$$

Note: The formulae $\text{rem}_{X, \alpha}^{\mathcal{S}} \geq 0$, $\text{loop}_X^{\mathcal{S}}(j) \geq 0$ (where $j \in [1, \alpha]$) and $\uparrow \text{loop}_{X, \alpha}^{\mathcal{S}} \geq 0$ stand for the tautology $\text{rem}_{X, \alpha}^{\mathcal{S}} \geq 1 \vee \neg \text{rem}_{X, \alpha}^{\mathcal{S}} \geq 1$.

Figure 5.13: The formula $\tau(\text{rem}_X^{\mathcal{W}} \geq \beta)$.

describes a path $\rho = (\ell_0, \dots, \ell_p)$ going from $\llbracket t \rrbracket_{s,h}^X$ to $\text{sby}_{s,h}^X(t)$, of length $p \geq 3$. Therefore, $\ell_{p-1} = l_{pre}$ is distinct from $\ell_0 = \llbracket t \rrbracket_{s,h}^X$ and $\ell_1 = h(\llbracket t \rrbracket_{s,h}^X)$, which implies $l_{pre} \in L_1$. To conclude the proof, it is sufficient to show that l_{pre} is the only location of L_1 that does not belong to $\text{Rem}[\mathcal{W}]_{s,h}^X$. By $(s, h) \models \text{var.sby}(t)$, we know that the location $\text{sby}_{s,h}^X(t)$ corresponds to a program variable, say x . Hence, $l_{pre} \in \text{Pred}[\mathcal{W}]_{s,h}^X$. We consider the set $L_2 \stackrel{\text{def}}{=} L_1 \setminus \{\ell\}$, and show that it only contains locations in $\text{Rem}[\mathcal{W}]_{s,h}^X$. By (III), L_2 does not contain labelled locations. Moreover, since $l_{pre} \notin L_2$, this set does not contain predecessors of locations assigned to program variables. Lastly, since $\text{Path}[\mathcal{S}]_{s,h}^X(t)$ describes the path ρ , which has length at least three, it cannot be that there is a location $\ell' \in L_2$ such that $h(\ell') = \ell'$. By Proposition 5.13, $L_2 \subseteq \text{Rem}[\mathcal{W}]_{s,h}^X$. This concludes the proof. Interestingly enough, under the hypothesis that $\text{Path}[\mathcal{S}]_{s,h}^X(\ell) \cap \text{Rem}[\mathcal{W}]_{s,h}^X \neq \emptyset$, we found that $\text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \text{Rem}[\mathcal{W}]_{s,h}^X$ is made of the three distinct locations $\llbracket t \rrbracket_{s,h}^X$, $h(\llbracket t \rrbracket_{s,h}^X)$ and ℓ , as intuitively depicted in the first memory state of in Figure 5.12.

Let us now characterise $\text{rem}_X^{\mathcal{W}} \geq \beta$ as a Boolean combination of core formulae in $\text{Core}[\mathcal{S}](s, h)$, concluding the proof.

case: $\varphi = \text{rem}_X^{\mathcal{W}} \geq \beta$. φ is equivalent to the formula given in Figure 5.13. Notice that this formula is a Boolean combination of formulae in $\text{Core}[\mathcal{S}](X, \alpha)$, as required by the lemma. Indeed, all its subformulae are made of formulae in $\text{Core}[\mathcal{S}](X, \alpha)$, with the only non-trivial case being the subformulae of the form $\text{seeslen}_X(t) \geq \beta_t + k$ (where $\beta_t \in [1, \alpha]$ and $k \in [0, 3]$). These subformulae are Boolean combinations of formulae in $\text{Core}[\mathcal{S}](X, \alpha)$, as the bound $\frac{1}{6}(\alpha + 1)(\alpha + 2)(\alpha + 3)$ given to β in formulae of the form $\text{sees}_X(t, t') \geq \beta$ is always at least $\alpha + 3$ (recall that $\alpha \geq 1$). We show that $\varphi \equiv \tau(\varphi)$.

(\Rightarrow): Suppose $(s, h) \models \varphi$, and so $\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X) \geq \beta$. We define the integers β_R , β_{\uparrow} and β_j (where $j \in [2, \alpha]$) as follows:

$$\begin{aligned}\beta_R &\stackrel{\text{def}}{=} \min(\alpha, \text{card}(\text{Rem}[S]_{s,h}^{X,\alpha})), \\ \beta_{\uparrow} &\stackrel{\text{def}}{=} \min(\alpha, \text{card}(\uparrow\text{Cycl}[S]_{s,h}^{X,\alpha})), \\ \beta_j &\stackrel{\text{def}}{=} \min(\alpha, \text{card}(\text{Cycl}[S]_{s,h}^X(j))).\end{aligned}$$

Similarly, given a term $t \in T[S]^X$ such that $\llbracket t \rrbracket_{s,h}^X$ is defined, we define β_t as follows:

$$\beta_t \stackrel{\text{def}}{=} \min(\alpha, \text{card}(\text{Rem}[W]_{s,h}^X \cap \text{Path}[S]_{s,h}^X(t))).$$

Among the terms in $T[S]^X$, let us consider a maximal subset T of them such that for every two distinct $t, t' \in T$, we have $\beta_t \geq 1$ and $\llbracket t \rrbracket_{s,h}^X \neq \llbracket t' \rrbracket_{s,h}^X$. Equivalently, T is a minimal set of terms that corresponds to all the locations ℓ such that $\text{Rem}[W]_{s,h}^X \cap \text{Path}[S]_{s,h}^X(\ell)$ is non-empty. From (\star) , we conclude that

$$\beta_R + \beta_{\uparrow} + \sum_{j \in [2, \alpha]} \beta_j + \sum_{t \in T} \beta_t \geq \beta.$$

Let us now show that $(s, h) \models \tau(\varphi)$ by showing that (s, h) satisfies the disjunct corresponding to the quantities we just defined. Directly from their definition, we conclude that (s, h) satisfies $\text{rem}_{X,\alpha}^S \geq \beta_R$, $\uparrow\text{loop}_{X,\alpha}^S \geq \beta_{\uparrow}$ and $\text{loop}_X^S(j) \geq \beta_j$, for every $j \in [2, \alpha]$. Moreover, by definition of T we derive that $(s, h) \models \text{alldiff}(T)$. Similarly, as for every $t \in T$ we have that $\text{Rem}[W]_{s,h}^X \cap \text{Path}[S]_{s,h}^X(t)$ is non-empty, it must be that $h(\llbracket t \rrbracket_{s,h}^X) \neq \llbracket t \rrbracket_{s,h}^X$. Indeed, *ad absurdum*, suppose that $h(\llbracket t \rrbracket_{s,h}^X) = \llbracket t \rrbracket_{s,h}^X$. Then, $\text{Path}[S]_{s,h}^X(t) = \{\llbracket t \rrbracket_{s,h}^X\}$, which implies that $\llbracket t \rrbracket_{s,h}^X \in \text{Rem}[W]_{s,h}^X$. However, this is contradictory, as no location in $\text{Rem}[W]_{s,h}^X$, can form a self-loop. Thus, $h(\llbracket t \rrbracket_{s,h}^X) \neq \llbracket t \rrbracket_{s,h}^X$, which in turn allows us to conclude that $(s, h) \models \neg\text{sees}_X(t, t) = 1$. In order to conclude the proof it is sufficient to show that, for every $t \in T$, (s, h) satisfies the following six conjuncts of $\tau(\varphi)$:

- a. $\text{var}_X(t) \wedge \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 3$,
- b. $\text{next}_X(t) \wedge \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 2$,
- c. $\text{var}_X(t) \wedge \neg\text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 2$,
- d. $\text{next}_X(t) \wedge \neg\text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 1$,
- e. $\text{unlab}(t) \wedge \text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t + 1$,
- f. $\text{unlab}(t) \wedge \neg\text{var.sby}(t) \Rightarrow \text{seeslen}_X(t) \geq \beta_t$.

Notice that, for each term in T , exactly one of the antecedents of these six implications is satisfied. Clearly, the split into the subformulae (a)–(f) follow the cases (A)–(F) discussed above. Below, we write l_{pre} for the the location in $\text{Path}[S]_{s,h}^X(t) \neq \emptyset$ such that $h(l_{\text{pre}}) = \text{sby}_{s,h}^X(t)$. We split the proof depending on these cases.

case: $(s, h) \models \text{var}_X(t) \wedge \text{var.sby}(t)$. From (A),

$$\text{Path}[S]_{s,h}^X(t) \cap \text{Rem}[W]_{s,h}^X = \text{Path}[S]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X), l_{\text{pre}}\}.$$

Since $\text{Path}[S]_{s,h}^X(t) \cap \text{Rem}[W]_{s,h}^X \neq \emptyset$, there is $\ell \in \text{Path}[S]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X), l_{\text{pre}}\}$. Let $\rho = (\ell_0, \dots, \ell_p)$ be the path described by $\text{Path}[S]_{s,h}^X(t)$, going from $\ell_0 = \llbracket t \rrbracket_{s,h}^X$ to $\ell_p = \text{sby}_{s,h}^X(t)$. In this path, $l_{\text{pre}} = \ell_{p-1}$. Since $\ell \notin \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X), l_{\text{pre}}\}$, $\ell = \ell_i$ for some $i \in [2, p-2]$. This implies that $\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X)$ and l_{pre} are all distinct. We have:

$$\text{card}(\text{Path}[S]_{s,h}^X(t) \cap \text{Rem}[W]_{s,h}^X) = \text{card}(\text{Path}[S]_{s,h}^X(t)) - 3.$$

Therefore, $\text{card}(\text{Path}[S]_{s,h}^X(t)) \geq \beta_t + 3$, which implies $(s, h) \models \text{seeslen}_X(t) \geq \beta_t + 3$.

case: $(s, h) \models \text{next}_X(t) \wedge \text{var.sby}(t)$. From (B),

$$\text{Path}[S]_{s,h}^X(t) \cap \text{Rem}[W]_{s,h}^X = \text{Path}[S]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, l_{\text{pre}}\}.$$

Similarly to the previous case, from $\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X \neq \emptyset$, we conclude that $\llbracket t \rrbracket_{s,h}^X$ and l_{pre} are distinct, and so $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) = \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) - 2$. Therefore, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \beta_t + 2$, which implies $(s, h) \models \text{seeslen}_X(t) \geq \beta_t + 2$.

case: $(s, h) \models \text{var}_X(t) \wedge \neg \text{var.sby}(t)$. From (C),

$$\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X)\}.$$

Similarly to the previous cases, by $\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X \neq \emptyset$, we conclude that $\llbracket t \rrbracket_{s,h}^X$ and $h(\llbracket t \rrbracket_{s,h}^X)$ are distinct, and so $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) = \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) - 2$. Therefore, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \beta_t + 2$, which implies $(s, h) \models \text{seeslen}_X(t) \geq \beta_t + 2$.

case: $(s, h) \models \text{next}_X(t) \wedge \neg \text{var.sby}(t)$. From (D),

$$\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}.$$

Since $\llbracket t \rrbracket_{s,h}^X \in \text{Path}[\mathcal{S}]_{s,h}^X(t)$, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) = \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) - 1$. Therefore, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \beta_t + 1$, which implies $(s, h) \models \text{seeslen}_X(t) \geq \beta_t + 1$.

case: $(s, h) \models \text{unlab}(t) \wedge \text{var.sby}(t)$. From (E),

$$\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{l_{\text{pre}}\}.$$

Since $l_{\text{pre}} \in \text{Path}[\mathcal{S}]_{s,h}^X(t)$, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) = \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) - 1$. Therefore, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \beta_t + 1$, which implies $(s, h) \models \text{seeslen}_X(t) \geq \beta_t + 1$.

case: $(s, h) \models \text{unlab}(t) \wedge \neg \text{var.sby}(t)$. By (F), $\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X = \text{Path}[\mathcal{S}]_{s,h}^X(t)$.

Therefore, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \beta_t$, which implies $(s, h) \models \text{seeslen}_X(t) \geq \beta_t$.

(\Leftarrow): Suppose $(s, h) \models \tau(\varphi)$, and therefore one disjunct ψ of $\tau(\varphi)$ is satisfied. Let $T \subseteq T[\mathcal{S}]^X$, $\beta_t \in [1, \alpha]$ (where $t \in T$), and $\beta_R, \beta_\uparrow, \beta_j \in [0, \alpha]$ (where $j \in [1, \alpha]$) be the set of terms and integers that correspond to the disjunct ψ . This also means that

$$\beta_R + \beta_\uparrow + \sum_{j \in [2, \alpha]} \beta_j + \sum_{t \in T} \beta_t \geq \beta.$$

We show that

- a. $\text{card}(\text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}) \geq \beta_R$, $\text{card}(\uparrow \text{Cycl}[\mathcal{S}]_{s,h}^{X,\alpha}) \geq \beta_\uparrow$, and $\text{card}(\text{Cycl}[\mathcal{S}]_{s,h}^X(j)) \geq \beta_j$ ($j \in [1, \alpha]$),
- b. for all $t \in T$, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) \geq \beta_t$.

From $(s, h) \models \text{alldiff}(t)$ we have $\text{card}(T) = \text{card}(\llbracket T \rrbracket_{s,h}^X)$, and so these two statements imply $(s, h) \models \varphi$ directly from (*). (a) holds from the fact that (s, h) satisfies $\text{rem}_{X,\alpha}^S \geq \beta_R$, $\uparrow \text{loop}_{X,\alpha}^S \geq \beta_\uparrow$ and, for every $j \in [1, \alpha]$, $\text{loop}_X^S(j) \geq \beta_j$. In order to prove (b), let us consider $t \in T$. From $(s, h) \not\models \text{sees}_X(t, t) = 1$ we conclude that $\text{Path}[\mathcal{S}]_{s,h}^X(t)$ cannot describe a self-loop, and so for every location ℓ in this set we have $h(\ell) \neq \ell$. The proof is now divided into six cases, depending on which formula among $\text{var}_X(t)$, $\text{next}_X(t)$ and $\text{unlab}(t)$ is satisfied, and on whether $\text{var.sby}(t)$ holds. Below, whenever $\text{Path}[\mathcal{S}]_{s,h}^X(t)$ is non-empty, we write l_{pre} for the location in $\text{Path}[\mathcal{S}]_{s,h}^X(t)$ such that $h(l_{\text{pre}}) = \text{sby}_{s,h}^X(t)$.

case: $(s, h) \models \text{var}_X(t) \wedge \text{var.sby}(t)$. From $(s, h) \models \psi$, we conclude that (s, h) satisfies $\text{seeslen}_X(t) \geq \beta_t + 3$, and thus $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \beta_t + 3 \geq 4$. This implies that $\llbracket t \rrbracket_{s,h}^X$, $h(\llbracket t \rrbracket_{s,h}^X)$ and l_{pre} are three distinct locations. From (A),

$$\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X = \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X), l_{\text{pre}}\}.$$

As $\{\llbracket t \rrbracket_{s,h}^X, h(\llbracket t \rrbracket_{s,h}^X), l_{\text{pre}}\} \subseteq \text{Path}[\mathcal{S}]_{s,h}^X(t)$, we derive $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) \geq \beta_t$.

case: $(s, h) \models \text{next}_X(t) \wedge \text{var.sby}(t)$. By $(s, h) \models \psi$, we have $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \beta_t + 2$.

This implies that $\llbracket t \rrbracket_{s,h}^X$ and l_{pre} are two distinct locations, and therefore, from (B), we derive that $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \text{Rem}[\mathcal{W}]_{s,h}^X) = 2$. So, $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) \geq \beta_t$.

- case:** $(s, h) \models \text{var}_X(t) \wedge \neg \text{var.sby}(t)$. By $(s, h) \models \psi$, we have $\text{card}(\text{Path}[s]_{s,h}^X(t)) \geq \beta_t + 2$. This implies that $\llbracket t \rrbracket_{s,h}^X$ and $h(\llbracket t \rrbracket_{s,h}^X)$ are two distinct locations, and thus, from (C), we derive that $\text{card}(\text{Path}[s]_{s,h}^X(t) \setminus \text{Rem}[\mathcal{W}]_{s,h}^X) = 2$. So, $\text{card}(\text{Path}[s]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) \geq \beta_t$.
- case:** $(s, h) \models \text{next}_X(t) \wedge \neg \text{var.sby}(t)$. By $(s, h) \models \psi$, we have $\text{card}(\text{Path}[s]_{s,h}^X(t)) \geq \beta_t + 1$. By (D), $\text{card}(\text{Path}[s]_{s,h}^X(t) \setminus \text{Rem}[\mathcal{W}]_{s,h}^X) = 1$. So, $\text{card}(\text{Path}[s]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) \geq \beta_t$.
- case:** $(s, h) \models \text{unlab}(t) \wedge \text{var.sby}(t)$. By $(s, h) \models \psi$, we have $\text{card}(\text{Path}[s]_{s,h}^X(t)) \geq \beta_t + 1$. By (E), $\text{card}(\text{Path}[s]_{s,h}^X(t) \setminus \text{Rem}[\mathcal{W}]_{s,h}^X) = 1$. So, $\text{card}(\text{Path}[s]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) \geq \beta_t$.
- case:** $(s, h) \models \text{unlab}(t) \wedge \neg \text{var.sby}(t)$. By $(s, h) \models \psi$, we have $\text{card}(\text{Path}[s]_{s,h}^X(t)) \geq \beta_t$. By (F), $\text{card}(\text{Path}[s]_{s,h}^X(t) \setminus \text{Rem}[\mathcal{W}]_{s,h}^X) = 0$. So, $\text{card}(\text{Path}[s]_{s,h}^X(t) \cap \text{Rem}[\mathcal{W}]_{s,h}^X) \geq \beta_t$.

Since (s, h) satisfies one of the six cases above (for every $t \in T$), (b) is satisfied. \square

Corollary 5.36. $\approx_{X,\alpha}^S \subseteq \approx_{X,\alpha}^W$.

Proof. Directly from Lemma 5.35. \square

Apart from Corollary 5.36, Lemma 5.35 has a second consequence: thanks to Lemma 5.15, it shows that every atomic formula of the weak fragment is equivalent to a Boolean combination of formulae in $\text{Core}[S](X, 1)$. We show that this is also the case for the atomic formulae $x \hookrightarrow^+ t$ and $u \hookrightarrow^+ u$, resulting in a proof that Boolean combinations of the core formulae defined in this section capture the atomic formulae of the strong fragment.

Lemma 5.37. Every atomic formula of the strong fragment written with variables from $X \cup \{u\}$ is equivalent to a Boolean combination of formulae from $\text{Core}[S](X, 1)$.

Proof. For the atomic formulae of \mathcal{W} , i.e. \top , emp , $t = t'$ and $t \hookrightarrow t'$ (where $t, t' \in X \cup \{u\}$), the result holds directly from Lemmata 5.15 and 5.35. It remains to prove that Boolean combinations of $\text{Core}[S](X, 1)$ capture the formulae $x \hookrightarrow^+ y$, $x \hookrightarrow^+ u$ and $u \hookrightarrow^+ u$, where $x, y \in X$. Let us first define a Boolean combination of core formulae in $\text{Core}[S](X, 1)$ stating that there is a non-empty path going from the location corresponding to a term t to the location corresponding to a term t' , where $t, t' \in T[S]^X$. We write $t \hookrightarrow_S^+ t'$ to denote this formula, which is defined as

$$\bigvee_{\substack{n \in [1, \text{card}(T[S]^X)] \\ (t_0, t_1, \dots, t_n) \in (T[S]^X)^{n+1} \\ t_0 = t, t_n = t'}} \left(\bigwedge_{j \in [0, n-1]} \text{sees}_X(t_j, t_{j+1}) \geq 1 \right).$$

Let us show that $(s, h) \models t \hookrightarrow_S^+ t'$ if and only if there is $\delta \geq 1$ such that $h^\delta(\llbracket t \rrbracket_{s,h}^X) = \llbracket t' \rrbracket_{s,h}^X$.

(\Rightarrow) : Suppose $(s, h) \models t \hookrightarrow_S^+ t'$. By definition, there is a sequence of terms t_0, \dots, t_n (where $n \in [1, \text{card}(T[S]^X)]$) such that $t_0 = t$, $t_n = t'$ and $(s, h) \models \text{sees}_X(t_j, t_{j+1}) \geq 1$ holds for all $j \in [0, n-1]$. So, given $j \in [0, n-1]$, there is $\delta_j \geq 1$ s.t. $h^{\delta_j}(\llbracket t_j \rrbracket_{s,h}^X) = \llbracket t_{j+1} \rrbracket_{s,h}^X$. Let $\delta = \sum_{j \in [0, n-1]} \delta_j$. We have $h^\delta(\llbracket t \rrbracket_{s,h}^X) = \llbracket t' \rrbracket_{s,h}^X$.

(\Leftarrow) : Suppose that there is $\delta \geq 1$ such that $h^\delta(\llbracket t \rrbracket_{s,h}^X) = \llbracket t' \rrbracket_{s,h}^X$, which means that h witnesses a non-empty path ρ of length δ , going from $\llbracket t \rrbracket_{s,h}^X$ to $\llbracket t' \rrbracket_{s,h}^X$. W.l.o.g. we can assume the path ρ to be minimal, i.e. $h^{\delta'}(\llbracket t \rrbracket_{s,h}^X) \neq \llbracket t' \rrbracket_{s,h}^X$ for all $\delta' \in [1, \delta - 1]$. With this assumption, if $\llbracket t \rrbracket_{s,h}^X \neq \llbracket t' \rrbracket_{s,h}^X$ then no location can appear twice in ρ . Otherwise, if $\llbracket t \rrbracket_{s,h}^X = \llbracket t' \rrbracket_{s,h}^X$ then $\llbracket t \rrbracket_{s,h}^X$ is the only location that appears twice in ρ (in this case, ρ describes a cycle). Let L be the set of labelled locations belonging to the path ρ , excluding $\llbracket t \rrbracket_{s,h}^X$ and $\llbracket t' \rrbracket_{s,h}^X$. Formally, $L = \{\ell \in \text{Lab}[S]_{s,h}^X \mid \text{there is } \delta' \in [1, \delta - 1] \text{ s.t. } h^{\delta'}(\llbracket t \rrbracket_{s,h}^X) = \ell\}$. Each location in L appears only once in ρ , and thus we can order them uniquely, following

their distance from $\llbracket t \rrbracket_{s,h}^X$. Given $\ell_1, \ell_2 \in L$, we write $\ell_1 <_\rho \ell_2$ whenever $\delta_1 < \delta_2$ holds for the only two lengths $\delta_1, \delta_2 \in [1, \delta - 1]$ such that $h^{\delta_1}(\llbracket t \rrbracket_{s,h}^X) = \ell_1$ and $h^{\delta_2}(\llbracket t \rrbracket_{s,h}^X) = \ell_2$. Informally, $\ell_1 <_\rho \ell_2$ holds whenever ℓ_1 precedes ℓ_2 in ρ . Let (ℓ_1, \dots, ℓ_k) be the sorted tuple of the elements of L , i.e. the tuple satisfying the conditions $\{\ell_1, \dots, \ell_k\} = L$ and for every $i, j \in [1, k]$, if $i < j$ then $\ell_i <_\rho \ell_j$. Since every location in L is labelled, this tuple corresponds to a tuple of k distinct terms (t_1, \dots, t_k) such that $\llbracket t_j \rrbracket_{s,h}^X = \ell_j$ for all $j \in [1, k]$. Let us consider the tuple $(t_0, t_1, \dots, t_k, t_{k+1})$, where $t_0 = t$ and $t_{k+1} = t'$. By definition of L , given $j \in [0, k]$, $\text{Path}[s]_{s,h}^X(t_j) \neq \emptyset$ and $\text{sby}_{s,h}^X(t_j) = \llbracket t_{j+1} \rrbracket_{s,h}^X$. Thus, $(s, h) \models \text{sees}_X(t_j, t_{j+1}) \geq 1$. To conclude, we show that $(t_0, t_1, \dots, t_k, t_{k+1})$ is in $(T[s]^X)^{n+1}$ for some $n \in [1, \text{card}(T[s]^X) + 1]$. This is quite easy to see: by definition of L , the terms t and t' cannot appear in the tuple (t_1, \dots, t_k) . As these two terms can be syntactically equal, this means that $k \leq \text{card}(T[s]^X) - 1$, which in turn bound the length of $(t_0, t_1, \dots, t_k, t_{k+1})$ to be at most $\text{card}(T[s]^X) + 1$.

We write $t \hookrightarrow_S^* t'$ for the formula $t = t' \vee t \hookrightarrow_S^+ t'$, which is satisfied whenever a memory state (s, h) witnesses a (possibly empty) path going from $\llbracket t \rrbracket_{s,h}^X$ to $\llbracket t' \rrbracket_{s,h}^X$. We use this formula to extend the definition of $t \hookrightarrow_S^+ t'$ for the case where $t' = u$. The formula $t \hookrightarrow_S^+ u$ is defined as

$$\bigvee_{t_1, t_2 \in T[s]^X} (t \hookrightarrow_S^* t_1 \wedge ((u = t_2 \wedge \text{sees}_X(t_1, t_2) \geq 1) \vee u \in \text{sees}_X(t_1, t_2) \geq (1, 1)))$$

Let us show that $(s, h) \models t \hookrightarrow_S^+ u$ if and only if there is $\delta \geq 1$ such that $h^\delta(\llbracket t \rrbracket_{s,h}^X) = s(u)$.

(\Rightarrow) : Suppose $(s, h) \models t \hookrightarrow_S^+ u$. By definition there are two terms $t_1, t_2 \in T[s]^X$ such that (s, h) satisfies $t \hookrightarrow^* t_1$ and either $u = t_2 \wedge \text{sees}_X(t_1, t_2) \geq 1$ or $u \in \text{sees}_X(t_1, t_2) \geq (1, 1)$. First, let us assume that $(s, h) \models t \hookrightarrow^* t_1 \wedge u = t_2 \wedge \text{sees}_X(t_1, t_2) \geq 1$. There are $\delta \geq 0$ and $\delta' \geq 1$ such that $h^\delta(\llbracket t \rrbracket_{s,h}^X) = \llbracket t_1 \rrbracket_{s,h}^X$ and $h^{\delta'}(\llbracket t_1 \rrbracket_{s,h}^X) = \llbracket t_2 \rrbracket_{s,h}^X$. Moreover, $\llbracket t_2 \rrbracket_{s,h}^X = s(u)$, and therefore $h^{\delta+\delta'}(\llbracket t \rrbracket_{s,h}^X) = s(u)$, as required by the right-hand side of the equivalence. Similarly, in the case where $(s, h) \models t \hookrightarrow^* t_1 \wedge u \in \text{sees}_X(t_1, t_2) \geq (1, 1)$, there are $\delta \geq 0$ and $\delta' \geq 1$ s.t. $h^\delta(\llbracket t \rrbracket_{s,h}^X) = \llbracket t_1 \rrbracket_{s,h}^X$ and $h^{\delta'}(\llbracket t_1 \rrbracket_{s,h}^X) = s(u)$. Again, $h^{\delta+\delta'}(\llbracket t \rrbracket_{s,h}^X) = s(u)$.

(\Leftarrow) : Suppose that there is $\delta \geq 1$ such that $h^\delta(\llbracket t \rrbracket_{s,h}^X) = s(u)$, which means that h witnesses a non-empty path ρ of length δ , going from $\llbracket t \rrbracket_{s,h}^X$ to $s(u)$. Let ℓ be the labelled location in this path that is closest to $s(u)$, excluding $s(u)$ itself. Formally, $\ell \in \text{Lab}[s]_{s,h}^X$ and there is $\delta_1 \in [0, \delta - 1]$ such that $h^{\delta_1}(\llbracket t \rrbracket_{s,h}^X) = \ell$ and for every $\delta_2 \in [1, \delta - 1 - \delta_1]$, $h^{\delta_2}(\ell) \notin \text{Lab}[s]_{s,h}^X$. Note that ℓ could be $\llbracket t \rrbracket_{s,h}^X$. Let $t_1 \in T[s]^X$ such that $\llbracket t_1 \rrbracket_{s,h}^X = \ell$. We have $(s, h) \models t \hookrightarrow_S^* t_1$. By definition, $\ell \in \text{dom}(h)$. As ℓ is a labelled location, this means that $\text{Path}[s]_{s,h}^X(\ell)$ is not empty, and therefore there is a term $t_2 \in T[s]^X$ such that $\llbracket t \rrbracket_{s,h}^X = \text{sby}_{s,h}^X(\ell)$. Again by definition of $\ell = \llbracket t_1 \rrbracket_{s,h}^X$, one of the following must hold:

1. $\llbracket t_2 \rrbracket_{s,h}^X = s(u)$,
2. $\llbracket t_2 \rrbracket_{s,h}^X \neq s(u)$ and $s(u) \in \text{Path}[s]_{s,h}^X(\ell)$.

Indeed, ℓ is the labelled location in ρ that is closest to $s(u)$, $s(u)$ excluded, and so $\text{sby}_{s,h}^X(\ell)$ is either $s(u)$ or reaches $s(u)$. If (1) holds, then $(s, h) \models u = t_2 \wedge \text{sees}_X(t_1, t_2) \geq 1$. If (2) holds, then $(s, h) \models u \in \text{sees}_X(t_1, t_2) \geq (1, 1)$. Either way, $(s, h) \models t \hookrightarrow_S^+ u$.

Thanks to the formula $t \hookrightarrow_S^+ t'$, the characterisation of the atomic formulae $x \hookrightarrow^+ y$, $x \hookrightarrow^+ u$ and $u \hookrightarrow^+ u$ in terms of Boolean combination of formulae in $\text{Core}[s](X, 1)$ is now straightforward:

case: $\varphi = x \hookrightarrow^+ y$. φ is equivalent to $x \hookrightarrow_S^+ y$.

case: $\varphi = x \hookrightarrow^+ u$. φ is equivalent to $x \hookrightarrow_S^+ u$.

case: $\varphi = u \hookrightarrow^+ u$. φ is equivalent to

$$\psi \stackrel{\text{def}}{=} \bigvee_{\beta \in [1, \alpha]} u \in \text{loop}_x^S(\beta) \vee u \in \uparrow\text{loop}_{x, \alpha}^S \vee \bigvee_{t \in T[S]^x} (t \hookrightarrow_S^+ t \wedge t \hookrightarrow_S^+ u)$$

Let us prove that $\varphi \equiv \psi$.

(\Rightarrow): $(s, h) \models u \hookrightarrow^+ u$, and thus there is $\delta \geq 1$ such that $h^\delta(s(u)) = s(u)$. Informally, $s(u)$ belongs to a cycle. Let L be the finite set of locations reachable from $s(x)$ in at least one step, i.e. $L = \{h^\delta(s(x)) \mid \delta \geq 1\}$. As $s(u)$ belongs to a cycle, $s(u) \in L$. We distinguish the following three cases:

1. L does not contain any labelled location of $\text{Lab}[S]_{s,h}^x$ and $\text{card}(L) = \beta \in [1, \alpha]$. As L describes a cycle of unlabelled locations and $s(u) \in L$, we have $(s, h) \models u \in \text{loop}_x^S(\beta)$.
2. $L \cap \text{Lab}[S]_{s,h}^x = \emptyset$ and $\text{card}(L) > \alpha$. Similarly to the case (1), $(s, h) \models u \in \uparrow\text{loop}_{x, \alpha}^S$.
3. L contains a labelled location $\ell \in \text{Lab}[S]_{s,h}^x$. Let $t \in T[S]^x$ such that $[t]_{s,h}^x = \ell$. Again recalling that L describes a cycle and $s(u) \in L$, we derive $(s, h) \models t \hookrightarrow_S^+ t \wedge t \hookrightarrow_S^+ u$.

In all three cases, $(s, h) \models \psi$.

(\Leftarrow): The proof of this direction is straightforward. \square

5.5.3 Step III: *-simulation.

We adapt the arguments used in Section 5.3.3 for the weak fragment in order to establish that the family of core formulae $\text{Core}[S](X, \alpha)$ enjoy the *-simulation property. Due to the complexity of the core formulae of the strong fragment, the technical steps required to prove this result reveal to be more involved and, despite the modularity of the game hopping argument, lead to proofs spawning several pages. To lighten the presentation, we show only the more interesting and challenging “hops”, leaving the others in Appendix C.

We start by defining the hop relation of the strong fragment, denoted by $\leftrightarrow_{X, \alpha}^S$.

Definition 5.38 (s -hop relation). We write $\leftrightarrow_{X, \alpha}^S$ for the relation on memory states such that

$$(s, h) \leftrightarrow_{X, \alpha}^S (s', h') \quad \text{iff} \quad \begin{aligned} &\text{for every two heaps } h_1 \text{ and } h_2 \text{ and every } \alpha_1 \geq 1 \text{ and } \alpha_2 \geq 1, \\ &\text{if } h = h_1 + h_2 \text{ and } \alpha = \alpha_1 + \alpha_2 \text{ then there are two heaps } h'_1 \text{ and } h'_2 \\ &\text{such that } h' = h'_1 + h'_2, (s, h_1) \approx_{X, \alpha_1}^S (s', h'_1) \text{ and } (s, h_2) \approx_{X, \alpha_2}^S (s', h'_2). \end{aligned}$$

The s -hop relation $\leftrightarrow_{X, \alpha}^S$ is both reflexive and transitive (see Lemma 5.9). We remind the reader that the *-simulation property corresponds to the inclusion $\approx_{X, \alpha}^S \subseteq \leftrightarrow_{X, \alpha}^S$ which, given two memory states $(s, h) \approx_{X, \alpha}^S (s', h')$, we prove by building a chain of hops

$$(s, h) = (s_1, h_1) \leftrightarrow_{X, \alpha}^S (s_2, h_2) \leftrightarrow_{X, \alpha}^S \dots \leftrightarrow_{X, \alpha}^S (s_{k-1}, h_{k-1}) \leftrightarrow_{X, \alpha}^S (s_k, h_k) = (s', h').$$

Each hop of the chain corresponds to one of several intermediate results, the first of which involves the case of memory states that are in the indistinguishability relation $\approx_{X, \alpha}^S$ for every $\alpha \geq 1$. For two memory states to satisfy this property, they must agree on the cardinality of every set of the partition defined in Section 5.5.1.

Lemma 5.39. For every $\alpha \geq 1$, $\left(\bigcap_{\alpha' \geq 1} \approx_{X, \alpha'}^S \right) \subseteq \leftrightarrow_{X, \alpha}^S$.

Proof. Let (s, h) and (s', h') be two memory states such that $((s, h), (s', h')) \in (\bigcap_{\alpha' \geq 1} \approx_{X, \alpha'}^S)$. This property of the two memory states allows us to construct a bijection $f : \text{LOC} \rightarrow \text{LOC}$ s.t.

- 1_f. $f(s(u)) = s'(u)$ and for every $t \in T[\mathcal{S}]^X$, if $\llbracket t \rrbracket_{s,h}^X$ is defined then $f(\llbracket t \rrbracket_{s,h}^X) = \llbracket t \rrbracket_{s',h'}^X$,
- 2_f. for every $t \in T[\mathcal{S}]^X$ and $\delta \in [1, \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t))]$, $f(h^\delta(\llbracket t \rrbracket_{s,h}^X)) = h'^\delta(\llbracket t \rrbracket_{s',h'}^X)$,
- 3_f. for every $x \in X$, $\text{Pred}[\mathcal{S}]_{s',h'}^X(x) = f(\text{Pred}[\mathcal{S}]_{s,h}^X(x))$,
- 4_f. given $\beta \in [1, \alpha]$, f induces a bijection from $\text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)$ to $\text{Cycl}[\mathcal{S}]_{s',h'}^X(\beta)$. Formally, for every $L \in \text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)$ there is $L' \in \text{Cycl}[\mathcal{S}]_{s',h'}^X(\beta)$ such that $f(L) = L'$, and for every $L' \in \text{Cycl}[\mathcal{S}]_{s',h'}^X(\beta)$ there is $L \in \text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)$ such that $f^{-1}(L') = L$. Similarly, f induces a bijection from $\uparrow\text{Cycl}[\mathcal{S}]_{s,h}^{X,\alpha}$ to $\uparrow\text{Cycl}[\mathcal{S}]_{s',h'}^{X,\alpha}$,
- 5_f. $\text{Rem}[\mathcal{S}]_{s',h'}^{X,\alpha} = f(\text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha})$.

The existence f directly stems from $((s, h), (s', h')) \in (\bigcap_{\alpha' \geq 1} \approx_{X,\alpha'}^{\mathcal{S}})$ and Proposition 5.31. The constraint (1_f) can be satisfied as the two memory states satisfy the same core formulae of the form $t_1 = t_2$ and $u = t$. The second constraint can be enforced thanks to the equisatisfaction of the formulae of the form $\text{sees}_X(t_1, t_2) \geq \beta$ (for every $t_1, t_2 \in T[\mathcal{S}]^X$ and $\beta \geq 1$). Together with the first constraint, notice that the second constraint essentially states that the δ -th location reachable from $\llbracket t \rrbracket_{s,h}^X$ should be mapped through f with the δ -th location reachable from $\llbracket t \rrbracket_{s',h'}^X$ (formally shown below, see (B)). Moreover, from the equisatisfaction of the formulae $u \in \text{sees}_X(t_1, t_2) \geq (\beta_1, \beta_2)$ (for every $t_1, t_2 \in T[\mathcal{S}]^X$ and $\beta_1, \beta_2 \geq 1$), we conclude that if $s(u)$ is the δ -th location reachable from $\llbracket t \rrbracket_{s,h}^X$, then $s'(u)$ is the δ -th location reachable from $\llbracket t \rrbracket_{s',h'}^X$, and vice versa. Thus, the constraints (1_f) and (2_f) are simultaneously satisfiable. Similarly, the constraint (3_f) can be satisfied thanks to the equisatisfaction of the core formulae $\text{pred}_X^S(x) \geq \beta$ and $u \in \text{pred}_X^S(x)$ (for every $x \in X$ and $\beta \geq 1$). The constraint (4_f) relies on the equisatisfaction of the core formulae $\text{loop}_X^S(\beta_1) \geq \beta_2$, and $u \in \text{loop}_X^S(\beta_1)$ (for every $\beta_1, \beta_2 \geq 1$). Notice that, thanks to the hypothesis $((s, h), (s', h')) \in (\bigcap_{\alpha' \geq 1} \approx_{X,\alpha'}^{\mathcal{S}})$, the equisatisfaction of these core formulae entails the equisatisfaction of the formulae $\uparrow\text{loop}_{X,\alpha}^S \geq \beta_1$ and $u \in \uparrow\text{loop}_{X,\alpha}^S$ (for every $\alpha, \beta_1 \geq 1$). Lastly, the constraint (5_f) relies on the equisatisfaction of the core formulae $\text{rem}_{X,\alpha}^S \geq \beta$ and $u \in \text{rem}_{X,\alpha}^S$ (for every $\alpha, \beta \geq 1$). From Proposition 5.31, the sets in Definition 5.29 form a partition, and therefore all these constraints can be simultaneously satisfied.

In order to show that $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{S}} (s', h')$ holds, let us consider two disjoint heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Notice that this requires α to be at least two (otherwise the lemma trivially holds). Let us define the heaps h'_1 and h'_2 required by the relation $\leftrightarrow_{X,\alpha}^{\mathcal{S}}$ as follows:

$$h'_1 = \{(\ell, \ell') \in h' \mid f^{-1}(\ell) \in \text{dom}(h_1)\}, \quad h'_2 = \{(\ell, \ell') \in h' \mid f^{-1}(\ell) \in \text{dom}(h_2)\}.$$

Since f is a bijection and $h_1 \perp h_2$, we have $h' = h'_1 + h'_2$. Let us discuss the following properties (A)–(F) of h'_j (where $j \in \{1, 2\}$), which leads to $(s, h_j) \approx_{X,h_j}^{\mathcal{S}} (s', h'_j)$, as we later show.

- A. For every $t \in T[\mathcal{S}]^X$, $\llbracket t \rrbracket_{s,h_j}^X$ is defined iff so is $\llbracket t \rrbracket_{s',h'_j}^X$. If defined, $f(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t \rrbracket_{s',h'_j}^X$.

Proof of (A). We show that if $\llbracket t \rrbracket_{s,h_j}^X$ is defined then $f(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t \rrbracket_{s',h'_j}^X$. By symmetry, the same holds when $\llbracket t \rrbracket_{s',h'_j}^X$ is defined. The statement is trivial when t is a program variable. For the case of end-point variables, let us assume that there is $x \in X$ such that $t = e(x)$, and that $\llbracket e(x) \rrbracket_{s,h_j}^X$ is defined and equal to ℓ . By definition of $\llbracket . \rrbracket_{s,h_j}^X$, there is $\delta \geq 1$ such that $h_j^\delta(s(x)) = \ell$, and if $\ell \in \text{dom}(h)$ then ℓ belongs to a cycle in h_j whereas $h_j^{\delta-1}(s(x))$ does not. We divide the proof depending on whether $\ell \in \text{dom}(h_j)$.

case: $\ell \notin \text{dom}(h_j)$. h_j witnesses a path going from $s(x)$ to ℓ , and ℓ is the only location reachable from $s(x)$ that does not belong to $\text{dom}(h)$. Let L_1 be the set of locations

that describe this path (ℓ excluded). So, $L_1 \subseteq \text{dom}(h_j)$. As $h_j \subseteq h$, L_1 describes a path of h going from $s(x)$ to ℓ . From the properties (1_f) and (2_f), $f(L_1)$ describes a path of h' going from $s'(x)$ to $f(\ell)$. By definition of h'_j , $f(L_1) \subseteq \text{dom}(h'_j)$ and therefore $f(L_1)$ describes a path of h'_j going from $s'(x)$ to $f(\ell)$. Moreover, $f(\ell) \notin \text{dom}(h'_j)$. So, by definition of end-point variables, $f(\ell) = [\![e(x)]\!]_{s',h'_j}^X$.

case: $\ell \in \text{dom}(h_j)$. h_j witnesses two disjoint non-empty paths, one going from $s(x)$ to ℓ and one going from ℓ to itself. Let L_1 be the minimal set of locations that describe the first path (ℓ is excluded), and L_2 be the minimal set of locations that describe the second one. Notice that $L_1 \cup L_2 \subseteq \text{dom}(h_j)$ and $L_1 \cap L_2 = \emptyset$. Since $h_j \subseteq h$, L_1 describes a path in h going from $s(x)$ to ℓ , and L_2 describes a path in h going from ℓ to itself. From the properties (1_f) and (2_f), $f(L_1)$ describes a path in h' going from $s'(x)$ to $f(\ell)$, and $f(L_2)$ describes a path going from $f(\ell)$ to $f(\ell)$. Moreover, $f(L_1) \cap f(L_2) = \emptyset$. By definition of h'_j , $f(L_1) \cup f(L_2) \subseteq \text{dom}(h'_j)$. So, h'_j witnesses two disjoint non-empty paths, one going from $s(x)$ to ℓ and one going from ℓ to itself. By definition of end-point variables, $f(\ell) = [\![e(x)]\!]_{s',h'_j}^X$.

Let us now consider the case of meet-point variables. Suppose there are $x, y \in X$ such that $t = m(x, y)$ and $[\![m(x, y)]\!]_{s,h_j}^X = \ell$. By definition, there are $\delta_1, \delta_2 \geq 1$ such that $h_j^{\delta_1}(s(x)) = h_j^{\delta_2}(s(y)) = \ell$ and for all $\delta'_1 \in [0, \delta_1]$, $\delta'_2 \in [0, \delta_2]$, if $\delta'_1 + \delta'_2 < \delta_1 + \delta_2$ then $h_j^{\delta'_1}(s(x)) \neq h_j^{\delta'_2}(s(y))$. Moreover, for every $\delta' \geq 1$, it holds that $h^{\delta'}(\ell) \neq \ell$. Informally, this means that h_j witnesses two non-empty disjoint paths, one going from $s(x)$ to ℓ and one going from $s(y)$ to ℓ , where ℓ is a location that does not belong to a cycle. Let L_1 (resp. L_2) be the minimal set of locations that describe the path in h_j going from $s(x)$ (resp. $s(y)$) to ℓ . Notice that $L_1 \cup L_2 \subseteq \text{dom}(h_j)$ and $L_1 \cap L_2 = \emptyset$. As $h_j \subseteq h$, L_1 (resp. L_2) describes a path in h going from $s(x)$ (resp. $s(y)$) to ℓ . From the properties (1_f) and (2_f), $f(L_1)$ (resp. $f(L_2)$) describes a path in h' going from $s'(x)$ (resp. $s'(y)$) to $f(\ell)$. Moreover, $f(L_1) \cap f(L_2) = \emptyset$ and, by definition of h'_j , $f(L_1) \cup f(L_2) \subseteq \text{dom}(h'_j)$. Thus, h'_j witnesses two disjoint non-empty paths, one going from $s'(x)$ to $f(\ell)$ and one going from $s'(y)$ to $f(\ell)$. In order to conclude that $[\![m(x, y)]\!]_{s'}^X = f(\ell)$ it is sufficient to show that h'_j does not witness a cycle involving $f(\ell)$. *Ad absurdum*, suppose there is $\delta' \geq 1$ such that $h_j'^{\delta'}(f(\ell)) = f(\ell)$. Let L' be the minimal set of locations that describes this cycle of h'_j , i.e. $L' \stackrel{\text{def}}{=} \{\ell' \mid \text{there is } \delta' \geq 1, h_j'^{\delta'}(f(\ell)) = \ell'\}$. In particular, $f(\ell) \in L'$ and $L' \subseteq \text{dom}(h'_j)$. As $h'_j \subseteq h'$, L' describes a cycle in h' . From the properties (1_f) and (2_f), $f^{-1}(L')$ describes a cycle in h . By definition of h'_j , $f^{-1}(L') \subseteq \text{dom}(h_j)$. This means that $f^{-1}(L')$ describes a cycle in h_j . However, from $f(\ell) \in L'$, we conclude that $\ell \in f(L')$ (f is a bijection). This is contradictory, as ℓ does not belong to a cycle of h_j . Therefore, $f(\ell)$ does not belong to a cycle of h'_j . We conclude: $f(\ell) = [\![m(x, y)]\!]_{s',h'_j}^X$.

B. For every $t \in T[S]^X$,

- (a) when defined, $\text{sby}_{s',h'_j}^X(t) = f(\text{sby}_{s,h_j}^X(t))$,
- (b) $\text{Path}[S]_{s',h'_j}^X(t) = f(\text{Path}[S]_{s,h_j}^X(t))$,
- (c) let $\delta \in [1, \text{card}(\text{Path}[S]_{s,h_j}^X(t))]$. $h_j^\delta([\![t]\!]_{s,h_j}^X) = s(u)$ iff $h_j'^\delta([\![t]\!]_{s',h'_j}^X) = s'(u)$.

First, let us show the following result:

$$\text{for every } x \in X \text{ and } \delta \geq 0, h^\delta(s(x)) = \ell \text{ iff } h'^\delta(s'(x)) = f(\ell). \quad (\dagger)$$

We discuss the left-to-right direction. The other direction is symmetrical. The proof is by induction on δ .

base case: $\delta = 0$. Directly from (1_f).

induction step: $\delta \geq 1$. Let $\ell = h^\delta(s(x))$ and consider the greatest $\delta' \in [0, \delta - 1]$ such that $h^{\delta'}(s(x)) \in \text{Lab}[s]_{s,h}^X$. Since $s(x) \in \text{Lab}[s]_{s,h}^X$, δ' is well-defined. Let $t \in T[s]^X$ such that $\llbracket t \rrbracket_{s,h}^X = h^{\delta'}(s(x))$. By induction hypothesis and (1_f), $h'^{\delta'}(s'(x)) = f(\llbracket t \rrbracket_{s,h}^X) = \llbracket t \rrbracket_{s',h'}^X$. We divide the proof depending on whether $\ell = \text{sby}_{s,h}^X(t)$.

case: $\ell \neq \text{sby}_{s,h}^X(t)$. From the definition of t , we have that $\ell \in \text{Path}[s]_{s,h}^X(t)$. From the property (2_f), $f(h^{\delta-\delta'}(\llbracket t \rrbracket_{s,h}^X)) = h'^{\delta-\delta'}(\llbracket t \rrbracket_{s',h'}^X)$. This implies that $f(h^\delta(s(x))) = h'^\delta(s'(x))$, and so $h'^\delta(s'(x)) = f(\ell)$.

case: $\ell = \text{sby}_{s,h}^X(t)$. We notice that, for all $\delta'' \in [\beta' + 1, \beta']$, $h^{\delta''}(s(x)) \neq \text{sby}_{s,h}^X(t)$. Indeed, otherwise we would reach a contradiction with the fact that δ' is the greatest integer in $[0, \beta - 1]$ such that $h^{\delta'}(s(x)) \in \text{Lab}[s]_{s,h}^X$. This implies that the location $\ell' = h^{\delta-1}(s(x))$ belongs to $\text{Path}[s]_{s,h}^X(t)$. So, $\delta = \text{card}(\text{Path}[s]_{s,h}^X(t))$. By induction hypothesis, we have $h'^{\delta-1}(s'(x)) = f(\ell')$. By $((s, h), (s', h')) \in (\cap_{\alpha' \geq 1} \approx_{X,\alpha'}^S)$, $\text{card}(\text{Path}[s]_{s,h}^X(t)) = \text{card}(\text{Path}[s]_{s',h'}^X(t))$, which implies that $h'(f(\ell)) = \text{sby}_{s',h'}^X(t)$. From (1_f), $f(\text{sby}_{s,h}^X(t)) = \text{sby}_{s',h'}^X(t)$. Therefore, $h'^\delta(s'(x)) = f(\ell)$.

Thanks to (†), from the definition of h'_j we conclude that for every $x \in X$ and $\delta \geq 0$, $h_j^\delta(s(x)) = \ell$, if and only if $h_j'^\delta(s'(x)) = f(\ell)$. Directly from (A), this double implication can be extended to arbitrary terms:

$$\text{for every } t \in T[s]^X \text{ and } \delta \geq 0, h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) = \ell \text{ iff } h_j'^\delta(\llbracket t \rrbracket_{s',h'_j}^X) = f(\ell). \quad (\ddagger)$$

Let us now prove the statements (a)–(c).

Proof of (a). By definition, $\text{sby}_{s',h'_j}^X(t)$ belongs to $\text{Lab}[s]_{s',h'_j}^X$ and there is $\delta \geq 1$ such that $h_j'^\delta(\llbracket t \rrbracket_{s',h'_j}^X) = \text{sby}_{s',h'_j}^X(t)$ and for every $\delta' \in [1, \delta - 1]$, $h_j'^{\delta'}(\llbracket t \rrbracket_{s',h'_j}^X) \notin \text{Lab}[s]_{s',h'_j}^X$. Informally, $\text{sby}_{s',h'_j}^X(t)$ is the first labelled location reachable in h'_j starting from $\llbracket t \rrbracket_{s',h'_j}^X$. From (†), $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) = f^{-1}(\text{sby}_{s',h'_j}^X(t))$. By (A), this implies $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) \in \text{Lab}[s]_{s,h_j}^X$. Ad absurdum, suppose that $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X)$ is not the first labelled location reachable in h_j starting from $\llbracket t \rrbracket_{s,h_j}^X$. So, $h_j'^{\delta'}(\llbracket t \rrbracket_{s,h_j}^X) \in \text{Lab}[s]_{s,h_j}^X$ for some $\delta' \in [1, \delta - 1]$. By (A), $h_j'^{\delta'}(\llbracket t \rrbracket_{s',h'_j}^X) \in \text{Lab}[s]_{s',h'_j}^X$. However, as $\delta' \in [1, \delta - 1]$, this contradicts the statement $h_j'^\delta(\llbracket t \rrbracket_{s',h'_j}^X) = \text{sby}_{s',h'_j}^X(t)$. Therefore, $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X)$ is the first labelled location reachable in h_j starting from $\llbracket t \rrbracket_{s,h_j}^X$. We conclude that $\text{sby}_{s',h'_j}^X(t) = f(\text{sby}_{s,h_j}^X(t))$.

Proof of (b). (\subseteq): Suppose $\ell' \in \text{Path}[s]_{s',h'_j}^X(t)$. By definition, $\ell' \in \text{dom}(h'_j)$ and there is $\delta \geq 0$ such that $h_j'^\delta(\llbracket t \rrbracket_{s',h'_j}^X) = \ell'$ and for every $\delta' \in [1, \delta]$, $h_j'^{\delta'}(\llbracket t \rrbracket_{s',h'_j}^X) \neq \text{sby}_{s',h'_j}^X(t)$. By (†), $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) = f^{-1}(\ell')$, and to prove that $f^{-1}(\ell') \in \text{Path}[s]_{s,h_j}^X(t)$, it is sufficient to show that for all $\delta' \in [1, \delta]$, $h_j'^{\delta'}(\llbracket t \rrbracket_{s,h_j}^X) \neq \text{sby}_{s,h_j}^X(t)$. Ad absurdum, let us assume that there is $\delta' \in [1, \delta]$ such that $h_j'^{\delta'}(\llbracket t \rrbracket_{s,h_j}^X) = \text{sby}_{s,h_j}^X(\llbracket t \rrbracket_{s,h_j}^X)$. By (†), $h_j'^{\delta'}(\llbracket t \rrbracket_{s',h'_j}^X) = f(\text{sby}_{s,h_j}^X(t))$. However, by (a), this implies $h_j'^{\delta'}(\llbracket t \rrbracket_{s',h'_j}^X) = \text{sby}_{s',h'_j}^X(t)$, a contradiction. Thus, $f^{-1}(\ell') \in \text{Path}[s]_{s,h_j}^X(t)$.

(\supseteq): Symmetrical to the other direction.

Proof of (c). Directly from (‡) and $f(s(u)) = s'(u)$ (property (1_f)).

- C. For every $x \in X$, $\text{Pred}[s]_{s',h'_j}^X(x) = f(\text{Pred}[s]_{s,h_j}^X(x))$,

Proof of (C). (\subseteq): Consider $\ell' \in \text{Pred}[s]_{s',h'_j}^X(x)$. By definition, $h'_j(\ell') = s'(x)$ and for every $y \in X$ and $\delta \geq 0$, we have $h'^{\delta}(s'(y)) \neq \ell'$. We divide the proof in two cases, depending on whether ℓ' is reached, in h' , by a location corresponding to a program variable. Let $\ell = f^{-1}(\ell')$.

case: $h'^{\delta}(s'(y)) = \ell'$ for some $y \in X$ and $\delta \geq 0$. We have $h'^{\delta+1}(s'(y)) = s'(x)$. By (†), we derive $h^{\delta}(s(y)) = \ell$ and $h^{\delta+1}(s(y)) = f^{-1}(s'(x))$. With the property (1_f), this implies that $h(\ell) = s(x)$. Since $\ell' \in \text{dom}(h'_j)$, by definition of h'_j we conclude that $\ell \in \text{dom}(h_j)$ and so $h_j(\ell) = s(x)$. To conclude that $\ell \in \text{Pred}[s]_{s,h_j}^X(x)$, we show that for all $y \in X$ and $\delta' \geq 0$, $h_j^{\delta'}(s(y)) \neq \ell$. *Ad absurdum*, suppose there are $x \in X$ and $\delta' \geq 0$ such that $h_j^{\delta'}(s(y)) = \ell$. By (‡), $h_j^{\delta'}(s'(y)) = f(\ell) = \ell'$. However, this contradicts the fact that $\ell' \in \text{Pred}[s]_{s',h'_j}^X(x)$. Thus, $\ell \in \text{Pred}[s]_{s,h_j}^X(x)$.

case: for all $y \in X$ and $\delta \geq 0$, $h'^{\delta}(s'(y)) \neq \ell'$. Since $h'_j \subseteq h'$, we have $h'(\ell') = s'(x)$. So, $\ell' \in \text{Pred}[s]_{s',h'}^X(x)$. By (3_f), $\ell \in \text{Pred}[s]_{s,h}^X(x)$. By definition of h'_j , $\ell \in \text{dom}(h_j)$ and thus $h_j(\ell) = s(x)$. From $\ell \in \text{Pred}[s]_{s,h}^X(x)$, no program variable $y \in X$ is such that $h_j^{\delta'}(s(y)) = \ell$, for any $\delta' \geq 0$. By definition, $\ell \in \text{Pred}[s]_{s,h_j}^X(x)$.

(\supseteq): Symmetrical to the other direction.

- D. For every $\beta \in [1, \alpha_j]$, f induces a bijection from $\text{Cycl}[s]_{s,h_j}^X(\beta)$ to $\text{Cycl}[s]_{s',h'_j}^X(\beta)$,

Proof of (D). (\Rightarrow): Consider a set $L \in \text{Cycl}[s]_{s,h_j}^X(\beta)$. We prove that $f(L) \in \text{Cycl}[s]_{s',h'_j}^X(\beta)$.

By definition $L \cap \text{Lab}[s]_{s,h_j}^X = \emptyset$ and L describes a cycle in h_j of length β . Let $\ell_0, \dots, \ell_{\beta-1}$ be the β locations in L , so that for all $k \in [0, \beta-1]$, $h(\ell_k) = \ell_{(k+1 \bmod \beta)}$. We divide the proof depending on whether L contains a location in $\text{Lab}[s]_{s,h}^X$.

case: $L \cap \text{Lab}[s]_{s,h}^X \neq \emptyset$. Let $k \in [0, \beta-1]$ be an index such that $\ell_k \in \text{Lab}[s]_{s,h}^X$. From (1_f), $f(\ell_k) \in \text{Lab}[s]_{s,h}^X$. As ℓ_k corresponds to a term, we know that h witnesses a (possibly empty) path going from a location corresponding to a program variable to ℓ_k . By (†), for every $\delta \geq 0$, $h^{\delta}(\ell_k) = h'^{\delta}(f(\ell_k))$. As $h_j \subseteq h$, this implies that for every $i \in [0, \beta-1]$, $h'(f(\ell_i)) = f(\ell_{(i+1 \bmod \beta)})$. Since f is a bijection, $f(L)$ describes a cycle of length β in h' . By definition of h'_j , $f(L) \in \text{dom}(h'_j)$ and therefore $f(L)$ describes a cycle of length β in h'_j . Since $L \cap \text{Lab}[s]_{s,h_j}^X = \emptyset$, from (A) we conclude that $f(L) \cap \text{Lab}[s]_{s',h'_j}^X = \emptyset$. Thus, $f(L) \in \text{Cycl}[s]_{s',h'_j}^X(\beta)$.

case: $L \cap \text{Lab}[s]_{s,h}^X = \emptyset$. From $h_j \subseteq h$, we conclude that L describes a cycle of length β in h . Since the locations in L do not belong to $\text{Lab}[s]_{s,h}^X$, L is in $\text{Cycl}[s]_{s,h}^X(\beta)$. By (4_f), $f(L) \in \text{Cycl}[s]_{s',h}^X(\beta)$. By definition of h'_j , $f(L) \subseteq \text{dom}(h'_j)$. So, $f(L)$ describes a cycle of length β in h'_j . Since $L \cap \text{Lab}[s]_{s,h_j}^X = \emptyset$, from (A) we conclude that $f(L) \cap \text{Lab}[s]_{s',h'_j}^X = \emptyset$. Thus, $f(L) \in \text{Cycl}[s]_{s',h'_j}^X(\beta)$.

(\Leftarrow): Symmetrical to the other direction.

- E. f induces a bijection from $\text{↑Cycl}[s]_{s,h_j}^{X,\alpha}$ to $\text{↑Cycl}[s]_{s',h'_j}^{X,\alpha}$,

The proof of this statement is analogous to the one for the case (D).

- F. $\text{Rem}[s]_{s',h'_j}^{X,\alpha} = f(\text{Rem}[s]_{s,h_j}^{X,\alpha})$.

Proof of (F). (\subseteq): Let $\ell' \in \text{Rem}[s]_{s', h'_j}^{x, \alpha}$. Thus, $\ell' \in \text{dom}(h'_j)$, and moreover:

- * for every $x \in X$, $\ell' \notin \text{Pred}[s]_{s', h'_j}^X(x)$. Therefore, $f^{-1}(\ell') \notin \text{Pred}[s]_{s, h_j}^X(x)$ by (C),
- * for every $\ell_1 \in \text{Lab}[s]_{s', h'_j}^X$, $\ell' \notin \text{Path}[s]_{s', h'_j}^X(\ell_1)$. Therefore, from (B)(b) and (A), for every $\ell_2 \in \text{Lab}[s]_{s, h_j}^X$, $f^{-1}(\ell') \notin \text{Path}[s]_{s, h_j}^X(\ell_2)$,
- * for every $\beta \in [1, \alpha]$, $\ell' \notin \text{Cycl}[s]_{s', h'_j}^X(\beta)$. Thus, $f^{-1}(\ell') \notin \text{Cycl}[s]_{s, h_j}^X(\beta)$ by (D),
- * $\ell' \notin \uparrow\text{Cycl}[s]_{s', h'_j}^{x, \alpha}$. Therefore, $f^{-1}(\ell') \notin \uparrow\text{Cycl}[s]_{s, h_j}^{x, \alpha}$ by (E).

By definition of h'_j , $f^{-1}(\ell') \in \text{dom}(h_j)$. We conclude that $f^{-1}(\ell') \in \text{Rem}[s]_{s, h_j}^{x, \alpha}$.

(\supseteq): Symmetrical to the other direction.

The properties (A)–(F) lead directly to $(s, h_j) \approx_{X, \alpha_j}^S (s', h'_j)$. Indeed, let us consider a core formula φ in $\text{Core}[s](X, \alpha_j)$. We have $(s, h_j) \models \varphi$ iff $(s', h'_j) \models \varphi$, as discussed below:

case: $\varphi = t_1 = t_2$. Follows directly from (A).

case: $\varphi = \text{sees}_X(t_1, t_2) \geq \beta$. Follows from (A), (B)((a) and (b)), and the bijectivity of f .

case: $\varphi = \text{pred}_X^S(x) \geq \beta$. Follows directly from (C) and the bijectivity of f .

case: $\varphi = \text{loop}_X^S(\beta_1) \geq \beta_2$. Follows directly from (D) and the bijectivity of f .

case: $\varphi = \uparrow\text{loop}_X^S \geq \beta$. Follows directly from (E) and the bijectivity of f .

case: $\varphi = \text{rem}_{X, \alpha}^S \geq \beta$. Follows directly from (F) and the bijectivity of f .

case: $\varphi = u = t$. Follows directly from (A) and since $f(s(u)) = s'(u)$ (property (1_f) of f).

case: $\varphi = u \in \text{sees}_X(t_1, t_2) \geq (\beta_1, \beta_2)$. Follows directly from (B).

case: $\varphi = u \in \text{pred}_X^S(x)$. Follows directly from (C) and $f(s(u)) = s'(u)$.

case: $\varphi = u \in \text{loop}_X^S(\beta)$. Follows directly from (D) and $f(s(u)) = s'(u)$.

case: $\varphi = u \in \uparrow\text{loop}_X^S$. Follows directly from (E) and $f(s(u)) = s'(u)$.

case: $\varphi = u \in \text{rem}_{X, \alpha}^S$. Follows directly from (F) and $f(s(u)) = s'(u)$.

Therefore, $(s, h) \leftrightarrow_{X, \alpha}^S (s', h')$. □

Alongside Lemma 5.39, we show five intermediate results, one for each type of sets in the partition defined in Section 5.5.1, which allows us to build the chain of hops. Similarly to Lemma 5.19, in each of these results we consider two memory states (s, h) and (s, h') where the latter memory state is obtained from the former by slightly updating one set of the partition.

Lemma 5.40. Let (s, h) and (s, h') be two memory states such that $(s, h) \approx_{X, \alpha}^S (s, h')$ and for every $t \in T[s]^X$, $\llbracket t \rrbracket_{s, h}^X = \llbracket t \rrbracket_{s, h'}^X$. We have $(s, h) \leftrightarrow_{X, \alpha}^S (s, h')$ whenever one of the following holds:

- (I) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[s]_{s, h}^X(x)\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[s]_{s, h'}^X(x)\}$, for some $x \in X$,
- (II) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Path}[s]_{s, h}^X(\tilde{\ell})\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Path}[s]_{s, h'}^X(\tilde{\ell})\}$, for $\tilde{\ell} \in \text{Lab}[s]_{s, h}^X$,
- (III) $h \setminus \{(\ell, \ell') \in h \mid \ell \in [\text{Cycl}[s]_{s, h}^X(\beta)]^\flat\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in [\text{Cycl}[s]_{s, h'}^X(\beta)]^\flat\}$, for $\beta \in [1, \alpha]$,
- (IV) $h \setminus \{(\ell, \ell') \in h \mid \ell \in [\uparrow\text{Cycl}[s]_{s, h}^{X, \alpha}]^\flat\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in [\uparrow\text{Cycl}[s]_{s, h'}^{X, \alpha}]^\flat\}$,
- (V) $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[s]_{s, h}^{X, \alpha}\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[s]_{s, h'}^{X, \alpha}\}$.

The proofs of the statements (I)–(V) are, in their essence, all very similar. However, they heavily depend on the upper bounds given to the formulae in $\text{Core}[s](\mathbf{x}, \alpha)$ and discussed in Section 5.5.2. Because of these upper bounds, the statements (II), (III) and (IV) are definitely the ones that require a more sophisticated analysis. The proofs of (III) and (IV) are quite close, in view of the similarities between $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$ and $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}$. In what follows, we show the proofs for the statements (II) and (III), and refer the reader to Appendix C for the proofs of the statements (I), (IV) and (V). Despite the modularity introduced via game hopping, the proofs of (II) and (III) spawn several pages, and show that the technical constructions needed to prove the $*$ -simulation are hardly avoidable. Let us start with the easier proof, i.e. Lemma 5.40(III).

Proof of (III). Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Notice that this requires α to be at least two, otherwise the statement trivially holds. We write $\mathcal{L}(\alpha)$ and $\mathcal{R}(\alpha)$ for the upper bounds given to β' in formulae of the form $\text{loop}_{\mathbf{x}}^S(\beta) \geq \beta'$ and $\text{rem}_{\mathbf{x},\alpha}^S \geq \beta'$, respectively. That is, $\mathcal{L}(\alpha) = \frac{1}{2}\alpha(\alpha+3) - 1$ and $\mathcal{R}(\alpha) = \alpha$. As discussed during Example 5.33, we have:

$$\mathcal{L}(\alpha) \geq \mathcal{L}(\alpha_1) + \mathcal{L}(\alpha_2) + \mathcal{R}(\max(\alpha_1, \alpha_2)) + 1. \quad (\star)$$

Besides, if $\text{card}(\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)) < \mathcal{L}(\alpha)$ then the lemma holds directly from Lemma 5.39. Indeed, in this case, from the equisatisfaction of the core formulae of the form $\text{loop}_{\mathbf{x}}^S(\beta) \geq \beta'$ we conclude that $\text{card}(\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)) = \text{card}(\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}(\beta))$. By definition of h and h' we conclude that $((s, h), (s, h')) \in (\cap_{\alpha' \geq 1} \approx_{\mathbf{x},\alpha'}^S)$, which allows us to apply Lemma 5.39. Therefore, in the following we assume $\text{card}(\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)) \geq \mathcal{L}(\alpha)$, which implies $\text{card}(\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)) \geq \mathcal{L}(\alpha)$ again from the equisatisfaction of the core formulae $\text{loop}_{\mathbf{x}}^S(\beta) \geq \beta'$.

Assumption. Both $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$ and $\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}(\beta)$ have at least $\mathcal{L}(\alpha)$ elements.

We define the following three sets T_1 , T_2 and S :

- $T_1 = \{L \in \text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta) \mid L \subseteq \text{dom}(h_1)\}$,
- $T_2 = \{L \in \text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}(\beta) \mid L \subseteq \text{dom}(h_2)\}$,
- $S = \{(L_1, L_2) \mid L_1 \cup L_2 \in \text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta), \emptyset \neq L_1 \subseteq \text{dom}(h_1), \emptyset \neq L_2 \subseteq \text{dom}(h_2)\}$.

Given $j \in [1, 2]$, T_j contains every set of $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$ whose locations are memory cells of h_j . This means that the sets in T_1 and T_2 describe loops of length β in h_1 and h_2 , respectively. Instead, S contains pairs (L_1, L_2) of non-empty sets of locations that partition a set of $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$ and are such that L_1 contains memory cells of h_1 whereas L_2 contains memory cells of h_2 . Informally, S represent the loops in $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$ that are split between h_1 and h_2 . In particular, S is empty when $\beta = 1$, i.e. if we are dealing with self-loops. Notice that T_1 , T_2 and $\{L_1 \cup L_2 \mid (L_1, L_2) \in S\}$ are mutually disjoint sets, and that their union is $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$. The first crucial step in the proof is to define three similar sets T'_1 , T'_2 and S' , with respect to the set $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$. These sets should also satisfy cardinality constraints depending on $\mathcal{L}(\alpha_j)$ and $\mathcal{R}(\alpha_j)$ (where $j \in \{1, 2\}$), as well as constraints involving the location $s(u)$. More precisely, the following six properties shall be satisfied:

1. $S' \subseteq \{(L'_1, L'_2) \mid L'_1$ and L'_2 are non-empty and disjoint, and $L'_1 \cup L'_2 \in \text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)\}$,
2. T'_1 , T'_2 and $\{L'_1 \cup L'_2 \mid (L'_1, L'_2) \in S'\}$ are mutually disjoint. Their union is $\text{Cycl}[s]_{s,h}^{\mathbf{x}}(\beta)$,
3. for all $j \in \{1, 2\}$, $\min(\text{card}(T_j), \mathcal{L}(\alpha_j)) = \min(\text{card}(T'_j), \mathcal{L}(\alpha_j))$,
4. for all $j \in \{1, 2\}$, $\min(\text{card}([\pi_j(S)]^b), \mathcal{R}(\alpha_j)) = \min(\text{card}([\pi_j(S')]^b), \mathcal{R}(\alpha_j))$,
5. for all $j \in \{1, 2\}$, $s(u) \in [T_j]^b$ if and only if $s(u) \in [T'_j]^b$,

```

1: if  $\text{card}(S) \geq \max(\alpha_1, \alpha_2)$  then
2:   let  $T'_1 \subseteq \text{Cycl}[s]_{s,h}^X(\beta)$  such that •  $\text{card}(T'_1) = \min(\text{card}(T_1), \mathcal{L}(\alpha_1))$ ,
       •  $s(u) \in [T'_1]^\flat$  iff  $s(u) \in [T_1]^\flat$ .
3:   let  $T'_2 \subseteq \text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_1$  such that •  $\text{card}(T'_2) = \min(\text{card}(T_2), \mathcal{L}(\alpha_2))$ ,
       •  $s(u) \in [T'_2]^\flat$  iff  $s(u) \in [T_2]^\flat$ .
4:   if  $s(u) \in [\pi_1(S)]^\flat$  then
5:      $S' \leftarrow \{(L' \setminus \{\ell'\}, \{\ell'\}) \mid L' \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2) \text{ and } \ell' = \min(L' \setminus \{s(u)\})\}$ .
6:   else  $S' \leftarrow \{(\{\ell'\}, L' \setminus \{\ell'\}) \mid L' \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2) \text{ and } \ell' = \min(L' \setminus \{s(u)\})\}$ .
7: else
8:   let  $i \in \{1, 2\}$  such that  $\text{card}(T_i) \geq \mathcal{L}(\alpha_i)$ .
9:   let  $T'_{3-i} \subseteq \text{Cycl}[s]_{s,h}^X(\beta)$  such that •  $\text{card}(T'_{3-i}) = \min(\text{card}(T_{3-i}), \mathcal{L}(\alpha_{3-i}))$ ,
       •  $s(u) \in [T'_{3-i}]^\flat$  iff  $s(u) \in [T_{3-i}]^\flat$ .
10:  let  $Q$  be a set of  $\text{card}(S)$  sets in  $\text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_{3-i}$  such that  $s(u) \in [Q]^\flat$  iff  $s(u) \in [S]^\flat$ .
11:  let  $f: [Q]^\flat \rightarrow [S]^\flat$  be a bijection s.t. • for all  $L' \in Q$  there is  $(L_1, L_2) \in S$ ,  $f(L') = L_1 \cup L_2$ ,
       • if  $s(u) \in [Q]^\flat$  then  $f(s(u)) = s(u)$ .
12:   $S' \leftarrow \{(L'_1, L'_2) \mid L'_1 \cup L'_2 \in Q, (f(L'_1), f(L'_2)) \in S\}$ .
13:   $T'_i \leftarrow \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_{3-i} \cup Q)$ .

```

Figure 5.14: Strategy used to define T'_1 , T'_2 and S' .

6. for all $j \in \{1, 2\}$, $s(u) \in [\pi_j(S)]^\flat$ if and only if $s(u) \in [\pi_j(S')]^\flat$.

The definition of T'_1 , T'_2 and S' follows the strategy described in Figure 5.14. We remind the reader that both the sets $\text{Cycl}[s]_{s,h}^X(\beta)$ and $\text{Cycl}[s]_{s,h}^X(\beta)$ contain at least $\mathcal{L}(\alpha)$ elements. We show that, following this strategy, T'_1 , T'_2 and S' are well-defined and satisfy the properties (1)–(6). The proof is divided in two cases, depending on whether $\text{card}(S) \geq \max(\alpha_1, \alpha_2)$.

case: $\text{card}(S) \geq \max(\alpha_1, \alpha_2)$. We consider the lines 2–6 of the strategy. First (line 2), T'_1 is defined as a subset of $\min(\text{card}(T_1), \mathcal{L}(\alpha_1))$ elements of $\text{Cycl}[s]_{s,h}^X(\beta)$, such that $s(u) \in [T'_1]^\flat$ if and only if $s(u) \in [T_1]^\flat$. T'_1 is well-defined:

- From $\text{card}(\text{Cycl}[s]_{s,h}^X(\beta)) \geq \mathcal{L}(\alpha)$ and (★), the set $\text{Cycl}[s]_{s,h}^X(\beta)$ contains more than $\min(\text{card}(T_1), \mathcal{L}(\alpha_1))$ elements.
- As (s, h) and (s, h') equisatisfy the formula $u \in \text{loop}_X^S(\beta)$, $s(u) \in [\text{Cycl}[s]_{s,h}^X(\beta)]^\flat$ holds if and only if $s(u) \in [\text{Cycl}[s]_{s,h}^X(\beta)]^\flat$. If $[T_1]^\flat$ contains $s(u)$, then by definition of T_1 so does $[\text{Cycl}[s]_{s,h}^X(\beta)]^\flat$. Thus, we are able to construct T'_1 so that it contains the set $L' \in \text{Cycl}[s]_{s,h}^X(\beta)$ such that $s(u) \in L'$. Otherwise ($s(u) \notin [T_1]^\flat$), as $\text{Cycl}[s]_{s,h}^X(\beta)$ contains more than $\mathcal{L}(\alpha_1)$ elements, we can construct T'_1 without using sets that contain $s(u)$ (if any). Also, notice that in both cases we have

$$s(u) \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus T_1 \text{ if and only if } s(u) \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_1. \quad (\dagger_1)$$

The definition of T'_1 implies that both the properties (3) and (5) are satisfied for $j = 1$. Now, let us look at T'_2 , which is defined (line 3) as a subset of $\min(\text{card}(T_2), \mathcal{L}(\alpha_2))$ sets in $\text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_1$ such that $s(u) \in [T'_2]^\flat$ if and only if $s(u) \in [T_2]^\flat$. Again, T'_2 is well-defined:

- From the definition of T'_1 , $\text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_1$ contains at least $\mathcal{L}(\alpha) - \mathcal{L}(\alpha_1)$ elements, where $\mathcal{L}(\alpha) - \mathcal{L}(\alpha_1) > \mathcal{L}(\alpha_2) \geq \min(\text{card}(T_2), \mathcal{L}(\alpha_2))$ by (★).
- By definition of T_2 , if $s(u) \in [T_2]^\flat$ then $s(u) \in [\text{Cycl}[s]_{s,h}^X(\beta) \setminus T_1]^\flat$. In this case, from (†₁), we can construct T'_2 so that it contains the set $L' \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_1$

such that $s(u) \in L'$. Otherwise ($s(u) \notin [T_2]^\beta$), as $\text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_1$ contains more than $\min(\text{card}(T_2), \mathcal{L}(\alpha_2))$ elements, we can construct T'_2 without using sets that contain $s(u)$. In both cases we conclude that

$$s(u) \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T_1 \cup T_2) \text{ if and only if } s(u) \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2). \quad (\ddagger_2)$$

Following the definitions of T'_1 and T'_2 , we conclude that both the properties (3) and (5) are satisfied. Lastly, let us look at the definition of S'_1 , which is split into two cases (line 5 and 6, respectively). In both cases, for each $L' \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2)$, S'_1 contains exactly one pair (L'_1, L'_2) of non-empty disjoint sets L'_1 and L'_2 such that $L'_1 \cup L'_2 = L'$ (this holds thanks to the min function used in the definition of S'). Therefore, the property (1) is satisfied. Moreover, as $\{L'_1 \cup L'_2 \mid (L'_1, L'_2) \in T'_1\} = \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2)$, where T'_1 and T'_2 are disjoint subsets of $\text{Cycl}[s]_{s,h}^X(\beta)$, we conclude that the property (2) is satisfied. Let us now show that the same holds true for the properties (4) and (6).

- First of all, the hypothesis $\text{card}(S) \geq \max(\alpha_1, \alpha_2)$ implies that S contains at least $\max(\alpha_1, \alpha_2)$ pairs of non-empty sets of locations. This implies that $\text{card}([\pi_1(S)]^\beta) \geq \alpha_1$ and $\text{card}([\pi_2(S)]^\beta) \geq \alpha_2$. Similarly, S' contains $\text{card}(\text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2))$ pairs of non-empty sets of locations. By definition of T'_1 and T'_2 , together with (★):

$$\text{card}(\text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2)) \geq \mathcal{L}(\alpha) - \mathcal{L}(\alpha_1) - \mathcal{L}(\alpha_2) \geq \max(\alpha_1, \alpha_2).$$

Hence, $\text{card}([\pi_1(S')]^\beta) \geq \alpha_1$ and $\text{card}([\pi_2(S')]^\beta) \geq \alpha_2$. The property (4) is satisfied.

- In order to prove the property (6), we first notice that, by definition of T_1 , T_2 and S

$$(s(u) \in [\pi_1(S)]^\beta \text{ or } s(u) \in [\pi_2(S)]^\beta) \text{ iff } s(u) \in [\text{Cycl}[s]_{s,h}^X(\beta) \setminus (T_1 \cup T_2)]^\beta. \quad (\ddagger_3)$$

Now, let us now analyse the definition of S' . If $s(u) \in [\pi_1(S)]^\beta$, we follow the definition in line 5 and conclude that every element $(L'_1, L'_2) \in S'$ is such that $L'_2 = \{\ell'\}$, for some location $\ell' \neq s(u)$. From (†2) and (†3), $s(u) \in [\text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2)]^\beta$ and therefore $s(u)$ belongs to a set in $\pi_1(S')$, as required by the property (6). Otherwise ($s(u) \notin [\pi_1(S)]^\beta$), following the definition in line 6, every element $(L'_1, L'_2) \in S'$ is such that $L'_2 = \{\ell'\}$, for some $\ell' \neq s(u)$. So, $s(u) \notin [\pi_1(S')]^\beta$. If $s(u) \in [\pi_2(S)]^\beta$ then, again by (†2) and (†3), $s(u) \in [\text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2)]^\beta$, and therefore $s(u) \in [\pi_2(S')]^\beta$. If $s(u) \notin [\pi_2(S)]^\beta$ then, together with $s(u) \notin [\pi_1(S)]^\beta$, (†2) and (†3), we conclude that $s(u) \notin [\text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_1 \cup T'_2)]^\beta$. So $s(u) \notin [\pi_2(S')]^\beta$. The property (6) holds.

case: $\text{card}(S) < \max(\alpha_1, \alpha_2)$. We consider the lines 8–13 of the strategy. First of all, since $\text{card}(T_1) + \text{card}(T_2) + \text{card}(S) = \text{card}(\text{Cycl}[s]_{s,h}^X(\beta)) \geq \mathcal{L}(\alpha)$ and $\text{card}(S) < \max(\alpha_1, \alpha_2)$, from (★) we conclude that either $\text{card}(T_1) \geq \mathcal{L}(\alpha_1)$ or $\text{card}(T_2) \geq \mathcal{L}(\alpha_2)$. Therefore, the **let** instruction at line 8 correctly defines $i \in \{1, 2\}$ such that $\text{card}(T_i) \geq \mathcal{L}(\alpha_i)$. The strategy starts (line 9) by defining T'_{3-i} (where $3-i$ is the index in $\{1, 2\}$ different from i) as a subset of $\min(\text{card}(T_{3-i}), \mathcal{L}(\alpha_{3-i}))$ elements in $\text{Cycl}[s]_{s,h}^X(\beta)$, such that $s(u) \in [T'_{3-i}]^\beta$ if and only if $s(u) \in [T_{3-i}]^\beta$. Following the same reasoning provided for T_1 in the previous case of the proof, we conclude that T'_{3-i} is well-defined and the properties (3) and (5) are satisfied for $j = 3 - i$. Moreover, similarly to (†1), we have

$$s(u) \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_{3-i} \text{ iff } s(u) \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus T_{3-i}. \quad (\ddagger_4)$$

In line 10, we consider a set Q of $\text{card}(S)$ sets in $\text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_{3-i}$ such that $s(u) \in [Q]^\beta$ if and only if $s(u) \in [S]^\beta$. Following the same reasoning provided for T_2 in the previous case of the proof, this step is well-defined. In particular, for this we rely on the fact that $\text{card}(S) < \max(\alpha_1, \alpha_2)$ whereas $\text{Cycl}[s]_{s,h}^X(\beta) \setminus T'_{3-i} \geq \mathcal{L}(\alpha) - \mathcal{L}(\alpha_{3-i}) \geq \max(\alpha_1, \alpha_2)$.

Moreover, (‡4) allows us to satisfy the constraint $s(u) \in [Q]^\flat$ if and only if $s(u) \in [S]^\flat$. Similarly to (‡2), we have

$$s(u) \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_{3-i} \cup Q) \text{ iff } s(u) \in \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T_{3-i} \cup \widehat{S}), \quad (\ddagger_5)$$

where $\widehat{S} = \{L_1 \cup L_2 \mid (L_1, L_2) \in S\}$. Before analysing the definition of S' (lines 11 and 12), we consider the definition of T'_i . We have $T'_i = \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_{3-i} \cup Q)$ (line 13). By definition of T_1 , T_2 and S , $T_i = \text{Cycl}[s]_{s,h}^X(\beta) \setminus (T_{3-i} \cup \widehat{S})$. Therefore, directly from (‡5), we conclude that the property (5) is satisfied. Now, recall that $\text{card}(Q) = \text{card}(S) < \max(\alpha_1, \alpha_2)$, $\text{card}(T'_{3-i}) \leq \mathcal{L}(\alpha_{3-i})$ and $Q \cap T'_{3-i} = \emptyset$. From (★) and $\text{card}(\text{Cycl}[s]_{s,h}^X(\beta)) \geq \mathcal{L}(\alpha)$,

$$\text{card}(\text{Cycl}[s]_{s,h}^X(\beta) \setminus (T'_{3-i} \cup Q)) \geq \mathcal{L}(\alpha) - \mathcal{L}(\alpha_{3-i}) - \max(\alpha_1, \alpha_2) \geq \mathcal{L}(\alpha_i).$$

Thus, both $\text{card}(T_i)$ and $\text{card}(T'_i)$ have at least $\mathcal{L}(\alpha_i)$ elements, which allows us to conclude that (3) is satisfied. In order to conclude the proof, we need to analyse the definition of S' and show that the properties (1), (2), (4) and (6) are satisfied. Following line 11, the strategy considers a bijection from $[Q]^\flat$ to $[S]^\flat$ satisfying the two following conditions:

- for all $L' \in Q$, $f(L') = L_1 \cup L_2$, where (L_1, L_2) is a pair in S . Notice that f induces a bijection from Q to S , mapping sets of locations to pair of sets of locations.
- if $s(u) \in [Q]^\flat$ then $f(s(u)) = s(u)$. Notice that this constraint can always be satisfied, as Q is defined so that $s(u) \in [Q]^\flat$ holds if and only if $s(u) \in [S]^\flat$. Together with the first condition, this means that the set of locations $L' \in Q$ such that $s(u) \in L'$, if it exists, it is mapped to a pair $(L_1, L_2) \in S$ such that $s(u) \in L_1 \cup L_2$.

S' is defined as $\{(L'_1, L'_2) \mid L'_1 \cup L'_2 \in Q, (f(L'_1), f(L'_2)) \in S\}$ (line 12). First of all, this means that $[S']^\flat = [Q]^\flat$, and therefore by definition of Q , T'_{3-i} and T'_i we conclude that the property (2) holds. Let us show that also (1) is satisfied. Given a pair $(L'_1, L'_2) \in S'$, by definition $L'_1 \cup L'_2 \in Q$ and so $L_1 \cup L_2 \in \text{Cycl}[s]_{s,h}^X(\beta)$. Moreover, we have $(f(L'_1), f(L'_2)) \in S$, which implies that $f(L'_1)$ and $f(L'_2)$ are disjoint and non-empty, directly by definition of S . As f is a bijection, the same holds for the two sets L'_1 and L'_2 . Thus, (1) is satisfied. Lastly, the properties (4) and (6) hold. Indeed, from the definition of S' we deduce that $f(\pi_1(S')) = \pi_1(S)$ and $f(\pi_2(S')) = \pi_2(S)$. Property (6) follows from $f(s(u)) = s(u)$. Property (4) holds as f is a bijection, and so $\text{card}([\pi_j(S')]^\flat) = \text{card}([\pi_j(S)]^\flat)$, for all $j \in \{1, 2\}$.

After showing that T'_1 , T'_2 and S' satisfy (1)–(6), we rely on these sets in order to define the heaps h'_1 and h'_2 such that $(s, h_1) \approx_{X,\alpha_1}^S (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^S (s, h'_2)$ (as required by the hop relation $\leftrightarrow_{X,\alpha}^S$). First, let us define the two heaps $\widehat{h}_1 \stackrel{\text{def}}{=} h_1 \setminus \{(\ell, \ell') \in h_1 \mid \ell \in [T_1 \cup \pi_1(S)]^\flat\}$ and $\widehat{h}_2 \stackrel{\text{def}}{=} h_2 \setminus \{(\ell, \ell') \in h_2 \mid \ell \in [T_2 \cup \pi_2(S)]^\flat\}$. By definition of T_1 , T_2 and S , $\text{dom}(\widehat{h}_1) \cap \text{Cycl}[s]_{s,h}^X(\beta)$ and $\text{dom}(\widehat{h}_2) \cap \text{Cycl}[s]_{s,h}^X(\beta)$ are both empty. Moreover, by $h = h_1 + h_2$, we have that

$$h = \widehat{h}_1 + \widehat{h}_2 + \{(\ell, \ell') \in h \mid \ell \in [\text{Cycl}[s]_{s,h}^X(\beta)]^\flat\}.$$

From the hypothesis $h \setminus \{(\ell, \ell') \in h \mid \ell \in [\text{Cycl}[s]_{s,h}^X(\beta)]^\flat\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in [\text{Cycl}[s]_{s,h}^X(\beta)]^\flat\}$ we derive $\widehat{h}_1 + \widehat{h}_2 = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in [\text{Cycl}[s]_{s,h}^X(\beta)]^\flat\}$. We define the heaps h'_1 and h'_2 as:

$$h'_1 \stackrel{\text{def}}{=} \widehat{h}_1 + \{(\ell, \ell') \in h' \mid \ell \in [T'_1 \cup \pi_1(S')]^\flat\}, \quad h'_2 \stackrel{\text{def}}{=} \widehat{h}_2 + \{(\ell, \ell') \in h' \mid \ell \in [T'_2 \cup \pi_2(S')]^\flat\}.$$

From (2), the heaps $\{(\ell, \ell') \in h' \mid \ell \in [T'_1 \cup \pi_1(S')]^\flat\}$ and $\{(\ell, \ell') \in h' \mid \ell \in [T'_2 \cup \pi_2(S')]^\flat\}$ are disjoint, and their union is $\{(\ell, \ell') \in h' \mid \ell \in [\text{Cycl}[s]_{s,h}^X(\beta)]^\flat\}$. Thus, h'_1 and h'_2 are well-defined, they are disjoint, and $h' = h'_1 + h'_2$. We now prove seven properties of h'_1 and h'_2 . Except for the the first one, i.e. (O), these properties are analogous to the properties (A)–(F) in the proof of Lemma 5.39, and so they lead to $(s, h) \leftrightarrow_{X,\alpha}^S (s, h')$ rather straightforwardly. Let $j \in \{1, 2\}$.

- O. Let $x \in X$ and $\delta \geq 0$. $h_j^\delta(s(x))$ and $h'_j{}^\delta(s(x))$ are equidefined. When defined, they are equal.

This statement is not very surprising. Intuitively, \widehat{h}_j is obtained from both h_j and h'_j by removing memory cells that correspond to unlabelled cycles of (s, h) and (s, h') , respectively. However, these memory cells cannot be reached by locations corresponding to program variables, as otherwise the cycle is not unlabelled (as shown in the formal proof below). Besides, we highlight the fact that this statement is also found in the proof of the cases (I), (IV) and (V) of Lemma 5.40. Similarly to the case under analysis, these cases treat memory cells that are not reached by locations corresponding to program variables.

Proof of (O). The proof is by induction on δ .

base case: $\delta = 0$. Straightforward.

induction step: $\delta \geq 1$. Suppose $h_j^\delta(s(x)) = \ell$. Let $\ell' = h_j^{\delta-1}(s(x))$. So, $h_j(\ell') = \ell$.

Ad absurdum, let us suppose that there is $L \in \text{Cycl}[s]_{s,h}^X(\beta)$ such that $\ell' \in L$. Since from $h_j \subseteq h$ we conclude that $h^\delta(s(x)) = \ell'$, which in turn implies that $s(x)$ reaches the cycle described by L . So, there is a location $\ell'' \in L$ that is the first location reachable from $s(x)$ that belongs to the cycle described by L . From the semantics of end-point variables, $\llbracket t \rrbracket_{s,h}^X = \ell''$. However, this is contradictory, as L does not contain labelled locations. Thus, $\ell' \notin [\text{Cycl}[s]_{s,h}^X(\beta)]^b$. By definition of \widehat{h}_j , $\ell' \in \text{dom}(h_j)$ and $\widehat{h}_j(\ell') = \ell$. Since $\widehat{h}_j \subseteq h'_j$, we conclude that $h'_j(\ell') = \ell$. By $\ell' = h_j^{\delta-1}(s(x))$ and the induction hypothesis, $\ell' = h'_j{}^{\delta-1}(s(x))$. Therefore, $h'_j{}^\delta(s(x)) = \ell$. This concludes the proof for the case where $h_j^\delta(s(x))$ is defined. The other case, i.e. when $h'_j{}^\delta(s(x))$ is defined, is symmetrical.

- A. For all $t \in T[s]^X$, $\llbracket t \rrbracket_{s,h_j}^X$ and $\llbracket t \rrbracket_{s,h'_j}^X$ are equidefined. When defined, they are equal.

Proof of (A). Follows directly from (O) and the definition of $\llbracket \cdot \rrbracket^X$. We show the case for end-point variables and leave the case of meet-point variables to the reader. Suppose $\llbracket e(x) \rrbracket_{s,h_j}^X = \ell$. By definition of end-point variable, there is $\delta \geq 1$ such that $h_j^\delta(s(x)) = \ell$ and, if $\ell \in \text{dom}(h_j)$ then ℓ belongs to a cycle in h_j whereas $h_j^{\delta-1}(s(x))$ does not. From (O), $h'_j{}^\delta(s(x)) = \ell$, $h'_j{}^{\delta-1}(s(x)) = h_j^{\delta-1}(s(x))$, and $\ell \in \text{dom}(h_j)$ if and only if $\ell \in \text{dom}(h'_j)$. Again by (O), if ℓ (resp. $h_j^{\delta-1}(s(x))$) belongs to a cycle in h_j then it belongs to a cycle in h'_j , and vice versa. We conclude that $\llbracket e(x) \rrbracket_{s,h_j}^X = \llbracket e(x) \rrbracket_{s,h'_j}^X$.

The proof that if $\llbracket e(x) \rrbracket_{s,h'_j}^X$ is defined, then $\llbracket e(x) \rrbracket_{s,h_j}^X = \llbracket e(x) \rrbracket_{s,h'_j}^X$ is symmetrical.

- B. For every $t \in T[s]^X$,

(a) $\text{sby}_{s,h_j}^X(t)$ and $\text{sby}_{s,h'_j}^X(t)$ are equidefined. When defined, they are equal,

(b) $\text{Path}[s]_{s,h_j}^X(t) = \text{Path}[s]_{s,h'_j}^X(t)$,

(c) let $\delta \in [1, \text{card}(\text{Path}[s]_{s,h_j}^X(t))]$. $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) = s(u)$ iff $h'_j{}^\delta(\llbracket t \rrbracket_{s,h'_j}^X) = s(u)$.

Proof of (B). As in the case of (A), the statements (a)–(c) follow directly from (O) and the definition of $\llbracket \cdot \rrbracket^X$. In particular, given $t \in T[s]^X$ and $\delta \geq 0$, by (O), we conclude that, when defined, $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) = h'_j{}^\delta(\llbracket t \rrbracket_{s,h'_j}^X)$. This implies (c). Moreover, by (A), $\text{Lab}[s]_{s,h_j}^X = \text{Lab}[s]_{s,h'_j}^X$, which allows us to conclude that (a) and (b) hold.

- C. For every $x \in X$, $\text{Pred}[s]_{s,h_j}^X(x) = \text{Pred}[s]_{s,h'_j}^X(x)$.

Proof of (C). (\subseteq): Let $\ell \in \text{Pred}[\mathcal{S}]_{s,h_j}^X(x)$, and so $h_j(\ell) = s(x)$ and, in h_j , ℓ is not reached by any location corresponding to program variables in X . As $h_j \subseteq h$ we have $h(\ell) = s(x)$. *Ad absurdum*, suppose there is $L \in \text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)$ such that $\ell \in L$. As L describes a cycle, we have $s(x) = h(\ell) \in L$. However, this is contradictory, as L does not contain labelled locations. Thus, $\ell \notin [\text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)]^\flat$. By definition of \widehat{h}_j together with the fact that $\ell \in \text{dom}(h_j)$, we conclude that $\ell \in \text{dom}(\widehat{h}_j)$ and $\widehat{h}_j(\ell) = s(x)$. Since $\widehat{h}_j \subseteq h'_j$, $h'_j(\ell) = s(x)$. In order to conclude that $\ell \in \text{Pred}[\mathcal{S}]_{s,h'_j}^X(x)$, it is sufficient to show that, in h'_j , ℓ is not reached by any location corresponding to program variables in X . *Ad absurdum*, suppose there is $y \in X$ and $\delta \geq 0$ such that $h'^\delta(s(y)) = \ell$. Since $\ell \in \text{dom}(h'_j)$, there must be a term $t \in T[\mathcal{S}]^X$ such that $\ell \in \text{Path}[\mathcal{S}]_{s,h'_j}^X(t)$. However, by (b) this implies that $\ell \in \text{Path}[\mathcal{S}]_{s,h_j}^X(t)$. By Proposition 5.31, this contradicts the fact that $\ell \in \text{Pred}[\mathcal{S}]_{s,h_j}^X(x)$. Thus, ℓ is not reached by any location corresponding to program variables in X , which allows us to conclude that $\ell \in \text{Pred}[\mathcal{S}]_{s,h'_j}^X(x)$.

(\supseteq): Symmetrical to the other direction.

D. For every $\beta' \in [1, \alpha_j]$,

- (d) $\min(\text{card}(\text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta')), \mathcal{L}(\alpha_j)) = \min(\text{card}(\text{Cycl}[\mathcal{S}]_{s,h'_j}^X(\beta')), \mathcal{L}(\alpha_j))$,
- (e) $s(u) \in [\text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta')]^\flat$ if and only if $s(u) \in [\text{Cycl}[\mathcal{S}]_{s,h'_j}^X(\beta')]^\flat$.

We start by showing the following equivalence and inclusions:

- (f) $\text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta') \setminus T_j = \text{Cycl}[\mathcal{S}]_{s,h'_j}^X(\beta') \setminus T'_j$,
- (g) if $\beta' = \beta$ then $T_j \subseteq \text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta')$ and $T'_j \subseteq \text{Cycl}[\mathcal{S}]_{s,h'_j}^X(\beta')$.

Proof of (f). (\subseteq): Suppose $L \in \text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta') \setminus T_j$. By definition of $\text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta')$:

- h. L does not contain locations from $\text{Lab}[\mathcal{S}]_{s,h_j}^X$,
- i. L describes a cycle in h_j .

From (i), it cannot be that $L \in \pi_j(S)$. *Ad absurdum*, suppose $L \in \pi_j(S)$. This means that there is a non-empty L' such that $L \cup L' \in \text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)$ and $L \cap L' = \emptyset$. By definition, $L \cup L'$ is a set of β locations describing a cycle (of length β) in h . From $L' \neq \emptyset$, we conclude that L cannot describe a cycle in h . However, directly from $h_j \subseteq h$, this implies that it cannot describe a cycle in h_j , in contradiction with (h). Thus, $L \notin \pi_j(S)$. By definition of \widehat{h}_j , we conclude that $L \subseteq \text{dom}(\widehat{h}_j)$. By $\widehat{h}_j \subseteq h_j$, L describes a cycle in \widehat{h}_j . By $\widehat{h}_j \subseteq h'_j$, L describes a cycle in h'_j . Moreover, by definition of \widehat{h}_j , $L \notin T'_j$. Lastly, from (A) and (h), L does not contain locations from $\text{Lab}[\mathcal{S}]_{s,h'_j}^X$. So, $L \in \text{Cycl}[\mathcal{S}]_{s,h'_j}^X(\beta') \setminus T'_j$.

(\supseteq): Symmetrical to the other direction.

Proof of (g). We prove the inclusion $T_j \subseteq \text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta)$. Suppose $L \in T_j$. By definition of T_j , $L \in \text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)$, which means that:

- * L describes a cycle in T ,
- * L does not contain locations from $\text{Lab}[\mathcal{S}]_{s,h}^X$. Equivalently, for every $\ell \in L$, $x \in X$ and $\delta \geq 0$, $h^\delta(s(x)) \neq \ell$.

As $h_j \subseteq h$ and $L \in \text{dom}(h_j)$, these two properties carry over to h_j (they are monotonous under subheaps). By definition, $L \in \text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta)$.

The proof of the inclusion $T'_j \subseteq \text{Cycl}[\mathcal{S}]_{s,h'_j}^X(\beta)$ is analogous.

Proof of (D). First, if $\beta' \neq \beta$ then both $T_j \cap \text{Cycl}[s]_{s,h_j}^X(\beta')$ and $T'_j \cap \text{Cycl}[s]_{s,h_j}^X(\beta')$ are empty. Indeed, T_j and T'_j contain sets of cardinality β , whereas $\text{Cycl}[s]_{s,h_j}^X(\beta')$ and $\text{Cycl}[s]_{s,h_j}^X(\beta')$ contain sets of cardinality β' . Thus, both (d) and (e) hold by (f).

Let us assume $\beta' = \beta$. We start by proving that (d) holds. The property (3) of the construction states that $\min(\text{card}(T_j), \mathcal{L}(\alpha_j)) = \min(\text{card}(T'_j), \mathcal{L}(\alpha_j))$. Since for all $a, b, c, d \in \mathbb{N}$, $\min(a, d) = \min(b, d)$ implies $\min(a + c, d) = \min(b + c, d)$,

$$\begin{aligned} & \min(\text{card}(T_j) + \text{card}(\text{Cycl}[s]_{s,h_j}^X(\beta) \setminus T_j), \mathcal{L}(\alpha_j)) \\ &= \min(\text{card}(T'_j) + \text{card}(\text{Cycl}[s]_{s,h_j}^X(\beta) \setminus T_j), \mathcal{L}(\alpha_j)). \end{aligned}$$

From (f), $\text{card}(\text{Cycl}[s]_{s,h_j}^X(\beta) \setminus T_j) = \text{card}(\text{Cycl}[s]_{s,h'_j}^X(\beta) \setminus T'_j)$, and so

$$\begin{aligned} & \min(\text{card}(T_j) + \text{card}(\text{Cycl}[s]_{s,h_j}^X(\beta) \setminus T_j), \mathcal{L}(\alpha_j)) \\ &= \min(\text{card}(T'_j) + \text{card}(\text{Cycl}[s]_{s,h'_j}^X(\beta) \setminus T'_j), \mathcal{L}(\alpha_j)). \end{aligned} \tag{\dagger}$$

By (g), we have $\text{Cycl}[s]_{s,h_j}^X(\beta) = (\text{Cycl}[s]_{s,h_j}^X(\beta) \setminus T_j) \cup T_j$ and so

$$\text{card}(\text{Cycl}[s]_{s,h_j}^X(\beta)) = \text{card}(\text{Cycl}[s]_{s,h_j}^X(\beta) \setminus T_j) + \text{card}(T_j).$$

Similarly (again from (g)),

$$\text{card}(\text{Cycl}[s]_{s,h'_j}^X(\beta)) = \text{card}(\text{Cycl}[s]_{s,h'_j}^X(\beta) \setminus T'_j) + \text{card}(T'_j).$$

Therefore, from (†), we conclude that (d) holds.

Let us prove (e). For the left-to-right direction, suppose $s(u) \in [\text{Cycl}[s]_{s,h_j}^X(\beta)]^\flat$.

By (g), we have either $s(u) \in [\text{Cycl}[s]_{s,h_j}^X(\beta) \setminus T_j]^\flat$ or $s(u) \in [T_j]^\flat$. In the former case, directly from (f), we conclude that $s(u) \in [\text{Cycl}[s]_{s,h'_j}^X(\beta)]^\flat$. In the latter case, from the property (5) of the construction, we have $s(u) \in [T'_j]^\flat$. By (g), $s(u) \in [\text{Cycl}[s]_{s,h'_j}^X(\beta)]^\flat$.

The right-to-left direction is proved symmetrically.

E. (j) $\min(\text{card}(\uparrow\text{Cycl}[s]_{s,h_j}^{X,\alpha_j}), \mathcal{L}(\alpha_j)) = \min(\text{card}(\uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}), \mathcal{L}(\alpha_j))$,

(k) $s(u) \in [\uparrow\text{Cycl}[s]_{s,h_j}^{X,\alpha_j}]^\flat$ if and only if $s(u) \in [\uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}]^\flat$.

The proofs of these two statements rely on the following equality and inclusions:

(l) $\uparrow\text{Cycl}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j = \uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j} \setminus T'_j$,

(m) if $\beta > \alpha_j$ then $T_j \subseteq \uparrow\text{Cycl}[s]_{s,h_j}^{X,\alpha_j}$ and $T'_j \subseteq \uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}$.

The statement (l) is proved similarly to (f), whereas the statement (m) is proved analogously to (g). Then, the proof of (E) follows similarly to (D).

F. (n) $\min(\text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j}), \mathcal{R}(\alpha_j)) = \min(\text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j}), \mathcal{R}(\alpha_j))$,

(o) $s(u) \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ if and only if $s(u) \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

We start by showing the following equality and inclusions:

(p) $\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus [\pi_j(S)]^\flat = \text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus [\pi_j(S')]^\flat$,

(q) $[\pi_j(S)]^\flat \subseteq \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ and $[\pi_j(S')]^\flat \subseteq \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

Proof of (p). (\subseteq): Let $\ell \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus [\pi_j(S)]^\flat$. As $\ell \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$, we have

r. $\ell \in \text{dom}(h_j)$

s. ℓ does not belong to $\text{Lab}[s]_{s,h_j}^X$, $\text{Pred}[s]_{s,h_j}^X(x)$, $\text{Path}[s]_{s,h_j}^X(\ell)$, $\text{Cycl}[s]_{s,h_j}^X(\beta')$ and $\uparrow\text{Cycl}[s]_{s,h_j}^{X,\alpha_j}$, for every $x \in X$, $\ell \in \text{Lab}[s]_{s,h_j}^X$ and $\beta' \in [1, \alpha_j]$.

Ad absurdum, suppose $\ell \in [T_j]^\beta$. Then, there is $L \in T_j$ such that $\ell \in L$. By definition of T_j , L describes a cycle in h . By definition of h_j , we have $L \subseteq \text{dom}(h_j)$, which in turn implies that L describes a cycle in h . However, this implies that there is $\delta \geq 0$ such that $h_j(\ell) = \ell$, in contradiction with $\ell \in \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$. Hence, $\ell \notin [T_j]^\beta$. Together with $\ell \notin [\pi_j(S)]^\beta$ and (r), this means that $\ell \in \text{dom}(h_j)$. By definition of h'_j , we conclude that $\ell \in \text{dom}(h'_j)$, $\ell \notin [\pi_j(S')]^\beta$ and $\ell \notin [T'_j]^\beta$. From (s), this implies that ℓ does not belong to $\text{Lab}[S]_{s,h'_j}^X$ (by (A)), $\text{Pred}[S]_{s,h'_j}^X(x)$ (by (C)), $\text{Path}[S]_{s,h'_j}^X(\ell)$ (by (A) and (b)), $\text{Cycl}[S]_{s,h'_j}^X(\beta')$ (by (f)) and $\text{↑Cycl}[S]_{s,h'_j}^{X,\alpha_j}$ (by (l)), for all $x \in X$, $\ell \in \text{Lab}[S]_{s,h'_j}^X$ and $\beta' \in [1, \alpha_j]$. Therefore, $\ell \in \text{Rem}[S]_{s,h'_j}^{X,\alpha_j} \setminus [\pi_j(S')]^\beta$.

(\supseteq): Symmetrical to the other direction.

Proof of (q). We prove the inclusion $[\pi_j(S)]^\beta \subseteq \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$. Let $\ell \in [\pi_j(S)]^\beta$, and so $\ell \in \text{dom}(h_j)$. Consider the set of locations $L_1 \in \pi_j(S)$ such that $\ell \in L$. By definition of S , ℓ belongs to a set $L \in \text{Cycl}[S]_{s,h}^X(\beta)$ such that

- t. L contains β locations and describes a cycle in h (of length β),
- u. L does not contain locations in $\text{Lab}[S]_{s,h}^X$. Equivalently, for every $\ell' \in L$, $x \in X$ and $\delta \geq 0$, $h_j^\delta(s(x)) \neq \ell'$.
- v. there is a non-empty set of locations $L_2 \in \pi_{3-j}(S)$ such that $L_1 \cup L_2 = L$.

By (t), (v) and $h_j \subseteq h$ we conclude that L_1 does not describe a cycle in h_j . Thus, $\ell \notin \text{Cycl}[S]_{s,h_j}^X(\beta')$ (for every $\beta' \in [1, \alpha_j]$) and $\ell \notin \text{↑Cycl}[S]_{s,h_j}^{X,\beta'}$. From (u) and $h_j \subseteq h$, for every $\ell \in L$, $x \in X$ and $\delta \geq 0$, $h_j^\delta(s(x)) \neq \ell$. This implies that $\ell \notin \text{Lab}[S]_{s,h_j}^X$ and $\ell \notin \text{Path}[S]_{s,h_j}^X(\ell')$, for every $\ell' \in \text{Lab}[S]_{s,h_j}^X$. Lastly, as L describes a cycle and $\ell \in L$, we derive $h_j(\ell) \in L$, which in turn implies that $h_j(\ell)$ does not correspond to a variable. So, $\ell \notin \text{Pred}[S]_{s,h_j}^X(x)$, for all $x \in X$. By definition of $\text{Rem}[S]_{s,h_j}^{X,\alpha_j}$, $\ell \in \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$.

The proof of the inclusion $[\pi_j(S')]^\beta \subseteq \text{Rem}[S]_{s,h'_j}^{X,\alpha_j}$ is analogous.

Proof of (F). We start by proving (n). The property (4) of the construction states that $\min(\text{card}([\pi_j(S)]^\beta), \mathcal{R}(\alpha_j)) = \min(\text{card}([\pi_j(S')]^\beta), \mathcal{R}(\alpha_j))$. Since for all $a, b, c, d \in \mathbb{N}$, $\min(a, d) = \min(b, d)$ implies $\min(a + c, d) = \min(b + c, d)$,

$$\begin{aligned} & \min(\text{card}([\pi_j(S)]^\beta) + \text{card}(\text{Rem}[S]_{s,h_j}^{X,\alpha_j} \setminus [\pi_j(S)]^\beta), \mathcal{R}(\alpha_j)) \\ &= \min(\text{card}([\pi_j(S')]^\beta) + \text{card}(\text{Rem}[S]_{s,h'_j}^{X,\alpha_j} \setminus [\pi_j(S')]^\beta), \mathcal{R}(\alpha_j)). \end{aligned}$$

From (p), $\text{card}(\text{Rem}[S]_{s,h_j}^{X,\alpha_j} \setminus T_j) = \text{card}(\text{Rem}[S]_{s,h'_j}^{X,\alpha_j} \setminus T'_j)$, and so

$$\begin{aligned} & \min(\text{card}([\pi_j(S)]^\beta) + \text{card}(\text{Rem}[S]_{s,h_j}^{X,\alpha_j} \setminus [\pi_j(S)]^\beta), \mathcal{R}(\alpha_j)) \\ &= \min(\text{card}([\pi_j(S')]^\beta) + \text{card}(\text{Rem}[S]_{s,h'_j}^{X,\alpha_j} \setminus [\pi_j(S')]^\beta), \mathcal{R}(\alpha_j)). \end{aligned} \tag{\dagger}$$

By (q), we have $\text{Rem}[S]_{s,h_j}^{X,\alpha_j} = (\text{Rem}[S]_{s,h_j}^{X,\alpha_j} \setminus [\pi_j(S)]^\beta) \cup [\pi_j(S)]^\beta$ and so

$$\text{card}(\text{Rem}[S]_{s,h_j}^{X,\alpha_j}) = \text{card}(\text{Rem}[S]_{s,h_j}^{X,\alpha_j} \setminus [\pi_j(S)]^\beta) + \text{card}([\pi_j(S)]^\beta).$$

Similarly (again from (q)),

$$\text{card}(\text{Rem}[S]_{s,h'_j}^{X,\alpha_j}) = \text{card}(\text{Rem}[S]_{s,h'_j}^{X,\alpha_j} \setminus [\pi_j(S')]^\beta) + \text{card}([\pi_j(S')]^\beta).$$

Therefore, from (†), we conclude that (n) holds.

Let us prove (o). For the left-to-right direction, suppose $s(u) \in \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$. By (q), we have either $s(u) \in \text{Rem}[S]_{s,h_j}^{X,\alpha_j} \setminus [\pi_j(S)]^\beta$ or $s(u) \in [\pi_j(S)]^\beta$. In the former case,

directly from (p), we conclude that $s(u) \in \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$. In the latter case, from the property (6) of the construction, we have $s(u) \in [\pi_j(S')]^b$. By (q), $s(u) \in \text{Rem}[S]_{s,h'_j}^{X,\alpha_j}$. The right-to-left direction is proved symmetrically.

The properties (A)–(F) lead directly to $(s, h_j) \approx_{X,\alpha_j}^S (s', h'_j)$, with the same case analysis provided at the end of the proof of Lemma 5.39. Therefore, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h')$. \square

We now move to the proof of Lemma 5.40(II), which deals with the $*$ -simulation in case of two memory states (s, h) and (s, h') whose differences are localised to the sets $\text{Path}[S]_{s,h}^X(t)$ and $\text{Path}[S]_{s,h'}^X(t)$. The proof follows the same steps of Lemma 5.40(III). First, we partition $\text{Path}[S]_{s,h}^X(t)$ in multiple sets, following the definition of h_1 and h_2 . We identify analogous sets that partition $\text{Path}[S]_{s,h'}^X(t)$, which are then used to define h'_1 and h'_2 . Lastly, discuss several properties of these two heaps, which allows us to establish $(s, h_1) \approx_{X,\alpha_1}^S (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^S (s, h'_2)$. Despite the simple steps, the technical developments needed to correctly show this result are quite involved.

Proof of (II). We assume α to be at least two, otherwise the lemma trivially holds. Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Moreover, let $t \in T[S]^X$ be a term such that $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X = \tilde{\ell}$. The first equality holds from the hypothesis,

$$\text{for every } t \in T[S]^X, \quad \llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X, \quad (=t)$$

which will be used throughout the proof. In what follows, we write $S(\alpha)$ and $R(\alpha)$ for the upper bounds given to β in core formulae of the form $\text{sees}_X(t, t') \geq \beta$ and $\text{rem}_{X,\alpha}^S \geq \beta$, respectively. Similarly, we write $S_{\text{left}}(\alpha)$ (resp. $S_{\text{right}}(\alpha)$) for the upper bound given to $\overleftarrow{\beta}$ (resp. $\overrightarrow{\beta}$) in formulae of the form $u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$. More precisely, we have:

$$\begin{aligned} S(\alpha) &= \frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3), & R(\alpha) &= \alpha, \\ S_{\text{left}}(\alpha) &= \frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1, & S_{\text{right}}(\alpha) &= \frac{1}{2}\alpha(\alpha+3). \end{aligned}$$

One can show that the following (in)equalities hold:

- (\star_1) $S(\alpha) = S_{\text{left}}(\alpha) + S_{\text{right}}(\alpha)$,
- (\star_2) $S_{\text{left}}(\alpha) \geq S(\max(\alpha_1, \alpha_2)) + 1$,
- (\star_3) $S_{\text{right}}(\alpha) \geq S_{\text{right}}(\max(\alpha_1, \alpha_2)) + R(\alpha_1) + R(\alpha_2) + 1$.

Similarly to Lemma 5.40(III), if $\text{card}(\text{Path}[S]_{s,h}^X(t)) < S(\alpha)$ then the lemma holds by Lemma 5.39. Indeed, in this case, the equisatisfaction of the formulae of the form $\text{sees}_X(t, t') \geq \beta$, together with the hypothesis that for every term $t \in T[S]^X$, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$, allows us to conclude that $\text{card}(\text{Path}[S]_{s,h}^X(t)) = \text{card}(\text{Path}[S]_{s,h'}^X(t))$. By definition of h and h' we conclude that $((s, h), (s, h')) \in (\bigcap_{\alpha' \geq 1} \approx_{X,\alpha'}^S)$, which allows us to apply Lemma 5.39. Therefore, in the following we assume $\text{card}(\text{Path}[S]_{s,h}^X(t)) \geq S(\alpha)$, which implies $\text{card}(\text{Path}[S]_{s,h'}^X(t)) \geq S(\alpha)$ again from the equisatisfaction of the core formulae $\text{sees}_X(t, t') \geq \beta$.

Assumption. Both $\text{Path}[S]_{s,h}^X(t)$ and $\text{Path}[S]_{s,h'}^X(t)$ have at least $S(\alpha)$ elements.

We remind the reader that whenever $\text{Path}[S]_{s,h}^X(t)$ is non-empty, it describes a path in h , going from $\llbracket t \rrbracket_{s,h}^X$ to $\text{sby}_{s,h}^X(t)$. We define the following subsets and locations of $\text{Path}[S]_{s,h}^X(t)$:

- l_{pre} is the only location in $\text{Path}[S]_{s,h}^X(t)$ such that $h(l_{\text{pre}}) = \text{sby}_{s,h}^X(t)$,
- given $j \in \{1, 2\}$, $P_j \stackrel{\text{def}}{=} \{\ell \in \text{Path}[S]_{s,h}^X(t) \cap \text{dom}(h_j) \mid l_{\text{pre}} \neq \ell = h_j^\delta(\llbracket t \rrbracket_{s,h}^X), \text{ for some } \delta \geq 0\}$,
- given $j \in \{1, 2\}$, $R_j \stackrel{\text{def}}{=} (\text{sby}_{s,h}^X(t) \cap \text{dom}(h_j)) \setminus (P_j \cup \{l_{\text{pre}}\})$.

We notice that l_{pre} is the predecessor of $\text{sby}_{s,h}^X(t)$ in the path described by $\text{Path}[s]_{s,h}^X(t)$. Since the definition of the relation $\leftrightarrow_{X,\alpha}^S$ is symmetrical with respect to the subheaps h_1 and h_2 , without loss of generality we assume $l_{pre} \in \text{dom}(h_1)$.

Assumption. $l_{pre} \in \text{dom}(h_1)$.

The *crux* of the proof is dealing with how this path is split between the heaps h_1 and h_2 , while understanding what properties the locations in $\text{Path}[s]_{s,h}^X(t)$ have on these subheaps. The two sets P_1 and P_2 are introduced with this in mind. In particular, P_j contains all the locations in $\text{Path}[s]_{s,h}^X(t) \cap \text{dom}(h_j)$ that, in h_j , are reachable from $\llbracket t \rrbracket_{s,h}^X$, with the exception of l_{pre} . Given $j \in \{1, 2\}$, this means that $\llbracket t \rrbracket_{s,h}^X \in P_j$ if and only if $\llbracket t \rrbracket_{s,h}^X \in \text{dom}(h_j)$, and thus exactly one set among P_1 and P_2 is empty. Whenever non-empty, P_j is a minimal set describing a path in both h and h_j , going from $\llbracket t \rrbracket_{s,h}^X$ to a location that is not in P_j . We write e_{P_j} to denote this location. It could be that $e_{P_j} = l_{pre}$. In this case, following the assumption $l_{pre} \in \text{dom}(h_1)$, if $j = 1$ then $\text{Path}[s]_{s,h}^X(t) \subseteq \text{dom}(h_1)$, and therefore $\text{Path}[s]_{s,h}^X(t)$ describes a path in h_1 , going from $\llbracket t \rrbracket_{s,h}^X$ to $\text{sby}_{s,h}^X(t)$. Otherwise, if $e_{P_j} \neq l_{pre}$, then $e_{P_j} \in \text{dom}(h_{3-j})$ and, from the definition of the sets above, we conclude that $e_{P_j} \in R_{3-j}$. Together, the two sets R_1 and R_2 contain all the locations of $\text{Path}[s]_{s,h}^X(t)$ that are not l_{pre} nor they belong to P_1 or P_2 . Among these locations, R_1 contains the ones in $\text{dom}(h_1)$, whereas R_2 contains the ones in $\text{dom}(h_2)$. Fundamentally, the sets P_1 , P_2 , R_1 , R_2 and $\{l_{pre}\}$ are mutually disjoint, with their union being $\text{Path}[s]_{s,h}^X(t)$.

Now, we aim at defining similar sets P'_1 , P'_2 , R'_1 and R'_2 , with respect to $\text{Path}[s]_{s,h'}^X(t)$. First of all, we let l'_{pre} be the only location in $\text{Path}[s]_{s,h'}^X(t)$ such that $h'(l'_{pre}) = \text{sby}_{s,h'}^X(t)$. These sets shall satisfy the following eight constraints:

1. P'_1 , P'_2 , R'_1 , R'_2 and $\{l'_{pre}\}$ are mutually disjoint. Their union is $\text{Path}[s]_{s,h'}^X(t)$,
2. $P'_1 = \emptyset$ if and only if $P'_2 \neq \emptyset$,

Moreover, for every $j \in \{1, 2\}$,

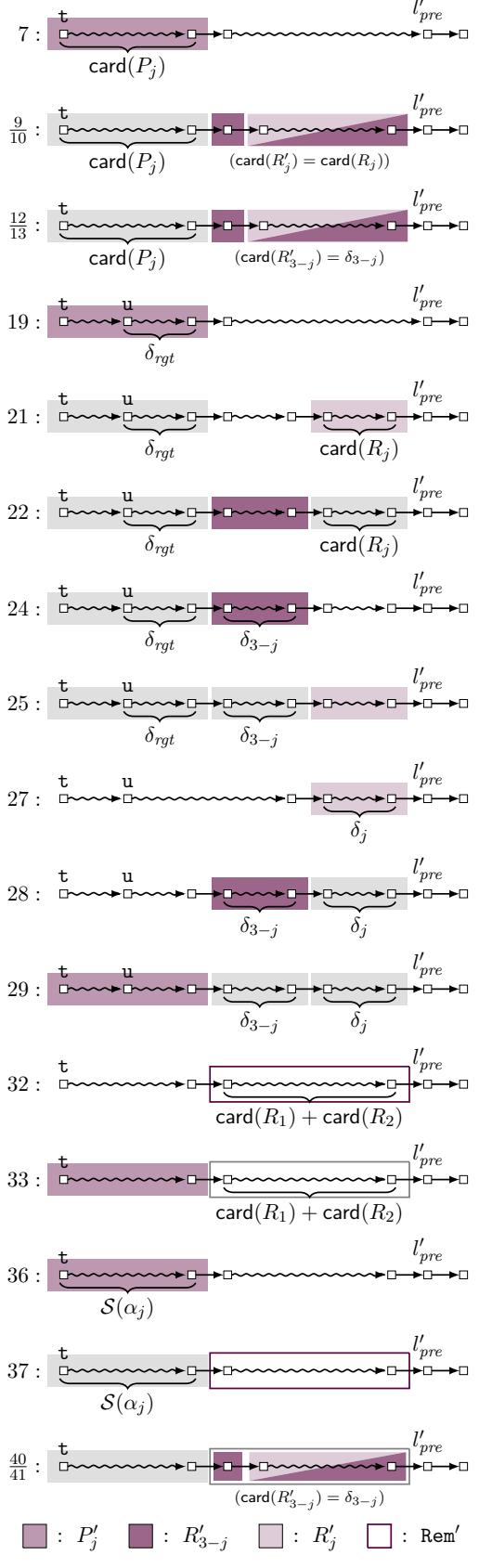
3. if non-empty, P'_j is a minimal set describing a path in h' , going from $\llbracket t \rrbracket_{s,h}^X$ to a location in $R'_{3-j} \cup \{l'_{pre}\}$, which is denoted by $e_{P'_j}$,
4. $\min(\text{card}(P_j), \mathcal{S}(\alpha_j)) = \min(\text{card}(P'_j), \mathcal{S}(\alpha_j))$,
5. $\min(\text{card}(R_j), \mathcal{R}(\alpha_j)) = \min(\text{card}(R'_j), \mathcal{R}(\alpha_j))$,
6. $s(u) \in R_j$ if and only if $s(u) \in R_j$,
7. if there is $\delta_1 \in [0, \text{card}(P_j)]$ such that $h^{\delta_1}(\llbracket t \rrbracket_{s,h}^X) = s(u)$, then there is $\delta_2 \in [0, \text{card}(P'_j)]$ such that $h'^{\delta_2}(\llbracket t \rrbracket_{s,h'}^X) = s(u)$, $\min(\text{card}(P_j) - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(P'_j) - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j))$ and $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j))$.
8. if there is $\delta_2 \in [0, \text{card}(P'_j)]$ such that $h'^{\delta_2}(\llbracket t \rrbracket_{s,h'}^X) = s(u)$, then there is $\delta_1 \in [0, \text{card}(P_j)]$ such that $h^{\delta_1}(\llbracket t \rrbracket_{s,h}^X) = s(u)$, $\min(\text{card}(P_j) - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(P'_j) - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j))$ and $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j))$.

The definition of P'_1 , P'_2 , R'_1 and R'_2 follows the strategy in Figure 5.15. We remind the reader that both the sets $\text{Path}[s]_{s,h}^X(t)$ and $\text{Path}[s]_{s,h'}^X(t)$ contain at least $\mathcal{S}(\alpha)$ elements. The right hand side of Figure 5.15 schematically highlights the sets that are defined in the corresponding lines of the strategy. We now analyse the strategy, and prove that it leads to sets that satisfy the properties (1)–(8). For simplicity, let us assume that $\text{Path}[s]_{s,h}^X(t)$ and $\text{Path}[s]_{s,h'}^X(t)$ are the minimal sets that describe the paths $\rho = (\ell_0, \dots, \ell_n)$ and $\rho' = (\ell'_0, \dots, \ell'_m)$, respectively. Following the terminology introduced in Definition 5.30, this means that $\text{Path}[s]_{s,h}^X(t) = \{\ell_0, \dots, \ell_{n-1}\}$

```

1: Path'  $\leftarrow$  Path $[s]_{s,h'}^X(t)$ .
2: let  $j \in \{1, 2\}$  s.t.  $P_j \neq \emptyset$ .
3:  $P'_{3-j} \leftarrow \emptyset$ .
4:  $\delta_j \leftarrow \min(\text{card}(R_j), \mathcal{R}(\alpha_j))$ .
5:  $\delta_{3-j} \leftarrow \min(\text{card}(R_{3-j}), \mathcal{R}(\alpha_{3-j}))$ .
6: if  $\text{card}(P_j) < \mathcal{S}(\alpha_j)$  then
7:    $P'_j \leftarrow \{h'^{\delta}([\![t]\!]_{s,h}^X) \mid \delta \in [0, \text{card}(P_j) - 1]\}$ .
8:   if  $\text{card}(R_j) < \mathcal{R}(\alpha_j)$  then
9:     let  $R'_j \subseteq \text{Path}' \setminus (P'_j \cup \{l'_{pre}\})$  such that
      •  $\text{card}(R'_j) = \text{card}(R_j)$ ,
      •  $h'^{\text{card}(P_j)}([\![t]\!]_{s,h'}^X) \notin R'_j$ ,
      •  $s(u) \in R'_j$  iff  $s(u) \in R_j$ .
10:     $R'_{3-j} \leftarrow \text{Path}' \setminus (P_j \cup R'_j \cup \{l'_{pre}\})$ .
11: else
12:   let  $R'_{3-j} \subseteq \text{Path}' \setminus (P'_j \cup \{l'_{pre}\})$  such that
      •  $\text{card}(R'_{3-j}) = \delta_{3-j}$ ,
      •  $h'^{\text{card}(P_j)}([\![t]\!]_{s,h'}^X) \in R'_{3-j}$ ,
      •  $s(u) \in R'_{3-j}$  iff  $s(u) \in R_{3-j}$ .
13:    $R'_j \leftarrow \text{Path}' \setminus (P_j \cup R'_{3-j} \cup \{l'_{pre}\})$ .
14: else if  $s(u) \in P_j$  or  $s(u) = e_{P_j}$  then
15:   let  $\delta_{rgt} \in [0, \text{card}(P_j)]$  s.t.  $h^{(\text{card}(P_j) - \delta_{rgt})}([\![t]\!]_{s,h}^X) = s(u)$ .
16:   let  $\delta'_{lt} \in [0, \text{card}(\text{Path}')]$  s.t.  $h'^{\delta'_{lt}}([\![t]\!]_{s,h'}^X) = s(u)$ .
17:   if  $\delta_{rgt} < \mathcal{S}_{\text{right}}(\alpha_j)$  then
18:      $\bar{\delta} \leftarrow \delta'_{lt} + \delta_{rgt}$ .
19:      $P'_j \leftarrow \{h'^{\delta}([\![t]\!]_{s,h}^X) \mid \delta \in [0, \bar{\delta} - 1]\}$ .
20:     if  $\text{card}(R_j) < \mathcal{R}(\alpha_j)$  then
21:        $R'_j \leftarrow \{\ell \in \text{Path}' \mid \exists \delta \in [1, \text{card}(R_j)], h'^{\delta}(\ell) = l'_{pre}\}$ .
22:        $R'_{3-j} \leftarrow \text{Path}' \setminus (P_j \cup R'_j \cup \{l'_{pre}\})$ .
23:     else
24:        $R'_{3-j} \leftarrow \{h'^{(\bar{\delta} + \delta)}([\![t]\!]_{s,h}^X) \mid \delta \in [0, \delta_{3-j} - 1]\}$ .
25:        $R'_j \leftarrow \text{Path}' \setminus (P_j \cup R'_{3-j} \cup \{l'_{pre}\})$ .
26:   else
27:      $R'_j \leftarrow \{\ell \in \text{Path}' \mid \exists \delta \in [1, \delta_j] \text{ s.t. } h'^{\delta}(\ell) = l'_{pre}\}$ .
28:      $R'_{3-j} \leftarrow \left\{ \ell \in \text{Path}' \mid \begin{array}{l} h'^{(\delta_j + \delta)}(\ell) = l'_{pre}, \\ \text{for some } \delta \in [1, \delta_{3-j}] \end{array} \right\}$ .
29:      $P'_j \leftarrow \text{Path}' \setminus (R'_j \cup R'_{3-j} \cup \{l'_{pre}\})$ .
30: else
31:   if  $\text{card}(R_1) < \mathcal{R}(\alpha_1)$  and  $\text{card}(R_2) < \mathcal{R}(\alpha_2)$  then
32:      $\text{Rem}' \leftarrow \left\{ \ell \in \text{Path}' \mid \begin{array}{l} h'^{\delta}(\ell) = l'_{pre}, \text{ for some} \\ \delta \in [1, \text{card}(R_1) + \text{card}(R_2)] \end{array} \right\}$ .
33:      $P'_j \leftarrow \text{Path}' \setminus (\text{Rem}' \cup \{l'_{pre}\})$ .
34:   let  $k \in \{1, 2\}$ .
35: else
36:    $P'_j \leftarrow \{h'^{\delta}([\![t]\!]_{s,h'}^X) \mid \delta \in [0, \mathcal{S}(\alpha_j) - 1]\}$ .
37:    $\text{Rem}' \leftarrow \text{Path}' \setminus (P'_j \cup \{l'_{pre}\})$ .
38:   let  $k \in \{1, 2\}$  such that  $\text{card}(R_k) \geq \mathcal{R}(\alpha_k)$ .
39:   let  $e_{P'_j} \in \text{Rem}'$  be such that there is  $\ell \in P'_j$ ,  $h'(\ell) = e_{P'_j}$ .
40:   let  $R'_{3-k} \subseteq \text{Rem}'$  s.t. •  $\text{card}(R'_{3-k}) = \delta_{3-k}$ ,
      •  $e_{P'_j} \in R_{3-k}$  iff  $k \neq j$ ,
      •  $s(u) \in R'_{3-k}$  iff  $s(u) \in R_{3-k}$ .
41:    $R'_k \leftarrow \text{Rem}' \setminus R'_{3-k}$ .

```

Figure 5.15: Strategy to define P'_1 , P'_2 , R'_1 and R'_2 .

and $\text{Path}[\mathcal{S}]_{s,h'}^X(\mathbf{t}) = \{\ell'_0, \dots, \ell'_{m-1}\}$, where the locations $\ell_0, \dots, \ell_{n-1}$ (resp. $\ell'_0, \dots, \ell'_{m-1}$) are all distinct, and by definition of these two sets we have

$$\begin{aligned} \ell_0 &= \llbracket \mathbf{t} \rrbracket_{s,h}^X, & \ell_{n-1} &= l_{\text{pre}}, & \ell_n &= \text{sby}_{s,h}^X(\mathbf{t}), & n &= \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t})), \\ \ell'_0 &= \llbracket \mathbf{t} \rrbracket_{s,h'}^X, & \ell'_{m-1} &= l'_{\text{pre}}, & \ell'_m &= \text{sby}_{s,h'}^X(\mathbf{t}), & m &= \text{card}(\text{Path}[\mathcal{S}]_{s,h'}^X(\mathbf{t})). \end{aligned}$$

Let $j \in \{1, 2\}$ such that $P_j \neq \emptyset$ (see line 2 of the strategy). Notice that this implies $P_{3-j} = \emptyset$ and, as in line 3, $P'_{3-j} \stackrel{\text{def}}{=} \emptyset$, as required by the property (4). P_j is a minimal set describing a non-empty path in h , going from $\llbracket \mathbf{t} \rrbracket_{s,h}^X$ to e_{P_j} . Let $k = \text{card}(P_j)$. We have $P_j = \{\ell_0, \dots, \ell_{k-1}\}$ and $\ell_k = e_{P_j}$. Since P_j , R_j , R_{3-j} and $\{l_{\text{pre}}\}$ are mutually disjoint sets whose union is $\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t})$, we conclude that $R_j \cup R_{3-j}$ is the set $\text{Rem} \stackrel{\text{def}}{=} \{\ell_k, \dots, \ell_{n-2}\}$. This set can be empty, which leads to $e_{P_j} = \ell_k = \ell_{n-1} = l_{\text{pre}}$. Otherwise, by definition of P_j , $\ell_k \in R_{3-j}$. Below, we divide the analysis in the three following cases:

- $\text{card}(P_j) < \mathcal{S}(\alpha_j)$ (lines 7–13 of the strategy),
- $\text{card}(P_j) \geq \mathcal{S}(\alpha_j)$, and $s(\mathbf{u}) \in P_j$ or $s(\mathbf{u}) = e_{P_j}$ (lines 15–29),
- $s(\mathbf{u}) \notin P_j$, $s(\mathbf{u}) \neq e_{P_j}$ and $\text{card}(P_j) \geq \mathcal{S}(\alpha_j)$ (lines 31–41).

case: $\text{card}(P_j) < \mathcal{S}(\alpha_j)$ (lines 7–13). In line 7, we define P'_j as the set of locations that are reachable from $\llbracket \mathbf{t} \rrbracket_{s,h'}^X$ in at most $\text{card}(P_j) - 1$ steps. So, $P'_j = (\ell'_0, \dots, \ell'_{\text{card}(P_j)-1})$, and $e_{P'_j} = \ell'_{\text{card}(P_j)}$. Since $\text{card}(P_j) < \mathcal{S}(\alpha_j) < \mathcal{S}(\alpha) \leq \text{card}(\text{Path}[\mathcal{S}]_{s,h'}^X(\mathbf{t}))$, we conclude that $P'_j \subseteq \text{Path}[\mathcal{S}]_{s,h'}^X(\mathbf{t})$. We prove the properties (2), (4), (7) and (8).

Proof of (4). Trivially, $\text{card}(P_j) = \text{card}(P'_j)$.

Proof of (2). We have $P'_{3-j} = \emptyset$ (line 3) and, from (4) together with $P_j \neq \emptyset$, we conclude that $P'_j \neq \emptyset$.

Proof of (7) and (8). The memory states (s, h) and (s, h') satisfy the same formulae of the form $\mathbf{u} \in \text{sees}_X(\mathbf{t}, \mathbf{t}') \geq (\beta, 1)$ and $\mathbf{u} = \mathbf{t}$, where $\beta \in [1, \mathcal{S}_{\text{left}}(\alpha)]$. From the semantics of these formulae, we conclude that for every $i \in [0, \mathcal{S}_{\text{left}}(\alpha) - 1]$, $s(\mathbf{u}) = \ell_i$ if and only if $s(\mathbf{u}) = \ell'_i$. From (★2), we have $\mathcal{S}_{\text{left}}(\alpha) - 1 \geq \mathcal{S}(\alpha_j) > \text{card}(P_j) = \text{card}(P'_j)$. We conclude that, for every $i \in [0, \text{card}(P_j)]$, $s(\mathbf{u}) = \ell_i$ if and only if $s(\mathbf{u}) = \ell'_i$. This generalises both (7) and (8). In particular, notice that this implies that $e_{P_j} = s(\mathbf{u})$ if and only if $e_{P'_j} = s(\mathbf{u})$.

In what follows, let $\text{Rem}' = \text{Path}[\mathcal{S}]_{s,h'}^X(\mathbf{t}) \setminus (P'_j \cup \{l'_{\text{pre}}\})$. So, $\text{Rem}' = \{\ell'_{\text{card}(P_j)}, \dots, \ell'_{m-2}\}$. As we aim at satisfying the property (1), clearly the strategy should define the disjoint sets R'_j and R'_{3-j} so that $R'_j \cup R'_{3-j} = \text{Rem}'$. For the moment, we prove the following statements about Rem and Rem' :

- (ρ₁) $\text{card}(\text{Rem}) \geq \mathcal{R}(\alpha)$ and $\text{card}(\text{Rem}') \geq \mathcal{R}(\alpha)$.
- (ρ₂) $s(\mathbf{u}) \in \text{Rem}$ if and only if $s(\mathbf{u}) \in \text{Rem}'$.

Proof of (ρ₁) Since $\text{card}(P_j) = \text{card}(P'_j)$ and $\text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t})) \geq \mathcal{S}(\alpha)$, we have

$$\begin{aligned} \text{card}(R) &= \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t})) - \text{card}(P_j) - 1 \geq \mathcal{S}(\alpha) - \text{card}(P_j) - 1, \\ \text{card}(R') &= \text{card}(\text{Path}[\mathcal{S}]_{s,h'}^X(\mathbf{t})) - \text{card}(P'_j) - 1 \geq \mathcal{S}(\alpha) - \text{card}(P'_j) - 1. \end{aligned}$$

Then, $\text{card}(P_j) < \mathcal{S}(\alpha_j)$ together with (★₁)–(★₃) we conclude

$$\begin{aligned} \mathcal{S}(\alpha) - \text{card}(P_j) - 1 &= \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha) - \text{card}(P_j) - 1 \\ &\geq \mathcal{S}_{\text{right}}(\alpha) \geq \mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2) \geq \mathcal{R}(\alpha). \end{aligned}$$

Proof of (ρ₂) Consider a term $t' \in T[S]^X$ such that $\llbracket t' \rrbracket_{s,h}^X = \text{sby}_{s,h}^X(t)$. The memory states (s, h) and (s, h') satisfy the same formulae of the form $u \in \text{sees}_X(t, t') \geq (1, \beta)$ and $u = t$, where $\beta \in [1, \mathcal{S}_{\text{right}}(\alpha)]$. From the semantics of these formulae, we have

$$s(u) \in \text{Path}[S]_{s,h}^X(t) \text{ if and only if } s(u) \in \text{Path}[S]_{s,h'}^X(t).$$

Moreover, we notice that the formula

$$u \in \text{sees}_X(t, t') \geq (1, 1) \wedge \neg u \in \text{sees}_X(t, t') \geq (1, 2)$$

is satisfied by (s, h) (resp. (s, h')) if and only if $s(u) = l_{\text{pre}}$ (resp. $s(u) = l'_{\text{pre}}$). Lastly, from the properties (7) and (8), we know that $s(u) \in P_j$ if and only if $s(u) \in P'_j$.

(\Rightarrow): Suppose $s(u) \in \text{Rem}$. By definition of Rem , we have $s(u) \in \text{Path}[S]_{s,h}^X(t)$, $s(u) \notin P_j$ and $s(u) \neq l_{\text{pre}}$. Thus, $s(u) \in \text{Path}[S]_{s,h'}^X(t)$, $s(u) \notin P'_j$ and $s(u) \neq l'_{\text{pre}}$. By definition of Rem' , $s(u) \in \text{Rem}'$.

(\Leftarrow): Symmetrical to the other direction.

We now proceed with the definition of R'_1 and R'_2 , which is split depending on whether $\text{card}(R_j) < \mathcal{R}(\alpha_j)$, as shown in line 8.

case: $\text{card}(R_j) < \mathcal{R}(\alpha_j)$ (lines 9 and 10). In line 9, we define R'_j to be such that

$$R'_j \subseteq \text{Rem}', \quad \text{card}(R'_j) = \text{card}(R_j), \quad s(u) \in R'_j \text{ iff } s(u) \in R_j, \quad e_{P'_j} \notin R'_j.$$

As $R_j \subseteq \text{Rem}$, from (ρ₁) and (ρ₂) all these constraints can be clearly satisfied. Moreover, from (ρ₁) and by $\text{card}(R_j) < \mathcal{R}(\alpha_j) = \mathcal{R}(\alpha) - \mathcal{R}(\alpha_{3-j})$ we also conclude that

$$\text{card}(\text{Rem}) - \text{card}(R_j) \geq \mathcal{R}(\alpha_{3-j}), \quad \text{card}(\text{Rem}') - \text{card}(R'_j) \geq \mathcal{R}(\alpha_{3-j}).$$

Instead, from (ρ₂) we have

$$s(u) \in (\text{Rem} \setminus R_j) \text{ if and only if } s(u) \in (\text{Rem}' \setminus R'_j).$$

In line 10, we define R'_{3-j} to be the set of every location in $\text{Path}[S]_{s,h'}^X(t)$ that is not in P'_j or R'_j , nor it is equal to l'_{pre} . By definition of Rem' , $R'_{3-j} = \text{Rem}' \setminus R'_j$. We know that $R_{3-j} = \text{Rem} \setminus R_j$. So, R'_{3-j} and R_{3-j} contain at least $\mathcal{R}(\alpha_{3-j})$ locations, one of which is $e_{P'_j}$ (resp. e_{P_j}). Moreover, $s(u) \in R_{3-j}$ if and only if $s(u) \in R'_{3-j}$. We prove the properties (1), (3), (5) and (6).

Proof of (1). Directly from the definition of these sets.

Proof of (5). $\text{card}(R_j) = \text{card}(R'_j)$, $\text{card}(R_{3-j}) \geq \mathcal{R}(\alpha_{3-j})$ and $\text{card}(R'_{3-j}) \geq \mathcal{R}(\alpha_{3-j})$.

Proof of (3). P'_j describes $(\ell'_0 = \llbracket t \rrbracket_{s,h'}^X, \dots, \ell'_{\text{card}(P_j)})$, where $\ell'_{\text{card}(P_j)} = e_{P'_j} \in R_{3-j}$.

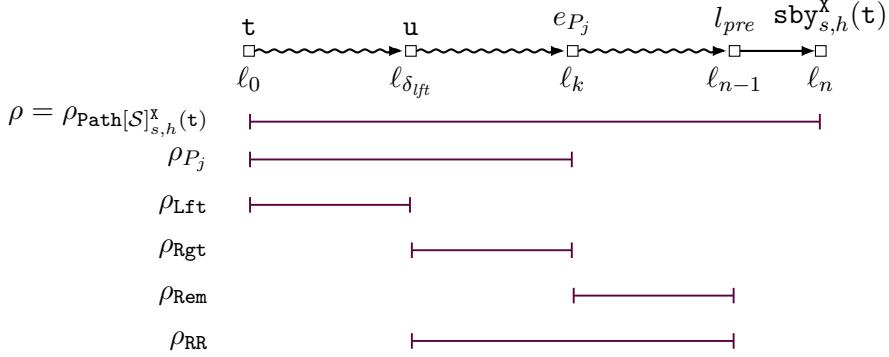
Proof of (6). Already derived above.

So, in this branch of the strategy, all the properties (1)–(8) are satisfied.

case: $\text{card}(R_j) \geq \mathcal{R}(\alpha_j)$ (lines 12 and 13). In line 12, we define R'_{3-j} to be such that

$$\begin{aligned} R'_{3-j} &\subseteq \text{Rem}', & \text{card}(R'_{3-j}) &= \min(\text{card}(R_{3-j}), \mathcal{R}(\alpha_{3-j})), \\ s(u) &\in R'_{3-j} \text{ iff } s(u) \in R_{3-j}, & e_{P'_j} &\in R'_{3-j}. \end{aligned}$$

As $R_{3-j} \subseteq \text{Rem}$, from (ρ₁) and (ρ₂), the first three constraints can be trivially satisfied. The last one, i.e. $e_{P'_j} \in R'_{3-j}$, requires some analysis, as it can only be satisfied if $\text{card}(R'_{3-j}) \geq 1$ and thus (from the second constraint) when $\text{card}(R_{3-j}) \geq 1$. We can easily see that this is the case, i.e. R_{3-j} is non-empty. *Ad absurdum*, suppose $R_{3-j} = \emptyset$. From the definition of P_j , we conclude that $e_{P_j} = l_{\text{pre}}$. Therefore, P_j and $\{l_{\text{pre}}\}$ partition $\text{Path}[S]_{s,h}^X(t)$. However, this implies $R_j = \emptyset$, in contradiction with $\text{card}(R_j) \geq \mathcal{S}(\alpha_j)$. Thus, $R_{3-j} \neq \emptyset$, and so R'_{3-j} is well-defined. From (ρ₁) and $\mathcal{R}(\alpha) = \mathcal{R}(\alpha_j) + \mathcal{R}(\alpha_{3-j})$, we have $\text{card}(\text{Rem}') - \text{card}(R'_{3-j}) \geq \mathcal{R}(\alpha_j)$. From (ρ₂),

Figure 5.16: Second case of the construction; paths of h .

$$s(u) \in (\text{Rem} \setminus R_{3-j}) \text{ if and only if } s(u) \in (\text{Rem}' \setminus R'_{3-j}).$$

In line 13, we define R'_j to be the set of every location in $\text{Path}[S]_{s,h'}^X(t)$ that is not in P'_j or R'_{3-j} , nor it is equal to l'_{pre} . By definition of Rem' , $R'_j = \text{Rem}' \setminus R'_{3-j}$. We know that $R_j = \text{Rem} \setminus R_{3-j}$. So, both R'_j and R_j contain at least $\mathcal{R}(\alpha_j)$ locations, none of which is $e_{P'_j}$ (resp. e_{P_j}). Moreover, $s(u) \in R_j$ if and only if $s(u) \in R'_j$. We prove the properties (1), (3), (5) and (6).

Proof of (1). Directly from the definition of these sets.

Proof of (5). We have $\text{card}(R'_{3-j}) = \min(\text{card}(R_{3-j}), \mathcal{R}(\alpha_{3-j}))$, whereas both R_j and R'_j have at least $\mathcal{R}(\alpha_j)$ locations.

Proof of (3). P'_j describes $(\ell'_0 = \llbracket t \rrbracket_{s,h'}^X, \dots, \ell'_{\text{card}(P_j)})$, where $\ell'_{\text{card}(P_j)} = e_{P'_j} \in R_{3-j}$.

Proof of (6). Already derived above.

So, in this branch of the strategy, all the properties (1)–(8) are satisfied.

case: $\text{card}(P_j) \geq \mathcal{S}(\alpha_j)$, and $s(u) \in P_j$ or $s(u) = e_{P_j}$ (**lines 15–29**). By definition of P_j , there is $\delta_{lft} \in [0, k]$ such that $\ell_{\delta_{lft}} = s(u)$. Equivalently, $h^{\delta_{lft}}(\llbracket t \rrbracket_{s,h}^X) = s(u)$. The set $\text{Lft} \stackrel{\text{def}}{=} \{\ell_0, \dots, \ell_{\delta_{lft}-1}\}$ is the minimal set describing the path in h going from $\llbracket t \rrbracket_{s,h}^X$ to $s(u)$. Following the strategy (line 15), we consider $\delta_{rgt} \in [0, k]$ such that $h^{(k-\delta_{rgt})}(\llbracket t \rrbracket_{s,h}^X) = s(u)$. So, $\delta_{rgt} = k - \delta_{lft}$. Informally, δ_{rgt} corresponds to the length of the minimal path ρ_{rgt} in h , going from $s(u)$ to e_{P_j} . The set $\text{Rgt} \stackrel{\text{def}}{=} \{\ell_{\delta_{lft}}, \dots, \ell_{\delta_{lft}+\delta_{rgt}-1}\}$ is disjoint from Lft and $\text{Lft} \cup \text{Rgt} = P_j$. Notice that, if $s(u) = e_{P_j}$, then Rgt is empty and $\delta_{rgt} = 0$ (and vice versa). Lastly, we define $\text{RR} \stackrel{\text{def}}{=} \text{Rgt} \cup \text{Rem} = \{\ell_{\delta_{lft}}, \dots, \ell_{n-2}\}$. Figure 5.16 should help the reader navigate between the various subsets of $\text{Path}[S]_{s,h}^X(t)$ introduced at this stage. Therein, for a set S among the ones we defined, ρ_S stands for the path described by it. We now define similar sets for (s, h') . Following line 16, we let δ'_{lft} be the length of the minimal path in h' going from $\llbracket t \rrbracket_{s,h'}^X$ to $s(u)$. This path is described by the set $\text{Lft}' \stackrel{\text{def}}{=} \{\ell'_0, \dots, \ell'_{\delta'_{lft}-1}\}$, where $\ell'_0 = \llbracket t \rrbracket_{s,h'}^X$ and $\ell'_{\delta'_{lft}} = s(u)$. We recall that the memory states (s, h) and (s, h') satisfy the same core formulae of the form $u \in \text{sees}_X(t, t') \geq (\beta, 1)$ and $u = t$, where $\beta \in [1, \mathcal{S}_{\text{left}}(\alpha)]$. From the semantics of these formulae, $\text{Lft}' \subseteq \text{Path}[S]_{s,h'}^X(t)$ is well-defined (see the proof of (S1), below). We can relate the cardinalities

$$\text{card}(\text{Lft}) = \delta_{lft},$$

$$\text{card}(\text{Lft}') = \delta'_{lft},$$

as follows:

$$\min(\text{card}(\text{Lft}), \mathcal{S}_{\text{left}}(\alpha)) = \min(\text{card}(\text{Lft}'), \mathcal{S}_{\text{left}}(\alpha)). \quad (\varsigma_1)$$

Proof of (s1). If (s, h) satisfies $\mathbf{u} = \mathbf{t}$, then so does (s, h') and $\delta_{\text{lft}} = \delta'_{\text{lft}} = 0$. In this case, $\text{Lft} = \text{Lft}' = \emptyset$. Otherwise, as $s(\mathbf{u}) \in P_j \cup \{e_{P_j}\}$, we deduce that (s, h) satisfies the formulae $\mathbf{u} \in \text{sees}_X(\mathbf{t}, \mathbf{t}') \geq (\beta, 1)$, for every $\beta \in [1, \min(\delta_{\text{lft}'}, \mathcal{S}_{\text{left}}(\alpha))]$. Then (s, h') satisfies the same formulae, and so there is $\delta \in [1, \text{card}(\text{Path}[s]_{s, h'}^X(\mathbf{t})) - 1]$ such that $h^\delta([\mathbf{t}]_{s, h'}^X) = s(\mathbf{u})$ and $\min(\delta, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta_{\text{lft}}, \mathcal{S}_{\text{left}}(\alpha))$. In this case, $\delta'_{\text{lft}} = \delta$.

We define $\text{RR}' = \{\ell'_{\delta'_{\text{lft}}}, \dots, \ell'_{m-2}\}$. Notice that RR' is a subset of $\text{Path}[s]_{s, h'}^X(\mathbf{t})$ and it is a minimal set describing a path in h' going from $s(\mathbf{u})$ to l'_{pre} . We can relate the cardinalities

$$\text{card}(\text{RR}) = n - 1 - \delta_{\text{lft}}, \quad \text{card}(\text{RR}') = m - 1 - \delta'_{\text{lft}},$$

as follows:

$$\min(\text{card}(\text{RR}), \mathcal{S}_{\text{right}}(\alpha) - 1) = \min(\text{card}(\text{RR}'), \mathcal{S}_{\text{right}}(\alpha) - 1). \quad (\varsigma_2)$$

Proof of (s2). The memory states (s, h) and (s, h') satisfy the same core formulae of the form $\mathbf{u} \in \text{sees}_X(\mathbf{t}, \mathbf{t}') \geq (1, \beta)$, where $\beta \in [1, \mathcal{S}_{\text{right}}(\alpha)]$. From the semantics of these core formulae, if $(s, h) \models \mathbf{u} \in \text{sees}_X(\mathbf{t}, \mathbf{t}') \geq (1, \mathcal{S}_{\text{right}}(\alpha))$, then the path $(\ell_{\delta_{\text{lft}}}, \dots, \ell_{n-2}, \ell_{n-1}, \ell_n)$ has length at least $\mathcal{S}_{\text{right}}(\alpha)$. Recall that $\ell_0, \dots, \ell_{n-1}$ are all distinct. So, $\text{card}(\{\ell_{\delta_{\text{lft}}}, \dots, \ell_{n-2}, \ell_{n-1}\}) \geq \mathcal{S}_{\text{right}}(\alpha)$, and thus $\text{card}(\text{RR}) \geq \mathcal{S}_{\text{right}}(\alpha) - 1$. Alike, (s, h') satisfies $\mathbf{u} \in \text{sees}_X(\mathbf{t}, \mathbf{t}') \geq (1, \mathcal{S}_{\text{right}}(\alpha))$, and $\text{card}(\text{RR}') \geq \mathcal{S}_{\text{right}}(\alpha) - 1$. Else, suppose there is $\beta \in [1, \mathcal{S}_{\text{right}}(\alpha) - 1]$ such that $(s, h) \models \mathbf{u} \in \text{sees}_X(\mathbf{t}, \mathbf{t}') \geq (1, \beta)$ but $(s, h) \not\models \mathbf{u} \in \text{sees}_X(\mathbf{t}, \mathbf{t}') \geq (1, \beta + 1)$. In this case, RR contains exactly $\beta - 1$ locations and, as (s, h') satisfies the same core formulae, so does RR' .

Following the strategy (line 17), we split the proof depending on whether $\delta_{\text{rgt}} < \mathcal{S}_{\text{right}}(\alpha_j)$.

case: $\delta_{\text{rgt}} < \mathcal{S}_{\text{right}}(\alpha_j)$ (lines 18–25). Let $\text{Rgt}' \stackrel{\text{def}}{=} \{\ell'_{\delta'_{\text{lft}}}, \dots, \ell'_{\delta'_{\text{lft}} + \delta_{\text{rgt}} - 1}\}$. We have

$$\text{card}(\text{Rgt}') = \text{card}(\text{Rgt}) = \delta_{\text{rgt}}.$$

We show that

$$\min(\text{card}(\text{RR}) - \delta_{\text{rgt}}, \mathcal{R}(\alpha)) = \min(\text{card}(\text{RR}') - \delta_{\text{rgt}}, \mathcal{R}(\alpha)). \quad (\varsigma_3)$$

Notice that, since $\mathcal{R}(\alpha) \geq 1$ and $\text{card}(\text{RR}) - \text{card}(\text{Rgt}) \geq 0$, this implies that $\text{card}(\text{RR}') - \delta_{\text{rgt}} \geq 0$ and so, by definition of Rgt' and RR' , we conclude $\text{Rgt}' \subseteq \text{RR}'$.

Proof of (s3). Directly from (s2) we can subtract δ_{rgt} on both sides to conclude that

$$\begin{aligned} &\min(\text{card}(\text{RR}) - \delta_{\text{rgt}}, \mathcal{S}_{\text{right}}(\alpha) - 1 - \delta_{\text{rgt}}) \\ &= \min(\text{card}(\text{RR}') - \delta_{\text{rgt}}, \mathcal{S}_{\text{right}}(\alpha) - 1 - \delta_{\text{rgt}}). \end{aligned}$$

To conclude the proof it is sufficient to show that $\mathcal{S}_{\text{right}}(\alpha) - 1 - \delta_{\text{rgt}} \geq \mathcal{R}(\alpha)$.

From (s3), we have

$$\mathcal{S}_{\text{right}}(\alpha) - 1 - \delta_{\text{rgt}} \geq \mathcal{S}_{\text{right}}(\alpha_j) + \mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2) - \delta_{\text{rgt}}.$$

From $\mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2) = \mathcal{R}(\alpha)$ and $\delta_{\text{rgt}} < \mathcal{S}_{\text{right}}(\alpha_j)$, The right-hand side of this inequality is at least $\mathcal{R}(\alpha)$.

We define $\text{Rem}' \stackrel{\text{def}}{=} \text{RR}' \setminus \text{Rgt}' = \{\ell'_{\delta'_{\text{lft}} + \delta_{\text{rgt}}}, \dots, \ell'_{m-2}\}$. Since $\text{Rgt}' \subseteq \text{RR}'$, we have $\text{card}(\text{Rem}') = \text{card}(\text{RR}') - \text{card}(\text{Rgt}')$, and (s3) can be restated as

$$\min(\text{card}(\text{Rem}), \mathcal{R}(\alpha)) = \min(\text{card}(\text{Rem}'), \mathcal{R}(\alpha)).$$

Let $\bar{\delta} = \delta'_{\text{lft}} + \delta_{\text{rgt}}$ (as in line 18). As formalised in line 19, we define P'_j to be the set of locations reachable from $[\mathbf{t}]_{s, h'}^X$ in at most $\bar{\delta} - 1$ steps. So, $P'_j = \{\ell_0, \dots, \ell_{\bar{\delta}-1}\}$, which is the union of the disjoint sets Lft' and Rgt' . Here, $e_{P'_j} \stackrel{\text{def}}{=} h'^{\bar{\delta}}([\mathbf{t}]_{s, h_j}^X) = \ell'_{\bar{\delta}}$,

and $P'_j \subseteq \text{Path}[s]_{s,h'}^X(t)$ is a minimal set describing the non-empty path in h' going from $\llbracket t \rrbracket_{s,h'}^X$ to the location $e_{P'_j}$. We prove the properties (2), (4), (7) and (8).

Proof of (4). We have,

$$\text{card}(P_j) = \text{card}(\text{Lft}) + \text{card}(\text{Rgt}), \quad \text{card}(P'_j) = \text{card}(\text{Lft}') + \text{card}(\text{Rgt}).$$

Therefore, directly from (s1)

$$\min(\text{card}(P_j), \mathcal{S}_{\text{left}}(\alpha)) = \min(\text{card}(P'_j), \mathcal{S}_{\text{left}}(\alpha)).$$

By (s2), $\mathcal{S}_{\text{left}}(\alpha) \geq \mathcal{S}(\alpha_j)$, and by hypothesis $\text{card}(P_j) \geq \mathcal{S}(\alpha_j)$. From the equivalence above, we conclude that $\text{card}(P'_j) \geq \mathcal{S}(\alpha_j)$.

Proof of (2). As $P_j \neq \emptyset$ is non-empty, by (4), so is P'_j . $P'_{3-j} = \emptyset$ (line 3) proves (2).

Proof of (7) and (8). With respect to the statements (7) and (8), we have $\delta_1 = \delta_{\text{lft}}$ and $\delta_2 = \delta_{\text{rgt}}$. Since $\text{card}(P_j) - \delta_1 = \delta_{\text{rgt}} = \text{card}(P'_j) - \delta_2$ the first equivalence in both (7) and (8) is satisfied, i.e.

$$\min(\text{card}(P_j) - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(P'_j) - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j))$$

From (s1), so is the second equivalence.

We now proceed with the definition of R'_1 and R'_2 , which is split depending on whether $\text{card}(R_j) < \mathcal{R}(\alpha_j)$, as shown in line 20.

case: $\text{card}(R_j) < \mathcal{R}(\alpha_j)$ (lines 21 and 22). As formalised in line 21, we define R'_j to be the set of locations in $\text{Path}[s]_{s,h'}^X(t)$ that reach l'_{pre} in at most $\text{card}(R_j)$ steps, l'_{pre} excluded. See the corresponding drawing in the right-hand side of Figure 5.15 for a representation of this set. So, $R'_j = \{\ell'_{m-1-\text{card}(R_j)}, \dots, \ell'_{m-2}\}$ and $\text{card}(R'_j) = \text{card}(R_j)$. From $R_j \subseteq \text{Rem}$, we have $\text{card}(\text{Rem}) - \text{card}(R_j) \geq 0$. Together with $\text{card}(R_j) < \mathcal{R}(\alpha_j)$ and by (s3), this implies $m-1 - \text{card}(R'_j) = \text{card}(\text{Rem}') - \text{card}(R'_j) \geq \delta'_{\text{lft}} + \delta_{\text{rgt}}$ and so $R'_j \subseteq \text{Rem}'$. Then, R'_{3-j} is defined (line 22) as the locations in $\text{Path}[s]_{s,h'}^X(t)$ that are not in P'_j or in R'_j , nor they are equal to l'_{pre} . Precisely, we have $R'_{3-j} = \{\ell'_{\bar{\delta}}, \dots, \ell'_{m-1-\text{card}(R_j)-1}\}$. Notice that $R'_j \cup R'_{3-j} = \text{Rem}'$. We prove the properties (1), (3), (5) and (6).

Proof of (1). Recall that $\text{Path}[s]_{s,h'}^X(t) = \{\ell'_0, \dots, \ell'_{m-1}\}$, where $\ell'_0, \dots, \ell'_{m-1}$ are all distinct. Explicitly, we defined:

$$\begin{aligned} P'_j &= \{\ell'_0, \dots, \ell'_{\bar{\delta}-1}\}, & R'_{3-j} &= \{\ell'_{\bar{\delta}}, \dots, \ell'_{m-1-\text{card}(R_j)-1}\}, \\ R'_j &= \{\ell'_{m-1-\text{card}(R_j)}, \dots, \ell'_{m-2}\}, & l'_{\text{pre}} &= \ell'_{m-1}. \end{aligned}$$

Proof of (5). We have $\text{card}(R_j) = \text{card}(R'_j)$. Let us show that

$$\min(\text{card}(R_{3-j}), \mathcal{R}(\alpha_{3-j})) = \min(\text{card}(R'_{3-j}), \mathcal{R}(\alpha_{3-j})).$$

Since $R_j \cup R_{3-j} = \text{Rem}$ and $R'_j \cup R'_{3-j} = \text{Rem}'$, where R_j and R_{3-j} (resp. R'_j and R'_{3-j}) are disjoint, by (s3) we derive

$$\min(\text{card}(R_j) + \text{card}(R_{3-j}), \mathcal{R}(\alpha)) = \min(\text{card}(R'_j) + \text{card}(R'_{3-j}), \mathcal{R}(\alpha)).$$

Then, the desired equivalence holds from the fact that $\mathcal{R}(\alpha) = \mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2)$ and $\text{card}(R_j) < \mathcal{R}(\alpha_j)$.

Proof of (3). We know that P'_j describes a path in h' , starting from $\llbracket t \rrbracket_{s,h'}^X$. We must check that $e_{P'_j} \in R'_{3-j} \cup \{l'_{\text{pre}}\}$. If $R'_{3-j} \neq \emptyset$, then $e_{P'_j} = \ell'_{\bar{\delta}} \in R'_{3-j}$. Otherwise, by (5), R'_{3-j} is also empty, and so $e_{P'_j} = l'_{\text{pre}}$. By definition, means that P_j and $\{l'_{\text{pre}}\}$ partition $\text{Path}[s]_{s,h}^X(t)$, and $R_j = \emptyset$. Again by (5), $R'_j = \emptyset$ and thus $e_{P'_j} = \ell'_{\bar{\delta}} = \ell'_{m-1} = l'_{\text{pre}}$.

Proof of (6). $s(u)$ belongs to both P_j and P'_j . By (1) and since P_j is disjoint from R_1 and R_2 , we conclude that for every $i \in \{1, 2\}$ $s(u) \notin R_i$ and $s(u) \notin R'_i$.

Therefore, in this branch of the strategy, all the properties (1)–(8) are satisfied.

case: $\text{card}(R_j) \geq \mathcal{R}(\alpha_j)$ (lines 24 and 25). Following the strategy, in line 24 we let R'_{3-j} be the set of locations that are reached from $e_{P'_j} = h^{\bar{\delta}}([\mathbf{t}]_{s,h'}^{\mathbf{x}})$ in at most $\delta_{3-j} = \min(\text{card}(R_{3-j}), \mathcal{R}(\alpha_{3-j})) - 1$ steps. Explicitly, $R'_{3-j} = \{\delta_{\bar{\delta}}, \dots, \delta'_{\bar{\delta} + \delta_{3-j} - 1}\}$. As $\mathcal{R}(\alpha_{3-j}) < \mathcal{R}(\alpha)$, by (s3) we conclude that $R'_{3-j} \subseteq \text{Rem}'$. Then (line 25), R_j is defined as the locations in $\text{Path}[s]_{s,h'}^{\mathbf{x}}(\mathbf{t})$ that are not in P'_j , nor in R'_{3-j} , and they are different from l'_{pre} . In particular, $R_j = \{\ell'_{\bar{\delta} + \delta_{3-j}}, \dots, \ell'_{m-2}\}$ and $R_j = \text{Rem} \setminus R_{3-j}$. We prove the properties (1), (3), (5) and (6).

Proof of (1). Explicitly, we defined:

$$\begin{aligned} P'_j &= \{\ell'_0, \dots, \ell'_{\bar{\delta}-1}\}, & R'_{3-j} &= \{\ell'_{\bar{\delta}}, \dots, \ell'_{\bar{\delta} + \delta_{3-j} - 1}\}, \\ R'_j &= \{\ell'_{\bar{\delta} + \delta_{3-j}}, \dots, \ell'_{m-2}\}, & l'_{pre} &= \ell'_{m-1}. \end{aligned}$$

Proof of (5). By definition, $\text{card}(R'_{3-j}) = \min(\text{card}(R_{3-j}), \mathcal{R}(\alpha_{3-j}))$, which shows the statement for the index $3 - j$. Thanks to this equivalence, from (s3) we conclude that

$$\begin{aligned} &\min(\text{card}(\text{Rem}) - \text{card}(R'_{3-j}), \mathcal{R}(\alpha) - \text{card}(R'_{3-j})) \\ &= \min(\text{card}(\text{Rem}') - \text{card}(R'_{3-j}), \mathcal{R}(\alpha) - \text{card}(R'_{3-j})). \end{aligned}$$

Recall that $\text{card}(R_j) \geq \mathcal{R}(\alpha_j)$, $R_j \cup R_{3-j} = \text{Rem}$ and $R'_j \cup R'_{3-j} = \text{Rem}'$, where R_j and R_{3-j} (resp. R'_j and R'_{3-j}) are disjoint. Then, the equivalence above leads to $\text{card}(R'_j) \geq \mathcal{R}(\alpha_j)$ directly from

$$\begin{aligned} \mathcal{R}(\alpha) - \text{card}(R'_{3-j}) &\geq \mathcal{R}(\alpha) - \mathcal{R}(\alpha_{3-j}) = \mathcal{R}(\alpha_j) \\ \text{card}(\text{Rem}) - \text{card}(R'_{3-j}) &\geq \text{card}(\text{Rem}) - \text{card}(R_{3-j}) = \text{card}(R_j) \geq \mathcal{R}(\alpha_j) \\ \text{card}(R'_j) &= \text{card}(\text{Rem}') - \text{card}(R'_{3-j}) \end{aligned}$$

Proof of (3). Proved from (5), as shown in the previous case of the proof.

Proof of (6). As in the previous case, for all $i \in \{1, 2\}$ $s(u) \notin R_i$ and $s(u) \notin R'_i$. So, in this branch of the strategy, all the properties (1)–(8) are satisfied.

case: $\delta_{rgt} \geq \mathcal{S}_{\text{right}}(\alpha_j)$ (lines 27–29). In line 27, we define R'_j to be the set of locations in $\text{Path}[s]_{s,h'}^{\mathbf{x}}(\mathbf{t})$ that reach l'_{pre} in at most $\delta_j = \min(\text{card}(R_j), \mathcal{R}(\alpha_j))$ steps, l'_{pre} excluded. Afterwards (line 28), R'_{3-j} is defined as the set of locations in $\text{Path}[s]_{s,h'}^{\mathbf{x}}(\mathbf{t})$ that reach $\ell'_{m-1-\delta_j}$ in at most $\delta_{3-j} = \min(\text{card}(R_{3-j}), \mathcal{R}(\alpha_{3-j}))$ steps, $\ell'_{m-1-\delta_j}$ excluded. See the corresponding drawings in the right-hand side of Figure 5.15 for a representation of these two sets. So, $\text{card}(R'_j) = \delta_j$ and $\text{card}(R'_{3-j}) = \delta_{3-j}$. Explicitly,

$$R'_j = \{\ell'_{m-1-\delta_j}, \dots, \ell'_{m-2}\}, \quad R'_{3-j} = \{\ell'_{m-1-(\delta_j + \delta_{3-j})}, \dots, \ell'_{m-2-\delta_j}\}.$$

Clearly, $R'_j \cap R'_{3-j} = \emptyset$. Let $\text{Rem}' \stackrel{\text{def}}{=} R'_j \cup R'_{3-j}$. From $\text{Rem} = R_j \cup R_{3-j}$, together with the definition of δ_j and δ_{3-j} , as well as $\mathcal{R}(\alpha_j) + \mathcal{R}(\alpha_{3-j}) = \mathcal{R}(\alpha)$, we derive

$$\text{card}(\text{Rem}') = \min(\text{card}(\text{Rem}), \mathcal{R}(\alpha)). \quad (\text{s4})$$

By relying on (s2) we show that

$$\text{card}(\text{RR}') - \text{card}(\text{Rem}') \geq \mathcal{S}_{\text{right}}(\alpha_j). \quad (\text{s5})$$

Notice that, since $\mathcal{S}_{\text{right}}(\alpha_j) \geq 1$ and $\text{card}(\text{RR}) - \text{card}(\text{Rem}) = \text{card}(\text{Rgt}) \geq 0$, this implies that $\text{card}(\text{RR}') - \text{card}(\text{Rem}') \geq 0$ and so, directly from the definitions of RR' and Rem' , we conclude that $\text{Rem}' \subseteq \text{RR}'$. This also shows that R'_j and R'_{3-j} are well-defined.

Proof of (s5). From (s2) we have

$$\begin{aligned} &\min(\text{card}(\text{RR}) - \text{card}(\text{Rem}'), \mathcal{S}_{\text{right}}(\alpha) - 1 - \text{card}(\text{Rem}')) \\ &= \min(\text{card}(\text{RR}') - \text{card}(\text{Rem}'), \mathcal{S}_{\text{right}}(\alpha) - 1 - \text{card}(\text{Rem}')). \end{aligned}$$

By (\star_3) together with $(\textcolor{red}{s}_4)$, $\mathcal{S}_{\text{right}}(\alpha) - 1 - \text{card}(\text{Rem}') \geq \mathcal{S}_{\text{right}}(\alpha_j)$. Then, $(\textcolor{red}{s}_5)$ follows directly from:

$$\min(\text{card}(\text{RR}) - \text{card}(\text{Rem}')) \geq \text{card}(\text{RR}) - \text{card}(\text{Rem}) = \text{card}(\text{Rgt}) = \delta_{rgt} \geq \mathcal{S}_{\text{right}}(\alpha_j).$$

We let $\text{Rgt}' = \text{RR}' \setminus \text{Rem}' = \{\ell'_{\delta_{lft}}, \dots, \ell'_{m-2-(\delta_j+\delta_{3-j})}\}$. As $\text{Rem}' \subseteq \text{RR}'$, from $(\textcolor{red}{s}_5)$ we have $\text{card}(\text{Rgt}') \geq \mathcal{S}_{\text{right}}(\alpha_j)$. In line 29, we define P'_j to be the set of locations of $\text{Path}[s]_{s,h'}^X(t)$ that are not in R'_j or in R'_{3-j} , nor they are equal to l'_{pre} . Explicitly, $P'_j = \{\ell'_0, \dots, \ell'_{m-2-(\delta_j+\delta_{3-j})}\}$. So, by definition, $P'_j = \text{Lft}' \cup \text{Rgt}'$. We prove that the properties $(1)-(8)$ are satisfied.

Proof of (1). Explicitly, we defined:

$$\begin{aligned} P'_j &= \{\ell'_0, \dots, \ell'_{m-2-(\delta_j+\delta_{3-j})}\}, & R'_{3-j} &= \{\ell'_{m-1-(\delta_j+\delta_{3-j})}, \dots, \ell'_{m-2-\delta_j}\}, \\ R_j &= \{\ell'_{m-1-\delta_j}, \dots, \ell'_{m-2}\}, & l'_{pre} &= \ell'_{m-1}. \end{aligned}$$

Proof of (4). Recall that $\text{card}(P_j) \geq \mathcal{S}(\alpha_j)$. By (1) , together with $\text{card}(\text{Rem}') \leq \mathcal{R}(\alpha)$, and $\text{card}(\text{Path}[s]_{s,h'}^X(t)) \geq \mathcal{S}(\alpha)$, we have

$$\begin{aligned} \text{card}(P'_j) &= \text{card}(\text{Path}[s]_{s,h'}^X(t)) - (\text{card}(\text{Rem}') + 1) \\ &\geq \text{card}(\text{Path}[s]_{s,h'}^X(t)) - (\mathcal{R}(\alpha) + 1) \geq \mathcal{S}(\alpha) - (\mathcal{R}(\alpha) + 1). \end{aligned}$$

From (\star_1) and (\star_3) , we conclude that $\text{card}(P'_j) \geq \mathcal{S}(\alpha_j)$, since

$$\mathcal{S}(\alpha) - (\mathcal{R}(\alpha) + 1) \geq \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha_j) \geq \mathcal{S}(\alpha_j).$$

Proof of (2). We have $\text{card}(P'_j) \geq \mathcal{S}(\alpha_j)$ and $P_{3-j} = \emptyset$ (line 3).

Proof of (7) and (8). With respect to the statements (7) and (8) , we have $\delta_1 = \delta_{lft}$ and $\delta_2 = \delta_{rgt}$. By $\text{card}(\text{Rgt}) = \delta_{rgt} \geq \mathcal{S}_{\text{right}}(\alpha_j)$ and $\text{card}(\text{Rgt}') \geq \mathcal{S}_{\text{right}}(\alpha_j)$, the first equivalence in both (7) and (8) is satisfied. By $(\textcolor{red}{s}_1)$, so is the second equivalence.

Proof of (5). Directly from $\text{card}(R'_i) = \min(\text{card}(R_i), \mathcal{R}(\alpha_i))$, for all $i \in \{1, 2\}$.

Proof of (3). Proved from (5) , as shown in the previous two cases of the proof.

Proof of (6). As in the previous case, for all $i \in \{1, 2\}$ $s(u) \notin R_i$ and $s(u) \notin R'_i$.

case: $s(u) \notin P_j$, $s(u) \neq e_{P_j}$ and $\text{card}(P_j) \geq \mathcal{S}(\alpha_j)$ (**lines 31–41**). Following the strategy, we divide the proof depending on whether $\text{card}(R_1) < \mathcal{R}(\alpha_1)$ and $\text{card}(R_2) < \mathcal{R}(\alpha_2)$ hold.

case: $\text{card}(R_1) < \mathcal{R}(\alpha_1)$ and $\text{card}(R_2) < \mathcal{R}(\alpha_2)$ (**lines 32–34**). In line 32, we define Rem' to be the set of locations in $\text{Path}[s]_{s,h'}^X(t)$ that reach l'_{pre} in at most $\text{card}(\text{Rem}) = \text{card}(R_1) + \text{card}(R_2)$ steps, l'_{pre} excluded. So, $\text{Rem}' = \{\ell'_{m-1-\text{card}(\text{Rem})}, \dots, \ell'_{m-2}\}$ and

$$\text{card}(\text{Rem}') = \text{card}(R_1) + \text{card}(R_2). \quad (\tau_1)$$

Clearly, as $\text{Path}[s]_{s,h'}^X(t)$ contains at least $\mathcal{S}(\alpha) > \mathcal{R}(\alpha)$ locations, Rem' is well-defined. Afterwards (line 33), we define P'_j as the set of locations in $\text{Path}[s]_{s,h'}^X(t)$ that are not in Rem' nor they are equal to l'_{pre} . Explicitly, $P'_j = \{\ell'_0, \dots, \ell'_{m-2-\text{card}(\text{Rem})}\}$ and $e_{P'_j} = \ell'_{m-1-\text{card}(\text{Rem})}$. We prove the three following statements:

$$(\tau_2) \text{ card}(P'_j) \geq \mathcal{S}(\alpha_j), \quad (\tau_3) s(u) \notin P'_j \cup \{e_{P'_j}\}, \quad (\tau_4) s(u) \in \text{Rem} \text{ iff } s(u) \in \text{Rem}'.$$

Proof of (τ₂). As $\text{card}(\text{Path}[s]_{s,h'}^X(t)) \geq \mathcal{S}(\alpha)$ and $\text{card}(\text{Rem}') < \mathcal{R}(\alpha)$, by $(\star_1)-(\star_3)$, $\text{card}(P'_j) = \text{card}(\text{Path}[s]_{s,h'}^X(t)) - \text{card}(\text{Rem}') - 1 \geq \mathcal{S}(\alpha) - \mathcal{R}(\alpha) - 1 \geq \mathcal{S}(\alpha_j)$.

Proof of (τ₃). We recall that P_j is a minimal set describing a non-empty path starting at $\llbracket t \rrbracket_{s,h}^X$. In particular, at the beginning of the proof, we assumed $P_j = \{\ell_0, \dots, \ell_{k-1}\}$, where $k = \text{card}(P_j)$, and $R_1 \cup R_2 = \text{Rem} = \{\ell_k, \dots, \ell_{n-2}\}$. Recall moreover that $\ell_{n-1} = l_{pre}$. So, the locations in $\text{Path}[s]_{s,h}^X(t)$ that belong to

P_j precede the ones in R_1 or R_2 . Since we are assuming $s(u) \notin P_j \cup \{l_{pre}\}$, we conclude that if $s(u) \in \text{Path}[s]_{s,h}^X(t)$, then $s(u) \in \{\ell_{k+1}, \dots, \ell_{n-1}\}$. In terms of core formulae, this means that among the core formulae of the form $u \in \text{sees}_X(t, t') \geq (1, \beta)$, where $\beta \in [1, S_{\text{right}}(\alpha)]$, (s, h) can only satisfy the ones where $\beta \leq \text{card}(\text{Rem})$. As $(s, h) \approx_{X,\alpha}^S (s, h')$, the same holds for (s, h') . Thus, if $s(u) \in \text{Path}[s]_{s,h'}^X(t)$, $s(u) \in \{\ell'_{m-\text{card}(\text{Rem})}, \dots, \ell'_{m-1}\}$. Lastly, by definition of P'_j and $e_{P'_j} = \ell_{m-1-\text{card}(\text{Rem})}$, we conclude that $s(u) \notin P'_j \cup \{e_{P'_j}\}$.

Proof of (τ_4) . As previously noticed, the formula

$$u \in \text{sees}_X(t, t') \geq (1, 1) \wedge \neg u \in \text{sees}_X(t, t') \geq (1, 2)$$

is satisfied by (s, h) (resp. (s, h')) if and only if $s(u) = l_{pre}$ (resp. $s(u) = l'_{pre}$). So, from $(s, h) \approx_{X,\alpha}^S (s, h')$, we know that $s(u) = l_{pre}$ iff $s(u) = l'_{pre}$.

(\Rightarrow) : Suppose $s(u) \in \text{Rem}$. As $l_{pre} \notin \text{Rem}$, we have $s(u) \neq l_{pre}$, which in turn implies $s(u) \neq l'_{pre}$ from the double implication above. Moreover, (s, h) satisfies $u \in \text{sees}_X(t, t') \geq (1, 1)$, for some $t' \in T[s]^X$, and thus, by $(s, h) \approx_{X,\alpha}^S (s, h')$, so does (s, h') . This implies that $s(u) \in \text{Path}[s]_{s,h'}^X(t)$. We know that $s(u) \notin P'_j \cup \{e'\}$. Therefore, $s(u)$ must be in $\{\ell'_{m-\text{card}(\text{Rem})}, \dots, \ell'_{m-2}\} \subseteq \text{Rem}'$.

Directly from (τ_1) we conclude that the properties (4) and (2) are satisfied (recall that $P_{3-j} = \emptyset$, see line 3). Similarly, from (τ_2) , we conclude that the properties (7) and (8) are satisfied. Following the strategy, we now move to the case where $\text{card}(R_1) \geq \mathcal{R}(\alpha_1)$ or $\text{card}(R_2) \geq \mathcal{R}(\alpha_2)$. We leave open the definition of R_1 and R_2 , which is carried out below, for both the cases (as in lines 39–41).

case: $\text{card}(R_1) \geq \mathcal{R}(\alpha_1)$ or $\text{card}(R_2) \geq \mathcal{R}(\alpha_2)$ (lines 36–38). In line 36, we define P'_j as the set of locations reachable from $\llbracket t \rrbracket_{s,h'}^X$ in at most $S(\alpha_j) - 1$ steps. So, $\text{card}(P'_j) = S(\alpha_j)$. As $\text{card}(\text{Path}[s]_{s,h'}^X(t)) \geq S(\alpha) > S(\alpha_j) - 1$, P'_j is well-defined and a subset of $\text{Path}[s]_{s,h'}^X(t)$. Explicitly, $P'_j = \{\ell'_0, \dots, \ell'_{S(\alpha_j)-1}\}$ and $e_{P'_j} = \ell'_{S(\alpha_j)}$. Afterwards (line 37) we define Rem' as the set of locations in $\text{Path}[s]_{s,h'}^X(t)$ that are not in P'_j nor they are equal to l'_{pre} . Explicitly, $\text{Rem}' = \{\ell'_{S(\alpha_j)}, \dots, \ell'_{m-2}\}$. We prove that:

$$(\tau_5) \text{ card}(\text{Rem}') \geq \mathcal{R}(\alpha), \quad (\tau_6) s(u) \notin P'_j \cup \{e_{P'_j}\}, \quad (\tau_7) s(u) \in \text{Rem} \text{ iff } s(u) \in \text{Rem}'.$$

Proof of (τ_5) . As $\text{card}(\text{Path}[s]_{s,h'}^X(t)) \geq S(\alpha)$ and $\text{card}(P'_j) = S(\alpha_j)$, by (\star_1) – (\star_3) , $\text{card}(\text{Rem}') = \text{card}(\text{Path}[s]_{s,h'}^X(t)) - \text{card}(P'_j) - 1 \geq S(\alpha) - S(\alpha_j) - 1 \geq \mathcal{R}(\alpha)$.

Proof of (τ_6) . Ad absurdum, suppose $s(u) \in P'_j \cup \{e_{P'_j}\}$. Since $\text{card}(P'_j) = S(\alpha_j)$, this implies that there is $\delta \in [0, S(\alpha_j)]$ such that $h'^{\delta}(\llbracket t \rrbracket_{s,h'}^X) = s(u)$. We divide the proof depending on whether $\beta = 0$.

case: $\beta = 0$. If $\beta = 0$ then $\llbracket t \rrbracket_{s,h'}^X = s(u)$ and, from the equisatisfaction of the formula $u = t$ we conclude that $\llbracket t \rrbracket_{s,h}^X = s(u)$. However, by definition of P_j , this implies $s(u) \in P_j$: a contradiction.

case: $\beta \geq 1$. Let $t' \in T[s]^X$ such that $\llbracket t' \rrbracket_{s,h'}^X = \text{sby}_{s,h'}^X(t)$. Notice that $S_{\text{left}}(\alpha) \geq S(\max(\alpha_1, \alpha_2)) + 1$ (i.e. (\star_2)), and so the formulae $u \in \text{sees}_X(t, t') \geq (\beta, 1)$ and $u \in \text{sees}_X(t, t') \geq (\beta + 1, 1)$ belongs to $\text{Core}[s](X, \alpha)$. We have,

$$(s, h') \models u \in \text{sees}_X(t, t') \geq (\beta, 1), \quad (s, h') \not\models u \in \text{sees}_X(t, t') \geq (\beta + 1, 1).$$

From $(s, h) \approx_{X,\alpha}^S (s, h')$, we have $(s, h) \models u \in \text{sees}_X(t, t') \geq (\beta, 1)$ and $(s, h) \not\models u \in \text{sees}_X(t, t') \geq (\beta + 1, 1)$. Therefore, $\ell_{\beta} = s(u)$. However, this location belongs to P_j , again in contradiction with $s(u) \notin P_j$.

In both cases, we derive $s(u) \notin P'_j \cup \{e_{P'_j}\}$.

Proof of (τ_7) . Analogous to the proof of (τ_4) .

Directly from $\text{card}(P'_j) = \mathcal{S}(\alpha_j)$ we conclude that the properties (4) and (2) are satisfied. Similarly, from (τ_6) , we conclude that the properties (7) and (8) are satisfied. We now move to the definition of R'_1 and R'_2 .

Below, we let $k \in \{1, 2\}$ be an index such that $\text{card}(R_k) \geq \mathcal{R}(\alpha_k)$, if any. Otherwise, let k take an arbitrary value in $\{1, 2\}$. This step corresponds to the lines 34 and 38 of the strategy. In line 40, we define R'_{3-k} to be a subset of Rem' such that

- $\text{card}(R'_{3-k}) = \min(\text{card}(R_{3-k}), \mathcal{R}(\alpha_{3-k}))$,
- $e_{P'_j} \in R'_{3-k}$ if and only if $k \neq j$,
- $s(u) \in R'_{3-k}$ if and only if $s(u) \in R_{3-k}$.

From $(\tau_1)/(\tau_5)$ and $(\tau_4)/(\tau_7)$ (depending on which of the two cases above holds), these three conditions can always be satisfied. Following line 41 in the strategy, we define $R'_k = \text{Rem}' \setminus R'_{3-k}$. We prove that the properties (1) , (3) (5) and (6) .

Proof of (1) . Directly from the definition of P'_j , R'_j and R'_{3-j} .

Proof of (5) . The property holds by definition for the set R'_{3-k} . We show that

$$\min(R_k, \mathcal{R}(\alpha_k)) = \min(R'_k, \mathcal{R}(\alpha_k)).$$

we divide the proof depending on whether $R_k < \mathcal{R}(\alpha_k)$.

case: $R_k < \mathcal{R}(\alpha_k)$. From the definition of k , we derive that $\text{card}(R_{3-k}) < \mathcal{R}(\alpha_{3-k})$.

So, the strategy follows the case corresponding to the lines 32–34. From (τ_1) , $\text{card}(R') = \text{card}(R_k) + \text{card}(R_{3-k})$. By definition of R'_{3-k} , $\text{card}(R_{3-k}) = \text{card}(R'_{3-k})$. By definition of R'_k , we derive $\text{card}(R'_k) = \text{card}(R') - \text{card}(R'_{3-k}) = \text{card}(R_k)$.

case: $R_k \geq \mathcal{R}(\alpha_k)$. The strategy follows the case corresponding to the lines 36–38.

Form (τ_5) , $\text{card}(\text{Rem}') \geq \mathcal{R}(\alpha)$. By definition of R'_{3-k} , $\text{card}(R_{3-k}) \leq \mathcal{R}(\alpha_{3-k})$.

We have $\text{card}(R_k) = \text{card}(\text{Rem}') - \text{card}(R_{3-k}) \geq \mathcal{R}(\alpha) - \mathcal{R}(\alpha_{3-k}) \geq \mathcal{R}(\alpha_k)$.

Proof of (3) . Proved from (5) , as done in previous cases of the proof.

Proof of (6) . The property holds by definition for the set R'_{3-k} . For R'_k :

(\Rightarrow) : Suppose $s(u) \in R_k$. By definition, $s(u) \in \text{Rem}$ and $s(u) \notin R_{3-k}$. By definition of R'_{3-k} , $s(u) \notin R'_{3-k}$. From $(\tau_4)/(\tau_7)$, $s(u) \in \text{Rem}'$. As $R'_k = \text{Rem}' \setminus R'_{3-k}$, $s(u) \in R'_k$.

(\Leftarrow) : Symmetrical to the other direction.

All the properties (1) – (8) are satisfied. This concludes the analysis of the strategy in Figure 5.15, which is found to correctly define the sets we need in order to prove the lemma.

After showing that P'_1 , P'_2 , R'_1 and R'_2 satisfy (1) – (8) , we rely these sets in order to define the heaps h'_1 and h'_2 such that $(s, h_1) \approx_{X, \alpha_1}^{\mathcal{S}} (s, h'_1)$ and $(s, h_2) \approx_{X, \alpha_2}^{\mathcal{S}} (s, h'_2)$. First, we define the heaps $\widehat{h}_1 \stackrel{\text{def}}{=} h_1 \setminus \{(\ell, \ell') \in h_1 \mid \ell \in P_1 \cup R_1 \cup \{l_{\text{pre}}\}\}$ and $\widehat{h}_2 \stackrel{\text{def}}{=} h_2 \setminus \{(\ell, \ell') \in h_2 \mid \ell \in P_2 \cup R_2\}$. By definition of P_1 , P_2 , R_1 and R_2 , and the assumption $l_{\text{pre}} \in \text{dom}(h_1)$, both $\text{dom}(\widehat{h}_1) \cap \text{Path}[\mathcal{S}]_{s,h}^X(t)$ and $\text{dom}(\widehat{h}_2) \cap \text{Path}[\mathcal{S}]_{s,h}^X(t)$ are empty. Moreover, from $h = h_1 + h_2$, we have that

$$h = \widehat{h}_1 + \widehat{h}_2 + \{(\ell, \ell') \in h \mid \ell \in \text{Path}[\mathcal{S}]_{s,h}^X(t)\}.$$

From the hypothesis $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Path}[\mathcal{S}]_{s,h}^X(t)\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Path}[\mathcal{S}]_{s,h'}^X(t)\}$ we derive $\widehat{h}_1 + \widehat{h}_2 = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Path}[\mathcal{S}]_{s,h'}^X(t)\}$. We define the heaps h'_1 and h'_2 as follows:

$$h'_1 \stackrel{\text{def}}{=} \widehat{h}_1 + \{(\ell, \ell') \in h' \mid \ell \in P_1 \cup R_1 \cup \{l'_{\text{pre}}\}\}, \quad h'_2 \stackrel{\text{def}}{=} \widehat{h}_2 + \{(\ell, \ell') \in h' \mid \ell \in P_2 \cup R_2\}.$$

From (1), the heaps $\{(\ell, \ell') \in h' \mid \ell \in P_1 \cup R_1 \cup \{l'_{pre}\}\}$ and $\{(\ell, \ell') \in h' \mid \ell \in P_2 \cup R_2\}$ are disjoint, and their union is $\{(\ell, \ell') \in h' \mid \ell \in \text{Path}[s]_{s,h'}^X(t)\}$. Thus, h'_1 and h'_2 are well-defined, they are disjoint, and $h' = h'_1 + h'_2$. We prove the properties (A)–(F) (introduced below), which are analogous to the homonymous properties in Lemma 5.40(III), and thus lead to $(s, h) \leftrightarrow_{X,\alpha}^S (s, h')$. Let $j \in \{1, 2\}$. As done in Lemma 5.40(III), before showing (A)–(F) we introduce a set of auxiliary properties, grouped under the name (O).

- O. (a) $e_{P_j} = l'_{pre}$ if and only if $e_{P'_j} = l'_{pre}$,
- (b) $\text{Path}[s]_{s,h}^X(t) \subseteq \text{dom}(h_j)$ if and only if $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h'_j)$,
- (c) Let ℓ and ℓ' be two locations such that
 - * h_j witnesses a (possibly empty) path going from $s(x)$ to ℓ , for some $x \in X$,
 - * $\ell, \ell' \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$,
 - * h_j witnesses a non-empty path going from ℓ to ℓ' .
 Then, $\ell, \ell' \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, and h'_j witnesses a non-empty path, from ℓ to ℓ' .
- (d) Let ℓ and ℓ' be two locations such that
 - * h'_j witnesses a (possibly empty) path going from $s(x)$ to ℓ , for some $x \in X$,
 - * $\ell, \ell' \notin \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$,
 - * h'_j witnesses a non-empty path going from ℓ to ℓ' .
 Then, $\ell, \ell' \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, and h_j witnesses a non-empty path, from ℓ to ℓ' .

Proof of (a). Directly from from (4) and (5) we derive the following double implication:

$$P_j \neq \emptyset \text{ and } R_{3-j} = \emptyset \text{ if and only if } P'_j \neq \emptyset \text{ and } R'_{3-j} = \emptyset.$$

Thus, in order to show (a), we prove that

$$e_{P'_j} = l'_{pre} \text{ if and only if } P'_j \neq \emptyset \text{ and } R'_{3-j} = \emptyset,$$

and, similarly, that $e_{P_j} = l'_{pre} \in \text{dom}(h_j)$ holds if and only if so do $P_j \neq \emptyset$ and $R_{3-j} = \emptyset$. The proof of the latter double implication is analogous to the former one, and it is left to the reader.

(\Rightarrow): Suppose $e_{P'_j} = l'_{pre}$. By definition, $e_{P'_j}$ is only defined if P'_j is non-empty. In this case, by (3), P'_j describes a path going from $\llbracket t \rrbracket_{s,h'}^X$ to l'_{pre} . As $h(l'_{pre}) = \text{sby}_{s,h'}^X(t)$, this implies that P'_j and $\{l'_{pre}\}$ partition $\text{Path}[s]_{s,h'}^X(t)$, which in turn means that $R'_j = \emptyset$.

(\Leftarrow): Directly from (3).

Proof of (b). Since we are assuming $l'_{pre} \in \text{dom}(h_1)$ and by definition $l'_{pre} \in \text{dom}(h'_1)$, (b) trivially holds for $j = 2$ (both sides of the double implication are false). Let us assume $j = 1$ and, for the left-to-right direction, that $\text{Path}[s]_{s,h}^X(t) \subseteq \text{dom}(h_1)$. Since $h_1 \perp h_2$, $\text{Path}[s]_{s,h}^X(t) \cap \text{dom}(h_2) = \emptyset$ and therefore $R_2 = \emptyset$ and $P_2 = \emptyset$. As $\text{Path}[s]_{s,h}^X(t)$ is non-empty, the latter equality implies $P_1 \neq \emptyset$, which allows us to conclude $e_{P_1} = l'_{pre}$ as discussed in the proof of (a). Thus, $e_{P'_1} = l'_{pre}$. By (3), P'_1 describes a path in h'_1 , going from $\llbracket t \rrbracket_{s,h'}^X$ to l'_{pre} . As moreover $l'_{pre} \in \text{dom}(h'_1)$ and $h'_1(l'_{pre}) = \text{sby}_{s,h'}^X(t)$, we conclude that $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h'_1)$. The right-to-left direction follows analogously.

Before proving (c), we show a fundamental property of locations in $\text{Path}[s]_{s,h}^X(t)$.

- (κ) Consider a location $\ell_1 \in \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, for some memory state (\hat{s}, \hat{h}) . Let ℓ_2 be a location such that $\hat{h}(\ell_2) = \ell_1$ and \hat{h} witnesses a path going from $s(x)$ to ℓ_2 , for some $x \in X$. Then, $\ell_2 \in \text{Path}[s]_{s,h}^X(t)$.

Proof of (κ). Since $\text{Path}[\mathcal{S}]_{s,\widehat{h}}^X(t)$ describes a path in \widehat{h} , going from $\llbracket t \rrbracket_{s,\widehat{h}}^X$ to $\text{sby}_{s,\widehat{h}}^X(t)$, the location $\ell_1 \in \text{Path}[\mathcal{S}]_{s,\widehat{h}}^X(t) \setminus \{\llbracket t \rrbracket_{s,\widehat{h}}^X\}$ is not the first location of this path, i.e. there is $\ell \in \text{Path}[\mathcal{S}]_{s,\widehat{h}}^X(t)$ such that $\widehat{h}(\ell) = \ell_1$. *Ad absurdum*, suppose $\ell \neq \ell_2$. We show that this implies that $\ell_1 \in \text{Lab}[\mathcal{S}]_{s,\widehat{h}}^X$. As $\llbracket t \rrbracket_{s,\widehat{h}}^X$ is the only labelled location in $\text{Path}[\mathcal{S}]_{s,\widehat{h}}^X(t)$, this contradicts the fact that $\ell_1 \in \text{Path}[\mathcal{S}]_{s,\widehat{h}}^X(t) \setminus \{\llbracket t \rrbracket_{s,\widehat{h}}^X\}$. We consider two (possibly syntactically equal) program variables $x, y \in X$ such that t is written using x , and \widehat{h} witnesses a path going from $s(y)$ to ℓ_2 (which exists by hypothesis). Regardless of whether t is a variable, end-point variable or meet-point variable, \widehat{h} witnesses a path going from $s(x)$ to $\llbracket t \rrbracket_{s,\widehat{h}}^X$, and thus from $s(x)$ to ℓ . Now, if ℓ_1 does not belong to a cycle we conclude that it is the first location reachable from both $s(x)$ and $s(y)$, which in turn implies $\ell_1 = \llbracket m(x, y) \rrbracket_{s,\widehat{h}}^X$. Instead, if ℓ_1 does belong to a cycle, we divide the proof depending on whether ℓ belongs to a cycle.

case: ℓ belongs to a cycle. As $\widehat{h}(\ell) = \ell_1$, both ℓ and ℓ_1 belongs to the same cycle.

Since we are assuming $\ell \neq \ell_2$, we conclude that ℓ_2 does not belong to a cycle.

Hence, as $\widehat{h}(\ell_2) = \ell_1$, in \widehat{h} , ℓ_1 is the first location reachable from $s(y)$ that belongs to a cycle. By definition of end-point variables, $\llbracket e(y) \rrbracket_{s,\widehat{h}}^X = \ell_1$.

case: ℓ does not belong to a cycle. Similarly to the previous case, as ℓ_1 belongs to a cycle and $h_1(\ell) = \ell_1$, we conclude that, in \widehat{h} , ℓ_1 is the first location reachable from $s(x)$ that belongs to a cycle. Therefore, $\llbracket e(x) \rrbracket_{s,\widehat{h}}^X = \ell_1$.

Since ℓ_1 cannot be a labelled location, we conclude that $\ell_2 = \ell \in \text{Path}[\mathcal{S}]_{s,\widehat{h}}^X(t)$.

Proof of (c). Let ℓ and ℓ' be two locations such that

- * h_j witnesses a (possibly empty) path going from $s(x)$ to ℓ , for some $x \in X$,
- * $\ell, \ell' \notin \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$,
- * h_j witnesses a non-empty path going from ℓ to ℓ' .

Note that if a heap witnesses a non-empty path going from a location ℓ_1 to a location ℓ_2 , then it witnesses a *minimal* (i.e. the shortest) non-empty path going from ℓ_1 to ℓ_2 . Formally, a non-empty path $(\ell'_0, \dots, \ell'_n)$ is minimal if for every $i \in [1, n - 1]$, $\ell'_i \neq \ell'_n$. So, let δ be the length of the minimal non-empty path in h_j , going from ℓ to ℓ' . We show that $\ell, \ell' \notin \text{Path}[\mathcal{S}]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$, and h'_j witnesses a non-empty path, going from ℓ to ℓ' . The proof is by induction on the length δ , with the natural induction hypothesis stating that the property holds for paths of length less than δ .

base case: $\delta = 1$. So, $h_j(\ell) = \ell'$. Since ℓ' does not belong to $\text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, and $\text{Path}[\mathcal{S}]_{s,h}^X(t)$ describes a minimal path of length at least $\mathcal{S}(\alpha) \geq 2$, going from $\llbracket t \rrbracket_{s,h}^X$ to $\text{sby}_{s,h}^X(t)$, we have $\ell \neq \llbracket t \rrbracket_{s,h}^X$. By $\ell \notin \text{Path}[\mathcal{S}]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, this implies $\ell \notin \text{Path}[\mathcal{S}]_{s,h}^X(t)$. From $\ell \in \text{dom}(h_j)$ and by definition of \widehat{h}_j , we conclude that $\ell \in \text{dom}(\widehat{h}_j)$ and $\widehat{h}_j(\ell) = \ell'$. By $\widehat{h}_j \subseteq h'_j$, $h'_j(\ell) = \ell'$. In order to conclude the proof of the base case, it is sufficient to show that the locations ℓ and ℓ' do not belong to $\text{Path}[\mathcal{S}]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$. The proof is the same for both locations. In what follows, we write $\bar{\ell}$ for either ℓ or ℓ' . *Ad absurdum*, suppose $\bar{\ell} \in \text{Path}[\mathcal{S}]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$, and so $\bar{\ell} \in \text{Path}[\mathcal{S}]_{s,h'}^X(t)$ and $\bar{\ell} \neq \llbracket t \rrbracket_{s,h'}^X = \llbracket t \rrbracket_{s,h}^X$, where the last equivalence holds by $(=t)$. Thus,

$$\bar{\ell} \notin \text{dom}(h' \setminus \{(\ell_1, \ell_2) \in h' \mid \ell_1 \in \text{Path}[\mathcal{S}]_{s,h'}^X(t)\}).$$

From the hypothesis

$h \setminus \{(\ell_1, \ell_2) \in h \mid \ell_1 \in \text{Path}[s]_{s,h}^X(t)\} = h' \setminus \{(\ell_1, \ell_2) \in h' \mid \ell_1 \in \text{Path}[s]_{s,h'}^X(t)\}$ we conclude that $\bar{\ell} \notin \text{dom}(h \setminus \{(\ell_1, \ell_2) \in h \mid \ell_1 \in \text{Path}[s]_{s,h}^X(t)\})$. We divide the proof depending on whether or not $\bar{\ell} \in \text{dom}(h)$. In both cases we reach a contradiction, allowing us to conclude the proof of the base case.

case: $\bar{\ell} \in \text{dom}(h)$. From $\bar{\ell} \notin \text{dom}(h \setminus \{(\ell_1, \ell_2) \in h \mid \ell_1 \in \text{Path}[s]_{s,h}^X(t)\})$ we conclude $\bar{\ell} \in \text{Path}[s]_{s,h}^X(t)$. However, by $\bar{\ell} \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, this implies that $\bar{\ell}$ must be $\llbracket t \rrbracket_{s,h}^X$, a contradiction.

case: $\bar{\ell} \notin \text{dom}(h)$. We recall that h_j witnesses a path going from $s(x)$ to ℓ , for some $x \in X$. As $h_j(\ell) = \ell'$, h_j witnesses a path going from $s(x)$ to ℓ' . By $h_j \subseteq h$, we conclude that h witnesses a path going from $s(x)$ to $\bar{\ell}$. We derive a contradiction with $\bar{\ell} \in \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$. If $s(x) = \bar{\ell}$, then by definition $\bar{\ell} \notin \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$. Indeed, $\llbracket t \rrbracket_{s,h'}^X$ is the only labelled location in $\text{Path}[s]_{s,h'}^X(t)$. Instead, if $s(x) \neq \bar{\ell}$, then there is $\delta'' \geq 1$ such that $h^{\delta''}(s(x)) = \bar{\ell}$. As $\bar{\ell} \notin \text{dom}(h)$, by definition of end-point variables, we derive $\llbracket e(x) \rrbracket_{s,h}^X = \bar{\ell}$. By $(=t)$, $\bar{\ell} = \llbracket e(x) \rrbracket_{s,h'}^X$. Again, $\bar{\ell} \notin \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$.

induction step: $\delta \geq 2$. Let $(\ell_0 = \ell, \dots, \ell_\delta = \ell')$ be the minimal non-empty path going from ℓ to ℓ' , in h_j . We divide the proof depending on whether $\ell_0 = \llbracket t \rrbracket_{s,h}^X$.

case: $\ell_0 = \llbracket t \rrbracket_{s,h}^X$. Since $\ell_\delta \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, we conclude that the path described by $\text{Path}[s]_{s,h}^X(t)$ must be a prefix of the path $(\ell_0, \dots, \ell_\delta)$. In particular, given $i = \text{card}(\text{Path}[s]_{s,h}^X(t)) \leq \delta$, we have $\ell_i = \text{sby}_{s,h}^X(t)$. Moreover, $\text{Path}[s]_{s,h}^X(t) \subseteq \text{dom}(h_j)$, which in turn implies $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h'_j)$, by (b). Thanks to $(=t)$, the set $\text{Path}[s]_{s,h'}^X(t)$ describes a minimal non-empty path ρ in h'_j , going from $\llbracket t \rrbracket_{s,h'}^X = \llbracket t \rrbracket_{s,h}^X = \ell_0$ to $\text{sby}_{s,h'}^X(t) = \text{sby}_{s,h}^X(t) = \ell_i$. The location ℓ_i is labelled and so it cannot belong to $\text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. If $i = \delta$, the proof is complete. Otherwise, as h_j witnesses a path going from $s(x)$ to $\ell_0 = \ell$, for some $x \in X$, it also witnesses a path going from $s(x)$ to ℓ_i . From $i < \delta$, $(\ell_i, \dots, \ell_\delta)$ is non-empty, which allows us to apply the induction hypothesis, and conclude that h'_j witnesses a non-empty path going from ℓ_i to ℓ_δ , and that moreover $\ell_\delta \notin \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$. Together with the path ρ , we conclude that h'_j witnesses a non-empty path going from $\ell = \ell_0$ to $\ell' = \ell_\delta$. Lastly, by $(=t)$, $\ell = \llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$ and so $\ell \notin \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$.

case: $\ell_0 \neq \llbracket t \rrbracket_{s,h}^X$. In this case, $\ell_0 \notin \text{Path}[s]_{s,h}^X(t)$. Let us consider the second location in the path $(\ell_0, \dots, \ell_\delta)$ of h_j , i.e. $\ell_1 = h_j(\ell_0)$. As we are assuming $\delta \geq 2$ and this path to be minimal, $\ell_1 \neq \ell_0$ and path $(\ell_1, \dots, \ell_\delta)$ is the minimal non-empty path in h_j , of length $\delta - 1$ and going from ℓ_1 to ℓ_n . From $\ell_0 \notin \text{Path}[s]_{s,h}^X(t)$ and by definition of \widehat{h}_j , $\ell_0 \in \text{dom}(\widehat{h}_j)$ and $\widehat{h}_j(\ell_0) = \ell_1$. By definition of h'_j , $h'_j(\ell_0) = \ell_1$ and $\ell_0 \notin \text{Path}[s]_{s,h'}^X(t)$. So, h'_j witnesses a non-empty path, from ℓ_0 to ℓ_1 . As $\ell_0 \notin \text{Path}[s]_{s,h}^X(t)$, by (κ) , either $\ell_1 \notin \text{Path}[s]_{s,h}^X(t)$ or $\ell_1 = \llbracket t \rrbracket_{s,h}^X$, which allows us to conclude that $\ell_1 \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. Moreover, as h_j witnesses a path going from $s(x)$ to $\ell_0 = \ell$, for some $x \in X$, it also witnesses a path going from $s(x)$ to ℓ_1 . Relying on the path $(\ell_1, \dots, \ell_\delta)$, we apply the induction hypothesis, and conclude that h'_j witnesses minimal non-empty path $(\ell'_1, \dots, \ell'_m)$, where $\ell'_1 = \ell_1$ and $\ell'_m = \ell_\delta$. Moreover, $\ell_\delta \notin \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$. Together with $h'_j(\ell_0) = \ell_1$, we conclude that $(\ell_0, \ell'_1, \dots, \ell'_m)$ is a non-empty path in h'_j , going from $\ell_0 = \ell$ to $\ell_\delta = \ell'$.

Proof of (d). This proof carries out as the one for the property (c). Everything is symmetric. The base case of the induction relies on the equivalences in ($=_t$) and

$$h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Path}[s]_{s,h}^X(t)\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Path}[s]_{s,h'}^X(t)\}.$$

For the induction step, we rely on the symmetrical property (b) (used in the first case of the induction step), and on the fact that $\widehat{h_j}$ is a subheap of both h_j and h'_j (used in the second case of the induction step).

We now move to the properties (A)–(F), whose proofs heavily rely on (O).

- A. For every $t' \in T[s]^X$, $\llbracket t' \rrbracket_{s,h_j}^X$ and $\llbracket t' \rrbracket_{s,h'_j}^X$ are equidefined. When they are defined, either
 - $\llbracket t' \rrbracket_{s,h_j}^X = \llbracket t' \rrbracket_{s,h'_j}^X \notin (\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, or
 - $\llbracket t' \rrbracket_{s,h_j}^X = e_{P_j} \notin \text{dom}(h_j)$ and $\llbracket t' \rrbracket_{s,h'_j}^X = e_{P'_j} \notin \text{dom}(h'_j)$.

Proof of (A). The property trivially holds when t' is a program variable. Below, we split the proof depending on whether $t' = m(x, y)$ or $t' = e(x)$, for some $x, y \in X$.

case: $t' = m(x, y)$. We show that $\llbracket m(x, y) \rrbracket_{s,h_j}^X$ is defined if and only if so is $\llbracket m(x, y) \rrbracket_{s,h'_j}^X$.

Moreover, we show that, whenever defined,

$$\llbracket m(x, y) \rrbracket_{s,h_j}^X = \llbracket m(x, y) \rrbracket_{s,h'_j}^X \notin (\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}.$$

(\Rightarrow): Let us assume $\llbracket m(x, y) \rrbracket_{s,h_j}^X = \ell_m$, and show that $\ell_m = \llbracket m(x, y) \rrbracket_{s,h'_j}^X$ and $\ell_m \notin (\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. By definition, there are $\delta_1, \delta_2 \geq 1$ such that $h_j^{\delta_1}(s(x)) = h_j^{\delta_2}(s(y)) = \ell_m$ and for all $\delta'_1 \in [0, \delta_1]$, $\delta'_2 \in [0, \delta_2]$, if $\delta'_1 + \delta'_2 < \delta_1 + \delta_2$ then $h_j^{\delta'_1}(s(x)) \neq h_j^{\delta'_2}(s(y))$. Moreover, for every $\delta' \geq 1$, it holds that $h^{\delta'}(\ell) \neq \ell_m$. Informally, this means that h_j witnesses two non-empty disjoint paths, one going from $s(x)$ to ℓ_m and one going from $s(y)$ to ℓ_m , where ℓ_m is a location that does not belong to a cycle. Specifically, there must be two locations ℓ_x and ℓ_y such that $\ell_x \neq \ell_y$, $h_j(\ell_x) = h_j(\ell_y) = \ell_m$, and h_j witnesses two (possibly empty) disjoint paths, one going from $s(x)$ to ℓ_x and one going from $s(y)$ to ℓ_y . First of all, we show that $\ell_m \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. Ad absurdum, suppose $\ell_m \in \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. We can apply (κ) and conclude that both ℓ_x and ℓ_y belong to $\text{Path}[s]_{s,h}^X(t)$. However, as $\text{Path}[s]_{s,h}^X(t)$ describes a path in h and $h(\ell_x) = h(\ell_y) = \ell_m$ (by $h_j \subseteq h$), this implies $\ell_x = \ell_y$: a contradiction. Thus, $\ell_m \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. By definition, both $s(x)$ and $s(y)$ do not belong to $\text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, which allows us to apply (c) (twice) and conclude that h'_j witnesses two non-empty paths ρ_x and ρ_y , one going from $s(x)$ to ℓ_m and the other going from $s(y)$ to ℓ_m , respectively. Moreover, $\ell_m \notin \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$. As $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$ (by ($=_t$)), this allows us to conclude that $\ell_m \notin (\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. Afterwards, the fact that $\ell_m = \llbracket m(x, y) \rrbracket_{s,h'_j}^X$ follows directly from the definition of meet-point variables together with the following two statements:

- * ℓ_m does not belong to a cycle in h'_j .

Ad absurdum, suppose h'_j witnesses a non-empty path going from ℓ_m to itself. Recall that ℓ_m does not belong to $\text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$ and it is reached, in h'_j , by $s(x)$. We apply (d) and conclude that h_j witnesses a non-empty path going from ℓ_m to itself. This contradicts $\ell_m = \llbracket m(x, y) \rrbracket_{s,h_j}^X$.

- * the paths ρ_x and ρ_y are disjoint.

Ad absurdum, suppose that, in h'_j , the two paths ρ_x and ρ_y are not disjoint. Since we just showed that ℓ_m does not belong to a cycle, we know that $\llbracket m(x, y) \rrbracket_{s, h'_j}^X$ is defined. Notice that this also means that ρ_x (resp. ρ_y) is the only path in h'_j , going from $s(x)$ (resp. $s(y)$) to ℓ_m . So, h'_j witnesses three disjoint non-empty paths: one from $s(x)$ to $\llbracket m(x, y) \rrbracket_{s, h'_j}^X$, one from $s(y)$ to $\llbracket m(x, y) \rrbracket_{s, h'_j}^X$, and one from $\llbracket m(x, y) \rrbracket_{s, h'_j}^X$ to ℓ_m . Exactly as done for ℓ_m , we rely on (κ) to derive that $\llbracket m(x, y) \rrbracket_{s, h'_j}^X \notin \text{Path}[S]_{s, h'}^X(t) \setminus \{\llbracket t \rrbracket_{s, h'}^X\}$. By (d), this allows us to conclude that h_j witnesses three non-empty paths, one from $s(x)$ to $\llbracket m(x, y) \rrbracket_{s, h'_j}^X$, one from $s(y)$ to $\llbracket m(x, y) \rrbracket_{s, h'_j}^X$, and one from $\llbracket m(x, y) \rrbracket_{s, h'_j}^X$ to ℓ_m . However, this contradicts $\ell_m = \llbracket m(x, y) \rrbracket_{s, h_j}^X$.

(\Leftarrow): Symmetrical to the other direction.

Before considering the case of $t' = e(x)$, we discuss a property of end-point variables.

(λ) Let (\hat{s}, \hat{h}) be a memory state and let $x \in X$. Consider a subheap $\tilde{h} \subseteq \hat{h}$. If $\llbracket e(x) \rrbracket_{s, \tilde{h}}^X$ is defined and belongs to $\text{dom}(\tilde{h})$, then $\llbracket e(x) \rrbracket_{s, \hat{h}}^X = \llbracket e(x) \rrbracket_{s, \tilde{h}}^X$.

Proof of (λ). Suppose $\llbracket e(x) \rrbracket_{s, \tilde{h}}^X$ defined and equal to $\ell \in \text{dom}(\tilde{h})$. By definition of end-point variables, there is $\delta \geq 1$ such that $\tilde{h}^\delta(\hat{s}(x)) = \ell$ and ℓ belongs to a cycle in \tilde{h} whereas $\tilde{h}^{\delta-1}(\hat{s}(x))$ does not belong to a cycle. Let L be a minimal set of locations describing the cycle in \tilde{h} involving ℓ . In particular, $\ell \in L$ whereas $\tilde{h}^{\delta-1}(\hat{s}(x)) \notin L$. As $\tilde{h} \subseteq \hat{h}$, $\tilde{h}^\delta(\hat{s}(x)) = \ell$, and $\ell \in L$, where L is again a minimal set of locations describing a cycle in \hat{h} . Clearly, $\tilde{h}^{\delta-1}(\hat{s}(x))$ cannot belong to a cycle in \hat{h} , since otherwise it would belong to the same cycle of ℓ and we would conclude that $\tilde{h}^{\delta-1}(\hat{s}(x)) \in L$, a contradiction. By definition of end-point variables, $\ell = \llbracket e(x) \rrbracket_{s, \hat{h}}^X$.

We are now ready to complete the proof of (A).

case: $t' = e(x)$. We show that $\llbracket e(x) \rrbracket_{s, h_j}^X$ is defined if and only if so is $\llbracket e(x) \rrbracket_{s, h'_j}^X$. If defined, either $\llbracket e(x) \rrbracket_{s, h_j}^X = \llbracket e(x) \rrbracket_{s, h'_j}^X \notin (\text{Path}[S]_{s, h}^X(t) \cup \text{Path}[S]_{s, h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s, h}^X\}$ or $(\llbracket e(x) \rrbracket_{s, h_j}^X = e_{P_j} \notin \text{dom}(h_j) \text{ and } \llbracket e(x) \rrbracket_{s, h'_j}^X = e_{P'_j} \notin \text{dom}(h'_j))$.

(\Rightarrow): Suppose $\llbracket e(x) \rrbracket_{s, h_j}^X = \ell_e$. We divide the proof in two cases, depending on whether or not $\ell_e = e_{P_j}$.

case: $\ell_e = e_{P_j}$. Following its definition, $P_j \neq \emptyset$ and it describes a path in h_j going from $\llbracket t \rrbracket_{s, h}^X$ to $\ell_e \in R_{3-j} \cup \{l_{pre}\}$. In particular, this means that $\llbracket t \rrbracket_{s, h}^X \in P_j$ and thus $\llbracket t \rrbracket_{s, h}^X \neq \ell_e$ (recall that P_j , R_{3-j} and $\{l_{pre}\}$ are mutually disjoint). This allows us to conclude that $\ell_e \notin \text{dom}(h_j)$. Indeed, suppose *ad absurdum* that $\ell_e \in \text{dom}(h_j)$. By (λ), we conclude that $\ell_e = \llbracket e(x) \rrbracket_{s, h}^X$. However, this implies that $\ell_e \in \text{Lab}[S]_{s, h}^X$, in contradiction with $\llbracket t \rrbracket_{s, h}^X \neq \ell_e$. Indeed, we recall that $\llbracket t \rrbracket_{s, h}^X$ is the only labelled location of $\text{Path}[S]_{s, h}^X(t)$. Thus, $\ell_e \notin \text{dom}(h_j)$. As ℓ_e corresponds to the end-point variable $e(x)$, there is $\delta \geq 1$ such that $h_j^\delta(s(x)) = \ell_e$. Then, h_j witnesses a (possibly empty) path ρ going from $s(x)$ to $\llbracket t \rrbracket_{s, h}^X$. Indeed, the opposite leads to a contradiction, as it implies that there is a location ℓ' reached by $s(x)$ such that $h_j(\ell') = \ell_e$ and $\ell' \notin \text{Path}[S]_{s, h}^X(t)$, which is impossible by (κ). Afterwards, the fact that $\llbracket e(x) \rrbracket_{s, h'_j}^X = e_{P'_j}$ follows directly from the definition of end-point variables together with the following three statements:

- * h'_j witnesses a path going from $s(x)$ to $\llbracket t \rrbracket_{s,h'}^x$.

If $s(x) = \llbracket t \rrbracket_{s,h}^x$ then by (= t) $s(x) = \llbracket t \rrbracket_{s,h'}^x$ and the result trivially holds. Otherwise, the path ρ in h_j is non-empty, and since both $s(x)$ and $\llbracket t \rrbracket_{s,h}^x$ do not belong to $\text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, the result holds directly from (c).

- * h'_j witnesses a path going from $\llbracket t \rrbracket_{s,h'}^x$ to $e_{P'_j}$.

Directly from the properties (4) and (3) of the construction, as $P_j \neq \emptyset$.

- * $e_{P'_j} \notin \text{dom}(h'_j)$.

We know that $e_{P_j} \in R_{3-j} \cup \{l_{pre}\}$ and $e_{P_j} \notin \text{dom}(h_j)$. We remind the reader that we are assuming $l_{pre} \in \text{dom}(h_1)$ and so, if $e_{P_j} = l_{pre}$, then $j = 2$. Moreover, by definition of h'_1 , $l'_{pre} \in \text{dom}(h'_1)$. *Ad absurdum*, suppose $e_{P'_j} \in \text{dom}(h'_j)$. From the property (3) of the construction, $e_{P'_j} \in R_{3-j} \cup \{l'_{pre}\}$. Since the locations in R_{3-j} belongs to $\text{dom}(h_{3-j})$, we conclude that $e_{P'_j} = l'_{pre}$. However, as $l'_{pre} \in \text{dom}(h'_j)$, this implies $j = 1$. This is contradictory, as $e_{P'_j} = l'_{pre}$ implies, by (a), $e_{P_j} = l_{pre}$ and so $j = 2$. Thus, $e_{P'_j} \notin \text{dom}(h'_j)$.

case: $\ell_e \neq e_{P_j}$. As ℓ_e corresponds to the end-point variable $e(x)$, there is $\delta \geq 1$ such that $h_j^\delta(s(x)) = \ell_e$, and if $\ell_e \in \text{dom}(h_j)$ then ℓ_e belongs to a cycle in h_j whereas $h_j^{\delta-1}(s(x))$ does not. First, we show that $\ell_e \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. *Ad absurdum*, suppose $\ell_e \in \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. In particular, this means that $\ell_e \notin \text{Lab}[s]_{s,h}^x$. If $\ell_e \in \text{dom}(h_j)$ then contradiction is direct: by (λ) we have $\ell_e = \llbracket e(x) \rrbracket_{s,h}^x \in \text{Lab}[s]_{s,h}^x$. Otherwise, suppose $\ell_e \notin \text{dom}(h_j)$. As already noticed in the previous case of the proof, the fact that $h_j^\delta(s(x)) = \ell_e$ and $\ell_e \in \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$ hold imply, by (κ), that h_j witnesses a path ρ going from $s(x)$ to $\llbracket t \rrbracket_{s,h}^x$. Since $\ell_e \notin \text{dom}(h_j)$, we conclude that h_j witnesses a path going from $\llbracket t \rrbracket_{s,h}^x$ to ℓ_e . However, this implies $\ell_e = e_{P_j}$, by definition of P_j . Thus, $\ell_e \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. Since $s(x) \in \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, we apply (c) and conclude that h'_j witnesses a non-empty path ρ going from $s(x)$ to ℓ_e , and $\ell_e \notin \text{Path}[s]_{s,h'}^x(t) \setminus \{\llbracket t \rrbracket_{s,h'}^x\}$. As $\llbracket t \rrbracket_{s,h}^x = \llbracket t \rrbracket_{s,h'}^x$ (by (= t)), this allows us to conclude that $\ell_e \notin (\text{Path}[s]_{s,h}^x(t) \cup \text{Path}[s]_{s,h'}^x(t)) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. In order to show that $\ell_e = \llbracket e(x) \rrbracket_{s,h'_j}^x$, we divide the proof depending on whether or not $\ell_e \in \text{dom}(h_j)$.

case: $\ell_e \in \text{dom}(h_j)$. From (λ), we have $\ell_e = \llbracket e(x) \rrbracket_{s,h_j}^x = \llbracket e(x) \rrbracket_{s,h}^x$. By (= t), $\llbracket e(x) \rrbracket_{s,h}^x = \llbracket e(x) \rrbracket_{s,h'}^x$, and to conclude the proof it is sufficient to show that $\llbracket e(x) \rrbracket_{s,h'}^x = \llbracket e(x) \rrbracket_{s,h'_j}^x$. By definition of end-point variable, h_j witnesses a non-empty path going from ℓ_e to itself. By (c), h'_j witnesses a non-empty path going from ℓ_e to itself. Together with the path ρ , this implies that $\llbracket e(x) \rrbracket_{s,h'_j}^x$ is defined and in $\text{dom}(h'_j)$. By (λ), $\llbracket e(x) \rrbracket_{s,h'}^x = \llbracket e(x) \rrbracket_{s,h'_j}^x$.

case: $\ell_e \notin \text{dom}(h_j)$. First, suppose $\ell_e = \llbracket t \rrbracket_{s,h}^x$. As $\ell_e \notin \text{dom}(h_j)$, this implies $P_j = \emptyset$. From the property (4) of the construction, $P'_j = \emptyset$, which implies $\llbracket t \rrbracket_{s,h'}^x \notin \text{dom}(h'_j)$. By (= t), $\llbracket t \rrbracket_{s,h'}^x = \ell_e$, which implies $\ell_e = \llbracket e(x) \rrbracket_{s,h'_j}^x$, thanks to the path ρ . Lastly, suppose $\ell_e \neq \llbracket t \rrbracket_{s,h}^x$, and so $\ell_e \notin \text{Path}[s]_{s,h}^x(t)$. *Ad absurdum*, suppose $\ell_e \in \text{dom}(h'_j)$. From $\ell_e \notin \text{dom}(h_j)$ and $\widehat{h_j} \subseteq h_j$ we conclude $\ell_e \notin \text{dom}(\widehat{h_j})$. By definition of h'_j , we derive $\ell_e \in \text{Path}[s]_{s,h'}^x(t)$. However, as $\ell_e \notin \text{Path}[s]_{s,h'}^x(t) \setminus \{\llbracket t \rrbracket_{s,h'}^x\}$, this implies $\ell_e = \llbracket t \rrbracket_{s,h'}^x$. This

is contradictory, as it allows us to derive $\ell_e = \llbracket t \rrbracket_{s,h}^X$, by (= t). Therefore, $\ell_e \notin \text{dom}(h'_j)$. Thanks to the path ρ , this implies $\ell_e = \llbracket e(x) \rrbracket_{s,h'_j}^X$.

(\Leftarrow): Symmetrical to the other direction.

B. For every $t' \in T[S]^X$,

- (e) for every $t'' \in T[S]^X$, $\llbracket t'' \rrbracket_{s,h_j}^X = \text{sby}_{s,h_j}^X(t')$ iff $\llbracket t'' \rrbracket_{s,h'_j}^X = \text{sby}_{s,h'_j}^X(t')$,
- (f) $\min(\text{card}(\text{Path}[S]_{s,h_j}^X(t')), \mathcal{S}(\alpha_j)) = \min(\text{card}(\text{Path}[S]_{s,h_j}^X(t')), \mathcal{S}(\alpha_j))$,
- (g) If $h_j^{\delta_1}(\llbracket t' \rrbracket_{s,h_j}^X) = s(u)$ for some $\delta_1 \in [0, \text{card}(\text{Path}[S]_{s,h_j}^X(t'))]$, then $h_j^{\delta_2}(\llbracket t' \rrbracket_{s,h'_j}^X) = s(u)$ for some $\delta_2 \in [0, \text{card}(\text{Path}[S]_{s,h'_j}^X(t'))]$ s.t. $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j))$ and $\min(\text{card}(\text{Path}[S]_{s,h_j}^X(t')) - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(\text{Path}[S]_{s,h'_j}^X(t')) - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j))$.
- (h) If $h_j^{\delta_2}(\llbracket t' \rrbracket_{s,h'_j}^X) = s(u)$ for some $\delta_2 \in [0, \text{card}(\text{Path}[S]_{s,h'_j}^X(t'))]$, then $h_j^{\delta_1}(\llbracket t' \rrbracket_{s,h_j}^X) = s(u)$ for some $\delta_1 \in [0, \text{card}(\text{Path}[S]_{s,h_j}^X(t'))]$ s.t. $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j))$ and $\min(\text{card}(\text{Path}[S]_{s,h_j}^X(t')) - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(\text{Path}[S]_{s,h'_j}^X(t')) - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j))$.

Before tackling the proofs of (e)–(h), let us discuss an easy intermediate result.

- (i) $\llbracket t' \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$ if and only if $\llbracket t' \rrbracket_{s,h'_j}^X \in \text{dom}(h'_j)$.

Proof of (i). (\Rightarrow): We divide the proof depending on whether $\llbracket t' \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h}^X$ holds.

- case:** $\llbracket t' \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h}^X$. Since $\llbracket t' \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$, we have $P_j \neq \emptyset$. From the properties (4) and (3) of the construction, $P'_j \neq \emptyset$ and $\llbracket t \rrbracket_{s,h'}^X \in \text{dom}(h'_j)$. By (= t), $\llbracket t \rrbracket_{s,h'}^X = \llbracket t \rrbracket_{s,h}^X = \llbracket t' \rrbracket_{s,h_j}^X$. By (A), $\llbracket t' \rrbracket_{s,h_j}^X = \llbracket t' \rrbracket_{s,h'_j}^X$. So, $\llbracket t' \rrbracket_{s,h'_j}^X \in \text{dom}(h'_j)$.
- case:** $\llbracket t' \rrbracket_{s,h_j}^X \neq \llbracket t \rrbracket_{s,h}^X$. As $\llbracket t' \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$, By (A), $\llbracket t' \rrbracket_{s,h'_j}^X = \llbracket t' \rrbracket_{s,h_j}^X$ and moreover $\llbracket t' \rrbracket_{s,h_j}^X \notin \text{Path}[S]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. Thus, $\llbracket t' \rrbracket_{s,h_j}^X \notin \text{Path}[S]_{s,h}^X(t)$. By definition of \widehat{h}_j , $\llbracket t' \rrbracket_{s,h_j}^X \in \widehat{h}_j$. As $\widehat{h}_j \subseteq h'_j$, $\llbracket t' \rrbracket_{s,h'_j}^X = \llbracket t' \rrbracket_{s,h_j}^X \in \widehat{h}_j$.

(\Leftarrow): Symmetrical to the other direction.

Thanks to the various intermediate results we established up to this point, the statements (e)–(h) can be proved in unison, with the construction we describe below. Additionally, this construction allows us to show the following statements, which are helpful in later stages of the proof (see (F)):

- (j) $R_j \cap \text{Path}[S]_{s,h_j}^X(t') = \emptyset$ and $R'_j \cap \text{Path}[S]_{s,h'_j}^X(t') = \emptyset$.
- (k) $l_{\text{pre}} \in \text{Path}[S]_{s,h_j}^X(t')$ if and only if $l'_{\text{pre}} \in \text{Path}[S]_{s,h'_j}^X(t')$.
- (l) $P_j \cap \text{Path}[S]_{s,h_j}^X(t')$ is either \emptyset or P_j . Similarly, $P'_j \cap \text{Path}[S]_{s,h'_j}^X(t')$ is either \emptyset or P'_j . Lastly, $P_j \subseteq \text{Path}[S]_{s,h_j}^X(t')$ if and only if $P'_j \subseteq \text{Path}[S]_{s,h'_j}^X(t')$.

In order to show (e)–(h) and (j)–(l), we divide the proof depending on whether or not $\llbracket t' \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$ (and so $\llbracket t' \rrbracket_{s,h'_j}^X \in \text{dom}(h'_j)$, thanks to (i)) hold.

case: $\llbracket t' \rrbracket_{s,h_j}^X \notin \text{dom}(h_j)$. Notice that, in this case, $\text{Path}[S]_{s,h_j}^X(t') = \text{Path}[S]_{s,h'_j}^X(t') = \emptyset$.

In this case, the proof is straightforward. Both $\text{sby}_{s,h_j}^X(t')$ and $\text{sby}_{s,h'_j}^X(t')$ are not defined, and so (e) trivially holds. The statements (f) and (j)–(l) follow directly from $\text{Path}[S]_{s,h_j}^X(t') = \text{Path}[S]_{s,h'_j}^X(t') = \emptyset$. The statements (g) and (h) collapse to

$$\llbracket t' \rrbracket_{s,h_j}^X = s(u) \text{ if and only if } \llbracket t' \rrbracket_{s,h'_j}^X = s(u).$$

If $\llbracket t' \rrbracket_{s,h_j}^X \neq e_{P_j}$, this double implication holds directly from (A). Otherwise, $\llbracket t' \rrbracket_{s,h_j}^X = e_{P_j}$ implies, $\llbracket t' \rrbracket_{s,h'_j}^X = e_{P'_j}$, by (A). The left-to-right direction of the double implication holds by (7), whereas the right-to-left direction holds by (8).

case: $\llbracket t' \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$. In this case, both $\text{Path}[s]_{s,h_j}^X(t')$ and $\text{Path}[s]_{s,h'_j}^X(t')$ are non-empty, and thus they describe two non-empty paths, say $\rho = (\ell_0, \dots, \ell_n)$ in h_j and $\rho' = (\ell'_0, \dots, \ell'_m)$ in h'_j , respectively. Here, $n = \text{card}(\text{Path}[s]_{s,h_j}^X(t'))$ whereas $m = \text{card}(\text{Path}[s]_{s,h'_j}^X(t'))$. These two paths are such that

$$\ell_0 = \llbracket t' \rrbracket_{s,h_j}^X, \quad \ell_n = \text{sby}_{s,h_j}^X(t'), \quad \ell'_0 = \llbracket t' \rrbracket_{s,h'_j}^X, \quad \ell'_m = \text{sby}_{s,h'_j}^X(t').$$

Moreover, ρ is the minimal non-empty path in h_j , going from ℓ_0 to ℓ_n , whereas ρ' is the minimal non-empty path in h'_j , going from ℓ'_0 to ℓ'_m . As $\llbracket t' \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$, from (A), $\ell_0 = \llbracket t' \rrbracket_{s,h_j}^X = \llbracket t' \rrbracket_{s,h'_j}^X = \ell'_0 \notin (\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$.

In order to prove (e)–(h), the idea is to first notice that there is a maximal prefix of the path ρ , say $\rho'' = (\ell_0, \dots, \ell_k)$ (for some $k \in [0, n]$), such that for every $i \in [0, k]$, $\ell_i = \ell'_i$ and, if $i < k$, then $\ell_i \in \text{dom}(\widehat{h_j})$. Informally, ρ'' is the maximal prefix of both ρ and ρ' that does not involve elements in $\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)$, except maybe for ℓ_k . One possibility is for ρ'' to be exactly ρ and ρ' . Otherwise, ρ'' is a strict prefix of the two paths. Then, $\ell_k = \llbracket t \rrbracket_{s,h}^X$, and ρ and ρ' can diverge on the locations ℓ_{k+1} and ℓ'_{k+1} , which belong to $\text{Path}[s]_{s,h}^X(t)$ and $\text{Path}[s]_{s,h'}^X(t)$, respectively. This case leads to the following two possibilities:

- $\ell_n = e_{P_j}$ and $\ell'_m = e_{P'_j}$, or
- $\ell_{k_1} = \text{sby}_{s,h}^X(t)$ and $\ell'_{k_2} = \text{sby}_{s,h'}^X(t)$, where $k_1 \in [k+1, n]$ and $k_2 \in [k+1, m]$.

In this case, the suffix $(\ell_{k_1}, \dots, \ell_n)$ of ρ equals the suffix $(\ell'_{k_2}, \dots, \ell'_m)$ of ρ' .

The three possibilities we just described are depicted in Figure 5.17. As we will see, once this analysis is properly formalised, the statements (e)–(h) follow rather easily, by relying on the properties of the construction (in particular, (4), (7) and (8)). We start the analysis on ρ and ρ' by proving a statement that helps us characterise ρ'' .

- (ξ) Let $k \in [0, n]$. If for every $i < k$ we have $\ell_i \neq \llbracket t \rrbracket_{s,h}^X$, then $\ell_k = \ell'_k$ and ℓ_k does not belong to $(\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$.

The proof is by induction on k .

base case: $k = 0$. we already showed that

$$\ell_0 = \ell'_0 \notin (\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}.$$

induction step: $k \geq 1$. Together with $\ell_{k-1} \neq \llbracket t \rrbracket_{s,h}^X$, by induction hypothesis, $\ell_{k-1} = \ell'_{k-1}$ and $\ell_{k-1} \notin \text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)$. By definition of $\widehat{h_j}$, we conclude $\widehat{h_j}(\ell_{k-1}) = \ell_k$. From $\widehat{h_j} \subseteq h'_j$, $h'_j(\ell_{k-1}) = \ell_k$. By the definition of ρ' , this implies $\ell_k = \ell'_k$. Let us show that ℓ_k does not belong to $(\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. *Ad absurdum*, suppose that either $\ell_k \in \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$ or $\ell_k \in \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$ hold (recall that $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$, by (=t)). Consider a program variable $x \in X$ used to write the term t' . By definition, h_j (resp. h'_j) witnesses a path going from $s(x)$ to $\ell_{k-1} = \ell'_{k-1}$. Since $h_j(\ell_{k-1}) = \ell_k$ and $h'_j(\ell_{k-1}) = \ell_k$, we can apply (κ). However, this is contradictory, as it allows us to derive $\ell_{k-1} \in \text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)$. Therefore, ℓ_k does not belong to $(\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$.

Pattern:	Condition:
$\rho : \ell_0 \xrightarrow{\quad} \ell_n$ $\rho' : \ell'_0 \xrightarrow{\quad} \ell'_m$	For every $k \in [0, n - 1]$, $\ell_k \neq \llbracket t \rrbracket_{s,h}^X$
$\rho : \ell_0 \xrightarrow{\quad} \ell_k \xrightarrow{\quad} \ell_n$ $\rho' : \ell'_0 \xrightarrow{\quad} \ell'_k \xrightarrow{\quad} \ell'_m$	There is $k \in [0, n - 1]$ such that $\ell_k = \llbracket t \rrbracket_{s,h}^X$. Moreover, $\ell_n = e_{P_j}$ and $\ell'_m = e_{P'_j}$.
$\rho : \ell_0 \xrightarrow{\quad} \ell_k \xrightarrow{\quad} \ell_{k_1} \xrightarrow{\quad} \ell_n$ $\rho' : \ell'_0 \xrightarrow{\quad} \ell'_{k_2} \xrightarrow{\quad} \ell'_m$	There is $k \in [0, n - 1]$ such that $\ell_k = \llbracket t \rrbracket_{s,h}^X$. Moreover, $\ell_{k_1} = \text{sb}_{s,h}^X(t)$ and $\ell'_{k_2} = \text{sb}_{s,h'}^X(t)$.

: $\text{Path}[s]_{s,h}^X(t)$: $\text{Path}[s]_{s,h'}^X(t)$

Figure 5.17: Possible relations between ρ and ρ' .

Let us continue the analysis of ρ and ρ' . Following Figure 5.17, we divide the proof depending on whether there is $k \in [0, n - 1]$ such that $\ell_k = \llbracket t \rrbracket_{s,h}^X$.

case: $\ell_k \neq \llbracket t \rrbracket_{s,h}^X$, **for every** $k \in [0, n - 1]$. This case corresponds to the first pattern in Figure 5.17. Recall that $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$, by $(=_{\text{t}})$. Thanks to (ξ) , we have

for all k in $[0, n - 1]$, $\ell'_k = \ell_k \notin \text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)$.

By definition of ρ and ρ' , this implies that both the sets $\text{Path}[s]_{s,h_j}^X(t') \cap \text{Path}[s]_{s,h}^X(t)$ and $\text{Path}[s]_{s,h_j}^X(t') \cap \text{Path}[s]_{s,h'}^X(t)$ are empty. (ξ) also implies

$$\ell_n = \ell'_n \notin (\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}.$$

Thus, $\rho = \rho'$.

Proof of (e). Since $\ell_n = \ell'_n \notin (\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, we conclude that $\ell_n \neq e_{P_j}$ and $\ell'_n \neq e_{P'_j}$. Then, (e) follows from (A).

Proof of (f). By $\rho = \rho'$, $\text{card}(\text{Path}[s]_{s,h_j}^X(t')) = \text{card}(\text{Path}[s]_{s,h'_j}^X(t'))$.

Proof of (g) and (h). Again by $\rho = \rho'$, for every $i \in [0, n]$ (where $n = m$), $\ell_i = s(u)$ if and only if $\ell'_i = s(u)$. So, if $s(u)$ belongs to either $\text{Path}[s]_{s,h_j}^X(t')$ or $\text{Path}[s]_{s,h'_j}^X(t')$, then it appears in the same position of the two paths ρ and ρ' described by these sets. This property generalises (g) and (h).

Proof of (j)–(l). All three statements follow from $\text{Path}[s]_{s,h_j}^X(t') \cap \text{Path}[s]_{s,h}^X(t) = \emptyset$ and $\text{Path}[s]_{s,h'_j}^X(t') \cap \text{Path}[s]_{s,h'}^X(t) = \emptyset$.

case: $\ell_k = \llbracket t \rrbracket_{s,h}^X$, **for some** $k \in [0, n - 1]$. We notice that, as ρ is a minimal path going from ℓ_0 to ℓ_n , it cannot be that there is $i \in [0, n - 1]$ such that $i \neq k$ and $\ell_i = \llbracket t \rrbracket_{s,h}^X$, i.e. k is the only position in $[0, n - 1]$ such that $\ell_k = \llbracket t \rrbracket_{s,h}^X$. We write ρ''

for the path (ℓ_0, \dots, ℓ_k) . For every $i \in [0, k-1]$, we have that $\ell_i \neq \llbracket t \rrbracket_{s,h}^X$. By (ξ), this implies that for every $u \in [0, k]$, $\ell_u = \ell'_u$. Equivalently, $\rho'' = (\ell'_0, \dots, \ell'_k)$. Moreover, by (=t), $\ell'_k = \llbracket t \rrbracket_{s,h'}^X$, whereas, for all $i \in [0, k-1]$, $\ell'_i \notin \text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)$ (with ξ). Notice that this means that ρ'' is described by the set $\{\ell_0, \dots, \ell_{k-1}\} = \{\ell'_0, \dots, \ell'_{k-1}\}$, which does not contain elements in $\text{Path}[s]_{s,h}^X(t) \cup \text{Path}[s]_{s,h'}^X(t)$. We divide the proof in two cases, depending on whether $\ell_n = e_{P_j}$.

case: $\ell_n = e_{P_j}$. In this case, ρ can be split into two paths: its prefix ρ'' that goes from ℓ_0 to $\ell_k = \llbracket t \rrbracket_{s,h}^X$, and the path described by P_j , that goes from ℓ_k to e_{P_j} . In particular, $n = k + \text{card}(P_j)$. This pattern is depicted in the second case of Figure 5.17. Furthermore, we notice that, by (A), $e_{P_j} \notin \text{dom}(h_j)$. Since $e_{P_j} \in \text{dom}(h)$, this implies that $e_{P_j} \in \text{dom}(h_{3-j})$. We show that $\ell_m = e_{P'_j}$. *Ad absurdum*, suppose $\ell_m \neq e_{P'_j}$. Since $\ell_m = \text{sby}_{s,h'_j}^X(t') \in \text{Lab}[s]_{s,h'_j}^X$, from (A) we conclude that $\ell_m \notin \text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$ (recall that $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$, by (=t)). Since we are assuming $\ell_k = \llbracket t \rrbracket_{s,h}^X$, for some $k \in [0, n-1]$, this implies that the path described by $\text{Path}[s]_{s,h'}^X(t)$ must be included in ρ' , i.e. $m \geq \text{card}(\text{Path}[s]_{s,h'}^X(t)) + k$. So $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h'_j)$. However, this implies $\text{Path}[s]_{s,h}^X(t) \subseteq \text{dom}(h_j)$ directly by (b), in contradiction with $e_{P_j} \in \text{dom}(h_{3-j})$. Therefore, $\ell_m = e_{P'_j}$. Following the property (3) of the construction, ρ' can be split into two paths: its prefix ρ'' that goes from ℓ_0 to $\ell_k = \llbracket t \rrbracket_{s,h'}^X$, and the path described by P'_j , that goes from ℓ_k to $e_{P'_j}$. In particular $m = k + \text{card}(P'_j)$.

Proof of (e). Follows by (A), as $\text{sby}_{s,h_j}^X(t') = \ell_n = e_{P_j}$ and $\text{sby}_{s,h'_j}^X(t') = \ell_m = e_{P'_j}$.

Below, we recall that $n = \text{card}(\text{Path}[s]_{s,h_j}^X(t'))$ and $m = \text{card}(\text{Path}[s]_{s,h'_j}^X(t'))$.

Proof of (f). From $n = k + \text{card}(P_j)$ and $m = k + \text{card}(P'_j)$ we have,

$$\text{card}(\text{Path}[s]_{s,h_j}^X(t')) = k + \text{card}(P_j), \quad \text{card}(\text{Path}[s]_{s,h'_j}^X(t')) = k + \text{card}(P'_j).$$

From (4), $\min(\text{card}(P_j), \mathcal{S}(\alpha_j)) = \min(\text{card}(P'_j), \mathcal{S}(\alpha_j))$, which implies (f).

Proof of (g). Suppose there is $\delta_1 \in [0, n]$ such that $h_j^{\delta_1}(\llbracket t \rrbracket_{s,h_j}^X) = s(u)$. We split the proof depending on whether $\delta_1 \leq k$.

case: $\delta_1 \leq k$. In this case, $n - \delta_1 = (k - \delta_1) + \text{card}(P_j)$. $s(u)$ appears in ρ'' and so $h_j^{\delta_1}(\llbracket t \rrbracket_{s,h'_j}^X) = s(u)$. We have $m - \delta_1 = (k - \delta_1) + \text{card}(P'_j)$. From (4),

$$\min(\text{card}(P_j), \mathcal{S}(\alpha_j)) = \min(\text{card}(P'_j), \mathcal{S}(\alpha_j)),$$

which allows us to deduce $\min(n - \delta_1, \mathcal{S}(\alpha_j)) = \min(m - \delta_1, \mathcal{S}(\alpha_j))$. From (★1),

$$\min(n - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(m - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)).$$

By defining $\delta_2 = \delta_1$, the statement (g) is verified.

case: $\delta_1 \geq k$. In this case, $s(u)$ belongs to $P_j \cup \{e_{P_j}\}$. There is $\delta'_1 \in [0, \text{card}(P_j)]$ such that $\delta_1 = k + \delta'_1$. As $h_j \subseteq h$, from the property (7) of the construction, there is $\delta'_2 \in [0, \text{card}(P'_j)]$ such that $h_j^{\delta'_2}(\llbracket t \rrbracket_{s,h'}^X) = s(u)$ and

$$\min(\delta'_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta'_2, \mathcal{S}_{\text{left}}(\alpha_j)),$$

$$\min(\text{card}(P_j) - \delta'_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(P'_j) - \delta'_2, \mathcal{S}_{\text{right}}(\alpha_j)).$$

As $P'_j \subseteq \text{dom}(h'_j)$, we conclude that $h_j^{\delta'_2}(\llbracket t \rrbracket_{s,h'}^X) = s(u)$. Let $\delta_2 = k + \delta'_2$.

From $\ell_k = \llbracket t \rrbracket_{s,h'}^X$, we derive $h_j^{\delta_2}(\llbracket t \rrbracket_{s,h'}^X) = s(u)$. By $m = k + \text{card}(P'_j)$, we have $\delta_2 \in [0, m]$. To conclude the proof, we must prove that

$$\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j)),$$

$$\min(n - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(m - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j)).$$

Since $\delta_1 = k + \delta'_1$ and $\delta_2 = k + \delta'_2$, the first equivalence holds thanks to $\min(\delta'_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta'_2, \mathcal{S}_{\text{left}}(\alpha_j))$. As $n = k + \text{card}(P_j)$ and $m = k + \text{card}(P'_j)$, we have $n - \delta_1 = \text{card}(P_j) - \delta'_1$, $m - \delta_2 = \text{card}(P'_j) - \delta'_2$. Then, the second equivalence holds from $\min(\text{card}(P_j) - \delta'_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(P'_j) - \delta'_2, \mathcal{S}_{\text{right}}(\alpha_j))$.

Proof of (h). Symmetrical to the proof of (g). We rely on (8) instead of (7).

Proof of (j)–(l). Among the locations in $\text{Path}[s]_{s,h}^X(t)$, the set $\text{Path}[s]_{s,h_j}^X(t')$ only contains the ones in P_j . Thus,

$$P_j \subseteq \text{Path}[s]_{s,h_j}^X(t'), \quad R_j \cap \text{Path}[s]_{s,h_j}^X(t') = \emptyset, \quad l_{\text{pre}} \notin \text{Path}[s]_{s,h_j}^X(t').$$

Similarly, among the locations in $\text{Path}[s]_{s,h'}^X(t)$, the set $\text{Path}[s]_{s,h'_j}^X(t')$ only contains the ones in P'_j . Thus,

$$P'_j \subseteq \text{Path}[s]_{s,h'_j}^X(t'), \quad R'_j \cap \text{Path}[s]_{s,h'_j}^X(t') = \emptyset, \quad l'_{\text{pre}} \notin \text{Path}[s]_{s,h'_j}^X(t').$$

Therefore, the statements (j)–(l) are verified.

In order to conclude the proof of (B), we treat the case where $\ell_n \neq e_{P_j}$.

case: $\ell_n \neq e_{P_j}$. Since $\ell_m = \text{sby}_{s,h'_j}^X(t') \in \text{Lab}[s]_{s,h'_j}^X$, From (A), we derive that $\ell_n \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. As we are assuming $\ell_k = \llbracket t \rrbracket_{s,h}^X$, where $k \in [0, n-1]$, this implies that the path described by $\text{Path}[s]_{s,h}^X(t)$ must be included in ρ , i.e. $n \geq \text{card}(\text{Path}[s]_{s,h}^X(t)) + k$. In particular, $k_1 = k + \text{card}(\text{Path}[s]_{s,h'}^X(t))$ is such that $\ell_{k_1} = \text{sby}_{s,h}^X(t)$, and the path ρ can be split into the three paths below:

- its prefix ρ'' that goes from ℓ_0 to $\ell_k = \llbracket t \rrbracket_{s,h}^X$,
- the path described by $\text{Path}[s]_{s,h}^X(t)$, that goes from $\llbracket t \rrbracket_{s,h}^X$ to $\ell_{k_1} = \text{sby}_{s,h}^X(t)$,
- the suffix $\hat{\rho} \stackrel{\text{def}}{=} (\ell_{k_1}, \dots, \ell_n)$.

As ρ is a minimal path in h_j , going from ℓ_0 to ℓ_n , these three paths are mutually disjoint, and $n = k + \text{card}(\text{Path}[s]_{s,h}^X(t)) + (n - k_1)$. This pattern is depicted in the third case of Figure 5.17. We show that ρ' can be decomposed in a similar way. From (b), we derive $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h'_j)$. This means that $e_{P'_j} \in \text{dom}(h'_j)$, and thus, by (A), the labelled location ℓ_m is not $e_{P'_j}$ nor it belongs to $\text{Path}[s]_{s,h'}^X(t) \setminus \{\llbracket t \rrbracket_{s,h'}^X\}$. Therefore, from $\ell'_k = \llbracket t \rrbracket_{s,h'}^X$, the path described by $\text{Path}[s]_{s,h'}^X(t)$ is included in ρ' , i.e. $m \geq \text{card}(\text{Path}[s]_{s,h'}^X(t)) + k$. Consider $k_2 = k + \text{card}(\text{Path}[s]_{s,h'}^X(t))$. We have $\ell'_{k_2} = \text{sby}_{s,h'}^X(t)$, and ρ' can be split into the three paths below:

- its prefix ρ'' that goes from ℓ'_0 to $\ell'_k = \llbracket t \rrbracket_{s,h'}^X$,
- the path described by $\text{Path}[s]_{s,h'}^X(t)$, that goes from $\llbracket t \rrbracket_{s,h'}^X$ to $\ell'_{k_2} = \text{sby}_{s,h'}^X(t)$,
- the suffix $\hat{\rho}' = (\ell'_{k_2}, \dots, \ell'_m)$.

Since ρ' is a minimal path in h'_j , going from ℓ'_0 to ℓ'_m , these three paths are mutually disjoint, and $m = k + \text{card}(\text{Path}[s]_{s,h'}^X(t)) + (m - k_2)$. The following three statements show that $\hat{\rho} = \hat{\rho}'$.

- $\ell_{k_1} = \text{sby}_{s,h}^X(t) = \text{sby}_{s,h'}^X(t) = \ell'_{k_2}$.

This holds directly from (=t) together with the fact that (s, h) and (s, h') satisfy the same core formulae of the form $\text{sees}_X(t, t'') \geq 1$,

- $\{\ell_{k_1}, \dots, \ell_{n-1}\} \subseteq \text{dom}(\widehat{h_j})$, and $\{\ell'_{k_2}, \dots, \ell'_{m-1}\} \subseteq \text{dom}(\widehat{h_j})$.

For the first inclusion, we recall that ρ is a minimal path from ℓ_0 and ℓ_n . Since $\text{Path}[s]_{s,h}^X(t) = \{\ell_k, \dots, \ell_{k_1-1}\}$, for every location $\ell \in \{\ell_{k_1-1}, \dots, \ell_{n-1}\}$ we have $\ell \notin \text{Path}[s]_{s,h}^X(t)$. As $\ell \in \text{dom}(h_j)$, we derive $\ell \in \text{dom}(\widehat{h_j})$, by definition of $\widehat{h_j}$.

The second inclusion is proven analogously.

- $\widehat{\rho}$ and $\widehat{\rho}'$ have the same length, i.e. $n - k_1 = m - k_2$.

Together, the previous two statements show that $\ell_{k_1+i} = \ell'_{k_2+i}$ holds for every $i \in [0, \min(n - k_1, m - k_2)]$. *Ad absurdum*, suppose that $n - k_1 \neq m - k_2$. In this case, $\widehat{\rho}$ and $\widehat{\rho}'$ have different lengths. Let us assume $n - k_1 < m - k_2$. The case where $n - k_1 > m - k_2$ is analogous. The location ℓ_n is different from ℓ'_m and appears in both $\widehat{\rho}'$ and in ρ' . From $\ell_n \neq e_{P_j}$ and by (A), we conclude that $\ell_n \in \text{Lab}[\mathcal{S}]_{s,h'_j}^X$. However, this implies that the path

$$(\ell'_0, \dots, \ell'_{k_2} = \ell_{k_1}, \dots, \ell'_{k_2 + \min(n - k_1, m - k_2)} = \ell_n)$$

obtained from ρ' by removing the locations after ℓ_n is a path in h'_j , going from $\llbracket t' \rrbracket_{s,h'_j}^X$ to a labelled location. This is contradictory, as ρ is the minimal path in h'_j going from $\llbracket t' \rrbracket_{s,h'_j}^X$ to $\text{sby}_{s,h'_j}^X(t')$. According to Definition 5.27, with the exception of $\llbracket t' \rrbracket_{s,h'_j}^X$ and $\text{sby}_{s,h'_j}^X(t')$, no other location in ρ belongs to $\text{Lab}[\mathcal{S}]_{s,h'_j}^X$. Thus, $n - k_1 = m - k_2$.

Thanks to $\widehat{\rho} = \widehat{\rho}'$, we can now show (e)–(h), ending the proof of (B).

Proof of (e). Holds directly from (A), as we show that $\ell_n = \ell'_m$. If $\widehat{\rho}$ is not an empty path, this holds by $\widehat{\rho} = \widehat{\rho}'$. Else, $\ell_n = \ell_{k_1} = \text{sby}_{s,h}^X(t) = \text{sby}_{s,h'}^X(t) = \ell'_{k_2} = \ell'_m$.

Below, we recall that $n = \text{card}(\text{Path}[\mathcal{S}]_{s,h_j}^X(t'))$ and $m = \text{card}(\text{Path}[\mathcal{S}]_{s,h'_j}^X(t'))$.

Proof of (f). The following inequalities imply (f):

$$\text{card}(\text{Path}[\mathcal{S}]_{s,h_j}^X(t')) \geq \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \mathcal{S}(\alpha)$$

$$\text{card}(\text{Path}[\mathcal{S}]_{s,h'_j}^X(t')) \geq \text{card}(\text{Path}[\mathcal{S}]_{s,h'}^X(t)) \geq \mathcal{S}(\alpha).$$

Proof of (g). Suppose there is $\delta_1 \in [0, n]$ such that $h_j^{\delta_1} = s(u)$. We split the proof depending on whether $\delta_1 < k$, $\delta_1 > k_1$ or $\delta_1 \in [k, k_1]$.

case: $\delta_1 < k$. $s(u)$ appears in ρ'' and so $h_j'^{\delta_1}(\llbracket t' \rrbracket_{s,h'_j}^X) = s(u)$. We have,

$$n - \delta_1 \geq \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \mathcal{S}(\alpha),$$

$$m - \delta_1 \geq \text{card}(\text{Path}[\mathcal{S}]_{s,h'}^X(t)) \geq \mathcal{S}(\alpha).$$

By (★₁), defining $\delta_2 = \delta_1$ verifies the statement (g).

case: $\delta_1 > k_1$. $s(u)$ appears in $\widehat{\rho}$. In particular, $h_j^{\delta_1 - k_1}(\ell_{k_1}) = s(u)$. Following the statement (g), let $\delta_2 \stackrel{\text{def}}{=} k_2 + (\delta_1 - k_1)$. By $(\ell_{k_1}, \dots, \ell_n) = \widehat{\rho} = \widehat{\rho}' = (\ell'_{k_2}, \dots, \ell'_m)$, we derive $h_j'^{\delta_2 - k_2}(\ell_m) = s(u)$ and $\delta_2 \in [0, m]$. Therefore, $h_j'^{\delta_2}(\llbracket t' \rrbracket_{s,h'_j}^X) = s(u)$ follows by definition of $\widehat{\rho}'$. We have,

$$\delta_1 \geq k_1 \geq \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t)) \geq \mathcal{S}(\alpha),$$

$$\delta_2 \geq k_2 \geq \text{card}(\text{Path}[\mathcal{S}]_{s,h'}^X(t)) \geq \mathcal{S}(\alpha).$$

By (★₁), this implies $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j))$. Lastly, we have

$$m - \delta_2 = m - (k_2 + \delta_1 - k_1) = n - \delta_1,$$

where the last equality holds by $m - k_2 = n - k_1$. Thus, (g) is satisfied.

case: $\delta_1 \in [k, k_1]$. Let $\delta'_1 = \delta_1 - k$. From $\ell_k = \llbracket t \rrbracket_{s,h}^X$, we have $h_j^{\delta'_1}(\llbracket t \rrbracket_{s,h}^X) = s(u)$. Moreover, since $k_1 = k + \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t))$, we have $\delta'_1 \in [0, \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(t))]$.

From $h_j \subseteq h$, $h_j^{\delta'_1}(\llbracket t \rrbracket_{s,h}^X) = s(u)$. We remind the reader that (s, h) and (s, h') satisfy the same core formulae of the form $u = t$, as well as the core formulae of the form $u \in \text{sees}_X(t, t'') \geq (\beta_1, \beta_2)$, where $\beta_1 \in [1, \mathcal{S}_{\text{left}}(\alpha)]$ and $\beta_2 \in [1, \mathcal{S}_{\text{right}}(\alpha)]$. From the semantics of these formulae, we conclude that there is $\delta'_2 \in [0, \text{Path}[\mathcal{S}]_{s,h'}^X(t)]$ such that

- * $h'^{\delta'_2}(\llbracket t \rrbracket_{s,h'}^X) = s(u)$,
 - * $\min(\delta'_1, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta'_2, \mathcal{S}_{\text{left}}(\alpha))$,
 - * $\min(\text{card}(\text{Path}[s]_{s,h}^X(t)) - \delta'_1, \mathcal{S}_{\text{right}}(\alpha)) = \min(\text{card}(\text{Path}[s]_{s,h'}^X(t)) - \delta'_2, \mathcal{S}_{\text{right}}(\alpha))$.
- Since $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h'_j)$, we conclude that $h'^{\delta'_2}(\llbracket t \rrbracket_{s,h'}^X) = s(u)$. Let us define $\delta_2 \stackrel{\text{def}}{=} k + \delta'_2$. As $\ell_k = \llbracket t \rrbracket_{s,h'}^X$, by definition of ρ' we conclude that $h'^{\delta_2}(\llbracket t' \rrbracket_{s,h'_j}^X) = s(u)$. Since $\delta_1 = k + \delta'_1$ and $\mathcal{S}_{\text{left}}(\alpha_j) \leq \mathcal{S}_{\text{left}}(\alpha)$, the equivalence $\min(\delta'_1, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta'_2, \mathcal{S}_{\text{left}}(\alpha))$ implies $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j))$. From $k_1 = k + \text{card}(\text{Path}[s]_{s,h}^X(t))$ and $\delta'_1 = \delta_1 - k$ we derive

$$n - \delta_1 = (n - k_1) + (\text{card}(\text{Path}[s]_{s,h}^X(t)) - \delta'_1).$$

From $k_2 = k + \text{card}(\text{Path}[s]_{s,h'}^X(t))$, $\delta'_2 = \delta_2 - k$ and $n - k_1 = m - k_2$, we have

$$m - \delta_2 = (n - k_1) + (\text{card}(\text{Path}[s]_{s,h'}^X(t)) - \delta'_2).$$

By $\min(\text{card}(\text{Path}[s]_{s,h}^X(t)) - \delta'_1, \mathcal{S}_{\text{right}}(\alpha)) = \min(\text{card}(\text{Path}[s]_{s,h'}^X(t)) - \delta'_2, \mathcal{S}_{\text{right}}(\alpha))$ and **(★1)**, this allows us to derive that

$$\min(n - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(m - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j)),$$

which concludes the proof of **(g)**.

Proof of (h). Symmetrical to the proof of **(g)**.

Proof of (j)–(l). Since $\text{Path}[s]_{s,h}^X(t) \subseteq \text{dom}(h_j)$ and $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h_j)$, we conclude that $R_{3-j} = \emptyset$ and $R'_{3-j} = \emptyset$. By definition of P_j (resp. **(3)**), we have $e_{P_j} = l_{\text{pre}}$ (resp. $e_{P'_j} = l'_{\text{pre}}$), and thus P_j and $\{l_{\text{pre}}\}$ (resp. P'_j and $\{l'_{\text{pre}}\}$) partition $\text{Path}[s]_{s,h}^X(t)$ (resp. $\text{Path}[s]_{s,h'}^X(t)$). Therefore, $R_j = \emptyset$ and $R'_j = \emptyset$. From $\text{Path}[s]_{s,h}^X(t) \subseteq \text{Path}[s]_{s,h_j}^X(t')$ and $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{Path}[s]_{s,h'_j}^X(t')$ we conclude that

$$R_j \subseteq \text{Path}[s]_{s,h_j}^X(t') = \emptyset \quad P_j \subseteq \text{Path}[s]_{s,h_j}^X(t'), \quad l_{\text{pre}} \in \text{Path}[s]_{s,h_j}^X(t'),$$

$$R'_j \subseteq \text{Path}[s]_{s,h'_j}^X(t') = \emptyset \quad P'_j \subseteq \text{Path}[s]_{s,h'_j}^X(t'), \quad l'_{\text{pre}} \in \text{Path}[s]_{s,h'_j}^X(t').$$

Therefore, the statements **(j)–(l)** are verified.

C. For every $x \in X$,

- (m) $\text{card}(\text{Pred}[s]_{s,h_j}^X(x)) = \text{card}(\text{Pred}[s]_{s,h'_j}^X(x))$,
- (n) $s(u) \in \text{Pred}[s]_{s,h_j}^X(x)$ if and only if $s(u) \in \text{Pred}[s]_{s,h'_j}^X(x)$.

First, we prove two intermediate results.

- (o) $\text{Pred}[s]_{s,h_j}^X(x) \setminus \{l_{\text{pre}}\} = \text{Pred}[s]_{s,h'_j}^X(x) \setminus \{l'_{\text{pre}}\}$,
- (p) $l_{\text{pre}} \in \text{Pred}[s]_{s,h_j}^X(x)$ if and only if $l'_{\text{pre}} \in \text{Pred}[s]_{s,h'_j}^X(x)$.

Proof of (o). (\subseteq): Consider a location $\ell \in \text{Pred}[s]_{s,h_j}^X(x) \setminus \{l_{\text{pre}}\}$. By definition, $h_j(\ell) = s(x)$ and h_j does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. In particular, this implies that ℓ is not assigned to a program variable. Moreover, from $\ell \neq l_{\text{pre}}$, we conclude that ℓ does not belong to the set $\text{Path}[s]_{s,h}^X(t)$. Indeed, from its definition, l_{pre} is the only location of this set that can point to a program variable. By definition of $\widehat{h_j}$, we conclude that $\ell \in \widehat{h_j}$ and $\widehat{h_j}(\ell) = s(x)$. Thus, by definition of h'_j , both $h'_j(\ell) = s(x)$ and $\ell \notin \text{Path}[s]_{s,h'}^X(t)$ hold. In order to conclude that $\ell \in \text{Pred}[s]_{s,h'_j}^X(x)$, it is sufficient to show that h'_j does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. *Ad absurdum*, suppose that h'_j witnesses such a path. As we know that ℓ is not assigned to a program variable, this path must be non-empty. We apply **(d)** and

conclude that h_j witnesses a path going from $s(y)$ to ℓ : a contradiction. Therefore, $\ell \in \text{Pred}[s]_{s,h'_j}^X(x)$. Lastly, again from $\ell \notin \text{Path}[s]_{s,h'}^X(t)$, we have $\ell \neq l'_{pre}$.

(\supseteq): Symmetrical to the other direction.

Proof of (p). (\Rightarrow): Suppose $l_{pre} \in \text{Pred}[s]_{s,h_j}^X(x)$. So, $h_j(l_{pre}) = s(x)$ and h_j does not witness a path going from $s(y)$ to l_{pre} , for any $y \in X$. $l_{pre} \notin \text{Lab}[s]_{s,h_j}^X$. Notice that then, from the assumption $l_{pre} \in \text{dom}(h_1)$, we have $j = 1$. Moreover, by definition of l_{pre} , $\text{sby}_{s,h}^X(t) = s(x)$. As (s, h) and (s, h') satisfy the same formulae of the form $t = t'$, by (= t), $\text{sby}_{s,h'}^X(t) = s(x)$. By definition of l'_{pre} , $h'(l'_{pre}) = s(x)$, which in turn implies $h'_1(l'_{pre}) = s(x)$ by definition of h'_1 . In order to conclude that $l'_{pre} \in \text{Pred}[s]_{s,h'_j}^X(x)$, we show that h'_j does not witness a path going from $s(y)$ to l'_{pre} , for any $y \in X$. *Ad absurdum*, suppose that such a path exists. In this case, we conclude that $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h'_j)$ and that h'_j witnesses a path going from $s(y)$ to $\llbracket t \rrbracket_{s,h'}^X$. From (b), $\text{Path}[s]_{s,h}^X(t) \subseteq \text{dom}(h_j)$ and so h_j witnesses a path going from $\llbracket t \rrbracket_{s,h}^X$ to l_{pre} . Clearly, if $s(y) = \llbracket t \rrbracket_{s,h}^X$, we derive a contradiction. Otherwise, $s(y) \neq \llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$ implies that the path in h'_j going from $s(y)$ to $\llbracket t \rrbracket_{s,h'}^X$ is non-empty. By (d) we conclude that h_j witnesses a path going from $s(y)$ to $\llbracket t \rrbracket_{s,h}^X$. However, this implies that h_j witnesses a path going from $s(y)$ to l_{pre} : a contradiction. We conclude that h'_j does not witness a path going from $s(y)$ to l'_{pre} . Thus, $l'_{pre} \in \text{Pred}[s]_{s,h'_j}^X(x)$.

(\Leftarrow): Symmetrical to the other direction.

Proof of (m). Directly from (o) and (p).

Proof of (n). (\Rightarrow): Suppose $s(u) \in \text{Pred}[s]_{s,h_j}^X(x)$. If $s(u) \neq l_{pre}$ then the result holds directly by (o). Otherwise, suppose $s(u) = l_{pre}$. Thus, $l_{pre} \in \text{Pred}[s]_{s,h_j}^X(x)$ which implies $l'_{pre} \in \text{Pred}[s]_{s,h'_j}^X(x)$, by (p). To conclude the proof, we show that $l'_{pre} = s(u)$. By definition of l_{pre} , the memory state (s, h) satisfies the formula $u \in \text{sees}_X(t, x) \geq (1, 1)$ whereas it does not satisfy the formula $u \in \text{sees}_X(t, x) \geq (1, 2)$. We notice that both core formulae belong to $\text{Core}[s](X, \alpha')$, for every $\alpha' \geq 1$. From $(s, h) \approx^s X, \alpha(s, h')$, $(s, h') \models u \in \text{sees}_X(t, x) \geq (1, 1)$ and $(s, h') \not\models u \in \text{sees}_X(t, x) \geq (1, 2)$. Thus, $\text{sby}_{s,h'}^X(t) = s(x)$ and there is $\bar{\ell} \in \text{Path}[s]_{s,h'}^X(t)$ such that $\bar{\ell} = s(u)$ and $h(\bar{\ell}) = s(x)$. By definition, $\bar{\ell} = l'_{pre}$, and therefore $l'_{pre} = s(u)$.

(\Leftarrow): Symmetrical to the other direction.

- D. For every $\beta \in [1, \alpha_j]$, $\text{Cycl}[s]_{s,h_j}^X(\beta) = \text{Cycl}[s]_{s,h'_j}^X(\beta)$

We prove an intermediate result which helps us showing both (D) and (E) (below).

(μ) Let (\hat{s}, \hat{h}) be a memory state s.t. $\text{Path}[s]_{\hat{s}, \hat{h}}^X(t) \neq \emptyset$. Let L be a minimal set of locations describing a cycle in \hat{h} . If $\text{Path}[s]_{\hat{s}, \hat{h}}^X(t) \cap L \neq \emptyset$ then $\text{Path}[s]_{\hat{s}, \hat{h}}^X(t) \subseteq L$.

Proof of (μ). Let ℓ be the first location that, in \hat{h} , is reachable from $\llbracket t \rrbracket_{\hat{s}, \hat{h}}^X$ and belongs to both $\text{Path}[s]_{\hat{s}, \hat{h}}^X(t)$ and L . Since L is a minimal set of locations describing a cycle, $\ell \in L$ implies that every location reachable from ℓ belongs to L . So, (μ) holds as soon as we show that $\ell = \llbracket t \rrbracket_{\hat{s}, \hat{h}}^X$. By definition of $\llbracket . \rrbracket_{\hat{s}, \hat{h}}^X$, $\llbracket t \rrbracket_{\hat{s}, \hat{h}}^X$ is reached by a location corresponding to a program variable $x \in X$. By definition of ℓ , we conclude that ℓ is the first location reachable from $s(x)$ that belongs to a cycle. By definition of end-point variables, $\ell = \llbracket e(x) \rrbracket_{\hat{s}, \hat{h}}^X$, and thus ℓ is a labelled location. By definition of $\text{Path}[s]_{\hat{s}, \hat{h}}^X(t)$, we know that $\llbracket t \rrbracket_{\hat{s}, \hat{h}}^X$ is the only labelled locations belonging to this set. Thus, $\ell = \llbracket t \rrbracket_{\hat{s}, \hat{h}}^X$.

Proof of (D). (\subseteq): Consider $L \in \text{Cycl}[s]_{s,h_j}^X(\beta)$. So,

- h. L describes a cycle of size β in h_j ,
- i. every $\ell \in L$ does not belong to $\text{Lab}[s]_{s,h_j}^X$.

From (h) and $h_j \subseteq h$, L describes a cycle of size β in h . Clearly, $\text{Path}[s]_{s,h}^X(t)$ cannot be a subset of L as we have

$$\text{card}(\text{Path}[s]_{s,h}^X(t)) \geq S(\alpha) > \beta = \text{card}(L).$$

By (μ), we conclude that $\text{Path}[s]_{s,h}^X(t) \cap L = \emptyset$, which in turn implies that L describes a cycle of size β in \widehat{h}_j , directly from the definition of h_j . From $\widehat{h}_j \subseteq h'_j$, L describes a cycle of size β in \widehat{h}_j . Clearly, $l'_{\text{pre}} \notin L$, as otherwise, again by (μ), we would be able to conclude that $\text{Path}[s]_{s,h'}^X(t) \subseteq L$, in contradiction with the cardinalities of these two sets. Therefore, from (i) and (A), we conclude that $L \cap \text{Lab}[s]_{s,h'_j}^X = \emptyset$. By definition, $L \in \text{Cycl}[s]_{s,h'_j}^X(\beta)$.

(\supseteq): Symmetrical to the other direction.

- E. (q) $\text{card}(\uparrow \text{Cycl}[s]_{s,h_j}^{X,\alpha_j}) = \text{card}(\uparrow \text{Cycl}[s]_{s,h'_j}^{X,\alpha_j})$,
(r) $s(u) \in [\uparrow \text{Cycl}[s]_{s,h_j}^{X,\alpha_j}]^\flat$ if and only if $s(u) \in [\uparrow \text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}]^\flat$.

First, we prove two intermediate results.

- (s) $\{L \in \uparrow \text{Cycl}[s]_{s,h_j}^{X,\alpha_j} \mid L \cap \text{Path}[s]_{s,h}^X(t) = \emptyset\} = \{L' \in \uparrow \text{Cycl}[s]_{s,h'_j}^{X,\alpha_j} \mid L' \cap \text{Path}[s]_{s,h'}^X(t) = \emptyset\}$,
(t) $\text{Path}[s]_{s,h}^X(t) \subseteq [\uparrow \text{Cycl}[s]_{s,h_j}^{X,\alpha_j}]^\flat$ if and only if $\text{Path}[s]_{s,h'}^X(t) \subseteq [\uparrow \text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}]^\flat$.

Proof of (s). (\subseteq): Consider $L \in \uparrow \text{Cycl}[s]_{s,h_j}^{X,\alpha_j}$ such that $L \cap \text{Path}[s]_{s,h}^X(t) = \emptyset$. By definition, L describes a cycle in h_j , of size greater than α_j . Moreover, every location in L does not belong to $\text{Lab}[s]_{s,h_j}^X$. As done in the proof of (D), $L \cap \text{Path}[s]_{s,h}^X(t) = \emptyset$ implies that L describes a cycle of size greater than α_j , in \widehat{h}_j . From $\widehat{h}_j \subseteq h'_j$, L describes a cycle of size greater than α_j , in h'_j . Moreover, $l'_{\text{pre}} \notin L$, as otherwise, by (μ), $\text{Path}[s]_{s,h'}^X(t) \subseteq L$, in contradiction with $L \cap \text{Path}[s]_{s,h}^X(t) = \emptyset$ (both L and $\text{Path}[s]_{s,h}^X(t)$ are non-empty). Thus, from (A) and $L \cap \text{Lab}[s]_{s,h_j}^X = \emptyset$, we conclude that $L \cap \text{Lab}[s]_{s,h'_j}^X = \emptyset$. By definition, $L \in \uparrow \text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}$.

(\supseteq): Symmetrical to the other direction.

Proof of (t). (\Rightarrow): Suppose $\text{Path}[s]_{s,h}^X(t) \subseteq [\uparrow \text{Cycl}[s]_{s,h_j}^{X,\alpha_j}]^\flat$ and thus let us consider a set $L \in \uparrow \text{Cycl}[s]_{s,h_j}^{X,\alpha_j}$ such that $\text{Path}[s]_{s,h}^X(t) \cap L \neq \emptyset$. As $h_j \subseteq h$, L describes a cycle in both h_j and h . By (μ) we conclude that $\text{Path}[s]_{s,h}^X(t) \subseteq L$. As $L \subseteq \text{dom}(h_j)$, by (b) we have $\text{Path}[s]_{s,h'}^X(t) \subseteq \text{dom}(h'_j)$ (and $j = 1$). To conclude the proof, we show

$$(L \setminus \text{Path}[s]_{s,h}^X(t)) \cup \text{Path}[s]_{s,h'}^X(t) \in \uparrow \text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}. \quad (\S_1)$$

First of all, since $\text{Path}[s]_{s,h}^X(t) \subseteq L \subseteq \text{dom}(h_j)$ and $h_j \subseteq h$, we conclude that $\text{Path}[s]_{s,h}^X(t)$ describes a path in h_j , going from $\llbracket t \rrbracket_{s,h}^X$ to $\text{sby}_{s,h}^X(t)$, and moreover this path belongs to a cycle of length greater than α_j , which in turn is described by the set L . We notice that we can have $L = \text{Path}[s]_{s,h}^X(t)$. We distinguish two cases, based on the truth of this equality.

case: $L = \text{Path}[s]_{s,h}^X(t)$. In this case, $L \setminus \text{Path}[s]_{s,h}^X(t) = \emptyset$ and thus, in order to prove (§1), we show $\text{Path}[s]_{s,h'}^X(t) \in \uparrow \text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}$. The set $\text{Path}[s]_{s,h}^X(t)$ describes a cycle in both h and h_j , so $\llbracket t \rrbracket_{s,h}^X = \text{sby}_{s,h}^X(t)$. As (s, h) and (s, h') satisfy

the same formulae of the form $t = t'$ and $\text{sees}_X(t, t') \geq 1$, we conclude that $\llbracket t \rrbracket_{s, h'}^X = \text{sby}_{s, h'}^X(t)$. Since $\text{Path}[s]_{s, h'}^X(t) \subseteq \text{dom}(h'_j)$ and $h'_j \subseteq h'$, this implies that $\text{Path}[s]_{s, h'}^X(t)$ describes a cycle in h'_j . Let us show that this cycle is unlabelled. In h (resp. h'), the only labelled location of $\text{Path}[s]_{s, h}^X(t)$ (resp. $\text{Path}[s]_{s, h'}^X(t)$) is the one that corresponds to t . Since $\text{Path}[s]_{s, h}^X(t) = L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha_j}$, by definition of $\uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha_j}$ we conclude that $\llbracket t \rrbracket_{s, h}^X$ does not belong to $\text{Lab}[s]_{s, h_j}^X$. By (A), $\llbracket t \rrbracket_{s, h'}^X$ does not belong to $\text{Lab}[s]_{s, h'_j}^X$. So, $\text{Path}[s]_{s, h'}^X(t) \in \uparrow\text{Cycl}[s]_{s, h'_j}^{X, \alpha_j}$.

case: $L \neq \text{Path}[s]_{s, h}^X(t)$. In this case, $L \setminus \text{Path}[s]_{s, h}^X(t)$ is a minimal set describing a path, in both h and h_j , going from $\text{sby}_{s, h}^X(t)$ to $\llbracket t \rrbracket_{s, h}^X$. By definition of $\widehat{h_j}$, $L \setminus \text{Path}[s]_{s, h}^X(t)$ is a minimal set describing a path in $\widehat{h_j}$, going from $\text{sby}_{s, h}^X(t)$ to $\llbracket t \rrbracket_{s, h}^X$. By $\widehat{h_j} \subseteq h'_j$, the same holds for h'_j , going from $\text{sby}_{s, h}^X(t)$ to $\llbracket t \rrbracket_{s, h}^X$. As (s, h) and (s, h') satisfy the same formulae of the form $t = t'$, by (=t), $\text{sby}_{s, h}^X(t) = \text{sby}_{s, h'}^X(t)$ and $\llbracket t \rrbracket_{s, h}^X = \llbracket t \rrbracket_{s, h'}^X$. From $\text{Path}[s]_{s, h'}^X(t) \subseteq \text{dom}(h'_j)$, $\text{Path}[s]_{s, h'}^X(t)$ is the minimal set that describes the path in h'_j going from $\llbracket t \rrbracket_{s, h'}^X$ to $\text{sby}_{s, h'}^X(t)$. From $L \setminus \text{Path}[s]_{s, h}^X(t) \subseteq \text{dom}(\widehat{h_j})$, the sets $L \setminus \text{Path}[s]_{s, h}^X(t)$ and $\text{Path}[s]_{s, h'}^X(t)$ are disjoint, and thus $(L \setminus \text{Path}[s]_{s, h}^X(t)) \cup \text{Path}[s]_{s, h'}^X(t)$ describes a cycle in h'_j . Moreover, it is the minimal set that describes that cycle. Since the set $\text{Path}[s]_{s, h'}^X(t)$ contains at least $\mathcal{S}(\alpha)$ locations, the length of the cycle is greater than α_j . Similarly to the previous case of the proof, by (A), the cycle is unlabelled. Thus, (§1) holds.

(\Leftarrow): Symmetrical to the other direction. In particular, if $\text{Path}[s]_{s, h'}^X(t) \cap L \neq \emptyset$ holds for some $L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha_j}$, one can show that

$$(L \setminus \text{Path}[s]_{s, h'}^X(t)) \cup \text{Path}[s]_{s, h}^X(t) \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha_j}. \quad (\S_2)$$

Proof of (q). Thanks to (s), in order to prove (q) it is sufficient to show that the cardinalities of the two following sets coincide:

$$\{L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha} \mid L \cap \text{Path}[s]_{s, h}^X(t) \neq \emptyset\}, \{L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha} \mid L \cap \text{Path}[s]_{s, h'}^X(t) \neq \emptyset\}.$$

In fact, we notice that, whenever non-empty, these two sets can only contain one element. Indeed, if we consider a set $L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha}$ such that $L \cap \text{Path}[s]_{s, h}^X(t) \neq \emptyset$, we know that it describes a cycle in h . Therefore, by (μ), $\text{Path}[s]_{s, h}^X(t) \subseteq L$ and so the cardinality of $\{L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha} \mid L \cap \text{Path}[s]_{s, h}^X(t) \neq \emptyset\}$ is one (recall that the sets in $\uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha}$ are mutually disjoint). A similar analysis can be done for $\{L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha} \mid L \cap \text{Path}[s]_{s, h'}^X(t) \neq \emptyset\}$. By (t), $L \cap \text{Path}[s]_{s, h}^X(t) \neq \emptyset$, for some $L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha}$, if and only if there is $L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, \alpha}$ s.t. $L \cap \text{Path}[s]_{s, h'}^X(t) \neq \emptyset$.

We conclude that the cardinalities of the two sets under analysis coincide.

Proof of (r). (\Rightarrow): Suppose that there is $L \in \uparrow\text{Cycl}[s]_{s, h_j}^{X, t}$ such that $s(u) \in L$. First of all, if $s(u) \in \text{Path}[s]_{s, h}^X(t)$, then (r) follows from (t). Otherwise, $s(u) \in L \setminus \text{Path}[s]_{s, h}^X(t)$. If $\text{Path}[s]_{s, h}^X(t) \cap L = \emptyset$, then (r) follows directly from (s). Else, $\text{Path}[s]_{s, h}^X(t) \cap L \neq \emptyset$ allows us to apply the construction in the proof of (t) in order to conclude that (§1) holds. As $s(u) \in L \setminus \text{Path}[s]_{s, h}^X(t)$, this implies (r).

(\Leftarrow): Symmetrical to the other direction.

$$F. (u) \min(\text{card}(\text{Rem}[s]_{s, h_j}^{X, \alpha_j}), \mathcal{R}(\alpha_j)) = \min(\text{card}(\text{Rem}[s]_{s, h'_j}^{X, \alpha_j}), \mathcal{R}(\alpha_j)),$$

(v) $s(u) \in \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$ if and only if $s(u) \in \text{Rem}[S]_{s,h'_j}^{X,\alpha_j}$.

First, we prove four intermediate results.

(w) $\text{Rem}[S]_{s,h_j}^{X,\alpha_j} \setminus (R_j \cup P_j \cup \{l_{pre}\}) = \text{Rem}[S]_{s,h'_j}^{X,\alpha_j} \setminus (R'_j \cup P'_j \cup \{l'_{pre}\})$

(x) $R_j \subseteq \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$ and $R'_j \subseteq \text{Rem}[S]_{s,h'_j}^{X,\alpha_j}$.

(y) $l_{pre} \in \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$ if and only if $l'_{pre} \in \text{Rem}[S]_{s,h'_j}^{X,\alpha_j}$.

(z) $P_j \cap \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$ is either \emptyset or P_j . Similarly, $P'_j \cap \text{Rem}[S]_{s,h'_j}^{X,\alpha_j}$ is either \emptyset or P'_j . Lastly, $P_j \subseteq \text{Rem}[S]_{s,h_j}^{X,\alpha_j}$ if and only if $P'_j \subseteq \text{Rem}[S]_{s,h'_j}^{X,\alpha_j}$.

Proof of (w). (\subseteq): Consider $\ell \in \text{Rem}[S]_{s,h_j}^{X,\alpha_j} \setminus (R_j \cup P_j \cup \{l_{pre}\})$. By definition of $\text{Rem}[S]_{s,h_j}^{X,\alpha_j}$, $\ell \in \text{dom}(h_j)$ and ℓ does not belong to any of the sets

$$\text{Path}[S]_{s,h_j}^X(t'), \quad \text{Pred}[S]_{s,h_j}^X(x), \quad [\text{Cycl}[S]_{s,h_j}^X(\beta)]^\flat, \quad [\uparrow\text{Cycl}[S]_{s,h_j}^X]^\flat,$$

where $t' \in T[S]^X$ and $x \in X$, $\beta \in [1, \alpha_j]$. Moreover, $\ell \notin R_{3-j}$ and $\ell \notin P_{3-j}$, as R_{3-j} and P_{3-j} contain locations in $\text{dom}(h_{3-j})$. Directly from the definition of these sets, together with $\ell \notin R_j \cup P_j \cup \{l_{pre}\}$, this allows us to conclude that $\ell \notin \text{Path}[S]_{s,h}^X(t)$. By definition of \widehat{h}_j , we derive $\ell \in \text{dom}(\widehat{h}_j)$. By definition of h'_j , we conclude that $\ell \in \text{dom}(h'_j)$ and $\ell \notin \text{Path}[S]_{s,h'}^X(t)$. In order to show that $\ell \in \text{Rem}[S]_{s,h'_j}^{X,\alpha_j}$, thus concluding the proof, we prove that ℓ does not belong to any of the sets

$$\text{Path}[S]_{s,h'_j}^X(t'), \quad \text{Pred}[S]_{s,h'_j}^X(x), \quad [\text{Cycl}[S]_{s,h'_j}^X(\beta)]^\flat, \quad [\uparrow\text{Cycl}[S]_{s,h'_j}^X]^\flat,$$

where $t' \in T[S]^X$ and $x \in X$, $\beta \in [1, \alpha_j]$.

* For every $t' \in T[S]^X$, $\ell \notin \text{Path}[S]_{s,h'_j}^X(t')$.

Ad absurdum, suppose $\ell \in \text{Path}[S]_{s,h'_j}^X(t')$. Since $\ell \in \text{dom}(h_j)$, by (A) we conclude that $\ell \neq \llbracket t' \rrbracket_{s,h'_j}^X$. Therefore, h'_j witnesses a non-empty path going from $\llbracket t' \rrbracket_{s,h'_j}^X$, and moreover $\ell \notin \text{Lab}[S]_{s,h'_j}^X$. Consider a variable $x \in X$ that is used to write the term t' . From the definition of $\llbracket \cdot \rrbracket_{s,h'_j}^X$, h'_j witnesses a (possibly empty) path going from $s(x)$ to $\llbracket t' \rrbracket_{s,h'_j}^X$. We deduce that h'_j witnesses a non-empty path going from $s(x)$ to ℓ . However, by (d), this implies that h_j witnesses a non-empty path going from $s(x)$ to ℓ . As $\ell \in \text{dom}(h_j)$, this implies that there is $t'' \in T[S]^X$ such that $\ell \in \text{Path}[S]_{s,h_j}^X(t'')$: a contradiction. Thus, $\ell \notin \text{Path}[S]_{s,h'_j}^X(t')$.

* For every $x \in X$, $\ell \notin \text{Pred}[S]_{s,h}^X(x)$.

Directly from (o) and as $\ell \neq l'_{pre} \in \text{Path}[S]_{s,h'_j}^X(t)$.

* For every $\beta \in [1, \alpha_j]$, $\ell \notin [\text{Cycl}[S]_{s,h'_j}^X(\beta)]^\flat$.

Directly from (D).

* $\ell \notin [\uparrow\text{Cycl}[S]_{s,h'_j}^X]^\flat$.

Ad absurdum, suppose there is $L \in \uparrow\text{Cycl}[S]_{s,h'_j}^{X,\alpha_j}$ such that $\ell \in L$. We show that this is contradictory, as it implies $\ell \in [\uparrow\text{Cycl}[S]_{s,h'_j}^{X,\alpha_j}]^\flat$. If $L \cap \text{Path}[S]_{s,h'}^X(t')$ is empty, then this holds directly from (s). Otherwise, by (§2) we have

$$(L \setminus \text{Path}[S]_{s,h'}^X(t)) \cup \text{Path}[S]_{s,h}^X(t) \in \uparrow\text{Cycl}[S]_{s,h_j}^{X,\alpha_j},$$

which allows us again to derive $\ell \in [\uparrow\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}]^\flat$, as $\ell \notin \text{Path}[\mathcal{S}]_{s,h'}^{\mathbf{X}}(\mathbf{t})$. Therefore, $\ell \notin L$.

(\supseteq): Symmetrical to the other direction.

Proof of (x). We show the inclusion $R_j \subseteq \text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}$. The inclusion $R'_j \subseteq \text{Rem}[\mathcal{S}]_{s,h'_j}^{\mathbf{X},\alpha_j}$ is proved analogously. Consider $\ell \in R_j$. By definition, $\ell \in \text{dom}(h_j)$. Moreover,

- * for every $\mathbf{x} \in \mathbf{X}$, $\ell \notin \text{Pred}[\mathcal{S}]_{s,h_j}^{\mathbf{X}}(\mathbf{x})$.

Ad absurdum, assume the contrary. There is $\mathbf{x} \in \mathbf{X}$ such that $h_j(\ell) = s(\mathbf{x})$. Since $s(\mathbf{x})$ is a labelled location and $\ell \in \text{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{t})$, we conclude that $s(\mathbf{x}) = \text{sby}_{s,h}^{\mathbf{X}}(\mathbf{t})$. However, this implies $\ell = l_{\text{pre}}$ directly from the definition of l_{pre} . This is contradictory, as R_j and $\{l_{\text{pre}}\}$ are disjoint. Thus, for every $\mathbf{x} \in \mathbf{X}$, $\ell \notin \text{Pred}[\mathcal{S}]_{s,h_j}^{\mathbf{X}}(\mathbf{x})$.

- * For every $\mathbf{t}' \in T[\mathcal{S}]^{\mathbf{X}}$, $\ell \notin \text{Path}[\mathcal{S}]_{s,h_j}^{\mathbf{X}}(\mathbf{t}')$.

Directly from (j).

- * ℓ does not belong neither to $[\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{X}}(\beta)]^\flat$ ($\beta \in [1, \alpha_j]$) nor to $[\uparrow\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}]^\flat$.

More generally, we show that ℓ does not belong to a cycle. *Ad absurdum*, suppose L to be a minimal set of locations describing a cycle in h_j , such that $\ell \in L$. As it is minimal and it describes a cycle $L \subseteq \text{dom}(h_j)$. From $h_j \subseteq h$, we conclude that L is a minimal set of locations describing a cycle in h . From $\ell \in L \cap \text{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{t})$ and by (u), we have $\text{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{t}) \subseteq L$. Therefore, $\text{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{t}) \subseteq \text{dom}(h_j)$. This means that $R_{3-j} = \emptyset$. By definition $e_{P_j} \in R_{3-j} \cup l_{\text{pre}}$, and therefore $e_{P_j} = l_{\text{pre}}$. However, by definition of P_j and l_{pre} , this implies that P_j and $\{l_{\text{pre}}\}$ partition $\text{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{t})$, and so $R_j = \emptyset$: a contradiction. Therefore, ℓ does not belong to a cycle.

By definition of $\text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}$, we conclude that $\ell \in \text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}$.

Proof of (y). (\Rightarrow): Suppose $l_{\text{pre}} \in \text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}$. Since we are assuming $l_{\text{pre}} \in \text{dom}(h_1)$, $j = 1$.

By definition of h'_1 , $l'_{\text{pre}} \in \text{dom}(h'_1)$. As in the previous two proofs, in order to show that $\ell \in \text{Rem}[\mathcal{S}]_{s,h'_j}^{\mathbf{X},\alpha_j}$ we prove that l'_{pre} does not belong to any of the sets:

$$\text{Path}[\mathcal{S}]_{s,h'_j}^{\mathbf{X}}(\mathbf{t}'), \quad \text{Pred}[\mathcal{S}]_{s,h'_j}^{\mathbf{X}}(\mathbf{x}), \quad [\text{Cycl}[\mathcal{S}]_{s,h'_j}^{\mathbf{X}}(\beta)]^\flat, \quad [\uparrow\text{Cycl}[\mathcal{S}]_{s,h'_j}^{\mathbf{X},\alpha_j}]^\flat.$$

All the proofs essentially use the fact that $l_{\text{pre}} \in \text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}$, which implies that l_{pre} does not belong to the sets:

$$\text{Path}[\mathcal{S}]_{s,h_j}^{\mathbf{X}}(\mathbf{t}'), \quad \text{Pred}[\mathcal{S}]_{s,h_j}^{\mathbf{X}}(\mathbf{x}), \quad [\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{X}}(\beta)]^\flat, \quad [\uparrow\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}]^\flat.$$

In particular,

- * From (k), for every $\mathbf{t} \in T[\mathcal{S}]^{\mathbf{X}}$, $l'_{\text{pre}} \notin \text{Path}[\mathcal{S}]_{s,h'_j}^{\mathbf{X}}(\mathbf{t}')$.
- * From (p), for every $\mathbf{x} \in \mathbf{X}$, $l'_{\text{pre}} \notin \text{Pred}[\mathcal{S}]_{s,h'_j}^{\mathbf{X}}(\mathbf{x})$.
- * From (D), for every $\beta \in [1, \alpha_j]$, $l'_{\text{pre}} \notin [\text{Cycl}[\mathcal{S}]_{s,h'_j}^{\mathbf{X}}(\beta)]^\flat$.
- * *Ad absurdum*, suppose there is a set $L \in \uparrow\text{Cycl}[\mathcal{S}]_{s,h'_j}^{\mathbf{X},\alpha_j}$ such that $l'_{\text{pre}} \in L$. From $L \cap \text{Path}[\mathcal{S}]_{s,h'}^{\mathbf{X}}(\mathbf{t}) \neq \emptyset$ and (§2), $(L \setminus \text{Path}[\mathcal{S}]_{s,h'}^{\mathbf{X}}(\mathbf{t})) \cup \text{Path}[\mathcal{S}]_{s,h}^{\mathbf{X}}(\mathbf{t}) \in \uparrow\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}$. However, this implies $l_{\text{pre}} \in [\uparrow\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{X},\alpha_j}]^\flat$, a contradiction. So, $\ell \notin [\uparrow\text{Cycl}[\mathcal{S}]_{s,h'_j}^{\mathbf{X},\alpha_j}]^\flat$.

(\Leftarrow): Symmetrical to the other direction.

Proof of (z). We start by showing that $P_j \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ is either \emptyset or P_j . The proof that $(P'_j \cap \text{Rem}[s]_{s,h'_j}^{X,\alpha_j})$ is either \emptyset or P'_j) is analogous. In particular, we show that if there is $\ell \in P_j \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ then $P_j \subseteq \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$. So, let us assume $\ell \in P_j \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ and consider $\ell' \in P_j$. By definition of P_j , $\ell' \in \text{dom}(h_j)$. Afterwards, $\ell' \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ holds directly from the three statements below:

- * for every $t' \in T[s]^X$, $\ell' \notin \text{Path}[s]_{s,h_j}^X(t')$

Ad absurdum, suppose $\ell' \in \text{Path}[s]_{s,h_j}^X(t')$, for some $t' \in T[s]^X$. Since $\ell' \in P_j$ and by (1), we conclude that $P_j \subseteq \text{Path}[s]_{s,h_j}^X(t')$. However, this implies $\ell \in \text{Path}[s]_{s,h_j}^X(t')$, in contradiction with $\ell \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$.

- * for every $x \in X$, $\ell' \notin \text{Pred}[s]_{s,h_j}^X(x)$.

Ad absurdum, suppose $\ell' \in \text{Pred}[s]_{s,h_j}^X(x)$, for some $x \in X$. By $h_j \subseteq h$, $h(\ell') = s(x)$. Since $\ell' \in \text{Path}[s]_{s,h}^X(t)$, by definition of l_{pre} , we conclude that $\ell' = l_{\text{pre}}$. However, this is contradictory, as $\ell' \in P_j$ but P_j and $\{l_{\text{pre}}\}$ are disjoint. Therefore, for every $x \in X$, $\ell' \notin \text{Pred}[s]_{s,h_j}^X(x)$.

- * ℓ' does not belong neither to $[\text{Cycl}[s]_{s,h_j}^X(\beta)]^\flat$ ($\beta \in [1, \alpha_j]$) nor to $[\uparrow\text{Cycl}[s]_{s,h_j}^X]^\flat$.

More generally, we show that ℓ' does not belong to an unlabelled cycle. *Ad absurdum*, suppose L to be a minimal set of locations describing a cycle in h_j , such that $\ell' \in L$ and L does not contain labelled locations in $\text{Lab}[s]_{s,h_j}^X$. Since $h_j \subseteq h$, this implies that L is a minimal set of locations describing a cycle in h . From $\ell' \in \text{Path}[s]_{s,h}^X(t)$ and by (μ), we conclude that $\text{Path}[s]_{s,h}^X(t) \subseteq L$. So, $\ell \in L$. However, since L is a minimal set describing an unlabelled cycle in h_j , we conclude that ℓ belongs to either $[\text{Cycl}[s]_{s,h_j}^X(\beta)]^\flat$, for some $\beta \in [1, \beta]$, or $\ell \in [\uparrow\text{Cycl}[s]_{s,h_j}^X]^\flat$. As this contradicts the fact that $\ell \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$, we conclude that ℓ' does not belong to an unlabelled cycle.

Let us now discuss the double implication

$$P_j \subseteq \text{Rem}[s]_{s,h_j}^{X,\alpha_j} \text{ if and only if } P'_j \subseteq \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}.$$

We show the left-to-right direction. The right-to-left direction follows symmetrically.

(\Rightarrow): Suppose $P_j \subseteq \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$. Let us consider $\ell' \in P'_j$. This means that $P'_j \neq \emptyset$ and so, by (4), $P_j \neq \emptyset$. Thus, let us also consider a location $\ell \in P_j$ which, by hypothesis, belongs to $\text{Rem}[s]_{s,h_j}^{X,\alpha_j}$. Directly from $\ell' \in P'_j$, $\ell' \in \text{dom}(h_j)$. Afterwards, $\ell' \in \text{Rem}[s]_{s,h'_j}^{X,\alpha'_j}$ holds directly from the three statements below, concluding the proof:

- * for every $t' \in T[s]^X$, $\ell' \notin \text{Path}[s]_{s,h'_j}^X(t')$.

Ad absurdum, suppose $\ell' \in \text{Path}[s]_{s,h'_j}^X(t')$, for some $t' \in T[s]^X$. As $\ell' \in P'_j$ and by (1), we conclude that $P'_j \subseteq \text{Path}[s]_{s,h'_j}^X(t')$ and $P_j \subseteq \text{Path}[s]_{s,h_j}^X(t')$.

However, this implies $\ell \in \text{Path}[s]_{s,h_j}^X(t')$, in contradiction with $\ell \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$. Thus, for every $t' \in T[s]^X$, $\ell' \notin \text{Path}[s]_{s,h'_j}^X(t')$.

- * for every $x \in X$, $\ell' \notin \text{Pred}[s]_{s,h'_j}^X(x)$.

We already proved an analogous statement above (w.r.t. (s, h'_j)). Briefly, it cannot be that $\ell' \in \text{Pred}[s]_{s,h'_j}^X(x)$, for some $x \in X$, as otherwise we would be able to derive $\ell' = l_{\text{pre}}$, in contradiction with $\ell' \in P'_j$.

- * ℓ' does not belong to neither $[\text{Cycl}[s]_{s,h'_j}^X(\beta)]^\flat$ ($\beta \in [1, \alpha_j]$) nor to $[\uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}]^\flat$.

More generally, we show that ℓ' does not belong to an unlabelled cycle. *Ad absurdum*, suppose ℓ to be a minimal set of locations describing a cycle in h'_j , such that $\ell' \in L$ and L does not contain labelled locations in $\text{Lab}[s]_{s,h'_j}^X$. Since $h'_j \subseteq h'$, this implies that L is a minimal set of locations describing a cycle in h' . From $\ell' \in \text{Path}[s]_{s,h'}^X(t)$ and by (μ), we derive $\text{Path}[s]_{s,h'}^X(t) \subseteq L$. Now, it cannot be that L belongs to $\text{Cycl}[s]_{s,h'_j}^X(\beta)$, for some $\beta \in [1, \alpha_j]$, as it would imply $\text{card}(\text{Path}[s]_{s,h'}^X(t)) \leq \beta$, in contradiction with the assumption $\text{card}(\text{Path}[s]_{s,h'}^X(t)) \geq \mathcal{S}(\alpha_j) > \alpha_j$.

Therefore, $L \in \uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}$. From $\text{Path}[s]_{s,h'}^X(t) \subseteq [\uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}]^\flat$ and by (t), we conclude that $\text{Path}[s]_{s,h'}^X(t) \subseteq [\uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}]^\flat$. However, this allows us to derive $\ell \in [\uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}]^\flat$, in contradiction with $\ell \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$. Thus, ℓ' does not belong to an unlabelled cycle.

At last, we are ready to show (F), essentially concluding the proof.

Proof of (u). By definition of the sets P_1 , P_2 , R_1 , R_2 and $\{l_{pre}\}$, together with the assumption $l_{pre} \in \text{dom}(h_1)$, we have:

$$\text{Path}[s]_{s,h}^X(t) \cap \text{dom}(h_1) = P_1 \cup R_1 \cup \{l_{pre}\}, \quad \text{Path}[s]_{s,h}^X(t) \cap \text{dom}(h_2) = P_2 \cup R_2.$$

Similarly, by definition of h'_1 and h'_2 ,

$$\text{Path}[s]_{s,h'}^X(t) \cap \text{dom}(h'_1) = P'_1 \cup R'_1 \cup \{l'_{pre}\}, \quad \text{Path}[s]_{s,h'}^X(t) \cap \text{dom}(h'_2) = P'_2 \cup R'_2.$$

Let $j \in \{1, 2\}$. If $j = 1$, let T_j and T'_j be $\{l_{pre}\}$ and $\{l'_{pre}\}$, Else, let $T_j = T'_j = \emptyset$. We recall that $\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \subseteq \text{dom}(h_j)$ and $\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \subseteq \text{dom}(h'_j)$, which allows us to obtain the following equalities:

$$\begin{aligned} \text{Rem}[s]_{s,h_j}^{X,\alpha_j} &= (\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus \text{Path}[s]_{s,h}^X(t)) \cup (\text{Path}[s]_{s,h}^X(t) \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}) \\ &= (\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus (P_j \cup R_j \cup T_j)) \cup ((P_j \cup R_j \cup T_j) \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}), \\ \text{Rem}[s]_{s,h'_j}^{X,\alpha_j} &= (\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus \text{Path}[s]_{s,h'}^X(t)) \cup (\text{Path}[s]_{s,h'}^X(t) \cap \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}) \\ &= (\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus (P'_j \cup R'_j \cup T'_j)) \cup ((P'_j \cup R'_j \cup T'_j) \cap \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}). \end{aligned}$$

Above, all unions are between disjoint sets. Let us use the following abbreviations:

$$\begin{aligned} m &= \text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus (P_j \cup R_j \cup T_j)), & p &= \text{card}(P_j \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}), \\ r &= \text{card}(R_j \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}), & t &= \text{card}(T_j \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}), \\ m' &= \text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus (P'_j \cup R'_j \cup T'_j)), & p' &= \text{card}(P'_j \cap \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}), \\ r' &= \text{card}(R'_j \cap \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}), & t' &= \text{card}(T'_j \cap \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}). \end{aligned}$$

From the previous equalities above, we derive

$$\text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j}) = m + p + r + t, \quad \text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j}) = m' + p' + r' + t'.$$

Afterwards, $\min(\text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j}), \mathcal{R}(\alpha_j)) = \min(\text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j}), \mathcal{R}(\alpha_j))$ follows from the four statements below:

$$* \ m = m'.$$

Directly from (w).

$$* \ \min(p, \mathcal{R}(\alpha_j)) = \min(p', \mathcal{R}(\alpha_j)),$$

From (z) and the property (4), as $\mathcal{S}(\alpha_j) \geq \mathcal{R}(\alpha_j)$ (see (\star_1) and (\star_2)).

$$* \min(r, \mathcal{R}(\alpha_j)) = \min(r', \mathcal{R}(\alpha_j)).$$

From (x) and the property (5).

$$* t = t'.$$

Directly from (y).

Proof of (v). We consider the two equalities derived in the proof of (u):

$$\text{Rem}[s]_{s,h_j}^{X,\alpha_j} = (\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus (P_j \cup R_j \cup T_j)) \cup ((P_j \cup R_j \cup T_j) \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}),$$

$$\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} = (\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus (P'_j \cup R'_j \cup T'_j)) \cup ((P'_j \cup R'_j \cup T'_j) \cap \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}).$$

where, if $j = 1$, then T_j and T'_j are $\{l_{pre}\}$ and $\{l'_{pre}\}$. Otherwise, $T_j = T'_j = \emptyset$.

(\Rightarrow) : Suppose $s(u) \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$. Following the equalities above, we divide the proof in four cases:

case: $s(u) \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus (P_j \cup R_j \cup T_j)$. By (w), $s(u) \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus (P'_j \cup R'_j \cup T'_j)$.

case: $s(u) \in P_j \cap \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$. From (z), $P_j \subseteq \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ and $P'_j \subseteq \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$. From the property (7), $s(u)$ belongs to P'_j , and thus it belongs to $\text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

case: $s(u) \in R_j$. From (x), $R'_j \subseteq \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$. From the property (6) of the construction, $s(u)$ belongs to R'_j , and thus it belongs to $\text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

case: $s(u) \in T_j$. By definition of T_j , $s(u) = l_{pre}$. Ad absurdum, assume $s(u) \neq l'_{pre}$.

By definition of l_{pre} and l'_{pre} , we conclude that there is $t' \in T[s]^X$ such that

$$* (s, h) \models u \in \text{sees}_X(t, t') \geq (1, 1) \text{ and } (s, h) \not\models u \in \text{sees}_X(t, t') \geq (1, 2),$$

$$* \text{If } (s, h') \models u \in \text{sees}_X(t, t') \geq (1, 1) \text{ then } (s, h') \models u \in \text{sees}_X(t, t') \geq (1, 2).$$

This contradicts $(s, h) \approx_{X,\alpha}^S (s, h')$. Thus, $s(u) = l'_{pre}$. By (y), $s(u) \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

We conclude that $s(u) \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

(\Leftarrow) : Symmetrical to the other direction. The second case relies on (8) instead of (7).

The properties (A)–(F) lead directly to $(s, h_j) \approx_{X,\alpha_j}^S (s', h'_j)$ with the same case analysis provided at the end of the proof of Lemma 5.39. Therefore, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h')$. \square

Once Lemma 5.40 is established, we conclude that the core formulae $\text{Core}[s](X, \alpha)$ enjoy the $*$ -simulation property. As in the case of the $*$ -simulation property of the weak fragment (Lemma 5.20), this is done by considering two memory states $(s, h) \approx_{X,\alpha}^S (s', h)$ and build a chain of hops going from (s, h) to (s', h') . This chain of hops is built by carrying out an induction on the number of sets in the partition having different cardinalities with respect to the two memory states. Each hop corresponds to one of the cases in Lemma 5.40 or Lemma 5.39.

Lemma 5.41 (s : $*$ -simulation). $\approx_{X,\alpha}^S \subseteq \leftrightarrow_{X,\alpha}^S$.

With no surprises, the proof follows closely the one of Lemma 5.20. It is given in Appendix C.

5.5.4 Step IV: \exists -simulation.

We show that the core formulae $\text{Core}[s](X, \alpha)$ enjoys the \exists -simulation property. As done for Lemma 5.23, this result is proved by checking how the satisfaction of formulae in $\text{Obs}[s](X, \alpha)$ changes as the quantified variable u is reassigned.

Lemma 5.42 ($s : \exists$ -simulation). Suppose $(s, h) \approx_{\mathbf{X}, \alpha}^{\mathcal{S}} (s', h')$. For every location $\ell_1 \in \text{LOC}$ there is a location $\ell_2 \leq \text{maxval}_{\mathbf{X}}(s', h') + 1$ such that $(s[\mathbf{u} \leftarrow \ell_1], h) \approx_{\mathbf{X}, \alpha}^{\mathcal{S}} (s'[\mathbf{u} \leftarrow \ell_2], h')$.

Proof. As it was the case for the weak fragment, we notice that the sets in Definition 5.29 (i.e. predecessors sets, paths sets, etc.) do not depend on the location assigned to the variable name $\mathbf{u} \notin \mathbf{X}$. More precisely, for every memory state (\hat{s}, \hat{h}) and location $\hat{\ell}$ the following equivalences hold (where $\mathbf{x} \in \mathbf{X}$, $\ell \in \text{Lab}[s]_{\hat{s}, \hat{h}}^{\mathbf{x}}$ and $\beta \in [1, \alpha]$):

$$\begin{array}{lll} \text{Lab}[s]_{\hat{s}, \hat{h}}^{\mathbf{x}} & = & \text{Lab}[s]_{\hat{s}[\mathbf{u} \leftarrow \hat{\ell}], \hat{h}}^{\mathbf{x}}, \\ \text{Path}[s]_{\hat{s}, \hat{h}}^{\mathbf{x}}(\ell) & = & \text{Path}[s]_{\hat{s}[\mathbf{u} \leftarrow \hat{\ell}], \hat{h}}^{\mathbf{x}}(\ell), \\ \uparrow\text{Cycl}[s]_{\hat{s}, \hat{h}}^{\mathbf{x}, \alpha} & = & \uparrow\text{Cycl}[s]_{\hat{s}[\mathbf{u} \leftarrow \hat{\ell}], \hat{h}}^{\mathbf{x}, \alpha}, \end{array} \quad \begin{array}{lll} \text{Pred}[s]_{\hat{s}, \hat{h}}^{\mathbf{x}}(\mathbf{x}) & = & \text{Pred}[s]_{\hat{s}[\mathbf{u} \leftarrow \hat{\ell}], \hat{h}}^{\mathbf{x}}(\mathbf{x}), \\ \text{Cycl}[s]_{\hat{s}, \hat{h}}^{\mathbf{x}}(\beta) & = & \text{Cycl}[s]_{\hat{s}[\mathbf{u} \leftarrow \hat{\ell}], \hat{h}}^{\mathbf{x}}(\beta), \\ \text{Rem}[s]_{\hat{s}, \hat{h}}^{\mathbf{x}, \alpha} & = & \text{Rem}[s]_{\hat{s}[\mathbf{u} \leftarrow \hat{\ell}], \hat{h}}^{\mathbf{x}, \alpha}. \end{array}$$

We denote these equivalences by (Inv-u). Directly from them, we notice that for every core formula φ in $\text{Sk}[s](\mathbf{X}, \alpha)$ and $\ell_1, \ell_2 \in \text{LOC}$, we have

$$\begin{aligned} (s[\mathbf{u} \leftarrow \ell_1], h) \models \varphi, \quad & \text{iff } (s, h) \models \varphi, & (\text{by (Inv-u)}) \\ \text{iff } (s', h') \models \varphi, & & (\text{by } (s, h) \approx_{\mathbf{X}, \alpha}^{\mathcal{S}} (s', h')) \\ \text{iff } (s'[\mathbf{u} \leftarrow \ell_2], h') \models \varphi. & & (\text{by (Inv-u)}) \end{aligned}$$

Therefore, in order to prove the result it is sufficient to show that for every $\ell_1 \in \text{LOC}$ there is $\ell_2 \leq \text{maxval}_{\mathbf{X}}(s', h') + 1$ such that the memory states $(s[\mathbf{u} \leftarrow \ell_1], h)$ and $(s'[\mathbf{u} \leftarrow \ell_2], h')$ agree on the satisfaction of every core formula in $\text{Obs}[s](\mathbf{X}, \alpha)$. The choice for ℓ_2 depends on whether ℓ_1 is a labelled location and on whether it belongs to one of the sets in Definition 5.29:

case: $\ell_1 \in \text{Lab}[s]_{s, h}^{\mathbf{x}}$. Let $\mathbf{t} \in T[s]^{\mathbf{x}}$ be such that $\llbracket \mathbf{t} \rrbracket_{s, h}^{\mathbf{x}} = \ell_1$. By $(s, h) \approx_{\mathbf{X}, \alpha}^{\mathcal{S}} (s', h')$, $\llbracket \mathbf{t} \rrbracket_{s', h'}^{\mathbf{x}}$ is defined. Consider $\ell_2 = \llbracket \mathbf{t} \rrbracket_{s', h'}^{\mathbf{x}}$. If \mathbf{t} is syntactically equal to some $\mathbf{x} \in \mathbf{X}$, then $\ell_2 \in s'(\mathbf{X})$. If instead \mathbf{t} is syntactically equal to $\mathbf{m}(\mathbf{x}, \mathbf{y})$ or $\mathbf{e}(\mathbf{x})$ (for some $\mathbf{x}, \mathbf{y} \in \mathbf{X}$), then $\ell_2 \in \text{ran}(h')$. Therefore, $\ell_2 \leq \text{maxval}_{\mathbf{X}}(s', h') + 1$. We show that the two memory states $(s[\mathbf{u} \leftarrow \ell_1], h)$ and $(s'[\mathbf{u} \leftarrow \ell_2], h')$ satisfy the same core formulae in $\text{Obs}[s](\mathbf{X}, \alpha)$. Given a formula φ in

$$\begin{cases} \mathbf{u} \in \text{loop}_{\mathbf{X}}^{\mathcal{S}}(\beta), \mathbf{u} \in \uparrow\text{loop}_{\mathbf{X}, \alpha}^{\mathcal{S}}, & \overleftarrow{\beta} \in \left[1, \frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1\right] \\ \mathbf{u} \in \text{sees}_{\mathbf{X}}(\mathbf{t}_1, \mathbf{t}_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta}), & \overrightarrow{\beta} \in \left[1, \frac{1}{2}\alpha(\alpha+3)\right], \beta \in [1, \alpha] \\ \mathbf{u} \in \text{pred}_{\mathbf{X}}^{\mathcal{S}}(\mathbf{x}), \mathbf{u} \in \text{rem}_{\mathbf{X}, \alpha}^{\mathcal{S}}, & \mathbf{x} \in \mathbf{X}, \mathbf{t}_1, \mathbf{t}_2 \in T[s]^{\mathbf{x}} \end{cases}$$

by (Inv-u) we conclude that $(s[\mathbf{u} \leftarrow \ell_1], h) \not\models \varphi$ and $(s'[\mathbf{u} \leftarrow \ell_2], h') \not\models \varphi$. Indeed, all the formulae in this set require that \mathbf{u} corresponds to an unlabelled location. Now, let us consider a core formula of the form $\mathbf{u} = \mathbf{t}'$, where $\mathbf{t}' \in T[w]^{\mathbf{x}}$. We have

$$\begin{aligned} (s[\mathbf{u} \leftarrow \ell_1], h) \models \mathbf{u} = \mathbf{t}', & & \\ \Leftrightarrow \ell_1 = \llbracket \mathbf{t}' \rrbracket_{s[\mathbf{u} \leftarrow \ell_1], h}^{\mathbf{x}}, & & (\text{by definition of } \models) \\ \Leftrightarrow \ell_1 = \llbracket \mathbf{t}' \rrbracket_{s, h}^{\mathbf{x}} = \llbracket \mathbf{t} \rrbracket_{s, h}^{\mathbf{x}}, & & (\text{by hypothesis } \ell_1 = \llbracket \mathbf{t} \rrbracket_{s, h}^{\mathbf{x}} \text{ and } \llbracket \mathbf{t}' \rrbracket_{s[\mathbf{u} \leftarrow \ell_1], h}^{\mathbf{x}} = \llbracket \mathbf{t}' \rrbracket_{s, h}^{\mathbf{x}}) \\ \Leftrightarrow \ell_2 = \llbracket \mathbf{t}' \rrbracket_{s', h'}^{\mathbf{x}} = \llbracket \mathbf{t} \rrbracket_{s', h'}^{\mathbf{x}}, & & (\text{from } \ell_2 = \llbracket \mathbf{t} \rrbracket_{s', h'}^{\mathbf{x}} \text{ and } (s, h) \approx_{\mathbf{X}, \alpha}^{\mathcal{S}} (s', h')) \\ \Leftrightarrow \ell_2 = \llbracket \mathbf{t}' \rrbracket_{s'[\mathbf{u} \leftarrow \ell_2], h'}^{\mathbf{x}}, & & \\ & & (\text{from } \llbracket \mathbf{t}' \rrbracket_{s'[\mathbf{u} \leftarrow \ell_1], h'}^{\mathbf{x}} = \llbracket \mathbf{t}' \rrbracket_{s', h'}^{\mathbf{x}}. \text{ The right-to-left direction also uses } \ell_2 = \llbracket \mathbf{t}' \rrbracket_{s', h'}^{\mathbf{x}}) \\ \Leftrightarrow (s'[\mathbf{u} \leftarrow \ell_2], h') \models \mathbf{u} = \mathbf{t}'. & & (\text{by definition of } \models) \end{aligned}$$

case: $\ell_1 \notin \text{Lab}[s]_{s, h}^{\mathbf{x}}$ and there is $\mathbf{t} \in T[s]^{\mathbf{x}}$ s.t. $\ell_1 \in \text{Path}[s]_{s, h}^{\mathbf{x}}(\mathbf{t})$. In this case ℓ_1 belongs to the minimal path in h going from $\llbracket \mathbf{t} \rrbracket_{s, h}^{\mathbf{x}}$ to $\text{sby}_{s, h}^{\mathbf{x}}(\mathbf{t})$, and it is different to both these two labelled locations. Roughly speaking, in this case we mainly focus on defining the location ℓ_2

so that it belongs to $\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$ and it is such that the memory states $(s[\mathbf{u} \leftarrow \ell_1], h)$ and $(s'[\mathbf{u} \leftarrow \ell_2], h')$ satisfy the same core formulae of the form $\mathbf{u} \in \text{sees}_X(\mathbf{t}_1, \mathbf{t}_2) \geq (\beta_1, \beta_2)$, where $\beta_1 \in [1, \frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1]$ and $\beta_2 \in [1, \frac{1}{2}\alpha(\alpha+3)]$. We write $\mathcal{S}_{\text{left}}(\alpha)$ and $\mathcal{S}_{\text{right}}(\alpha)$ for $\frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1$ and $\frac{1}{2}\alpha(\alpha+3)$, respectively, i.e. the two upper bounds for β_1 and β_2 . Following the semantics of the core formulae of the form $\mathbf{u} \in \text{sees}_X(\mathbf{t}_1, \mathbf{t}_2) \geq (\beta_1, \beta_2)$, we consider the lengths δ_{left} and δ_{right} such that

$$\delta_{\text{left}} + \delta_{\text{right}} = \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t})), \quad h^{\delta_{\text{left}}}([\mathbf{t}]_{s,h}^X) = \ell_1, \quad h^{\delta_{\text{right}}}(\ell_1) = \text{sby}_{s,h}^X(\mathbf{t}).$$

Informally, δ_{left} corresponds to the length of the path inside h going from $[\mathbf{t}]_{s,h}^X$ to ℓ_1 , whereas δ_{right} corresponds to the length of the path inside h going from ℓ_1 to $\text{sby}_{s,h}^X(\mathbf{t})$. As ℓ_1 is an unlabelled location, δ_{left} and δ_{right} are at least 1. We define the location ℓ_2 following the procedure below:

```

if  $\delta_{\text{left}} < \mathcal{S}_{\text{left}}(\alpha)$  then
     $\ell_2 \leftarrow h'^{\delta_{\text{left}}}([\mathbf{t}]_{s',h'}^X),$ 
else if  $\delta_{\text{right}} < \mathcal{S}_{\text{right}}(\alpha)$  then
    let  $\ell_2$  be the location in  $\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$  such that  $h'^{\delta_{\text{right}}}(\ell_2) = \text{sby}_{s',h'}^X(\mathbf{t})$ ,
    else (i.e.  $\delta_{\text{left}} \geq \mathcal{S}_{\text{left}}(\alpha)$  and  $\delta_{\text{right}} \geq \mathcal{S}_{\text{right}}(\alpha)$ )
         $\ell_2 \leftarrow h'^{\mathcal{S}_{\text{left}}(\alpha)}([\mathbf{t}]_{s',h'}^X).$ 

```

We now show that this procedure correctly defines ℓ_2 as a location in $\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$, which directly implies $\ell_2 \leq \text{maxval}_X(s', h') + 1$ by definition of maximum value. From $(s, h) \approx_{X,\alpha}^S (s', h')$ it holds that (s, h) and (s', h') satisfy the same core formulae of the form $\text{sees}_X(\mathbf{t}_1, \mathbf{t}_2) \geq \beta$, where $\mathbf{t}_1, \mathbf{t}_2 \in \mathbf{T}[\mathcal{S}]^X$ and $\beta \in [1, \frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3)]$. Thus,

$$\begin{aligned} & \min(\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t}), \frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3)) \\ &= \min(\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t}), \frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3)) \end{aligned} \tag{†}$$

As one can show that $\frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3) = \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha)$, if $\delta_{\text{left}} < \mathcal{S}_{\text{left}}(\alpha)$ holds (first case of the procedure), then $\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t})$ contains more than δ_{left} locations. From (†) we conclude that $\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$ describes a path in h' going from $[\mathbf{t}]_{s',h'}^X$ to $\text{sby}_{s',h'}^X(\mathbf{t})$ and of length greater than δ_{left} . Thus, ℓ_2 is well-defined and it belongs to $\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$. Similarly, if the second case of the procedure is applied, then the path in h' going from $[\mathbf{t}]_{s',h'}^X$ to $\text{sby}_{s',h'}^X(\mathbf{t})$ more than δ_{right} locations, which again means that ℓ_2 belongs to $\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$. Lastly, in the third case of the procedure, $\delta_{\text{left}} \geq \mathcal{S}_{\text{left}}(\alpha)$ implies that $\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$ has more than $\mathcal{S}_{\text{left}}(\alpha)$ locations. Again, $\ell_2 \in \text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$. Let us refine the analysis on the location ℓ_2 . We consider the two lengths δ'_{left} and δ'_{right} such that

$$\begin{aligned} \delta'_{\text{left}} + \delta'_{\text{right}} &= \text{card}(\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})), \\ h^{\delta'_{\text{left}}}([\mathbf{t}]_{s',h'}^X) &= \ell_2, \quad h^{\delta'_{\text{right}}}(\ell_2) = \text{sby}_{s',h'}^X(\mathbf{t}). \end{aligned}$$

As ℓ_2 belongs to $\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t})$, these two lengths exist and are uniquely defined. By (†),

$$\min(\delta_{\text{left}} + \delta_{\text{right}}, \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha)) = \min(\delta'_{\text{left}} + \delta'_{\text{right}}, \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha)). \tag{‡}$$

We show that

- I. $\min(\delta_{\text{left}}, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta'_{\text{left}}, \mathcal{S}_{\text{left}}(\alpha))$,
- II. $\min(\delta_{\text{right}}, \mathcal{S}_{\text{right}}(\alpha)) = \min(\delta'_{\text{right}}, \mathcal{S}_{\text{right}}(\alpha))$.

The proof is divided in three cases, following the procedure used to define ℓ_2 .

case: $\delta_{\text{left}} < \mathcal{S}_{\text{left}}(\alpha)$. We follow the first case of the procedure, and conclude that $\delta'_{\text{left}} = \delta_{\text{left}}$ holds. Thus, (I) trivially holds, whereas (II) is proved from (‡). Indeed, by subtracting δ_{left} on both side of the equation in (‡), and relying on the fact the sum distributes over the min function, from $\delta'_{\text{left}} = \delta_{\text{left}}$ we conclude that

$$\min(\delta_{\text{right}}, \mathcal{S}_{\text{left}}(\alpha) - \delta_{\text{left}} + \mathcal{S}_{\text{right}}(\alpha)) = \min(\delta'_{\text{right}}, \mathcal{S}_{\text{left}}(\alpha) - \delta_{\text{left}} + \mathcal{S}_{\text{right}}(\alpha)).$$

As $\mathcal{S}_{\text{left}}(\alpha) - \delta_{\text{left}} > 0$, this implies (II) directly from the fact that for every $a, b, c, d \geq 0$,

$$\min(a, b + c) = \min(d, b + c) \Rightarrow \min(a, c) = \min(d, c).$$

case: $\delta_{\text{left}} \geq \mathcal{S}_{\text{left}}(\alpha)$ and $\delta_{\text{right}} < \mathcal{S}_{\text{right}}(\alpha)$. We follow the second case of the procedure, and conclude that $\delta'_{\text{right}} = \delta_{\text{right}}$ holds. Therefore, (II) trivially holds, whereas (I) is proved from (‡) (the proof is analogous to the one described in the previous case in order to prove (II)).

case: $\delta_{\text{left}} \geq \mathcal{S}_{\text{left}}(\alpha)$ and $\delta_{\text{right}} \geq \mathcal{S}_{\text{right}}(\alpha)$. We follow the third case of the procedure, and conclude that $\delta'_{\text{right}} = \mathcal{S}_{\text{left}}(\alpha)$ holds. Thus, (I) trivially holds. From (‡), we conclude that $\delta'_{\text{left}} + \delta'_{\text{right}} \geq \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha)$. From $\delta'_{\text{right}} = \mathcal{S}_{\text{left}}(\alpha)$ we conclude that $\delta'_{\text{right}} \geq \mathcal{S}_{\text{right}}(\alpha)$, which proves (II).

Now, let us show that $(s[\mathbf{u} \leftarrow \ell_1], h)$ and $(s'[\mathbf{u} \leftarrow \ell_2], h')$ satisfy the same core formulae of $\text{Obs}[\mathcal{S}](X, \alpha)$. Notice that we have $\ell_1 \in \text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t})$ and $\ell_2 \in \text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$. Moreover, $\ell_1 \notin \text{Lab}[\mathcal{S}]_{s,h}^X$ and $\ell_2 \notin \text{Lab}[\mathcal{S}]_{s',h'}^X$. This latter statement holds directly from the definition of $\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t})$ together with the fact that $\delta_{\text{left}}, \delta_{\text{right}} \geq 1$, which in turn implies $\delta'_{\text{left}}, \delta'_{\text{right}} \geq 1$, by (I) and (II). Because of this, it is quite easy to see that both memory states do not satisfy any of the core formulae in

$$\left\{ \begin{array}{l} \mathbf{u} = \mathbf{t}_1, \mathbf{u} \in \text{loop}_X^{\mathcal{S}}(\beta), \mathbf{u} \in \uparrow \text{loop}_{X,\alpha}^{\mathcal{S}}, \\ \mathbf{u} \in \text{pred}_X^{\mathcal{S}}(x), \mathbf{u} \in \text{rem}_{X,\alpha}^{\mathcal{S}} \end{array} \middle| \beta \in [1, \alpha], x \in X \right\}.$$

leaving the satisfiability of the core formulae of the form $\mathbf{u} \in \text{sees}_X(\mathbf{t}_1, \mathbf{t}_2) \geq (\beta_1, \beta_2)$ to be checked. Let φ be the core formula $\mathbf{u} \in \text{sees}_X(\mathbf{t}_1, \mathbf{t}_2) \geq (\beta_1, \beta_2)$, where $\beta_1 \in [1, \mathcal{S}_{\text{left}}(\alpha)]$ and $\beta_2 \in [1, \mathcal{S}_{\text{right}}(\alpha)]$. We prove that

$$(s[\mathbf{u} \leftarrow \ell_1], h) \models \varphi \text{ if and only if } (s'[\mathbf{u} \leftarrow \ell_2], h') \models \varphi.$$

(\Rightarrow): By definition, there are $\delta_1 \geq \beta_1$ and $\delta_2 \geq \beta_2$ such that

$$\delta_1 + \delta_2 = \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t}_1)), \quad h^{\delta_1}([\mathbf{t}_1]_{s,h}^X) = \ell_1, \quad h^{\delta_2}(\ell_1) = [\mathbf{t}_2]_{s,h}^X.$$

Therefore, $\delta_{\text{left}} = \delta_1$ and $\delta_{\text{right}} = \delta_2$. Moreover, $[\mathbf{t}_1]_{s,h}^X = [\mathbf{t}]_{s,h}^X$ and $[\mathbf{t}_2]_{s,h}^X = \text{sby}_{s,h}^X(\mathbf{t})$. From the equisatisfaction of the core formulae of the form $\mathbf{t}' = \mathbf{t}''$, we conclude that $[\mathbf{t}_1]_{s',h'}^X = [\mathbf{t}]_{s',h'}^X$ and $[\mathbf{t}_2]_{s',h'}^X = \text{sby}_{s',h'}^X(\mathbf{t})$. Thus, δ'_{left} and δ'_{right} are such that

$$\delta'_{\text{left}} + \delta'_{\text{right}} = \text{card}(\text{Path}[\mathcal{S}]_{s',h'}^X(\mathbf{t}_1)), \quad h^{\delta'_{\text{left}}}([\mathbf{t}_1]_{s',h'}^X) = \ell_2, \quad h^{\delta'_{\text{right}}}(\ell_2) = [\mathbf{t}_2]_{s',h'}^X.$$

From the semantics of φ , in order to conclude that $(s'[\mathbf{u} \leftarrow \ell_2], h') \models \varphi$ holds it is sufficient to show that $\delta'_{\text{left}} \geq \beta_1$ and $\delta'_{\text{right}} \geq \beta_2$. As $\beta_1 \leq \mathcal{S}_{\text{left}}(\alpha)$, the inequality $\delta'_{\text{left}} \geq \beta_1$ holds directly from $\delta_1 \geq \beta_1$ and (I). As $\beta_1 \leq \mathcal{S}_{\text{right}}(\alpha)$, the inequality $\delta'_{\text{right}} \geq \beta_2$ holds directly from $\delta_2 \geq \beta_2$ and (II).

(\Leftarrow): Symmetrical to the other direction, thanks to (I) and (II).

case: there is $x \in X$ such that $\ell_1 \in \text{Pred}[\mathcal{S}]_{s,h}^X(x)$. From the equisatisfaction of the formulae in $\text{Sk}[\mathcal{S}](X, \alpha)$, the set $\text{Pred}[\mathcal{S}]_{s',h'}^X(x)$ is not empty. Consider ℓ_2 to be a location in this set. Hence $\ell_2 \in \text{dom}(h')$, which implies that $\ell_2 \leq \text{maxval}_X(s', h') + 1$. Given a formula φ in

$$\left\{ \begin{array}{l} u = t_1, u \in \text{loop}_{X}^S(\beta), u \in \uparrow \text{loop}_{X,\alpha}^S, \\ u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta}), \\ u \in \text{rem}_{X,\alpha}^S \end{array} \right| \begin{array}{l} \overleftarrow{\beta} \in \left[1, \frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1 \right] \\ \overrightarrow{\beta} \in \left[1, \frac{1}{2}\alpha(\alpha+3) \right], \beta \in [1, \alpha] \\ t_1, t_2 \in T[S]^X \end{array} \right\}.$$

it is quite easy to see that $(s[u \leftarrow \ell_1], h) \not\models \varphi$ and $(s'[u \leftarrow \ell_2], h') \not\models \varphi$. Indeed, all the formulae in this set require ℓ_1 (resp. ℓ_2) to not belong to $\text{Pred}[S]_{s,h}^X(x)$ (resp. $\text{Pred}[S]_{s',h'}^X(x)$). Let us now consider a variable $y \in X$. We show that

$$\begin{aligned} (s[u \leftarrow \ell_1], h) \models u \in \text{pred}_X^S(y) &\text{ if and only if } (s'[u \leftarrow \ell_2], h') \models u \in \text{pred}_X^S(y). \\ (s[u \leftarrow \ell_1], h) \models u \in \text{pred}_X^S(y), & \\ \Leftrightarrow \ell_1 \in \text{Pred}[S]_{s,h}^X(y), & \quad (\text{by definition of } \models) \\ \Leftrightarrow \ell_1 \in \text{Pred}[S]_{s,h}^X(x) \text{ and } \llbracket x \rrbracket_{s,h}^X = \llbracket y \rrbracket_{s,h}^X, & \\ & \quad (\text{from } \ell_1 \in \text{Pred}[S]_{s,h}^X(x) \text{ and the definition of } \text{Pred}[S]_{s,h}^X(x)) \\ \Leftrightarrow \ell_2 \in \text{Pred}[S]_{s',h'}^X(x) \text{ and } \llbracket x \rrbracket_{s',h'}^X = \llbracket y \rrbracket_{s',h'}^X, & \\ & \quad (\text{by def. of } \ell_2 \text{ and as } (s[u \leftarrow \ell_1], h) \text{ and } (s'[u \leftarrow \ell_2], h') \text{ equisatisfy the formula } x = y) \\ \Leftrightarrow \ell_2 \in \text{Pred}[S]_{s',h'}^X(y), & \quad (\text{by definition of } \text{Pred}[S]_{s',h'}^X(y)) \\ \Leftrightarrow (s'[u \leftarrow \ell_2], h') \models u \in \text{pred}_X^S(y). & \quad (\text{by definition of } \models) \end{aligned}$$

case: there is $\beta \in [1, \alpha]$ and $L \in \text{Cycl}[S]_{s,h}^X(\beta)$ such that $\ell_1 \in L$. From the equisatisfaction of the formulae in $\text{Sk}[S](X, \alpha)$, the set $\text{Cycl}[S]_{s',h'}^X(\beta)$ is not empty. Consider ℓ_2 to be a location in a set of $\text{Cycl}[S]_{s',h'}^X(\beta)$. Hence $\ell_2 \in \text{dom}(h')$, which implies that $\ell_2 \leq \text{maxval}_X(s', h') + 1$. Given a formula φ in

$$\left\{ \begin{array}{l} u = t_1, u \in \text{loop}_{X}^S(\beta'), u \in \uparrow \text{loop}_{X,\alpha}^S, \\ u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta}), \\ u \in \text{pred}_X^S(x), u \in \text{rem}_{X,\alpha}^S \end{array} \right| \begin{array}{l} \overleftarrow{\beta} \in \left[1, \frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1 \right] \\ \overrightarrow{\beta} \in \left[1, \frac{1}{2}\alpha(\alpha+3) \right], \beta' \in [1, \alpha] \setminus \{\beta\} \\ x \in X, t_1, t_2 \in T[S]^X \end{array} \right\}.$$

it is quite easy to see that $(s[u \leftarrow \ell_1], h) \not\models \varphi$ and $(s'[u \leftarrow \ell_2], h') \not\models \varphi$. The set above contains every core formula from $\text{Obs}[S](X, \alpha)$, with the exception of $u \in \text{loop}_{X}^S(\beta)$, which is satisfied by both $(s[u \leftarrow \ell_1], h)$ and $(s'[u \leftarrow \ell_2], h')$ directly by definition of ℓ_1 and ℓ_2 .

case: there is a set $L \in \uparrow \text{Cycl}[S]_{s,h}^{X,\alpha}$ such that $\ell_1 \in L$. From the equisatisfaction of the formulae in $\text{Sk}[S](X, \alpha)$, the set $\uparrow \text{Cycl}[S]_{s',h'}^{X,\alpha}$ is not empty. Consider ℓ_2 to be a location in a set of $\uparrow \text{Cycl}[S]_{s',h'}^{X,\alpha}$. Hence $\ell_2 \in \text{dom}(h')$, which implies that $\ell_2 \leq \text{maxval}_X(s', h') + 1$. The proof continues as in the previous case. Among the formulae in $\text{Obs}[S](X, \alpha)$, the memory states $(s[u \leftarrow \ell_1], h)$ and $(s'[u \leftarrow \ell_2], h')$ satisfy only the formula $u \in \uparrow \text{loop}_{X,\alpha}^S$.

case: $\ell_1 \in \text{Rem}[S]_{s,h}^{X,\alpha}$. From the equisatisfaction of the formulae in $\text{Sk}[S](X, \alpha)$, the set $\text{Rem}[S]_{s',h'}^{X,\alpha}$ is not empty. Consider ℓ_2 to be a location in this set. Hence $\ell_2 \in \text{dom}(h')$, which implies that $\ell_2 \leq \text{maxval}_X(s', h') + 1$. The proof continues as in the previous two cases. Among the formulae in $\text{Obs}[S](X, \alpha)$, the memory states $(s[u \leftarrow \ell_1], h)$ and $(s'[u \leftarrow \ell_2], h')$ satisfy only the formula $u \in \text{rem}_{X,\alpha}^S$.

case: $\ell_1 \notin \text{dom}(h) \cup \text{Lab}[S]_{s,h}^X$. Let $\ell_2 = \text{maxval}_X(s', h') + 1$. By definition of $\text{maxval}_X(s', h')$, we have $\ell_2 \notin \text{dom}(h')$ and $\ell_2 \notin \text{Lab}[S]_{s',h'}^X \subseteq \text{ran}(h') \cup s'(X)$. Thus, ℓ_2 is an unlabelled location that does not belong to any of the sets in Definition 5.29. By (Inv-u) and from the definition of the core formulae, for every φ in $\text{Obs}[S](X, \alpha)$ we have $(s[u \leftarrow \ell_1], h) \not\models \varphi$ and $(s'[u \leftarrow \ell_2], h') \not\models \varphi$. \square

5.6 CONNECTING THE TWO FAMILIES OF CORE FORMULAE

In the previous two sections, we introduced the core formulae $\text{Core}[\mathcal{W}](X, \alpha)$ and $\text{Core}[\mathcal{S}](X, \alpha)$ for the weak and strong fragments of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow\!\!\!\rightarrow]^{\mathcal{S}}_{\mathcal{W}})$, respectively. We have shown that both families of core formulae enjoy the $*$ -simulation (Lemmata 5.20 and 5.41) and \exists -simulation properties (Lemmata 5.23 and 5.42). Moreover, we have proved (Lemmata 5.15 and 5.37) that every atomic formula of the weak fragment (resp. strong fragment) can be expressed as a Boolean combinations of formulae from $\text{Core}[\mathcal{W}](X, 1)$ (resp. $\text{Core}[\mathcal{S}](X, 1)$). Lastly, we have shown that the indistinguishability relation $\approx_{X, \alpha}^{\mathcal{S}}$, defined from $\text{Core}[\mathcal{S}](X, \alpha)$, is a refinement of the indistinguishability relation $\approx_{X, \alpha}^{\mathcal{W}}$, defined from $\text{Core}[\mathcal{W}](X, \alpha)$ (Corollary 5.36). Thanks to these results, in this section we finally show that $\text{SL}([\exists]_1, *, [-*, \hookrightarrow\!\!\!\rightarrow]^{\mathcal{S}}_{\mathcal{W}})$ enjoy a polynomial small-heap property. Again, we assume $X \subseteq_{\text{fin}} \text{VAR} \setminus \{u\}$.

In order to obtain the small-heap property, the key step that we are missing consists in showing that the two families of core formulae effectively mimic the magic wand operator $s \multimap w$. As done for the operators $*$ and \exists , we call this property \multimap -simulation. Its statement is derived directly from the semantics of the separating implication $s \multimap w$: given two memory states $(s, h) \approx_{X, \alpha + \text{card}(X)}^{\mathcal{W}} (s', h')$ the \multimap -simulation asks that for every heap h_1 disjoint from h there is a heap h'_1 disjoint from h' such that $(s, h_1) \approx_{X, \alpha}^{\mathcal{S}} (s', h'_1)$ and $(s, h + h_1) \approx_{X, \alpha}^{\mathcal{W}} (s', h' + h'_1)$. Notice that the memory states (s, h_1) and (s', h'_1) must be indistinguishable with respect to the relation $\approx_{X, \alpha}^{\mathcal{S}}$ introduced for the strong fragment, which simulates the left-hand side of $s \multimap w$. Similarly, in order to simulate the right-hand side of $s \multimap w$, the memory states $(s, h + h_1)$ and $(s', h' + h'_1)$ must be indistinguishable with respect to the relation $\approx_{X, \alpha}^{\mathcal{W}}$.

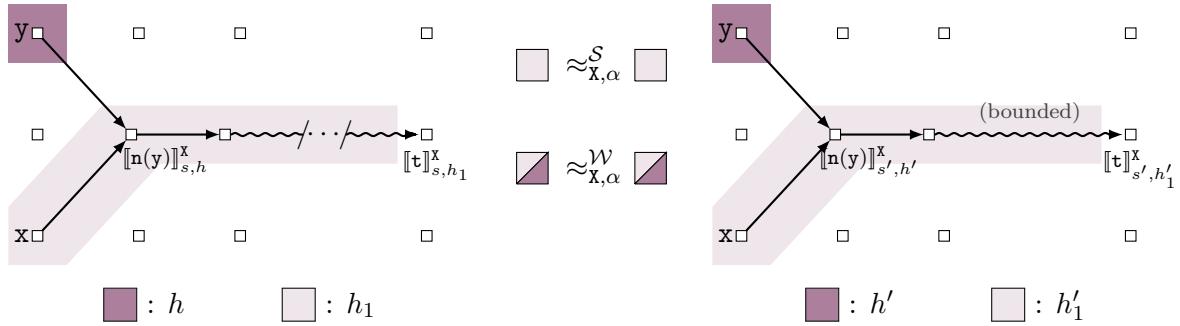
As we argued when considering the \exists -simulation property (Section 5.3.4), the \multimap -simulation we have just described must be strengthened in order to obtain a polynomial small-heap property. Any polynomial function will do the job and, in our case, we show that the maximum value of the memory state (s', h'_1) can be bounded by $\mathfrak{P}(\text{card}(X), \alpha)$, where $\mathfrak{P}(x, a) = (x + 1)(a + 3)^4$. With this constraint taken into account, the \multimap -simulation property is formalised as follows.

Lemma 5.43 (\multimap -simulation). Suppose $(s, h) \approx_{X, \alpha + \text{card}(X)}^{\mathcal{W}} (s', h')$. For every heap h_1 disjoint from h there is a heap h'_1 disjoint from h' such that

- (I) $(s, h_1) \approx_{X, \alpha}^{\mathcal{S}} (s', h'_1)$ and $(s, h + h_1) \approx_{X, \alpha}^{\mathcal{W}} (s', h' + h'_1)$,
- (II) $\text{maxval}_{X \cup \{u\}}(s', h'_1) \leq \text{maxval}_{X \cup \{u\}}(s', h') + \mathfrak{P}(\text{card}(X), \alpha)$.

The proof of this result, which is quite long and thus presented at the end of the section, is achieved by defining a “small” heap h'_1 so that the condition (II) is guaranteed by construction. In doing so, we need to carefully handle the labelled locations so that (I) holds. For instance, let us suppose the heaps h , h' and h_1 of Lemma 5.43 to be the ones represented in Figure 5.18. In particular, h_1 witnesses a path from $s(x)$ to the location corresponding to the term t , and in this path the location $h_1(s(x))$ corresponds to the term $n(y)$ with respect to the memory state (s, h) . So, $(s, h + h_1) \models n(x) = n(y)$. We construct the heap h'_1 so that it witnesses a path from $s'(x)$ to a location corresponding to the term t , and moreover it is such that $h'_1(s'(x))$ is the location $\llbracket n(y) \rrbracket_{s, h'}^X$. The length of this path can be bounded following the upper bound $\frac{1}{6}(\alpha + 1)(\alpha + 2)(\alpha + 3)$ given to β in the core formulae of the form $\text{sees}_X(t_1, t_2) \geq \beta$, leading to the satisfaction of the second point of Lemma 5.43. The hypothesis $(s, h) \approx_{X, \alpha + \text{card}(X)}^{\mathcal{W}} (s', h')$ guarantees that the construction can always be done correctly.

Once Lemma 5.43 is established, we can finally relate the equisatisfaction of a formula φ from $\text{SL}([\exists]_1, *, [-*, \hookrightarrow\!\!\!\rightarrow]^{\mathcal{S}}_{\mathcal{W}})$ to the indistinguishability relations. Essentially, we show that φ is

Figure 5.18: Construction for the $\rightarrow*$ -simulation. The heap h'_1 is an answer for h_1 .

equisatisfied by every two memory states satisfying the same core formulae from $\text{Core}[S](\mathbf{X}, \alpha)$, where the set \mathbf{X} contains the program variables appearing in φ (excluding u) and the positive integer α is at least the memory size of φ , defined below.

Definition 5.44 (Memory size). The *memory size* of a formula φ in $\text{SL}([\exists]_1, *, [\rightarrow*, \hookrightarrow^+]_{\mathcal{W}}^S)$, denoted by $|\varphi|_m$, is inductively defined as follows:

$$\begin{aligned}
 |\pi|_m &\stackrel{\text{def}}{=} 1, \text{ where } \pi \text{ is an atomic formula,} & |\exists u \varphi|_m &\stackrel{\text{def}}{=} |\varphi|_m, \\
 |\varphi_1 \wedge \varphi_2|_m &\stackrel{\text{def}}{=} \max(|\varphi_1|_m, |\varphi_2|_m), & |\neg \varphi|_m &\stackrel{\text{def}}{=} |\varphi|_m, \\
 |\varphi_1 \rightarrow \varphi_2|_m &\stackrel{\text{def}}{=} \text{card}(\text{fv}(\varphi_1 \rightarrow \varphi_2) \setminus \{u\}) + \max(|\varphi_1|_m, |\varphi_2|_m), & |\varphi_1 * \varphi_2|_m &\stackrel{\text{def}}{=} |\varphi_1|_m + |\varphi_2|_m.
 \end{aligned}$$

Notice that the memory size of a formula is always polynomial in the (tree) size of the formula and the number of its variables. In particular, one can show that $|\varphi|_m \leq |\varphi|^2$.

Lemma 5.45. Let φ be a formula in $\text{SL}([\exists]_1, *, [\rightarrow*, \hookrightarrow^+]_{\mathcal{W}}^S)$ s.t. $\text{fv}(\varphi) \setminus \{u\} \subseteq \mathbf{X}$ and $|\varphi|_m \leq \alpha$. Given two memory states (s, h) and (s', h') , if one of the following holds

(I) $(s, h) \approx_{\mathbf{X}, \alpha}^S (s', h')$, or (II) $(s, h) \approx_{\mathbf{X}, \alpha}^W (s', h')$ and φ is a formula from the weak fragment, then $(s, h) \models \varphi$ if and only if $(s', h') \models \varphi$.

Proof. Since $\approx_{\mathbf{X}, \alpha}^S \subseteq \approx_{\mathbf{X}, \alpha}^W$ (Corollary 5.36), if φ is a formula of the weak fragment then the hypothesis (I) implies the hypothesis (II). Thus, we rely on (II) when φ is from the weak fragment, and use (I) only otherwise. The proof is by structural induction on φ (with the natural induction hypothesis stating that the property holds for strict subformulae).

base case: φ is \top , $t_1 = t_2$, $t_1 \hookrightarrow t_2$ or emp (where $t_1, t_2 \in \mathbf{X} \cup \{u\}$). In this case, φ is a formula of the weak fragment, and the result follows directly from Lemma 5.15.

base case: $\varphi = t_1 \hookrightarrow^+ t_2$ (where $t_1, t_2 \in \mathbf{X} \cup \{u\}$). We remind the reader that, since φ is a formula in $\text{SL}([\exists]_1, *, [\rightarrow*, \hookrightarrow^+]_{\mathcal{W}}^S)$, if $t_1 = u$ then $t_2 = u$. In this case, φ is a formula of the strong fragment. The result follows directly from Lemma 5.37.

We omit the trivial cases for $\varphi = \neg \psi$ and $\varphi = \psi_1 \wedge \psi_2$. In the induction steps for $\varphi = \psi_1 * \psi_2$, $\varphi = \exists u \psi$ and $\varphi = \psi_1 \rightarrow \psi_2$, it is sufficient to show one direction of the double implication, since both $\approx_{\mathbf{X}, \alpha}^W$ and $\approx_{\mathbf{X}, \alpha}^S$ are symmetric relations.

induction step: $\varphi = \psi_1 * \psi_2$. We divide the proof depending on whether φ is in \mathcal{W} .

case: φ is in the weak fragment. We have $(s, h) \approx_{X,\alpha}^W (s', h')$ and $(s, h) \models \psi_1 * \psi_2$.

There are two disjoint heaps h_1 and h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \psi_1$ and $(s, h_2) \models \psi_2$. As $\alpha \geq |\psi_1 * \psi_2|_m = |\psi_1|_m + |\psi_2|_m$, there are $\alpha_1, \alpha_2 \geq 1$ such that $\alpha = \alpha_1 + \alpha_2$, $\alpha_1 \geq |\psi_1|_m$ and $\alpha_2 \geq |\psi_2|_m$. By Lemma 5.20 there are disjoint heaps h'_1 and h'_2 such that $h' = h'_1 + h'_2$, $(s, h_1) \approx_{X,\alpha_1}^W (s', h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^W (s', h'_2)$. As φ is in the weak fragment, the same holds for both ψ_1 and ψ_2 . Therefore, we can apply the induction hypothesis with (II), and conclude that $(s', h'_1) \models \psi_1$ and $(s', h'_2) \models \psi_2$. Thus, $(s', h') \models \psi_1 * \psi_2$.

case: φ is not in the weak fragment. We have $(s, h) \approx_{X,\alpha}^S (s', h')$ and $(s, h) \models \psi_1 * \psi_2$.

There are two disjoint heaps h_1 and h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \psi_1$ and $(s, h_2) \models \psi_2$. As $\alpha \geq |\psi_1 * \psi_2|_m = |\psi_1|_m + |\psi_2|_m$, there are $\alpha_1, \alpha_2 \geq 1$ such that $\alpha = \alpha_1 + \alpha_2$, $\alpha_1 \geq |\psi_1|_m$ and $\alpha_2 \geq |\psi_2|_m$. By Lemma 5.41 there are disjoint heaps h'_1 and h'_2 such that $h' = h'_1 + h'_2$, $(s, h_1) \approx_{X,\alpha_1}^S (s', h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^S (s', h'_2)$. By induction hypothesis with (I), $(s', h'_1) \models \psi_1$ and $(s', h'_2) \models \psi_2$. Thus, $(s', h') \models \psi_1 * \psi_2$.

induction step: $\varphi = \exists u \psi$. We divide the proof depending on whether φ is in W .

case: φ is in the weak fragment. We have $(s, h) \approx_{X,\alpha}^W (s', h')$ and $(s, h) \models \exists u \psi$. There is a location ℓ_1 such that $(s[u \leftarrow \ell_1], h) \models \psi$. By Lemma 5.23, there is a location ℓ_2 such that $(s[u \leftarrow \ell_1], h) \approx_{X,\alpha}^W (s'[u \leftarrow \ell_2], h')$. Notice that $|\psi|_m = |\varphi|_m \leq \alpha$. Moreover, as φ is a formula in the weak fragment, the same holds for ψ . Hence, we can apply the induction hypothesis with (II), and obtain $(s'[u \leftarrow \ell_2], h') \models \psi$. Thus, $(s', h') \models \exists u \psi$.

case: φ is not in the weak fragment. We have $(s, h) \approx_{X,\alpha}^S (s', h')$ and $(s, h) \models \exists u \psi$.

There is a location ℓ_1 such that $(s[u \leftarrow \ell_1], h) \models \psi$. By Lemma 5.42, there is a location ℓ_2 such that $(s[u \leftarrow \ell_1], h) \approx_{X,\alpha}^S (s'[u \leftarrow \ell_2], h')$. As $|\psi|_m = |\varphi|_m \leq \alpha$, by induction hypothesis with (I), $(s'[u \leftarrow \ell_2], h') \models \psi$. Thus, $(s', h') \models \exists u \psi$.

induction step: $\varphi = \psi_1 \dashv \psi_2$. We remind the reader that ψ_1 is from the strong fragment (which includes the weak fragment), whereas ψ_2 is from the weak fragment. As $\psi_1 \dashv \psi_2$ is from the weak fragment, it is sufficient to prove the result under the hypothesis (II). So, suppose $(s, h) \approx_{X,\alpha}^W (s', h')$ and $(s, h) \models \psi_1 \dashv \psi_2$. Let us define $Y \stackrel{\text{def}}{=} \text{fv}(\psi_1 \dashv \psi_2) \setminus \{u\}$ and $\alpha' \stackrel{\text{def}}{=} \max(|\psi_1|_m, |\psi_2|_m)$. By definition of memory size, $\text{card}(Y) + \alpha' = |\psi_1 \dashv \psi_2|_m \leq \alpha$. Notice that for every $\alpha_1 \leq \alpha_2$, $\text{Core}[W](X, \alpha_1) \subseteq \text{Core}[W](X, \alpha_2)$, which in turn means that $\approx_{X,\alpha_2}^W \subseteq \approx_{X,\alpha_1}^W$. Thus, $(s, h) \approx_{X,\text{card}(Y)+\alpha'}^W (s', h')$ holds. Let us prove that $(s', h') \models \psi_1 \dashv \psi_2$. Following the definition of $\psi_1 \dashv \psi_2$, we consider a heap h'_1 disjoint from h' and such that $(s', h'_1) \models \psi_1$. We prove that $(s', h' + h'_1) \models \psi_2$. By Lemma 5.43, there is a heap h_1 disjoint from h and such that $(s, h_1) \approx_{X,\alpha'}^S (s', h'_1)$ and $(s, h + h_1) \approx_{X,\alpha'}^W (s', h' + h'_1)$. Since $|\psi_1|_m \leq \alpha'$ and $(s', h'_1) \models \psi_1$, by induction hypothesis we obtain $(s, h_1) \models \psi_1$. Therefore, by $(s, h) \models \psi_1 \dashv \psi_2$ it holds that $(s, h + h_1) \models \psi_2$. Since $|\psi_2|_m \leq \alpha'$, again by induction hypothesis we obtain $(s', h' + h'_1) \models \psi_2$, concluding the proof. \square

Lemma 5.45 implies that the logic obtained by closing the core formulae of the strong fragment under Boolean connectives capture the expressive power of $\text{SL}([\exists]_1, *, [\dashv, \rightarrow^+]^S_W)$.

Theorem 5.46. Every formula φ in $\text{SL}([\exists]_1, *, [\dashv, \rightarrow^+]^S_W)$ is logically equivalent to a Boolean combination of core formulae from $\text{Core}[S](\text{fv}(\varphi) \setminus \{u\}, |\varphi|_m)$.

Proof. Following Lemma 5.45, the proof is quite standard. For a memory state (s, h) , $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq 1$, we write $\text{LIT}_{X,\alpha}(s, h)$ to denote the following set of literals:

$$\text{LIT}_{\mathbf{X}, \alpha}(s, h) \stackrel{\text{def}}{=} \{\psi \in \text{Core}[S](\mathbf{X}, \alpha) \mid (s, h) \models \psi\} \cup \{\neg\psi \mid (s, h) \not\models \psi \text{ and } \psi \in \text{Core}[S](\mathbf{X}, \alpha)\}$$

Informally $\text{LIT}_{\mathbf{X}, \alpha}(s, h)$ contains the literals obtained from core formulae in $\text{Core}[S](\mathbf{X}, \alpha)$, that are satisfied by (s, h) . Notice that $\text{card}(\text{Core}[S](\mathbf{X}, \alpha)) = \text{card}(\text{LIT}_{\mathbf{X}, \alpha}(s, h))$, as every core formula in $\text{Core}[S](\mathbf{X}, \alpha)$ appears (possibly negated) in $\text{LIT}_{\mathbf{X}, \alpha}(s, h)$. Therefore, $\text{LIT}_{\mathbf{X}, \alpha}(s, h)$ is finite. When considering the formula $\bigwedge_{\psi \in \text{LIT}_{\mathbf{X}, \alpha}(s, h)} \psi$, by definition of $\approx_{\mathbf{X}, \alpha}^S$ we have the following equivalence:

$$(s', h') \models \bigwedge_{\psi \in \text{LIT}_{\mathbf{X}, \alpha}(s, h)} \psi \quad \text{if and only if} \quad (s, h) \approx_{\mathbf{X}, \alpha}^S (s', h').$$

Consider now the formula φ in the statement of the lemma. Let $\mathbf{X} \stackrel{\text{def}}{=} \text{fv}(\varphi) \setminus \{\mathbf{u}\}$ and $\alpha \stackrel{\text{def}}{=} |\varphi|_{\mathbf{m}}$. The infinite expression $\psi_{\text{inf}} \stackrel{\text{def}}{=} \bigvee_{(s, h) \models \varphi} (\bigwedge_{\psi \in \text{LIT}_{\mathbf{X}, \alpha}(s, h)} \psi)$ is equivalent to a (finite) Boolean combination ψ_{fin} of formulae from $\text{Core}[S](\mathbf{X}, \alpha)$. Indeed, since $\text{LIT}_{\mathbf{X}, \alpha}(s, h)$ is finite, there are only finitely many conjunctions of the form $\bigwedge_{\psi \in \text{LIT}_{\mathbf{X}, \alpha}(s, h)} \psi$, which implies that ψ_{inf} is equivalent to a finite disjunction of conjunctions of core formulae literals. In order to conclude the proof, we show that φ is logically equivalent to ψ_{inf} . Suppose that $(s, h) \models \varphi$. Obviously, we have $(s, h) \models \bigwedge_{\psi \in \text{LIT}_{\mathbf{X}, \alpha}(s, h)} \psi$ and therefore $(s, h) \models \psi_{\text{inf}}$. Conversely, suppose that $(s, h) \models \psi_{\text{inf}}$. This means that there is a memory state (s', h') such that $(s', h') \models \varphi$ and $(s, h) \models \bigwedge_{\psi \in \text{LIT}_{\mathbf{X}, \alpha}(s', h')} \psi$. Since $(s, h) \approx_{\mathbf{X}, \alpha}^S (s', h')$ and $(s', h') \models \varphi$, by Lemma 5.45 we conclude that $(s, h) \models \varphi$. \square

5.6.1 Small-heap property and PSpace-completeness.

Following Theorem 5.46, we establish that $\text{SL}([\exists]_1, *, [-*, \hookrightarrow]^S_W)$ enjoy the polynomial small-heap property and provide a PSPACE algorithm for its satisfiability problem. In order to show the small-heap property, we first prove that it holds for Boolean combinations of core formulae and then transfer this result to $\text{SL}([\exists]_1, *, [-*, \hookrightarrow]^S_W)$, thanks to Theorem 5.46.

Lemma 5.47. Every satisfiable Boolean combination of formulae from $\text{Core}[S](\mathbf{X}, \alpha)$ is satisfied by a memory state (s, h) such that $\text{card}(h)$ is bounded by a polynomial in $\mathcal{O}(\text{card}(\mathbf{X})\alpha^4)$.

To prove this lemma we rely on Lemma 5.43. A stand-alone proof that uses directly the definitions of the core formulae is also possible, and leads to a better asymptotical bound on the cardinality of the heap. As we are only interested in finding a *polynomial* bound for $\text{card}(h)$, we prefer the less technically involved proof presented here.

Proof. Suppose φ to be a satisfiable Boolean combination of formulae from $\text{Core}[S](\mathbf{X}, \alpha)$, and let (s, h) be a memory state such that $(s, h) \models \varphi$. We can assume every variable in $\mathbf{X} \cup \{\mathbf{u}\}$ to be mapped to a location that is at most $\text{card}(\mathbf{X} \cup \{\mathbf{u}\})$, i.e. $\text{maxval}_{\mathbf{X} \cup \{\mathbf{u}\}}(s, \emptyset) \leq \text{card}(\mathbf{X} \cup \{\mathbf{u}\})$. This property is without loss of generality, as locations can be reordered to satisfy it. Let us consider the memory state (s, \emptyset) . As $\approx_{\mathbf{X}, \alpha + \text{card}(\mathbf{X})}^W$ is reflexive, $(s, \emptyset) \approx_{\mathbf{X}, \alpha + \text{card}(\mathbf{X})}^W (s, \emptyset)$. By Lemma 5.43 there is a heap h' such that

$$1. (s, h) \approx_{\mathbf{X}, \alpha}^S (s, h'), \quad 2. \text{maxval}_{\mathbf{X} \cup \{\mathbf{u}\}}(s, h') \leq \text{maxval}_{\mathbf{X} \cup \{\mathbf{u}\}}(s, \emptyset) + \mathfrak{P}(\text{card}(\mathbf{X}), \alpha).$$

From (1), (s, h) and (s, h') satisfy the same formulae in $\text{Core}[S](\mathbf{X}, \alpha)$. Thus, $(s, h') \models \varphi$. From (2) and $\text{maxval}_{\mathbf{X} \cup \{\mathbf{u}\}}(s, \emptyset) \leq \text{card}(\mathbf{X} \cup \{\mathbf{u}\})$, $\text{card}(h')$ is bounded in $\mathcal{O}(\text{card}(\mathbf{X})\alpha^4)$. \square

Corollary 5.48 (Small-heap property). Every satisfiable φ in $\text{SL}([\exists]_1, *, [-*, \hookrightarrow]^S_W)$ is satisfied by a memory state (s, h) s.t. $\text{card}(h) \leq \mathfrak{Q}(|\varphi|)$, where $\mathfrak{Q} : \mathbb{N} \rightarrow \mathbb{N}$ is a polynomial of degree 9.

Proof. Directly from Theorem 5.46 and Lemma 5.47, if φ is satisfiable then it is satisfied by a memory state (s, h) such that $\text{card}(h)$ is bounded by a polynomial in $\mathcal{O}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\})|\varphi|_{\mathbf{m}}^4)$. As $|\varphi|_{\mathbf{m}} \leq |\varphi|^2$ and $\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}) \leq |\varphi|$, this polynomial is in $\mathcal{O}(|\varphi|^9)$. \square

We are now ready to define an algorithm for the satisfiability problem of $\text{SL}([\exists]_1, *, [\neg*, \hookrightarrow^{\mathcal{S}}]_{\mathcal{W}})$ that runs in NPSPACE (i.e. non-deterministic PSPACE). By Savitch Theorem, this shows the existence of a deterministic algorithm running in PSPACE [127]. Given a formula φ , the pseudo-code of the non-deterministic algorithm $\text{sat}(\varphi)$ is described in Figure 5.19. To correctly analyse the algorithm, we recall the semantics of the classical non-deterministic instructions used therein:

- **choose x satisfying $P(x)$** : the program (non-deterministically) branches on every value for x enjoying the predicate $P(x)$. If at least one of the branch terminates with the instruction **succeed**, then this instruction is semantically equivalent to **succeed**. If all the branches terminate, and they do so with the instruction **fail**, then this instruction is semantically equivalent to **fail**.
- **succeed** : the program terminates successfully.
- **fail** : the program terminates unsuccessfully.

Informally, in the algorithm is quite standard: it guesses a memory state (s, h) and checks if it satisfies φ by calling the model-checking algorithm $\text{mc}(s, h, \varphi)$. Fundamentally, the memory state (s, h) can be effectively represented in polynomial space. According to Corollary 5.48, the maximum value of (s, h) is bounded by the polynomial \mathfrak{Q} and the cardinality of $\text{fv}(\varphi) \cup \{u\}$ (line 3), which bound the locations in the heap and in the store, respectively. Indeed, from the notion of X-heap isomorphism memory states and Proposition 2.10 we know that the domain of the store can be restricted to $\text{fv}(\varphi) \cup \{u\}$, leading to a finite representation of the store as a structure of size linear in $|\varphi|$. Because of this, in order to prove that $\text{sat}(\cdot)$ is a decision procedure for satisfiability that runs in NPSPACE it is sufficient to show that $\text{mc}(\cdot)$ is a model-checking procedure also running in NPSPACE. Essentially, $\text{mc}(s, h, \varphi)$ checks whether $(s, h) \models \varphi$ holds with a linear-depth recursive algorithm that internalises the semantics of φ (see e.g. [33]). The various simulation properties of the core formulae ensure that only a polynomial amount of locations ever needs to be considered during execution, leading to NPSPACE.

Lemma 5.49. Let (s, h) be a memory state and let φ be a formula in $\text{SL}([\exists]_1, *, [\neg*, \hookrightarrow^{\mathcal{S}}]_{\mathcal{W}})$.

(I) $\text{mc}(s, h, \varphi)$ always terminates with either **succeed** or **fail**, and runs in space

$$\mathcal{O}\left(\maxval_{\text{fv}(\varphi) \cup \{u\}}(s, h) + |\varphi| \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_{\mathfrak{m}})\right),$$

(II) $\text{mc}(s, h, \varphi) = \text{succeed}$ if and only if $(s, h) \models \varphi$.

Both (I) and (II) follow by structural induction on φ . The key ingredients that allow us to show (I) are given by the space constraints given during the formalisation of the \exists -simulation and $\neg*$ -simulation properties, together with the linear recursion depth, in the size of φ , of the algorithm. In the proof below we refer to the line numbers of Figure 5.19.

Proof of (I). We remind the reader that we assume $\text{LOC} = \mathbb{N}$ (Assumption 5.21). We analyse the space complexity of the algorithm $\text{mc}(\cdot)$ in terms of number of locations needed in order to represent the memory states taken into account throughout its execution. As the algorithm only allocates (in memory) these locations, polynomially bound their number entails that the algorithm runs in NPSPACE. We write $\text{SPACE}[\text{mc}(s, h, \varphi)]$ for the maximal location considered by the algorithm when executing $\text{mc}(s, h, \varphi)$. With this in mind, the proof by structural induction on φ uses the following induction hypothesis:

for every (s, h) and φ , $\text{mc}(s, h, \varphi)$ terminates with **succeed** or **fail**,
and $\text{SPACE}[\text{mc}(s, h, \varphi)] \leq \maxval_{\text{fv}(\varphi) \cup \{u\}}(s, h) + |\varphi| \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_{\mathfrak{m}})$.

Assuming a reasonably succinct encoding of memory states, this induction hypothesis entails (I).

```

1: sat( $\varphi$ ):
2:    $Y \leftarrow \text{fv}(\varphi) \cup \{u\}$ 
3:   choose  $(s, h)$  satisfying  $\maxval_Y(s, h) \leq 2\Omega(|\varphi|) + \text{card}(Y)$            ▷ Corollary 5.48
4:   mc( $s, h, \varphi$ )
5: mc( $s, h, \varphi$ ):
6:   switch  $\varphi$  do
7:     • case  $\pi$  (atomic formula):
8:       atomic_mc( $s, h, \pi$ )
9:     • case  $\neg\psi$ :
10:      if  $\text{mc}(s, h, \psi) = \text{succeed}$  then fail else succeed
11:     • case  $\psi_1 \wedge \psi_2$ :
12:       if  $\text{mc}(s, h, \psi_1) = \text{succeed}$  and  $\text{mc}(s, h, \psi_2) = \text{succeed}$  then succeed else fail
13:     • case  $\psi_1 * \psi_2$ :
14:       choose  $(h_1, h_2)$  satisfying  $h_1 \perp h_2$  and  $h_1 + h_2 = h$ 
15:       if  $\text{mc}(s, h_1, \psi_1) = \text{succeed}$  and  $\text{mc}(s, h_2, \psi_2) = \text{succeed}$  then succeed else fail
16:     • case  $\exists u \psi$ :
17:       choose  $\ell$  satisfying  $\ell' \leq \maxval_{\text{fv}(\varphi)}(s, h) + 1$            ▷ Lemmata 5.23 and 5.42
18:        $\text{mc}(s[u \leftarrow \ell], h, \psi)$ 
19:     • case  $\psi_1 \multimap \psi_2$ :
20:       choose  $h'$  satisfying  $h' \perp h$  and                               ▷ Lemma 5.43
21:          $\maxval_{\text{fv}(\varphi) \cup \{u\}}(s, h') \leq \maxval_{\text{fv}(\varphi) \cup \{u\}}(s, h) + \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_{\mathfrak{m}})$ 
22:       if  $\text{mc}(s, h', \psi_1) = \text{succeed}$  and  $\text{mc}(s, h + h', \psi_2) = \text{succeed}$  then succeed else fail
23: atomic_mc( $s, h, \pi$ )                                         ▷ runs in PTIME
24:   switch  $\pi$  do
25:     • case  $\top$ :
26:       succeed
27:     • case  $\text{emp}$ :
28:       if  $h = \emptyset$  then succeed else fail
29:     • case  $t_1 = t_2$ :
30:       if  $s(t_1) = s(t_2)$  then succeed else fail
31:     • case  $t_1 \hookrightarrow t_2$ :
32:       if  $(s(t_1), s(t_2)) \in h$  then succeed else fail
33:     • case  $t_1 \hookrightarrow^+ t_2$ :
34:        $\ell \leftarrow s(x)$ 
35:       for  $\delta \leftarrow 0, \delta \leq \text{card}(h), \delta++$  do
36:         if  $\ell \notin \text{dom}(h)$  then fail
37:          $\ell \leftarrow h(\ell)$ 
38:         if  $\ell = s(t_2)$  then succeed
39:       end for
40:       fail

```

} checks whether there is $\delta \geq 1$ such that $h^\delta(s(t_1)) = s(t_2)$.

Figure 5.19: A NPSPACE algorithm for the satisfiability problem of $\text{SL}([\exists]_1, *, [\multimap, \hookrightarrow^+]_{\mathcal{W}}^S)$.

base case: φ atomic formula. In this case, $\text{mc}(s, h, \varphi)$ calls the procedure $\text{atomic_mc}(s, h, \varphi)$ (line 8). This procedure runs in PTIME, terminates with either `succeed` or `fail`, and allocates a constant amount of memory. In particular, both the cases in lines 30 and 32 simply access the store and the heap with respect to two variables t_1 and t_2 in $\text{fv}(\varphi) \cup \{u\}$. So, the algorithm lookup locations that are less or equal than $\text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h)$. Similarly, in lines 34–40 the algorithm considers a location ℓ and an integer δ . Both these data are at most $\text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h)$. Thus, $\text{SPACE}[\text{mc}(s, h, \varphi)] \leq \text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h)$.

We omit the trivial cases for $\varphi = \neg\psi$ and $\varphi = \psi_1 \wedge \psi_2$.

induction step: $\varphi = \psi_1 * \psi_2$. Let us pick two heaps h_1 and h_2 such that $h = h_1 + h_2$, as done by the choose instruction in line 14. Given $j \in \{1, 2\}$, by induction hypothesis $\text{mc}(s, h_j, \psi_j)$ terminates with either `succeed` or `fail`, and $\text{SPACE}[\text{mc}(s, h_j, \psi_j)]$ is at most $\text{maxval}_{\text{fv}(\psi_j) \cup \{u\}}(s, h_j) + |\psi_j| \mathfrak{P}(\text{card}(\text{fv}(\psi_j) \setminus \{u\}), |\psi_j|_m)$. From line 15, we conclude that $\text{mc}(s, h, \varphi)$ also terminates with `succeed` or `fail`. Since $h_1 \subseteq h$ and $h_2 \subseteq h$, the locations considered by the algorithm in this case (lines 14 and 15) follow the equivalence

$$\text{SPACE}[\text{mc}(s, h, \varphi)] = \max(\text{SPACE}[\text{mc}(s, h_1, \psi_1)], \text{SPACE}[\text{mc}(s, h_2, \psi_2)]),$$

which entails $\text{SPACE}[\text{mc}(s, h, \varphi)] \leq \text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h) + |\varphi| \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_m)$. Indeed, this latter inequality is satisfied as soon as we show that

$$\begin{aligned} \text{maxval}_{\text{fv}(\psi_j) \cup \{u\}}(s, h_j) &\leq \text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h), \\ |\psi_j| \mathfrak{P}(\text{card}(\text{fv}(\psi_j) \setminus \{u\}), |\psi_j|_m) &\leq |\varphi| \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_m). \end{aligned}$$

The first inequality holds as we recall that $h_j \subseteq h$ and $\text{fv}(\psi_j) \subseteq \text{fv}(\varphi)$. Moreover, $|\psi_j| \leq |\varphi|$, $|\psi_j|_m \leq |\varphi|_m$, and \mathfrak{P} is monotonically increasing, which verifies the second inequality.

induction step: $\varphi = \exists u \psi$. Let us pick a location $\ell \leq \text{maxval}_{\text{fv}(\varphi)}(s, h) + 1$, as done by the choose instruction in line 17. By induction hypothesis, $\text{mc}(s[u \leftarrow \ell], h, \psi)$ terminates with either `succeed` or `fail`, and $\text{SPACE}[\text{mc}(s[u \leftarrow \ell], h, \psi)] \leq \text{maxval}_{\text{fv}(\psi) \cup \{u\}}(s[u \leftarrow \ell], h) + |\psi| \mathfrak{P}(\text{card}(\text{fv}(\psi) \setminus \{u\}), |\psi|_m)$. From line 18, we conclude that $\text{mc}(s, h, \varphi)$ also terminates with either `succeed` or `fail`. The locations considered by the algorithm in this case (lines 17 and 18) follow the equivalence

$$\text{SPACE}[\text{mc}(s, h, \varphi)] = \max(\ell, \text{SPACE}[\text{mc}(s[u \leftarrow \ell], h, \psi)]).$$

From the upper bound on ℓ and $\text{fv}(\psi) \setminus \{u\} \subseteq \text{fv}(\varphi)$, we have:

$$\begin{aligned} \text{maxval}_{\text{fv}(\psi) \cup \{u\}}(s[u \leftarrow \ell], h) &= \max(\ell, \text{maxval}_{\text{fv}(\psi) \setminus \{u\}}(s, h)) \\ &\leq \text{maxval}_{\text{fv}(\varphi)}(s, h) + 1 \\ &\leq \text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h) + 1 \end{aligned}$$

So, $\text{SPACE}[\text{mc}(s, h, \varphi)] \leq \text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h) + 1 + |\psi| \mathfrak{P}(\text{card}(\text{fv}(\psi) \setminus \{u\}), |\psi|_m)$. Moreover, as $|\varphi| = |\psi| + 1$, $\text{fv}(\psi) \setminus \{u\} = \text{fv}(\varphi) \setminus \{u\}$, $|\varphi|_m = |\psi|_m$ and \mathfrak{P} is a monotonically increasing strictly positive function, $1 + |\psi| \mathfrak{P}(\text{card}(\text{fv}(\psi) \setminus \{u\}), |\psi|_m) \leq |\varphi| \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_m)$. Therefore, $\text{SPACE}[\text{mc}(s, h, \varphi)] \leq \text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h) + |\varphi| \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_m)$.

induction step: $\varphi = \psi_1 \circledast \psi_2$. As done by the choose instruction in lines 20–21, Let us pick a heap h' disjoint from h and such that

$$\text{maxval}_{\text{fv}(\varphi)}(s, h') \leq \text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h) + \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_m).$$

By induction hypothesis, both $\text{mc}(s, h', \psi_1)$ and $\text{mc}(s, h + h', \psi_2)$ terminate with either `succeed` or `fail`. Moreover,

1. $\text{SPACE}[\text{mc}(s, h', \psi_1)] \leq \text{maxval}_{\text{fv}(\psi_1) \cup \{u\}}(s, h') + |\psi_1| \mathfrak{P}(\text{card}(\text{fv}(\psi_1) \setminus \{u\}), |\psi_1|_m),$

$$2. \text{SPACE}[\text{mc}(s, h+h', \psi_2)] \leq \text{maxval}_{\text{fv}(\psi_2) \cup \{\mathbf{u}\}}(s, h+h') + |\psi_2| \mathfrak{P}(\text{card}(\text{fv}(\psi_2) \setminus \{\mathbf{u}\}), |\psi_2|_{\mathfrak{m}}).$$

From line 22, $\text{mc}(s, h, \varphi)$ terminates with `succeed` or `fail`. The locations considered by the algorithm in this case (lines 20–22) follow the equivalence

$$\text{SPACE}[\text{mc}(s, h, \varphi)] = \max(\text{maxval}_{\text{fv}(\varphi)}(s, h'), \text{SPACE}[\text{mc}(s, h', \psi_1)], \text{SPACE}[\text{mc}(s, h+h', \psi_2)]).$$

Following the definition of h' , the first argument of the max function above is at most $\text{maxval}_{\text{fv}(\varphi) \cup \{\mathbf{u}\}}(s, h) + \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}})$. Thus, in order to conclude the proof it is sufficient to show that the right-hand side of the inequalities in (1) and (2) is at most $\text{maxval}_{\text{fv}(\varphi) \cup \{\mathbf{u}\}}(s, h) + |\varphi| \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}})$. For the right-hand side of (1) we have:

$$\begin{aligned} & \text{maxval}_{\text{fv}(\psi_1) \cup \{\mathbf{u}\}}(s, h') + |\psi_1| \mathfrak{P}(\text{card}(\text{fv}(\psi_1) \setminus \{\mathbf{u}\}), |\psi_1|_{\mathfrak{m}}) \\ & \leq \underbrace{\text{maxval}_{\text{fv}(\varphi) \cup \{\mathbf{u}\}}(s, h) + \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}})}_{\text{by definition of } h'} + |\psi_1| \mathfrak{P}(\text{card}(\text{fv}(\psi_1) \setminus \{\mathbf{u}\}), |\psi_1|_{\mathfrak{m}}) \\ & \leq \text{maxval}_{\text{fv}(\varphi) \cup \{\mathbf{u}\}}(s, h) + \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}}) + |\psi_1| \underbrace{\mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}})}_{\text{by } |\psi_1|_{\mathfrak{m}} \leq |\varphi|_{\mathfrak{m}}, \text{fv}(\psi_1) \subseteq \text{fv}(\varphi) \text{ and } \mathfrak{P} \text{ monotonically increasing}} \\ & \leq \text{maxval}_{\text{fv}(\varphi) \cup \{\mathbf{u}\}}(s, h) + |\varphi| \underbrace{\mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}})}_{\text{as } |\psi_1| + 1 \leq |\varphi|} \end{aligned}$$

Instead, for the right-hand side of (2) we have:

$$\begin{aligned} & \text{maxval}_{\text{fv}(\psi_2) \cup \{\mathbf{u}\}}(s, h+h') + |\psi_2| \mathfrak{P}(\text{card}(\text{fv}(\psi_2) \setminus \{\mathbf{u}\}), |\psi_2|_{\mathfrak{m}}) \\ & \leq \underbrace{\max(\text{maxval}_{\text{fv}(\psi_2) \cup \{\mathbf{u}\}}(s, h), \text{maxval}_{\text{fv}(\psi_2) \cup \{\mathbf{u}\}}(s, h'))}_{\text{as } \text{maxval}_Y(s, h+h') = \max(\text{maxval}_Y(s, h), \text{maxval}_Y(s, h'))} + |\psi_2| \mathfrak{P}(\text{card}(\text{fv}(\psi_2) \setminus \{\mathbf{u}\}), |\psi_2|_{\mathfrak{m}}) \\ & \leq \underbrace{\text{maxval}_{\text{fv}(\varphi) \cup \{\mathbf{u}\}}(s, h) + \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}})}_{\text{by definition of } h'} + |\psi_2| \underbrace{\mathfrak{P}(\text{card}(\text{fv}(\psi_2) \setminus \{\mathbf{u}\}), |\psi_2|_{\mathfrak{m}})}_{\text{by } |\psi_2|_{\mathfrak{m}} \leq |\varphi|_{\mathfrak{m}}, \text{fv}(\psi_2) \subseteq \text{fv}(\varphi) \text{ and } \mathfrak{P} \text{ monotonically increasing}} \\ & \leq \text{maxval}_{\text{fv}(\varphi) \cup \{\mathbf{u}\}}(s, h) + \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}}) + |\psi_2| \underbrace{\mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{\mathbf{u}\}), |\varphi|_{\mathfrak{m}})}_{\text{as } |\psi_2| + 1 \leq |\varphi|} \end{aligned}$$

□

Proof of (II). By induction on φ (with the natural induction hypothesis stating that the property holds for strict subformulae). The base case for atomic formulae is direct from their semantics. We also omit the trivial cases for $\varphi = \neg\psi$ and $\varphi = \psi_1 \wedge \psi_2$.

induction step: $\varphi = \psi_1 * \psi_2$. (\Rightarrow): Suppose $(s, h) \models \psi_1 * \psi_2$. There are two disjoint heaps h_1 and h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \psi_1$ and $(s, h_2) \models \psi_2$. By induction hypothesis, $\text{mc}(s, h_1, \psi_1) = \text{succeed}$ and $\text{mc}(s, h_2, \psi_2) = \text{succeed}$. The pair of heaps (h_1, h_2) is among the ones considered by the choose instruction in line 14. Therefore, from the if-then-else of line 15, $\text{mc}(s, h, \psi_1 * \psi_2) = \text{succeed}$.

(\Leftarrow): Suppose $\text{mc}(s, h, \psi_1 * \psi_2) = \text{succeed}$. From lines 14–15, this means that there is a pair of heaps (h_1, h_2) such that $h_1 \perp h_2$, $h_1 + h_2 = h$, $\text{mc}(s, h_1, \psi_1) = \text{succeed}$ and $\text{mc}(s, h_2, \psi_2) = \text{succeed}$. By induction hypothesis, $(s, h_1) \models \psi_1$ and $(s, h_2) \models \psi_2$. Thus, $(s, h) \models \psi_1 * \psi_2$.

induction step: $\varphi = \exists \mathbf{u} \psi$. (\Rightarrow): Suppose $(s, h) \models \exists \mathbf{u} \psi$, and so there is a location ℓ_1 such that $(s[\mathbf{u} \leftarrow \ell_1], h) \models \psi$. Let us suppose that φ is a formula of the weak fragment. Let $\alpha \stackrel{\text{def}}{=} |\varphi|_{\mathfrak{m}}$ and $\mathbf{x} \stackrel{\text{def}}{=} \text{fv}(\varphi)$ (note: $\mathbf{u} \notin \text{fv}(\varphi)$). Since $\approx_{\mathbf{x}, \alpha}^W$ is reflexive, we have $(s, h) \approx_{\mathbf{x}, \alpha}^W (s, h)$. By Lemma 5.23 there is $\ell_2 \leq \text{maxval}_{\mathbf{x}}(s, h) + 1$ such that $(s[\mathbf{u} \leftarrow \ell_1], h) \approx_{\mathbf{x}, \alpha}^W (s[\mathbf{u} \leftarrow \ell_2], h)$.

By Lemma 5.45(II), $(s[u \leftarrow \ell_2], h) \models \psi$, and therefore $\text{mc}(s[u \leftarrow \ell_2], h, \psi) = \text{succeed}$ by induction hypothesis. The location ℓ_2 is among the ones considered by the choose instruction in line 17. Thus, from line 18, $\text{mc}(s, h, \exists u \psi) = \text{succeed}$. The proof is analogous in the case that φ is not a formula of the weak fragment. It is sufficient to consider $\approx_{X,\alpha}^S$ instead of $\approx_{X,\alpha}^W$, and rely on Lemma 5.42 and Lemma 5.45(I) instead of Lemma 5.23 and Lemma 5.45(II), respectively.

(\Leftarrow): Suppose $\text{mc}(s, h, \exists u \psi) = \text{succeed}$. From lines 17 and 18, this means that there is a location $\ell \leq \text{maxval}_{\text{fv}(\varphi)}(s, h) + 1$ such that $\text{mc}(s[u \leftarrow \ell], h, \psi) = \text{succeed}$. By induction hypothesis, $(s[u \leftarrow \ell], h) \models \psi$. Thus, $(s, h) \models \exists u \psi$.

induction step: $\varphi = \psi_1 \multimap \psi_2$. (\Rightarrow): Suppose $(s, h) \models \psi_1 \multimap \psi_2$, and so there is a heap h_1 disjoint from h and such that $(s, h_1) \models \psi_1$ and $(s, h+h_1) \models \psi_2$. Notice that φ is a formula of the weak fragment. Let $X \stackrel{\text{def}}{=} \text{fv}(\varphi) \setminus \{u\}$ and $\alpha \stackrel{\text{def}}{=} |\varphi|_m$. We have $(s, h) \approx_{X,\alpha+\text{card}(X)}^W (s, h)$ by reflexivity of $\approx_{X,\alpha+\text{card}(X)}^W$. By Lemma 5.43, there is a heap h'_1 disjoint from h and such that $(s, h_1) \approx_{X,\alpha}^S (s, h'_1)$, $(s, h+h_1) \approx_{X,\alpha}^W (s, h+h'_1)$, and $\text{maxval}_{X \cup \{u\}}(s, h'_1) \leq \text{maxval}_{X \cup \{u\}}(s, h) + \mathfrak{P}(\text{card}(X), \alpha)$. By Lemma 5.45(I) $(s, h'_1) \models \psi_1$, and by Lemma 5.45(II), $(s, h+h'_1) \models \psi_2$. By induction hypothesis, $\text{mc}(s, h'_1, \psi_1) = \text{succeed}$ and $\text{mc}(s, h+h'_1, \psi_2) = \text{succeed}$. The heap h'_1 is among the ones considered by the choose instruction in lines 20 and 21. Thus, from line 22, $\text{mc}(s, h, \psi_1 \multimap \psi_2) = \text{succeed}$.

(\Leftarrow): Suppose $\text{mc}(s, h, \psi_1 \multimap \psi_2) = \text{succeed}$. From lines 20–22, there is a heap h' disjoint from h and s.t. $\text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h') \leq \text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h) + \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_m)$, $\text{mc}(s, h', \psi_1) = \text{succeed}$ and $\text{mc}(s, h+h', \psi_2) = \text{succeed}$. By induction hypothesis, $(s, h') \models \psi_1$ and $(s, h+h') \models \psi_2$. Thus, $(s, h) \models \psi_1 \multimap \psi_2$. \square

Lemma 5.49 allows us to prove the main result of the chapter.

Theorem 5.50. The satisfiability problem of $\text{SL}([\exists]_1, *, [-*, \hookrightarrow^+])_W^S$ is PSPACE-complete.

Proof. The PSPACE-hardness is inherited from $\text{SL}(*, -*)$ [44]. We show that $\text{sat}(\varphi)$ runs in NPSPACE and it is a decision procedure for satisfiability, leading to the result by Savitch Theorem [127]. From line 3, the algorithm guesses a memory state (s, h) such that $\text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s, h)$ is at most $2\mathfrak{Q}(|\varphi|) + \text{card}(\text{fv}(\varphi) \cup \{u\})$, where \mathfrak{Q} is a polynomial in $\mathcal{O}(|\varphi|^9)$. Afterwards, the algorithm calls $\text{mc}(s, h, \varphi)$ (line 4). By Lemma 5.49(I), this procedure runs in space

$$\mathcal{O}(\mathfrak{Q}(|\varphi|) + \text{card}(\text{fv}(\varphi) \cup \{u\}) + |\varphi| \mathfrak{P}(\text{card}(\text{fv}(\varphi) \setminus \{u\}), |\varphi|_m)).$$

where \mathfrak{P} is a polynomial in $\mathcal{O}(\text{card}(\text{fv}(\varphi) \setminus \{u\}) |\varphi|_m^4)$. As $|\varphi|_m \leq |\varphi|^2$ and $\text{card}(\text{fv}(\varphi) \cup \{u\}) \leq |\varphi|$, we conclude that $\text{sat}(\varphi)$ runs in space $\mathcal{O}(|\varphi|^{10})$. We now show that

$$\varphi \text{ is satisfiable if and only if } \text{sat}(\varphi) = \text{succeed}.$$

(\Leftarrow): Suppose $\text{sat}(\varphi) = \text{succeed}$. Form the lines 3 and 4 there is a heap (s, h) such that $\text{mc}(s, h, \varphi) = \text{succeed}$. By Lemma 5.49(II), $(s, h) \models \varphi$ and therefore φ is satisfiable.

(\Rightarrow): Suppose φ satisfiable. By Corollary 5.48, it is satisfied by a memory state (s, h) such that $\text{card}(h) \leq \mathfrak{Q}(|\varphi|)$. Thus, $\text{dom}(h) \cup \text{ran}(h)$ contains at most $2\mathfrak{Q}(|\varphi|)$ locations. Regarding the store, $s(\text{fv}(\varphi) \cup \{u\}) \setminus (\text{dom}(h) \cup \text{ran}(h))$ contains at most $\text{card}(\text{fv}(\varphi) \cup \{u\})$ locations. Therefore, (s, h) is $(\text{fv}(\varphi) \cup \{u\})$ -heap-isomorphic to a memory state (s', h') whose maximum value $\text{maxval}_{\text{fv}(\varphi) \cup \{u\}}(s', h')$ is at most $2\mathfrak{Q}(|\varphi|) + \text{card}(\text{fv}(\varphi) \cup \{u\})$. The memory state (s', h') is among the ones considered by the choice instruction in line 3. By Proposition 2.10, $(s', h') \models \varphi$. By Lemma 5.49(II), $\text{mc}(s', h', \varphi) = \text{succeed}$. Therefore, $\text{sat}(\varphi) = \text{succeed}$. \square

Corollary 5.51. The decision problems for the properties of acyclicity and garbage freedom of formulae in $\text{SL}([\exists]_1, *, [\neg*, \hookrightarrow^{\dagger}]_{\mathcal{W}}^S)$ can be decided in PSPACE.

Proof. By Theorem 5.50 and classical arguments, the validity and the entailment problems of $\text{SL}([\exists]_1, *, [\neg*, \hookrightarrow^{\dagger}]_{\mathcal{W}}^S)$ can be solved in PSPACE (we recall that the entailment $\varphi \models \psi$ holds if and only if $\varphi \wedge \neg\psi$ is unsatisfiable). Since the decision problems for the properties of acyclicity and garbage freedom can be expressed as entailment queries (Section 5.1.2), we conclude. \square

5.6.2 One last step: the $\neg*$ -simulation property.

In the last section, we derived a polynomial small-heap property for $\text{SL}([\exists]_1, *, [\neg*, \hookrightarrow^{\dagger}]_{\mathcal{W}}^S)$, under the assumption that Lemma 5.43 holds. The last task of the chapter is to show that this is indeed the case. Let us recall the statement of this lemma.

Lemma 5.43 ($\neg*$ -simulation). Suppose $(s, h) \approx_{X, \alpha + \text{card}(X)}^{\mathcal{W}} (s', h')$. For every heap h_1 disjoint from h there is a heap h'_1 disjoint from h' such that

- (I) $(s, h_1) \approx_{X, \alpha}^S (s', h'_1)$ and $(s, h + h_1) \approx_{X, \alpha}^{\mathcal{W}} (s', h' + h'_1)$,
- (II) $\text{maxval}_{X \cup \{u\}}(s', h'_1) \leq \text{maxval}_{X \cup \{u\}}(s', h') + \mathfrak{P}(\text{card}(X), \alpha)$.

Despite being easier than the $*$ -simulation property, the construction we develop in order to show the $\neg*$ -simulation property is still quite technical. One of the difficulties in building h'_1 is to analyse which locations are labelled, with respect to the set $\text{Lab}[s]_{s', h'_1}^X$. In order to simplify this task, we show an easy criterion to check whether a location is not labelled.

Lemma 5.52. Let (s, h) be a memory state and let $\ell \in \text{Lab}[s]_{s, h}^X$ be a labelled location that is not assigned to a program variable in X . If $\ell \in \text{dom}(h)$, then there are two distinct locations ℓ_1 and ℓ_2 such that $h(\ell_1) = h(\ell_2) = \ell$.

Proof. The proof is straightforward when ℓ corresponds to a meet-point variable, say $m(x, y)$. Indeed, $[m(x, y)]_{s, h}^X = \ell$ implies that h witnesses two disjoint non-empty path, one going from $s(x)$ to ℓ and one going from $s(y)$ to ℓ . The fact that the two paths are disjoint and non-empty ends the proof. Otherwise, suppose that ℓ corresponds to an end-point variable, say $e(x)$. As $\ell \in \text{dom}(h)$, $[e(x)]_{s, h}^X = \ell$ implies that there is $\delta \geq 1$ such that $h^\delta(s(x)) = \ell$, and ℓ belongs to a cycle whereas $h^{\delta-1}(s(x))$ does not. As ℓ belongs to a cycle, there is a location $\ell' \neq h^{\delta-1}(s(x))$ in this cycle, for which $h(\ell') = \ell$. \square

Thanks to Lemma 5.52, we can carefully construct the heap h'_1 required by Lemma 5.43 so that locations in $\text{dom}(h'_1)$ that we do not want to be labelled all have at most one predecessor. For instance, if we want h'_1 to witness an unlabelled cycle of length 2, we construct h'_1 so that there are distinct locations ℓ_1 and ℓ_2 that do not correspond to program variable in X , and

- $\{\ell_1, \ell_2\}$ is a minimal set describing a cycle in h'_1 ,
- no location ℓ' other than ℓ_1 and ℓ_2 is such that $h(\ell') = \ell_1$ or $h(\ell') = \ell_2$.

By Lemma 5.52, we conclude ℓ_1 and ℓ_2 do not belong to $\text{Lab}[s]_{s', h'_1}^X$, and thus $\{\ell_1, \ell_2\}$ describes an unlabelled cycle. At last, let us prove Lemma 5.43.

Proof of Lemma 5.43. As done in the statement of the lemma, we consider two memory states (s, h) and (s', h') such that $(s, h) \approx_{X, \alpha + \text{card}(X)}^{\mathcal{W}} (s', h')$, and let h_1 be a heap disjoint from h . We want to define h'_1 so that the cardinality constraint imposed by (II) is satisfied by construction. Afterwards, the main technical developments involve showing that h'_1 satisfies (I), i.e.

$$(s, h_1) \approx_{\mathbf{X}, \alpha}^{\mathcal{S}} (s', h'_1), \quad (s, h + h_1) \approx_{\mathbf{X}, \alpha}^{\mathcal{W}} (s', h' + h'_1).$$

As a first step, we want to specify a set of locations that, with respect to the heap h'_1 we later construct, should correspond to labelled locations in $\text{Lab}[\mathcal{S}]_{s', h'_1}^{\mathbf{X}}$. We need to be careful, as some of them could already belong to $\text{Lab}[\mathcal{W}]_{s', h'}^{\mathbf{X}}$. Moreover, as we want (s, h_1) and (s', h'_1) (resp. $(s, h + h_1)$ and $(s', h' + h'_1)$) to satisfy the same core formulae of the form $\mathbf{t} = \mathbf{t}'$, these locations should be related to the locations in $\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}}$. For this reason, we define an injection

$$\mathbf{f} : \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}} \cup \text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \rightarrow \text{LOC},$$

that shall satisfy the following properties:

- 0_f . if $s(u) \in \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}} \cup \text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}}$, then $\mathbf{f}(s(u)) = s'(u)$. Otherwise, $s'(u) \notin \text{ran}(\mathbf{f})$.
 - 1_f . for every $\mathbf{t} \in T[\mathcal{W}]^{\mathbf{X}}$, if $\llbracket \mathbf{t} \rrbracket_{s, h}^{\mathbf{X}}$ is defined then $\mathbf{f}(\llbracket \mathbf{t} \rrbracket_{s, h}^{\mathbf{X}}) = \llbracket \mathbf{t} \rrbracket_{s', h'}^{\mathbf{X}}$.
- Moreover, for every $\ell \in \text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}}$,
- 2_f . if $\ell \in \text{Pred}[\mathcal{W}]_{s, h}^{\mathbf{X}}(\mathbf{x})$ for some $\mathbf{x} \in \mathbf{X}$, then $\mathbf{f}(\ell) \in \text{Pred}[\mathcal{W}]_{s', h'}^{\mathbf{X}}(\mathbf{x})$,
 - 3_f . if $\ell \in \text{Self}[\mathcal{W}]_{s, h}^{\mathbf{X}}$, then $\mathbf{f}(\ell) \in \text{Self}[\mathcal{W}]_{s', h'}^{\mathbf{X}}$,
 - 4_f . if $\ell \in \text{Rem}[\mathcal{W}]_{s, h}^{\mathbf{X}}$ then $\mathbf{f}(\ell) \in \text{Rem}[\mathcal{W}]_{s', h'}^{\mathbf{X}}$,
 - 5_f . otherwise, if $\ell \neq s(u)$ then $\mathbf{f}(\ell) > \text{maxval}_{\mathbf{X} \cup \{u\}}(s', h')$.

We notice that these properties exhaust every possible location in $\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \cup \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}}$. The case where $s(u)$ is a labelled location is considered separately, in (0_f) . Otherwise, the locations in $\text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}}$ are dealt with in (1_f) , whereas (2_f) – (5_f) characterise \mathbf{f} for the remaining labelled locations. Below, we show the existence of \mathbf{f} , and track our progress with respect to (II) , by bounding the number of locations in $\text{ran}(\mathbf{f})$ that exceed $\text{maxval}_{\mathbf{X} \cup \{u\}}(s', h')$.

A. An injection $\mathbf{f} : \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}} \cup \text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \rightarrow \text{LOC}$ satisfying (0_f) – (5_f) exists,

B. $\text{card}(\{\ell \mid \mathbf{f}(\ell) > \text{maxval}_{\mathbf{X} \cup \{u\}}(s', h')\}) \leq \text{card}(\mathbf{X})$.

Proof of (A). Fundamentally, the existence of \mathbf{f} follows from $(s, h) \approx_{\mathbf{X}, \alpha + \text{card}(\mathbf{X})}^{\mathcal{W}} (s', h')$, where we highlight the parameter $\alpha + \text{card}(\mathbf{X})$. Thanks to the formulae $u = t$ and $t_1 = t_2$ of $\text{Core}[\mathcal{W}](\mathbf{X}, \alpha + \text{card}(\mathbf{X}))$, there is an injection \mathbf{f} satisfying both (0_f) and (1_f) . Now, for the remaining constraints (2_f) – (5_f) , we notice that ℓ does not belong to $\text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}}$, and therefore these constraints are independent from (1_f) . In order to show that we can build an injection satisfying not only (0_f) and (1_f) but also (2_f) – (5_f) , it is sufficient to check that

1. $s(u) \in \text{Pred}[\mathcal{W}]_{s, h}^{\mathbf{X}}(\mathbf{x})$ if and only if $s'(u) \in \text{Pred}[\mathcal{W}]_{s', h'}^{\mathbf{X}}(\mathbf{x})$,
2. $\min(\text{card}(\text{Pred}[\mathcal{W}]_{s, h}^{\mathbf{X}}(\mathbf{x})), \text{card}(\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}}))$
 $= \min(\text{card}(\text{Pred}[\mathcal{W}]_{s', h'}^{\mathbf{X}}(\mathbf{x})), \text{card}(\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}})),$
3. $s(u) \in \text{Self}[\mathcal{W}]_{s, h}^{\mathbf{X}}$ if and only if $s'(u) \in \text{Self}[\mathcal{W}]_{s', h'}^{\mathbf{X}}$,
4. $\min(\text{card}(\text{Self}[\mathcal{W}]_{s, h}^{\mathbf{X}}), \text{card}(\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}}))$
 $= \min(\text{card}(\text{Self}[\mathcal{W}]_{s', h'}^{\mathbf{X}}), \text{card}(\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}})),$
5. $s(u) \in \text{Rem}[\mathcal{W}]_{s, h}^{\mathbf{X}}$ if and only if $s'(u) \in \text{Rem}[\mathcal{W}]_{s', h'}^{\mathbf{X}}$,
6. $\min(\text{card}(\text{Rem}[\mathcal{W}]_{s, h}^{\mathbf{X}}), \text{card}(\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}}))$
 $= \min(\text{card}(\text{Rem}[\mathcal{W}]_{s', h'}^{\mathbf{X}}), \text{card}(\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}})).$

Indeed, (1) , (3) and (5) are required in order to satisfy (0_f) , whereas (2) , (4) and (6) allows us to build an injection. Thanks to the formulae $u \in \text{pred}_{\mathbf{X}}^{\mathcal{W}}(\mathbf{x})$, $u \in \text{self}_{\mathbf{X}}^{\mathcal{W}}(\mathbf{x})$ and $u \in \text{rem}_{\mathbf{X}}^{\mathcal{W}}$, from $(s, h) \approx_{\mathbf{X}, \alpha + \text{card}(\mathbf{X})}^{\mathcal{W}} (s', h')$, we conclude that (1) , (3) and (5) hold. For the three remaining properties, we have $(\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus \text{Lab}[\mathcal{W}]_{s, h}^{\mathbf{X}}) \subseteq (\text{Lab}[\mathcal{S}]_{s, h_1}^{\mathbf{X}} \setminus s(\mathbf{X}))$

and therefore, by Lemma 5.25, $\text{card}(\text{Lab}[S]_{s,h_1}^X \setminus \text{Lab}[W]_{s,h}^X) \leq \text{card}(X)$. This allow us to derive (2), (4) and (6). Indeed, thanks to the core formulae $\text{pred}_X^W(x) \geq \beta$, $\text{self}_X^W \geq \beta$ and $\text{rem}_X^W \geq \beta$, where $\beta \in [1, \alpha + \text{card}(X)]$, from $(s, h) \approx_{X,\alpha+\text{card}(X)}^W (s', h')$ we have.

- $\min(\text{card}(\text{Pred}[W]_{s,h}^X(x)), \alpha + \text{card}(X)) = \min(\text{card}(\text{Pred}[W]_{s',h'}^X(x)), \alpha + \text{card}(X))$,
- $\min(\text{card}(\text{Self}[W]_{s,h}^X), \alpha + \text{card}(X)) = \min(\text{card}(\text{Self}[W]_{s',h'}^X), \alpha + \text{card}(X))$,
- $\min(\text{card}(\text{Rem}[W]_{s,h}^X), \alpha + \text{card}(X)) = \min(\text{card}(\text{Rem}[W]_{s',h'}^X), \alpha + \text{card}(X))$.

Proof of (B). Following the characterisation of f , $f(\ell) > \text{maxval}_{X \cup \{u\}}(s', h')$ holds only in the case (5_f). Thus, $\ell \in (\text{Lab}[S]_{s,h_1}^X \setminus \text{Lab}[W]_{s,h}^X) \subseteq (\text{Lab}[S]_{s,h_1}^X \setminus s(X))$. By Lemma 5.25, these locations are at most $\text{card}(X)$.

In view of (B), we can safely assume that the locations in $\{\ell \mid f(\ell) > \text{maxval}_{X \cup \{u\}}(s', h')\}$ belongs to the interval $[\text{maxval}_{X \cup \{u\}}(s', h'), \text{maxval}_{X \cup \{u\}}(s', h') + \text{card}(X)]$. That is,

$$6_f. \quad \max(\text{ran}(f)) \leq \text{maxval}_{X \cup \{u\}}(s, h) + \text{card}(X).$$

Similarly to the *-simulation property, we now consider specific subsets of $\text{dom}(h_1)$ and aim at defining similar sets of locations, that are later used to construct h'_1 . We define,

- for all $\ell \in s(X)$, $P_\ell \stackrel{\text{def}}{=} \text{Pred}[S]_{s,h_1}^X(\ell) \setminus \text{dom}(f)$,
- for all $\ell \in \text{Lab}[S]_{s,h_1}^X$, $S_\ell \stackrel{\text{def}}{=} \text{Path}[S]_{s,h_1}^X(\ell) \setminus \text{dom}(f)$,
- for all $\beta \in [1, \alpha]$, $C_\beta \stackrel{\text{def}}{=} \{L \in \text{Cycl}[S]_{s,h_1}^X(\beta) \mid L \cap \text{dom}(f) = \emptyset\}$,
- for all $\beta \in [1, \alpha]$, $\bar{C}_\beta \stackrel{\text{def}}{=} \{L \in \text{Cycl}[S]_{s,h_1}^X(\beta) \mid L \cap \text{dom}(f) \neq \emptyset\}$,
- $U \stackrel{\text{def}}{=} \{L \in \uparrow\text{Cycl}[S]_{s,h_1}^{X,\alpha} \mid L \cap \text{dom}(f) = \emptyset\}$,
- $\bar{U} \stackrel{\text{def}}{=} \{L \in \uparrow\text{Cycl}[S]_{s,h_1}^{X,\alpha} \mid L \cap \text{dom}(f) \neq \emptyset\}$,
- $R \stackrel{\text{def}}{=} \text{Rem}[S]_{s,h_1}^{X,\alpha} \setminus \text{dom}(f)$.

From Proposition 5.31, we conclude that these sets are mutually disjoint and, together with $\text{dom}(f)$, their union includes $\text{dom}(h_1)$. For simplicity, below we assume $s(X) = \{\ell_1^X, \dots, \ell_n^X\}$ and $\text{Lab}[S]_{s,h_1}^X = \{\ell_1^L, \dots, \ell_m^L\}$. We aim at defining the sets $P'_{f(\ell_1^X)}, \dots, P'_{f(\ell_n^X)}$, $S'_{f(\ell_1^X)}, \dots, S'_{f(\ell_m^L)}$, C'_1, \dots, C'_α , $\bar{C}'_1, \dots, \bar{C}'_\alpha$, U' , \bar{U}' and R' . In short, we call these sets the *prime sets*. Intuitively, the prime sets should mimic the “non-prime” sets defined above. For instance, once the definition of h'_1 is complete, we expect $P'_{f(\ell^X)}$ to satisfy the equality $P'_{f(\ell^X)} = \text{Pred}[S]_{s',h'_1}^X(f(\ell^X)) \setminus \text{ran}(f)$. Moreover, the cardinality of each prime set should be bounded and, together with $\text{ran}(f)$, they should cover $\text{dom}(h'_1)$, allowing us to prove (II). We write $\mathcal{P}(\alpha)$, $\mathcal{S}(\alpha)$, $\mathcal{L}(\alpha)$ and $\mathcal{R}(\alpha)$ for the upper bounds of the various core formulae in $\text{Sk}[S](X, \alpha)$. That is,

$$\mathcal{P}(\alpha) = \alpha, \quad \mathcal{S}(\alpha) = \frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3), \quad \mathcal{L}(\alpha) = \frac{1}{2}\alpha(\alpha+3)-1, \quad \mathcal{R}(\alpha) = \alpha.$$

The properties (1)–(8) that characterise the prime sets are given in Figure 5.20. There, we recall that given a set T of sets of locations, we write $[T]^b$ for $\bigcup_{L \in T} L$. Below, we show that these sets are well-defined, despite the number of required properties. We also derive a bound on the overall number of locations needed to define all the prime sets.

C. A family of sets satisfying (1)–(8) exists. They are disjoint from h' .

D. The following equalities are satisfied:

-
1. The prime sets are mutually disjoint,
 2. for every $\ell \in s(\mathbb{X})$, $P'_{\mathfrak{f}(\ell)}$ is a set of locations such that
 - (p₁) $\text{card}(P'_{\mathfrak{f}(\ell)}) = \min(\text{card}(P_\ell), \mathcal{P}(\alpha))$,
 - (p₂) $s(\mathbf{u}) \in P_\ell$ if and only if $s'(\mathbf{u}) \in P'_{\mathfrak{f}(\ell)}$,
 - (p₃) for every $\ell' \in P'_{\mathfrak{f}(\ell)} \setminus \{s'(\mathbf{u})\}$, $\ell' > \text{maxval}_{\mathbb{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbb{X})$.
 3. for every $\ell \in \text{Lab}[s]_{s, h_1}^{\mathbb{X}}$, $S'_{\mathfrak{f}(\ell)}$ is a set of locations such that
 - (s₁) $\text{card}(S'_{\mathfrak{f}(\ell)}) = \min(\text{card}(S_\ell), \mathcal{S}(\alpha))$,
 - (s₂) $s(\mathbf{u}) \in S_\ell$ if and only if $s'(\mathbf{u}) \in S'_{\mathfrak{f}(\ell)}$,
 - (s₃) for every $\ell' \in S'_{\mathfrak{f}(\ell)} \setminus \{s'(\mathbf{u})\}$, $\ell' > \text{maxval}_{\mathbb{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbb{X})$.
 4. for every $\beta \in [1, \alpha]$, C'_β is a set of sets of locations, such that
 - (c₁) $\text{card}(C'_\beta) = \min(\text{card}(C_\beta), \mathcal{L}(\alpha))$,
 - (c₂) $s(\mathbf{u}) \in [C_\beta]^\flat$ if and only if $s'(\mathbf{u}) \in [C'_\beta]^\flat$,
 - (c₃) every set $L \in C'_\beta$ has cardinality β ,
 - (c₄) for every $\ell \in [C'_\beta]^\flat \setminus \{s'(\mathbf{u})\}$, $\ell > \text{maxval}_{\mathbb{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbb{X})$,
 5. for every $\beta \in [1, \alpha]$, \overline{C}'_β is a set of sets of locations such that
 - (c₅) there is a bijection $\mathbf{g}_\beta : \overline{C}_\beta \rightarrow \overline{C}'_\beta$,
 - (c₆) $s(\mathbf{u}) \in [\overline{C}_\beta]^\flat$ if and only if $s'(\mathbf{u}) \in [\overline{C}'_\beta]^\flat$,
 - (c₇) for every $\ell \in \text{dom}(\mathfrak{f})$ and $L \in \overline{C}_\beta$, $\ell \in L$ if and only if $\mathfrak{f}(\ell) \in \mathbf{g}_\beta(L)$,
 - (c₈) every set $L \in \overline{C}'_\beta$ has cardinality β ,
 - (c₉) for every $\ell \in [\overline{C}'_\beta]^\flat \setminus (\text{ran}(\mathfrak{f}) \cup \{s'(\mathbf{u})\})$, $\ell > \text{maxval}_{\mathbb{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbb{X})$.
 6. U' is a set of sets of locations, such that
 - (u₁) $\text{card}(U') = \min(\text{card}(U), \mathcal{L}(\alpha))$,
 - (u₂) $s(\mathbf{u}) \in [U]^\flat$ if and only if $s'(\mathbf{u}) \in [U']^\flat$,
 - (u₃) for every $\ell \in [U']^\flat \setminus \{s'(\mathbf{u})\}$, $\ell > \text{maxval}_{\mathbb{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbb{X})$,
 - (u₄) every set $L \in U'$ has cardinality $\alpha + 1$.
 7. \overline{U}' is a set of sets of locations such that
 - (u₅) there is a bijection $\mathbf{g}^\uparrow : \overline{U} \rightarrow \overline{U}'$,
 - (u₆) $s(\mathbf{u}) \in [\overline{U}]^\flat$ if and only if $s'(\mathbf{u}) \in [\overline{U}']^\flat$,
 - (u₇) for every $\ell \in \text{dom}(\mathfrak{f})$ and $L \in \overline{U}$, $\ell \in L$ if and only if $\mathfrak{f}(\ell) \in \mathbf{g}^\uparrow(L)$,
 - (u₈) for every $L \in \overline{U}$, $\text{card}(\mathbf{g}^\uparrow(L) \setminus \text{ran}(\mathfrak{f})) = \alpha + 1$,
 - (u₉) for every $\ell \in [\overline{U}']^\flat \setminus (\text{ran}(\mathfrak{f}) \cup \{s'(\mathbf{u})\})$, $\ell > \text{maxval}_{\mathbb{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbb{X})$.
 8. R' is a set of locations such that
 - (r₁) $\text{card}(R') = \min(\text{card}(R), \mathcal{R}(\alpha))$,
 - (r₂) $s(\mathbf{u}) \in R$ if and only if $s'(\mathbf{u}) \in R'$,
 - (r₃) for every $\ell \in R' \setminus \{s'(\mathbf{u})\}$, $\ell > \text{maxval}_{\mathbb{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbb{X})$.

Figure 5.20: Properties of the prime sets.

- for all $\ell \in s(\mathbf{X})$, $P'_{\mathbf{f}(\ell)} \cap \text{ran}(\mathbf{f}) = \emptyset$, $U' \cap \text{ran}(\mathbf{f}) = \emptyset$,
- for all $\ell \in \text{Lab}[s]_{s,h_1}^{\mathbf{X}}$, $S'_{\mathbf{f}(\ell)} \cap \text{ran}(\mathbf{f}) = \emptyset$, $R' \cap \text{ran}(\mathbf{f}) = \emptyset$.
- for all $\beta \in [1, \alpha]$, $C'_{\beta} \cap \text{ran}(\mathbf{f}) = \emptyset$,

E. Excluding $\text{ran}(\mathbf{f})$, less than $(\text{card}(\mathbf{X}) + 1)((\alpha + 3)^4 - 1)$ locations appear in prime sets.

Proof of (C). First of all, we notice that, with the exception of $s'(\mathbf{u})$ and locations in $\text{ran}(\mathbf{f})$, every location appearing in a prime set must be greater than $\text{maxval}_{\mathbf{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbf{X})$ (see (p₃), (s₃), (c₄), (c₉), (u₃), (u₉) and (r₃)). By definition of maximum value and by (6_f), these locations do not belong to $\text{dom}(h')$, nor to $\{s'(\mathbf{u})\} \cup \text{ran}(\mathbf{f})$. Moreover, since there are infinitely many locations greater than $\text{maxval}_{\mathbf{X} \cup \{\mathbf{u}\}}(s', h') + \text{card}(\mathbf{X})$, we can easily assume that they do not appear in two different prime sets. Therefore, to prove (1) and show that the prime sets are disjoint from $\text{dom}(h')$, it is sufficient to show that:

γ_1 . Every location $\ell \in \{s'(\mathbf{u})\} \cup \text{ran}(\mathbf{f})$ appears in at most one prime set. If ℓ appears in a prime set, then $\ell \notin \text{dom}(h')$.

In order to show this result, we look at the “non-prime” sets $P_{\ell_1^{\mathbf{X}}}, \dots, P_{\ell_n^{\mathbf{X}}}, S_{\ell_1^{\mathbf{L}}}, \dots, S_{\ell_m^{\mathbf{L}}}, C_1, \dots, C_{\alpha}, \bar{C}_1, \dots, \bar{C}_{\alpha}, U, \bar{U}$ and R , which we know to be mutually disjoint. Therefore,

γ_2 . Every location in $\{s(\mathbf{u})\} \cup \text{dom}(\mathbf{f})$ appears in at most one “non-prime” set.

Moreover, as they are constructed from locations in $\text{dom}(h_1)$, we conclude that no location in these sets belongs to $\text{dom}(h)$. This allows us to conclude that

γ_3 . if $s(\mathbf{u})$ belongs to a “non-prime” set, then $s(\mathbf{u}) \notin \text{dom}(h')$.

Suppose that $s(\mathbf{u})$ belongs to a “non-prime” set. *Ad absurdum*, assume that $s'(\mathbf{u}) \in \text{dom}(h')$. We consider the Boolean combination of core formulae from $\text{Core}[\mathcal{W}](\mathbf{X}, 1)$ below:

$$\bigvee_{t \in T[\mathcal{W}]^{\mathbf{X}}} (u = t \wedge t \hookrightarrow _) \vee \bigvee_{x \in \mathbf{X}} u \in \text{pred}_{\mathbf{X}}^{\mathcal{W}}(x) \vee u \in \text{self}_{\mathbf{X}}^{\mathcal{W}} \vee u \in \text{rem}_{\mathbf{X}}^{\mathcal{W}}$$

It is rather easy to see that this formula is satisfied by a memory state (s'', h'') if and only if $s''(\mathbf{u}) \in \text{dom}(h'')$. Thanks to this formula, by $(s, h) \approx_{\mathbf{X}, \alpha + \text{card}(\mathbf{X})}^{\mathcal{W}} (s', h')$ and $s'(\mathbf{u}) \in \text{dom}(h')$, we conclude that $s(\mathbf{u}) \in \text{dom}(h)$. However, this contradicts the fact that $s(\mathbf{u})$ belongs to a “non-prime” set.

γ_4 . if $\ell \in \text{dom}(\mathbf{f})$ belongs to a “non-prime” set, then $\mathbf{f}(\ell) \notin \text{dom}(h')$.

Suppose that $\ell \in \text{dom}(\mathbf{f})$ belongs to a “non-prime” set. *Ad absurdum*, assume that $\mathbf{f}(\ell) \in \text{dom}(h')$. From the definition of \mathbf{f} , ℓ cannot correspond to the case (5_f), and thus it corresponds to one among (1_f)–(4_f). If it corresponds to (1_f), then there is a term $t \in T[\mathcal{W}]^{\mathbf{X}}$ such that $\llbracket t \rrbracket_{s,h}^{\mathbf{X}} = \ell$ and $\mathbf{f}(\ell) = \llbracket t \rrbracket_{s',h'}^{\mathbf{X}}$. Thanks to the formula $t \hookrightarrow _$, by $(s, h) \approx_{\mathbf{X}, \alpha + \text{card}(\mathbf{X})}^{\mathcal{W}} (s', h')$ and $\mathbf{f}(\ell) \in \text{dom}(h')$, we conclude that $\ell \in \text{dom}(h)$. Similarly, if ℓ corresponds to a case among (2_f)–(4_f), then $\ell \in \text{dom}(h)$.

In both cases, $\ell \in \text{dom}(h)$ contradicts the fact that ℓ is in a “non-prime” set.

The properties (γ_2), (γ_3) and (γ_4) allow not only to show (γ_1), but also to prove that the prime sets are well-defined. Indeed, with the constraints (p₂), (s₂), (c₂), (c₆), (u₂), (u₆), and (r₂), we ask $s'(\mathbf{u})$ to belong to a prime set Q' if and only if $s(\mathbf{u})$ belongs to the corresponding “non-prime” set Q . From (γ_3), we conclude that if $s'(\mathbf{u})$ appears in a prime set, then $s(\mathbf{u}) \notin \text{dom}(h')$. From (γ_2), only one “non-prime” set can contain $s(\mathbf{u})$ and therefore at most one prime set contains $s'(\mathbf{u})$. So, (γ_1) holds for $\ell = s'(\mathbf{u})$. Notice that this implies all the constraints required in (2), (3), (4), (6), and (8) can be satisfied. Indeed, apart from $s'(\mathbf{u})$, all the prime sets corresponding to these

constraints must only contain locations greater than $\text{maxval}_{X \cup \{u\}}(s', h') + \text{card}(X)$, which we already treated at the beginning of the proof. Let us consider a location $\ell \in \text{ran}(f)$ in a prime set. From the properties required by prime sets, ℓ belongs can only belong to $[\bar{C}'_\beta]^\flat$ for some $\beta \in [1, \alpha]$, or to $[\bar{U}']^\flat$, as shown in (c₇) and (u₇). These constraints imply that $f^{-1}(\ell)$ belongs to the corresponding “non-prime” set among $[\bar{C}_1]^\flat, \dots, [\bar{C}_1]^\flat, [\bar{U}']^\flat$. From (γ₂) we conclude that ℓ can only appear in at most one prime set. From (γ₄), $\ell \notin \text{dom}(h')$. Therefore, we conclude that (γ₁) is satisfied. Moreover, (5) and (7) also hold. Indeed, with the exception of $s'(u) \cup \text{ran}(f)$, the prime sets $[\bar{C}_1]^\flat, \dots, [\bar{C}_1]^\flat, [\bar{U}']^\flat$ corresponding to these constraints must only contain locations greater than $\text{maxval}_{X \cup \{u\}}(s', h') + \text{card}(X)$, which we already treated at the beginning of the proof.

Proof of (D). The proof of all these statements are very similar. In the following, we show that $S'_{f(\ell)} \cap \text{ran}(f) = \emptyset$, where $\ell \in \text{Lab}[s]_{s, h_1}^X$. Let $\ell' \in S'_{f(\ell)}$. From (s₃), either $\ell' = s'(u)$ or $\ell' > \text{maxval}_{X \cup \{u\}}(s', h') + \text{card}(X)$. In the latter case, by (6_f), we conclude that $\ell' \notin \text{ran}(f)$. In the former case, if $\ell' = s'(u)$ then, by (s₂), $s(u) \in S_\ell$. By definition of S_ℓ , $s(u) \notin \text{dom}(f)$. From (0_f), we conclude that $\ell' = s'(u) \notin \text{ran}(f)$.

Proof of (E). By (p₁), $\text{card}([\{P'_{f(\ell)} \mid \ell \in s(X)\}]^\flat) \leq \text{card}(X) \times \mathcal{P}(\alpha)$. By (s₁) and Lemma 5.25, $\text{card}([\{S'_{f(\ell)} \mid \ell \in \text{Lab}[s]_{s, h_1}^X\}]^\flat) \leq \text{card}(\text{Lab}[s]_{s, h_1}^X) \times \mathcal{S}(\alpha) \leq 2 \times \text{card}(X) \times \mathcal{S}(\alpha)$. By (c₁) and (c₃), $\text{card}([\{C'_\beta \mid \beta \in [1, \alpha]\}]^\flat) \leq \mathcal{L}(\alpha) \sum_{\beta \in [1, \alpha]} \beta = \mathcal{L}(\alpha)(\frac{1}{2}\alpha(\alpha + 1))$. Similarly, from (u₁) and (c₄), $\text{card}([\bar{U}']^\flat) = \mathcal{L}(\alpha)(\alpha + 1)$. From (r₁), $\text{card}(R') \leq \mathcal{R}(\alpha)$. Lastly, we look at the sets $\bar{C}'_1, \dots, \bar{C}_\alpha, \bar{U}'$. They all contain sets of locations where at least one location is in $\text{ran}(f)$. This means that $\text{card}(\bar{U}' \cup \bigcup_{\beta \in [1, \alpha]} \bar{C}'_\beta) \leq \text{card}(\text{ran}(f))$. As f is injective, $\text{card}(\text{ran}(f)) = \text{card}(\text{dom}(f)) = \text{card}(\text{Lab}[s]_{s, h_1}^X \cup \text{Lab}[w]_{s, h}^X)$. A part from $s(X)$, which is included in both $\text{Lab}[s]_{s, h_1}^X$ and $\text{Lab}[w]_{s, h}^X$, $\text{Lab}[w]_{s, h}^X$ contains the locations that, in (s, h) , correspond to next-point variables. By Lemma 5.25, $\text{card}(\text{Lab}[s]_{s, h_1}^X \cup \text{Lab}[w]_{s, h}^X) \leq 3 \times \text{card}(X)$. By (u₈) and (c₈), every set in $\bar{C}'_1, \dots, \bar{C}_\alpha, \bar{U}'$ contains at most $\alpha + 1$ locations that are not in $\text{ran}(f)$. We conclude that

$$\text{card}([\bar{U}' \cup \bigcup_{\beta \in [1, \alpha]} \bar{C}'_\beta]^\flat) \leq \text{card}(\text{ran}(f)) + (\alpha + 1)\text{card}(\text{ran}(f)) \leq 3 \times \text{card}(X)(\alpha + 2).$$

All considering, we conclude that the locations appearing in all prime sets are at most

$$\text{card}(X)(\mathcal{P}(\alpha) + 2\mathcal{S}(\alpha) + 3(\alpha + 2)) + \frac{1}{2} \times \mathcal{L}(\alpha)(\alpha + 1)(\alpha + 2) + \mathcal{R}(\alpha).$$

With a simple numerical analysis, one can show that $(\text{card}(X) + 1)((\alpha + 3)^4 - 1)$ is strictly above of the last expression, for every $\alpha \geq 1$ and $\text{card}(X) \geq 0$.

From (E) and (6_f), we can safely assume that the locations occurring in prime sets are (strictly) bounded by the location $\text{maxval}_{X \cup \{u\}}(s', h') + (\text{card}(X) + 1)(\alpha + 3)^4$. That is,

- (*) The maximal location in a prime set is less than $\text{maxval}_{X \cup \{u\}}(s', h') + (\text{card}(X) + 1)(\alpha + 3)^4$.

The heap h'_1 . We move to the definition of h'_1 . Fundamentally, we define this heap so that every location in its domain is either in $\text{ran}(f)$ or in a prime set. Similarly, every location in $\text{ran}(h'_1)$ shall be either in $\text{ran}(f)$ or in a prime set, with the exception of a single location ℓ_r . Thanks to (*), we let ℓ_r be the location $\text{maxval}_{X \cup \{u\}}(s', h') + (\text{card}(X) + 1)(\alpha + 3)^4$. The definition of h'_1 is achieved by first defining several subheaps, which are described in Figure 5.21. The definitions of the heaps, as well as their properties, are all quite straightforward, the only exception being the heap $s_{f(\ell)}$, where $\ell \in \text{Lab}[s]_{s, h_1}^X$. Let us show that this heap is well-defined.

- (s₉) A heap $s_{f(\ell)}$ satisfying (s₄)–(s₈) exists. Moreover, $\text{Path}[s]_{s, h_1}^X(\ell) = \emptyset$ iff $s_{f(\ell)} = \emptyset$.

predecessor heaps. Given $\ell \in s(\mathbf{X})$, we consider the heap $P_{f(\ell)}$ defined as

$$P_{f(\ell)} \stackrel{\text{def}}{=} \{\ell' \mapsto f(\ell) \mid \ell' \in P'_{f(\ell)} \text{ or } \ell' \in \text{ran}(f) \text{ and } f^{-1}(\ell') \in \text{Pred}[S]_{s,h_1}^{\mathbf{X}}(\ell)\}.$$

By definition, we have

- (p4) $\text{dom}(P_{f(\ell)}) = P'_{f(\ell)} \cup \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Pred}[S]_{s,h_1}^{\mathbf{X}}(\ell)\},$
- (p5) if $P_{f(\ell)} \neq \emptyset$, then $\text{ran}(P_{f(\ell)}) = \{f(\ell)\}.$

bounded cycle heaps. Let $\beta \in [1, \alpha]$. Given $L \in C'_\beta \cup \bar{C}'_\beta$, let $\ell_0 < \dots < \ell_{\beta-1}$ be the locations in L . By (c3) and (c8), $\text{card}(L) = \beta$. We define the heap $h_L = \{\ell_j \mapsto \ell_{j+1 \bmod \beta} \mid i \in [0, \beta-1]\}$. Afterwards, we consider the heap C_β defined as

$$C_\beta = h_{L_1} + \dots + h_{L_k},$$

where $\{L_1, \dots, L_k\} = C'_\beta \cup \bar{C}'_\beta$. By definition, we have

- (c10) Every $L \in C'_\beta \cup \bar{C}'_\beta$ describes a cycle in C_β , of length β ,
- (c11) $\text{dom}(C_\beta) = [C'_\beta \cup \bar{C}'_\beta]^\flat$,
- (c12) $\text{ran}(C_\beta) = [C'_\beta \cup \bar{C}'_\beta]^\flat$. More precisely, for every $L \in C'_\beta \cup \bar{C}'_\beta$ $C_\beta(L) = L$.

unbounded cycle heap. Given $L \in U' \cup \bar{U}'$, let $\ell_0 < \dots < \ell_{\text{card}(L)-1}$ be the locations in L . By (u4) and (u8), $\text{card}(L) \geq \alpha+1$. We define the heap $h_L = \{\ell_j \mapsto \ell_{j+1 \bmod \text{card}(L)} \mid i \in [0, \text{card}(L)-1]\}$. Afterwards, we consider the heap U defined as

$$U = h_{L_1} + \dots + h_{L_k},$$

where $\{L_1, \dots, L_k\} = U' \cup \bar{U}'$. By definition, we have

- (u10) Every $L \in U' \cup \bar{U}'$ is a minimal set describing a cycle in U , of length at least $\alpha+1$,
- (u11) $\text{dom}(U) = [U' \cup \bar{U}']^\flat$,
- (u12) $\text{ran}(U) = [U' \cup \bar{U}']^\flat$. More precisely, for every $L \in U' \cup \bar{U}'$ $U(L) = L$.

remainder heap. We consider the heap R defined as:

$$R \stackrel{\text{def}}{=} \{\ell' \mapsto \ell_r \mid \ell' \in R' \text{ or } \ell' \in \text{ran}(f) \text{ and } f^{-1}(\ell') \in \text{Rem}[S]_{s,h_1}^{\mathbf{X},\alpha}\}.$$

By definition, we have

- (r4) $\text{dom}(R) = R' \cup \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Rem}[S]_{s,h_1}^{\mathbf{X},\alpha}\},$
- (r5) if $R \neq \emptyset$, then $\text{ran}(R) = \{\ell_r\}$.

path heaps. Consider $\ell \in \text{Lab}[S]_{s,h_1}^{\mathbf{X}}$. We consider a heap $S_{f(\ell)}$ characterised by (s4)–(s8) below:

- (s4) $\text{dom}(S_{f(\ell)}) = S'_{f(\ell)} \cup \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Path}[S]_{s,h_1}^{\mathbf{X}}(\ell)\}.$

If $\text{dom}(S_{f(\ell)}) \neq \emptyset$, then

- (s5) $\text{dom}(S_{f(\ell)})$ is a minimal set describing a path $(\ell_0, \ell_1, \dots, \ell_{\text{card}(S_{f(\ell)})-1}, \ell_{\text{card}(S_{f(\ell)})})$ in $S_{f(\ell)}$, where $\ell_0 = f(\ell)$ and $\ell_{\text{card}(S_{f(\ell)})} = f(\text{sby}_{s,h_1}^{\mathbf{X}}(\ell))$,
- (s6) if $h_1(\ell) \in \text{dom}(f)$, then $\ell_1 = f(h_1(\ell))$. Otherwise, $\ell_1 \in S'_{f(\ell)}$,
- (s7) Let ℓ' be the only location in $\text{Path}[S]_{s,h_1}^{\mathbf{X}}(\ell)$ such that $h_1(\ell') = \text{sby}_{s,h_1}^{\mathbf{X}}(\ell)$. If $\ell' \in \text{dom}(f)$ then $\ell_{\text{card}(S_{f(\ell)})-1} = f(\ell')$. Otherwise, $\ell_{\text{card}(S_{f(\ell)})-1} \in S'_{f(\ell)}$,
- (s8) if $h_1^{\delta_1}(\ell) = s(u)$ for some $\delta_1 \in [0, \text{card}(\text{Path}[S]_{s,h_1}^{\mathbf{X}}(\ell))]$, then there is $\delta_2 \in [0, \text{card}(S_{f(\ell)})]$ s.t.

$$S_{f(\ell)}^{\delta_2}(f(\ell)) = s'(u), \quad \min(\delta_1, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha)),$$

$$\min(\text{card}(\text{Path}[S]_{s,h_1}^{\mathbf{X}}(\ell)) - \delta_1, \mathcal{S}_{\text{right}}(\alpha)) = \min(\text{card}(S_{f(\ell)}) - \delta_2, \mathcal{S}_{\text{right}}(\alpha)),$$

where $\mathcal{S}_{\text{left}}(\alpha) = \frac{1}{6}(\alpha+1)(\alpha+2)+1$ and $\mathcal{S}_{\text{right}}(\alpha) = \frac{1}{2}\alpha(\alpha+3)$.

Figure 5.21: Subheaps of h'_1 .

Proof of (s9). If $\text{Path}[\mathcal{S}]_{s,h_1}^X(\ell) = \emptyset$, then $\text{dom}(\mathfrak{f}) \cap \text{Path}[\mathcal{S}]_{s,h_1}^X(\ell) = \emptyset$ and, by definition of S_ℓ , we have $S_\ell = \emptyset$. From (s1) and (s4), we derive $\text{dom}(S_{\mathfrak{f}(\ell)}) = \emptyset$. Since (s5)–(s8) are only applied whenever $\text{dom}(S_{\mathfrak{f}(\ell)}) \neq \emptyset$, we conclude that (s9) is satisfied.

Below, we assume that $\text{Path}[\mathcal{S}]_{s,h_1}^X(\ell)$ is non-empty, and thus it is a minimal set describing a path in $\rho = (\ell_0, \dots, \ell_k)$, from $\ell_0 = \ell$ to $\ell_k = \mathbf{sby}_{s,h_1}^X(\ell)$. Thus, $k = \text{card}(\text{Path}[\mathcal{S}]_{s,h_1}^X(\ell))$. By definition of S_ℓ ,

$$\text{Path}[\mathcal{S}]_{s,h_1}^X(\ell) = S_\ell \cup (\text{Path}[\mathcal{S}]_{s,h_1}^X(\ell) \cap \text{dom}(\mathfrak{f})),$$

where the union on the right hand side of this equivalence is between disjoint sets. We aim at building the heap $S_{\mathfrak{f}(\ell)}$ that satisfies (s4)–(s8). As required by (s4), the domain of this heap should be exactly the union of the two disjoint (by (D)) sets $S'_{\mathfrak{f}(\ell)}$ and $Q \stackrel{\text{def}}{=} \{\ell' \in \text{ran}(\mathfrak{f}) \mid \mathfrak{f}^{-1}(\ell') \in \text{Path}[\mathcal{S}]_{s,h_1}^X(\ell)\}$. As required by (s5), we want $\text{dom}(S_{\mathfrak{f}(\ell)})$ to describe a path going from $\mathfrak{f}(\ell)$ to $\mathfrak{f}(\mathbf{sby}_{s,h_1}^X(\ell))$. Notice that, as $\text{Path}[\mathcal{S}]_{s,h_1}^X(\ell)$ is non-empty, $\ell \in \text{Path}[\mathcal{S}]_{s,h_1}^X(\ell)$, and therefore $\mathfrak{f}(\ell) \in Q$.

We define $S_{\mathfrak{f}(\ell)}$ by first defining three disjoint heaps h_{pre} , h_{ct} and h_{sfx} . Informally, with respect to the path ρ' described by $S'_{\mathfrak{f}(\ell)}$, we expect h_{pre} to describe a prefix of ρ' , h_{sfx} to describe a suffix of ρ' , and lastly h_{ct} to describe the remaining (central) part of ρ' . We start by defining h_{pre} using the strategy below:

```

if  $\ell_1 \in \text{dom}(\mathfrak{f})$  then
   $h_{pre} \leftarrow \{\mathfrak{f}(\ell) \mapsto \mathfrak{f}(\ell_1)\},$ 
else if  $\ell_1 = s(u)$  then
   $h_{pre} \leftarrow \{\mathfrak{f}(\ell) \mapsto s'(u)\}$ 
else
  let  $\ell'$  be a location in  $S'_{\mathfrak{f}(\ell)}$  such that  $\ell' \neq s'(u)$ ,
   $h_{pre} \leftarrow \{\mathfrak{f}(\ell) \mapsto \ell'\}.$ 

```

Thanks to (0_f), (s1) and (s2), the heap h_{pre} is well-defined. Let ℓ' be the only location in $\text{ran}(h_{pre})$. Directly from the definition of h_{pre} , the following three properties are satisfied:

- pre1. $\text{dom}(h_{pre}) = \{\mathfrak{f}(\ell)\}$, $\text{ran}(h_{pre}) = \{\ell'\}$,
- pre2. $\ell_1 \in \text{dom}(\mathfrak{f})$ if and only if $\ell' \in \text{ran}(\mathfrak{f})$. If $\ell_1 \in \text{dom}(\mathfrak{f})$ then $\ell' = \mathfrak{f}(\ell_1)$.
- pre3. $\ell_1 \in S_\ell$ if and only if $\ell' \in S'_{\mathfrak{f}(\ell)}$,
- pre4. $\ell_1 = s(u)$ if and only if $\ell' = s'(u)$.

The proof splits depending on whether $k = 1$, $k = 2$ or $k > 2$.

case: $k = 1$. We have $\rho = (\ell_0, \ell_1)$, where $\ell_0 = \ell$ and $\ell_1 = \mathbf{sby}_{s,h_1}^X(\ell)$. So, $\text{Path}[\mathcal{S}]_{s,h_1}^X(t) = \{\ell\} \subseteq \text{dom}(\mathfrak{f})$, and thus $S_\ell = \emptyset$. By (s1), $S_{\mathfrak{f}(\ell)} = \emptyset$ and, by definition, $Q = \{\mathfrak{f}(\ell)\}$. We define $S_{\mathfrak{f}(\ell)}$ as h_{pre} . As $\ell_1 = \mathbf{sby}_{s,h_1}^X(\ell) \in \text{ran}(\mathfrak{f})$, by (pre1) and (pre2), we have $S_{\mathfrak{f}(\ell)} = \{\mathfrak{f}(\ell) \mapsto \mathfrak{f}(\mathbf{sby}_{s,h_1}^X(t))\}$. Thus, $\text{dom}(S_{\mathfrak{f}(\ell)}) = \{\mathfrak{f}(\ell)\} = Q \cup S'_{\mathfrak{f}(\ell)}$. We conclude that (s4)–(s7) are trivially satisfied, whereas (s8) holds directly from (0_f).

case: $k = 2$. We have $\rho = (\ell_0, \ell_1, \ell_2)$. So, $\text{Path}[\mathcal{S}]_{s,h_1}^X(\ell) = \{\ell, \ell_1\}$, where $\ell_0 = \ell \in \text{dom}(\mathfrak{f})$, $\ell \neq \ell_1$, and ℓ_1 can be either in $\text{dom}(\mathfrak{f})$ or in S_ℓ . By (pre2) this means that $\ell' \neq \mathfrak{f}(\ell_0)$ and, together with (pre3), we derive $\{\mathfrak{f}(\ell), \ell'\} = S_{\mathfrak{f}(\ell)} \cup Q$. By (pre1), $\ell' \notin \text{dom}(h_{pre})$. Lastly, notice that $\ell_2 = \mathbf{sby}_{s,h_1}^X(\ell)$, and thus from $\ell_1 \neq \ell_2$ (ρ is a minimal non-empty path) and by (pre2), $\ell' \neq \mathfrak{f}(\mathbf{sby}_{s,h_1}^X(\ell))$. We define $S_{\mathfrak{f}(\ell)}$ as $h_{pre} + \{\ell' \mapsto \mathfrak{f}(\mathbf{sby}_{s,h_1}^X(\ell))\}$. So, $\text{dom}(S_{\mathfrak{f}(\ell)}) = S_{\mathfrak{f}(\ell)} \cup Q$ is a minimal set describing the path $(\mathfrak{f}(\ell), \ell', \mathfrak{f}(\mathbf{sby}_{s,h_1}^X(\ell)))$.

We conclude that (s_4) and (s_5) are satisfied. Moreover, (s_6) and (s_7) holds directly from (pre_2) and (pre_3) . Lastly, (s_7) holds from (pre_4) and (0_f) .

case: $k > 2$. In this case, ℓ_0 , ℓ_{k-1} and ℓ_1 are all different. Since $\ell_0 = \ell \in \text{dom}(f)$, from (pre_2) , we conclude that $f(\ell) \neq \ell'$ and, by (pre_1) , $\ell' \notin \text{dom}(h_{\text{pre}})$. We define h_{sfx} following the strategy below:

```

if  $\ell_{k-1} \in \text{dom}(f)$  then
   $h_{\text{sfx}} \leftarrow \{f(\ell_{k-1}) \mapsto f(\text{sby}_{s,h_1}^X(\ell))\},$ 
else if  $\ell_{k-1} = s(u)$  then
   $h_{\text{sfx}} \leftarrow \{s'(u) \mapsto f(\text{sby}_{s,h_1}^X(\ell))\}$ 
else
  let  $\ell''$  be a location in  $S'_{f(\ell)}$  such that  $\ell'' \neq s'(u)$  and  $\ell'' \neq \ell'$ ,
   $h_{\text{sfx}} \leftarrow \{\ell'' \mapsto f(\text{sby}_{s,h}^X(\ell))\}.$ 

```

Notice that if the locations ℓ_1 , ℓ_{k-1} and $s(u)$ are all distinct and they all belong to S_ℓ , then, by (s_1) , $\text{card}(S_{f(\ell)}) \geq 3$. Because of this, the location ℓ'' selected in the last branch of the strategy above is well-defined. Together with (0_f) and (s_2) , this leads to h_{sfx} being well-defined. Let ℓ'' be the only location in $\text{dom}(h_{\text{sfx}})$. The following four properties are satisfied:

- sfx₁. $\text{dom}(h_{\text{sfx}}) = \{\ell''\}$ and $\text{ran}(h_{\text{sfx}}) = \{f(\text{sby}_{s,h_1}^X(\ell))\}$,
- sfx₂. $\ell_{k-1} \in \text{dom}(f)$ if and only if $\ell'' \in \text{ran}(f)$. If $\ell_{k-1} \in \text{dom}(f)$ then $\ell'' = f(\ell_{k-1})$,
- sfx₃. $\ell_{k-1} \in S_\ell$ if and only if $\ell'' \in S'_{f(\ell)}$,
- sfx₄. $\ell_{k-1} = s(u)$ if and only if $\ell'' = s'(u)$.

Since ℓ_{k-1} is different from $\ell_0 \in \text{dom}(f)$, by (sfx_2) we conclude that $f(\ell) \neq \ell''$ and therefore $h_{\text{pre}} \perp h_{\text{sfx}}$. Similarly, since ℓ_{k-1} is different from $\ell_k = \text{sby}_{s,h_1}^X(\ell) \in \text{dom}(f)$, $\ell'' \neq f(\text{sby}_{s,h_1}^X(\ell))$. Lastly, as $\ell_1 \neq \ell_{k-1}$, we have $\ell' \neq \ell''$. Indeed, if either ℓ_1 or ℓ_{k-1} belongs to $\text{dom}(f)$, then $\ell' \neq \ell''$ holds from (pre_2) and (sfx_2) . Otherwise, $\ell' \neq \ell''$ holds directly as the last branch of the strategy used to define h_{sfx} expressively asks for ℓ'' to be different from ℓ' . All considering, we conclude that $\text{dom}(h_{\text{pre}})$ and $\text{dom}(h_{\text{sfx}})$ describe two distinct paths $\rho_L = (f(\ell), \ell')$ and $\rho_R = (\ell'', f(\text{sby}_{s,h_1}^X(\ell)))$, respectively, where the only two locations that can be equal are $f(\ell)$ and $f(\text{sby}_{s,h_1}^X(\ell))$.

In order to define $S_{f(\ell)}$, we want to build a heap h_{ct} such that

- ct₁. $\text{dom}(h_{ct}) = (S'_{f(\ell)} \cup Q) \setminus \{f(\ell), \ell''\}$,
- ct₂. $\text{dom}(h_{ct})$ is a minimal set describing a path ρ_C going from ℓ' to ℓ'' .
- ct₃. if $h_1^{\delta_1}(\ell) = s(u)$ for some $\delta_1 \in [2, k-2]$, then there is $\delta_2 \in [1, \text{card}(h_{ct}) - 1]$ s.t.

$$h_{ct}^{\delta_2}(\ell') = s'(u), \quad \min(\delta_1, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta_2 + 1, \mathcal{S}_{\text{left}}(\alpha)),$$

$$\min(k - \delta_1, \mathcal{S}_{\text{right}}(\alpha)) = \min(\text{card}(S'_{f(\ell)} \cup Q) - (\delta_2 + 1), \mathcal{S}_{\text{right}}(\alpha)).$$

With these properties, the heap $S_{f(\ell)} = h_{\text{pre}} + h_{ct} + h_{\text{sfx}}$ is defined and satisfies (s_4) – (s_8) . Indeed, from (ct_1) together with $\text{dom}(h_L) \cup \text{dom}(h_{\text{sfx}}) = \{f(\ell), \ell''\} \subseteq S'_{f(\ell)} \cup Q$, we derive (s_4) . Moreover, by (ct_2) together with the definition of ρ_L and ρ_C , we derive (s_5) . The property (s_6) is taken care of by (pre_2) , whereas (s_7) follows from (sfx_2) . Lastly, (s_8) follows from (0_f) , which takes care of the cases where $\ell = s(u)$ or $\text{sby}_{s,h_1}^X(\ell) = s(u)$, from (pre_4) and (sfx_4) , which deals with the cases where $\ell_1 = s(u)$ or $\ell_{k-1} = s(u)$, and from (ct_3) , which deals with occurrences of $s(u)$ among $\{\ell_2, \dots, \ell_{k-2}\}$. To conclude the proof, we show the existence of h_{ct} .

Since \mathfrak{f} is an injection, we know that $\text{card}(\text{Path}[S]_{s,h_1}^X(\ell) \cap \text{dom}(\mathfrak{f})) = \text{card}(Q)$. From (s₁), this allows us to deduce that

$$\min(k, \mathcal{S}(\alpha)) = \min(\text{card}(S'_{\mathfrak{f}(\ell)} \cup Q), \mathcal{S}(\alpha)). \quad (\ddagger_1)$$

where we recall that $k = \text{card}(\text{Path}[S]_{s,h_1}^X(\ell)) = \text{card}(S_\ell \cup \text{Path}[S]_{s,h_1}^X(\ell) \cap \text{dom}(\mathfrak{f}))$. Moreover, from (0_f), (s₂), (pre₄) and (sfx₄),

$$s(u) \in \text{Path}[S]_{s,h_1}^X(\ell) \setminus \{\ell_0, \ell_1, \ell_{k-1}\} \text{ iff } s'(u) \in (S'_{\mathfrak{f}(\ell)} \cup Q) \setminus \{\mathfrak{f}(\ell), \ell', \ell''\} \quad (\ddagger_2)$$

Let ℓ'_1, \dots, ℓ'_j be distinct locations such that $\{\ell'_1, \dots, \ell'_j\} = (S'_{\mathfrak{f}(\ell)} \cup Q) \setminus \{\mathfrak{f}(\ell), \ell', \ell'', s'(u)\}$. Notice that, if $s(u) \in (S'_{\mathfrak{f}(\ell)} \cup Q) \setminus \{\mathfrak{f}(\ell), \ell', \ell''\}$, then $j = \text{card}(S'_{\mathfrak{f}(\ell)} \cup Q) - 4$. Let us write ℓ'_0 for ℓ' and ℓ'_{j+1} for ℓ'' . We define h_{ct} following the strategy below:

```

1: if  $s(u) \notin \text{Path}[S]_{s,h_1}^X(\ell) \setminus \{\ell_0, \ell_1, \ell_{k-1}\}$  then
2:    $h_{ct} \leftarrow \{\ell' \mapsto \ell'_1 \mapsto \dots \mapsto \ell'_j \mapsto \ell''\}$ ,
3: else
4:   let  $\delta \in [2, k-2]$  such that  $\ell_\delta = s(u)$ .
5:   if  $\delta < \mathcal{S}_{\text{left}}(\alpha)$  then
6:      $h_{ct} \leftarrow \{\ell' = \ell'_0 \mapsto \dots \mapsto \ell'_{\delta-2} \mapsto s'(u) \mapsto \ell'_{\delta-1} \mapsto \dots \mapsto \ell'_{j+1} = \ell''\}$ .
7:   else if  $k - \delta < \mathcal{S}_{\text{right}}(\alpha)$  then
8:      $h_{ct} \leftarrow \{\ell' = \ell'_0 \mapsto \dots \mapsto \ell_{j-(k-\delta-2)} \mapsto s'(u) \mapsto \ell'_{j-(k-\delta-3)} \mapsto \dots \mapsto \ell'_{j+1} = \ell''\}$ .
9:   else
10:     $h_{ct} \leftarrow \{\ell' = \ell'_0 \mapsto \dots \mapsto \ell'_{\mathcal{S}_{\text{left}}(\alpha)-2} \mapsto s'(u) \mapsto \ell'_{\mathcal{S}_{\text{left}}(\alpha)-1} \mapsto \dots \mapsto \ell'_{j+1} = \ell''\}$ .

```

To show that h_{ct} satisfies (ct₁)–(ct₃), we reason by cases, according to the strategy.

case: $s(u) \notin \text{Path}[S]_{s,h_1}^X(\ell) \setminus \{\ell_0, \ell_1, \ell_{k-1}\}$ (**line 2**). From (dd₂), $s'(u) \notin (S'_{\mathfrak{f}(\ell)} \cup Q) \setminus \{\mathfrak{f}(\ell), \ell', \ell''\}$. The heap h_{ct} is defined so that $\text{dom}(h_{ct}) = \{\ell', \ell'_1, \dots, \ell'_j\}$. The property (ct₃) is trivially satisfied, and by definition of ℓ'_1, \dots, ℓ'_j , we derive (ct₁). Lastly, one can notice that $\text{dom}(h_{ct})$ describes the path $\rho_C = (\ell', \ell'_1, \dots, \ell'_j, \ell'')$. Since all the locations $\ell', \ell'_1, \dots, \ell'_j, \ell''$ are distinct, $\text{dom}(h_{ct})$ is a minimal set describing this path. Thus, (ct₂) is satisfied.

case: there is $\delta \in [2, k-2]$ such that $s(u) = \ell_\delta$ and $\delta < \mathcal{S}_{\text{left}}(\alpha)$ (**line 6**). By (dd₂), $s'(u)$ belongs to $(S'_{\mathfrak{f}(\ell)} \cup Q) \setminus \{\mathfrak{f}(\ell), \ell', \ell''\}$. So, $j = \text{card}(S'_{\mathfrak{f}(\ell)} \cup Q) - 4$. We recall that $k = \text{card}(\text{Path}[S]_{s,h_1}^X(\ell))$. First of all, notice that h_{ct} is defined so that $\text{dom}(h_{ct})$ describes the path $\rho_C = (\ell'_0, \dots, \ell'_{\delta-2}, s'(u), \ell'_{\delta-1}, \dots, \ell'_{j+1})$. We need to check that h_{ct} is well-defined, that is $\ell'_{\delta-2}$ should belong to $\{\ell'_0, \dots, \ell'_j\}$. Since $\delta \in [2, k-2]$, we know that $\delta - 2 \geq 0$, and thus we just need to check that $j - (\delta - 2) \geq 0$. Ad absurdum, suppose $j - (\delta - 2) < 0$. So, $\text{card}(S'_{\mathfrak{f}(\ell)} \cup Q) - 2 - \delta < 0$ and we have:

$$\text{card}(S'_{\mathfrak{f}(\ell)} \cup Q) < 2 + \delta < 2 + \mathcal{S}_{\text{left}}(\alpha) \leq \mathcal{S}(\alpha),$$

where the last inequality holds as $\mathcal{S}(\alpha) = \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha)$ and $\mathcal{S}_{\text{right}}(\alpha) \geq 2$, for every $\alpha \geq 1$. From (dd₁) we conclude that $k = \text{card}(S'_{\mathfrak{f}(\ell)} \cup Q)$. However, this implies $k - 2 - \delta < 0$, which contradicts $\delta \in [2, k-2]$. We conclude that $\ell'_{\delta-2} \in \{\ell'_0, \dots, \ell'_j\}$, and so h_{ct} is well-defined. Fundamentally, from its definition, $h_{ct}^{\delta-1}(\ell') = s'(u)$. As in the previous case, (ct₁) and (ct₂) are obviously satisfied. Let us look at (ct₃). Let $\delta_1 \stackrel{\text{def}}{=} \delta$ and $\delta_2 \stackrel{\text{def}}{=} \delta - 1$. By definition of h_{ct} , $h_{ct}^{\delta_2}(\ell') = s'(u)$. Since $\delta_2 + 1 = \delta_1$, we conclude that $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta_2 + 1, \mathcal{S}_{\text{left}}(\alpha))$ is satisfied. From (dd₁) we conclude that

$$\min(k - \delta_1, \mathcal{S}(\alpha) - \delta) = \min(\text{card}(S'_{\mathfrak{f}(\ell)} \cup Q) - (\delta_2 + 1), \mathcal{S}(\alpha) - \delta).$$

Since $\mathcal{S}(\alpha) - \delta = \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha) - \delta \geq \mathcal{S}_{\text{right}}(\alpha)$, this equivalence implies (ct₃).

case: there is $\delta \in [2, k-2]$ such that $s(\mathbf{u}) = \ell_\delta$ and $k-\delta < \mathcal{S}_{\text{right}}(\alpha)$ (line 8). By (‡₂), $s'(\mathbf{u})$ belongs to $(S'_{\mathbf{f}(\ell)} \cup Q) \setminus \{\mathbf{f}(\ell), \ell', \ell''\}$. So, $j = \text{card}(S'_{\mathbf{f}(\ell)} \cup Q) - 4$. We notice that h_{ct} is defined so that $\text{dom}(h_{ct})$ describes the path

$$\rho_C = (\ell'_0, \dots, \ell'_{j-(k-\delta-2)}, s'(\mathbf{u}), \ell'_{j-(k-\delta-3)}, \dots, \ell'_{j+1}).$$

We need to check that h_{ct} is well-defined, that is $\ell'_{j-(k-\delta-2)}$ should belong to $\{\ell'_0, \dots, \ell'_j\}$. Since $\delta \in [2, k-2]$, we know that $j-(k-\delta-2) \leq j$, and thus we just need to check that $j-(k-\delta-2) \geq 0$. *Ad absurdum*, suppose $j-(k-\delta-2) < 0$. So, $\text{card}(S'_{\mathbf{f}(\ell)} \cup Q) - 2 - (k-\delta) < 0$ and we have:

$$\text{card}(S'_{\mathbf{f}(\ell)} \cup Q) < 2 + (k-\delta) < 2 + \mathcal{S}_{\text{right}}(\alpha) \leq \mathcal{S}(\alpha),$$

where the last inequality holds as $\mathcal{S}(\alpha) = \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha)$ and $\mathcal{S}_{\text{left}}(\alpha) \geq 2$, for every $\alpha \geq 1$. From (‡₁) we conclude that $k = \text{card}(S'_{\mathbf{f}(\ell)} \cup Q)$. However, this implies $k < 2 + (k-\delta)$ and so $\delta < 2$, which contradicts $\delta \in [2, k-2]$. We conclude that $\ell'_{j-(k-\delta-2)} \in \{\ell'_0, \dots, \ell'_j\}$, and so h_{ct} is well-defined. Fundamentally, from its definition, $h_{ct}^{(j-(k-\delta-3))}(\ell') = s'(\mathbf{u})$. As in the previous case, (ct₁) and (ct₂) are obviously satisfied. Let us look at (ct₃). Let $\delta_1 \stackrel{\text{def}}{=} \delta$ and $\delta_2 \stackrel{\text{def}}{=} j-(k-\delta-3)$. By definition of h_{ct} , $h_{ct}^{\delta_2}(\ell') = s'(\mathbf{u})$. Since $\text{card}(S'_{\mathbf{f}(\ell)} \cup Q) = j+4$, we have:

$$\text{card}(S'_{\mathbf{f}(\ell)} \cup Q) - (\delta_2 + 1) = j+4 - (j-(k-\delta-3)+1) = k-\delta = k-\delta_1,$$

which implies that the last equivalence required by (ct₃) holds. By (‡₁) we have

$$\min(k-(k-\delta), \mathcal{S}(\alpha)-(k-\delta)) = \min(j+4-(k-\delta), \mathcal{S}(\alpha)-(k-\delta)).$$

This implies that the second equivalence in (ct₃) holds, since $k-(k-\delta) = \delta_1$, $j+4-(k-\delta) = \delta_2+1$ and $\mathcal{S}(\alpha)-(k-\delta) \geq \mathcal{S}(\alpha)-\mathcal{S}_{\text{right}}(\alpha) = \mathcal{S}_{\text{left}}(\alpha)$. We conclude that (ct₃) is satisfied.

case: there is $\delta \in [2, k-2]$ such that $s(\mathbf{u}) = \ell_\delta$, $\delta \geq \mathcal{S}_{\text{left}}(\alpha)$ and $k-\delta \geq \mathcal{S}_{\text{right}}(\alpha)$ (line 10). From (‡₂), $s'(\mathbf{u}) \in (S'_{\mathbf{f}(\ell)} \cup Q) \setminus \{\mathbf{f}(\ell), \ell', \ell''\}$. Notice that h_{ct} is defined so that $\text{dom}(h_{ct})$ describes the path $\rho_C = (\ell'_0, \dots, \ell'_{\mathcal{S}_{\text{left}}(\alpha)-2}, s'(\mathbf{u}), \ell'_{\mathcal{S}_{\text{left}}(\alpha)-1}, \dots, \ell'_{j+1})$. Exactly as in the first case (corresponding to the line 2 of the strategy), one can show that $\ell'_{\mathcal{S}_{\text{left}}(\alpha)-2}$ belongs to $\{\ell'_0, \dots, \ell'_j\}$, and that therefore h_{ct} is well-defined. As in the previous cases, (ct₁) and (ct₂) are obviously satisfied. Let us look at (ct₃). Let $\delta_1 \stackrel{\text{def}}{=} \delta$ and $\delta_2 \stackrel{\text{def}}{=} \mathcal{S}_{\text{left}}(\alpha) - 1$. By definition of h_{ct} , $h_{ct}^{\delta_2}(\ell') = s'(\mathbf{u})$. Since $\delta_1 \geq \mathcal{S}_{\text{left}}(\alpha)$ and $\delta_2+1 = \mathcal{S}_{\text{left}}(\alpha)$, we conclude that $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta_2+1, \mathcal{S}_{\text{left}}(\alpha))$ is satisfied. From $\delta \geq \mathcal{S}_{\text{left}}(\alpha)$ and $k-\delta \geq \mathcal{S}_{\text{right}}(\alpha)$ we derive that $k \geq \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha) \geq \mathcal{S}(\alpha)$. From (‡₁) we conclude that $\text{card}(S'_{\mathbf{f}(\ell)} \cup Q) \geq \mathcal{S}(\alpha)$ and thus, by definition of δ_2 , $\text{card}(S'_{\mathbf{f}(\ell)} \cup Q) - (\delta_2 + 1) \geq \mathcal{S}_{\text{right}}(\alpha)$. As both $\text{card}(S'_{\mathbf{f}(\ell)} \cup Q) - (\delta_2 + 1)$ and $k-\delta$ are at least $\mathcal{S}_{\text{right}}(\alpha)$, (ct₃) holds.

We conclude that h_{ct} exists, which in turn implies that $\mathbf{s}_{\mathbf{f}(\ell)} = h_{\text{pre}} + h_{ct} + h_{\text{sfz}}$ is well-defined, and satisfies (s₄)–(s₈).

We are finally ready to define h'_1 . Recall that $s(\mathbb{X}) = \{\ell_1^\mathbb{X}, \dots, \ell_n^\mathbb{X}\}$ and $\text{Lab}[s]_{s,h_1}^\mathbb{X} = \{\ell_1^\mathbb{L}, \dots, \ell_m^\mathbb{L}\}$. Checking that the heaps $P_{\mathbf{f}(\ell_1^\mathbb{X})}, \dots, P_{\mathbf{f}(\ell_n^\mathbb{X})}, C_1, \dots, C_\alpha, U, R, S_{\mathbf{f}(\ell_1^\mathbb{L})}, \dots, S_{\mathbf{f}(\ell_m^\mathbb{L})}$ are disjoint is straightforward, and follows directly by (1), Proposition 5.31 and the fact that \mathbf{f} is an injection. We define h'_1 as the union:

$$h'_1 = P_{\mathbf{f}(\ell_1^\mathbb{X})} + \dots + P_{\mathbf{f}(\ell_n^\mathbb{X})} + C_1 + \dots + C_\alpha + U + R + S_{\mathbf{f}(\ell_1^\mathbb{L})} + \dots + S_{\mathbf{f}(\ell_m^\mathbb{L})}.$$

A quick check on the various heaps defined above leads to the following three properties:

(p₁) $\text{dom}(h'_1)$ is the union of all prime sets, together with a subset of $\text{ran}(\mathbf{f})$.

(ρ_2) $\text{ran}(h'_1)$ is included in the union of all prime sets, plus $\text{ran}(\mathbf{f})$ and $\{\ell_r\}$.

(ρ_3) Let \widehat{h} be the heap among $P_{\mathbf{f}}(\ell_1^X), \dots, P_{\mathbf{f}}(\ell_n^X), C_1, \dots, C_\alpha, U$ and R . Let $\ell \in \text{dom}(\widehat{h})$. If there is a location ℓ' such that $h'_1(\ell') = \ell$, then $\ell' \in \text{dom}(\widehat{h})$.

Proof of (ρ_1) . Directly from (p_4) , (c_{11}) , (u_{11}) , (r_4) and (s_4) .

Proof of (ρ_2) Directly from (p_5) , (c_{12}) , (u_{12}) , (r_5) and (s_5) .

Proof of (ρ_3) . Again from (p_5) , (c_{12}) , (u_{12}) , (r_5) and (s_5) , together with the injectivity of \mathbf{f} and the fact that the sets in Definition 5.29 are mutually disjoint.

We prove that h' is disjoint from h'_1 , and that satisfies both (I) and (II) . Let us first show that (II) holds and that $h' \perp h'_1$.

Proof of (II) . Directly from (ρ_1) and (ρ_2) , together with (\star) , (6_f) , and the definition of ℓ_r .

Proof that $h' \perp h'_1$. From (ρ_1) and (C) , it is sufficient to show that $\text{dom}(h'_1) \cap \text{ran}(\mathbf{f})$ is disjoint from $\text{dom}(h')$. Consider $\ell \in \text{dom}(h'_1) \cap \text{ran}(\mathbf{f})$. We show $\ell \notin \text{dom}(h')$. We divide the proof depending on the heaps introduced to define h'_1 .

case: $\ell \in \text{dom}(P_{\mathbf{f}}(\ell'))$, for some $\ell' \in s(X)$. From (p_4) , $\mathbf{f}^{-1}(\ell) \in \text{Pred}[S]_{s,h_1}^X(\ell')$. This implies that $\mathbf{f}^{-1}(\ell) \in \text{dom}(h_1)$ and $\mathbf{f}^{-1}(\ell) \notin \text{Lab}[S]_{s,h_1}^X$. Notice that this means that $\mathbf{f}^{-1}(\ell)$ is not assigned to a program variable in X . By definition of \mathbf{f} , $\mathbf{f}^{-1}(\ell)$ belongs to $\text{Lab}[W]_{s,h}^X$, and it corresponds to a next-point variable. Let $x \in X$ such that $\llbracket n(x) \rrbracket_{s,h}^X = \mathbf{f}^{-1}(\ell)$. By (1_f) , $\ell = \llbracket n(x) \rrbracket_{s',h'}^X$. Since h is disjoint from h_1 , $\mathbf{f}^{-1}(\ell) \notin \text{dom}(h)$, which in turn implies that $(s, h) \not\models n(x) \hookrightarrow \perp$. From $(s, h) \approx_{X,\alpha+\text{card}(X)}^W (s', h')$, we conclude that $(s', h') \not\models n(x) \hookrightarrow \perp$, and therefore $\ell \notin \text{dom}(h')$.

case: $\ell \in \text{dom}(C_\beta)$, for some $\beta \in [1, \alpha]$. Since $\ell \in \text{ran}(\mathbf{f})$, $\ell \in [\overline{C}_\beta]^\flat$. From (c_5) and (c_7) , $\mathbf{f}^{-1}(\ell) \in [\overline{C}_\beta]^\flat$. By definition of \overline{C}_β , this means that $\ell \in [\text{Cycl}[S]_{s,h_1}^X(\beta)]^\flat$. We conclude that $\mathbf{f}^{-1}(\ell) \in \text{dom}(h_1)$ and $\mathbf{f}^{-1}(\ell) \notin \text{Lab}[S]_{s,h_1}^X$. As in the previous case, this implies that ℓ does not belong to $\text{dom}(h')$.

case: $\ell \in \text{dom}(R)$ or $\ell \in \text{dom}(U)$. Both cases are similar to the previous two cases.

case: $\ell \in \text{dom}(S_{\mathbf{f}}(\ell'))$, for some $\ell' \in \text{Lab}[S]_{s,h_1}^X$. By (s_4) , $\mathbf{f}^{-1}(\ell) \in \text{Path}[S]_{s,h_1}^X(\ell')$. Thus, $\mathbf{f}^{-1}(\ell) \in \text{dom}(h_1)$, and therefore $\mathbf{f}^{-1}(\ell) \notin \text{dom}(h)$. If $\mathbf{f}^{-1}(\ell) \in \text{Lab}[W]_{s,h}^X$, we conclude $\ell \notin \text{dom}(h')$ as in the first case of the proof. Otherwise, suppose $\mathbf{f}^{-1}(\ell) \notin \text{Lab}[W]_{s,h}^X$, and so by definition of \mathbf{f} , $\mathbf{f}^{-1}(\ell) \in \text{Lab}[S]_{s,h_1}^X \setminus \text{Lab}[W]_{s,h}^X$. Since $\mathbf{f}^{-1}(\ell) \notin \text{dom}(h)$, $\mathbf{f}^{-1}(\ell)$ does not satisfy the premises of (2_f) – (4_f) , and thus by (5_f) we conclude that $\ell > \text{maxval}_{X \cup \{u\}}(s', h')$. By definition of maximum value, $\ell \notin \text{dom}(h')$.

To prove (I) we essentially rely on the set of properties used throughout the proofs of the two $*$ -simulation properties (see e.g. Lemmata 5.18 and 5.39).

Towards $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$. We show that $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$ by relying on the properties (s_O) – (s_F) below. The first set of properties, grouped under the label (s_O) , are auxiliary properties that shed some light on the labelled locations in $\text{Lab}[S]_{s',h'_1}^X$.

- s_O . (b) Let \widehat{h} be a heap among $C_1, \dots, C_\alpha, U, R$. For every $\ell \in \text{dom}(\widehat{h}) \cup \text{ran}(\widehat{h})$ and $x \in X$, the heap h'_1 does not witness a (possibly empty) path going from $s'(x)$ to ℓ .
- (c) Let $\ell' \in s(X)$. For every $\ell \in \text{dom}(P_{\mathbf{f}}(\ell'))$ and $x \in X$, the heap h'_1 does not witness a (possibly empty) path going from $s'(x)$ to ℓ .
- (d) Let $\ell' \in \text{Lab}[S]_{s,h_1}^X$. For every $\ell \in \text{dom}(S_{\mathbf{f}}(\ell')) \setminus \{\mathbf{f}(\ell')\}$, $\ell \notin \text{Lab}[S]_{s',h'_1}^X$.
- (e) If $\ell \in \text{Lab}[S]_{s',h'_1}^X$, then $\ell \in \mathbf{f}(\text{Lab}[S]_{s,h_1}^X)$.

Proof of (b). We divide the proof depending on \hat{h} .

case: $\hat{h} = c_\beta$, for some $\beta \in [1, \alpha]$. Consider $\ell \in \text{dom}(c_\beta) \cup \text{ran}(c_\beta)$. By (c₁₀)–(c₁₂), ℓ belongs to a set of locations $L \in C'_\beta \cup \bar{C}'_\beta$. L describes a cycle in h'_1 , which implies that $\ell \in \text{dom}(c_\beta)$. By (ρ₃), we know that if there is $\ell' \in \text{dom}(h'_1)$ such that $h'_1(\ell') = \ell$, then $\ell' \in \text{dom}(c_\beta)$. Therefore, to conclude the proof it is sufficient to show that no location in L corresponds to a program variable in X . Let $\ell' \in L$. We divide the proof depending on whether $L \in C'_\beta$ or $L \in \bar{C}'_\beta$.

case: $L \in C'_\beta$. By (c₄), we have either $\ell' = s(u)$ or $\ell' > \text{maxval}_{X \cup \{u\}}(s', h')$. In the latter case, $\ell' \notin s'(X)$ holds directly from the definition of maximum value. In the former case, by (s₂) we conclude that $s(u) \in [C_\beta]^\flat$, and so, by definition of C_β , $s(u) \notin \text{dom}(f)$. By (0_f), $\ell' = s(u) \notin \text{ran}(f)$. Since, by (1_f), we have $s'(X) \subseteq \text{ran}(f)$, we derive $\ell' \notin s'(X)$.

case: $L \in \bar{C}'_\beta$. By (u₉), either $\ell' \in \text{ran}(f) \cup \{s(u)\}$ or $\ell' > \text{maxval}_{X \cup \{u\}}(s', h')$. The latter case leads to $\ell' \notin s'(X)$, as in the previous case of the proof. In the former case, if $\ell' \in \text{ran}(f)$ then, by (c₇), $f^{-1}(\ell') \in [\bar{C}_\beta]^\flat$. By definition, $\bar{C}_\beta \subseteq \text{Cycl}[s]_{s, h_1}^X(\beta)$, which in turn implies that $f^{-1}(\ell')$ belongs to an unlabelled cycle described by an element in $\text{Cycl}[s]_{s, h_1}^X(\beta)$. Therefore, $f^{-1}(\ell') \notin s(X)$. By (1_f), we derive $\ell' \notin s'(X)$. Lastly, suppose $\ell' = s(u) \notin \text{ran}(f)$. Trivially, $\ell' \notin \text{ran}(f)$ implies $\ell' \notin s'(X)$, again by (1_f).

case: $\hat{h} = u$. Analogous to the previous case of the proof. We rely on the properties (u₁₀)–(u₁₂) of U , as well as the properties of U' and \bar{U}' , in (6) and (7).

case: $\hat{h} = r$. Consider $\ell \in \text{dom}(r) \cup \text{ran}(r)$. We divide the proof depending on whether $\ell \in \text{dom}(r)$ or $\ell \in \text{ran}(r)$.

case: $\ell \in \text{dom}(r)$. By (ρ₃), we know that if there is $\ell' \in \text{dom}(h'_1)$ such that $h'_1(\ell') = \ell$, then $\ell' \in \text{dom}(r)$. Therefore, to conclude that h'_1 does not witness a path going from a location in $s'(X)$ to ℓ , it is sufficient to show that $\text{dom}(r) \cap s'(X) = \emptyset$. Without loss of generality, we simply show that $\ell \notin s'(X)$. By (r₄), either $\ell \in R'$, or $\ell \in \text{ran}(f)$ and $f^{-1}(\ell) \in \text{Rem}[s]_{s, h_1}^{X, \alpha}$. First, suppose that $\ell \in \text{ran}(f)$ and $f^{-1}(\ell) \in \text{Rem}[s]_{s, h_1}^{X, \alpha}$. From the definition of $\text{Rem}[s]_{s, h_1}^{X, \alpha}$, $f^{-1}(\ell) \notin s(X)$. From (1_f), $\ell \notin s'(X)$. Otherwise, let us suppose that $\ell \in R'$. By (r₃), either $\ell = s(u)$ or $\ell > \text{maxval}_{X \cup \{u\}}(s', h')$. In the latter case, by definition of maximum value, $\ell \notin s'(X)$. In the former case, from (r₂) we derive $s(u) \in R$. By definition of R , $s(u) \notin \text{dom}(f)$. By (0_f), $\ell = s(u) \notin \text{ran}(f)$. by (1_f), $\ell \notin s'(X)$.

case: $\ell \in \text{ran}(r)$. By (r₅), $\ell = \ell_r = \text{maxval}_{X \cup \{u\}}(s', h') + (\text{card}(X) + 1)(\alpha + 3)^4$, by (★) and (6_f), we know that ℓ_r does not belong to a prime set, nor to $\text{ran}(f)$. Therefore, by (p₅), (c₁₂), (u₁₂), (p₄), and (s₅), we conclude that for all $\ell' \in \text{dom}(h'_1)$, if $h'_1(\ell') = \ell_r$ then $\ell' \in \text{dom}(r)$. In the previous case of the proof, we have shown that h'_1 does not witness any path going from a location in $s'(X)$ to a location in $\text{dom}(r)$. From $\ell_r \notin \text{ran}(f)$, by (1_f) we conclude that $\ell_r \notin s(X)$. We conclude that ℓ is not reached by a location in $s'(X)$.

Proof of (c). Since $\ell \in \text{dom}(P_{f(\ell')})$, by (ρ₃), we know that if there is $\ell'' \in \text{dom}(h'_1)$ such that $h'_1(\ell'') = \ell$, then $\ell'' \in \text{dom}(P_{f(\ell')})$. So, to conclude that h'_1 does not witness a path going from a location in $s'(X)$ to ℓ , it is sufficient to prove that $\text{dom}(P_{f(\ell')}) \cap s'(X) = \emptyset$. The proof of this statement carries out analogously to the proof of (b), in the case where $\hat{h} = r$ and $\ell \in \text{dom}(r)$.

Proof of (d). Let $\ell \in \text{dom}(s_{f(\ell')}) \setminus \{f(\ell')\}$. Similarly to (ρ_3) , we can show that if there is $\ell'' \in \text{dom}(h'_1)$ such that $h'_1(\ell'') = \ell$, then $\ell'' \in \text{dom}(s_{f(\ell')})$. Indeed, by (s_4) , ℓ belongs to either $S'_{f(\ell')}$ or to $\{\ell'' \in \text{ran}(f) \mid f^{-1}(\ell'') \in \text{Path}[s]_{s,h_1}^X(\ell')\}$. We conclude that:

- * $\ell \notin \text{ran}(s_{f(\tilde{\ell})})$, for all $\tilde{\ell} \in \text{Lab}[s]_{s,h_1}^X \setminus \{\ell'\}$.

By (s_4) and (s_5) , $\text{ran}(s_{f(\tilde{\ell})})$ is equivalent to

$$\left((S'_{f(\tilde{\ell})} \cup \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Path}[s]_{s,h_1}^X(\tilde{\ell})\}) \setminus \{f(\tilde{\ell})\} \right) \cup \{f(\text{sby}_{s,h_1}^X(\tilde{\ell}))\}.$$

Clearly, ℓ cannot belong to the set on the left hand side of the union above.

This follows directly from $\tilde{\ell} \neq \ell'$, which implies that $S'_{f(\tilde{\ell})} \cap S'_{f(\ell')} = \emptyset$ and

$\text{Path}[s]_{s,h_1}^X(\tilde{\ell}) \cap \text{Path}[s]_{s,h_1}^X(\ell') = \emptyset$. *Ad absurdum*, suppose $\ell = f(\text{sby}_{s,h_1}^X(\tilde{\ell}))$. By (D) , $\ell \notin S'_{f(\ell')}$ and so $\text{sby}_{s,h_1}^X(\tilde{\ell}) \in \text{Path}[s]_{s,h_1}^X(\ell')$. As $\text{sby}_{s,h_1}^X(\tilde{\ell}) \in \text{Lab}[s]_{s,h_1}^X$, we derive $\text{sby}_{s,h_1}^X(\tilde{\ell}) = \ell'$. Indeed, we recall that ℓ' is the only labelled location in $\text{Path}[s]_{s,h_1}^X(\ell')$. However, this contradicts $\ell \neq f(\ell')$.

- * $\ell \notin \text{ran}(P_{f(\tilde{\ell})})$, for all $\tilde{\ell} \in s(X)$.

By (p_5) , $\text{ran}(P_{f(\tilde{\ell})}) = \{f(\tilde{\ell})\}$. *Ad absurdum*, suppose $\ell = f(\tilde{\ell})$. By (D) , $\ell \notin S'_{f(\ell')}$ and so $\tilde{\ell} \in \text{Path}[s]_{s,h_1}^X(\ell')$. As $\tilde{\ell} \in s(X) \subseteq \text{Lab}[s]_{s,h_1}^X$, we derive $\tilde{\ell} = \ell'$. However, this contradicts $\ell \neq f(\ell')$.

- * $\ell \notin \text{ran}(C_\beta)$, for all $\beta \in [1, \alpha]$.

Follows by $\ell \in \text{dom}(s_{f(\ell')})$ and $s_{f(\ell')} \perp C_\beta$, as $\text{ran}(C_\beta) = \text{dom}(C_\beta)$ ((c_{11}) and (c_{12})).

- * $\ell \notin \text{ran}(U)$.

Similar to the previous case, as $\text{ran}(C_\beta) = \text{dom}(C_\beta)$, by (u_{11}) and (u_{12}) .

- * $\ell \notin \text{ran}(R)$.

By (r_5) , $\text{ran}(R) = \{\ell_r\}$. By definition of ℓ_r , it does not belong to a prime set, nor to $\text{ran}(f)$. Thus, $\ell_r \neq \ell$.

From this case analysis and the definition of h'_1 , every location ℓ'' such that $h'_1(\ell'') = \ell$ must belong to $\text{ran}(s_{f(\ell')})$. From (s_5) , we conclude that there is at most one location $\ell'' \in \text{ran}(s_{f(\ell')})$ such that $h'_1(\ell'') = \ell$. We apply Lemma 5.52, deriving $\ell \notin \text{Lab}[s]_{s',h'_1}^X$.

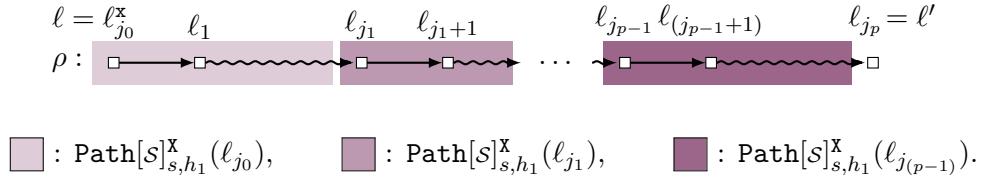
Proof of (e). Suppose $\ell \in \text{Lab}[s]_{s',h'_1}^X$. If $\ell = s'(x)$, for some $x \in X$, then, from $X \subseteq T[W]^L$ and (1_f) , we derive $\ell \in \text{ran}(f)$ and $f^{-1}(s(x))$. As $X \subseteq T[s]^X$, we conclude $\ell \in \text{Lab}[s]_{s,h_1}^X$. Otherwise, let us suppose that ℓ is not assigned to a program variable, and corresponds instead to either a meet-point variable or an end-point variable. In both cases, by definition of $\llbracket . \rrbracket_{s',h'_1}^X$, we conclude that h'_1 witnesses a non-empty path going from $s'(x)$ to ℓ , for some $\ell \in X$. In particular, this implies that $\ell \in \text{ran}(h'_1)$. By (b) , $\ell \notin \text{ran}(\hat{h})$, where \hat{h} is any heap among

$$P_{f(\ell_1^X)}, \dots, P_{f(\ell_n^X)}, C_1, \dots, C_\alpha, U, R.$$

Since ℓ is not assigned to a program variable, by (p_5) , for all $\ell' \in s(X)$, $\ell \notin \text{ran}(P_{f(\ell')})$. Therefore, by definition of h'_1 , there is $\ell' \in \text{Lab}[s]_{s,h'_1}^X$ such that $\ell \in \text{ran}(s_{f(\ell')})$. Now, if $\ell \in \text{dom}(s_{f(\ell')})$, then, by (d) , we conclude that $\ell = f(\ell')$, concluding the proof. Otherwise, $\ell \in \text{ran}(s_{f(\ell')}) \setminus \text{dom}(s_{f(\ell')})$ and, directly from (s_5) , we conclude that $\ell = f(\text{sby}_{s,h_1}^X(\ell'))$. As $\text{sby}_{s,h_1}^X(\ell') \in \text{Lab}[s]_{s,h_1}^X$, again, we conclude that $\ell \in f(\text{Lab}[s]_{s,h_1}^X)$.

s_A . For all $t \in T[s]^X$, $\llbracket t \rrbracket_{s,h_1}^X$ and $\llbracket t \rrbracket_{s',h'_1}^X$ are equidefined. If defined, $f(\llbracket t \rrbracket_{s,h_1}^X) = \llbracket t \rrbracket_{s',h'_1}^X$.

Before proving (s_A) , we show the two intermediate results below.

Figure 5.22: Splitting ρ depending on its labelled locations.

- (f) Let $\ell, \ell' \in \text{Lab}[s]_{s,h_1}^X$ be such that h_1 witnesses a minimal non-empty path $\rho = (\ell_0, \dots, \ell_k)$ going from ℓ to ℓ' . Let $j_0 < \dots < j_p$ be the indices in $[0, k]$ such that $\{\ell_{j_0}, \dots, \ell_{j_{p-1}}\} = \{\ell_0, \dots, \ell_k\} \cap \text{Lab}[s]_{s,h_1}^X$. Then, $\text{dom}(s_{f(\ell_{j_0})} + \dots + s_{f(\ell_{j_p})})$ is a minimal set describing a non-empty path in h'_1 going from $f(\ell)$ to $f(\ell')$.
- (g) Let $\ell, \ell' \in f(\text{Lab}[s]_{s,h_1}^X)$ such that h'_1 witnesses a non-empty path $\rho = (\ell_0, \dots, \ell_k)$ going from ℓ to ℓ' . Let $j_0 < \dots < j_p$ be the indices in $[0, k]$ such that $\{\ell_{j_0}, \dots, \ell_{j_p}\} = \{\ell_0, \dots, \ell_k\} \cap f(\text{Lab}[s]_{s,h_1}^X)$. Then, $\bigcup_{i \in [0, p-1]} \text{Path}[s]_{s,h_1}^X(\ell_{j_i})$ is a minimal set describing a non-empty path in h_1 going from $f^{-1}(\ell)$ to $f^{-1}(\ell')$.

Proof of (f). We remind the reader that as $\rho = (\ell_0, \dots, \ell_k)$ is minimal and non-empty, the locations $\ell_1, \dots, \ell_{k-1}$ are distinct, and they are different from ℓ_k . Since ℓ and ℓ' belong to $\text{Lab}[s]_{s,h_1}^X$, we have $\ell_{j_0} = \ell$ and $\ell_{j_p} = \ell'$. Moreover, for every $i \in [0, k-1]$, $\ell_i \in \text{dom}(h_1)$. Let $i \in [0, p-1]$. Following the definition of $\ell_{j_0}, \dots, \ell_{j_p}$, we conclude that, for every $i' \in [j_i + 1, j_{i+1} - 1]$, we have $\ell_{i'} \notin \text{Lab}[s]_{s,h_1}^X$. According to Definition 5.27, $s_{\text{by}_{s,h_1}^X}(\ell_{j_i}) = \ell_{j_{i+1}}$ and, according to Definition 5.29, $\text{Path}[s]_{s,h_1}^X(\ell_{j_i}) = \{\ell_{j_i}, \ell_{j_{i+1}}, \dots, \ell_{j_{i+1}-1}\}$. Notice that this implies that $\bigcup_{i \in [0, p-1]} \text{Path}[s]_{s,h_1}^X(\ell_{j_i})$ is a minimal set describing the path ρ , as roughly displayed in Figure 5.22. Now, since $\ell_{j_i} \in \text{dom}(h_1)$, from (s5) and $s_{f(j_i)} \subseteq h'_1$, we derive that $\text{dom}(s_{f(j_i)})$ is a minimal set describing a path in h'_1 going from $f(\ell_{j_i})$ to $f(\ell_{j_{i+1}})$. We recall that f is injective and so for all distinct $i, i' \in [0, p-1]$, $\ell_{j_i} \neq \ell_{j_{i'}}$, which in turn implies that $s_{f(j_i)} \perp s_{f(j_{i'})}$. Moreover, as ρ is minimal, for every $i \in [1, p-1]$, $\ell_{j_i} \neq \ell_{j_{i-1}} = \ell'$. As $\ell = \ell_{j_0}$, we conclude that $\text{dom}(s_{f(\ell_{j_0})} + \dots + s_{f(\ell_{j_p})})$ is a minimal set describing a non-empty path in h'_1 going from $f(\ell)$ to $f(\ell')$.

Proof of (g). Analogous to the proof of (f). Briefly, since $\rho = (\ell_0, \dots, \ell_k)$ is a non-empty path in h'_1 , for every $i \in [0, k-1]$ we have $\ell_i \in \text{dom}(h'_1)$. We analyse the labelled locations $\ell_{j_0}, \dots, \ell_{j_{p-1}}$, where $\ell_{j_0} = \ell$. Let $i \in [0, p-1]$. As $\ell_{j_i} \in f(\text{Lab}[s]_{s,h_1}^X)$ and $\ell_{j_i} \in \text{dom}(h'_1)$, the heap $s_{\ell_{j_i}}$ is non-empty. Following the definition of $\ell_{j_0}, \dots, \ell_{j_p}$, we conclude that, for every $i' \in [j_i + 1, j_{i+1} - 1]$, we have $\ell_{i'} \notin f(\text{Lab}[s]_{s,h_1}^X)$. From (s5), this allows us to conclude that, $\text{dom}(s_{\ell_{j_i}})$ describes a path from ℓ_{j_i} to $\ell_{j_{i+1}} = f(s_{\text{by}_{s,h_1}^X}(f^{-1}(\ell_{j_i})))$. As done in (f), by minimality of ρ and injectivity of f , this allows us to conclude that $\bigcup_{i \in [0, p-1]} \text{Path}[s]_{s,h_1}^X(\ell_{j_i})$ is a minimal set describing a non-empty path in h_1 , going from $f^{-1}(\ell)$ to $f^{-1}(\ell')$.

Proof of (sA). If t is a program variable, then the statement follows directly from (1_f). Below, we divide the proof depending on whether t is a meet-point variable or an end-point variable.

case: $t = m(x, y)$, **for some** $x, y \in X$. (\Rightarrow): Suppose $\llbracket m(x, y) \rrbracket_{s,h_1}^X = \ell$. We prove $f(\ell) = \llbracket m(x, y) \rrbracket_{s',h'_1}^X$. By definition of meet-point variables, h_1 witnesses two non-empty disjoint paths ρ and ρ' , where ρ goes from $s(x)$ to ℓ , whereas ρ' goes from

$s(y)$ to ℓ . Moreover, ℓ does not belong to a cycle. By (f), we conclude that h'_1 witnesses two non-empty paths, one going from $f(s(x)) = s'(x)$ to $f(\ell)$, and one going from $f(s(y)) = s'(y)$ to $f(\ell)$. Moreover, as ρ and ρ' are disjoint, so are these two paths of h'_1 . To conclude that $f(\ell) = \llbracket m(x, y) \rrbracket_{s', h'_1}^X$ it is sufficient to show that h'_1 does not witness a cycle that involves $f(\ell)$. *Ad absurdum*, suppose that h'_1 witnesses a non-empty path going from $f(\ell)$ to $f(\ell)$. We apply (g) and conclude that h_1 witnesses a non-empty path going from ℓ to ℓ . However, this contradicts $\llbracket m(x, y) \rrbracket_{s, h_1}^X = \ell$. Thus, h'_1 does not witness a cycle that involves $f(\ell)$. So, $f(\ell) = \llbracket m(x, y) \rrbracket_{s', h'_1}^X$.

(\Leftarrow): Suppose $\llbracket m(x, y) \rrbracket_{s', h'_1}^X = \ell$. We prove that $f^{-1}(\ell) = \llbracket m(x, y) \rrbracket_{s, h_1}^X$. Thanks to (e), $\ell \in f(\text{Lab}[s]_{s, h_1}^X)$. This allows us to prove the result analogously to the left-to-right direction. Indeed, by definition of meet-point variables, h'_1 witnesses two non-empty disjoint paths ρ and ρ' , where ρ goes from $s'(x)$ to ℓ whereas ρ' goes from $s'(y)$ to ℓ . Moreover, ℓ does not belong to a cycle. By (1_f), $f^{-1}(s'(x)) = s(x)$ and $f^{-1}(s'(y)) = s(y)$. This implies $s(x), s(y) \in f(\text{Lab}[s]_{s, h_1}^X)$, which allows us to apply (g). We conclude that h_1 witnesses two paths, one going from $s(x)$ to $f^{-1}(\ell)$ and one going from $s(y)$ to $f^{-1}(\ell)$. As ρ and ρ' are disjoint, so are these two paths of h_1 . To conclude that $f^{-1}(\ell) = \llbracket m(x, y) \rrbracket_{s, h_1}^X$, it is sufficient to show that $f^{-1}(\ell)$ does not belong to a cycle of h_1 . As $f^{-1}(\ell) \in \text{Lab}[s]_{s, h_1}^X$, this is done symmetrically to the other direction. *Ad absurdum*, suppose that h_1 witnesses a non-empty path going from $f^{-1}(\ell)$ to itself. By (f), h'_1 witnesses a non-empty path going from ℓ to itself. However, this contradicts $\llbracket m(x, y) \rrbracket_{s', h'_1}^X = \ell$. We conclude that $f^{-1}(\ell) = \llbracket m(x, y) \rrbracket_{s, h_1}^X$.

case: $t = e(x)$, for some $x \in X$. (\Rightarrow): Suppose $\llbracket e(x) \rrbracket_{s, h_1}^X = \ell$. We show that $\ell = \llbracket e(x) \rrbracket_{s', h'_1}^X$. By definition of end-point variable, h_1 witnesses a non-empty path ρ going from $s(x)$ to ℓ . Moreover, if $\ell \in \text{dom}(h_1)$, then h_1 witnesses a non-empty path ρ' that is disjoint from ρ and goes from ℓ to ℓ . By (f), h'_1 witnesses a non-empty path ρ'' going from $f(s(x)) = s'(x)$ to $f(\ell)$. To conclude that $\ell = \llbracket e(x) \rrbracket_{s', h'_1}^X$, we show that if $f(\ell) \in \text{dom}(h'_1)$, then h'_1 witnesses a non-empty path going from $f(\ell)$ to $f(\ell)$, that is disjoint from ρ'' . Suppose that $f(\ell) \in \text{dom}(h'_1)$. By definition of $s_{f(\ell)}$, $\text{Path}[s]_{s, h_1}^X(\ell) \neq \emptyset$ and $\ell \in \text{dom}(h_1)$. By semantics of end-point variables, this implies that ρ' , as defined above, is a path in h_1 . Again by (f), we conclude that h'_1 witnesses a non-empty path going from $f(\ell)$ to $f(\ell)$. As ρ and ρ' are disjoint, this path is disjoint from ρ'' , concluding the proof.

(\Leftarrow): Suppose $\llbracket e(x) \rrbracket_{s', h'_1}^X = \ell$. Thanks to (e), $\ell \in f(\text{Lab}[s]_{s, h_1}^X)$. This allows us to show that $f^{-1}(\ell) = \llbracket e(x) \rrbracket_{s, h_1}^X$ analogously to the left-to-right direction, by relying on (f) and (g).

s_B . For every $t \in T[s]^X$,

- (h) $\text{sby}_{s, h_1}^X(t)$ and $\text{sby}_{s', h'_1}^X(t)$ are equidefined. When defined, $f(\text{sby}_{s, h_1}^X(t)) = \text{sby}_{s', h'_1}^X(t)$.
- (i) $\min(\text{card}(\text{Path}[s]_{s, h_1}^X(t)), \mathcal{S}(\alpha)) = \min(\text{card}(\text{Path}[s]_{s', h'_1}^X(t)), \mathcal{S}(\alpha))$.
- (j) If $h_1^{\delta_1}(\llbracket t \rrbracket_{s, h_1}^X) = s(u)$ for some $\delta_1 \in [0, \text{card}(\text{Path}[s]_{s, h_1}^X(t))]$, then $h'_1{}^{\delta_2}(\llbracket t \rrbracket_{s', h'_1}^X) = s'(u)$ for some $\delta_2 \in [0, \text{card}(\text{Path}[s]_{s', h'_1}^X(t))]$ s.t. $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j))$ and $\min(\text{card}(\text{Path}[s]_{s, h_1}^X(t)) - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(\text{Path}[s]_{s', h'_1}^X(t)) - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j))$.

- (k) If $h'_1(\llbracket t \rrbracket_{s', h'_1}^X) = s'(u)$ for some $\delta_2 \in [0, \text{card}(\text{Path}[s]_{s', h'_1}^X(t))]$, then $h_j^{\delta_1}(\llbracket t \rrbracket_{s, h_1}^X) = s(u)$ for some $\delta_1 \in [0, \text{card}(\text{Path}[s]_{s, h_1}^X(t))]$ s.t. $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha_j)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha_j))$ and $\min(\text{card}(\text{Path}[s]_{s, h_1}^X(t)) - \delta_1, \mathcal{S}_{\text{right}}(\alpha_j)) = \min(\text{card}(\text{Path}[s]_{s', h'_1}^X(t)) - \delta_2, \mathcal{S}_{\text{right}}(\alpha_j))$.

These four statements essentially rely on the intermediate result below:

- (l) If $\llbracket t \rrbracket_{s, h_1}^X$ is defined, then $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)}) = \text{Path}[s]_{s', h'_1}^X(\llbracket t \rrbracket_{s', h'_1}^X)$.

Proof of (l). By (s_A), $f(\llbracket t \rrbracket_{s, h_1}^X) = \llbracket t \rrbracket_{s', h'_1}^X$. If $\llbracket t \rrbracket_{s, h_1}^X \notin \text{dom}(h_1)$ then $\text{Path}[s]_{s, h_1}^X(t) = \emptyset$, which implies $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)}) = \emptyset$, by (s₉). This implies $\llbracket t \rrbracket_{s', h'_1}^X \notin \text{dom}(h'_1)$ and so $\text{Path}[s]_{s', h'_1}^X(\llbracket t \rrbracket_{s', h'_1}^X) = \emptyset$. Otherwise ($\llbracket t \rrbracket_{s, h_1}^X \in \text{dom}(h_1)$), from (s₅), $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)})$ is a minimal set describing a non-empty path $\rho = (\ell_0, \dots, \ell_n)$, in both $s_{f(\llbracket t \rrbracket_{s, h_1}^X)}$ and h'_1 , from $\llbracket t \rrbracket_{s', h'_1}^X$ to $f(sby_{s, h_1}^X(t))$. Notice that $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)}) = \{\ell_0, \dots, \ell_{n-1}\}$. Again by (s_A), $f(sby_{s, h_1}^X(t)) \in \text{Lab}[s]_{s', h'_1}^X$. From (d), the locations $\ell_1, \dots, \ell_{n-1}$ are not in $\text{Lab}[s]_{s', h'_1}^X$. This means that $\ell_n = f(sby_{s, h_1}^X(t))$ is the first labelled location reached, in h'_1 , from $\llbracket t \rrbracket_{s', h'_1}^X$. According to Definition 5.27, $\ell_n = sby_{s', h'_1}^X(t)$. According to Definition 5.29, $\text{Path}[s]_{s', h'_1}^X(t) = \{\ell_0, \dots, \ell_{n-1}\} = \text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)})$.

Proof of (h). (\Rightarrow): If $sby_{s, h_1}^X(t)$ is defined, then so is $\llbracket t \rrbracket_{s, h_1}^X$, and $\text{Path}[s]_{s, h_1}^X(t) \neq \emptyset$. By (l), $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)}) = \text{Path}[s]_{s', h'_1}^X(\llbracket t \rrbracket_{s', h'_1}^X)$. From $\text{Path}[s]_{s, h_1}^X(t) \neq \emptyset$ and (s₉), we conclude that $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)}) \neq \emptyset$ and thus, by (s₅), it is a minimal set describing a path in h'_1 , going from $f(\llbracket t \rrbracket_{s', h'_1}^X) = \llbracket t \rrbracket_{s', h'_1}^X$ (by (s_A)) to $f(sby_{s, h_1}^X(t))$. By definition, $\text{Path}[s]_{s', h'_1}^X(\llbracket t \rrbracket_{s', h'_1}^X)$ is a minimal set describing a non-empty path going from $\llbracket t \rrbracket_{s', h'_1}^X$ to $sby_{s', h'_1}^X(t)$. The two paths coincide. So, $f(sby_{s, h_1}^X(t)) = sby_{s', h'_1}^X(t)$.

(\Leftarrow): If $sby_{s', h'_1}^X(t)$ is defined, then so is $\llbracket t \rrbracket_{s', h'_1}^X$, and $\llbracket t \rrbracket_{s', h'_1}^X \in \text{dom}(h'_1)$. By (s_A), so is $f^{-1}(\llbracket t \rrbracket_{s', h'_1}^X) = \llbracket t \rrbracket_{s, h_1}^X$. By (l), $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)}) = \text{Path}[s]_{s', h'_1}^X(\llbracket t \rrbracket_{s', h'_1}^X)$. From $\llbracket t \rrbracket_{s', h'_1}^X \in \text{dom}(h'_1)$, we conclude that $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)})$ is non-empty. Afterwards, the proof that $f(sby_{s, h_1}^X(t)) = sby_{s', h'_1}^X(t)$ carries out as in the left-to-right direction.

Proof of (i). As f is injective, the sets $\text{Path}[s]_{s, h_1}^X(t) \cap \text{dom}(f)$ and $\{\ell \in \text{ran}(f) \mid f^{-1}(\ell) \in \text{Path}[s]_{s, h_1}^X(t)\}$ have the same cardinality. Together with (s₁), this implies that

$$\begin{aligned} & \min(\text{card}(S_{\llbracket t \rrbracket_{s, h_1}^X}) + \text{card}(\text{Path}[s]_{s, h_1}^X(t) \cap \text{dom}(f)), \mathcal{S}(\alpha)) \\ &= \min(\text{card}(S'_{f(\llbracket t \rrbracket_{s, h_1}^X)}) + \text{card}(\{\ell \in \text{ran}(f) \mid f^{-1}(\ell) \in \text{Path}[s]_{s, h_1}^X(t)\}), \mathcal{S}(\alpha)). \end{aligned}$$

On the left hand side of this equivalence, $\text{card}(S_{\llbracket t \rrbracket_{s, h_1}^X}) + \text{card}(\text{Path}[s]_{s, h_1}^X(t) \cap \text{dom}(f))$ is equivalent to $\text{card}(\text{Path}[s]_{s, h_1}^X(t))$. Indeed, from the definition of $S_{\llbracket t \rrbracket_{s, h_1}^X}$, the set $\text{Path}[s]_{s, h_1}^X(t)$ is the union of the two disjoint sets $S_{\llbracket t \rrbracket_{s, h_1}^X}$ and $\text{Path}[s]_{s, h_1}^X(t) \cap \text{dom}(f)$. On the right hand side, directly from (s₄) and (D), we have

$$\text{card}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)}) = \text{card}(S'_{f(\llbracket t \rrbracket_{s, h_1}^X)}) + \text{card}(\{\ell \in \text{ran}(f) \mid f^{-1}(\ell) \in \text{Path}[s]_{s, h_1}^X(t)\}).$$

From (s_A), $f(\llbracket t \rrbracket_{s, h_1}^X) = \llbracket t \rrbracket_{s', h'_1}^X$. Afterwards, by (l), we conclude:

$$\min(\text{card}(\text{Path}[s]_{s, h_1}^X(t)), \mathcal{S}(\alpha)) = \min(\text{card}(\text{Path}[s]_{s', h'_1}^X(t)), \mathcal{S}(\alpha)).$$

Proof of (j). As $\llbracket t \rrbracket_{s, h_1}^X$ is defined, by (l), $\text{dom}(s_{f(\llbracket t \rrbracket_{s, h_1}^X)}) = \text{Path}[s]_{s', h'_1}^X(t)$. Then, the result holds directly from (s₈).

Proof of (k). First of all, if $s'(u) = \llbracket t \rrbracket_{s', h'_1}^X$ or $s'(u) = \text{sby}_{s', h'_1}^X(t)$, then by (s_A) and (0_f), we conclude that $s(u) = \llbracket t \rrbracket_{s, h_1}^X$ or $s(u) = \text{sby}_{s, h_1}^X(t)$, respectively, which ends the proof. Otherwise, suppose there is $\delta_2 \in [1, \text{card}(\text{Path}[s]_{s', h'_1}^X(t)) - 1]$ such that $h'_1{}^{\delta_2}(\llbracket t \rrbracket_{s', h'_1}^X) = s'(u)$. Since $\text{Path}[s]_{s', h'_1}^X(t)$ is a minimal set describing the path in h'_1 , going from $\llbracket t \rrbracket_{s', h'_1}^X$ to $\text{sby}_{s', h'_1}^X(t)$, δ_2 is unique. Therefore, in order to conclude the proof, it is sufficient to show that $s(u) \in \text{Path}[s]_{s, h_1}^X(\ell)$, and apply (j). From (l), (s₄) and (s₅), we conclude that either $s'(u) \in S'_{f(\llbracket t \rrbracket_{s, h_1}^X)}$ or $s'(u) \in \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Path}[s]_{s, h_1}^X(t)\}$. In the former case, by (s₂) $s(u) \in S_{\llbracket t \rrbracket_{s, h_1}^X} \subseteq \text{Path}[s]_{s, h_1}^X(t)$. Similarly, in the latter case, (0_f) allows us to conclude that $s(u) \in \text{Path}[s]_{s, h_1}^X(t)$.

s_C . For every $x \in X$,

- (m) $\min(\text{card}(\text{Pred}[s]_{s, h_1}^X(x)), \mathcal{P}(\alpha)) = \min(\text{card}(\text{Pred}[s]_{s', h'_1}^X(x)), \mathcal{P}(\alpha))$,
- (n) $s(u) \in \text{Pred}[s]_{s, h_1}^X(x)$ if and only if $s'(u) \in \text{Pred}[s]_{s', h'_1}^X(x)$.

Essentially, both statements rely on the following result:

- (o) $\text{dom}(P_{f(s(x))}) = \text{Pred}[s]_{s', h'_1}^X(x)$.

Proof of (o). From (1_f), $f(s(x)) = s'(x)$.

(\subseteq): Let $\ell \in \text{dom}(P_{f(s(x))})$. By (p₅), we have $h'_1(\ell) = s'(x)$. Thanks to (c), this implies $\ell \in \text{Pred}[s]_{s', h'_1}^X(x)$, directly by definition of $\text{Pred}[s]_{s', h'_1}^X(x)$.

(\supseteq): Suppose $\ell \in \text{Pred}[s]_{s', h'_1}^X(x)$. Thus, $h'_1(\ell) = s'(x)$. This implies $\ell \in \text{dom}(h'_1)$. We prove that $\ell \in \text{dom}(P_{f(s(x))})$ by showing that ℓ cannot belong to the domain of the other heaps used to define h'_1 . By Proposition 5.31, $\ell \notin \text{Path}[s]_{s', h'_1}^X(f(\ell))$, for any $\ell \in \text{Lab}[s]_{s, h_1}^X$. From (l), ℓ does not belong to $\text{dom}(s_{f(\ell_1)} + \dots + s_{f(\ell_m)})$. Since $h'_1(\ell) \in s'(X)$, by (b), $\ell \notin \text{dom}(C_1 + \dots + C_\alpha + U + R)$. By definition of h'_1 , we conclude that there is $\ell' \in s(X)$ such that $\ell \in \text{dom}(P_{f(\ell')})$. Since $f(s(x)) = s'(x)$ and $h'_1(\ell) = s'(x)$, by definition of $P'_{f(s(x))}$, we derive $\ell' = s(x)$. So, $\ell \in \text{dom}(P_{f(s(x))})$.

Proof of (m). As f is injective, we have

$$\text{card}(\text{Pred}[s]_{s, h_1}^X(x) \cap \text{dom}(f)) = \text{card}(\{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Pred}[s]_{s, h_1}^X(x)\}).$$

By (p₁), $\min(\text{card}(P_{s(x)}), \mathcal{P}(\alpha)) = \min(\text{card}(P'_{f(s(x))}), \mathcal{P}(\alpha))$. This implies that

$$\begin{aligned} & \min(\text{card}(P_{s(x)}) + \text{card}(\text{Pred}[s]_{s, h_1}^X(x) \cap \text{dom}(f)), \mathcal{P}(\alpha)) \\ &= \min(\text{card}(P'_{f(s(x))}) + \text{card}(\{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Pred}[s]_{s, h_1}^X(x)\}), \mathcal{P}(\alpha)). \end{aligned}$$

On the left hand side of this equality, $\text{card}(P_{s(x)}) + \text{card}(\text{Pred}[s]_{s, h_1}^X(x) \cap \text{dom}(f))$ is equivalent to $\text{card}(\text{Pred}[s]_{s, h_1}^X(x))$. Indeed, by definition of $P_{s(x)}$, the set $\text{Pred}[s]_{s, h_1}^X(x)$ is the union of the two disjoint sets $P_{s(x)}$ and $\text{Pred}[s]_{s, h_1}^X(x) \cap \text{dom}(f)$. On the right hand side, directly from (p₄) and (D), we have

$$\text{card}(P_{f(s(x))}) = \text{card}(P'_{f(s(x))}) + \text{card}(\{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Pred}[s]_{s, h_1}^X(x)\}).$$

From (o), we conclude:

$$\min(\text{card}(\text{Pred}[s]_{s, h_1}^X(x)), \mathcal{P}(\alpha)) = \min(\text{card}(\text{Pred}[s]_{s', h'_1}^X(x)), \mathcal{P}(\alpha)).$$

Proof of (n). As discussed during the proof of (m),

$$\text{Pred}[s]_{s, h_1}^X(x) = P_{s(x)} \cup \text{Pred}[s]_{s, h_1}^X(x) \cap \text{dom}(f),$$

$$\text{Pred}[s]_{s', h'_1}^X(x) = P'_{f(s(x))} \cup \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Pred}[s]_{s, h_1}^X(x)\}.$$

(\Rightarrow): Suppose $s(u) \in \text{Pred}[s]_{s,h_1}^X(x)$. By (p2), if $s(u) \in P_{s(x)}$, then $s'(u) \in P'_{f(s(x))}$. From (0f), if $s(u) \in \text{Pred}[s]_{s,h_1}^X(x) \cap \text{dom}(f)$ then we derive $s'(u) \in \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Pred}[s]_{s,h_1}^X(x)\}$. In both cases, we conclude that $s'(u) \in \text{Pred}[s]_{s',h'_1}^X(x)$.

(\Leftarrow): Symmetrical to the other direction.

s_D . For every $\beta \in [1, \alpha]$,

- (p) $\min(\text{card}(\text{Cycl}[s]_{s,h_1}^X(\beta)), \mathcal{L}(\alpha)) = \min(\text{card}(\text{Cycl}[s]_{s',h'_1}^X(\beta)), \mathcal{L}(\alpha))$,
- (q) $s(u) \in [\text{Cycl}[s]_{s,h_1}^X(\beta)]^\flat$ if and only if $s'(u) \in [\text{Cycl}[s]_{s',h'_1}^X(\beta)]^\flat$.

Both statements rely on the following result

$$(r) C'_\beta \cup \overline{C}'_\beta = \text{Cycl}[s]_{s',h'_1}^X(\beta).$$

Proof of (r). (\subseteq): Let $L \in C_\beta \cup \overline{C}_\beta$. By definition $\text{card}(L) = \beta$. By (c10), L describes a cycle in C_β , of length β . By $C_\beta \subseteq h'_1$, L describes a cycle in h'_1 . As $L \subseteq \text{dom}(C_\beta)$, by (b) we conclude that L does not contain labelled locations in $\text{Lab}[s]_{s',h'_1}^X$. Indeed, from the semantics of $[\cdot]_{s',h'_1}^X$, labelled locations are either assigned to program variables or reached by one. We conclude that $L \in \text{Cycl}[s]_{s',h'_1}^X(\beta)$.

(\supseteq): Suppose $L \in \text{Cycl}[s]_{s',h'_1}^X(\beta)$. So, L is a set of cardinality β that describes a cycle in h'_1 , of length β . Moreover, $L \cap \text{Lab}[s]_{s',h'_1}^X = \emptyset$. We show that $L \subseteq \text{dom}(C_\beta)$, thus concluding the proof by (c11) and definition of C_β . Let \hat{h} be a heap among

$$C_1, \dots, C_{\beta-1}, C_{\beta+1}, \dots, C_\alpha, U.$$

By definition, \hat{h} only contains cycles, but they are of lengths different from β . Therefore, $L \cap \text{dom}(\hat{h}) = \emptyset$. Since $L \in \text{Cycl}[s]_{s',h'_1}^X(\beta)$, For all $x \in X$, $L \cap \text{Pred}[s]_{s',h'_1}^X(x) = \emptyset$, which in turn implies $L \cap \text{dom}(P_{s'(x)}) = \emptyset$, directly by (o). Similarly, by (l), for every $\ell \in \text{Lab}[s]_{s,h_1}^X$, $L \cap \text{dom}(S_{f(\ell)}) = \emptyset$. Lastly, $L \cap \text{dom}(R) = \emptyset$, as $\text{ran}(R) \subseteq \{\ell_r\}$ and $\ell_r \notin \text{dom}(h'_1)$, by (r5) and (p1). We have shown $L \cap \text{dom}(\hat{h}) = \emptyset$, for all \hat{h} among

$$C_1, \dots, C_{\beta-1}, C_{\beta+1}, \dots, C_\alpha, U, R, S_{f(\ell_1^L)}, \dots, S_{f(\ell_m^L)}.$$

By definition of h'_1 , we conclude that $L \subseteq \text{dom}(C_\beta)$.

Proof of (p). From (c5), $\text{card}(\overline{C}_\beta) = \text{card}(\overline{C}'_\beta)$. Together with (c1), we conclude

$$\min(\text{card}(C_\beta) + \text{card}(\overline{C}_\beta), \mathcal{L}(\alpha)) = \min(\text{card}(C'_\beta) + \text{card}(\overline{C}'_\beta), \mathcal{L}(\alpha)).$$

On the left hand side of this equality, by definition of C_β and \overline{C}_β , we derive $\text{card}(\text{Cycl}[s]_{s,h_1}^X(\beta)) = \text{card}(C_\beta) + \text{card}(\overline{C}_\beta)$. Similarly, on the right hand side, by (r) and the fact that C'_β and \overline{C}'_β are disjoint, $\text{card}(\text{Cycl}[s]_{s',h'_1}^X(\beta)) = \text{card}(C'_\beta) + \text{card}(\overline{C}'_\beta)$. This allows us to conclude the proof.

Proof of (q). By definition of C_β and \overline{C}_β , $\text{Cycl}[s]_{s,h_1}^X(\beta) = C_\beta \cup \overline{C}_\beta$. Then, the result holds directly from (r), together with (c2) and (c6).

- (s) $\min(\text{card}(\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}), \mathcal{L}(\alpha)) = \min(\text{card}(\uparrow\text{Cycl}[s]_{s',h'_1}^{X,\alpha}), \mathcal{L}(\alpha))$,
- (t) $s(u) \in [\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}]^\flat$ if and only if $s'(u) \in [\uparrow\text{Cycl}[s]_{s',h'_1}^{X,\alpha}]^\flat$.

Both statements rely on the following result

$$(u) U' \cup \overline{U}' = \uparrow\text{Cycl}[s]_{s',h'_1}^{X,\alpha}.$$

Proof of (s)–(u). Analogous to (p)–(r).

- (v) $\min(\text{card}(\text{Rem}[s]_{s,h_1}^{X,\alpha}), \mathcal{R}(\alpha)) = \min(\text{card}(\text{Rem}[s]_{s',h'_1}^{X,\alpha}), \mathcal{R}(\alpha))$,

(w) $s(u) \in \text{Rem}[S]_{s,h_1}^{X,\alpha}$ if and only if $s'(u) \in \text{Rem}[S]_{s',h'_1}^{X,\alpha}$.

Both statements rely on the following result

(x) $\text{dom}(R) = \text{Rem}[S]_{s',h'_1}^{X,\alpha}$.

Proof of (x). (\subseteq): Let $\ell \in \text{dom}(R)$. Given $\ell' \in \text{Lab}[S]_{s',h'_1}^X$, by (1), $\ell \notin \text{Path}[S]_{s',h'_1}^X(\ell')$.

Given $x \in X$, from (o), $\ell \notin \text{Pred}[S]_{s',h'_1}^X(\ell')$. Given $\beta \in [1, \alpha]$, by (r) and (c11), $\ell \notin \text{Cycl}[S]_{s',h'_1}^X(\beta)$. Lastly, (u) and (u11) imply $\ell \notin \text{↑Cycl}[S]_{s',h'_1}^{X,\alpha}$. As $\ell \in \text{dom}(R) \subseteq \text{dom}(h'_1)$, by definition of $\text{Rem}[S]_{s',h'_1}^{X,\alpha}$, we conclude that $\ell \in \text{Rem}[S]_{s',h'_1}^{X,\alpha}$.

(\supseteq): Symmetrical to the other direction.

Proof of (v). As f is injective, we have

$$\text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha} \cap \text{dom}(f)) = \text{card}(\{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Rem}[S]_{s,h_1}^{X,\alpha}\}).$$

By (r1), $\min(\text{card}(R), \mathcal{R}(\alpha)) = \min(\text{card}(R'), \mathcal{R}(\alpha))$. This implies that

$$\begin{aligned} & \min(\text{card}(R) + \text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha} \cap \text{dom}(f)), \mathcal{R}(\alpha)) \\ &= \min(\text{card}(R') + \text{card}(\{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Rem}[S]_{s,h_1}^{X,\alpha}\}), \mathcal{R}(\alpha)). \end{aligned}$$

On the left hand side of this equivalence, $\text{card}(R) + \text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha} \cap \text{dom}(f))$ is equivalent to $\text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha})$, directly by definition of R . On the right hand side, directly from (r4) and (D), $\text{card}(R) = \text{card}(R') + \text{card}(\{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Rem}[S]_{s,h_1}^{X,\alpha}\})$. From (x), we conclude:

$$\min(\text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha}), \mathcal{R}(\alpha)) = \min(\text{card}(\text{Rem}[S]_{s',h'_1}^{X,\alpha}), \mathcal{R}(\alpha)).$$

Proof of (w). As discussed during the proof of (v),

$$\text{Rem}[S]_{s,h_1}^{X,\alpha} = R \cup \text{Rem}[S]_{s,h_1}^{X,\alpha} \cap \text{dom}(f),$$

$$\text{Rem}[S]_{s',h'_1}^{X,\alpha} = R' \cup \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Rem}[S]_{s,h_1}^{X,\alpha}\}.$$

(\Rightarrow): Suppose $s(u) \in \text{Rem}[S]_{s,h_1}^{X,\alpha}$. By (r2), if $s(u) \in R$, then $s'(u) \in R'$. By (0f), if $s(u) \in \text{Rem}[S]_{s,h_1}^{X,x} \cap \text{dom}(f)$ then $s'(u) \in \{\ell' \in \text{ran}(f) \mid f^{-1}(\ell') \in \text{Rem}[S]_{s,h_1}^{X,\alpha}\}$. In both cases, we conclude that $s'(u) \in \text{Rem}[S]_{s',h'_1}^{X,\alpha}$.

(\Leftarrow): Symmetrical to the other direction.

Proof of $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$. Let us consider a core formula φ in $\text{Core}[S](X, \alpha_j)$. We have $(s, h_j) \models \varphi$ iff $(s', h'_j) \models \varphi$, as discussed below:

case: $\varphi = t_1 = t_2$. Follows directly from (sA).

case: $\varphi = \text{sees}_X(t_1, t_2) \geq \beta$. Follows from (sA), (sB)((h) and (i)).

case: $\varphi = \text{pred}_X^S(x) \geq \beta$. Follows directly from (m).

case: $\varphi = \text{loop}_X^S(\beta_1) \geq \beta_2$. Follows directly from (p).

case: $\varphi = \text{↑loop}_{X,\alpha}^S \geq \beta$. Follows directly from (s).

case: $\varphi = \text{rem}_{X,\alpha}^S \geq \beta$. Follows directly from (v).

case: $\varphi = u = t$. Follows directly from (sA) and (0f).

case: $\varphi = u \in \text{sees}_X(t_1, t_2) \geq (\beta_1, \beta_2)$. Follows directly from (j) and (k).

case: $\varphi = u \in \text{pred}_X^S(x)$. Follows directly from (n).

case: $\varphi = u \in \text{loop}_X^S(\beta)$. Follows directly from (q).

case: $\varphi = u \in \text{↑loop}_{X,\alpha}^S$. Follows directly from (t).

case: $\varphi = u \in \text{rem}_{X,\alpha}^S$. Follows directly from (w).

Therefore, $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$.

We remind the reader that, by Corollary 5.36, $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$ implies $(s, h_1) \approx_{X,\alpha}^W (s', h'_1)$.

Towards $(s, h + h_1) \approx_{X,\alpha}^W (s', h' + h'_1)$. Below, we write \vec{h} as a shortcut for $h + h_1$. Similarly, \vec{h}' is short for $h' + h'_1$. Let $x \in X$. As a first step towards $(s, \vec{h}) \approx_{X,\alpha}^W (s', \vec{h}')$, we want to understand the relationships between $\llbracket n(x) \rrbracket_{s,\vec{h}}^X$, and $\llbracket n(x) \rrbracket_{s',\vec{h}'}^X$. We recall that $\llbracket n(x) \rrbracket_{s,h}^X$ is defined if and only if $s(x) \in \text{dom}(h)$. Whenever it is defined, $\llbracket n(x) \rrbracket_{s,h}^X = h(s(x))$. Therefore,

- from $\vec{h} = h + h_1$, $\llbracket n(x) \rrbracket_{s,\vec{h}}^X$ is defined if and only if so is one among $\llbracket n(x) \rrbracket_{s,h}^X$ or $\llbracket n(x) \rrbracket_{s,h_1}^X$,
- from $h \perp h_1$, $\llbracket n(x) \rrbracket_{s,h}^X$ and $\llbracket n(x) \rrbracket_{s,h_1}^X$ cannot be both defined.

We relate $\llbracket n(x) \rrbracket_{s,\vec{h}}^X$ and $\llbracket n(x) \rrbracket_{s',\vec{h}'}^X$, following the analysis below:

case: $\llbracket n(x) \rrbracket_{s,h}^X$ is defined. By $\llbracket n(x) \rrbracket_{s,h}^X \in \text{Lab}[W]_{s,h}^X$ and (1_f), we derive $f(\llbracket n(x) \rrbracket_{s,h}^X) = \llbracket n(x) \rrbracket_{s',h'}^X$.

Directly from Lemma 5.14(I), $\llbracket n(x) \rrbracket_{s,h}^X = \llbracket n(x) \rrbracket_{s,\vec{h}}^X$ and $\llbracket n(x) \rrbracket_{s',h'}^X = \llbracket n(x) \rrbracket_{s',\vec{h}'}^X$. Therefore:

(π₁) If $\llbracket n(x) \rrbracket_{s,h}^X$ (equivalently, $\llbracket n(x) \rrbracket_{s',h'}^X$) is defined, then $f(\llbracket n(x) \rrbracket_{s,\vec{h}}^X) = \llbracket n(x) \rrbracket_{s',\vec{h}'}^X$.

case: $\llbracket n(x) \rrbracket_{s,h_1}^X$ is defined. Since $s(x) \in \text{dom}(h_1)$, in this case we have $\text{Path}[S]_{s,h_1}^X(x) \neq \emptyset$. We split the analysis in the following three cases:

case: $\text{card}(\text{Path}[S]_{s,h_1}^X(x)) = 1$. From (i), the assumption $\text{card}(\text{Path}[S]_{s,h_1}^X(x)) = 1$ is equivalent to $\text{card}(\text{Path}[S]_{s',h'_1}^X(x)) = 1$. By definition of $\text{Path}[S]_{s,h}^X(x)$ and $\text{Path}[S]_{s',h'_1}^X(x)$, we have $h_1(s(x)) = \llbracket n(x) \rrbracket_{s,h_1}^X = \text{sby}_{s,h_1}^X(x)$ and $h'_1(s'(x)) = \llbracket n(x) \rrbracket_{s',h'_1}^X = \text{sby}_{s',h'_1}^X(x)$. From (h), $f(\text{sby}_{s,h_1}^X(x)) = \text{sby}_{s',h'_1}^X(x)$. Thanks to Lemma 5.14(I), we obtain:

(π₂) If $\text{card}(\text{Path}[S]_{s,h_1}^X(x)) = 1$ (or $\text{card}(\text{Path}[S]_{s',h'_1}^X(x)) = 1$), then $f(\llbracket n(x) \rrbracket_{s,\vec{h}}^X) = \llbracket n(x) \rrbracket_{s',\vec{h}'}^X$.

case: $\text{card}(\text{Path}[S]_{s,h_1}^X(x)) \geq 2$ and $\llbracket n(y) \rrbracket_{s,h}^X = \llbracket n(x) \rrbracket_{s,h_1}^X$, for some $y \in X$. In this case, we apply (π₁) in order to conclude that $f(\llbracket n(x) \rrbracket_{s,h_1}^X) = \llbracket n(y) \rrbracket_{s',h'}^X$. Thanks to the property (s₆), we derive $\llbracket n(x) \rrbracket_{s',h'_1}^X = \llbracket n(y) \rrbracket_{s',h'}^X$. Notice that, by (i) and (s₆), the assumption

$\text{card}(\text{Path}[S]_{s,h_1}^X(x)) \geq 2$ and there is $y \in X$ such that $\llbracket n(y) \rrbracket_{s,h}^X = \llbracket n(x) \rrbracket_{s,h_1}^X$,

is equivalent to

$\text{card}(\text{Path}[S]_{s',h'_1}^X(x)) \geq 2$ and there is $y \in X$ such that $\llbracket n(y) \rrbracket_{s',h'}^X = \llbracket n(x) \rrbracket_{s',h'_1}^X$.

From $f(\llbracket n(x) \rrbracket_{s,h_1}^X) = \llbracket n(y) \rrbracket_{s',h'}^X$ and by Lemma 5.14(I),

(π₃) If $\text{card}(\text{Path}[S]_{s,h_1}^X(x)) \geq 2$ and there is $y \in X$ such that $\llbracket n(y) \rrbracket_{s,h}^X = \llbracket n(x) \rrbracket_{s,h_1}^X$ (equiv. $\text{card}(\text{Path}[S]_{s',h'_1}^X(x)) \geq 2$ and there is $y \in X$ such that $\llbracket n(y) \rrbracket_{s',h'}^X = \llbracket n(x) \rrbracket_{s',h'_1}^X$), then $f(\llbracket n(x) \rrbracket_{s,\vec{h}}^X) = \llbracket n(x) \rrbracket_{s',\vec{h}'}^X$.

case: $\text{card}(\text{Path}[S]_{s,h_1}^X(x)) \geq 2$ and for all $y \in X$, $\llbracket n(y) \rrbracket_{s,h}^X \neq \llbracket n(x) \rrbracket_{s,h_1}^X$. Similarly to the previous case, by (i) and (s₆), this assumption is equivalent to

$\text{card}(\text{Path}[S]_{s',h'_1}^X(x)) \geq 2$ and for every $y \in X$, $\llbracket n(y) \rrbracket_{s',h'}^X \neq \llbracket n(x) \rrbracket_{s',h'_1}^X$.

Differently from the other cases, we cannot rely on f . Indeed, by $\text{card}(\text{Path}[S]_{s,h_1}^X(x)) \geq 2$ we derive $\llbracket n(x) \rrbracket_{s,h_1}^X \in \text{Path}[S]_{s,h_1}^X(x) \setminus \{s(x)\}$. Since $s(x)$ is the only location in $\text{Path}[S]_{s,h_1}^X(x)$ that belongs to $\text{Lab}[S]_{s,h_1}^X$, this implies $\llbracket n(x) \rrbracket_{s,h_1}^X \notin \text{Lab}[S]_{s,h_1}^X$. Since for every $y \in X$, $\llbracket n(y) \rrbracket_{s,h}^X \neq \llbracket n(x) \rrbracket_{s,h_1}^X$, we conclude that $\llbracket n(x) \rrbracket_{s,h_1}^X \notin \text{Lab}[W]_{s,h}^X$ (recall that locations assigned to program variables belong to both $\text{Lab}[S]_{s,h_1}^X$ and $\text{Lab}[W]_{s,h}^X$). By definition of f , $\llbracket n(x) \rrbracket_{s,h_1}^X \notin \text{dom}(f)$. This implies that $\llbracket n(x) \rrbracket_{s,h_1}^X \in S_{s(x)}$ and, by (s₆), $\llbracket n(x) \rrbracket_{s',h'_1}^X \in S'_{s'(x)}$. By Lemma 5.14(I),

(π₄) Whenever $\text{card}(\text{Path}[S]_{s,h_1}^X(x)) \geq 2$ and $\llbracket n(y) \rrbracket_{s,h}^X \neq \llbracket n(x) \rrbracket_{s,h_1}^X$ holds for every $y \in X$, (equiv. $\text{card}(\text{Path}[S]_{s',h'_1}^X(x)) \geq 2$ and for every $y \in X$, $\llbracket n(y) \rrbracket_{s',h'}^X \neq \llbracket n(x) \rrbracket_{s',h'_1}^X$), then $\llbracket n(x) \rrbracket_{s,\vec{h}}^X \in S_{s(x)}$ and $\llbracket n(x) \rrbracket_{s',\vec{h}'}^X \in S'_{s'(x)}$.

Let $x, y \in X$, $\beta \in [1, \alpha]$ and $t \in T[S]^X$.

- (σ_1) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Pred}[\mathcal{W}]_{s, h}^X(y)$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Pred}[\mathcal{W}]_{s', h'}^X(y)$,
- (σ_2) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Pred}[\mathcal{S}]_{s, h_1}^X(y)$ if and only if $\llbracket n(x) \rrbracket_{s', h'}^X \in \text{Pred}[\mathcal{S}]_{s', \vec{h}'}^X(y)$,
- (σ_3) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Self}[\mathcal{W}]_{s, h}^X$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Self}[\mathcal{W}]_{s', h'}^X$,
- (σ_4) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in [\text{Cycl}[\mathcal{S}]_{s, h_1}^X(\beta)]^\flat$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in [\text{Cycl}[\mathcal{S}]_{s', h'_1}^X(\beta)]^\flat$,
- (σ_5) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in [\uparrow\text{Cycl}[\mathcal{S}]_{s, h_1}^{X, \alpha}]^\flat$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in [\uparrow\text{Cycl}[\mathcal{S}]_{s', h'_1}^{X, \alpha}]^\flat$,
- (σ_6) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Rem}[\mathcal{W}]_{s, h}^X$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Rem}[\mathcal{W}]_{s', h'}^X$,
- (σ_7) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Rem}[\mathcal{S}]_{s, h_1}^{X, \alpha}$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Rem}[\mathcal{S}]_{s', h'_1}^{X, \alpha}$,

Suppose that $\text{Path}[\mathcal{S}]_{s, h_1}^X(t)$ describes the path $\rho = (\ell_0, \dots, \ell_p)$ in h_1 , from $\llbracket t \rrbracket_{s, h_1}^X$ to $\text{sby}_{s, h_1}^X(t)$. Suppose that $\text{Path}[\mathcal{S}]_{s', h'_1}^X(t)$ describes the path $\rho' = (\ell'_0, \dots, \ell'_q)$ in h'_1 , from $\llbracket t \rrbracket_{s', h'_1}^X$ to $\text{sby}_{s', h'_1}^X(t)$.

We have:

- (σ_8) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Path}[\mathcal{S}]_{s, h_1}^X(t)$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Path}[\mathcal{S}]_{s', h'_1}^X(t)$
- (σ_9) Let $i \in \{0, 1\}$. $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \ell_i$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \ell'_i$,
- (σ_{10}) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \ell_{p-1}$ if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \ell'_{q-1}$,

Figure 5.23: Connecting next-point variables.

case: $\llbracket n(x) \rrbracket_{s, h}^X$ and $\llbracket n(x) \rrbracket_{s, h_1}^X$ are not defined. In this case, $\llbracket n(x) \rrbracket_{s, \vec{h}}^X$ is not defined. From $(s, h) \approx_{X, \alpha + \text{card}(X)}^{\mathcal{W}} (s', h')$ and $(s, h_1) \approx_{X, \alpha}^{\mathcal{W}} (s', h'_1)$, we have:

- $(s, h) \models n(x) = n(x)$ if and only if $(s', h') \models n(x) = n(x)$.
- $(s, h_1) \models n(x) = n(x)$ if and only if $(s', h'_1) \models n(x) = n(x)$.

From the semantics of $n(x) = n(x)$, $\llbracket n(x) \rrbracket_{s', h'}^X$ and $\llbracket n(x) \rrbracket_{s', h'_1}^X$ are not defined. Thus, $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ is not defined. Symmetrically, if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ is not defined then so is $\llbracket n(x) \rrbracket_{s, \vec{h}}^X$.

- (π_5) $\llbracket n(x) \rrbracket_{s, \vec{h}}^X$ is defined if and only if $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ is defined.

The analysis above is exhaustive: whenever $\llbracket n(x) \rrbracket_{s, \vec{h}}^X$ is defined (equivalently, $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ is defined, by (π_5)), then the premises of exactly one statement among (π_1)–(π_4) are verified. We can summarise (π_1)–(π_5) as follows:

- (π_6) Whenever $\llbracket n(x) \rrbracket_{s, \vec{h}}^X$ or $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ are defined, we have either $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ or $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in S_{s(x)}$ and $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in S'_{s'(x)}$.

All these properties are essential and, together with the definitions of the heaps given in Figure 5.21, (π_1)–(π_5), allow us to derive the properties (σ_1)–(σ_{10}) described in Figure 5.23. Let us discuss the validity of (σ_1)–(σ_{10}). The proofs of (σ_1), (σ_3) (σ_6) are very similar, and so are the proofs of (σ_2), (σ_4), (σ_5) and (σ_7). Below, we show the proofs of (σ_1), (σ_4) and (σ_8)–(σ_{10}).

Proof of (σ_1). (\Rightarrow): Suppose $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Pred}[\mathcal{W}]_{s, h}^X(y)$. Since $S_{s(x)} \subseteq \text{dom}(h_1)$ and $h \perp h_1$, $S_{s(x)} \cap \text{Pred}[\mathcal{W}]_{s, h}^X(y) = \emptyset$. From (π_6) we conclude that $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. By (2 f), $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Pred}[\mathcal{W}]_{s', h'}^X(y)$.

(\Leftarrow): Symmetrical to the other direction.

Proof of (σ_4) (\Rightarrow): Suppose $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Cycl}[s]_{s, h_1}^X(\beta)$. Since $S_{s(x)} \subseteq \text{Path}[s]_{s, h_1}^X(x)$, from Proposition 5.31, $\text{Cycl}[s]_{s, h_1}^X(\beta) \cap S_{s(x)} = \emptyset$. From (π_6) we derive $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. By definition of C_β and \bar{C}_β , since $\text{Cycl}[s]_{s, h_1}^X(\beta) = C_\beta \cup \bar{C}_\beta$, we conclude that $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \bar{C}_\beta$. Since $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(f)$, from (c_7) , $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) \in [\bar{C}_\beta]^\flat$. From (c_{11}) and (r) , $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Cycl}[s]_{s', h'_1}^X(\beta)$.

(\Leftarrow) : Symmetrical to the other direction.

Proof of (σ_8) – (σ_{10}) . The statements (σ_8) – (σ_{10}) trivially hold if $\text{Path}[s]_{s, h_1}^X(t) = \emptyset$, which implies $\text{Path}[s]_{s', h'_1}^X(t) = \emptyset$ by (i) . Below, we assume $\text{Path}[s]_{s, h_1}^X(t)$ and $\text{Path}[s]_{s', h'_1}^X(t)$ non-empty. Let $\rho = (\ell_0, \dots, \ell_p)$ be the path in h_1 , described by $\text{Path}[s]_{s, h_1}^X(t)$, going from $\llbracket t \rrbracket_{s, h_1}^X$ to $\text{sby}_{s, h_1}^X(t)$. Similarly, let $\rho' = (\ell'_0, \dots, \ell'_q)$ be the path in h'_1 , described by $\text{Path}[s]_{s', h'_1}^X(t)$, going from $\llbracket t \rrbracket_{s', h'_1}^X$ to $\text{sby}_{s', h'_1}^X(t)$.

(\Rightarrow) : The left-to-right directions of all the statements (σ_8) – (σ_{10}) assume $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Path}[s]_{s, h_1}^X(t)$. We divide the proof depending on whether $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(f)$.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(f)$. In this case, $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \notin S_{s(x)}$ and so, by (π_6) , $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. From (l) and (s_5) , $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Path}[s]_{s', h'_1}^X(t)$, which proves (σ_8) .

Now, if $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket t \rrbracket_{s, h_1}^X = \ell_0$ then, from (s_A) , $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \llbracket t \rrbracket_{s', h'_1}^X = \ell'_0$. Similarly, if $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = h_1(\llbracket t \rrbracket_{s, h_1}^X) = \ell_1$ then $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = h'_1(\llbracket t \rrbracket_{s', h'_1}^X) = \ell'_1$, by (s_6) . Moreover, if $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \ell_{p-1}$ then, by (s_7) , $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \ell'_{q-1}$. Thus, (σ_9) and (σ_{10}) hold.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \notin \text{dom}(f)$. By (π_6) , $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in S_{s(x)}$ and $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in S'_{s(x)}$. Since it holds that $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Path}[s]_{s, h_1}^X(t)$, we have $\llbracket t \rrbracket_{s, h_1}^X = s(x)$ and so, by (s_A) , and $\llbracket t \rrbracket_{s', h'_1}^X = s'(x)$. Notice that this means that $\text{Path}[s]_{s, h_1}^X(t) = \text{Path}[s]_{s, h_1}^X(x)$ and $\text{Path}[s]_{s', h'_1}^X(t) = \text{Path}[s]_{s', h'_1}^X(x)$, which implies $s(x) \in \text{dom}(h_1)$ and $s'(x) \in \text{dom}(h'_1)$. By (l) and (s_4) , $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Path}[s]_{s', h'_1}^X(t)$, which proves (σ_8) .

Now, since $\llbracket t \rrbracket_{s, h_1}^X \in \text{dom}(f)$ and $\llbracket t \rrbracket_{s', h'_1}^X \in \text{ran}(f)$, obviously $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \neq \llbracket t \rrbracket_{s, h_1}^X = \ell_0$ and $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \neq \llbracket t \rrbracket_{s', h'_1}^X = \ell'_0$. By $\llbracket t \rrbracket_{s, h_1}^X = s(x) \in \text{dom}(h_1)$ and $\llbracket t \rrbracket_{s', h'_1}^X = s'(x) \in \text{dom}(h'_1)$, we derive $\ell_1 = h_1(\llbracket t \rrbracket_{s, h_1}^X) = \llbracket n(x) \rrbracket_{s, \vec{h}}^X$ and $\ell'_1 = h'_1(\llbracket t \rrbracket_{s', h'_1}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. Therefore, (σ_9) holds. To show (σ_{10}) , let us assume that $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \ell_{p-1}$. By definition of ℓ_{p-1} , we conclude that $\rho = (\ell_0, \ell_1, \ell_2)$ and therefore $\text{Path}[s]_{s, h_1}^X(t) = \{\ell_0, \ell_1\}$. Since we already showed that $\ell_0 \neq \ell_1$, we conclude that $\text{card}(\text{Path}[s]_{s, h_1}^X(t)) = 2$. By (i) , $\text{card}(\text{Path}[s]_{s', h'_1}^X(t)) = 2$, and thus $\rho' = (\ell'_0, \ell'_1, \ell'_2)$. So, $\ell'_{q-1} = \ell'_1 = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$.

(\Leftarrow) : Symmetrical to the other direction.

We show that $(s, \vec{h}) \approx_{\mathbb{X}, \alpha}^{\mathcal{W}} (s', \vec{h}')$ by relying on the properties (w_A) – (w_D) below.

w_A . Let $t \in T[\mathcal{W}]^X$.

- (a') $\llbracket t \rrbracket_{s, \vec{h}}^X$ and $\llbracket t \rrbracket_{s', \vec{h}'}^X$ are equidefined,
- (b') given $t' \in T[\mathcal{W}]^X$, $\llbracket t \rrbracket_{s, \vec{h}}^X = \llbracket t' \rrbracket_{s, \vec{h}}^X$ iff $\llbracket t \rrbracket_{s', \vec{h}'}^X = \llbracket t' \rrbracket_{s', \vec{h}'}^X$,
- (c') $\llbracket t \rrbracket_{s, \vec{h}}^X \in \text{dom}(\vec{h})$ if and only if $\llbracket t \rrbracket_{s', \vec{h}'}^X \in \text{dom}(\vec{h}')$,
- (d') given $t' \in \mathbb{X} \cup \{t\}$, we have $\vec{h}(\llbracket t \rrbracket_{s, \vec{h}}^X) = \llbracket t' \rrbracket_{s, \vec{h}}^X$ if and only if $\vec{h}'(\llbracket t \rrbracket_{s, \vec{h}}^X) = \llbracket t' \rrbracket_{s', \vec{h}'}^X$,
- (e') $s(u) = \llbracket t \rrbracket_{s, \vec{h}}^X$ if and only if $s'(u) = \llbracket t \rrbracket_{s', \vec{h}'}^X$.

Proof of (a'). Trivial if t is a program variable. Follows from (π_5) for next-point variables.

Proof of (b'). If t and t' are program variables, (b') follows from $(s, h) \approx_{X, \alpha + \text{card}(X)}^W (s', h')$.

Indeed, (s, h) and (s', h') equisatisfy the core formula $t = t'$, which means that $s(t) = s(t')$ if and only if $s'(t) = s'(t')$. Below, we prove (b') in the case where at least one among t and t' is a next-point variable. Without loss of generality, we assume t to be a next-point variable $n(x)$, where $x \in X$, and divide the proof depending on whether t' is a program variable.

case: $t' = y$, for some $y \in X$. (\Rightarrow): Suppose $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = s(y)$, i.e. $\vec{h}(s(x)) = s(y)$.

Since $\vec{h} = h + h_1$, either $h(s(x)) = s(y)$ or $h_1(s(x)) = s(y)$ holds, which in turn implies that $(s, h) \models x \hookrightarrow y$ or $(s, h_1) \models x \hookrightarrow y$. By $(s, h) \approx_{X, \alpha + \text{card}(X)}^W (s', h')$ and $(s, h_1) \approx_{X, \alpha}^W (s', h'_1)$, we conclude that either $(s', h') \models x \hookrightarrow y$ or $(s', h'_1) \models x \hookrightarrow y$.

From the semantics $x \hookrightarrow y$ and by $\vec{h}' = h' + h'_1$, we derive $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = s'(y)$.

(\Leftarrow): Symmetrical to the other direction.

case: $t' = n(y)$, for some $y \in X$. (\Rightarrow): Suppose $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket n(y) \rrbracket_{s, \vec{h}}^X$, and therefore $\vec{h}(s(x)) = \vec{h}(s(y))$. We divide the proof in the following two cases:

case: $\{s(x), s(y)\} \subseteq \text{dom}(h)$ or $\{s(x), s(y)\} \subseteq \text{dom}(h_1)$. Let us consider the case where $\{s(x), s(y)\} \subseteq \text{dom}(h_1)$. The case where $\{s(x), s(y)\} \subseteq \text{dom}(h)$ is analogous. Since $h_1 \subseteq \vec{h}$, we conclude that $h_1(s(x)) = h_1(s(y))$. Thanks to the core formula $n(x) = n(y)$, by $(s, h_1) \approx_{X, \alpha}^W (s', h'_1)$, we have $h'_1(s'(x)) = h'_1(s'(y))$. From $h'_1 \subseteq \vec{h}'$, $\vec{h}'(s'(x)) = \vec{h}'(s'(y))$.

case: $s(x) \in \text{dom}(h)$ and $s(y) \in \text{dom}(h_1)$, or $s(y) \in \text{dom}(h)$ and $s(x) \in \text{dom}(h_1)$.

We consider the case where $s(x) \in \text{dom}(h)$ and $s(y) \in \text{dom}(h_1)$. The other case is analogous. From $s(x) \in \text{dom}(h)$, we conclude that $\llbracket n(x) \rrbracket_{s, h}^X$ is defined, and so by (π_1) , $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. As $s(y) \in \text{dom}(h_1)$, $\text{Path}[s]_{s, h_1}^X(y) \neq \emptyset$. If $\text{card}(\text{Path}[s]_{s', h'_1}^X(x)) = 1$, then, by (π_2) , we conclude that $f(\llbracket n(y) \rrbracket_{s, \vec{h}}^X) = \llbracket n(y) \rrbracket_{s', \vec{h}'}^X$. By $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket n(y) \rrbracket_{s, \vec{h}}^X$, we derive $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \llbracket n(y) \rrbracket_{s', \vec{h}'}^X$. Otherwise, from Lemma 5.14(I), $\llbracket n(y) \rrbracket_{s, h_1}^X = \llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket n(y) \rrbracket_{s, \vec{h}}^X = \llbracket n(x) \rrbracket_{s, h}^X \in \text{Lab}[W]_{s, h}^X$. Together with $\text{card}(\text{Path}[s]_{s', h'_1}^X(x)) \geq 2$, this allows us to apply (π_3) , and conclude that $f(\llbracket n(y) \rrbracket_{s, \vec{h}}^X) = \llbracket n(y) \rrbracket_{s', \vec{h}'}^X$. Again from $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket n(y) \rrbracket_{s, \vec{h}}^X$, we derive $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \llbracket n(y) \rrbracket_{s', \vec{h}'}^X$.

(\Leftarrow): Symmetrical to the other direction

Proof of (c') and (d'). If $t = x$ for some $x \in X$, then both statements follow from (b') .

Indeed, for (c') , we have the following chain of equivalences:

$$s(x) \in \text{dom}(\vec{h}) \text{ iff } \llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket n(x) \rrbracket_{s, \vec{h}}^X \text{ iff } \llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X \text{ iff } s'(x) \in \text{dom}(\vec{h}'),$$

where the first and last double implications hold by definition of $\llbracket \cdot \rrbracket_{\cdot}^X$, whereas the central one is from (b') . Concerning (d') , in this case t' is a program variable, say y . Then, again from the definition of $\llbracket \cdot \rrbracket_{\cdot}^X$ and (b') ,

$$\vec{h}(s(x)) = s(y) \text{ iff } \llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket y \rrbracket_{s, \vec{h}}^X \text{ iff } \llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \llbracket y \rrbracket_{s', \vec{h}'}^X \text{ iff } \vec{h}'(s'(x)) = s'(y).$$

Below, we assume $t = n(x)$, for some $x \in X$. So, t' is either a program variable in X or $n(x)$. We prove (c') and (d') together.

(\Rightarrow): Suppose $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(\vec{h})$. dividing the proof in the following three cases:

case: $\{s(x), \llbracket n(x) \rrbracket_{s, \vec{h}}^X\} \subseteq \text{dom}(h)$ or $\{s(x), \llbracket n(x) \rrbracket_{s, \vec{h}}^X\} \subseteq \text{dom}(h_1)$. Let us consider the case where $\{s(x), \llbracket n(x) \rrbracket_{s, \vec{h}}^X\} \subseteq \text{dom}(h_1)$. The other case is analogous. Since $s(x) \in \text{dom}(h_1)$, by Lemma 5.14(I), we have $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket n(x) \rrbracket_{s, h_1}^X$. Thanks to the core formula $n(x) \hookrightarrow _$, by $(s, h_1) \approx_{X, \alpha}^W (s', h'_1)$, we derive $\llbracket n(x) \rrbracket_{s', h'_1}^X \in \text{dom}(s', h'_1)$. By Lemma 5.14(I) and $h'_1 \subseteq \vec{h}'$, we derive $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{dom}(s', \vec{h}')$, as required by (c'). In order to prove (d'), let us assume that $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket t' \rrbracket_{s, \vec{h}}^X$. From $\{s(x), \llbracket n(x) \rrbracket_{s, \vec{h}}^X\} \subseteq \text{dom}(h_1)$ together with $t' \in X \cup \{n(x)\}$ we conclude that $h_1(\llbracket n(x) \rrbracket_{s, h_1}^X) = \llbracket t' \rrbracket_{s, h_1}^X$. Thanks to the core formulae $n(x) \hookrightarrow x$ and $n(x) \hookrightarrow n(x)$, by $(s, h_1) \approx_{X, \alpha}^W (s', h'_1)$, we derive $h'_1(\llbracket n(x) \rrbracket_{s', h'_1}^X) = \llbracket t' \rrbracket_{s', h'_1}^X$. By Lemma 5.14(I) and $h'_1 \subseteq \vec{h}'$, we conclude that $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = \llbracket t' \rrbracket_{s', \vec{h}'}^X$, as required by (d').

case: $s(x) \in \text{dom}(h_1)$ and $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(h)$. In this case, by Lemma 5.14(I), we have $h_1(s(x)) = \llbracket n(x) \rrbracket_{s, \vec{h}}^X \notin \text{dom}(h_1)$. Therefore $\text{card}(\text{Path}[s]_{s, h_1}^X(x)) = 1$ and thus, by (π₂), $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. We divide the proof in the cases below:

case: there is $t'' \in T[W]^X$ such that $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket t'' \rrbracket_{s, h}^X$. From Lemma 5.14(I), $\llbracket t'' \rrbracket_{s, \vec{h}}^X$. If t'' is a program variable, we can apply (c') and (d') (we dealt with program variables at the beginning of the proof), to conclude that:

i. $\llbracket t'' \rrbracket_{s, \vec{h}}^X \in \text{dom}(\vec{h})$ if and only if $\llbracket t'' \rrbracket_{s', \vec{h}'}^X \in \text{dom}(\vec{h}')$,

ii. given $t' \in X \cup \{t''\}$, $\vec{h}(\llbracket t'' \rrbracket_{s, \vec{h}}^X) = \llbracket t' \rrbracket_{s, \vec{h}}^X$ iff $\vec{h}'(\llbracket t'' \rrbracket_{s', \vec{h}'}^X) = \llbracket t' \rrbracket_{s', \vec{h}'}^X$.

These two statements hold even when $t'' = n(y)$, for some $y \in X$. Indeed, in this case, by Lemma 5.14(I), $\llbracket n(y) \rrbracket_{s, h}^X = \llbracket n(y) \rrbracket_{s, \vec{h}}^X$ which, together with $\llbracket n(y) \rrbracket_{s, h}^X = \llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(h)$, allows us to conclude that $\{s(y), \llbracket n(y) \rrbracket_{s, \vec{h}}^X\} \subseteq \text{dom}(h)$. We can then rely on the previous case of the proof, which leads to (i) and (ii). From $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ and $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \llbracket t'' \rrbracket_{s, h}^X$, by (1_f) we conclude that $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \llbracket t'' \rrbracket_{s', h'}^X = \llbracket t'' \rrbracket_{s', \vec{h}'}^X$, where the last equivalence holds by Lemma 5.14(I). Thus, (c') and (d') hold directly from (i) and (ii).

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Pred}[W]_{s, h}^X(y)$, for some $y \in X$. From (σ₁), we conclude that $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Pred}[W]_{s', h'}^X(y)$. This implies both (c') and (d'). Indeed, (c') holds directly from the fact that $\text{Pred}[W]_{s', h'}^X(x) \subseteq \text{dom}(h') \subseteq \text{dom}(\vec{h}')$. For (d'), we recall that $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Pred}[W]_{s, h}^X(y)$ implies $h(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = s(y)$. We consider a term $t' \in X \cup \{n(x)\}$ such that $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket t' \rrbracket_{s, \vec{h}}^X$. So, $\llbracket t' \rrbracket_{s, \vec{h}}^X = s(y)$. It cannot be that $t' = n(x)$: this would imply $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = s(y)$, and thus $s(y) \in \text{Pred}[W]_{s, h}^X(y)$, which is contradictory, by definition of $\text{Pred}[W]_{s, h}^X(y)$. Therefore, we have $t' = z$, for some $z \in X$. From $(s, h) \approx_{X, \alpha}^W (s', h')$, we conclude $s'(z) = s'(y)$ and so $\text{Pred}[W]_{s', h'}^X(z) = \text{Pred}[W]_{s', h'}^X(y)$. Then, by definition, $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = s'(z)$.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Self}[W]_{s, h}^X$. From (σ₃), we have $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Self}[W]_{s', h'}^X$. Similarly to the previous case, this implies both (c') and (d'). In particular, to prove (d'), directly from the semantics of $\text{Self}[W]_{s, h}^X$ and $\text{Self}[W]_{s', h'}^X$, one shows that both $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s, \vec{h}}^X$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ hold, whereas, for every $y \in X$, $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) \neq s(y)$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq s'(y)$.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Rem}[\mathcal{W}]_{s, h}^X$. From (σ_6) , $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Rem}[\mathcal{W}]_{s', h'}^X$. Similarly to the previous cases, this implies both (c') and (d') . In particular, to prove (d') , directly from the semantics of $\text{Rem}[\mathcal{W}]_{s, h}^X$ and $\text{Rem}[\mathcal{W}]_{s', h'}^X$, one shows that for every $t' \in X \cup \{n(x)\}$, $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) \neq \llbracket t' \rrbracket_{s, \vec{h}}^X$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq \llbracket t' \rrbracket_{s', \vec{h}'}^X$.

Thanks to Proposition 5.13, we know that four cases above exhaust every possibility for $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(h)$, which allows us to conclude that (c') and (d') are always verified (in this case of the proof).

case: $s(x) \in \text{dom}(h)$ and $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(h_1)$. Since $s(x) \in \text{dom}(h)$, $\llbracket n(x) \rrbracket_{s, h}^X$ is defined and so, by (π_1) , $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. As $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{dom}(h_1)$, we divide the proof following the partition of Proposition 5.31.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Pred}[S]_{s, h_1}^X(y)$, for some $y \in Y$. From (σ_2) , we conclude that $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Pred}[S]_{s', h_1'}^X(y)$. Similarly to the case above, involving the sets $\text{Pred}[\mathcal{W}]_{s, h}^X(y)$ and $\text{Pred}[\mathcal{W}]_{s', h'}^X(y)$, this implies both (c') and (d') . In particular, to prove (d') , directly from the semantics of $\text{Pred}[S]_{s, h_1}^X(y)$ and $\text{Pred}[S]_{s', h_1'}^X(y)$, together with $(s, h) \approx_{X, \alpha}^Y (s', h')$, one show that for every $z \in X$,

$$\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = s(z) \text{ iff } s(z) = s(y) \text{ iff } s'(z) = s'(y) \text{ iff } \vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = s'(z).$$

Moreover, $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) \neq \llbracket n(x) \rrbracket_{s, \vec{h}}^X$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in [\text{Cycl}[S]_{s, h_1}^X(1)]^\beta$. By (σ_4) , we have $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in [\text{Cycl}[S]_{s', h_1'}^X(1)]^\beta$. Similarly to the case above, involving the sets $\text{Self}[\mathcal{W}]_{s, h}^X$ and $\text{Self}[\mathcal{W}]_{s', h'}^X$, this implies both (c') and (d') . In particular, for (d') , directly from the semantics of $\text{Cycl}[S]_{s, h}^X(1)$ and $\text{Cycl}[S]_{s', h'}^X(1)$ we conclude that both $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s, \vec{h}}^X$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$ hold, whereas, for all $y \in X$, $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) \neq s(y)$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq s'(y)$.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in [\text{Cycl}[S]_{s, h_1}^X(\beta)]^\beta$, for some $\beta \in [2, \alpha]$. Thanks to (σ_4) , we have $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in [\text{Cycl}[S]_{s', h_1'}^X(\beta)]^\beta$. This implies both (c') and (d') . In particular, we recall that sets in $\text{Cycl}[S]_{s, h_1}^X(\beta)$ and $\text{Cycl}[S]_{s', h_1'}^X(\beta)$ describe unlabelled cycles of length $\beta \geq 2$. So, for every $t' \in X \cup \{n(x)\}$, $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) \neq \llbracket t' \rrbracket_{s, \vec{h}}^X$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq \llbracket t' \rrbracket_{s', \vec{h}'}^X$, which proves (d') .

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in [\uparrow \text{Cycl}[S]_{s, h_1}^{X, \alpha}]^\beta$. Analogous to the previous case. From (σ_5) we have $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in [\uparrow \text{Cycl}[S]_{s', h_1'}^{X, \alpha}]^\beta$. This implies (c') and (d') . For (d') , we have that for all $t' \in X \cup \{n(x)\}$, $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) \neq \llbracket t' \rrbracket_{s, \vec{h}}^X$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq \llbracket t' \rrbracket_{s', \vec{h}'}^X$.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Rem}[S]_{s, h_1}^{X, \alpha}$. Analogous to the previous case. Thanks to (σ_7) , we have $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Rem}[S]_{s', h_1'}^{X, \alpha}$. This implies both (c') and (d') . For (d') , we find that for every $t' \in X \cup \{n(x)\}$, $\vec{h}(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) \neq \llbracket t' \rrbracket_{s, \vec{h}}^X$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq \llbracket t' \rrbracket_{s', \vec{h}'}^X$.

case: $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Path}[S]_{s, h_1}^X(t'')$, where $t'' \in T[S]^X$. Let $\rho = (\ell_0, \dots, \ell_p)$ be the path in h_1 described by $\text{Path}[S]_{s, h_1}^X(t'')$. So, $\ell_0 = \llbracket t'' \rrbracket_{s, h_1}^X$ and $\ell_p = \text{sby}_{s, h_1}^X(t'')$. There is $j \in [0, p-1]$ such that $\llbracket n(x) \rrbracket_{s, \vec{h}}^X = \ell_j$. By (σ_B) , $\text{Path}[S]_{s', h_1'}^X(t'') \neq \emptyset$ and $\text{sby}_{s', h_1'}^X(t'') = f(\text{sby}_{s, h_1}^X(t''))$. Let $\rho' = (\ell'_0, \dots, \ell'_q)$ be the path in h_1' described by $\text{Path}[S]_{s', h_1'}^X(t'')$, where $\ell'_0 = \llbracket t'' \rrbracket_{s', h_1'}^X$ and $\ell_q = \text{sby}_{s', h_1'}^X(t'')$. From (σ_8) , there

is $k \in [0, q - 1]$ such that $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \ell'_k$. This implies (c'), since we have $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Path}[S]_{s', h'_1}^X(t'') \subseteq \text{dom}(h'_1) \subseteq \text{dom}(\vec{h}')$. In order to show (d'), we divide the proof depending on whether $j = p - 1$.

case: $j = p - 1$. From (σ₁₀), $k = q - 1$. Suppose that there is $t' \in X \cup \{n(x)\}$ such that $\vec{h}(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = \llbracket t' \rrbracket_{s', \vec{h}'}^X$.

- * Suppose $t' = z$, for some $z \in X$. Then, $\text{sby}_{s, h_1}^X(t'') = \ell_p = s(z)$. From (s_B) and (1_f), $\text{sby}_{s', h'_1}^X(t'') = \ell'_q = s'(z)$. Therefore, $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = s'(z)$.

- * Suppose $t' = n(x)$. Then, $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \text{sby}_{s, h_1}^X(t'')$. Since $\text{sby}_{s, h_1}^X(t'')$ is a labelled location in $\text{Lab}[S]_{s, h_1}^X$, and $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Path}[S]_{s, h_1}^X(t'')$, we derive that $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X = \llbracket t'' \rrbracket_{s, h_1}^X = \text{sby}_{s, h_1}^X(t'')$, and $\text{card}(\text{Path}[S]_{s, h_1}^X(t'')) = 1$. By (s_B), we have that $\llbracket t'' \rrbracket_{s', h'_1}^X = \text{sby}_{s', h'_1}^X(t'')$ and $\text{card}(\text{Path}[S]_{s', h'_1}^X(t'')) = 1$. From $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in \text{Path}[S]_{s', h'_1}^X(t'')$, this implies $h'_1(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. From $h'_1 \subseteq \vec{h}'$, we derive $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$.

case: $j \neq p - 1$. From (σ₁₀), $k \neq q - 1$. In this case, $\vec{h}(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = \ell_{j+1}$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) = \ell'_{k+1}$. Since ρ is a minimal path going from $\llbracket t'' \rrbracket_{s, h_1}^X$ to $\text{sby}_{s, h_1}^X(t'')$, we have $\ell_j \neq \ell_{j+1}$ and $\ell_{j+1} \notin s(X)$. Similarly, by definition of ρ' , $\ell_k \neq \ell_{k+1}$ and $\ell_{k+1} \notin s'(X)$. We derive that for all $t' \in X \cup \{n(x)\}$, $\vec{h}(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq \llbracket t' \rrbracket_{s', \vec{h}'}^X$ and $\vec{h}'(\llbracket n(x) \rrbracket_{s', \vec{h}'}^X) \neq \llbracket t' \rrbracket_{s', \vec{h}'}^X$. So, (d') is satisfied.

(\Leftarrow): Symmetrically, one can show the right-to-left directions of (c') and (d'). Indeed, despite its length, the proof of the left-to-right direction only uses symmetrical arguments, and relies on the symmetrical properties (σ₁)–(σ₁₀) and (s_B).

Proof of (e'). If t is a program variable, then (e') follows from $(s, h) \approx_{X, \alpha + \text{card}(X)}^{W'} (s', h')$, (more precisely, from the equisatisfaction of the core formulae $u = x$, where $x \in X$). Below, we assume t to be the next-point variable $n(x)$, where $x \in X$.

(\Rightarrow): Suppose $s(u) = \llbracket n(x) \rrbracket_{s, \vec{h}}^X$. From (π₆), we have either $f(\llbracket n(x) \rrbracket_{s, \vec{h}}^X) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$, or $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in S_{s(x)}$ and $\llbracket n(x) \rrbracket_{s', \vec{h}'}^X \in S'_{s'(x)}$. In the former case, by (v), we derive $s'(u) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. In the latter case, by definition of $S_{s(x)}$, $\llbracket n(x) \rrbracket_{s, \vec{h}}^X \in \text{Path}[S]_{s, h_1}^X(x)$, and so $h_1(s(x)) = \llbracket n(x) \rrbracket_{s, \vec{h}}^X$. Similarly, by (l), $h'_1(s(x)) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$. From (j) and $s(u) = \llbracket n(x) \rrbracket_{s, \vec{h}}^X$, we conclude that $s'(u) = \llbracket n(x) \rrbracket_{s', \vec{h}'}^X$.

(\Leftarrow): Symmetrical to the other direction, by relying on (k) instead of (j).

\mathcal{W}_B . For every $x \in X$,

$$(f') \min(\text{card}(\text{Pred}[\mathcal{W}]_{s, \vec{h}}^X(x)), \alpha) = \min(\text{card}(\text{Pred}[\mathcal{W}]_{s', \vec{h}'}^X(x)), \alpha),$$

$$(g') s(u) \in \text{Pred}[\mathcal{W}]_{s, \vec{h}}^X(x) \text{ if and only if } s'(u) \in \text{Pred}[\mathcal{W}]_{s', \vec{h}'}^X(x).$$

Before proving (f') and (g'), let us analyse the set $\text{Pred}[\mathcal{W}]_{s, \vec{h}}^X(x)$. From Lemma 5.14(II),

$$\text{Pred}[\mathcal{W}]_{s, h}^X(x) = (\text{Pred}[\mathcal{W}]_{s, \vec{h}}^X(x) \cap \text{dom}(h)) \cup \{\ell \in \text{Lab}[\mathcal{W}]_{s, \vec{h}}^X \setminus \text{Lab}[\mathcal{W}]_{s, h}^X \mid h(\ell) = s(x)\}.$$

Since the set $\text{Pred}[\mathcal{W}]_{s, \vec{h}}^X(x)$ does not contain locations in $\text{Lab}[\mathcal{W}]_{s, \vec{h}}^X$, the union on the right hand side of the equality above is between disjoint sets. We derive:

$$\text{Pred}[\mathcal{W}]_{s, \vec{h}}^X(x) \cap \text{dom}(h) = \text{Pred}[\mathcal{W}]_{s, h}^X(x) \setminus \{\ell \in \text{Lab}[\mathcal{W}]_{s, \vec{h}}^X \setminus \text{Lab}[\mathcal{W}]_{s, h}^X \mid h(\ell) = s(x)\}.$$

In this equivalence, the constraint $h(\ell) = s(x)$ appearing in the rightmost set is superfluous, as $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$ only contains elements satisfying $h(\ell) = s(x)$. Moreover, every location $\ell \in \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X \setminus \text{Lab}[\mathcal{W}]_{s,h}^X$ is not assigned to a program variable, as it would otherwise belong to $\text{Lab}[\mathcal{W}]_{s,h}^X$. Thus, there is $y \in X$ such that $\llbracket n(y) \rrbracket_{s,\vec{h}}^X = \ell$. We have:

$$(\mu_1) \quad \text{Pred}[\mathcal{W}]_{s,\vec{h}}^X(x) \cap \text{dom}(h) = \text{Pred}[\mathcal{W}]_{s,h}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}.$$

We notice that $\{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}$ is a slight abuse of notations, as $\llbracket n(y) \rrbracket_{s,\vec{h}}^X$ could be undefined. Here, we use it as a shortcut for $\{\ell \in \text{LOC} \mid \text{there is } y \in X, \ell = \llbracket n(y) \rrbracket_{s,\vec{h}}^X\}$.

With a similar analysis, we conclude that:

$$(\mu_2) \quad \text{Pred}[\mathcal{W}]_{s,\vec{h}}^X(x) \cap \text{dom}(h_1) = \text{Pred}[\mathcal{W}]_{s,h_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\},$$

$$(\mu_3) \quad \text{Pred}[\mathcal{W}]_{s',\vec{h}'}^X(x) \cap \text{dom}(h') = \text{Pred}[\mathcal{W}]_{s',h'}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\},$$

$$(\mu_4) \quad \text{Pred}[\mathcal{W}]_{s',\vec{h}'}^X(x) \cap \text{dom}(h'_1) = \text{Pred}[\mathcal{W}]_{s',h'_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}.$$

Moreover, from $\vec{h} = h + h_1$ and $\vec{h}' = h' + h'_1$,

$$(\mu_5) \quad \text{Pred}[\mathcal{W}]_{s,\vec{h}}^X(x) = (\text{Pred}[\mathcal{W}]_{s,\vec{h}}^X(x) \cap \text{dom}(h)) \cup (\text{Pred}[\mathcal{W}]_{s,\vec{h}}^X(x) \cap \text{dom}(h_1)),$$

$$(\mu_6) \quad \text{Pred}[\mathcal{W}]_{s',\vec{h}'}^X(x) = (\text{Pred}[\mathcal{W}]_{s',\vec{h}'}^X(x) \cap \text{dom}(h')) \cup (\text{Pred}[\mathcal{W}]_{s',\vec{h}'}^X(x) \cap \text{dom}(h'_1)).$$

Clearly, the union on the right hand side of these two equalities is between disjoint sets. In order to prove (f') and (g'), we need to represent $\text{Pred}[\mathcal{W}]_{s,h_1}^X(x)$ and $\text{Pred}[\mathcal{W}]_{s',h'_1}^X(x)$ using sets of the strong fragment, as indeed up to now we only analysed (s, h_1) and (s', h'_1) using these sets. Fortunately, from the proof of Lemma 5.35 (see (†)) we know that

$$\text{Pred}[\mathcal{W}]_{s,h_1}^X(x) = \text{Pred}[\mathcal{S}]_{s,h_1}^X(x) \cup \left\{ \ell \in \text{dom}(h_1) \setminus \text{Lab}[\mathcal{W}]_{s,h_1}^X \mid \begin{array}{l} h_1(\ell) = s(x), \ell \in \text{Path}[\mathcal{S}]_{s,h_1}^X(\ell') \\ \text{for some } \ell' \in \text{Lab}[\mathcal{S}]_{s,h_1}^X \end{array} \right\}.$$

Since $\text{Path}[\mathcal{S}]_{s,h_1}^X(\ell') \cap \text{Pred}[\mathcal{S}]_{s,h_1}^X(x) = \emptyset$, the two sets on the right hand side of this equality are disjoint. Following (μ2), we are interested in the set

$$\left\{ \ell \in \text{dom}(h_1) \setminus \text{Lab}[\mathcal{W}]_{s,h_1}^X \mid \begin{array}{l} h_1(\ell) = s(x), \ell \in \text{Path}[\mathcal{S}]_{s,h_1}^X(\ell') \\ \text{for some } \ell' \in \text{Lab}[\mathcal{S}]_{s,h_1}^X \end{array} \right\} \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}.$$

From Lemma 5.14(I), $s(X) \subseteq \text{Lab}[\mathcal{W}]_{s,h_1}^X \subseteq \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\} \cup s(X) = \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X$. Thus, the set above is equivalent to

$$L_1 \stackrel{\text{def}}{=} \left\{ \ell \in \text{dom}(h_1) \mid \begin{array}{l} h_1(\ell) = s(x), \ell \in \text{Path}[\mathcal{S}]_{s,h_1}^X(\ell') \\ \text{for some } \ell' \in \text{Lab}[\mathcal{S}]_{s,h_1}^X \end{array} \right\} \setminus \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X.$$

We conclude that:

$$\begin{aligned} & \text{Pred}[\mathcal{W}]_{s,\vec{h}}^X(x) \\ &= (\text{Pred}[\mathcal{W}]_{s,\vec{h}}^X(x) \cap \text{dom}(h)) \cup (\text{Pred}[\mathcal{W}]_{s,\vec{h}}^X(x) \cap \text{dom}(h_1)) && \text{(by (μ5))} \\ &= (\text{Pred}[\mathcal{W}]_{s,h}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \cup (\text{Pred}[\mathcal{W}]_{s,h_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) && \text{(by (μ1) and (μ2))} \\ &= (\text{Pred}[\mathcal{W}]_{s,h}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \cup (\text{Pred}[\mathcal{S}]_{s,h_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \cup L_1 && \text{(manipulation from (†))} \end{aligned}$$

Note that the three sets in the last expression are mutually disjoint. We write (μ7) to denote this last equality. With a similar analysis, from (μ6), (μ3), (μ4) and (†), the following equality (denoted by (μ8)) holds:

$$\text{Pred}[\mathcal{W}]_{s',\vec{h}'}^X(x) = (\text{Pred}[\mathcal{W}]_{s',h'}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) \cup (\text{Pred}[\mathcal{S}]_{s',h'_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) \cup L'_1,$$

where the three sets in the right hand side of the expression are mutually disjoint, and

$$L'_1 \stackrel{\text{def}}{=} \left\{ \ell \in \text{dom}(h'_1) \middle| \begin{array}{l} h'_1(\ell) = s'(x), \ell \in \text{Path}[S]_{s',h'_1}^X(\ell') \\ \text{for some } \ell' \in \text{Lab}[S]_{s',h'_1}^X \end{array} \right\} \setminus \text{Lab}[\mathcal{W}]_{s',\vec{h}'}^X.$$

We are now ready to prove (f') and (g').

Proof of (f'). We show the following three results:

- f₁. $\min(\text{Pred}[\mathcal{W}]_{s,h}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}, \alpha) = \min(\text{Pred}[\mathcal{W}]_{s',h'}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}, \alpha)$,
- f₂. $\min(\text{Pred}[S]_{s,h_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}, \alpha) = \min(\text{Pred}[S]_{s',h'_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}, \alpha)$,
- f₃. $\text{card}(L_1) = \text{card}(L'_1)$.

Afterwards, (f') holds directly from (μ₇) and (μ₈).

Proof of (f₁). By (b'), we have $\text{card}(\{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(\{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\})$.

Moreover, these two sets have at most $\text{card}(X)$ locations. By (σ₁), for every $y \in X$, $\llbracket n(y) \rrbracket_{s,\vec{h}}^X \in \text{Pred}[\mathcal{W}]_{s,h}^X(x)$ if and only if $\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \in \text{Pred}[\mathcal{W}]_{s',h'}^X(x)$. So,

$$\begin{aligned} k &\stackrel{\text{def}}{=} \text{card}(\text{Pred}[\mathcal{W}]_{s,h}^X(x) \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \\ &= \text{card}(\text{Pred}[\mathcal{W}]_{s',h'}^X(x) \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}). \end{aligned}$$

Now, from $(s, h) \approx_{X, \alpha + \text{card}(X)}^{\mathcal{W}} (s', h')$, the memory states (s, h) and (s', h') satisfy the same formulae of the form $\text{pred}_X^{\mathcal{W}}(x) \geq \beta$, where $\beta \in [1, \alpha + \text{card}(X)]$. From the semantics of these formulae, we conclude that

$$\begin{aligned} \min(\text{card}(\text{Pred}[\mathcal{W}]_{s,h}^X(x)), \alpha + \text{card}(X)) &= \min(\text{card}(\text{Pred}[\mathcal{W}]_{s',h'}^X(x)), \alpha + \text{card}(X)). \quad (\mu_9) \\ \text{By definition, } \text{card}(\text{Pred}[\mathcal{W}]_{s,h}^X(x)) &= \text{card}(\text{Pred}[\mathcal{W}]_{s,h}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) + k \\ \text{and } \text{card}(\text{Pred}[\mathcal{W}]_{s',h'}^X(x)) &= \text{card}(\text{Pred}[\mathcal{W}]_{s',h'}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) + k. \text{ By } (\mu_9), \\ &\min(\text{card}(\text{Pred}[\mathcal{W}]_{s,h}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha + \text{card}(X) - k) \\ &= \min(\text{card}(\text{Pred}[\mathcal{W}]_{s',h'}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha + \text{card}(X) - k). \end{aligned}$$

Since $k \leq \text{card}(X)$, this entails (f₁).

Proof of (f₂). As in (f₁), $\text{card}(\{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(\{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\})$.

From (σ₂) we conclude that

$$k \stackrel{\text{def}}{=} \text{card}(\text{Pred}[S]_{s,h_1}^X(x) \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(\text{Pred}[S]_{s',h'_1}^X(x) \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}).$$

We recall that, by definition of $P_{s(x)}$, by (o) and (p₄),

$$\text{Pred}[S]_{s,h_1}^X(x) = P_{s(x)} \cup (\text{Pred}[S]_{s,h_1}^X(x) \cap \text{dom}(f)),$$

$$\text{Pred}[S]_{s',h'_1}^X(x) = P'_{s'(x)} \cup \{\ell \in \text{ran}(f) \mid f^{-1}(\ell) \in \text{Pred}[S]_{s,h_1}^X(x)\}.$$

Since f is an injection $\text{Pred}[S]_{s,h_1}^X(x) \cap \text{dom}(f)$ and $\{\ell \in \text{ran}(f) \mid f^{-1}(\ell) \in \text{Pred}[S]_{s,h_1}^X(x)\}$ have the same cardinality, say n . Moreover, since $P_{s(x)}$ is disjoint from $S_{s(y)}$ and $P'_{s'(x)}$ is disjoint from $S'_{s'(y)}$, for all $y \in X$, by (π₆) we conclude that

$$\text{Pred}[S]_{s,h_1}^X(x) \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\} \subseteq \text{Pred}[S]_{s,h_1}^X(x) \cap \text{dom}(f),$$

$$\text{Pred}[S]_{s',h'_1}^X(x) \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\} \subseteq \{\ell \in \text{ran}(f) \mid f^{-1}(\ell) \in \text{Pred}[S]_{s,h_1}^X(x)\}.$$

Therefore,

$$\text{card}(\text{Pred}[S]_{s,h_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(P_{s(x)}) + n - k,$$

$$\text{card}(\text{Pred}[S]_{s',h'_1}^X(x) \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) = \text{card}(P'_{s'(x)}) + n - k.$$

Since $n - k \geq 0$ and, by (p₁), $\min(\text{card}(P_{s(x)}), \alpha) = \min(\text{card}(P'_{s'(x)}), \alpha)$, these two equalities imply (f₂).

Proof of (f₃). We define the two following sets:

$$Q \stackrel{\text{def}}{=} \{\ell' \in \text{Lab}[S]_{s,h_1}^X \mid \text{there is } \ell \in \text{Path}[S]_{s,h_1}^X(\ell') \text{ such that } \ell \in L_1\},$$

$$Q' \stackrel{\text{def}}{=} \{\ell' \in \text{Lab}[S]_{s',h'_1}^X \mid \text{there is } \ell \in \text{Path}[S]_{s',h'_1}^X(\ell') \text{ such that } \ell \in L'_1\}.$$

We show that $\text{card}(L_1) = \text{card}(Q)$. Let us assume that $\text{Lab}[S]_{s,h_1}^X = \{\ell_1^L, \dots, \ell_m^L\}$. By definition of L_1 , and since $\text{Path}[S]_{s',h'_1}^X(\ell_1^L), \dots, \text{Path}[S]_{s',h'_1}^X(\ell_m^L)$ are disjoint, every location $\ell \in L_1$ belongs to exactly one of these sets. The converse also holds, that is, for every $\ell' \in Q$ there is exactly one location $\ell \in \text{Path}[S]_{s,h_1}^X(t)$ such that $\ell \in L_1$. *Ad absurdum*, suppose that there are two locations $\ell_1, \ell_2 \in \text{Path}[S]_{s',h'_1}^X(\ell')$, where $\ell' \in \text{Lab}[S]_{s,h_1}^X$, such that $\ell_1, \ell_2 \in L_1$. By definition of L_1 , this implies $h_1(\ell_1) = s(x)$ and $h_1(\ell_2) = s(x)$. However, as $\text{Path}[S]_{s',h'_1}^X(\ell')$ describes a path in h_1 , this implies $\ell_1 = \ell_2$, a contradiction. We conclude that $\text{card}(L_1) = \text{card}(Q)$. Analogously, one shows that $\text{card}(L'_1) = \text{card}(Q')$. Since f is an injection, in order to show (f₃), it is sufficient to show that $f(Q) = Q'$.

(\subseteq): Let $\ell \in Q$. This implies that $\text{Path}[S]_{s,h_1}^X(\ell) \neq \emptyset$. From (s_A), there is $t \in T[S]^X$ such that $\llbracket t \rrbracket_{s,h_1}^X = \ell$ and $f(\ell) = \llbracket t \rrbracket_{s',h'_1}^X$. From (i), $\text{Path}[S]_{s',h'_1}^X(f(\ell)) \neq \emptyset$. Consider $\rho = (\ell_0, \dots, \ell_p)$ to be the path in h_1 described by $\text{Path}[S]_{s,h_1}^X(\ell)$, going from ℓ to $\text{sby}_{s,h_1}^X(\ell)$. Similarly let $\rho' = (\ell'_0, \dots, \ell'_q)$ be the path in h'_1 described by $\text{Path}[S]_{s',h'_1}^X(f(\ell))$, going from $f(\ell)$ to $\text{sby}_{s',h'_1}^X(f(\ell))$. From $\ell \in Q$, we conclude that $\ell_{p-1} \notin \text{Lab}[W]_{s,\vec{h}}^X$ and $\text{sby}_{s,h_1}^X(\ell) = s(x)$. Thanks to the core formula $\text{sees}_X(t, x) \geq 1$, by $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$, we have $\text{sby}_{s,h_1}^X(f(\ell)) = s'(x)$. In order to conclude that $f(\ell) \in Q'$, it is sufficient to show that $\ell'_{q-1} \notin \text{Lab}[W]_{s',\vec{h}'}^X$. Since $\ell_{p-1} \notin \text{Lab}[W]_{s,\vec{h}}^X$, directly from (σ₁₀) we conclude that ℓ'_{q-1} does not correspond to a next-point variable. So, let us consider the case of program variables. *Ad absurdum*, suppose $\ell'_{q-1} = s(y)$, for some $y \in X$. Since $\ell'_{q-1} \in \text{Path}[S]_{s',h'_1}^X(f(\ell))$, we conclude that $s'(y) = \ell'_{q-1} = f(\ell)$ (as usual, $f(\ell)$ is the only location in both $\text{Lab}[S]_{s,h_1}^X$ and $\text{Path}[S]_{s',h'_1}^X(f(\ell))$). By (s_A), $\ell = s(y)$. Moreover, $\text{card}(\text{Path}[S]_{s',h'_1}^X(f(\ell))) = 1$ and so, by (i), $\text{card}(\text{Path}[S]_{s,h_1}^X(\ell)) = 1$. Thus, $\ell_{p-1} = \ell = s(y)$. However, $\ell_{p-1} = s(y)$ contradicts $\ell_{p-1} \notin \text{Lab}[W]_{s,\vec{h}}^X$. Therefore, ℓ'_{q-1} is not assigned to a program variable. We conclude that $\ell'_{q-1} \notin \text{Lab}[W]_{s',\vec{h}'}^X$, and so $f(\ell) \in Q'$.

(\supseteq): Symmetrical to the other direction.

Proof of (g'). (\Rightarrow): Suppose $s(u) \in \text{Pred}[W]_{s,\vec{h}}^X(x)$. As showed in the second to last line of (μ₇), we have $s(u) \notin \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}$ and moreover either $s(u) \in \text{Pred}[W]_{s,h}^X(x)$ or $s(u) \in \text{Pred}[W]_{s,h_1}^X(x)$ holds. By (e'), $s'(u) \notin \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}$. Thanks to the formula $u \in \text{pred}_X^W(x)$, from $(s, h) \approx_{X,\alpha+\text{card}(X)}^W (s', h')$ and $(s, h_1) \approx_{X,\alpha}^W (s', h'_1)$, either $s'(u) \in \text{Pred}[W]_{s',h'}^X(x)$ or $s'(u) \in \text{Pred}[W]_{s',h'_1}^X(x)$ hold. So, $s'(u) \in \text{Pred}[W]_{s',\vec{h}'}^X(x)$.

(\Leftarrow): Symmetrical to the other direction.

$$\mathcal{W}_C. \quad (h') \quad \min(\text{card}(\text{Self}[W]_{s,\vec{h}}^X), \alpha) = \min(\text{card}(\text{Self}[W]_{s',\vec{h}'}^X), \alpha),$$

$$(i') \quad s(u) \in \text{Self}[W]_{s,\vec{h}}^X \text{ if and only if } s'(u) \in \text{Self}[W]_{s',\vec{h}'}^X.$$

Before proving (h') and (i'), let us analyse the set $\text{Self}[W]_{s,\vec{h}}^X$. From Lemma 5.14(III),

$$\text{Self}[W]_{s,h}^X = (\text{Self}[W]_{s,\vec{h}}^X \cap \text{dom}(h)) \cup \{\ell \in \text{Lab}[W]_{s,\vec{h}}^X \setminus \text{Lab}[W]_{s,h}^X \mid h(\ell) = \ell\}.$$

The union on the right hand side of the equality above is between disjoint sets, which allows us to derive:

$$\mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h) = \mathbf{Self}[\mathcal{W}]_{s,h}^X \setminus \{\ell \in \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X \setminus \text{Lab}[\mathcal{W}]_{s,h}^X \mid h(\ell) = \ell\}.$$

In this equality, the constraint $h(\ell) = \ell$ appearing in the rightmost set is superfluous, as $\mathbf{Self}[\mathcal{W}]_{s,h}^X$ only contains elements satisfying $h(\ell) = \ell$. As every $\ell \in \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X \setminus \text{Lab}[\mathcal{W}]_{s,h}^X$ is not assigned to a program variable, there is $y \in X$ such that $\llbracket n(y) \rrbracket_{s,\vec{h}}^X = \ell$. We have:

$$(\nu_1) \quad \mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h) = \mathbf{Self}[\mathcal{W}]_{s,h}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}.$$

With a similar analysis, we conclude that

$$(\nu_2) \quad \mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h_1) = \mathbf{Self}[\mathcal{W}]_{s,h_1}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\},$$

$$(\nu_3) \quad \mathbf{Self}[\mathcal{W}]_{s',\vec{h}'}^X \cap \text{dom}(h') = \mathbf{Self}[\mathcal{W}]_{s',h'}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\},$$

$$(\nu_4) \quad \mathbf{Self}[\mathcal{W}]_{s',\vec{h}'}^X \cap \text{dom}(h'_1) = \mathbf{Self}[\mathcal{W}]_{s',h'_1}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}.$$

Moreover, from $\vec{h} = h + h_1$ and $\vec{h}' = h' + h'_1$,

$$(\nu_5) \quad \mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X = (\mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h)) \cup (\mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h_1)),$$

$$(\nu_6) \quad \mathbf{Self}[\mathcal{W}]_{s',\vec{h}'}^X = (\mathbf{Self}[\mathcal{W}]_{s',\vec{h}'}^X \cap \text{dom}(h')) \cup (\mathbf{Self}[\mathcal{W}]_{s',\vec{h}'}^X \cap \text{dom}(h'_1)).$$

Similarly to the analysis done in order to prove (w_B) , we need to represent $\mathbf{Self}[\mathcal{W}]_{s,h_1}^X$ and $\mathbf{Self}[\mathcal{W}]_{s',h'_1}^X$ using sets of the strong fragment. Fortunately, from the proof of Lemma 5.35 (see (\ddagger)), we know that

$$\mathbf{Self}[\mathcal{W}]_{s,h_1}^X = [\text{Cycl}[\mathcal{S}]_{s,h_1}^X(1)]^\flat \cup \{\ell \in \text{Lab}[\mathcal{S}]_{s,h_1}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_1}^X \mid h_1(\ell) = \ell\}.$$

Since $[\text{Cycl}[\mathcal{S}]_{s,h_1}^X(1)]^\flat \cap \text{Lab}[\mathcal{S}]_{s,h_1}^X = \emptyset$, the two sets on the right hand side of this equality are disjoint. Following (ν_2) , we are interested in the set

$$\{\ell \in \text{Lab}[\mathcal{S}]_{s,h_1}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_1}^X \mid h_1(\ell) = \ell\} \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}.$$

From Lemma 5.14(I), $s(X) \subseteq \text{Lab}[\mathcal{W}]_{s,h_1}^X \subseteq \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\} \cup s(X) = \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X$. Thus, the set above is equivalent to

$$L_1 \stackrel{\text{def}}{=} \{\ell \in \text{Lab}[\mathcal{S}]_{s,h_1}^X \mid h_1(\ell) = \ell\} \setminus \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X.$$

We conclude that:

$$\begin{aligned} & \mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X \\ &= (\mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h)) \cup (\mathbf{Self}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h_1)) \quad (\text{by } (\nu_5)) \\ &= (\mathbf{Self}[\mathcal{W}]_{s,h}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \cup (\mathbf{Self}[\mathcal{W}]_{s,h_1}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \quad (\text{by } (\mu_1) \text{ and } (\nu_2)) \\ &= (\mathbf{Self}[\mathcal{W}]_{s,h}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \cup ([\text{Cycl}[\mathcal{S}]_{s,h_1}^X(1)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \cup L_1 \\ &\qquad\qquad\qquad (\text{manipulation from } (\ddagger)) \end{aligned}$$

Note that the three sets in the last expression are mutually disjoint. We write (ν_7) to denote this last equality. With a similar analysis, from (ν_6) , (ν_3) , (ν_4) and (\ddagger) , the following equality (denoted by (ν_8)) holds:

$$\mathbf{Self}[\mathcal{W}]_{s',\vec{h}'}^X = (\mathbf{Self}[\mathcal{W}]_{s',h'}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) \cup ([\text{Cycl}[\mathcal{S}]_{s',h'_1}^X(1)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) \cup L'_1,$$

where the three sets in the right hand side of the expression are mutually disjoint, and

$$L'_1 \stackrel{\text{def}}{=} \{\ell \in \text{Lab}[\mathcal{S}]_{s',h'_1}^X \mid h'_1(\ell) = \ell\} \setminus \text{Lab}[\mathcal{W}]_{s',\vec{h}'}^X.$$

We are now ready to prove (h') and (i') .

Proof of (h') . We show the following three results:

- $$\begin{aligned}
h_1. \quad & \min(\text{card}(\text{Self}[\mathcal{W}]_{s,h}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha) \\
&= \min(\text{card}(\text{Self}[\mathcal{W}]_{s',h'}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha), \\
h_2. \quad & \min(\text{card}([\text{Cycl}[S]_{s,h_1}^X(1)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha) \\
&= \min(\text{card}([\text{Cycl}[S]_{s',h'_1}^X(1)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha), \\
h_3. \quad & \text{card}(L_1) = \text{card}(L'_1).
\end{aligned}$$

Afterwards, (h') follows directly from (ν_7) and (ν_8) .

The proofs of (h_1) and (h_2) follow very closely the proofs of (f_1) and (f_2) shown in the previous step of the proof.

Proof of (h_1) . By (b') , we have $\text{card}(\{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(\{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\})$.

Moreover, these two sets have at most $\text{card}(X)$ locations. By (σ_3) , for every $y \in X$, $\llbracket n(y) \rrbracket_{s,\vec{h}}^X \in \text{Self}[\mathcal{W}]_{s,h}^X$ if and only if $\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \in \text{Self}[\mathcal{W}]_{s',h'}^X$. So,

$$\begin{aligned}
k &\stackrel{\text{def}}{=} \text{card}(\text{Self}[\mathcal{W}]_{s,h}^X \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \\
&= \text{card}(\text{Self}[\mathcal{W}]_{s',h'}^X \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}).
\end{aligned}$$

Now, from $(s,h) \approx_{X,\alpha+\text{card}(X)}^{\mathcal{W}} (s',h')$, the memory states (s,h) and (s',h') satisfy the same formulae of the form $\text{self}_X^{\mathcal{W}} \geq \beta$, where $\beta \in [1, \alpha + \text{card}(X)]$. From the semantics of these formulae, we conclude that

$$\min(\text{card}(\text{Self}[\mathcal{W}]_{s,h}^X), \alpha + \text{card}(X)) = \min(\text{card}(\text{Self}[\mathcal{W}]_{s',h'}^X), \alpha + \text{card}(X)). \quad (\nu_9)$$

By definition, $\text{card}(\text{Self}[\mathcal{W}]_{s,h}^X) = \text{card}(\text{Self}[\mathcal{W}]_{s,h}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) + k$. Similarly, $\text{card}(\text{Self}[\mathcal{W}]_{s',h'}^X) = \text{card}(\text{Self}[\mathcal{W}]_{s',h'}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) + k$. By (ν_9) ,

$$\begin{aligned}
&\text{min}(\text{card}(\text{Self}[\mathcal{W}]_{s,h}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha + \text{card}(X) - k) \\
&= \text{min}(\text{card}(\text{Self}[\mathcal{W}]_{s',h'}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha + \text{card}(X) - k).
\end{aligned}$$

Since $k \leq \text{card}(X)$, this entails (h_1) .

Proof of (h_2) . As in (f_1) , $\text{card}(\{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(\{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\})$.

From (σ_3) , we conclude that

$$\begin{aligned}
k &\stackrel{\text{def}}{=} \text{card}([\text{Cycl}[S]_{s,h_1}^X(1)]^\flat \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \\
&= \text{card}([\text{Cycl}[S]_{s',h'_1}^X(1)]^\flat \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}).
\end{aligned}$$

We recall that, by definition of C_1 and \bar{C}_1 , by (r) and (c_{11}) ,

$$\begin{aligned}
[\text{Cycl}[S]_{s,h_1}^X(1)]^\flat &= [C_1 \cup \bar{C}_1]^\flat = [C_1]^\flat \cup [\bar{C}_1]^\flat, \\
[\text{Cycl}[S]_{s',h'_1}^X(1)]^\flat &= [C'_1 \cup \bar{C}'_1]^\flat = [C'_1]^\flat \cup [\bar{C}'_1]^\flat.
\end{aligned}$$

We recall that every set in $\text{Cycl}[S]_{s,h_1}^X(1)$ or $\text{Cycl}[S]_{s',h'_1}^X(1)$ is made of a single location. From (c_5) and (c_7) , this means that $[\bar{C}_1]^\flat$ and $[\bar{C}'_1]^\flat$ have the same cardinality, say n . Similarly, from (c_1) ,

$$\min([C_1]^\flat, \mathcal{L}(\alpha)) = \min([C'_1]^\flat, \mathcal{L}(\alpha)), \quad (\nu_{10})$$

where we recall that $\mathcal{L}(\alpha) \geq \alpha$. Moreover, for all $y \in X$, C_1 and \bar{C}_1 are disjoint from $S_{s(y)}$, and C'_1 and \bar{C}'_1 are disjoint from $S'_{s'(y)}$. By (π_6) , together with the definition of C_1 and from (D) , we conclude that

$$\begin{aligned}
[\text{Cycl}[S]_{s,h_1}^X(1)]^\flat \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\} &\subseteq [\bar{C}_1]^\flat, \\
[\text{Cycl}[S]_{s',h'_1}^X(1)]^\flat \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\} &\subseteq [\bar{C}'_1]^\flat.
\end{aligned}$$

Therefore,

$$\text{card}([\text{Cycl}[s]_{s,h_1}^X(1)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(C_1) + n - k,$$

$$\text{card}([\text{Cycl}[s']_{s',h'_1}^X(x)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) = \text{card}(C'_1) + n - k.$$

Since $n - k \geq 0$, from (v₁₀) we derive (h₂).

Proof of (h₃). By definition of L_1 , we have $L_1 \subseteq \text{dom}(\mathfrak{f})$. Since \mathfrak{f} is an injection, we show (h₃) by proving that $\mathfrak{f}(L_1) = L'_1$.

(\subseteq): Suppose $\ell \in L_1$. By definition, we have $\ell \notin \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X$, $\ell \in \text{Lab}[s]_{s,h_1}^X$ and $h_1(\ell) = \ell$. The latter two properties imply that $\text{card}(\text{Path}[s]_{s,h_1}^X(\ell)) = 1$ and $\text{sby}_{s,h_1}^X(\ell) = \ell$. From (s_A) and (s_B), we conclude that $\text{card}(\text{Path}[s']_{s',h'_1}^X(\mathfrak{f}(\ell))) = 1$ and $\text{sby}_{s',h'_1}^X(\mathfrak{f}(\ell)) = \mathfrak{f}(\ell)$. So, $h_1(\mathfrak{f}(\ell)) = \mathfrak{f}(\ell)$ and, to conclude that $\mathfrak{f}(\ell) \in L'_1$, it is sufficient to show that $\mathfrak{f}(\ell) \notin \text{Lab}[\mathcal{W}]_{s',\vec{h}'}^X$. *Ad absurdum*, suppose $\mathfrak{f}(\ell) \in \text{Lab}[\mathcal{W}]_{s',\vec{h}'}^X$. If ℓ is assigned to a program variable then, by (1_f), we conclude that ℓ is assigned to a program variable, which contradicts $\ell \notin \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X$. Otherwise, suppose $\ell = \llbracket n(x) \rrbracket_{s',\vec{h}'}^X$, for some $x \in X$. Since $\mathfrak{f}(\ell) \in \text{ran}(\mathfrak{f})$, by (D) we conclude that $\mathfrak{f}(\ell) \notin S'_{s'(x)}$. However, by (π₆), this implies $\ell = \llbracket n(x) \rrbracket_{s,\vec{h}}^X$, again in contradiction with $\ell \notin \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X$. Therefore, $\mathfrak{f}(\ell) \notin \text{Lab}[\mathcal{W}]_{s',\vec{h}'}^X$. We conclude that $\mathfrak{f}(\ell) \in L'_1$.

(\supseteq): Symmetrical to the other direction.

Proof of (i'). (\Rightarrow): Suppose $s(u) \in \text{Self}[\mathcal{W}]_{s,\vec{h}}^X$. As showed in the second to last line of (v₇), $s(u) \notin \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}$ and we have either $s(u) \in \text{Self}[\mathcal{W}]_{s,h}^X$ or $s(u) \in \text{Self}[\mathcal{W}]_{s,h_1}^X$.

By (e'), $s'(u) \notin \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}$. Thanks to the core formula $u \in \text{self}_X^{\mathcal{W}}$, from $(s, h) \approx_{X,\alpha+\text{card}(X)}^{\mathcal{W}} (s', h')$ and $(s, h_1) \approx_{X,\alpha}^{\mathcal{W}} (s', h'_1)$, either $s'(u) \in \text{Self}[\mathcal{W}]_{s',h'}^X$ or $s'(u) \in \text{Self}[\mathcal{W}]_{s',h'_1}^X$ hold. So, $s'(u) \in \text{Self}[\mathcal{W}]_{s',\vec{h}'}^X$.

(\Leftarrow): Symmetrical to the other direction.

$$\mathcal{W}_D. \quad (j') \min(\text{card}(\text{Rem}[\mathcal{W}]_{s,\vec{h}}^X), \alpha) = \min(\text{card}(\text{Rem}[\mathcal{W}]_{s',\vec{h}'}^X), \alpha),$$

$$(k') \quad s(u) \in \text{Rem}[\mathcal{W}]_{s,\vec{h}}^X \text{ if and only if } s'(u) \in \text{Rem}[\mathcal{W}]_{s',\vec{h}'}^X.$$

Before proving (j') and (k'), we analyse the set $\text{Rem}[\mathcal{W}]_{s,\vec{h}}^X$. From Lemma 5.14(IV),

$$\text{Rem}[\mathcal{W}]_{s,h}^X = (\text{Rem}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h)) \cup \left\{ \ell \in \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X \setminus \text{Lab}[\mathcal{W}]_{s,h}^X \mid \begin{array}{l} \ell \in \text{dom}(h), h'(\ell) \neq \ell \\ \text{and } \forall x \in X, h(\ell) \neq s(x) \end{array} \right\}.$$

Since the set $\text{Rem}[\mathcal{W}]_{s,\vec{h}}^X$ does not contain locations in $\text{Lab}[\mathcal{W}]_{s,\vec{h}}^X$, the union on the right hand side of the equality above is between disjoint sets. We derive:

$$\text{Rem}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h) = \text{Rem}[\mathcal{W}]_{s,h}^X \setminus \left\{ \ell \in \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X \setminus \text{Lab}[\mathcal{W}]_{s,h}^X \mid \begin{array}{l} \ell \in \text{dom}(h), h'(\ell) \neq \ell \\ \text{and } \forall x \in X, h(\ell) \neq s(x) \end{array} \right\}.$$

In this equality, the constraint “ $\ell \in \text{dom}(h), h'(\ell) \neq \ell$ and $\forall x \in X, h(\ell) \neq s(x)$ ” appearing in the rightmost set is superfluous, as $\text{Rem}[\mathcal{W}]_{s,h}^X$ only contains elements satisfying it. Moreover, every location $\ell \in \text{Lab}[\mathcal{W}]_{s,\vec{h}}^X \setminus \text{Lab}[\mathcal{W}]_{s,h}^X$ is not assigned to a program variable, as it would otherwise belong to $\text{Lab}[\mathcal{W}]_{s,h}^X$. Thus, there is $y \in X$ such that $\llbracket n(y) \rrbracket_{s,\vec{h}}^X = \ell$. We have:

$$\text{Rem}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h) = \text{Rem}[\mathcal{W}]_{s,h}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}.$$

With a similar analysis, we conclude that

$$\text{Rem}[\mathcal{W}]_{s,\vec{h}}^X \cap \text{dom}(h_1) = \text{Rem}[\mathcal{W}]_{s,h_1}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\},$$

$$\begin{aligned}\text{Rem}[\mathcal{W}]_{s',\vec{h}'}^{\mathbf{X}} \cap \text{dom}(h') &= \text{Rem}[\mathcal{W}]_{s',h'}^{\mathbf{X}} \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^{\mathbf{X}} \mid y \in \mathbf{X}\}, \\ \text{Rem}[\mathcal{W}]_{s',\vec{h}'}^{\mathbf{X}} \cap \text{dom}(h'_1) &= \text{Rem}[\mathcal{W}]_{s',h'_1}^{\mathbf{X}} \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^{\mathbf{X}} \mid y \in \mathbf{X}\}.\end{aligned}$$

Moreover, from $\vec{h} = h + h_1$ and $\vec{h}' = h' + h'_1$,

$$\begin{aligned}(\xi_1) \quad \text{Rem}[\mathcal{W}]_{s,\vec{h}}^{\mathbf{X}}(x) &= (\text{Rem}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h)) \cup (\text{Rem}[\mathcal{W}]_{s,\vec{h}}^{\mathbf{X}} \cap \text{dom}(h_1)) \\ &= (\text{Rem}[\mathcal{W}]_{s,h}^{\mathbf{X}} \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^{\mathbf{X}} \mid y \in \mathbf{X}\}) \cup (\text{Rem}[\mathcal{W}]_{s,h_1}^{\mathbf{X}} \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^{\mathbf{X}} \mid y \in \mathbf{X}\}), \\ (\xi_2) \quad \text{Rem}[\mathcal{W}]_{s',\vec{h}'}^{\mathbf{X}}(x) &= (\text{Rem}[\mathcal{W}]_{s',\vec{h}'}^{\mathbf{X}} \cap \text{dom}(h')) \cup (\text{Rem}[\mathcal{W}]_{s',\vec{h}'}^{\mathbf{X}} \cap \text{dom}(h'_1)) \\ &= (\text{Rem}[\mathcal{W}]_{s',h'}^{\mathbf{X}} \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^{\mathbf{X}} \mid y \in \mathbf{X}\}) \cup (\text{Rem}[\mathcal{W}]_{s',h'_1}^{\mathbf{X}} \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^{\mathbf{X}} \mid y \in \mathbf{X}\}).\end{aligned}$$

As in (w_B) and (w_C) , we would like to represent $\text{Rem}[\mathcal{W}]_{s,h_1}^{\mathbf{X}}$ and $\text{Rem}[\mathcal{W}]_{s',h'_1}^{\mathbf{X}}$ in terms of sets of the strong fragment. Following the equation (\star) in the proof of Lemma 5.35, we know that

$$\text{Rem}[\mathcal{W}]_{s,h_1}^{\mathbf{X}} = \text{Rem}[S]_{s,h_1}^{\mathbf{X},\alpha} \cup \bigcup_{\beta \in [2,\alpha]} [\text{Cycl}[S]_{s,h_1}^{\mathbf{X}}(\beta)]^{\mathbb{P}} \cup [\uparrow \text{Cycl}[S]_{s,h_1}^{\mathbf{X},\alpha}]^{\mathbb{P}} \cup \bigcup_{\ell \in \text{Lab}[S]_{s,h_1}^{\mathbf{X}}} (\text{Path}[S]_{s,h_1}^{\mathbf{X}}(\ell) \cap \text{Rem}[\mathcal{W}]_{s,h_1}^{\mathbf{X}}).$$

We write (ξ_3) to denote this equivalence. Unfortunately, (ξ_3) is not as useful as the equivalences used during (w_B) and (w_C) , as $\text{Rem}[\mathcal{W}]_{s,h_1}^{\mathbf{X}}$ still appears in the right hand side. However, again in Lemma 5.35, we solved this issue by considering the following Boolean combinations of core formulae in $\text{Core}[S](\mathbf{X}, 1)$:

$$\begin{aligned}\text{var}_{\mathbf{X}}(t) &= \bigvee_{x \in \mathbf{X}} t = x, & \text{unlab}(t) &= t = t \wedge \neg(\text{var}_{\mathbf{X}}(t) \vee \text{next}_{\mathbf{X}}(t)), \\ \text{next}_{\mathbf{X}}(t) &= \neg \text{var}_{\mathbf{X}}(t) \wedge \bigvee_{x \in \mathbf{X}} t =_S n(x), & \text{var.sby}(t) &= \bigvee_{x \in \mathbf{X}} \text{sees}_{\mathbf{X}}(t, x) \geq 1,\end{aligned}$$

where $t \in T[S]^{\mathbf{X}}$ and $T \subseteq T[S]^{\mathbf{X}}$. The semantics of these formulae is recalled below, with respect to the memory state (s, h_1) :

$$\begin{aligned}(s, h_1) \models \text{var}_{\mathbf{X}}(t) &\quad \text{iff } \llbracket t \rrbracket_{s,h_1}^{\mathbf{X}} \text{ is defined and belongs to } s(\mathbf{X}), \\ (s, h_1) \models \text{next}_{\mathbf{X}}(t) &\quad \text{iff } \llbracket t \rrbracket_{s,h_1}^{\mathbf{X}} \text{ is defined and belongs to } \text{Lab}[\mathcal{W}]_{s,h_1}^{\mathbf{X}} \setminus s(\mathbf{X}), \\ (s, h_1) \models \text{unlab}(t) &\quad \text{iff } \llbracket t \rrbracket_{s,h_1}^{\mathbf{X}} \text{ is defined and does not belong to } \text{Lab}[\mathcal{W}]_{s,h_1}^{\mathbf{X}}, \\ (s, h_1) \models \text{var.sby}(t) &\quad \text{iff } \text{sby}_{s,h_1}^{\mathbf{X}}(t) \text{ is defined and belongs to } s(\mathbf{X}).\end{aligned}$$

Below, under the hypothesis that $\text{Path}[S]_{s,h_1}^{\mathbf{X}}(\ell) \neq \emptyset$, we write l_{pre} for the only location $\text{Path}[S]_{s,h_1}^{\mathbf{X}}(\ell)$ such that $h_1(\ell') = \text{sby}_{s,h_1}^{\mathbf{X}}(\ell)$. From these formulae, in Lemma 5.35 we realised (page 164) that if $\text{Path}[S]_{s,h_1}^{\mathbf{X}}(t) \neq \emptyset$ then the following six statements hold, where \mathcal{R} is short for $\text{Path}[S]_{s,h_1}^{\mathbf{X}}(t) \cap \text{Rem}[\mathcal{W}]_{s,h_1}^{\mathbf{X}}$.

- I. if $(s, h_1) \models \text{var}_{\mathbf{X}}(t) \wedge \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[S]_{s,h_1}^{\mathbf{X}}(t) \setminus \{\llbracket t \rrbracket_{s,h_1}^{\mathbf{X}}, h_1(\llbracket t \rrbracket_{s,h_1}^{\mathbf{X}}), l_{\text{pre}}\}$,
- II. if $(s, h_1) \models \text{next}_{\mathbf{X}}(t) \wedge \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[S]_{s,h_1}^{\mathbf{X}}(t) \setminus \{\llbracket t \rrbracket_{s,h_1}^{\mathbf{X}}, l_{\text{pre}}\}$,
- III. if $(s, h_1) \models \text{var}_{\mathbf{X}}(t) \wedge \neg \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[S]_{s,h_1}^{\mathbf{X}}(t) \setminus \{\llbracket t \rrbracket_{s,h_1}^{\mathbf{X}}, h_1(\llbracket t \rrbracket_{s,h_1}^{\mathbf{X}})\}$,
- IV. if $(s, h_1) \models \text{next}_{\mathbf{X}}(t) \wedge \neg \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[S]_{s,h_1}^{\mathbf{X}}(t) \setminus \{\llbracket t \rrbracket_{s,h_1}^{\mathbf{X}}\}$,
- V. if $(s, h_1) \models \text{unlab}(t) \wedge \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[S]_{s,h_1}^{\mathbf{X}}(t) \setminus \{l_{\text{pre}}\}$,
- VI. if $(s, h_1) \models \text{unlab}(t) \wedge \neg \text{var.sby}(t)$ then $\mathcal{R} = \text{Path}[S]_{s,h_1}^{\mathbf{X}}(t)$.

Fundamentally, since the formulae $\text{var}_{\mathbf{X}}(t)$, $\text{next}_{\mathbf{X}}(t)$, $\text{unlab}(t)$ and $\text{var.sby}(t)$ are Boolean combinations of formulae from $\text{Core}[S](\mathbf{X}, 1)$, by $(s, h_1) \approx_{\mathbf{X},\alpha}^S (s', h'_1)$, we conclude that (s, h_1) and (s', h'_1) satisfy the same premises of the implications in (I)–(VI). We use this in order to show (j') and (k').

Proof of (j'). We aim at showing that

$$\begin{aligned}
j_1. \quad & \min(\text{card}(\text{Rem}[\mathcal{W}]_{s,h}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha) \\
&= \min(\text{card}(\text{Rem}[\mathcal{W}]_{s',h'}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha), \\
j_2. \quad & \min(\text{card}(\text{Rem}[\mathcal{W}]_{s,h_1}^X \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha) \\
&= \min(\text{card}(\text{Rem}[\mathcal{W}]_{s',h'_1}^X \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha).
\end{aligned}$$

By (ξ_1) and (ξ_2) , together with $h \perp h_1$ and $h' \perp h'_1$, these two statements imply (j') . The statement (j_1) is proved similarly to (f_1) and (h_1) , and essentially follows from (σ_6) together with $(s, h) \approx_{X, \alpha + \text{card}(X)}^{\mathcal{W}} (s', h')$. We leave its proof to the reader, and focus instead on (j_2) . Following (ξ_3) we know that

$$\begin{aligned}
\text{Rem}[\mathcal{W}]_{s,h_1}^X &= \text{Rem}[S]_{s,h_1}^{X,\alpha} \cup \bigcup_{\beta \in [2,\alpha]} [\text{Cycl}[S]_{s,h_1}^X(\beta)]^\flat \cup [\uparrow \text{Cycl}[S]_{s,h_1}^{X,\alpha}]^\flat \cup \bigcup_{\ell \in \text{Lab}[S]_{s,h_1}^X} \mathcal{R}_\ell, \\
\text{Rem}[\mathcal{W}]_{s',h'_1}^X &= \text{Rem}[S]_{s',h'_1}^{X,\alpha} \cup \bigcup_{\beta \in [2,\alpha]} [\text{Cycl}[S]_{s',h'_1}^X(\beta)]^\flat \cup [\uparrow \text{Cycl}[S]_{s',h'_1}^{X,\alpha}]^\flat \cup \bigcup_{\ell \in \text{Lab}[S]_{s',h'_1}^X} \mathcal{R}'_\ell,
\end{aligned} \tag{\xi_4}$$

where $\mathcal{R}_\ell = \text{Path}[S]_{s,h_1}^X(\ell) \cap \text{Rem}[\mathcal{W}]_{s,h_1}^X$ and $\mathcal{R}'_\ell = \text{Path}[S]_{s',h'_1}^X(\ell) \cap \text{Rem}[\mathcal{W}]_{s',h'_1}^X$. Notice that the unions in the right hand side of these equations are all between disjoint sets. Therefore, in order to show (j_2) it is sufficient to show that

$$\begin{aligned}
j_3. \quad & \min(\text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha} \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha) \\
&= \min(\text{card}(\text{Rem}[S]_{s',h'_1}^{X,\alpha} \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha), \\
j_4. \quad & \text{for every } \beta \in [2,\alpha], \\
& \min(\text{card}([\text{Cycl}[S]_{s,h_1}^X(\beta)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha) \\
&= \min(\text{card}([\text{Cycl}[S]_{s',h'_1}^X(\beta)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha), \\
j_5. \quad & \min(\text{card}([\uparrow \text{Cycl}[S]_{s,h_1}^{X,\alpha}]^\flat \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha) \\
&= \min(\text{card}([\uparrow \text{Cycl}[S]_{s',h'_1}^{X,\alpha}]^\flat \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha), \\
j_6. \quad & \text{for every } \ell \in \text{Lab}[S]_{s,h_1}^X, \\
& \min(\text{card}(\mathcal{R}_\ell \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha) \\
&= \min(\text{card}(\mathcal{R}'_{f(\ell)} \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha).
\end{aligned}$$

In (j_6) , we recall that, by (\mathcal{S}_A) , $f(\text{Lab}[S]_{s,h_1}^X) = \text{Lab}[S]_{s',h'_1}^X$, which allows us to conclude that, by (ξ_4) , $(j_3) - (j_6)$ imply (j_2) . The proofs of (j_4) and (j_5) are very similar. Below, we show the proofs of (j_3) , (j_4) and (most interestingly) (j_6) .

Proof of (j_3) . By (b') , we have $\text{card}(\{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(\{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\})$.

From (σ_7) , we conclude that

$$\begin{aligned}
k &\stackrel{\text{def}}{=} \text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha} \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \\
&= \text{card}(\text{Rem}[S]_{s',h'_1}^{X,\alpha} \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}).
\end{aligned}$$

We recall that, by definition of R , by (x) and (r_4) ,

$$\begin{aligned}
\text{Rem}[S]_{s,h_1}^{X,\alpha} &= R \cup (\text{Rem}[S]_{s,h_1}^{X,\ell} \cap \text{dom}(f)), \\
\text{Rem}[S]_{s',h'_1}^{X,\alpha} &= R' \cup \{\ell \in \text{ran}(f) \mid f^{-1}(\ell) \in \text{Rem}[S]_{s,h_1}^{X,\alpha}\},
\end{aligned}$$

Since f is an injection $\text{Rem}[S]_{s,h_1}^{X,\alpha} \cap \text{dom}(f)$ and $\{\ell \in \text{ran}(f) \mid f^{-1}(\ell) \in \text{Rem}[S]_{s,h_1}^{X,\alpha}\}$ have the same cardinality, say n . Moreover, since R is disjoint from $S_{s(y)}$ and R' is disjoint from $S'_{s'(y)}$, for every $y \in X$, by (π_6) we conclude that

$$\begin{aligned}\text{Rem}[S]_{s,h_1}^{X,\alpha} \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\} &\subseteq \text{Rem}[S]_{s,h_1}^{X,\alpha} \cap \text{dom}(\mathfrak{f}), \\ \text{Rem}[S]_{s',h'_1}^{X,\alpha} \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\} &\subseteq \{\ell \in \text{ran}(\mathfrak{f}) \mid \mathfrak{f}^{-1}(\ell) \in \text{Rem}[S]_{s,h_1}^{X,\alpha}\}.\end{aligned}$$

Therefore,

$$\begin{aligned}\text{card}(\text{Rem}[S]_{s,h_1}^{X,\alpha} \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) &= \text{card}(R) + n - k, \\ \text{card}(\text{Rem}[S]_{s',h'_1}^{X,\alpha} \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) &= \text{card}(R') + n - k.\end{aligned}$$

Since $n - k \geq 0$ and, by (r1), $\min(\text{card}(R), \alpha) = \min(\text{card}(R'), \alpha)$, these two equalities imply (j3).

Proof of (j4). By (b'), we have $\text{card}(\{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(\{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\})$.

From (σ3), we conclude that

$$\begin{aligned}k &\stackrel{\text{def}}{=} \text{card}([\text{Cycl}[S]_{s,h_1}^X(\beta)]^\flat \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) \\ &= \text{card}([\text{Cycl}[S]_{s',h'_1}^X(\beta)]^\flat \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}).\end{aligned}$$

We recall that, by definition of C_β and \overline{C}_β , by (r) and (c11),

$$\begin{aligned}[\text{Cycl}[S]_{s,h_1}^X(\beta)]^\flat &= [C_\beta \cup \overline{C}_\beta]^\flat = [C_\beta]^\flat \cup [\overline{C}_\beta]^\flat, \\ [\text{Cycl}[S]_{s',h'_1}^X(\beta)]^\flat &= [C'_\beta \cup \overline{C}'_\beta]^\flat = [C'_\beta]^\flat \cup [\overline{C}'_\beta]^\flat.\end{aligned}$$

We recall that every set in $\text{Cycl}[S]_{s,h_1}^X(\beta)$ or $\text{Cycl}[S]_{s',h'_1}^X(\beta)$ contains exactly β locations. From (c5), (c7) and (c8), $[\overline{C}_\beta]^\flat$ (resp. $[\overline{C}'_\beta]^\flat$) can be split into two disjoint sets Q and \overline{Q} (resp. Q' and \overline{Q}') such that

$$\begin{aligned}m &\stackrel{\text{def}}{=} \text{card}(Q) = \text{card}(Q'), & \overline{Q} &\subseteq \text{dom}(\mathfrak{f}), & Q \cap \text{dom}(\mathfrak{f}) &= \emptyset, \\ n &\stackrel{\text{def}}{=} \text{card}(\overline{Q}) = \text{card}(\overline{Q}'), & \overline{Q}' &\subseteq \text{ran}(\mathfrak{f}), & Q' \cap \text{ran}(\mathfrak{f}) &= \emptyset.\end{aligned}$$

Moreover, from (c1), and (c3),

$$\min([C_\beta]^\flat, \beta \times \mathcal{L}(\alpha)) = \min([C'_\beta]^\flat, \beta \times \mathcal{L}(\alpha)), \quad (\xi_5)$$

where we recall that $\mathcal{L}(\alpha) \geq \alpha$. Moreover, for all $y \in X$, C_β and \overline{C}_β are disjoint from $S_{s(y)}$, and C'_β and \overline{C}'_β are disjoint from $S'_{s'(y)}$. By (π6), together with the definitions of C_β , Q and Q' , and from (D), we conclude that

$$\begin{aligned}[\text{Cycl}[S]_{s,h_1}^X(\beta)]^\flat \cap \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\} &\subseteq \overline{Q}, \\ [\text{Cycl}[S]_{s',h'_1}^X(\beta)]^\flat \cap \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\} &\subseteq \overline{Q}'.\end{aligned}$$

Therefore,

$$\text{card}([\text{Cycl}[S]_{s,h_1}^X(\beta)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}) = \text{card}(C_\beta) + m + n - k,$$

$$\text{card}([\text{Cycl}[S]_{s',h'_1}^X(\beta)]^\flat \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}) = \text{card}(P'_{s'(x)}) + m + n - k.$$

Since $n - k \geq 0$, from (ξ5) we derive (j4).

Proof of (j6). Let $\ell \in \text{Lab}[S]_{s,h_1}^X$. Consider a term $t \in T[S]^X$ such that $\llbracket t \rrbracket_{s,h_1}^X = \ell$. By (sA), $\mathfrak{f}(\ell) = \llbracket t \rrbracket_{s',h'_1}^X$. If $\text{Path}[S]_{s,h_1}^X(t) = \emptyset$ then, by (i), $\text{Path}[S]_{s'}^X(h'_1) = \emptyset$ and thus (j6) trivially holds. Below, let us assume that $\text{Path}[S]_{s,h_1}^X(t)$ and $\text{Path}[S]_{s',h'_1}^X(t)$ are non-empty. Let $\rho = (\ell_0, \dots, \ell_p)$ be the path described by $\text{Path}[S]_{s,h_1}^X(t)$, going from $\ell_0 = \llbracket t \rrbracket_{s,h_1}^X$ to $\ell_p = \text{sby}_{s,h}^X(t)$. Similarly, let $\rho' = (\ell'_0, \dots, \ell'_q)$ be the path described by $\text{Path}[S]_{s',h'_1}^X(t)$, going from $\ell'_0 = \llbracket t \rrbracket_{s',h'_1}^X$ to $\ell'_q = \text{sby}_{s',h'_1}^X(t)$. We write l_{pre} and l'_{pre} for the locations ℓ_{p-1} and ℓ'_{q-1} , respectively. Thus, $l_{\text{pre}} \in \text{Path}[S]_{s,h_1}^X(t)$ and $h_1(l_{\text{pre}}) = \text{sby}_{s,h_1}^X(t)$, whereas $l'_{\text{pre}} \in \text{Path}[S]_{s',h'_1}^X(t)$ and $h'_1(l_{\text{pre}}) = \text{sby}_{s',h'_1}^X(t)$. By $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$, (s, h_1) and (s', h'_1) equisatisfy the formulae $\text{var}_X(t)$, $\text{next}_X(t)$, $\text{unlab}(t)$ and $\text{var.sby}(t)$. Therefore, (s, h_1) and

(s', h'_1) satisfy the same premises of the statements (I)–(VI), which allows us to conclude that \mathcal{R}_ℓ and $\mathcal{R}'_{f(\ell)}$ can be expressed as

$$\mathcal{R}_\ell = \text{Path}[s]_{s,h_1}^X(t) \setminus L, \quad \mathcal{R}'_{f(\ell)} = \text{Path}[s]_{s',h'_1}^X(t) \setminus L'.$$

where L and L' are defined following the table below:

L_I. if $(s, h_1) \models \text{var}_X(t) \wedge \text{var.sby}(t)$ then

$$L \stackrel{\text{def}}{=} \{\llbracket t \rrbracket_{s,h_1}^X, h_1(\llbracket t \rrbracket_{s,h_1}^X), l_{pre}\}, \quad L' \stackrel{\text{def}}{=} \{\llbracket t \rrbracket_{s',h'_1}^X, h'_1(\llbracket t \rrbracket_{s',h'_1}^X), l'_{pre}\}.$$

L_{II}. if $(s, h_1) \models \text{next}_X(t) \wedge \text{var.sby}(t)$ then

$$L \stackrel{\text{def}}{=} \{\llbracket t \rrbracket_{s,h_1}^X, l_{pre}\} \quad L' \stackrel{\text{def}}{=} \{\llbracket t \rrbracket_{s',h'_1}^X, l'_{pre}\},$$

L_{III}. if $(s, h_1) \models \text{var}_X(t) \wedge \neg \text{var.sby}(t)$ then

$$L \stackrel{\text{def}}{=} \{\llbracket t \rrbracket_{s,h_1}^X, h_1(\llbracket t \rrbracket_{s,h_1}^X)\} \quad L' \stackrel{\text{def}}{=} \{\llbracket t \rrbracket_{s',h'_1}^X, h'_1(\llbracket t \rrbracket_{s',h'_1}^X)\},$$

L_{IV}. if $(s, h_1) \models \text{next}_X(t) \wedge \neg \text{var.sby}(t)$ then

$$L \stackrel{\text{def}}{=} \{\llbracket t \rrbracket_{s,h_1}^X\}, \quad L' \stackrel{\text{def}}{=} \{\llbracket t \rrbracket_{s',h'_1}^X\},$$

L_V. if $(s, h_1) \models \text{unlab}(t) \wedge \text{var.sby}(t)$ then

$$L \stackrel{\text{def}}{=} \{l_{pre}\}, \quad L' \stackrel{\text{def}}{=} \{l'_{pre}\},$$

L_{VI}. if $(s, h_1) \models \text{unlab}(t) \wedge \neg \text{var.sby}(t)$ then

$$L \stackrel{\text{def}}{=} \emptyset \quad L' \stackrel{\text{def}}{=} \emptyset.$$

Therefore, (j₆) requires us to prove that

$$\min(\text{card}((\text{Path}[s]_{s,h_1}^X(t) \setminus L) \setminus \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}), \alpha)$$

$$= \min(\text{card}((\text{Path}[s]_{s',h'_1}^X(t) \setminus L') \setminus \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}), \alpha).$$

For simplicity, we use the following shortcuts:

$$\begin{aligned} \mathcal{P} &= \text{Path}[s]_{s,h_1}^X(t) \cap \text{dom}(f), & \mathcal{N} &= \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}, \\ \mathcal{P}' &= \{\ell' \in \text{ran}(f) \mid f^{-1} \in \text{Path}[s]_{s,h_1}^X(t)\}, & \mathcal{N}' &= \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}. \end{aligned}$$

From (1), (s₄) and by definition of S_ℓ , we have:

$$\text{Path}[s]_{s,h_1}^X(t) = S_\ell \cup \mathcal{P}, \quad \text{Path}[s]_{s',h'_1}^X(t) = S'_{f(\ell)} \cup \mathcal{P}',$$

where the union on the right hand side of these two equalities is between disjoint sets. Since f is injective, $\text{card}(\mathcal{P}) = \text{card}(\mathcal{P}')$. To prove (j₆), we show that:

$$(j_7) \min(\text{card}((S_\ell \setminus L) \setminus \mathcal{N}), \alpha) = \min(\text{card}((S'_{f(\ell)} \setminus L') \setminus \mathcal{N}'), \alpha),$$

$$(j_8) \text{card}((\mathcal{P} \setminus L) \setminus \mathcal{N}) = \text{card}((\mathcal{P}' \setminus L') \setminus \mathcal{N}').$$

Proof of (j₇). We have:

$$x_1. \text{ By definition of } S_\ell \text{ and (D), } S_\ell \cap \text{dom}(f) = \emptyset \text{ and } S'_{f(\ell)} \cap \text{ran}(f) = \emptyset,$$

$$x_2. \text{ By (x}_1\text{), } \llbracket t \rrbracket_{s,h_1}^X \notin S_\ell \text{ and } \llbracket t' \rrbracket_{s',h'_1}^X \notin S'_{f(\ell)},$$

$$x_3. \text{ From (s}_6\text{), } h_1(\llbracket t \rrbracket_{s,h_1}^X) \in S_\ell \text{ if and only if } h'_1(\llbracket t \rrbracket_{s',h'_1}^X) \in S'_{f(\ell)},$$

$$x_4. \text{ From (s}_7\text{), } l_{pre} \in S_\ell \text{ if and only if } l'_{pre} \in S'_{f(\ell)},$$

$$x_5. \text{ By (s}_1\text{), } \min(\text{card}(S_\ell), \mathcal{S}(\alpha)) = \min(\text{card}(S'_{f(\ell)}), \mathcal{S}(\alpha)).$$

We notice that, for every $\alpha \geq 1$, $\mathcal{S}(\alpha) \geq \alpha + 3$. We divide the proof into the following two cases.

case: there is $x \in X$ such that $s(x) = \llbracket t \rrbracket_{s,h_1}^X$. We notice that, in this case, (s, h) satisfies $\text{var}_X(t)$ and so either (L_I) or (L_{III}) applies, which allows us to conclude that $\{\llbracket t \rrbracket_{s,h_1}^X, h_1(\llbracket t \rrbracket_{s,h_1}^X)\} \subseteq L$ and $\{\llbracket t \rrbracket_{s',h'_1}^X, h'_1(\llbracket t \rrbracket_{s',h'_1}^X)\} \subseteq L'$. Therefore, if $\text{card}(\text{Path}[s]_{s,h_1}^X(t)) \leq 2$, and so $S_\ell \subseteq \text{Path}[s]_{s,h_1}^X(t) \subseteq \{\llbracket t \rrbracket_{s,h_1}^X, h_1(\llbracket t \rrbracket_{s,h_1}^X)\}$, we have $\text{card}((S_\ell \setminus L) \setminus \mathcal{N}) = \text{card}((S'_\ell \setminus L') \setminus \mathcal{N}') = 0$ (which implies (j₇)). Indeed, by (i),

we derive $\text{card}(\text{Path}[S]_{s',h'_1}^X(t)) \leq 2$, and so $S'_{f(\ell)} \subseteq \{\llbracket t \rrbracket_{s',h'_1}^X, h_1(\llbracket t \rrbracket_{s',h'_1}^X)\}$. Otherwise, let us assume $\text{card}(\text{Path}[S]_{s,h_1}^X(t)) > 2$. By (i), $\text{card}(\text{Path}[S]_{s',h'_1}^X(t)) > 2$. From (π6), we know that for all $y \in X$, if $\llbracket n(y) \rrbracket_{s,h_1}^X$ (or $\llbracket n(y) \rrbracket_{s',h'_1}^X$) is defined, then either $f(\llbracket n(y) \rrbracket_{s,h_1}^X) = \llbracket n(y) \rrbracket_{s',h'_1}^X$, or $\llbracket n(y) \rrbracket_{s,h}^X \in S_{s(y)}$ and $\llbracket n(y) \rrbracket_{s',h'}^X \in S'_{s'(y)}$. This implies that $\llbracket n(x) \rrbracket_{s,h}^X$ could belong to $S_{s(x)} = S_\ell$ but, on the other hand, for every $y \in X$, if $\llbracket n(y) \rrbracket_{s,h}^X \neq \llbracket n(x) \rrbracket_{s,h}^X$ then it cannot be that $\llbracket n(y) \rrbracket_{s,h}^X \in S_{s(x)} = S_\ell$. Similarly, $\llbracket n(x) \rrbracket_{s',h'}^X$ could belong to $S'_{s'(x)} = S'_{f(\ell)}$ but, on the other hand, for every $y \in X$, if $\llbracket n(y) \rrbracket_{s',h'}^X \neq \llbracket n(x) \rrbracket_{s',h'}^X$ then it cannot be that $\llbracket n(y) \rrbracket_{s',h'}^X \in S'_{s'(x)} = S'_{f(\ell)}$. Since $h_1(\llbracket t \rrbracket_{s,h_1}^X) = \llbracket n(x) \rrbracket_{s,h}^X$ and $h'_1(\llbracket t \rrbracket_{s',h'_1}^X) = \llbracket n(x) \rrbracket_{s',h'}^X$, this allows us to conclude that $P \cap N \subseteq L$ and $P' \cap N' \subseteq L'$, which in turn implies

$$\text{card}((S_\ell \setminus L) \setminus N) = \text{card}(S_\ell \setminus L), \quad \text{card}((S'_{f(\ell)} \setminus L') \setminus N') = \text{card}(S'_{f(\ell)} \setminus L'). \quad (\xi_6)$$

From $\text{card}(\text{Path}[S]_{s,h_1}^X(t)) > 2$ (resp. $\text{card}(\text{Path}[S]_{s',h'_1}^X(t)) > 2$), we have that $\llbracket t \rrbracket_{s,h_1}^X, h_1(\llbracket t \rrbracket_{s,h_1}^X)$ and l_{pre} (resp. $\llbracket t \rrbracket_{s',h'_1}^X, h'_1(\llbracket t \rrbracket_{s',h'_1}^X)$ and l'_{pre}) are distinct locations. From (x2)–(x4), this allows us to conclude that L and L' , we derive $k \stackrel{\text{def}}{=} \text{card}(S_\ell \cap L) = \text{card}(S_{f(\ell)} \cap L') \leq 3$. From (x5) we know that

$$\min(\text{card}(S_\ell), \alpha + 3) = \min(\text{card}(S_{f(\ell)}), \alpha + 3).$$

Subtracting k in both sides, together with the fact that $k \leq 3$, leads to

$$\min(\text{card}(S_\ell) - k, \alpha) = \min(\text{card}(S'_{f(\ell)}) - k, \alpha).$$

Then, from $\text{card}(S_\ell \setminus L) = \text{card}(S_\ell) - k$ and $\text{card}(S'_{f(\ell)} \setminus L') = \text{card}(S'_{f(\ell)}) - k$, by (ξ6) we conclude that (j7) holds.

case: for all $x \in X$, $s(x) \neq \llbracket t \rrbracket_{s,h_1}^X$. In this case, $(s, h) \not\models \text{var}_X(x)$ and therefore, According to the cases (L_I)–(L_{VI}), $L \subseteq \{\llbracket t \rrbracket_{s,h_1}^X, l_{pre}\}$ and $L' \subseteq \{\llbracket t \rrbracket_{s',h'_1}^X, l'_{pre}\}$. From (π6), we know that for all $y \in X$, if $\llbracket n(y) \rrbracket_{s,h_1}^X$ (or $\llbracket n(y) \rrbracket_{s',h'_1}^X$) is defined, then either $f(\llbracket n(y) \rrbracket_{s,h_1}^X) = \llbracket n(y) \rrbracket_{s',h'_1}^X$, or $\llbracket n(y) \rrbracket_{s,h}^X \in S_{s(y)}$ and $\llbracket n(y) \rrbracket_{s',h'}^X \in S'_{s'(y)}$. Since for every $x \in X$ we have $s(x) \neq \llbracket t \rrbracket_{s,h_1}^X$, we derive that, for all $x \in X$, $S_\ell \neq S_{s(x)}$. Similarly, by (S_A), for every $x \in X$ we have $s'(x) \neq \llbracket t \rrbracket_{s',h'_1}^X$, and thus for all $x \in X$, $S'_{f(\ell)} \neq S'_{s'(x)}$. We deduce that $S_\ell \cap N = \emptyset$ and $S'_{f(\ell)} \cap N' = \emptyset$. As in the previous case, we derive

$$\text{card}((S_\ell \setminus L) \setminus N) = \text{card}(S_\ell \setminus L), \quad \text{card}((S'_{f(\ell)} \setminus L') \setminus N') = \text{card}(S'_{f(\ell)} \setminus L'). \quad (\xi_7)$$

By (i) we have that $\text{card}(\text{Path}[S]_{s,h_1}^X(t)) = 1$ if and only if $\text{card}(\text{Path}[S]_{s',h'_1}^X(t)) = 1$. Equivalently, $\llbracket t \rrbracket_{s,h_1}^X = l_{pre}$ if and only if $\llbracket t \rrbracket_{s',h'_1}^X = l'_{pre}$. From (x2)–(x4), this allows us to conclude that L and L' , we derive $k \stackrel{\text{def}}{=} \text{card}(S_\ell \cap L) = \text{card}(S_{f(\ell)} \cap L') \leq 2$. As in the previous case of the proof, together with (x5), this implies that

$$\min(\text{card}(S_\ell) - k, \alpha) = \min(\text{card}(S'_{f(\ell)}) - k, \alpha).$$

Then, from $\text{card}(S_\ell \setminus L) = \text{card}(S_\ell) - k$ and $\text{card}(S'_{f(\ell)} \setminus L') = \text{card}(S'_{f(\ell)}) - k$, by (ξ7) we conclude that (j7) holds.

Proof of (j8). We have:

- $f(\llbracket t \rrbracket_{s,h_1}^X) = \llbracket t \rrbracket_{s',h'_1}^X$.
- From (s6), $h_1(\llbracket t \rrbracket_{s,h_1}^X) \in \text{dom}(f)$ if and only if $h'_1(\llbracket t \rrbracket_{s',h'_1}^X) \in \text{ran}(f)$. Moreover, if $h_1(\llbracket t \rrbracket_{s,h_1}^X) \in \text{dom}(f)$, then $f(h_1(\llbracket t \rrbracket_{s,h_1}^X)) = h'_1(\llbracket t \rrbracket_{s',h'_1}^X)$.
- From (s7), $l_{pre} \in \text{dom}(f)$ if and only if $l'_{pre} \in \text{ran}(f)$. Moreover, if $l_{pre} \in \text{dom}(f)$, then $f(l_{pre}) = l'_{pre}$.

- By definition, $\llbracket t \rrbracket_{s,h_1}^X, l_{pre} \in \text{Path}[S]_{s,h_1}^X(t)$, and $\llbracket t \rrbracket_{s',h'_1}^X, l'_{pre} \in \text{Path}[S]_{s',h'_1}^X(t)$.
- From (i), $h_1(\llbracket t \rrbracket_{s,h_1}^X) \in \text{Path}[S]_{s,h_1}^X(t)$ if and only if $h_1(\llbracket t \rrbracket_{s',h'_1}^X) \in \text{Path}[S]_{s',h'_1}^X(t)$.

From these five statements, by definition of L ad L' , we derive $f(L \cap P) = L' \cap P'$. Since, by definition, for every $y \in X$, P is disjoint from $S_{s(y)}$ and P' is disjoint from $S'_{s'(y)}$, from (π_6) and (σ_8) we have $f(P \cap N) = P' \cap N'$. Together with $f(L \cap P) = L' \cap P'$, this allows us to derive

$$f(P \cap (L \cup N)) = P' \cap (L' \cup N').$$

Since f is injective, this implies $\text{card}(P \cap (L \cup N)) = \text{card}(P' \cap (L' \cup N'))$. Afterwards, since $\text{card}(P) = \text{card}(P')$, (j8) follows:

$$\begin{aligned} \text{card}((P \setminus L) \setminus N) &= \text{card}(P) - \text{card}(P \cap (L \cup N)) \\ &= \text{card}(P') - \text{card}(P' \cap (L' \cup N')) = \text{card}((P' \setminus L') \setminus N'). \end{aligned}$$

This concludes the proof of (j').

Proof of (k'). (\Rightarrow): Suppose $s(u) \in \text{Rem}[\mathcal{W}]_{s,\vec{h}}^X$. From (ξ1), $s(u) \notin \{\llbracket n(y) \rrbracket_{s,\vec{h}}^X \mid y \in X\}$ and we have either $s(u) \in \text{Rem}[\mathcal{W}]_{s,h}^X$ or $s(u) \in \text{Rem}[\mathcal{W}]_{s,h_1}^X$. By (e'), $s'(u) \notin \{\llbracket n(y) \rrbracket_{s',\vec{h}'}^X \mid y \in X\}$. Thanks to the formula $u \in \text{rem}_X^{\mathcal{W}}$, by $(s,h) \approx_{X,\alpha+\text{card}(X)}^{\mathcal{W}} (s',h')$ and $(s,h_1) \approx_{X,\alpha}^{\mathcal{W}} (s',h'_1)$, either $s'(u) \in \text{Rem}[\mathcal{W}]_{s',h'}^X$ or $s'(u) \in \text{Rem}[\mathcal{W}]_{s',h'_1}^X$ holds. By (ξ2), $s'(u) \in \text{Rem}[\mathcal{W}]_{s',\vec{h}'}^X$.

(\Leftarrow): Symmetrical to the other direction.

Proof of $(s,\vec{h}) \approx_{X,\alpha}^{\mathcal{W}} (s',\vec{h}')$. Thanks to the properties (wA)–(wD), we are now ready to prove that $(s,\vec{h}) \approx_{X,\alpha_j}^{\mathcal{W}} (s',\vec{h}')$. Consider a core formula φ in $\text{Core}[\mathcal{W}](X,\alpha)$. Then, $(s,\vec{h}) \models \varphi$ iff $(s,\vec{h}') \models \varphi$, as shown below:

- case: $\varphi = t_1 = t_2$. Follows directly from (wA)(b').
- case: $\varphi = t \hookrightarrow __$. Follows directly from (wA)(c').
- case: $\varphi = t \hookrightarrow x$ or $\varphi = t \hookrightarrow t$. Follows directly from (wA)(d').
- case: $\varphi = \text{pred}_X^{\mathcal{W}}(x) \geq \beta$. Follows directly from (wB)(f').
- case: $\varphi = \text{self}_X^{\mathcal{W}} \geq \beta$. Follows directly from (wC)(h').
- case: $\varphi = \text{rem}_X^{\mathcal{W}} \geq \beta$. Follows directly from (wD)(j').
- case: $\varphi = u = t$. Follows directly from (wA)(e').
- case: $\varphi = u \in \text{pred}_X^{\mathcal{W}}(x)$. Follows directly from (wB)(g').
- case: $\varphi = u \in \text{self}_X^{\mathcal{W}}$. Follows directly from (wC)(i').
- case: $\varphi = u \in \text{rem}_X^{\mathcal{W}}$. Follows directly from (wD)(k').

□

Conclusion

In Chapters 3, 4 and 5, we studied the computational complexity of separation logics featuring reachability predicates. Our main motivation was to design a separation logic that can express robustness properties of memory states, such as acyclicity and garbage freedom, while having a relatively low complexity.

In Chapter 3 we learned that the simple addition of reachability predicates in quantifier-free separation logic $\text{SL}(*, \rightarrow)$ makes the satisfiability problem of $\text{SL}(*, \rightarrow)$ jump from PSPACE to non RE. Surprisingly, this result already holds when $\text{SL}(*, \rightarrow)$ is enriched with the bounded reachability predicates $x \rightarrow^2 y$ and $x \rightarrow^3 y$, stating that the location corresponding to the variable y is reachable from the one corresponding to the variable x in exactly 2 and 3 steps, respectively.

In Chapter 4 we discovered interactions between reachability and submodel reasoning that lead to TOWER-hard logics. Our studies were carried out through the logic ALT, which we showed to be easily captured by several other logics that were already proved TOWER-hard, as for instance modal separation logic and quantified computation tree logic.

After the negative results of Chapters 3 and 4, in Chapter 5 we were finally able to reach our goal: we introduced the logic $\text{SL}([\exists]_1, *, [\neg, \rightarrow]^S_W)$, for which we proved a PSPACE upper bound of its satisfiability problem. This logic extends several well-known separation logics, such as the PSPACE-complete $\text{SL}([\exists]_1, *, \rightarrow)$ and $\text{SL}(*, \text{ls})$. Crucially, the robustness properties lying outside the expressive power of many fragments of separation logic can be directly expressed in $\text{SL}([\exists]_1, *, [\neg, \rightarrow]^S_W)$ as entailment queries, and checked in PSPACE.

Figure 5.24 summarises the results obtained in Chapters 3, 4 and 5.

Taming reachability predicates.

The logic $\text{SL}([\exists]_1, *, [\neg, \rightarrow]^S_W)$ was defined thanks to syntactical restrictions that let us avoid the sources of high complexity identified during Chapters 3 and 4. In particular, Chapter 3 shows the difficulties of dealing with separation logics featuring both reachability predicates and the separating implication \rightarrow , whereas Chapter 4 shows the intractability of the separating conjunction together with reachability predicates and one quantified variable name. In $\text{SL}([\exists]_1, *, [\neg, \rightarrow]^S_W)$, the results of Chapter 3 translate into a constraint on the reachability predicates occurring under the scope of a \neg , and the results of Chapter 4 translate into a constraint on the occurrences of the quantified variable name inside reachability predicates. Of course, when dealing with reachability predicates, other directions are possible. To this end, in [58], co-authored with S. Demri and E. Lozes, we introduce a guarded form of quantification that allows us to solve the issues in Chapter 4 and, for \rightarrow -free separation logics, lead to a PSPACE-complete satisfiability problem. Very recently, J. Pagel and F. Zuleger tackled the issues raised in Chapter 3 by modifying the notion of union of heaps, which leads to a new semantics for the connectives $*$ and \rightarrow [117]. Exactly as in the case of [58], under this new semantics the satisfiability problem

of $\text{SL}(*, \neg*, \text{ls})$ is shown to be PSPACE-complete, instead of non RE. Despite being unable to express the robustness properties (unlike $\text{SL}([\exists]_1, *, [\neg*, \rightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}})$), both the logics in [58] and [117] give new and interesting perspectives which could in the future improve the decision procedures of separation logics featuring reachability predicates.

An auxiliary logic on trees.

Despite not having practical application, the logic **ALT** introduced in Chapter 4 is theoretically interesting, as it happens to be easily captured by various non-elementary logics: first-order separation logic, quantified CTL, modal logic of heaps and modal separation logic. Through **ALT**, we were not only able to connect these logics, but also to refine their analysis and find strict fragments that are still TOWER-hard. Most importantly, with **ALT** we hope to have shown a set of simple and concrete properties, centered around reachability and submodel reasoning, that when put together lead to logics having a non-elementary satisfiability problem.

This work leaves a few questions open. First, the fragments of **ALT** where \blacklozenge or \blacklozenge^* are removed from the logic have not been studied yet. The logic without \blacklozenge^* is of particular interest, as it has strong connections with the sabotage logics from [4].

Second, the analysis done on first-order separation logic and on modal logic of heaps reveals that the complexity of these logics does not change when the $*$ operator and the `emp` predicate are replaced with the less general operators \blacklozenge and \blacklozenge^* (see e.g. Theorem 4.45). We find this point interesting, as from an overview of the literature, it seems that this result also holds for the separation logics considered in [22, 52, 55, 103, 107]. Moreover, for the logics whose expressiveness is known, i.e. the ones in [55, 103], it seems that also the expressive power remains unchanged. However, we struggle to see how to express the $*$ operator with \blacklozenge and \blacklozenge^* in an uniform way.

Lastly, Chapter 4 illustrates the potential of **ALT** as a tool for proving the TOWER-hardness of logics interpreted on tree-like structures. As the operators of our logic are simple, we hope **ALT** to be useful in order to study other logics with unknown complexities.

The core formulae technique.

In order to show the PSPACE upper bound of the satisfiability problem for $\text{SL}([\exists]_1, *, [\neg*, \rightarrow^\dagger]_{\mathcal{W}}^{\mathcal{S}})$, we relied on the core formulae technique introduced by E. Lozes in [104], which we extended to better suit our needs. Ultimately, the technique still suffers some drawbacks: it heavily relies on the ad-hoc definitions of core formulae and, despite the addition of game hops to make the proofs modular, the simulation arguments are lengthy and technical. Nonetheless, this technique is quite general and, as we will see in the following part of the thesis, well-suited to tackle problems that are outside the realm of computational complexity.

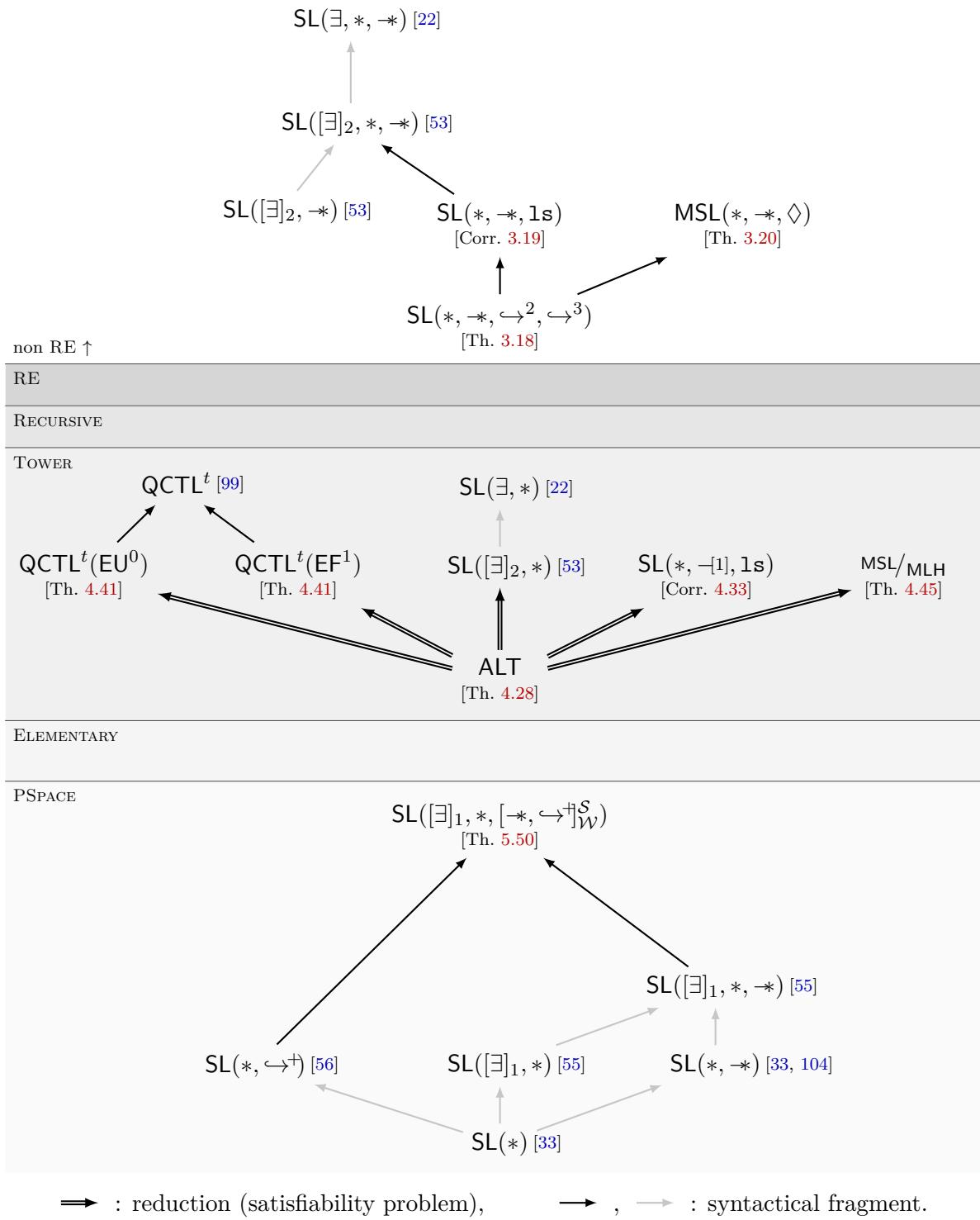


Figure 5.24: Recap: The complexity of Separation Logics.

Part II

Internal Calculi for Spatial Logics

Internal Proof Systems via Core Formulae

In Chapters 6 and 7, we look at the problem of designing internal calculi for separation logics and other spatial logics. The fact that the calculus is internal (or Hilbert-style) for a logic \mathcal{L} simply means that the axioms and inference rules involve schemas instantiated by formulae in \mathcal{L} (no use of nominals, labels or other syntactic objects that are not \mathcal{L} formulae).¹ Designing an internal calculus for your favourite logic is usually quite challenging. This does not lead necessarily to optimal decision procedures, but the completeness proof usually provides essential insights to better understand the logic at hand. That is why many logics related to program verification have been axiomatised, often requiring non-trivial completeness proofs. By way of example, there are internal calculi for the linear-time μ -calculus [95, 61], the modal μ -calculus [141] or for the alternating-time temporal logic ATL [81], the full computation tree logic CTL* [125], for probabilistic extensions of μ -calculus [100] or for a coalgebraic generalisation [129].

Internal calculi for separation logics.

Concerning separation logics, the literature on proof systems is quite vast, but almost completely limited to the abstract level of BBI. An Hilbert-style axiomatisation of BBI has been introduced in [75] (see Section 2.3.3). More recently, HyBBI [27], a hybrid version of Boolean BI has been introduced in order to axiomatise various classes of abstract separation logics; HyBBI naturally considers classes of abstract models (typically preordered partial monoids) but it does not fit exactly the heap semantics of separation logics. Furthermore, the addition of nominals (in the sense of hybrid modal logics, see e.g. [2]) extends substantially the object language.

In [91], labelled sequent calculi are designed for several abstract separation logics by considering different sets of properties. The sequents contain labelled formulae (a formula prefixed by a label to be interpreted as an abstract heap) as well as relational atoms to express relationships between abstract heaps. Though the framework in [91] is modular and very general to handle abstract separation logics, it is not tailored to separation logics with concrete semantics, see [91, Section 7]. In contrast, the paper [90] deals with first-order separation logic with concrete semantics and presents a sound labelled sequent calculus for it. Of course, the calculus cannot be complete (as the logic is not RE, see Theorem 2.13) but more importantly in the context of Chapter 6, completeness is not established for the quantifier-free fragment. In [90], the sequents contain labelled formulae and relational atoms, similarly to [91]. Hence, this does

¹We aim at defining *internal* calculi according to the terminology from the Workshop on External and Internal Calculi for Non-Classical Logics, FLOC'18, Oxford, <http://weic2018.loria.fr>.

not meet our requirements to have a pure axiomatisation in which only logical formulae from quantifier-free separation logic are allowed.

Modularity of the approaches from [25, 27, 91] is further developed in the recent work [59, 60] by proposing a framework for labelled tableaux systems parametrised by the choice of separation theories (in the very sense of [27]). It is remarkable that the developments in [59, 60] are very general as it can handle separation theories that can be expressed in the rich class of so-called coherent first-order formulae, a subset of first-order formulae with a special syntactic form. The first-order axioms are directly translated into inference rules. The calculi use labelled formulae (every formula is decorated by a sign and by a label) as well as constraints enforcing properties between worlds/resources. Unlike [74], the reasoning about labels is not outsourced but handled directly by the calculus. Similarly to the works [74, 27, 91], the labelled tableaux systems handle syntactic objects referring to semantical concepts related to the abstract separation logics that go beyond the only presence of formulae. In a way, modularity of the approach prevents from having an internal calculus.

Motivations.

Since the birth of separation logics, there has been a lot of interest in the study of decidability and computational complexity issues, see e.g. [33, 22, 19, 44, 55, 26] as well as the first part of the thesis, and comparatively less attention to the design of proof systems, and even less with the puristic approach that consists in discarding any external feature such as nominals or labels in the calculi. The well-known advantages of such an approach include an exhaustive understanding of the expressive power of the logic and discarding the use of any external artifact referring to semantical objects.

In this part of the thesis, we advocate a puristic approach and aim at designing a Hilbert-style proof systems for separation logics and more generally other instantiations of BBI, while remaining within the very logical language. Consequently, we only focus on axiomatising separation logics, and we have no claim for practical applications in the field of program verification with separation logics. Aiming at internal calculi is a non-trivial task as the general frameworks for abstract separation logics make use of labels, see e.g. [59, 91]. We cannot rely on label-free calculi for BBI, see e.g. [121, 75], as separation logics are instantiation of BBI interpreted on memory states and therefore require calculi that cannot abstract as much as it is the case for BBI. Finally, there are many translations from separation logics into logics or theories, see e.g. [35, 120, 22, 123]. However, completeness cannot in general be inherited by sublogics as the proof system should only use the sublogic and therefore the axiomatisation of sublogics may lead to different methods.

Contribution of Chapter 6.

We present the first Hilbert-style proof system for the quantifier-free separation logic $SL(*, -*)$, that uses axiom schemas and rules involving only formulae of this logic. Fundamentally, in order to design the proof system we rely on the core formulae technique introduced in Chapter 5. This leads to a modular axiomatisation, starting with a complete axiomatisation of a Boolean algebra of core formulae, and incrementally adding support for the multiplicative connectives: the separating conjunction and the separating implication. More precisely, given the axiomatisation of arbitrary Boolean combinations of core formulae, we add axioms and rules that allow to syntactically transform every formula of $SL(*, -*)$ into such Boolean combinations. Schematically, for

a valid formula φ of $\mathbf{SL}(*, \neg*)$, the proof system is able to derive whether $\vdash \varphi$ holds by showing $\vdash \varphi'$ and $\vdash \varphi' \Leftrightarrow \varphi$, where φ' is a Boolean combination of core formulae. Our methodology leads to a calculus that is divided in three parts: (1) the axiomatisation of Boolean combinations of core formulae, (2) axioms and inference rules to simulate a bottom-up elimination of the separating conjunction, and (3) axioms and inference rules to simulate a bottom-up elimination of the magic wand. A nice property of this methodology is that only the completeness of the axiomatisation of Boolean combinations of core formulae has to be proven by semantical means. The bottom-up elimination of $*$ and $\neg*$ is showed completely syntactically, which leads to a less error-prone proof of completeness of the full calculus. Such an approach that consists in first axiomatising a syntactic fragment of the whole logic (in our case, the core formulae), is best described in [61] (see also [141, 138, 143, 105]).

Contribution of Chapter 7.

The approach introduced in Chapter 6 can be followed for other separation logics, as we did in the paper [58], co-authored with S. Demri and E. Lozes, and in [57], co-authored with S. Demri and R. Fervari. To broaden our perspectives, in Chapter 7 we propose to look at ambient logics, which are instantiations of BBI tailored to reason on distributed systems, and design an Hilbert-style calculus for an ambient-like modal logic. Once again, in order to design the calculus we heavily rely on the core formulae technique. Besides showing another example of very natural Hilbert-style proof system designed with the help of core formulae, Chapter 7 shows interesting connections between separation logics and ambient logics, which we study in depth during the last part of the thesis.

6

A Complete Axiomatisation for Quantifier-free Separation Logic

Contents

6.1	Axiomatising $\text{SL}(*, \neg*)$, Internally	283
6.1.1	The core formulae of $\text{SL}(*, \neg*)$	284
6.1.2	Hilbert-style proof systems.	285
6.2	An Hilbert-style proof system for $\text{SL}(*, \neg*)$	285
6.3	Main ingredients of the method	290
6.4	A Simple Calculus for the Core Formulae	291
6.5	Syntactical elimination of the Separating Conjunction	295
6.6	Syntactical elimination of the Separating Implication	316

In this chapter

We present the first axiomatisation of the quantifier-free separation logic $\text{SL}(*, \neg*)$. The axiomatisation is internal (a.k.a. Hilbert-style), meaning that every axiom and rule of the calculus involves only formulae from $\text{SL}(*, \neg*)$. The proof system extends the axiomatisation of BBI (Section 2.3.3) with axioms that take in account the concrete semantics of separation logic, based on memory states.

In order to design the proof system and prove its adequacy, we rely on the core formulae technique introduced in Chapter 5. This leads to a modular calculus for $\text{SL}(*, \neg*)$, that is split in three subsystems, respectively dealing with: (1) the axiomatisation of Boolean combinations of core formulae, (2) the axiomatisation of the separating conjunction $*$ and (3) the axiomatisation of the separating implication $\neg*$.

Here is a roadmap of the chapter.

Section 6.1. We introduce the core formulae for $\text{SL}(*, \neg*)$ and standard notions for Hilbert-style (a.k.a. internal) proof systems. The set of core formulae for $\text{SL}(*, \neg*)$, firstly studied in [104], is built from the following abbreviations already introduced in Chapter 2:

$$x = y, \quad x \hookrightarrow y, \quad x \hookrightarrow _, \quad \text{size} \geq \beta,$$

where $x, y \in \text{VAR}$ and $\beta \in \mathbb{N}$. As required in Chapter 5, Boolean combinations of core formulae capture the expressiveness of $\text{SL}(*, \neg*)$. The opposite also holds: each core formula belongs to $\text{SL}(*, \neg*)$. This latter property is fundamental in the context of internal proof systems, as all axioms and rules should only consider formulae of the logic.

Section 6.2. We introduce the proof system $\mathcal{H}_C(*, \neg*)$ that shall be proven adequate (i.e. sound and complete) for $\text{SL}(*, \neg*)$. As already stated, the system is divided into three parts, dealing with Boolean combinations of core formulae, the separating conjunction $*$ and the separating implication $\neg*$, respectively. After familiarising with the axioms and rules by carrying out simple proofs in $\mathcal{H}_C(*, \neg*)$, we show that the proof system is sound.

Section 6.3. In this short section, we analyse the method used in order to prove completeness of the system. Following the split of $\mathcal{H}_C(*, \neg*)$ into three subsystems, the completeness argument follows as we show that:

1. the subsystem dealing with the axiomatisation of the core formulae, denoted by \mathcal{H}_C , is a complete proof system for Boolean combinations of core formulae,
2. the subsystem $\mathcal{H}_C(*)$, obtained from \mathcal{H}_C by adding axioms and rules to deal with the separating conjunction $*$, allows to translate into a Boolean combination of core formulae every formula of the form $\varphi * \psi$, where φ and ψ are Boolean combinations of core formulae,
3. the proof system $\mathcal{H}_C(*, \neg*)$, obtained from $\mathcal{H}_C(*)$ by adding axioms and rules to deal with the operator $\neg*$, allows to translate into a Boolean combination of core formulae every formula of the form $\varphi \neg* \psi$, where φ and ψ are Boolean combinations of core formulae.

Section 6.4. We consider the system \mathcal{H}_C , which extends the axiomatisation of propositional calculus due to Lukasiewicz [18] with axioms dealing with the memory model of $\text{SL}(*, \neg*)$, e.g.

$$x \hookrightarrow _ \wedge y \hookrightarrow _ \wedge x \neq y \Rightarrow \mathbf{size} \geq 2.$$

We prove that \mathcal{H}_C is complete with respect to Boolean combination of core formulae. The proof essentially boils down to a countermodel construction: given a conjunction φ of possibly negated core formulae (i.e. *core formulae literals*), if $\varphi \Rightarrow \perp$ it is not derivable in \mathcal{H}_C , then we construct a memory state satisfying φ .

Section 6.5. We move to the system $\mathcal{H}_C(*)$, and show that it is complete for $\mathbf{SL}(*, x \hookrightarrow _)$, i.e. the separation logic obtained from $\mathbf{SL}(*, \neg*)$ by replacing $\neg*$ with formulae of the form $x \hookrightarrow _$. The main technical contribution of the section is given by the lemma below.

Lemma 6.14. Let $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq \text{card}(X)$. Let φ and ψ in $\text{CoreTypes}(X, \alpha)$. If both φ and ψ are satisfiable, then $\vdash_{\mathcal{H}_C(*)} \varphi * \psi \Leftrightarrow \langle *\rangle(\varphi, \psi)$.

Here, $\text{CoreTypes}(X, \alpha)$ is a set of conjunction core formulae literals such that every Boolean combination of core formulae is equivalent to a disjunction of formulae from this set. Besides, $\langle *\rangle(\varphi, \psi)$ is a particular conjunction of core formulae literals, defined in terms of φ and ψ . Once Lemma 6.14 is established, it is quite easy to extend it to arbitrary Boolean combinations of core formulae: for every two Boolean combinations of core formulae φ and ψ there is a Boolean combination of core formulae χ such that $\varphi * \psi \Leftrightarrow \chi$ is derivable in $\mathcal{H}_C(*)$. Then, the completeness of $\mathcal{H}_C(*)$ with respect to the separation logic $\mathbf{SL}(*, x \hookrightarrow _)$ follows directly from the completeness of \mathcal{H}_C for Boolean combinations of core formulae.

Section 6.6. We consider $\mathcal{H}_C(*, \neg*)$ and show the following lemma.

Lemma 6.18 ($\neg*$ -simulation). Let $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq \text{card}(X)$. Let φ and ψ in $\text{CoreTypes}(X, \alpha)$. There is a conjunction $\chi \in \text{Conj}(\text{Core}(X, \alpha))$ such that $\vdash_{\mathcal{H}_C(*, \neg*)} (\varphi \neg* \psi) \Leftrightarrow \chi$.

This result, which analogously to Lemma 6.14 is shown syntactically inside $\mathcal{H}_C(*, \neg*)$, allows us to conclude that $\mathcal{H}_C(*, \neg*)$ is complete for $\mathbf{SL}(*, \neg*)$. Together with its soundness, we conclude.

Theorem 6.22. $\mathcal{H}_C(*, \neg*)$ is an adequate proof system for $\mathbf{SL}(*, \neg*)$.

6.1 AXIOMATISING $\text{SL}(*, \text{-}\ast)$, INTERNALLY

Given a separation logic \mathcal{L} featuring both the multiplicative connectives $*$ and $\text{-}\ast$, in Chapter 5 we introduced the concept of core formulae for \mathcal{L} . Fundamentally, the core formulae are a set of formulae that satisfy the following three properties:¹

1. atomic formulae of \mathcal{L} can be characterised as Boolean combinations of core formulae,
2. given two Boolean combinations of core formulae φ and ψ , there is a Boolean combination of core formulae χ such that $\chi \equiv \varphi * \psi$,
3. given two Boolean combinations of core formulae φ and ψ , there is a Boolean combination of core formulae χ such that $\chi \equiv \varphi \text{-}\ast \psi$.

In order to show these three properties, the arguments used throughout Chapter 5 are completely semantical. In this chapter, we tackle the open problem of defining Hilbert-style proof systems (the formal definition of these types of systems is recalled below) for the quantifier-free separation logic $\text{SL}(*, \text{-}\ast)$, by looking at the following question:

*“Can the properties of core formulae be proven syntactically,
by only relying on derivations inside $\text{SL}(*, \text{-}\ast)$ itself?”*

To answer this question, we first design a Hilbert-style proof system that is sound and complete with respect to Boolean combinations of core formulae. Afterwards, we enrich this system in order to mimic the simulation properties (2) and (3) above.

Let us start by recalling the grammar of $\text{SL}(*, \text{-}\ast)$ (as usual, $x, y \in \text{VAR}$):

$$\begin{array}{lll} \pi := & \top & (\text{true}) \\ & | & \text{emp} & (\text{empty predicate}) \\ & | & x = y & (\text{equality predicate}) \\ & | & x \hookrightarrow y & (\text{points-to predicate}) \end{array} \quad \begin{array}{lll} \varphi := & \pi & (\text{atomic formulae}) \\ & | & \varphi \Rightarrow \varphi \mid \neg \varphi & (\text{Boolean connectives}) \\ & | & \varphi * \varphi & (\text{separating conjunction}) \\ & | & \varphi \text{-}\ast \varphi & (\text{separating implication}) \end{array}$$

Notice that in this chapter, differently from the previous ones, we take the classical implication \Rightarrow as a primitive Boolean connective, and drop the conjunction \wedge . This is done solely because it allows us to define the proof system of $\text{SL}(*, \text{-}\ast)$ by extending the axiomatisation of BBI given in Section 2.3.3, which relies on the suite of Boolean connectives $\{\neg, \Rightarrow\}$. In practice, this change does not have any impact on the result, as we recall that both the suites of connectives $\{\neg, \Rightarrow\}$ and $\{\neg, \wedge\}$ are functionally complete: $\varphi \Rightarrow \psi$ is expressible with $\{\neg, \wedge\}$ as $\neg(\varphi \wedge \neg\psi)$, whereas the formula $\varphi \wedge \psi$ is expressible with $\{\neg, \Rightarrow\}$ as $\neg(\varphi \Rightarrow \neg\psi)$. Moreover, compared to $\{\neg, \wedge\}$ the suite $\{\neg, \Rightarrow\}$ allows for more elegant axiomatisations of classical propositional calculus. As done in Section 2.3.3, in this section we pick the axiomatisation of propositional calculus due to J. Lukasiewicz [18]. Thus, in this chapter, the formulae $\varphi \wedge \psi$ and $\varphi \vee \psi$ should be seen as abbreviations for $\neg(\varphi \Rightarrow \neg\psi)$ and $\neg\varphi \Rightarrow \psi$, respectively. Similarly, \perp and $\varphi \Leftrightarrow \psi$ stand for $\neg\top$ and $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$, respectively, whereas the subtraction $\varphi \text{-}\circledast \psi$ is a shorthand for $\neg(\varphi \text{-}\ast \neg\psi)$. We also remind the reader that, throughout the thesis, we follow the precedence $\{\neg\} > \{\wedge, \vee, *\} > \{\Rightarrow, \Leftrightarrow, \text{-}\ast, \text{-}\circledast\}$ for the various connectives (which, in this chapter, saves a lot of parentheses).

¹In this chapter, we are not particularly concerned by the finiteness of the set of core formulae, thus we remove this constraint from the set of essential properties.

Formula:	Definition:	Semantics w.r.t. (s, h)
$x \hookrightarrow _$	$x \hookrightarrow x \multimap \varphi$	$s(x) \in \text{dom}(h)$
$\mathbf{size} \geq 0$	\top	$\text{card}(h) \geq 0$
$\mathbf{size} \geq 1$	$\neg \mathbf{emp}$	$\text{card}(h) \geq 1$
$\mathbf{size} \geq \beta + 1$	$\neg \mathbf{emp} * \mathbf{size} \geq \beta$	$\text{card}(h) \geq \beta + 1$

Figure 6.1: Auxiliary formulae of $\mathbf{SL}(*, \multimap)$. Note: $x \in \mathbf{VAR}$ and $\beta \in \mathbb{N}$.

6.1.1 The core formulae of $\mathbf{SL}(*, \multimap)$.

In Figure 6.1 we recall some of the auxiliary formulae introduced in Section 2.1.1 for $\mathbf{SL}(\exists, *, \multimap)$, and that are still expressible in $\mathbf{SL}(*, \multimap)$. All these formulae play an important role in the design of the proof system. Indeed, it is well-known [104] that a suitable set of core formulae for $\mathbf{SL}(*, \multimap)$ is given by the formulae of the form

$$x = y, \quad x \hookrightarrow y, \quad x \hookrightarrow _, \quad \mathbf{size} \geq \beta,$$

where $x, y \in \mathbf{VAR}$ and $\beta \in \mathbb{N}$. Notably, every core formula *is* a formula of $\mathbf{SL}(*, \multimap)$. This simple but crucial insight allows us to freely use the core formulae to help us defining the proof system for $\mathbf{SL}(*, \multimap)$, without the risk of going outside the original language.

Definition 6.1 (Core formulae of $\mathbf{SL}(*, \multimap)$). Given $X \subseteq_{\text{fin}} \mathbf{VAR}$ and $\alpha \in \mathbb{N}$, we write $\mathbf{Core}(X, \alpha)$ to denote the following set of *core formulae*:

$$\{x = y, x \hookrightarrow _, x \hookrightarrow y, \mathbf{size} \geq \beta \mid x, y \in X, \beta \in [0, \alpha]\}.$$

We write $\mathbf{Bool}(\mathbf{Core}(X, \alpha))$ for the set of Boolean combinations of formulae from $\mathbf{Core}(X, \alpha)$. Similarly, $\mathbf{Conj}(\mathbf{Core}(X, \alpha))$ stands for the set of conjunctions of literals built upon $\mathbf{Core}(X, \alpha)$. Here, a *literal* is understood as a core formula or its negation. Let $\varphi = L_1 \wedge \dots \wedge L_n$ be a conjunction of literals L_1, \dots, L_n . We write $\mathbf{LIT}(\varphi)$ to denote $\{L_1, \dots, L_n\}$. In forthcoming developments, we are interested in the maximum β (if any) of formulae of the form $\mathbf{size} \geq \beta$ occurring positively in a conjunction of literals, if any. For this reason, we write $\max_{\mathbf{size}}(\varphi)$ for $\max(\{\beta \in \mathbb{N} \mid \mathbf{size} \geq \beta \in \mathbf{LIT}(\varphi)\} \cup \{0\})$. For instance, given $\varphi = x \hookrightarrow _ \wedge \mathbf{size} \geq 2 \wedge \neg \mathbf{size} \geq 4$, we have $\mathbf{LIT}(\varphi) = \{x \hookrightarrow _, \mathbf{size} \geq 2, \neg \mathbf{size} \geq 4\}$, and $\max_{\mathbf{size}}(\varphi) = 2$. Given two conjunctions of literals $\varphi \in \mathbf{Conj}(\mathbf{Core}(X, \alpha_1))$ and $\psi \in \mathbf{Conj}(\mathbf{Core}(X, \alpha_2))$, $\psi \subseteq_{\mathbf{LIT}} \varphi$ stands for $\mathbf{LIT}(\psi) \subseteq \mathbf{LIT}(\varphi)$. We introduce a few more shortcuts and we write

- $\chi \subseteq_{\mathbf{LIT}} \{\varphi \mid \psi\}$ for “ $\chi \subseteq_{\mathbf{LIT}} \varphi$ or $\chi \subseteq_{\mathbf{LIT}} \psi$ ”,
- $\{\varphi \mid \psi\} \subseteq_{\mathbf{LIT}} \chi$ for “ $\varphi \subseteq_{\mathbf{LIT}} \chi$ or $\psi \subseteq_{\mathbf{LIT}} \chi$ ”,
- $\chi \subseteq_{\mathbf{LIT}} \{\varphi ; \psi\}$ for “ $\chi \subseteq_{\mathbf{LIT}} \varphi$ and $\chi \subseteq_{\mathbf{LIT}} \psi$ ”.

Finally, given a finite set of formulae $\Gamma = \{\varphi_1, \dots, \varphi_n\}$, we write $\wedge \Gamma$ for $\varphi_1 \wedge \dots \wedge \varphi_n$. Similarly, $* \Gamma$ stands for $\varphi_1 * \dots * \varphi_n$. Notice that, since the separating conjunction $*$ is associative and commutative (as depicted for instance by the axiom system of BBI, see Section 2.3.3), the semantics of $* \Gamma$ is uniquely defined, regardless of the choice of ordering for $\varphi_1, \dots, \varphi_n$.

6.1.2 Hilbert-style proof systems.

A *Hilbert-style proof system* \mathcal{H} is defined as a set of tuples $((\Phi_1, \dots, \Phi_n), \Psi)$ with $n \geq 0$, where $\Phi_1, \dots, \Phi_n, \Psi$ are *formula schemata* (also known as *axiom schemata*). When $n \geq 1$, $((\Phi_1, \dots, \Phi_n), \Psi)$ is called an *inference rule*, otherwise it is an *axiom*. As usual, formula schemata generalise the notion of formulae by allowing metavariables for formulae (typically φ, ψ, χ), for program variables (typically x, y, z) or for any type of syntactic objects in formulae, depending on the context. The set of formulae *derivable* from \mathcal{H} is the least set S such that for all $((\Phi_1, \dots, \Phi_n), \Psi) \in \mathcal{H}$ and for all substitutions σ , if $\Phi_1\sigma, \dots, \Phi_n\sigma \in S$ then $\Psi\sigma \in S$. Informally, a substitution σ is simply an assignment from metavariables to syntactical objects, that is formulae, program variables etc.

We write $\vdash_{\mathcal{H}} \varphi$ whenever φ is derivable from \mathcal{H} . A proof system \mathcal{H} is *sound* if all derivable formulae are valid. \mathcal{H} is *complete* if all valid formulae are derivable. We say that \mathcal{H} is *adequate* whenever it is both sound and complete. Lastly, \mathcal{H} is *strongly complete* whenever for all sets of formulae Γ and formulae φ , we have $\Gamma \models \varphi$ (semantical entailment) if and only if $\vdash_{\mathcal{H} \cup \Gamma} \varphi$.

Interestingly enough, there is no strongly complete proof system for $\text{SL}(*, \neg*)$, as strong completeness implies compactness (see Theorem 4.11) and separation logic is not compact. We already discussed this result in the context of ALT (Section 4.2.2). Briefly, the formulae in the set $S = \{\text{size} \geq \beta \mid \beta \in \mathbb{N}\}$ cannot be satisfied by a memory state, as heaps have finite domains. However, all finite subsets of S are satisfiable.

Even for the weaker notion of completeness, we remind the reader that it is only possible to provide an adequate axiom system for a logic \mathcal{L} if the validity problem of \mathcal{L} is recursively enumerable. Therefore, it is not possible to axiomatise any of the separation logics in Chapter 3 (e.g. $\text{SL}(*, \neg*, \text{ls})$), as well as the first-order separation logic $\text{SL}(\exists, *, \neg*)$ (Theorem 2.13).

6.2 AN HILBERT-STYLE PROOF SYSTEM FOR $\text{SL}(*, \neg*)$

In Figure 6.2, we present the proof system $\mathcal{H}_C(*, \neg*)$ that shall be shown to be sound and complete for quantifier-free separation logic $\text{SL}(*, \neg*)$. In the axiom (SUB) , $\varphi[y \leftarrow x]$ stands for the formula obtained from φ by replacing with the variable x every occurrence of y . In the three axioms (MONO) , (IN) and (ALLOC) , the notation $\varphi \blacktriangleleft \mathcal{B}$ refers to the axiom schema φ assuming that the Boolean condition \mathcal{B} holds. We highlight the fact that, in these three axioms, \mathcal{B} is a simple syntactical condition. Lastly, we remind the reader that $i \div j \stackrel{\text{def}}{=} \max(0, i - j)$ (see (SIZE)).

$\mathcal{H}_C(*, \neg*)$ contains all the axioms from BBI presented in Section 2.3.3, with the only (stylistical) difference being that the two axioms (ID_L) and (ID_R) of BBI are grouped in the axioms (ID) of $\mathcal{H}_C(*, \neg*)$. In designing the system, we tried to define axioms that are as simple as possible, which helps highlighting the most fundamental properties of $\text{SL}(*, \neg*)$. Since the core formulae in $\mathcal{H}_C(*, \neg*)$ are mere abbreviations of $\text{SL}(*, \neg*)$ formulae, all the axioms in Figure 6.2 belong to the original language of $\text{SL}(*, \neg*)$, making $\mathcal{H}_C(*, \neg*)$ an Hilbert-style proof system. In order to show completeness of $\mathcal{H}_C(*, \neg*)$, we first establish completeness for subsystems of $\mathcal{H}_C(*, \neg*)$, with respect to syntactical fragments of $\text{SL}(*, \neg*)$. In particular, we consider:

1. \mathcal{H}_C : an adequate proof system for the propositional logic of core formulae (see Section 6.4),
2. $\mathcal{H}_C(*)$: an extension of \mathcal{H}_C that is adequate for the logic $\text{SL}(*, x \hookrightarrow _)$, i.e. the logic obtained from $\text{SL}(*, \neg*)$ by removing $\neg*$ at the price of adding the formula alloc $x \hookrightarrow _$ (Section 6.5).
3. The full $\mathcal{H}_C(*, \neg*)$, which can be seen as an extension of $\mathcal{H}_C(*)$ that allows to reason about the separating implication (Section 6.6).

Propositional Calculus:

- $$\begin{aligned} (\text{L}_1) \quad & (\neg\varphi \Rightarrow \varphi) \Rightarrow \varphi \\ (\text{L}_2) \quad & \varphi \Rightarrow (\neg\varphi \Rightarrow \psi) \\ (\text{L}_3) \quad & (\varphi \Rightarrow \psi) \Rightarrow ((\psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \chi)) \end{aligned}$$

$$(\text{MP}) \quad \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$$

Axioms of the core formulae:

- $$\begin{aligned} (\text{ID}) \quad & x = x & (\text{WEAK}) \quad x \hookrightarrow y \Rightarrow x \hookrightarrow _ \\ (\text{SUB}) \quad & \varphi \wedge x = y \Rightarrow \varphi[y \leftarrow x] & (\text{FUNC}) \quad x \hookrightarrow y \wedge x \hookrightarrow z \Rightarrow y = z \\ (\text{ALLOC}) \quad & x \hookrightarrow _ \wedge y \hookrightarrow _ \wedge x \neq y \Rightarrow \text{size} \geq 2 \end{aligned}$$

Axioms of the separating conjunction:

- $$\begin{aligned} (*_{\text{ID}}) \quad & \varphi \Leftrightarrow \varphi * \text{emp} & (*_{\text{MONO}}) \quad e * \top \Rightarrow e \quad \blacktriangleleft [e \in \{\neg\text{emp}, x = y, x \neq y, x \hookrightarrow y\}] \\ (*_{\text{ASSOC}}) \quad & (\varphi * \psi) * \chi \Leftrightarrow \varphi * (\psi * \chi) & (*_{\neg\text{ALLOC}}) \quad \neg x \hookrightarrow _ * \neg x \hookrightarrow _ \Rightarrow \neg x \hookrightarrow _ \\ (*_{\text{COM}}) \quad & \varphi * \psi \Rightarrow \psi * \varphi & (*_{\neg\text{PTO}}) \quad (x \hookrightarrow _ \wedge \neg x \hookrightarrow y) * \top \Rightarrow \neg x \hookrightarrow y \\ (*_{\text{ALIAS}}) \quad & (x \hookrightarrow _ * x \hookrightarrow _) \Leftrightarrow \perp & (*_{\neg\text{SIZE}}) \quad \neg \text{size} \geq \beta_1 * \neg \text{size} \geq \beta_2 \Rightarrow \neg \text{size} \geq \beta_1 + \beta_2 - 1 \\ (*_{\text{ATOM}}^1) \quad & \neg \text{emp} \Rightarrow \text{size} = 1 * \top & (*_{\text{ATOM}}^2) \quad x \hookrightarrow _ \Rightarrow (x \hookrightarrow _ \wedge \text{size} = 1) * \top \end{aligned}$$

Axioms of the separating implication:

- $$\begin{aligned} (*_{\infty}) \quad & (\text{size} = 1 \wedge \bigwedge_{x \in X} \neg x \hookrightarrow _) \dashv \top \quad \blacktriangleleft [X \subseteq_{\text{fin}} \text{VARD}] \\ (*_{\rightarrow}) \quad & \neg x \hookrightarrow _ \Rightarrow ((x \hookrightarrow y \wedge \text{size} = 1) \dashv \top) \\ (*_{\text{ALLOC}}) \quad & \neg x \hookrightarrow _ \Rightarrow ((x \hookrightarrow _ \wedge \text{size} = 1 \wedge \bigwedge_{y \in X} \neg x \hookrightarrow y) \dashv \top) \quad \blacktriangleleft [X \subseteq_{\text{fin}} \text{VARD}] \end{aligned}$$

Rules of inference for the multiplicative connectives:

$$(*) \quad \frac{\varphi \Rightarrow \chi}{\varphi * \psi \Rightarrow \chi * \psi} \quad (*_{\neg 1}) \quad \frac{\varphi \Rightarrow (\psi * \chi)}{\varphi * \psi \Rightarrow \chi} \quad (*_{\neg 2}) \quad \frac{\varphi * \psi \Rightarrow \chi}{\varphi \Rightarrow (\psi * \chi)}$$

Figure 6.2: The Hilbert-style proof system $\mathcal{H}_C(*, \neg)$.

For the completeness of \mathcal{H}_C and $\mathcal{H}_C(*)$, we need to consider intermediate axioms that reveal to be derivable in the full proof system $\mathcal{H}_C(*, \neg)$, and thus are omitted in Figure 6.2. By convention, these intermediate axioms are denoted with names of the form $I_i^?$.

Notice how the three systems \mathcal{H}_C , $\mathcal{H}_C(*)$, and $\mathcal{H}_C(*, \neg)$ roughly correspond to the three properties of core formulae discussed at the beginning of the chapter. In this regard, the main “task” of $\mathcal{H}_C(*)$ is to produce a bottom-up elimination of the separating conjunction $*$, at the price of introducing Boolean combinations of core formulae, which can be proved valid thanks

1	$\text{emp} \Rightarrow \neg\text{size} \geq 1$	($\neg\neg\text{I}$) and def. of $\text{size} \geq 1$
2	$x \hookrightarrow _ \wedge \text{size} = 1 \Rightarrow \neg\text{size} \geq 2$	($\wedge\text{E}_L$)
3	$\text{emp} * (x \hookrightarrow _ \wedge \text{size} = 1) \Rightarrow \neg\text{size} \geq 1 * \neg\text{size} \geq 2$	($*\text{I}_{LR}$), 1, 2
4	$\neg\text{size} \geq 1 * \neg\text{size} \geq 2 \Rightarrow \neg\text{size} \geq 2$	($\neg\text{size}^*$)
5	$\text{emp} * (x \hookrightarrow _ \wedge \text{size} = 1) \Rightarrow \neg\text{size} \geq 2$	($\Rightarrow\text{TR}$), 3, 4
6	$\text{emp} \Rightarrow (x \hookrightarrow _ \wedge \text{size} = 1 \neg* \neg\text{size} \geq 2)$	($*\text{2}$), 5

Figure 6.3: A proof of $\text{emp} \Rightarrow ((x \hookrightarrow _ \wedge \text{size} = 1) \neg* \neg\text{size} \geq 2)$.

to \mathcal{H}_C . Similarly, the axioms and rules added to $\mathcal{H}_C(*)$ to define $\mathcal{H}_C(*, \neg*)$ are dedicated to perform a bottom-up elimination of the separating implication. A merit of this methodology is that only the completeness of the calculus \mathcal{H}_C is proved using the standard countermodel method. The additional steps required to prove the completeness of $\mathcal{H}_C(*)$ and $\mathcal{H}_C(*, \neg*)$ are (almost) completely syntactical. For instance, to show the completeness of $\mathcal{H}_C(*)$, we consider arbitrary Boolean combinations of core formulae φ and ψ , and exhibiting a Boolean combination of core formulae χ such that $\varphi * \psi \Leftrightarrow \chi$ is valid. We show that this validity can be *syntactically* proved within $\mathcal{H}_C(*)$, and then rely on the fact that \mathcal{H}_C is complete for Boolean combination of core formulae to deduce that $\mathcal{H}_C(*)$ is complete for $\text{SL}(*, x \hookrightarrow _)$.

Along the chapter, we shall have the opportunity to explain the intuition between the axioms and rules. As we can see, the proof system is essentially divided in five parts. The first one corresponds to the standard axiomatisation of propositional calculus due to J. Łukasiewicz, and already introduced for BBI in Section 2.3.3. The second part is made of the five axioms ($\overline{\text{id}}$)–($\overline{\text{func}}$) that deal with the core formulae, and whose semantics should be quite immediate to grasp. The third part features the axioms (id^*), (assoc^*) and (com^*) that capture the notion of non-deterministic monoid of BBI, together with the axioms (alias^*)–(atom^*) that characterise how the separating conjunction behaves with respect to the core formulae. The fourth part features the axioms (∞^*), (\top^*), and (alloc^*) dedicated to the interaction between the separating implication and core formulae. They are expressed with the help of the subtraction operator $\neg*$ to ease the understanding. For instance, the axiom (∞^*) states that it is always possible to add a one-memory-cell heap h' to some heap h while none of the variables from a finite set X is allocated in h' . This natural property in our framework would not hold in general if LOC was not an infinite set. Obviously, the subtraction $\neg*$ is also understood as an abbreviation. Lastly, the rules of inference ($*$), ($*_1$) and ($*_2$) borrowed from BBI conclude the proof system.

Example 6.2. To get familiar with the axioms and the rules of $\mathcal{H}_C(*, \neg*)$, in Figure 6.3, we present a proof of $\text{emp} \Rightarrow (x \hookrightarrow _ \wedge \text{size} = 1 \neg* \neg\text{size} \geq 2)$. In the proof, a line “ $j \mid \chi \ A, i_1, \dots, i_k$ ” states that χ is a theorem denoted by the index j and derivable by the axiom or the rule A . If A is a rule, the indices $i_1, \dots, i_k < j$ denote the theorems used as premises in order to derive χ . When a formula is obtained as a propositional tautology or by pure propositional reasoning from other formulae, we may simply write “PC” (short for Propositional Calculus). Similarly, we provide any useful piece of information justifying the derivation, such as “Ind. hypothesis”, “See ...” or “Previously derived”. In the example, we use the rule ($*_2$), which together with the rule ($*_1$) states that the connectives $*$ and $\neg*$ are adjoint operators, as well as the axiom ($\neg\text{size}^*$), stating

that $\text{card}(h) \leq \beta_1 + \beta_2$ holds whenever a heap h can be split into two subheaps of cardinalities less than $\beta_1 + 1$ and $\beta_2 + 1$, respectively. We also use the following theorems and rules:

$$(\wedge E_L) \quad \psi \wedge \varphi \Rightarrow \varphi \quad (\neg\neg I) \quad \varphi \Rightarrow \neg\neg\varphi \quad (\Rightarrow TR) \quad \frac{\varphi \Rightarrow \chi \quad \chi \Rightarrow \psi}{\varphi \Rightarrow \psi} \quad (*I_{LR}) \quad \frac{\varphi \Rightarrow \varphi' \quad \psi \Rightarrow \psi'}{\varphi * \psi \Rightarrow \varphi' * \psi'}$$

The first two theorems and the first rule are derivable by pure propositional reasoning. By way of example, we show that the inference rule $(*I_{LR})$ is admissible.

<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">1</td><td style="padding: 2px;">$\varphi \Rightarrow \varphi'$</td><td style="padding: 2px;">Hypothesis</td><td style="padding: 2px;">5</td><td style="padding: 2px;">$\varphi' * \psi \Rightarrow \psi * \varphi'$</td><td style="padding: 2px;">$(*)_{\text{COM}}$</td></tr> <tr><td style="padding: 2px;">2</td><td style="padding: 2px;">$\psi \Rightarrow \psi'$</td><td style="padding: 2px;">Hypothesis</td><td style="padding: 2px;">6</td><td style="padding: 2px;">$\psi' * \varphi' \Rightarrow \varphi' * \psi'$</td><td style="padding: 2px;">$(*)_{\text{COM}}$</td></tr> <tr><td style="padding: 2px;">3</td><td style="padding: 2px;">$\varphi * \psi \Rightarrow \varphi' * \psi$</td><td style="padding: 2px;">$(*)$, 1</td><td style="padding: 2px;">7</td><td style="padding: 2px;">$\varphi * \psi \Rightarrow \psi * \varphi'$</td><td style="padding: 2px;">$(\Rightarrow TR)$, 3, 5</td></tr> <tr><td style="padding: 2px;">4</td><td style="padding: 2px;">$\psi * \varphi' \Rightarrow \psi' * \varphi'$</td><td style="padding: 2px;">$(*)$, 2</td><td style="padding: 2px;">8</td><td style="padding: 2px;">$\varphi * \psi \Rightarrow \varphi' * \psi'$</td><td style="padding: 2px;">$(\Rightarrow TR)$ twice, 7, 4, 6</td></tr> </table>	1	$\varphi \Rightarrow \varphi'$	Hypothesis	5	$\varphi' * \psi \Rightarrow \psi * \varphi'$	$(*)_{\text{COM}}$	2	$\psi \Rightarrow \psi'$	Hypothesis	6	$\psi' * \varphi' \Rightarrow \varphi' * \psi'$	$(*)_{\text{COM}}$	3	$\varphi * \psi \Rightarrow \varphi' * \psi$	$(*)$, 1	7	$\varphi * \psi \Rightarrow \psi * \varphi'$	$(\Rightarrow TR)$, 3, 5	4	$\psi * \varphi' \Rightarrow \psi' * \varphi'$	$(*)$, 2	8	$\varphi * \psi \Rightarrow \varphi' * \psi'$	$(\Rightarrow TR)$ twice, 7, 4, 6	
1	$\varphi \Rightarrow \varphi'$	Hypothesis	5	$\varphi' * \psi \Rightarrow \psi * \varphi'$	$(*)_{\text{COM}}$																				
2	$\psi \Rightarrow \psi'$	Hypothesis	6	$\psi' * \varphi' \Rightarrow \varphi' * \psi'$	$(*)_{\text{COM}}$																				
3	$\varphi * \psi \Rightarrow \varphi' * \psi$	$(*)$, 1	7	$\varphi * \psi \Rightarrow \psi * \varphi'$	$(\Rightarrow TR)$, 3, 5																				
4	$\psi * \varphi' \Rightarrow \psi' * \varphi'$	$(*)$, 2	8	$\varphi * \psi \Rightarrow \varphi' * \psi'$	$(\Rightarrow TR)$ twice, 7, 4, 6																				

Remark 6.3. An alternative proof of theorem 5 in Figure 6.3 consists in applying $(\Rightarrow TR)$ to theorem 2 and $\text{emp} * (x \hookrightarrow _ \wedge \text{size} = 1) \Rightarrow x \hookrightarrow _ \wedge \text{size} = 1$, which holds by $(*)_{\text{ID}}$ and $(*)_{\text{COM}}$.

Example 6.4. In Figure 6.4, we develop the proof of $\text{emp} \Rightarrow (x \hookrightarrow _ \wedge \text{size} = 1 \rightarrow \text{size} = 1)$ as a more complete example. We use the following theorems and rules:

$$(*\wedge D_L) \quad (\varphi * \psi) \wedge (\varphi * \chi) \Rightarrow (\varphi * \psi \wedge \chi) \quad (\wedge \top I_L) \quad \varphi \Rightarrow \top \wedge \varphi \quad (\wedge I_L) \quad \frac{\varphi \Rightarrow \chi}{\varphi \wedge \psi \Rightarrow \chi \wedge \psi}$$

The axiom $(\wedge \top I_L)$ and the rule $(\wedge I_L)$ are derivable by propositional reasoning. We show the admissibility of the axiom $(*\wedge D_L)$.

<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td style="padding: 2px;">1</td><td style="padding: 2px;">$\varphi \circledast (\neg\psi \vee \neg\chi) \Rightarrow (\varphi \circledast \neg\psi) \vee (\varphi \circledast \neg\chi)$</td><td style="padding: 2px;">$(I_{6.19.7}^*)$, Lemma 6.19</td></tr> <tr><td style="padding: 2px;">2</td><td style="padding: 2px;">$\neg\varphi \circledast \neg(\neg\psi \vee \neg\chi) \Rightarrow \neg(\varphi \circledast \neg\neg\psi) \vee \neg(\varphi \circledast \neg\neg\chi)$</td><td style="padding: 2px;">Def. \circledast, 1</td></tr> <tr><td style="padding: 2px;">3</td><td style="padding: 2px;">$\neg(\varphi \circledast \psi \wedge \chi) \Rightarrow \neg(\varphi \circledast \psi) \vee \neg(\varphi \circledast \chi)$</td><td style="padding: 2px;">Replacement of equivalents, 2</td></tr> <tr><td style="padding: 2px;">4</td><td style="padding: 2px;">$(\varphi \circledast \psi) \wedge (\varphi \circledast \chi) \Rightarrow (\varphi \circledast \psi \wedge \chi)$</td><td style="padding: 2px;">PC, 3</td></tr> </table>	1	$\varphi \circledast (\neg\psi \vee \neg\chi) \Rightarrow (\varphi \circledast \neg\psi) \vee (\varphi \circledast \neg\chi)$	$(I_{6.19.7}^*)$, Lemma 6.19	2	$\neg\varphi \circledast \neg(\neg\psi \vee \neg\chi) \Rightarrow \neg(\varphi \circledast \neg\neg\psi) \vee \neg(\varphi \circledast \neg\neg\chi)$	Def. \circledast , 1	3	$\neg(\varphi \circledast \psi \wedge \chi) \Rightarrow \neg(\varphi \circledast \psi) \vee \neg(\varphi \circledast \chi)$	Replacement of equivalents, 2	4	$(\varphi \circledast \psi) \wedge (\varphi \circledast \chi) \Rightarrow (\varphi \circledast \psi \wedge \chi)$	PC, 3	
1	$\varphi \circledast (\neg\psi \vee \neg\chi) \Rightarrow (\varphi \circledast \neg\psi) \vee (\varphi \circledast \neg\chi)$	$(I_{6.19.7}^*)$, Lemma 6.19											
2	$\neg\varphi \circledast \neg(\neg\psi \vee \neg\chi) \Rightarrow \neg(\varphi \circledast \neg\neg\psi) \vee \neg(\varphi \circledast \neg\neg\chi)$	Def. \circledast , 1											
3	$\neg(\varphi \circledast \psi \wedge \chi) \Rightarrow \neg(\varphi \circledast \psi) \vee \neg(\varphi \circledast \chi)$	Replacement of equivalents, 2											
4	$(\varphi \circledast \psi) \wedge (\varphi \circledast \chi) \Rightarrow (\varphi \circledast \psi \wedge \chi)$	PC, 3											

As a sanity check, we show that the proof system $\mathcal{H}_C(*, \circledast)$ is sound with respect to $\mathsf{SL}(*, \circledast)$. The proof does not pose any specific difficulty (as usual with most soundness proofs) but this is the opportunity for the reader to further familiarise with the axioms and rules from $\mathcal{H}_C(*, \circledast)$.

Lemma 6.5. $\mathcal{H}_C(*, \circledast)$ is sound.

Proof. We omit the proof of validity of the axioms of propositional calculus, as well as the admissibility of the rule of modus ponens (MP) . Similarly, the validity of the axioms $(*)_{\text{COM}}$, $(*)_{\text{ASSOC}}$ and $(*)_{\text{ID}}$ and the admissibility of the three rules $(*)$, (\circledast_1) and (\circledast_2) are inherited from BBI (see [75]). The validity of the axioms $(\overline{\text{id}})$, $(\overline{\text{sub}})$, $(\overleftarrow{\text{weak}})$, $(\overrightarrow{\text{func}})$ and $(\overline{\text{alloc}})$ is straightforward.

Validity of the axiom $(*)_{\text{ALIAS}}$.

Let us show that $(x \hookrightarrow _ * x \hookrightarrow _)$ is not satisfiable. *Ad absurdum*, suppose there is a memory state (s, h) such that $(s, h) \models (x \hookrightarrow _ * x \hookrightarrow _)$. By definition of \models , there are h_1, h_2 such that $h_1 \perp h_2$, $(h_1 + h_2) = h$, $(s, h_1) \models x \hookrightarrow _$ and $(s, h_2) \models x \hookrightarrow _$. Thus, $s(x) \in \text{dom}(h_1)$ and $s(x) \in \text{dom}(h_2)$, which leads to a contradiction with $h_1 \perp h_2$.

1	$\top * (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1) \Rightarrow (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1) * \top$	($_{\text{COM}}^*$)
2	$\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 \Rightarrow \mathbf{size} \geq 1$	($\wedge E_L$)
3	$\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \top \Rightarrow \mathbf{size} \geq 1 * \top$	($*$), 2
4	$\mathbf{size} \geq 1 * \top \Rightarrow \mathbf{size} \geq 1$	($_{\text{MONO}}^*$) ($\mathbf{size} \geq 1 \stackrel{\text{def}}{=} \neg \mathbf{emp}$)
5	$\top * (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1) \Rightarrow \mathbf{size} \geq 1$	($\Rightarrow \text{TR}$) twice, 1, 3, 4
6	$\top \Rightarrow (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \mathbf{size} \geq 1)$	($*_2$), 5
7	$\mathbf{emp} \Rightarrow (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 2)$	See Example 6.2
8	$(\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 2) \Rightarrow \top \wedge (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 2)$	($\wedge \text{TI}_L$)
9	$\top \wedge (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 2) \Rightarrow$ $((\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 1) \wedge (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 2))$	($\wedge I_L$), 6
10	$((\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 1) \wedge (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 2)) \Rightarrow$ $(\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \mathbf{size} = 1)$	($* \wedge D_L$) + Def. \mathbf{size}
11	$(\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \neg \mathbf{size} \geq 2) \Rightarrow (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \mathbf{size} = 1)$	($\Rightarrow \text{TR}$) twice, 8, 9, 10
12	$\mathbf{emp} \Rightarrow (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \mathbf{size} = 1)$	($\Rightarrow \text{TR}$), 7, 11

(recall that $\mathbf{size} = \beta$ is a shortcut for $\mathbf{size} \geq \beta \wedge \neg \mathbf{size} \geq \beta + 1$)

Figure 6.4: A proof of $\mathbf{emp} \Rightarrow (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1 * \mathbf{size} = 1)$.

Validity of the axiom $(_{\text{MONO}}^*)$.

The proof of the validity of every instantiation of $(_{\text{MONO}}^*)$ is similar (and quite easy), therefore we show just the case with $\mathbf{x} \hookrightarrow \mathbf{y} * \top \Rightarrow \mathbf{x} \hookrightarrow \mathbf{y}$. Suppose $(s, h) \models \mathbf{x} \hookrightarrow \mathbf{y} * \top$. Then, there is a subheap $h_1 \subseteq h$ such that $(s, h_1) \models \mathbf{x} \hookrightarrow \mathbf{y}$. Hence, $h_1(s(\mathbf{x})) = s(\mathbf{y})$. As $h_1 \subseteq h$, we obtain $h(s(\mathbf{x})) = s(\mathbf{y})$, which by definition implies $(s, h) \models \mathbf{x} \hookrightarrow \mathbf{y}$.

Validity of the axiom $(_{\text{ALLOC}}^*)$.

Suppose $(s, h) \models \neg \mathbf{x} \hookrightarrow _ * \neg \mathbf{x} \hookrightarrow _$. Then, there are two disjoint heaps h_1, h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \neg \mathbf{x} \hookrightarrow _$ and $(s, h_2) \models \neg \mathbf{x} \hookrightarrow _$. Then $s(\mathbf{x}) \notin \text{dom}(h_1)$ and $s(\mathbf{x}) \notin \text{dom}(h_2)$. Since $h = h_1 + h_2$, $\text{dom}(h) = \text{dom}(h_1) \cup \text{dom}(h_2)$ and therefore $s(\mathbf{x}) \notin \text{dom}(h)$. We conclude that $(s, h) \models \neg \mathbf{x} \hookrightarrow _$.

Validity of the axiom $(_{\text{PTO}}^*)$.

Suppose $(s, h) \models (\mathbf{x} \hookrightarrow _ \wedge \neg \mathbf{x} \hookrightarrow \mathbf{y}) * \top$. Then there is a subheap $h_1 \subseteq h$ such that $(s, h_1) \models \mathbf{x} \hookrightarrow _ \wedge \neg \mathbf{x} \hookrightarrow \mathbf{y}$. Hence, $s(\mathbf{x}) \in \text{dom}(h_1)$ and $h_1(s(\mathbf{x})) \neq s(\mathbf{y})$. As $h_1 \subseteq h$, we obtain $s(\mathbf{x}) \in \text{dom}(h)$ and $h(s(\mathbf{x})) \neq s(\mathbf{y})$ which by definition implies $(s, h) \models \neg \mathbf{x} \hookrightarrow \mathbf{y}$.

Validity of the axiom $(_{\text{ATOM}}^{*2})$.

Suppose $(s, h) \models \mathbf{x} \hookrightarrow _$. Then $s(\mathbf{x}) \in \text{dom}(h)$. Let $h_1 \stackrel{\text{def}}{=} \{s(\mathbf{x}) \mapsto h(s(\mathbf{x}))\}$. Trivially, $h_1 \subseteq h$ and $(s, h_1) \models \mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1$. We define h_2 as the unique heap such that $h_2 + h_1 = h$. Trivially, $(s, h_2) \models \top$. Hence, $(s, h) \models (\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1) * \top$.

The proof for the validity of the axiom $(_{\text{ATOM}}^1)$ is similar, and therefore omitted herein.

Validity of the axiom $(\neg \text{size}^*)$.

Let $\beta_1, \beta_2 \geq 0$. Suppose $(s, h) \models \neg \text{size} \geq \beta_1 * \neg \text{size} \geq \beta_2$. Since $\neg \text{size} \geq 0$ is not satisfiable, this implies that necessarily $\beta_1, \beta_2 \geq 1$. Hence, the axiom $(\neg \text{size}^*)$ is trivially valid when $\beta_1 = 0$ or $\beta_2 = 0$. In the sequel, $\beta_1, \beta_2 \geq 1$. Then, there are two disjoint heaps h_1, h_2 such that $h_1 + h_2 = h$, $(s, h_1) \models \neg \text{size} \geq \beta_1$ and $(s, h_2) \models \neg \text{size} \geq \beta_2$. By definition of size , $\text{card}(h_1) \leq \beta_1 - 1$ and $\text{card}(h_2) \leq \beta_2 - 1$. Since $\text{dom}(h) = \text{dom}(h_1) \cup \text{dom}(h_2)$, we obtain $\text{card}(h) \leq \beta_1 + \beta_2 - 2$, which implies $(s, h) \models \neg \text{size} \geq \beta_1 + \beta_2 - 1$.

Validity of the axiom $(\neg \infty^*)$.

Let $X \subseteq_{\text{fin}} \text{VAR}$ and (s, h) be a memory state. Let h_1 be a heap of size one such that $h_1(\ell) = \ell$ for some $\ell \notin \text{dom}(h) \cup s(X)$. We write $s(X)$ to denote the set $\{s(x) \mid x \in X\}$. Trivially $(s, h_1) \models \text{size} = 1 \wedge \bigwedge_{x \in X} \neg x \hookrightarrow __$. Moreover $h_1 \perp h$ holds, hence $h_1 + h_2$ is defined and $(s, h + h_1) \models \top$. Then, $(s, h) \models (\text{size} = 1 \wedge \bigwedge_{x \in X} \neg x \hookrightarrow __) \multimap \top$.

Validity of the axiom (\rightarrow^*) .

Suppose $(s, h) \models \neg x \hookrightarrow __$. Let h_1 be the heap of size one such that $h_1(s(x)) = s(y)$. Trivially, $(s, h_1) \models x \hookrightarrow y \wedge \text{size} = 1$. Moreover, as $s(x) \notin \text{dom}(h)$, $h_1 \perp h$ holds, hence $h_1 + h$ is defined and $(s, h + h_1) \models \top$. Then, $(s, h) \models (x \hookrightarrow y \wedge \text{size} = 1) \multimap \top$.

Validity of the axiom (alloc^*) .

Let $X \subseteq_{\text{fin}} \text{VAR}$ and suppose $(s, h) \models \neg x \hookrightarrow __$. Let h_1 be the heap of size one such that $h_1(s(x)) = \ell$ where $\ell \notin s(X)$. Trivially, $(s, h_1) \models x \hookrightarrow __ \wedge \text{size} = 1 \wedge \bigwedge_{y \in X} \neg x \hookrightarrow y$. Moreover, as $s(x) \notin \text{dom}(h)$, $h_1 \perp h$ holds, hence $h + h_1$ is defined and $(s, h + h_1) \models \top$. Then, $(s, h) \models (x \hookrightarrow __ \wedge \text{size} = 1 \wedge \bigwedge_{y \in X} \neg x \hookrightarrow y) \multimap \top$. \square

6.3 MAIN INGREDIENTS OF THE METHOD

Before showing completeness of $\mathcal{H}_C(*, -*)$, let us recall the key ingredients of the method we follow, not only to provide a vade mecum for axiomatising other separation logics, but also to identify the essential features and where variations are still possible. The Hilbert-style axiomatisation of $\text{SL}(*, -*)$ shall culminate with Theorem 6.22 that states the adequateness of the proof system $\mathcal{H}_C(*, -*)$.

In order to axiomatise $\text{SL}(*, -*)$ internally, as already emphasised several times, the core formulae play an essential role. The main properties of these formulae is that their Boolean combinations capture the full logic $\text{SL}(*, -*)$ [103] and (in order to have a Hilbert-style axiomatisation) all the core formulae can be expressed in $\text{SL}(*, -*)$. Generally speaking, our axiom system naturally leads to a form of constructive completeness, as advocated in [61, 105]: the axiomatisation provides proof-theoretical means to transform any formula into an equivalent Boolean combination of core formulae, and it contains also a part dedicated to the derivation of valid Boolean combinations of core formulae (understood as a syntactical fragment of $\text{SL}(*, -*)$). What is specific to each logic is the design of the set of core formulae and in the case of $\text{SL}(*, -*)$, this was already known since [103].

Derivations in the proof system $\mathcal{H}_C(*, -*)$ shall simulate the bottom-up elimination of separating connectives (see forthcoming Lemmata 6.15 and 6.18) when the arguments are two Boolean combinations of core formulae. This bottom-up elimination corresponds to a syntactical proof of the $*$ -simulation and $-*$ -simulation properties described throughout Chapter 5. To achieve this, $\mathcal{H}_C(*, -*)$ contains axiom schemas that perform such an elimination in multiple “small-step” derivations, e.g. by deriving a single $x \hookrightarrow y$ predicate from the formula $x \hookrightarrow y * \top$ (see

Propositional Calculus:

$$(L_1) \quad (\neg\varphi \Rightarrow \varphi) \Rightarrow \varphi$$

$$(L_2) \quad \varphi \Rightarrow (\neg\varphi \Rightarrow \psi)$$

$$(L_3) \quad (\varphi \Rightarrow \psi) \Rightarrow ((\psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \chi))$$

$$(MP) \quad \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$$

Axioms of the core formulae:

$$(\overline{=}) \quad x = x$$

$$(\xrightarrow{\text{WEAK}}) \quad x \xrightarrow{} y \Rightarrow x \xleftrightarrow{} _$$

$$(\overline{=}_{\text{SUB}}) \quad \varphi \wedge x = y \Rightarrow \varphi[y \leftarrow x]$$

$$(\xrightarrow{\text{FUNC}}) \quad x \xrightarrow{} y \wedge x \xrightarrow{} z \Rightarrow y = z$$

$$(\text{ALLOC}_{\text{SIZE}}) \quad x \xrightarrow{} _ \wedge y \xrightarrow{} _ \wedge x \neq y \Rightarrow \text{size} \geq 2$$

Intermediate axioms:

$$(I_1^C) \quad \text{size} \geq \beta + 1 \Rightarrow \text{size} \geq \beta$$

$$(I_2^C) \quad \bigwedge_{x \in X} (x \xrightarrow{} _ \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \Rightarrow \text{size} \geq \text{card}(X) \quad \blacktriangleleft [X \subseteq_{\text{fin}} \text{VAR}]$$

Figure 6.5: Proof system \mathcal{H}_C for Boolean combinations of core formulae.

the axiom (MONO^*) , in the case where $e = x \xrightarrow{} y$). Alternatively, it would have been possible to include “big-step” axiom schemas that, given two Boolean combinations of core formulae, derive the equivalent formula in one single derivation step (see e.g. [63]). To show the completeness of $\mathcal{H}_C(*, -*)$ we still derive these big-step formulae as theorems of the proof system, as we will see in Lemma 6.14 and Lemma 6.18. The main difference between small-step and big-step axioms is that the former provide a simpler understanding of the key properties of the logic.

6.4 A SIMPLE CALCULUS FOR THE CORE FORMULAE

To axiomatise $\text{SL}(*, -*)$, we start by introducing the proof system \mathcal{H}_C dedicated to Boolean combinations of core formulae, defined in Figure 6.5. It contains all the axioms and the rule of modus ponens from propositional calculus, together with the axioms of the core formulae $(\overline{=})$ – $(\xrightarrow{\text{FUNC}})$ that belongs to $\mathcal{H}_C(*, -*)$. Lastly, it features two intermediate axioms (I_1^C) and (I_2^C) which, as explained before, are needed to establish results about the axiomatisation of Boolean combinations of core formulae, but become derivable as soon as the multiplicative connectives are considered (and thus do not belong to $\mathcal{H}_C(*, -*)$).

Let us look at the axioms of \mathcal{H}_C more closely. The axiom schema $(\overline{=}_{\text{SUB}})$ states that whenever $x = y$ holds, then y can be replaced by x in an arbitrary formula φ . In its essence, this axiom, together with $(\overline{=})$, states that the equality between location forms an equivalence relation. Indeed, the formula $y = z \wedge x = y \Rightarrow x = z$ expressing that the equality between locations is a transitive relation corresponds to the instance of $(\overline{=}_{\text{SUB}})$ where $\varphi = y = z$, whereas its symmetry (i.e. $x = y \Rightarrow y = x$) is derived as follows:

1	$\neg y = x \wedge x = y \Rightarrow \neg x = x$	$(\overline{\text{SUB}})$, where $\varphi = \neg y = x$
2	$\neg y = x \wedge x = y \Rightarrow x = x$	PC, $(\overline{\text{ID}})$
3	$\neg y = x \wedge x = y \Rightarrow \perp$	PC, 1, 2
4	$x = y \Rightarrow y = x$	PC, 3

Let (s, h) be a memory state. The axiom $(\overline{\text{SUB}})$ subsumes several other easy properties of memory states. For instance, when instantiated with $\varphi = x \hookrightarrow y$, it leads to the formula $x \hookrightarrow y \wedge x = y \Rightarrow x \hookrightarrow x$ stating that if $h(s(x)) = s(y)$ and $s(x) = s(y)$ hold, then clearly the location $s(x)$ points to itself. The axiom $(\overline{\text{WEAK}})$ states that $h(s(x)) = s(y)$ implies $s(x) \in \text{dom}(h)$, whereas the axiom $(\overline{\text{FUNC}})$ characterises the fact that h is a (partial) function. The axiom $(\overline{\text{ALLOC}})$ states that if $s(x) \neq s(y)$ and $\{s(x), s(y)\} \subseteq \text{dom}(h)$, then $\text{card}(h) \geq 2$. Notably, this axiom is captured by the intermediate axiom (I_2^C) , which extends it arbitrary (finite) sets of variables. Formally, (I_2^C) states that given a finite set $X = \{x_1, \dots, x_n\}$ of program variables, if for every distinct $i, j \in [1, n]$, $s(x_i) \neq s(x_j)$ and $s(X) \subseteq \text{dom}(h)$, then $\text{card}(h) \geq n$. Lastly, the axiom (I_1^C) states that if $\text{dom}(h)$ contains at least $\beta + 1$ locations, then it has at least β locations.

Clearly, the two intermediate axiom schemata (I_1^C) and (I_2^C) are valid, which leads to the soundness of \mathcal{H}_C directly from Lemma 6.5.

Lemma 6.6. \mathcal{H}_C is sound.

In order to establish its completeness with respect to Boolean combinations of core formulae, we first show that \mathcal{H}_C is complete for a subclass of Boolean combinations of core formulae, namely for *core types* (defined below). Subsequently, we show that every formula in $\text{Bool}(\text{Core}(X, \alpha))$ is provably equivalent to a disjunction of core types (Lemma 6.10).

Definition 6.7 (Core types of $\text{SL}(*, *)$). Let $X \subseteq_{\text{fin}} \text{VAR}$ and consider a natural number $\alpha \geq 1$. We write $\text{CoreTypes}(X, \alpha)$ to denote the following set of *core types*:

$$\{\varphi \in \text{Conj}(\text{Core}(X, \alpha)) \mid \text{for all } \psi \in \text{Core}(X, \alpha), \{\psi \mid \neg\psi\} \subseteq_{\text{LIT}} \varphi, \text{ and } (\psi \wedge \neg\psi) \not\subseteq_{\text{LIT}} \varphi\}.$$

Essentially, every core type $\varphi \in \text{CoreTypes}(X, \alpha)$ is a conjunction from $\text{Conj}(\text{Core}(X, \alpha))$ such that for every core formula $\psi \in \text{Core}(X, \alpha)$, exactly one literal among ψ and $\neg\psi$ belongs to φ . The following refutational result shows that \mathcal{H}_C is (sound and) complete for core types.

Lemma 6.8. Let $\varphi \in \text{CoreTypes}(X, \alpha)$ with $\alpha \geq \text{card}(X)$. We have $\neg\varphi$ is valid iff $\vdash_{\mathcal{H}_C} \neg\varphi$.

Notably, among the proofs of completeness of \mathcal{H}_C , $\mathcal{H}_C(*)$ and $\mathcal{H}_C(*, *)$, the proof of this lemma is the only one that heavily relies on semantical means. All the other proofs of this chapter (soundness results excluded) are almost exclusively syntactical.

Proof. The right-to-left direction follows from the soundness of \mathcal{H}_C (Lemma 6.6), so we focus on the left-to-right direction. By contrapositive, let $\varphi \in \text{CoreTypes}(X, \alpha)$, with $\alpha \geq \text{card}(X)$ be such that $\nvdash_{\mathcal{H}_C} \varphi \Rightarrow \perp$. We prove that φ is satisfiable (and thus $\neg\varphi$ is not valid). During the proof, we write “**By** (statement₁) **it holds that** (statement₂)” as a shortcut for

“if (statement₂) does not hold then $\vdash_{\mathcal{H}_C} \varphi \Rightarrow \perp$ by (statement₁), a contradiction”.

For example, “**By axiom $(\overline{\text{ID}})$ it holds that** for every $x \in X$, $x = x \subseteq_{\text{LIT}} \varphi$ ” is a shortcut for “if (for every $x \in X$, $x = x \subseteq_{\text{LIT}} \varphi$) does not hold then $\vdash_{\mathcal{H}_C} \varphi \Rightarrow \perp$ by (axiom $(\overline{\text{ID}})$), a contradiction”. To be completely precise, we should often write in (statement₁) that propositional

reasoning is required (as in the example above). However, we decide to omit the use of propositional calculus whenever it is clearly used in a trivial way. In this way, we hope to improve the readability of the proof, which is mainly composed of a series of results shown by contradiction.

Let us start by considering the binary relation $\approx \subseteq X \times X$ defined as follows:

$$\approx \stackrel{\text{def}}{=} \{(x, y) \mid x = y \subseteq_{\text{LIT}} \varphi\}.$$

Since φ is a core type, for every $(x, y) \in X^2$, if $(x, y) \notin \approx$ then $(\neg x = y) \subseteq_{\text{LIT}} \varphi$. Previously (page 291), we have shown that thanks to $(\overline{\text{id}})$ and $(\overline{\text{sub}})$, the two following formulae (schemata, to be more precise) are derivable in \mathcal{H}_C :

$$\begin{aligned} & (\text{symmetry}) \quad x = y \Rightarrow y = x, \\ & (\text{transitivity}) \quad y = z \wedge x = y \Rightarrow x = z. \end{aligned}$$

Therefore,

By axioms $(\overline{\text{id}})$ and $(\overline{\text{sub}})$ it holds that \approx is an equivalence relation ■

Below, we write $[x]$ to denote the equivalence class of x with respect to \approx . Let us now consider the binary relation $f \subseteq (X/\approx) \times (X/\approx)$ (where X/\approx is the quotient set of X by \approx) below:

$$f \stackrel{\text{def}}{=} \{([x], [y]) \mid x \hookrightarrow y \subseteq_{\text{LIT}} \varphi\}$$

By instantiating $(\overline{\text{sub}})$ with $\varphi = y \hookrightarrow z$ and $\varphi = z \hookrightarrow y$, we obtain respectively:

$$\begin{aligned} & (\text{eq-cell-left}) \quad y \hookrightarrow z \wedge x = y \Rightarrow x \hookrightarrow z, \\ & (\text{eq-cell-right}) \quad z \hookrightarrow y \wedge x = y \Rightarrow z \hookrightarrow x. \end{aligned}$$

Therefore, by recalling that \mathcal{H}_C features the axiom $x \hookrightarrow y \wedge x \hookrightarrow z \Rightarrow y = z$ (i.e. $(\overleftarrow{\text{func}})$),

By axioms $(\overleftarrow{\text{func}})$ and $(\overline{\text{sub}})$ it holds that f is a partial map from X/\approx to X/\approx ■

We now consider the sets $A \stackrel{\text{def}}{=} \{[x] \mid x \hookrightarrow - \subseteq_{\text{LIT}} \varphi\}$.

By axiom $(\overleftarrow{\text{weak}})$ and def. of f it holds that $\text{dom}(f) \subseteq A$ ■

By axiom (I_2^C) and def. of A it holds that $\vdash_{\mathcal{H}_C} \varphi \Rightarrow \text{size} \geq \text{card}(A)$ ■

By $\vdash_{\mathcal{H}_C} \varphi \Rightarrow \text{size} \geq \text{card}(A)$ **and** $\alpha \geq \text{card}(X)$ **it holds that** $\text{size} \geq \text{card}(A) \subseteq_{\text{LIT}} \varphi$ ■

Let $n = \max_{\text{size}}(\varphi)$. We recall that, by definition of $\max_{\text{size}}(\varphi)$, n is the greatest $\beta \in \mathbb{N}$ such that $\text{size} \geq \beta$ occurs positively in φ . As $\text{size} \geq \text{card}(A) \subseteq_{\text{LIT}} \varphi$, we have $n \geq \text{card}(A)$.

From \approx , f , A and n , we now define a memory state (s, h) , and show that $(s, h) \models \varphi$. We fix an enumeration for the equivalence classes in X/\approx , i.e. a bijection $g : (X/\approx) \rightarrow [1, \text{card}(X/\approx)]$. We introduce $\text{card}(X/\approx) + (n - \text{card}(A)) + 1$ distinct locations denoted as follows:

$$\ell_1, \dots, \ell_{\text{card}(X/\approx)}, \ell'_1, \dots, \ell'_{n - \text{card}(A)}, \hat{\ell}.$$

Let s be a store such that, for every $x \in X$, $s(x) \stackrel{\text{def}}{=} \ell_{g([x])}$. Since \approx is an equivalence relation, we conclude that s is well-defined, i.e. $s : \text{VAR} \rightarrow \text{LOC}$. Let h be the heap defined as follows:

$$h(\ell) \stackrel{\text{def}}{=} \begin{cases} \ell_{g(f(C))} & \text{if there is } C \in A \text{ s.t. } \ell = \ell_{g(C)} \text{ and } C \in \text{dom}(f) \\ \hat{\ell} & \text{if there is } C \in A \text{ s.t. } \ell = \ell_{g(C)} \text{ and } C \notin \text{dom}(f) \\ \hat{\ell} & \text{if there is } i \in [1, n - \text{card}(A)] \text{ s.t. } \ell = \ell'_i \\ \text{undefined} & \text{otherwise} \end{cases}$$

Since f is a partial map from X/\approx to X/\approx , h is well-defined, i.e. $h : \text{LOC} \rightarrow_{\text{fin}} \text{LOC}$. Notice that, by definition, $\text{card}(h) = \text{card}(A) + (n - \text{card}(A)) = n$. We show that the memory state (s, h) satisfies φ . Let L be a literal in $\text{LIT}(\varphi)$, we prove that $(s, h) \models L$ by cases on the shape of L .

case: $L = x = y$. By definition of \approx , $x \approx y$. By definition of s , $s(x) = s(y)$. So, $(s, h) \models x = y$.

case: $L = \neg(x = y)$. *Ad absurdum*, suppose $s(x) = s(y)$. By definition of s , this implies that $[x] = [y]$ and thus $x \approx y$. By definition of \approx , we conclude that $x = y \subseteq_{\text{LIT}} \varphi$. However, this allows us to derive that $x = y \wedge \neg x = y \subseteq_{\text{LIT}} \varphi$, in contradiction with the fact that φ is a core type. Therefore, $s(x) \neq s(y)$. We conclude that $(s, h) \models \neg(x = y)$.

case: $L = x \hookrightarrow y$. By definition of f , $f([x]) = [y]$. By definition of h , $h(\ell_g([x])) = \ell_g([y])$. By definition of s , $s(x) = \ell_g([x])$ and $s(y) = \ell_g([y])$. Thus, $(s, h) \models x \hookrightarrow y$.

case: $L = \neg(x \hookrightarrow y)$. *Ad absurdum*, suppose $h(s(x)) = s(y)$. By definition of s , $s(x) = \ell_g([x])$ and $s(y) = \ell_g([y])$. Since $\ell_g([y]) \neq \hat{\ell}$, by definition of h we conclude that $f([x]) = [y]$. By definition of f , there are $z, v \in X$ such that $[x] = [z]$, $[y] = [v]$ and $z \hookrightarrow v \subseteq_{\text{LIT}} \varphi$. By definition of \approx , which we have shown to be an equivalence relation, $x = z \wedge y = v \subseteq_{\text{LIT}} \varphi$. From the theorems (eq-cell-left) and (eq-cell-right) (under suitable substitutions for the metavariables x , y and z of these two theorems), we conclude that $\vdash_{\mathcal{H}_C} \varphi \Rightarrow x \hookrightarrow y$. As φ is a core type, this implies that $x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$. However, this allows us to derive that $x \hookrightarrow y \wedge \neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$, in contradiction with the fact that φ is a core type. Therefore, $h(s(x)) \neq s(y)$. We conclude that $(s, h) \models \neg(x \hookrightarrow y)$.

case: $L = x \hookrightarrow _$. By definition of A , $[x] \in A$. By definition of h , $\ell_g([x]) \in \text{dom}(h)$. By definition of s , $s(x) = \ell_g([x])$. Thus, $(s, h) \models x \hookrightarrow _$.

case: $L = \neg(x \hookrightarrow _)$. *Ad absurdum*, suppose $s(x) \in \text{dom}(h)$. By definition of s , $s(x) = \ell_g([x])$. Since, for every $i \in [1, n - \text{card}(A)]$, $\ell_g([x]) \neq \ell'_i$, by definition of h we conclude that $\ell_g([x]) \in A$. By definition of A , there is $y \in X$ such that $[x] = [y]$ and $y \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$. By definition of \approx , which we have shown to be an equivalence relation, $x = y \subseteq_{\text{LIT}} \varphi$. By instantiating (SUB) with $\varphi = y \hookrightarrow _$ we conclude that the formula below is valid:

$$(\text{eq-alloc}) \quad y \hookrightarrow _ \wedge x = y \Rightarrow x \hookrightarrow _.$$

Thus, $\vdash_{\mathcal{H}_C} \varphi \Rightarrow x \hookrightarrow _$. As φ is a core type, this implies that $x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$. However, this allows us to derive that $x \hookrightarrow _ \wedge \neg x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$, in contradiction with the fact that φ is a core type. Therefore, $s(x) \notin \text{dom}(h)$. We conclude that $(s, h) \models \neg(x \hookrightarrow _)$.

case: $L = \text{size} \geq \beta$. By definition of $\text{max}_{\text{size}}(\varphi)$, $\beta \leq \text{max}_{\text{size}}(\varphi) = n$. Since $\text{card}(h) = n$, we have $\text{card}(h) \geq \beta$. We conclude that $(s, h) \models \text{size} \geq \beta$.

case: $L = \neg(\text{size} \geq \beta)$. First of all, let us show the following result:

(†) for every $\beta_1, \beta_2 \in \mathbb{N}$ such that $\beta_1 \leq \beta_2$, if $\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \varphi$ then $\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$.

The proof is by induction on the difference $\beta_2 - \beta_1$.

base case: $\beta_2 - \beta_1 = 0$. Straightforward.

induction case: $\beta_2 - \beta_1 \geq 1$. By $\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \varphi$ and (I₁^C), $\vdash_{\mathcal{H}_C} \varphi \Rightarrow \text{size} \geq \beta_2 - 1$.

As φ is a core type, this implies that $\text{size} \geq \beta_2 - 1 \subseteq_{\text{LIT}} \varphi$. As $(\beta_2 - 1) - \beta_1 < \beta_2 - \beta_1$, by induction hypothesis $\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$.

Now, *ad absurdum*, suppose $\beta \leq n$. As $n = \text{max}_{\text{size}}(\varphi)$, $\text{size} \geq n \subseteq_{\text{LIT}} \varphi$. From (†), $\text{size} \geq \beta \subseteq_{\text{LIT}} \varphi$. However, this allows us to derive that $\text{size} \geq \beta \wedge \neg \text{size} \geq \beta \subseteq_{\text{LIT}} \varphi$, in contradiction with the fact that φ is a core type. Therefore, $\beta > n$. By definition of h , $\text{card}(h) = n$. We conclude that $(s, h) \models \neg(\text{size} \geq \beta)$. \square

By classical reasoning, one can show that every $\varphi \in \text{Bool}(\text{Core}(X, \alpha))$ is provably equivalent to a disjunction of core types. Together with Lemma 6.8, this implies that \mathcal{H}_C is adequate with respect to the propositional logic of core formulae.

Theorem 6.9 (Adequacy). A Boolean combination of core formulae φ is valid iff $\vdash_{\mathcal{H}_C} \varphi$.

In order to prove Theorem 6.9, let us first establish the following simple lemma.

Lemma 6.10. Let $\varphi \in \text{Bool}(\text{Core}(X, \alpha))$. There is a disjunction $\psi = \psi_1 \vee \dots \vee \psi_n$ such that $\vdash_{\mathcal{H}_C} \varphi \Leftrightarrow \psi$ and for every $i \in [1, n]$, $\psi_i \in \text{CoreTypes}(X, \max(\text{card}(X), \alpha))$.

Proof. Let $\psi_1 \vee \dots \vee \psi_n$ be a formula in disjunctive normal form logically equivalent to φ . Since \mathcal{H}_C extends an adequate system for propositional calculus, $\vdash_{\mathcal{H}_C} \varphi \Leftrightarrow \psi_1 \vee \dots \vee \psi_n$. To conclude the proof, given $i \in [1, n]$, we show that $\vdash_{\mathcal{H}_C} \psi_i \Leftrightarrow \psi'_i$ where ψ'_i is a disjunction of core types from $\text{CoreTypes}(X, \max(\text{card}(X), \alpha))$. The proof is by induction on the number k of core formulae χ in $\text{Core}(X, \max(\text{card}(X), \alpha))$ such that neither χ nor $\neg\chi$ belongs to $\text{LIT}(\psi_i)$.

base case: $k = 0$. In this case, if $\psi_i \in \text{CoreTypes}(X, \max(\text{card}(X), \alpha))$ then we conclude. Else, since every core formula of $\text{Core}(X, \max(\text{card}(X), \alpha))$ belongs to $\text{LIT}(\psi_i)$, we deduce that $\text{LIT}(\psi_i)$ contains a core formula and its negation. Thus, ψ_i is unsatisfiable. Let ψ'_i be a core type in $\text{CoreTypes}(X, \max(\text{card}(X), \alpha))$ such that $\text{size} \geq 0$ belongs to $\text{LIT}(\psi'_i)$. ψ'_i is clearly unsatisfiable. By propositional reasoning, $\vdash_{\mathcal{H}_C} \psi_i \Leftrightarrow \psi'_i$.

induction step: $k \geq 1$. Let $\chi \in \text{Core}(X, \max(\text{card}(X), \alpha))$ be a core formula that does not occur neither positively nor negatively in ψ_i . By propositional reasoning, we deduce that $\vdash_{\mathcal{H}_C} \psi_i \Leftrightarrow (\psi_i \wedge \chi) \vee (\psi_i \wedge \neg\chi)$. By induction hypothesis, there are two disjunctions ψ'_i and ψ''_i of core types from $\text{CoreTypes}(X, \max(\text{card}(X), \alpha))$ such that $\vdash_{\mathcal{H}_C} \psi_i \wedge \chi \Leftrightarrow \psi'_i$ and $\vdash_{\mathcal{H}_C} \psi_i \wedge \neg\chi \Leftrightarrow \psi''_i$. By propositional reasoning, $\vdash_{\mathcal{H}_C} \psi_i \Leftrightarrow \psi'_i \vee \psi''_i$. \square

Proof of Theorem 6.9. Let φ be a Boolean combination of core formulae in $\text{CoreTypes}(X, \alpha)$.

(\Leftarrow): Directly from Lemma 6.6.

(\Rightarrow): Let us assume that φ is valid, and let us prove that $\vdash_{\mathcal{H}_C} \varphi$. By Lemma 6.10, there is a disjunction $\psi = \psi_1 \vee \dots \vee \psi_n$ of core types in $\text{CoreTypes}(X, \max(\text{card}(X), \alpha))$ such that $\vdash_{\mathcal{H}_C} (\neg\varphi) \Leftrightarrow \psi$. As φ is valid, the formulae $\neg\varphi$, ψ and all the ψ_i 's are unsatisfiable. Directly from Lemma 6.8, $\vdash_{\mathcal{H}_C} \psi_i \Rightarrow \perp$, for all i . By propositional reasoning, $\vdash_{\mathcal{H}_C} \varphi$. \square

6.5 SYNTACTICAL ELIMINATION OF THE SEPARATING CONJUNCTION

We define an Hilbert-style axiomatisation for $\text{SL}(*, x \hookrightarrow _)$, the logic obtained from $\text{SL}(*, *)$ by replacing the separating implication with formulae of the form $x \hookrightarrow _$. Fundamentally, within $\text{SL}(*, x \hookrightarrow _)$, the core formula $\text{size} \geq \beta$ can be encoded in the logic, according to its definition given in Figure 6.1. The proof system for $\text{SL}(*, x \hookrightarrow _)$, denoted by $\mathcal{H}_C(*)$, is defined in Figure 6.6. As we can see, $\mathcal{H}_C(*)$ is obtained by enriching \mathcal{H}_C with the axioms of Figure 6.2 that handle the separating conjunction, together with the inference rule (*). Moreover, it features three intermediate axioms, (I_3^*) , (I_4^*) and (I_5^*) .

Let us look further at the axioms in Figure 6.6. As already stated when introducing $\mathcal{H}_C(*, *)$, the axioms $(*)$ – (ASSOC) deal with the properties of non-deterministic monoid introduced for BBI in Section 2.3.3. The rule (*), again from BBI, entails that logical equivalence is a congruence for *. The axiom (ALIAS) states that it is not possible to split a single memory cell corresponding to the variable x between disjoint subheaps. In its essence, the axiom (ATOM^1) states that each memory cell of a heap can be separated from the others. The axiom (ATOM^2) is similar in nature, but specific to memory cells corresponding to program variables. The axiom (MONO) deals with monotonic properties that, whenever true in a subheap, are also verified in the original heap.

Axioms and the rule of modus ponens from \mathcal{H}_C (Figure 6.5).

Axioms of the separating conjunction:

$(\text{ID}^*) \quad \varphi \Leftrightarrow \varphi * \text{emp}$	$(\text{MONO}^*) \quad e * \top \Rightarrow e \quad \blacktriangleleft [e \in \{\neg \text{emp}, x = y, x \neq y, x \hookrightarrow y\}]$
$(\text{ASSOC}^*) \quad (\varphi * \psi) * \chi \Leftrightarrow \varphi * (\psi * \chi)$	$(\text{ALLOC}^*) \quad \neg x \hookrightarrow _ * \neg x \hookrightarrow _ \Rightarrow \neg x \hookrightarrow _$
$(\text{COM}^*) \quad \varphi * \psi \Rightarrow \psi * \varphi$	$(\text{SIZE}^*) \quad \neg \text{size} \geq \beta_1 * \neg \text{size} \geq \beta_2 \Rightarrow \neg \text{size} \geq \beta_1 + \beta_2 - 1$
$(\text{ALIAS}^*) \quad (x \hookrightarrow _ * x \hookrightarrow _) \Leftrightarrow \perp$	$(\text{PTO}^*) \quad (x \hookrightarrow _ \wedge \neg x \hookrightarrow y) * \top \Rightarrow \neg x \hookrightarrow y$
$(\text{ATOM}^1) \quad \neg \text{emp} \Rightarrow \text{size} = 1 * \top$	$(\text{ATOM}^2) \quad x \hookrightarrow _ \Rightarrow (x \hookrightarrow _ \wedge \text{size} = 1) * \top$

Rule of inference for the separating conjunction:

$$(*) \quad \frac{\varphi \Rightarrow \chi}{\varphi * \psi \Rightarrow \chi * \psi}$$

Intermediate axioms:

$$(I_3^*) \quad (\varphi \vee \psi) * \chi \Rightarrow (\varphi * \chi) \vee (\psi * \chi) \quad (I_4^*) \quad (\perp * \varphi) \Leftrightarrow \perp \quad (I_5^*) \quad x \hookrightarrow _ * \top \Rightarrow x \hookrightarrow _$$

Figure 6.6: The Hilbert-style proof system $\mathcal{H}_C(*)$.

For instance, the instantiation of (MONO^*) with $e = x \hookrightarrow y$ states that if $h'(s(x)) = s(y)$ holds for a memory state (s, h') , and $h' \subseteq h$, then $h(s(x)) = s(y)$. Notice that the intermediate axiom (I_5^*) is similar to (MONO^*) , and treats the monotonicity of $x \hookrightarrow _$. The axiom (ALLOC^*) characterises the fact that whenever two disjoint heaps are composed, if the location corresponding to x was not allocated in neither of the two heaps, then it is not allocated in the resulting heap. The axiom (SIZE^*) is quite similar, but deals with the cardinality of the two heaps. Essentially, it states that whenever we compose two disjoint heaps having sizes less than β_1 and β_2 , respectively, the resulting heap has less than $\beta_1 + \beta_2 - 1$ memory cells. The axiom (PTO^*) states that, given a memory state (s, h) and a heap $h' \subseteq h$, if $s(x) \in \text{dom}(h')$ but $h'(s(x)) \neq s(y)$, then $h(s(x)) \neq s(y)$. Lastly, the two intermediate axioms (I_3^*) and (I_4^*) are quite immediate. The axiom (I_4^*) states that \perp is an annihilator for $*$ (exactly as for the classical conjunction \wedge). The axiom (I_3^*) states that $*$ distributes over disjunctions. Notice that the right-to-left direction of (I_3^*) , i.e. $(\varphi * \chi) \vee (\psi * \chi) \Rightarrow (\varphi \vee \psi) * \chi$, is a theorem of $\mathcal{H}_C(*)$. Indeed,

1	$\varphi \Rightarrow \varphi \vee \psi$	PC	4	$\psi * \chi \Rightarrow (\varphi \vee \psi) * \chi$	$(*)$, 2
2	$\psi \Rightarrow \varphi \vee \psi$	PC	5	$(\varphi * \chi) \vee (\psi * \chi) \Rightarrow (\varphi \vee \psi) * \chi$	PC, 3, 4
3	$\varphi * \chi \Rightarrow (\varphi \vee \psi) * \chi$	$(*)$, 1			

Lemma 6.11. $\mathcal{H}_C(*)$ is sound.

Proof. The validity of all the axioms and the admissibility of all the rules has already been

established in Lemma 6.5, with the exception of the three intermediate axioms. The axioms (I_3^*) and (I_4^*) are valid formulae in BBI (see [75, Section 2]), and thus are also valid in separation logic, following the correspondence given by Proposition 2.23. The soundness of (I_5^*) is straightforward. Indeed, suppose $(s, h) \models x \hookrightarrow _ * \top$. So, there is $h' \subseteq h$ such that $(s, h') \models x \hookrightarrow _$. By definition of $x \hookrightarrow _$, $s(x) \in \text{dom}(h')$. By $h' \subseteq h$, $s(x) \in \text{dom}(h)$. We conclude that $(s, h) \models x \hookrightarrow _$. \square

$\mathcal{H}_C(*)$ is designed with the idea of being as simple as possible. On one side, this helps understanding the key ingredients of $\text{SL}(*, x \hookrightarrow _)$. On the other side, this makes the proof of completeness of $\mathcal{H}_C(*)$ more challenging. To work towards this proof while familiarising with the new axioms, we first show a set of intermediate theorems.

Lemma 6.12. The following formulae are theorem of $\mathcal{H}_C(*)$:

- $(I_{6.12.1}^*) \quad x \sim y \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge x \sim y) * \psi \quad \blacktriangleleft \sim \in \{=, \neq\}$
- $(I_{6.12.2}^*) \quad x = y \wedge ((\varphi \wedge x \hookrightarrow _) * \psi) \Rightarrow (\varphi \wedge y \hookrightarrow _) * \psi$
- $(I_{6.12.3}^*) \quad (\varphi \wedge x \hookrightarrow _) * \psi \Rightarrow \varphi * (\psi \wedge \neg x \hookrightarrow _)$
- $(I_{6.12.4}^*) \quad \neg x \hookrightarrow _ \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge \neg x \hookrightarrow _) * \psi$
- $(I_{6.12.5}^*) \quad x \hookrightarrow _ \wedge (\varphi * (\neg x \hookrightarrow _ \wedge \psi)) \Rightarrow (\varphi \wedge x \hookrightarrow _) * (\neg x \hookrightarrow _ \wedge \psi)$
- $(I_{6.12.6}^*) \quad x \hookrightarrow y \wedge ((\varphi \wedge x \hookrightarrow _) * \psi) \Rightarrow (\varphi \wedge x \hookrightarrow y) * \psi$
- $(I_{6.12.7}^*) \quad \neg x \hookrightarrow y \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge \neg x \hookrightarrow y) * \psi.$

To lighten the chapter from the burden of several syntactical proofs, below we present the proofs of the theorems $(I_{6.12.1}^*)$, $(I_{6.12.3}^*)$ and $(I_{6.12.5}^*)$, leaving the other proofs in Appendix D.

Proof of $(I_{6.12.1}^)$.*

1	$\varphi \Rightarrow (\varphi \wedge x \sim y) \vee (\varphi \wedge \neg x \sim y)$	PC
2	$\varphi * \psi \Rightarrow ((\varphi \wedge x \sim y) \vee (\varphi \wedge \neg x \sim y)) * \psi$	$(*)$, 1
3	$((\varphi \wedge x \sim y) \vee (\varphi \wedge \neg x \sim y)) * \psi \Rightarrow ((\varphi \wedge x \sim y) * \psi) \vee ((\varphi \wedge \neg x \sim y) * \psi)$	(I_3^*)
4	$\varphi \wedge \neg x \sim y \Rightarrow \neg x \sim y$	PC
5	$\psi \Rightarrow \top$	PC
6	$(\varphi \wedge \neg x \sim y) * \psi \Rightarrow (\neg x \sim y) * \top$	$(*I_{LR})$, 4, 5
7	$(\neg x \sim y) * \top \Rightarrow \neg x \sim y$	$(^*\text{MONO})$
8	$(\varphi \wedge \neg x \sim y) * \psi \Rightarrow \neg x \sim y$	$(\Rightarrow \text{TR})$, 6, 7
9	$((\varphi \wedge x \sim y) * \psi) \vee ((\varphi \wedge \neg x \sim y) * \psi) \Rightarrow ((\varphi \wedge x \sim y) * \psi) \vee \neg x \sim y$	8, PC
10	$\varphi * \psi \Rightarrow ((\varphi \wedge x \sim y) * \psi) \vee \neg x \sim y$	$(\Rightarrow \text{TR})$, 2, 3, 9
11	$x \sim y \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge x \sim y) * \psi$	10, PC \square

Proof of $(I_{6.12.3}^)$.*

1	$\psi \Rightarrow (\psi \wedge x \hookrightarrow _) \vee (\psi \wedge \neg x \hookrightarrow _)$	PC
2	$(\varphi \wedge x \hookrightarrow _) * \psi \Rightarrow$	

	$(\varphi \wedge x \hookrightarrow _) * ((\psi \wedge x \hookrightarrow _) \vee (\psi \wedge \neg x \hookrightarrow _))$	$(_{\text{COM}}^*), (*), 1$
3	$(\varphi \wedge x \hookrightarrow _) * ((\psi \wedge x \hookrightarrow _) \vee (\psi \wedge \neg x \hookrightarrow _)) \Rightarrow$ $((\varphi \wedge x \hookrightarrow _) * (\psi \wedge x \hookrightarrow _)) \vee ((\varphi \wedge x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _))$	$(_{\text{COM}}^*), (I_3^*), 2$
4	$\chi \wedge x \hookrightarrow _ \Rightarrow x \hookrightarrow _ \quad (\chi \in \{\varphi, \psi\}), \text{ PC}$	
5	$(\varphi \wedge x \hookrightarrow _) * (\psi \wedge x \hookrightarrow _) \Rightarrow x \hookrightarrow _ * x \hookrightarrow _ \quad (*I_{LR}), 4$	
6	$x \hookrightarrow _ * x \hookrightarrow _ \Rightarrow \perp \quad (_{\text{ALIAS}}^*)$	
7	$(\varphi \wedge x \hookrightarrow _) * (\psi \wedge x \hookrightarrow _) \Rightarrow \perp \quad (\Rightarrow \text{TR}), 5, 6$	
8	$(\varphi \wedge x \hookrightarrow _) * \psi \Rightarrow \perp \vee ((\varphi \wedge x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _)) \quad \text{PC}, 2, 3, 7$	
9	$(\varphi \wedge x \hookrightarrow _) * \psi \Rightarrow (\varphi \wedge x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _) \quad \text{PC}, 8$	
10	$\varphi \wedge x \hookrightarrow _ \Rightarrow \varphi \quad \text{PC}$	
11	$(\varphi \wedge x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _) \Rightarrow \varphi * (\psi \wedge \neg x \hookrightarrow _) \quad (*), 10$	
12	$(\varphi \wedge x \hookrightarrow _) * \psi \Rightarrow \varphi * (\psi \wedge \neg x \hookrightarrow _) \quad (\Rightarrow \text{TR}), 9, 11 \quad \square$	

Proof of $(I_{6.12.5}^)$.*

1	$\varphi \Rightarrow (\varphi \wedge x \hookrightarrow _) \vee (\varphi \wedge \neg x \hookrightarrow _) \quad \text{PC}$	
2	$\varphi * (\neg x \hookrightarrow _ \wedge \psi) \Rightarrow ((\varphi \wedge x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _)) \vee ((\varphi \wedge \neg x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _)) \quad (*), 1, (I_3^*)$	
3	$\chi \wedge \neg x \hookrightarrow _ \Rightarrow \neg x \hookrightarrow _ \quad (\chi \in \{\varphi, \psi\}), \text{ PC}$	
4	$(\varphi \wedge \neg x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _) \Rightarrow \neg x \hookrightarrow _ * \neg x \hookrightarrow _ \quad \text{PC}, (*I_{LR}), 3$	
5	$\neg x \hookrightarrow _ * \neg x \hookrightarrow _ \Rightarrow \neg x \hookrightarrow _ \quad (_{\neg \text{ALLOC}}^*)$	
6	$\varphi * (\neg x \hookrightarrow _ \wedge \psi) \Rightarrow ((\varphi \wedge x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _)) \vee \neg x \hookrightarrow _ \quad \text{PC}, 2, 4, 5$	
7	$x \hookrightarrow _ \wedge (\varphi * (\neg x \hookrightarrow _ \wedge \psi)) \Rightarrow (\varphi \wedge x \hookrightarrow _) * (\psi \wedge \neg x \hookrightarrow _) \quad \text{PC}, 6 \quad \square$	

Below, we show that in $\mathcal{H}_C(*)$ the axioms (I_1^C) and (I_2^C) of \mathcal{H}_C are superfluous and can be removed. Because of this, both (I_1^C) and (I_2^C) do not appear in the (final) proof system $\mathcal{H}_C(*, \neg *)$ given in Figure 6.2.

Lemma 6.13. The axioms (I_1^C) and (I_2^C) are derivable in $\mathcal{H}_C(*)$.

Proof of (I_1^C) . The proof is by induction on β .

base case: $\beta = 0$. The instance of the axiom (I_1^C) with $\beta = 0$ amounts to derive the formula $\text{size} \geq 1 \Rightarrow \text{size} \geq 0$. By definition, $\text{size} \geq 0 = \top$, and therefore by propositional reasoning, $\vdash_{\mathcal{H}_C(*)} \text{size} \geq 1 \Rightarrow \text{size} \geq 0$.

induction step: $\beta > 0$. By induction hypothesis, assume $\vdash_{\mathcal{H}_C(*)} \text{size} \geq \beta \Rightarrow \text{size} \geq \beta - 1$. The formula $\text{size} \geq \beta + 1 \Rightarrow \text{size} \geq \beta$ is derived as follows:

1	$\text{size} \geq \beta \Rightarrow \text{size} \geq \beta - 1 \quad \text{Induction hypothesis}$	
2	$(\text{size} \geq \beta) * \neg \text{emp} \Rightarrow (\text{size} \geq \beta - 1) * \neg \text{emp} \quad (*), 1$	

$$3 \quad \left| \begin{array}{l} \mathbf{size} \geq \beta + 1 \Rightarrow \mathbf{size} \geq \beta \end{array} \right. \quad \text{2, def. of } \mathbf{size} \quad \square$$

Before proving (I_2^C) , we derive the following intermediate theorem.

$$(I_{6.13.1}^*) \quad \Lambda_{x \in X}(x \hookrightarrow _ \wedge \Lambda_{y \in X \setminus \{x\}} x \neq y) \Rightarrow (\ast_{x \in X}(x \hookrightarrow _ \wedge \mathbf{size} = 1)) * \top \quad \blacktriangleleft [X \subseteq_{fin} \text{VAR}]$$

Proof of $(I_{6.13.1}^)$.* The proof is by induction on the size of X . We distinguish two base cases, one for $\text{card}(X) = 0$ and one for $\text{card}(X) = 1$.

base case: $\text{card}(X) = 0$. In this case, $(I_{6.13.1}^*)$ is $\top \Rightarrow \top * \top$.

1	$\mathbf{emp} \Rightarrow \top$	PC
2	$\top \Rightarrow \top * \mathbf{emp}$	$(\text{ID})^*$
3	$\top * \mathbf{emp} \Rightarrow \mathbf{emp} * \top$	$(\text{COM})^*$
4	$\mathbf{emp} * \top \Rightarrow \top * \top$	$(\ast), 1$
5	$\top \Rightarrow \top * \top$	$(\Rightarrow \text{TR}), 2, 3, 4$

base case: $\text{card}(X) = 1$. In this case, $(I_{6.13.1}^*)$ is exactly (ATOM^2) .

induction step: $\text{card}(X) \geq 2$. Let $z \in X$. By induction hypothesis,

$$\vdash_{\mathcal{H}_C(\ast)} \Lambda_{v \in X \setminus \{z\}}(x \hookrightarrow _ \wedge \Lambda_{w \in X \setminus \{v,z\}} v \neq w) \Rightarrow (\ast_{v \in X \setminus \{z\}}(v \hookrightarrow _ \wedge \mathbf{size} = 1)) * \top.$$

We write χ for the premise $\Lambda_{v \in X \setminus \{z\}}(x \hookrightarrow _ \wedge \Lambda_{w \in X \setminus \{v,z\}} v \neq w)$ above. Below, we aim at proving that

$$\vdash_{\mathcal{H}_C(\ast)} \Lambda_{x \in X}(x \hookrightarrow _ \wedge \Lambda_{y \in X \setminus \{x\}} x \neq y) \Rightarrow (z \hookrightarrow _ \wedge \mathbf{size} = 1) * \chi.$$

In this way, the provability of $(I_{6.13.1}^*)$ follows directly by induction hypothesis together with (COM^*) and (\ast) . We have,

1	$\Lambda_{x \in X}(x \hookrightarrow _ \wedge \Lambda_{y \in X \setminus \{x\}} x \neq y) \Rightarrow (z \hookrightarrow _ \wedge \mathbf{size} = 1) * \top$	(ATOM^2) and PC
2	$\top \Rightarrow \chi \vee \neg \chi$	PC
3	$(z \hookrightarrow _ \wedge \mathbf{size} = 1) * \top \Rightarrow (z \hookrightarrow _ \wedge \mathbf{size} = 1) * (\chi \vee \neg \chi)$	$(\ast), (\text{COM}^*), 2$
4	$(z \hookrightarrow _ \wedge \mathbf{size} = 1) * (\chi \vee \neg \chi) \Rightarrow$ $((z \hookrightarrow _ \wedge \mathbf{size} = 1) * \chi) \vee ((z \hookrightarrow _ \wedge \mathbf{size} = 1) * \neg \chi)$	(COM^*) and (I_3^*)
5	$\Lambda_{x \in X}(x \hookrightarrow _ \wedge \Lambda_{y \in X \setminus \{x\}} x \neq y) \Rightarrow$ $((z \hookrightarrow _ \wedge \mathbf{size} = 1) * \chi) \vee ((z \hookrightarrow _ \wedge \mathbf{size} = 1) * \neg \chi)$	$(\Rightarrow \text{TR})$ 1, 3, 4

By propositional reasoning, $\neg \chi$ is equivalent to $\bigvee_{v \in X \setminus \{z\}} (\neg x \hookrightarrow _ \vee \bigvee_{w \in X \setminus \{v,z\}} v = w)$. Due to the complexity of this formula, we proceed now rather informally, but our arguments entail the existence of a proper derivation. We aim at showing that

$$\vdash_{\mathcal{H}_C(\ast)} \Lambda_{x \in X}(x \hookrightarrow _ \wedge \Lambda_{y \in X \setminus \{x\}} x \neq y) \wedge ((z \hookrightarrow _ \wedge \mathbf{size} = 1) * \neg \chi) \Rightarrow \perp. \quad (\dagger)$$

By propositional calculus and (I_3^*) , we can distribute conjunctions and separating conjunctions over disjunctions. We derive:

$$\vdash_{\mathcal{H}_C(*)} \bigwedge_{x \in X} (x \hookrightarrow _ \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \wedge ((z \hookrightarrow _ \wedge \text{size} = 1) * \neg \chi) \Rightarrow \gamma' \vee \gamma'',$$

where γ' and γ'' are defined, respectively, as

$$\gamma' \stackrel{\text{def}}{=} \bigvee_{v \in X \setminus \{z\}} \left(\bigwedge_{x \in X} (x \hookrightarrow _ \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \wedge ((z \hookrightarrow _ \wedge \text{size} = 1) * \neg v \hookrightarrow _) \right),$$

$$\gamma'' \stackrel{\text{def}}{=} \bigvee_{\substack{v \in X \setminus \{z\} \\ w \in X \setminus \{z, v\}}} \left(\bigwedge_{x \in X} (x \hookrightarrow _ \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \wedge ((z \hookrightarrow _ \wedge \text{size} = 1) * v = w) \right).$$

In order to deduce (\dagger) it is sufficient to prove, in $\mathcal{H}_C(*)$, that every disjunct of γ' and γ'' implies \perp . Clearly, if γ' and γ'' do not have any disjunct, i.e. when $X \setminus \{z\}$ is empty, then the formula is propositionally equivalent to \perp , which allows us to conclude (\dagger) . Otherwise, let us consider each disjunct in γ' and γ'' (separately), and prove their inconsistency.

case: γ' . Let $v \in X \setminus \{z\}$. We show the inconsistency of

$$\bar{\gamma} \stackrel{\text{def}}{=} \bigwedge_{x \in X} (x \hookrightarrow _ \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \wedge ((z \hookrightarrow _ \wedge \text{size} = 1) * \neg v \hookrightarrow _).$$

6	$\bigwedge_{x \in X} (x \hookrightarrow _ \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \Rightarrow v \hookrightarrow _ \wedge v \neq z$	PC
7	$\bar{\gamma} \Rightarrow v \hookrightarrow _ \wedge v \neq z \wedge ((z \hookrightarrow _ \wedge \text{size} = 1) * \neg v \hookrightarrow _)$	PC
8	$v \hookrightarrow _ \wedge ((z \hookrightarrow _ \wedge \text{size} = 1) * \neg v \hookrightarrow _) \Rightarrow$ $((z \hookrightarrow _ \wedge \text{size} = 1 \wedge v \hookrightarrow _) * \neg v \hookrightarrow _)$	$(I_{6.12.5}^*)$
9	$v \neq z \wedge ((z \hookrightarrow _ \wedge \text{size} = 1 \wedge v \hookrightarrow _) * \neg v \hookrightarrow _) \Rightarrow$ $((z \hookrightarrow _ \wedge \text{size} = 1 \wedge v \hookrightarrow _ \wedge v \neq z) * \neg v \hookrightarrow _)$	$(I_{6.12.1}^*)$
10	$z \hookrightarrow _ \wedge v \hookrightarrow _ \wedge v \neq z \Rightarrow \text{size} \geq 2$	$(\text{ALLOC}_{\text{SIZE}})$
11	$\text{size} = 1 \Rightarrow \neg \text{size} \geq 2$	PC
12	$z \hookrightarrow _ \wedge \text{size} = 1 \wedge v \hookrightarrow _ \wedge v \neq z \Rightarrow \perp$	$(\Rightarrow \text{Tr})$, PC, 10, 11
13	$\bar{\gamma} \Rightarrow (z \hookrightarrow _ \wedge \text{size} = 1 \wedge v \hookrightarrow _ \wedge v \neq z) * \neg v \hookrightarrow _$	PC, 7, 8, 9
14	$(z \hookrightarrow _ \wedge \text{size} = 1 \wedge v \hookrightarrow _ \wedge v \neq z) * \neg v \hookrightarrow _ \Rightarrow \perp * \neg v \hookrightarrow _$	$(*)$, 12
15	$\perp * \neg v \hookrightarrow _ \Rightarrow \perp$	(I_4^*) , 14
16	$\bar{\gamma} \Rightarrow \perp$	PC, 13, 15

Since $\bar{\gamma}$ is an arbitrary disjunct appearing in γ' , we conclude that $\vdash_{\mathcal{H}_C(*)} \gamma' \Rightarrow \perp$.

case: γ'' . Let $v \in X \setminus \{z\}$ and $w \in X \setminus \{z, w\}$. Notice that if v or w do not exist, then γ'' is defined as \perp and so the proof is complete. Otherwise, we show the inconsistency of

$$\hat{\gamma} \stackrel{\text{def}}{=} \bigwedge_{x \in X} (x \hookrightarrow _ \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \wedge ((z \hookrightarrow _ \wedge \text{size} = 1) * v = w).$$

17	$\bigwedge_{x \in X} (x \hookrightarrow _ \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \Rightarrow v \neq w$	PC
18	$z \hookrightarrow _ \wedge \text{size} = 1 \Rightarrow \top$	PC
19	$(z \hookrightarrow _ \wedge \text{size} = 1) * v = w \Rightarrow v = w * \top$	$(*)$, 18, (COM^*)
20	$v = w * \top \Rightarrow v = w$	(MONO^*)
21	$((z \hookrightarrow _ \wedge \text{size} = 1) * v = w) \Rightarrow v = w$	$(\Rightarrow \text{Tr})$, 19, 20
23	$\hat{\gamma} \Rightarrow \perp$	PC, 17, 22

$$\begin{array}{ll}
\bigwedge \{x \sim y \subseteq_{\text{LIT}} \{\varphi \mid \psi\} \mid \sim \in \{=, \neq\}\} & \wedge \bigwedge \{x \hookrightarrow - \subseteq_{\text{LIT}} \{\varphi \mid \psi\}\} \\
\wedge \bigwedge \{\neg x \hookrightarrow - \subseteq_{\text{LIT}} \{\varphi ; \psi\}\} & \wedge \bigwedge \{\neg x \hookrightarrow y \mid x \hookrightarrow - \wedge \neg x \hookrightarrow y \subseteq_{\text{LIT}} \{\varphi \mid \psi\}\} \\
\wedge \bigwedge \{x \neq x \mid x \hookrightarrow - \subseteq_{\text{LIT}} \{\varphi ; \psi\}\} & \wedge \bigwedge \left\{ \text{size} \geq \beta_1 + \beta_2 \mid \begin{array}{l} \text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi \\ \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
\wedge \bigwedge \{x \hookrightarrow y \subseteq_{\text{LIT}} \{\varphi \mid \psi\}\} & \wedge \bigwedge \left\{ \neg \text{size} \geq \beta_1 + \beta_2 - 1 \mid \begin{array}{l} \neg \text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi \\ \neg \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi \end{array} \right\}
\end{array}$$

Figure 6.7: The formula $\langle *\rangle(\varphi, \psi)$.

Since $\hat{\gamma}$ is an arbitrary disjunct appearing in γ'' , we conclude that $\vdash_{\mathcal{H}_C(*)} \gamma'' \Rightarrow \perp$.

From $\vdash_{\mathcal{H}_C(*)} \gamma' \Rightarrow \perp$ and $\vdash_{\mathcal{H}_C(*)} \gamma'' \Rightarrow \perp$ we conclude that (\dagger) holds. From the theorem 5 derived in this proof, this allows us to conclude that

$$\vdash_{\mathcal{H}_C(*)} \bigwedge_{x \in X} (x \hookrightarrow - \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \Rightarrow (z \hookrightarrow - \wedge \text{size} = 1) * \chi. \quad \square$$

Let us move to the derivation of (I_2^C) .

Proof of (I_2^C) . Let $X \subseteq_{\text{fin}} \text{VAR}$. If $X = \emptyset$, the instance of the axiom (I_2^C) becomes $\top \Rightarrow \text{size} \geq 0$, that is $\top \Rightarrow \top$, by definition of $\text{size} \geq 0$. Below, assume $X \neq \emptyset$ and fix $z \in X$.

1	$\bigwedge_{x \in X} (x \hookrightarrow - \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \Rightarrow (\bigast_{x \in X} (x \hookrightarrow - \wedge \text{size} = 1)) * \top$	$(I_{6.13.1}^*)$
2	$x \hookrightarrow - \wedge \text{size} = 1 \Rightarrow \text{size} \geq 1$	PC, def. of $\text{size} = 1$
3	$(\bigast_{x \in X} (x \hookrightarrow - \wedge \text{size} = 1)) * \top \Rightarrow (\bigast_{x \in X} \text{size} \geq 1) * \top.$	multiple applications of $(*)$, 2, (COM^*) and $(\Rightarrow \text{TR})$
4	$(\bigast_{x \in X} \text{size} \geq 1) * \top \Rightarrow (\text{size} \geq 1 * \top) * (\bigast_{x \in X \setminus \{z\}} \text{size} \geq 1)$	$(\text{COM}^*), (\text{ASSOC}^*), \text{def. of } z$
5	$\text{size} \geq 1 * \top \Rightarrow \text{size} \geq 1$	$(\text{MONO}^*), \text{def. of } \text{size} \geq 1$
6	$(\text{size} \geq 1 * \top) * (\bigast_{x \in X \setminus \{z\}} \text{size} \geq 1) \Rightarrow (\bigast_{x \in X} \text{size} \geq 1)$	$(*)$
7	$(\bigast_{x \in X} \text{size} \geq 1) \Rightarrow \text{size} \geq \text{card}(X)$	$(\text{ASSOC}^*), \text{def. of } \text{size} \geq \text{card}(X)$
8	$\bigwedge_{x \in X} (x \hookrightarrow - \wedge \bigwedge_{y \in X \setminus \{x\}} x \neq y) \Rightarrow \text{size} \geq \text{card}(X)$	$(\Rightarrow \text{TR}), 1, 3, 4, 6, 7 \quad \square$

From now on, we understand $\mathcal{H}_C(*)$ as the proof system obtained from \mathcal{H}_C by adding all schemata from Figure 6.6 while removing (I_1^C) and (I_2^C) . We show that $\mathcal{H}_C(*)$ enjoys the $*$ -simulation property when the argument formulae are core types. That is, given two satisfiable core types φ and ψ , in $\text{CoreTypes}(X, \alpha)$, we show that the formula $\varphi * \psi$ is provably equivalent to the formula $\langle *\rangle(\varphi, \psi)$ belonging to $\text{Conj}(\text{Core}(X, 2\alpha))$, and defined in Figure 6.7.

Lemma 6.14. Let $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq \text{card}(X)$. Let φ and ψ in $\text{CoreTypes}(X, \alpha)$. If both φ and ψ are satisfiable, then $\vdash_{\mathcal{H}_C(*)} \varphi * \psi \Leftrightarrow \langle *\rangle(\varphi, \psi)$.

The equivalence $\varphi * \psi \Leftrightarrow \langle *\rangle(\varphi, \psi)$ is reminiscent to the one in [63, Lemma 3] that is proved semantically. In a way, because $\mathcal{H}_C(*)$ will reveal to be complete, the restriction of the proof of [63, Lemma 3] to $\text{SL}(*, x \hookrightarrow -)$ can be replayed completely syntactically within $\mathcal{H}_C(*)$.

Proof. First of all, let us briefly explain what is the rationale for having literals of the form $x \neq x$ in the definition of $\langle * \rangle(\varphi, \psi)$. Recall that $x \hookrightarrow_ - \subseteq_{\text{LIT}} \{\varphi ; \psi\}$ is a shortcut to state that $x \hookrightarrow_ -$ occurs in the core type φ and $x \hookrightarrow_ -$ also occurs in the core type ψ . Since $(x \hookrightarrow_ - \wedge \varphi') * (x \hookrightarrow_ - \wedge \psi')$ is unsatisfiable (see e.g. (ALIAS^*)), $x \hookrightarrow_ - \subseteq_{\text{LIT}} \{\varphi ; \psi\}$ entails that $\langle * \rangle(\varphi, \psi)$ should be unsatisfiable. That is why, if $x \hookrightarrow_ - \subseteq_{\text{LIT}} \{\varphi ; \psi\}$, then $x \neq x$ is part of $\langle * \rangle(\varphi, \psi)$.

(\Rightarrow): Let us show that $\vdash_{\mathcal{H}_C(*)} \varphi * \psi \Rightarrow \langle * \rangle(\varphi, \psi)$. We establish that $\vdash_{\mathcal{H}_C(*)} \varphi * \psi \Rightarrow L$ holds for every literal L of $\langle * \rangle(\varphi, \psi)$. We reason by a case analysis on $L \subseteq_{\text{LIT}} \langle * \rangle(\varphi, \psi)$.

case: L is an (in)equality or $L = x \hookrightarrow y$. For all the equalities and inequalities in φ or ψ , as well as all the literals of the form $x \hookrightarrow y$, $\vdash_{\mathcal{H}_C(*)} \varphi * \psi \Rightarrow L$ follow from the rule $(*)$ and the axiom (MONO^*) . Let us provide below the proper derivation when L is a literal in φ that is an equality, an inequality or of the form $x \hookrightarrow y$.

$1 \quad \varphi \Rightarrow L$ $2 \quad \psi \Rightarrow \top$ $3 \quad \varphi * \psi \Rightarrow L * \top$	PC PC $(\text{I}_{LR}), 1, 2$	$4 \quad L * \top \Rightarrow L$ $5 \quad \varphi * \psi \Rightarrow L$	(MONO^*) $(\Rightarrow \text{TR}), 3, 4$
---	---	--	--

Assume there is a literal $x \neq x$ that occurs in $\langle * \rangle(\varphi, \psi)$. As both φ and ψ are satisfiable, and thanks to $(\overline{\text{id}})$, this is necessarily due to $x \hookrightarrow_ -$ occurring both in φ and ψ .

$1 \quad \varphi \Rightarrow x \hookrightarrow_ -$ $2 \quad \psi \Rightarrow x \hookrightarrow_ -$ $3 \quad \varphi * \psi \Rightarrow x \hookrightarrow_ - * x \hookrightarrow_ -$	PC PC $(\text{I}_{LR}), 1, 2$	$4 \quad x \hookrightarrow_ - * x \hookrightarrow_ - \Rightarrow \perp$ $5 \quad \perp \Rightarrow x \neq x$ $6 \quad \varphi * \psi \Rightarrow x \neq x$	(ALIAS^*) PC $(\Rightarrow \text{TR}), 4, 5$
---	---	--	--

case: $L = x \hookrightarrow_ -$. Follows from (I_5^*) and $(*)$.

case: $L = \neg x \hookrightarrow_ -$. Follows from $(\neg \text{ALLOC}^*)$ and $(*)$.

case: $L = \neg x \hookrightarrow y$. Let $\neg x \hookrightarrow y$ be a literal occurring in $\langle * \rangle(\varphi, \psi)$. So, $x \hookrightarrow_ - \wedge \neg x \hookrightarrow y$ occurs in φ or ψ , say in φ (the other case is equivalent, due to (COM^*)).

$1 \quad \varphi \Rightarrow x \hookrightarrow_ - \wedge \neg x \hookrightarrow y$ $2 \quad \psi \Rightarrow \top$ $3 \quad \varphi * \psi \Rightarrow (x \hookrightarrow_ - \wedge \neg x \hookrightarrow y) * \top$ $4 \quad (x \hookrightarrow_ - \wedge \neg x \hookrightarrow y) * \top \Rightarrow \neg x \hookrightarrow y$ $5 \quad \varphi * \psi \Rightarrow \neg x \hookrightarrow y$	PC PC $(\text{I}_{LR}), 1, 2$ $(\neg \text{PTO})$ $(\Rightarrow \text{TR}), 3, 4$
--	---

case: $L = \text{size} \geq \beta_1 + \beta_2$, where $\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$ and $\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$.

$1 \quad \varphi \Rightarrow \text{size} \geq \beta_1$ $2 \quad \psi \Rightarrow \text{size} \geq \beta_2$ $3 \quad \varphi * \psi \Rightarrow \text{size} \geq \beta_1 * \text{size} \geq \beta_2$ $4 \quad \varphi * \psi \Rightarrow \text{size} \geq (\beta_1 + \beta_2)$	PC PC $(\text{I}_{LR}), 1, 2$ Def. size
--	---

Notice that, as φ and ψ are satisfiable core types, $\mathbf{size} \geq 0$ appears positively in both these formulae, and thus appears in $\langle *\rangle(\varphi, \psi)$.

case: $L = \neg\mathbf{size} \geq \beta_1 + \beta_2 \doteq 1$, where $\neg\mathbf{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$ and $\neg\mathbf{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$.

1	$\varphi \Rightarrow \neg\mathbf{size} \geq \beta_1$	PC
2	$\psi \Rightarrow \neg\mathbf{size} \geq \beta_2$	PC
3	$\varphi * \psi \Rightarrow \neg\mathbf{size} \geq \beta_1 * \neg\mathbf{size} \geq \beta_2$	($\mathbf{*I}_{LR}$), 1, 2
4	$\neg\mathbf{size} \geq \beta_1 * \neg\mathbf{size} \geq \beta_2 \Rightarrow \neg\mathbf{size} \geq \beta_1 + \beta_2 \doteq 1$	($\mathbf{(\neg size)^*}$)
5	$\varphi * \psi \Rightarrow \neg\mathbf{size} \geq \beta_1 + \beta_2 \doteq 1$	($\Rightarrow \mathbf{TR}$), 3, 4

(\Leftarrow): Let us show that $\vdash_{\mathcal{H}_C(*)} \langle *\rangle(\varphi, \psi) \Rightarrow \varphi * \psi$. If $\langle *\rangle(\varphi, \psi)$ is unsatisfiable, then by completeness of \mathcal{H}_C (Theorem 6.9), $\vdash_{\mathcal{H}_C} \langle *\rangle(\varphi, \psi) \Rightarrow \perp$, and thus $\vdash_{\mathcal{H}_C} \langle *\rangle(\varphi, \psi) \Rightarrow \varphi * \psi$. Since $\mathcal{H}_C(*)$ includes \mathcal{H}_C , we conclude that $\vdash_{\mathcal{H}_C(*)} \langle *\rangle(\varphi, \psi) \Rightarrow \varphi * \psi$. Otherwise, below, we assume $\langle *\rangle(\varphi, \psi)$ to be satisfiable. In particular, this implies that no literals of the form $x \neq x$ or $\neg\mathbf{size} \geq 0$ appear in $\langle *\rangle(\varphi, \psi)$. Moreover, by definition of $\langle *\rangle(\varphi, \psi)$, this implies that φ , ψ and $\langle *\rangle(\varphi, \psi)$ agree on the satisfaction of the core formulae $x = y$, i.e. φ , ψ and $\langle *\rangle(\varphi, \psi)$ contain exactly the same (in)equalities. Since φ is satisfiable, these equalities define an equivalence relation. Let x_1, \dots, x_n be a maximal enumeration of representatives of the equivalence classes (one per equivalence class) such that $x_i \hookrightarrow -$ occurs in $\langle *\rangle(\varphi, \psi)$. As it is maximal, for every $x \hookrightarrow - \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi)$ there is $i \in [1, n]$ such that x_i is syntactically equal to x . Consequently, from the definition of $\langle *\rangle(\varphi, \psi)$, if $x \hookrightarrow -$ occurs in φ or in ψ , then there is some x_i such that $x = x_i$ occurs in φ (and therefore also in ψ and in $\langle *\rangle(\varphi, \psi)$). Let us define the formula ALLOC below:

$$\text{ALLOC} \stackrel{\text{def}}{=} (x_1 \hookrightarrow - \wedge \mathbf{size} = 1) * \dots * (x_n \hookrightarrow - \wedge \mathbf{size} = 1).$$

We have,

1	$\langle *\rangle(\varphi, \psi) \Rightarrow \bigwedge_{i \in [1, n]} (x_i \hookrightarrow - \wedge \bigwedge_{j \in [1, n] \setminus \{i\}} x_i \neq x_j)$	PC, def. of x_1, \dots, x_n
2	$\bigwedge_{i \in [1, n]} (x_i \hookrightarrow - \wedge \bigwedge_{j \in [1, n] \setminus \{i\}} x_i \neq x_j) \Rightarrow \text{ALLOC} * \top$	($I_{6.13.1}^*$)
3	$\langle *\rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \top$	($\Rightarrow \mathbf{TR}$), 1, 2

Moreover, we show that $\vdash_{\mathcal{H}_C(*)} \text{ALLOC} \Rightarrow \mathbf{size} \geq n$ and $\vdash_{\mathcal{H}_C(*)} \text{ALLOC} \Rightarrow \neg\mathbf{size} \geq n+1$ (theorems 4 and 7 below), and so $\vdash_{\mathcal{H}_C(*)} \text{ALLOC} \Rightarrow \mathbf{size} = n$.

1	$\chi \wedge \mathbf{size} = 1 \Rightarrow \mathbf{size} \geq 1$	PC, def. of $\mathbf{size} = 1$
2	$\chi \wedge \mathbf{size} = 1 \Rightarrow \neg\mathbf{size} \geq 2$	PC, def. of $\mathbf{size} = 1$
3	$\text{ALLOC} \Rightarrow \mathbf{*}_{i \in [1, n]} \mathbf{size} \geq 1$	multiple applications of ($*$), 1, (\mathbf{COM}^*) and ($\Rightarrow \mathbf{TR}$)
4	$\text{ALLOC} \Rightarrow \mathbf{size} \geq n$	3, def. of $\mathbf{size} \geq n$
5	$\text{ALLOC} \Rightarrow \mathbf{*}_{i \in [1, n]} \neg\mathbf{size} \geq 2$	multiple applications of ($*$), 2, (\mathbf{COM}^*) and ($\Rightarrow \mathbf{TR}$)
6	$\mathbf{*}_{i \in [1, n]} \neg\mathbf{size} \geq 2 \Rightarrow \neg\mathbf{size} \geq n+1$	n applications of ($\mathbf{(\neg size)^*}$) and ($*$)

7	$\text{ALLOC} \Rightarrow \neg\text{size} \geq n + 1$	$(\Rightarrow\text{TR})$, 5, 6
8	$\text{ALLOC} \Rightarrow \text{size} = n$	PC, 4, 7, def. of $\text{size} = n$

After deriving $\vdash_{\mathcal{H}_C(*)} \langle *\rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \top$ and $\vdash_{\mathcal{H}_C(*)} \text{ALLOC} \Rightarrow \text{size} = n$, the proof is divided in three steps: (1) we isolate the allocated cells and the garbage, (2) we distribute the $x \hookrightarrow _$ and $\text{size} \geq \beta$ literals according to the goal $\varphi * \psi$ and (3) we add the missing literals.

Step 1, isolating allocated cells and garbage. Since $\langle *\rangle(\varphi, \psi)$ is a conjunction of literals built from core formulae, we can rely on $\max_{\text{size}}(\langle *\rangle(\varphi, \psi))$, i.e. the maximum β among the formulae $\text{size} \geq \beta$ appearing positively in $\langle *\rangle(\varphi, \psi)$. First, we show some important properties of $\langle *\rangle(\varphi, \psi)$, related to $\max_{\text{size}}(\langle *\rangle(\varphi, \psi))$.

A. $\max_{\text{size}}(\langle *\rangle(\varphi, \psi)) = \max_{\text{size}}(\varphi) + \max_{\text{size}}(\psi)$,

B. If there is $\beta \in \mathbb{N}$ such that $\neg\text{size} \geq \beta \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi)$, then

$$\neg\text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi, \quad \neg\text{size} \geq \max_{\text{size}}(\psi) + 1 \subseteq_{\text{LIT}} \psi.$$

C. If there is $\beta \in \mathbb{N}$ such that $\neg\text{size} \geq \beta \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi)$, then

$$\neg\text{size} \geq \max_{\text{size}}(\langle *\rangle(\varphi, \psi)) + 1 \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi).$$

Proof of (A). By definition of $\max_{\text{size}}(\cdot)$, we know that $\text{size} \geq \max_{\text{size}}(\varphi) \subseteq_{\text{LIT}} \varphi$ and $\text{size} \geq \max_{\text{size}}(\psi) \subseteq_{\text{LIT}} \psi$. By definition of $\langle *\rangle(\varphi, \psi)$, this allows us to conclude that $\text{size} \geq \max_{\text{size}}(\varphi) + \max_{\text{size}}(\psi) \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi)$. *Ad absurdum*, suppose that $\max_{\text{size}}(\varphi) + \max_{\text{size}}(\psi) \neq \max_{\text{size}}(\langle *\rangle(\varphi, \psi))$ and thus, by definition of $\max_{\text{size}}(\cdot)$, there is $\beta > \max_{\text{size}}(\varphi) + \max_{\text{size}}(\psi)$ such that $\text{size} \geq \beta \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi)$. By definition of $\langle *\rangle(\varphi, \psi)$, we conclude that there are β_1 and β_2 such that $\beta_1 + \beta_2 = \beta$, $\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$ and $\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$. As $\beta_1 + \beta_2 > \max_{\text{size}}(\varphi) + \max_{\text{size}}(\psi)$, either $\beta_1 > \max_{\text{size}}(\varphi)$ or $\beta_2 > \max_{\text{size}}(\psi)$. Let us assume $\beta_1 > \max_{\text{size}}(\varphi)$ (the other case is analogous). We have $\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$. However, this is contradictory, since by definition of $\max_{\text{size}}(\cdot)$ for all $\beta' > \max_{\text{size}}(\varphi)$, $\text{size} \geq \beta' \not\subseteq_{\text{LIT}} \varphi$. Thus, $\max_{\text{size}}(\varphi) + \max_{\text{size}}(\psi) = \max_{\text{size}}(\langle *\rangle(\varphi, \psi))$.

Proof of (B). Let $\beta \in \mathbb{N}$ such that $\neg\text{size} \geq \beta \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi)$. By definition of $\langle *\rangle(\varphi, \psi)$, this implies that there are $\beta_1, \beta_2 \in [0, \alpha]$ such that $\beta = \beta_1 + \beta_2 - 1$, $\neg\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$ and $\neg\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$. Since φ and ψ are satisfiable, by definition of $\max_{\text{size}}(\cdot)$, we derive that $\beta_1 > \max_{\text{size}}(\varphi)$ and $\beta_2 > \max_{\text{size}}(\psi)$. This implies that the core formula $\text{size} \geq \max_{\text{size}}(\varphi) + 1$ belongs to $\text{Core}(X, \alpha)$ and, analogously, that the core formula $\text{size} \geq \max_{\text{size}}(\psi) + 1$ belongs to $\text{Core}(X, \alpha)$. Since φ is in $\text{CoreTypes}(X, \alpha)$, this implies that $\text{size} \geq \max_{\text{size}}(\varphi) + 1$ is an atomic formula appearing in φ . By definition of $\max_{\text{size}}(\varphi)$, the formula cannot appear positively, i.e. $\neg\text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi$. Analogously, ψ is in $\text{CoreTypes}(X, \alpha)$, which leads to $\neg\text{size} \geq \max_{\text{size}}(\psi) + 1 \subseteq_{\text{LIT}} \psi$.

Proof of (C). Directly from (A) and (B). Indeed, by definition of $\langle *\rangle(\varphi, \psi)$, we know that for every $\neg\text{size} \geq \beta \subseteq_{\text{LIT}} \varphi$ and every $\neg\text{size} \geq \beta' \subseteq_{\text{LIT}} \psi$, $\neg\text{size} \geq \beta + \beta' - 1 \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi)$.

Now, let us consider $\beta_g = \max_{\text{size}}(\langle *\rangle(\varphi, \psi)) - n$. We define the formula GARB below:

$$\text{GARB} \stackrel{\text{def}}{=} \begin{cases} \text{size} = \beta_g & \text{if } \neg\text{size} \geq \beta \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi), \text{ for some } \beta \\ \text{size} \geq \beta_g & \text{otherwise,} \end{cases}$$

where we recall that $\text{size} = \beta_g$ stands for $\text{size} \geq \beta_g \wedge \neg(\text{size} \geq \beta_g + 1)$. Notice that GARB is a conjunction of literals where at least one $\text{size} \geq \beta$ occurs positively (i.e. $\text{size} \geq 0$). The

objective of this step of the proof is to show that $\vdash_{\mathcal{H}_C(*)} \langle * \rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \text{GARB}$. First, we focus on the positive part of GARB, and prove $\vdash_{\mathcal{H}_C(*)} \langle * \rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \text{size} \geq \beta_g$. If $\beta_g = 0$ then $\text{size} \geq \beta_g = \top$ and we have already shown $\vdash_{\mathcal{H}_C(*)} \langle * \rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \top$. So, let us assume that $\beta_g > 1$. Notice that $\max_{\text{size}}(\langle * \rangle(\varphi, \psi)) \doteq n = \max_{\text{size}}(\langle * \rangle(\varphi, \psi)) - n$. We have

1	$\top \Rightarrow \text{size} \geq \beta_g \vee \neg \text{size} \geq \beta_g$	PC
2	$\text{ALLOC} * \top \Rightarrow \text{ALLOC} * (\text{size} \geq \beta_g \vee \neg \text{size} \geq \beta_g)$	$(*)$, (COM^*) , 1
3	$\text{ALLOC} * (\text{size} \geq \beta_g \vee \neg \text{size} \geq \beta_g) \Rightarrow$ $(\text{ALLOC} * \text{size} \geq \beta_g) \vee (\text{ALLOC} * \neg \text{size} \geq \beta_g)$	(I_3^*) , (COM^*)
4	$\text{ALLOC} \Rightarrow \neg \text{size} \geq n + 1$	Previously derived
5	$\text{ALLOC} * \neg \text{size} \geq \beta_g \Rightarrow \neg \text{size} \geq n + 1 * \neg \text{size} \geq \beta_g$	$(*)$, 4
6	$(\neg \text{size} \geq n + 1) * \neg \text{size} \geq \beta_g \Rightarrow \neg \text{size} \geq \max_{\text{size}}(\langle * \rangle(\varphi, \psi))$	$(\neg \text{size}^*)$, def. of β_g
7	$\text{ALLOC} * \top \Rightarrow (\text{ALLOC} * \text{size} \geq \beta_g) \vee \neg \text{size} \geq \max_{\text{size}}(\langle * \rangle(\varphi, \psi))$	PC, 2, 3, 5, 6
8	$\langle * \rangle(\varphi, \psi) \Rightarrow \text{size} \geq \max_{\text{size}}(\langle * \rangle(\varphi, \psi))$	PC, def. of $\max_{\text{size}}(\cdot)$
9	$\langle * \rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \top$	Previously derived
10	$\langle * \rangle(\varphi, \psi) \Rightarrow (\text{ALLOC} * \text{size} \geq \beta_g) \vee \neg \text{size} \geq \max_{\text{size}}(\langle * \rangle(\varphi, \psi))$	$(\Rightarrow \text{TR})$, 7, 9
11	$\langle * \rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \text{size} \geq \beta_g$	PC, 8, 10

If for every β , $\neg \text{size} \geq \beta \not\subseteq_{\text{LIT}} \langle * \rangle(\varphi, \psi)$, then by definition of GARB we conclude that

$$\vdash_{\mathcal{H}_C(*)} \langle * \rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \text{GARB}.$$

Otherwise, suppose that there is β such that $\neg \text{size} \geq \beta \subseteq_{\text{LIT}} \langle * \rangle(\varphi, \psi)$. So, GARB is defined as $\text{size} \geq \beta_g \wedge \neg(\text{size} \geq \beta_g + 1)$. By (C), $\neg \text{size} \geq \max_{\text{size}}(\langle * \rangle(\varphi, \psi)) + 1 \subseteq_{\text{LIT}} \langle * \rangle(\varphi, \psi)$. By propositional reasoning,

$$\vdash_{\mathcal{H}_C(*)} \langle * \rangle(\varphi, \psi) \Rightarrow \neg \text{size} \geq \max_{\text{size}}(\langle * \rangle(\varphi, \psi)) + 1.$$

Then, $\langle * \rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \text{GARB}$ is derived as follows:

1	$\text{size} \geq \beta_g \Rightarrow (\text{size} \geq \beta_g \wedge \text{size} \geq \beta_g + 1) \vee \text{size} = \beta_g$	PC, def. of $\text{size} = \beta_g$
2	$\text{ALLOC} * \text{size} \geq \beta_g \Rightarrow$ $\text{ALLOC} * ((\text{size} \geq \beta_g \wedge \text{size} \geq \beta_g + 1) \vee \text{size} = \beta_g)$	$(*)$, (COM^*) , 1
3	$\text{ALLOC} * ((\text{size} \geq \beta_g \wedge \text{size} \geq \beta_g + 1) \vee \text{size} = \beta_g) \Rightarrow$ $(\text{ALLOC} * (\text{size} \geq \beta_g \wedge \text{size} \geq \beta_g + 1)) \vee (\text{ALLOC} * \text{size} = \beta_g)$	(I_3^*) , (COM^*)
4	$\text{size} \geq \beta_g \wedge \text{size} \geq \beta_g + 1 \Rightarrow \text{size} \geq \beta_g + 1$	PC
5	$\text{ALLOC} \Rightarrow \text{size} \geq n$	Previously derived
6	$\text{ALLOC} * (\text{size} \geq \beta_g \wedge \text{size} \geq \beta_g + 1) \Rightarrow \text{size} \geq n * \text{size} \geq \beta_g + 1$	$(*\text{I}_{LR})$, 4, 5
7	$\text{size} \geq n * \text{size} \geq \beta_g + 1 \Rightarrow \text{size} \geq \max_{\text{size}}(\langle * \rangle(\varphi, \psi)) + 1$	(ASSOC^*) , def. of $\text{size} \geq \beta$
8	$\text{ALLOC} * \text{size} \geq \beta_g \Rightarrow$	

	$\text{size} \geq \max_{\text{size}}(\langle *\rangle(\varphi, \psi)) + 1 \vee (\text{ALLOC} * \text{size} = \beta_g)$	PC, 2, 3, 6, 7
9	$\langle *\rangle(\varphi, \psi) \Rightarrow \text{ALLOC} * \text{size} \geq \beta_g$	Previously derived
10	$\langle *\rangle(\varphi, \psi) \Rightarrow \text{size} \geq \max_{\text{size}}(\langle *\rangle(\varphi, \psi)) + 1 \vee (\text{ALLOC} * \text{size} = \beta_g)$	($\Rightarrow \text{TR}$), 8, 9
11	$\langle *\rangle(\varphi, \psi) \Rightarrow \neg \text{size} \geq \max_{\text{size}}(\langle *\rangle(\varphi, \psi)) + 1$	PC, see above
12	$\langle *\rangle(\varphi, \psi) \Rightarrow (\text{ALLOC} * \underbrace{\text{size} = \beta_g}_{\text{GARB}})$	PC, 10, 11

Step 2, distributing alloc and size literals. In this step, we aim at showing that

$$\vdash_{\mathcal{H}_C(*)} \text{ALLOC} * \text{GARB} \Rightarrow \varphi^{(1)} * \psi^{(1)}$$

where $\varphi^{(1)}$ and $\psi^{(1)}$ are two formulae defined as follows:

$$\begin{aligned} \varphi^{(1)} &\stackrel{\text{def}}{=} \begin{cases} \text{size} = \max_{\text{size}}(\varphi) \wedge \bigwedge \{\mathbf{x}_i \hookrightarrow - \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\} & \text{if } \max_{\text{size}}(\varphi) < \alpha \\ \text{size} \geq \max_{\text{size}}(\varphi) \wedge \bigwedge \{\mathbf{x}_i \hookrightarrow - \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\} & \text{otherwise} \end{cases} \\ \psi^{(1)} &\stackrel{\text{def}}{=} \begin{cases} \text{size} = \max_{\text{size}}(\psi) \wedge \bigwedge \{\mathbf{x}_i \hookrightarrow - \subseteq_{\text{LIT}} \psi \mid i \in [1, n]\} & \text{if } \max_{\text{size}}(\psi) < \alpha \\ \text{size} \geq \max_{\text{size}}(\psi) \wedge \bigwedge \{\mathbf{x}_i \hookrightarrow - \subseteq_{\text{LIT}} \psi \mid i \in [1, n]\} & \text{otherwise} \end{cases} \end{aligned}$$

Before tackling this derivation, a few more steps are required. First of all, notice that, if there is a formula $\mathbf{x} \hookrightarrow -$ occurring both in φ and ψ , then, by definition of $\langle *\rangle(\varphi, \psi)$, $\mathbf{x} \neq \mathbf{x}$ occurs in $\langle *\rangle(\varphi, \psi)$. This contradicts fact that $\langle *\rangle(\varphi, \psi)$ is satisfiable. Therefore, we derive that the set of variables $\mathbf{x}_1, \dots, \mathbf{x}_n$ can be split into two disjoint subsets, the one “allocated” in φ , and the others in ψ . Let n_φ (resp. n_ψ) denote the number of equivalence classes of variables allocated in φ (resp. ψ). Clearly, $n = n_\varphi + n_\psi$. Moreover, since φ and ψ are satisfiable core types in $\text{CoreTypes}(\mathbf{X}, \alpha)$, where $\alpha \geq \text{card}(\mathbf{X})$, we must have $n_\varphi \leq \max_{\text{size}}(\varphi)$ and $n_\psi \leq \max_{\text{size}}(\psi)$ (see axiom (I_2^C)). By (A), we conclude that $n \leq \max_{\text{size}}(\langle *\rangle(\varphi, \psi))$. We define the formulae:

$$\begin{aligned} \text{ALLOC}(\varphi) &\stackrel{\text{def}}{=} \star \{ \mathbf{x}_i \hookrightarrow - \wedge \text{size} = 1 \mid \mathbf{x}_i \hookrightarrow - \subseteq_{\text{LIT}} \varphi, i \in [1, n] \} \\ \text{GARB}(\varphi) &\stackrel{\text{def}}{=} \begin{cases} \text{size} = \max_{\text{size}}(\varphi) - n_\varphi & \text{if } \max_{\text{size}}(\varphi) < \alpha \\ \text{size} \geq \max_{\text{size}}(\varphi) - n_\varphi & \text{otherwise} \end{cases} \end{aligned}$$

Notice that, since $\max_{\text{size}}(\varphi) \geq n_\varphi$, the formula $\text{GARB}(\varphi)$ is well-defined. The formulae $\text{ALLOC}(\psi)$ and $\text{GARB}(\psi)$ are defined accordingly. Obviously, ALLOC is equal to $\text{ALLOC}(\varphi) * \text{ALLOC}(\psi)$ modulo associativity and commutativity for the separating conjunction $*$. Hence, by taking advantage of the axioms (COM^*) and (ASSOC^*) , we have

$$\vdash_{\mathcal{H}_C(*)} \text{ALLOC} \Leftrightarrow \text{ALLOC}(\varphi) * \text{ALLOC}(\psi).$$

Let us now look at $\text{GARB}(\varphi)$ and $\text{GARB}(\psi)$. We aim at deriving

$$\vdash_{\mathcal{H}_C(*)} \text{GARB} \Rightarrow \text{GARB}(\varphi) * \text{GARB}(\psi).$$

Since, φ is a core type, we know that if $\max_{\text{size}}(\varphi) < \alpha$ then, by definition of $\max_{\text{size}}(\varphi)$, $\neg \text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi$. A similar analysis can be done for ψ , which leads to the two following equivalences, by definition of $\text{GARB}(\varphi)$ and $\text{GARB}(\psi)$:

- $\neg \text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi$ if and only if $\text{GARB}(\varphi) = (\text{size} = \max_{\text{size}}(\varphi) - n_\varphi)$,
- $\neg \text{size} \geq \max_{\text{size}}(\psi) + 1 \subseteq_{\text{LIT}} \psi$ if and only if $\text{GARB}(\psi) = (\text{size} = \max_{\text{size}}(\psi) - n_\psi)$.

By definition of GARB, (B) and (C), we know that $\text{GARB} = (\text{size} = \max_{\text{size}}(\langle *\rangle(\varphi, \psi)) \doteq n)$ holds if and only if $\neg \text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi$ and $\neg \text{size} \geq \max_{\text{size}}(\psi) + 1 \subseteq_{\text{LIT}} \psi$. From $n \leq \max_{\text{size}}(\langle *\rangle(\varphi, \psi))$ and by relying on the previous two equivalences, this allows us to conclude that:

D. $\text{GARB}(\varphi) = (\text{size} = \max_{\text{size}}(\varphi) - n_\varphi)$ and $\text{GARB}(\psi) = (\text{size} = \max_{\text{size}}(\psi) - n_\psi)$ if and only if $\text{GARB} = (\text{size} = \max_{\text{size}}(\langle *\rangle(\varphi, \psi)) - n)$.

To show $\vdash_{\mathcal{H}_C(*)} \text{GARB} \Rightarrow (\text{GARB}(\varphi) * \text{GARB}(\psi))$, we split the proof depending on whether $\text{GARB}(\varphi) = (\text{size} = \max_{\text{size}}(\varphi) - n_\varphi)$ and $\text{GARB}(\psi) = (\text{size} = \max_{\text{size}}(\psi) - n_\psi)$ hold.

case: $\text{GARB}(\varphi) \neq (\text{size} = \max_{\text{size}}(\varphi) - n_\varphi)$ and $\text{GARB}(\psi) \neq (\text{size} = \max_{\text{size}}(\psi) - n_\psi)$.

We have $\text{GARB}(\varphi) = (\text{size} \geq \max_{\text{size}}(\varphi) - n_\varphi)$ and $\text{GARB}(\psi) = (\text{size} \geq \max_{\text{size}}(\psi) - n_\psi)$. By definition of GARB and (D), $\text{GARB} = (\text{size} \geq \max_{\text{size}}(\langle *\rangle(\varphi, \psi)) - n)$. By $n = n_\varphi + n_\psi$ and (A), $\max_{\text{size}}(\langle *\rangle(\varphi, \psi)) - n = (\max_{\text{size}}(\varphi) - n_\varphi) + (\max_{\text{size}}(\psi) - n_\psi)$. By definition of the core formula $\text{size} \geq \beta$, GARB is already equivalent to $\text{GARB}(\varphi) * \text{GARB}(\psi)$, modulo associativity and commutativity for the separating conjunction $*$. Hence, by taking advantage of the axioms (COM^{*}) and (ASSOC^{*}), we have $\vdash_{\mathcal{H}_C(*)} \text{GARB} \Rightarrow \text{GARB}(\varphi) * \text{GARB}(\psi)$.

case: $\text{GARB}(\varphi) = (\text{size} = \max_{\text{size}}(\varphi) - n_\varphi)$ and $\text{GARB}(\psi) \neq (\text{size} = \max_{\text{size}}(\psi) - n_\psi)$.

We have $\text{GARB}(\psi) = (\text{size} \geq \max_{\text{size}}(\psi) - n_\psi)$ and, by definition of GARB and (D), together with $n = n_\varphi + n_\psi$ and (A), $\text{GARB} = (\text{size} \geq (\max_{\text{size}}(\varphi) - n_\varphi) + (\max_{\text{size}}(\psi) - n_\psi))$. In this case, $\text{GARB} \Rightarrow \text{GARB}(\varphi) * \text{GARB}(\psi)$ is an instantiation of the following valid formula with $\beta_1 = \max_{\text{size}}(\varphi) - n_\varphi$ and $\beta_2 = \max_{\text{size}}(\psi) - n_\psi$:

$$\text{size} \geq \beta_1 + \beta_2 \Rightarrow \text{size} = \beta_1 * \text{size} \geq \beta_2.$$

The derivability of this formula in $\mathcal{H}_C(*)$ is proven by induction on β_1 . The derivation for the base case $\beta_1 = 0$ is:

1	$\text{size} \geq \beta_2 \Rightarrow \text{emp} * \text{size} \geq \beta_2$	(ID [*])
2	$\text{emp} \Rightarrow \text{size} \geq 0 \wedge \neg \text{size} \geq 1$	PC, def. of $\text{size} \geq 1$
3	$\text{emp} * \text{size} \geq \beta_2 \Rightarrow \text{size} = 0 * \text{size} \geq \beta_2$	(*), 2, def. of $\text{size} = 0$
4	$\text{size} \geq \beta_2 \Rightarrow \text{size} = 0 * \text{size} \geq \beta_2$	($\Rightarrow \text{TR}$), 1, 3

For the induction step, let us suppose the formula to be derivable for a certain β_1 , and let us prove that it is also derivable for $\beta_1 + 1$.

1	$\text{size} \geq \beta_1 + 1 + \beta_2 \Rightarrow \text{size} \geq 1 * \text{size} \geq \beta_1 + \beta_2$	def. of $\text{size} \geq \beta$, (COM [*]), (ASSOC [*])
2	$\text{size} \geq 1 \Rightarrow \text{size} = 1 * \top$	(ATOM [*] 1), def. of $\text{size} \geq 1$
3	$\text{size} \geq 1 * \text{size} \geq \beta_1 + \beta_2 \Rightarrow$ $(\text{size} = 1 * \top) * \text{size} \geq \beta_1 + \beta_2$	(*), 2
4	$(\text{size} = 1 * \top) * \text{size} \geq \beta_1 + \beta_2 \Rightarrow$ $\text{size} = 1 * \text{size} \geq \beta_1 + \beta_2$	PC, (COM [*]), (ASSOC [*]), (MONO [*])
5	$\text{size} \geq \beta_1 + \beta_2 \Rightarrow \text{size} = \beta_1 * \text{size} \geq \beta_2$	Induction Hypothesis
6	$\text{size} = 1 * \text{size} \geq \beta_1 + \beta_2 \Rightarrow$	

	$(\text{size} = 1 * \text{size} = \beta_1) * \text{size} \geq \beta_2$	$(_{\text{COM}}^*), (*), ({}_{\text{ASSOC}}^*)$
7	$\text{size} = \tilde{\beta} \Rightarrow \text{size} \geq \tilde{\beta}$	PC, def. of $\text{size} = \tilde{\beta}$
8	$\text{size} = \tilde{\beta} \Rightarrow \neg \text{size} \geq \tilde{\beta} + 1$	PC, def. of $\text{size} = \tilde{\beta}$
9	$\text{size} = 1 * \text{size} = \beta_1 \Rightarrow \text{size} \geq 1 * \text{size} \geq \beta_1$	$(*I_{LR}), 7$
10	$\text{size} = 1 * \text{size} = \beta_1 \Rightarrow \neg \text{size} \geq 2 * \neg \text{size} \geq \beta_1 + 1$	$(*I_{LR}), 8$
11	$\text{size} \geq 1 * \text{size} \geq \beta_1 \Rightarrow \text{size} \geq \beta_1 + 1$	def. of $\text{size} \geq \beta$, $(_{\text{COM}}^*), ({}_{\text{ASSOC}}^*)$
12	$\neg \text{size} \geq 2 * \neg \text{size} \geq \beta_1 + 1 \Rightarrow \neg \text{size} \geq \beta_1 + 2$	$({}_{\neg \text{SIZE}}^*)$
13	$\text{size} = 1 * \text{size} = \beta_1 \Rightarrow \text{size} = \beta_1 + 1$	PC, 9–12, def. of $\text{size} = \beta_1$
14	$(\text{size} = 1 * \text{size} = \beta_1) * \text{size} \geq \beta_2 \Rightarrow$ $\text{size} = \beta_1 + 1 * \text{size} \geq \beta_2$	$(*), 13$
15	$\text{size} \geq \beta_1 + 1 + \beta_2 \Rightarrow \text{size} = \beta_1 + 1 * \text{size} \geq \beta_2$	$(\Rightarrow \text{TR}), 1, 3, 4, 6, 14$

case: $\text{GARB}(\varphi) \neq (\text{size} = \max_{\text{size}}(\varphi) - n_\varphi)$ and $\text{GARB}(\psi) = (\text{size} = \max_{\text{size}}(\psi) - n_\psi)$.

Analogous to the previous case. We have $\text{GARB}(\varphi) = (\text{size} \geq \max_{\text{size}}(\varphi) - n_\varphi)$ and $\text{GARB} = (\text{size} \geq (\max_{\text{size}}(\varphi) - n_\varphi) + (\max_{\text{size}}(\psi) - n_\psi))$. We instantiate the theorem

$$\text{size} \geq \beta_1 + \beta_2 \Rightarrow \text{size} = \beta_1 * \text{size} \geq \beta_2,$$

shown derivable in the previous case of the proof, with $\beta_1 = \max_{\text{size}}(\psi) - n_\psi$ and $\beta_2 = \max_{\text{size}}(\varphi) - n_\varphi$. This corresponds to $\text{GARB} \Rightarrow \text{GARB}(\varphi) * \text{GARB}(\psi)$. Then, by commutativity of the separating conjunction (axiom $(_{\text{COM}}^*)$) and propositional reasoning, we conclude that $\vdash_{\mathcal{H}_C(*)} \text{GARB} \Rightarrow \text{GARB}(\varphi) * \text{GARB}(\psi)$.

case: $\text{GARB}(\varphi) = (\text{size} = \max_{\text{size}}(\varphi) - n_\varphi)$ and $\text{GARB}(\psi) = (\text{size} = \max_{\text{size}}(\psi) - n_\psi)$.

By (D), $n = n_\varphi + n_\psi$ and (A), $\text{GARB} = (\text{size} = (\max_{\text{size}}(\varphi) - n_\varphi) + (\max_{\text{size}}(\psi) - n_\psi))$. In this case, $\text{GARB} \Rightarrow \text{GARB}(\varphi) * \text{GARB}(\psi)$ is an instantiation of the following valid formula, with $\beta_1 = \max_{\text{size}}(\varphi) - n_\varphi$ and $\beta_2 = \max_{\text{size}}(\psi) - n_\psi$:

$$\text{size} = \beta_1 + \beta_2 \Rightarrow \text{size} = \beta_1 * \text{size} = \beta_2.$$

Here is the derivation of this formula:

1	$\text{size} = \beta_1 + \beta_2 \Rightarrow \text{size} \geq \beta_1 + \beta_2$	PC, def. of $\text{size} = \beta$
2	$\text{size} \geq \beta_1 + \beta_2 \Rightarrow \text{size} = \beta_1 * \text{size} \geq \beta_2$	Previously derived
3	$\text{size} \geq \beta_2 \Rightarrow (\text{size} \geq \beta_2 \wedge \text{size} \geq \beta_2 + 1) \vee \text{size} = \beta_2$	PC, def. of $\text{size} = \beta_2$
4	$\text{size} = \beta_1 * \text{size} \geq \beta_2 \Rightarrow$ $\text{size} = \beta_1 * ((\text{size} \geq \beta_2 \wedge \text{size} \geq \beta_2 + 1) \vee \text{size} = \beta_2)$	$(_{\text{COM}}^*), (*), 3$
5	$\text{size} = \beta_1 * ((\text{size} \geq \beta_2 \wedge \text{size} \geq \beta_2 + 1) \vee \text{size} = \beta_2) \Rightarrow$ $(\text{size} = \beta_1 * (\text{size} \geq \beta_2 \wedge \text{size} \geq \beta_2 + 1)) \vee (\text{size} = \beta_1 * \text{size} = \beta_2)$	$(_{\text{COM}}^*), (I_3^*)$
6	$\text{size} \geq \tilde{\beta} \wedge \chi \Rightarrow \text{size} \geq \tilde{\beta}$	PC
7	$\text{size} = \beta_1 * (\text{size} \geq \beta_2 \wedge \text{size} \geq \beta_2 + 1) \Rightarrow$ $\text{size} \geq \beta_1 * \text{size} \geq \beta_2 + 1$	PC, $(*I_{LR}), 6$

8	$\mathbf{size} \geq \beta_1 * \mathbf{size} \geq \beta_2 + 1 \Rightarrow \mathbf{size} \geq \beta_1 + \beta_2 + 1$	$(\mathbf{COM}^*), (\mathbf{ASSOC}^*)$
9	$\mathbf{size} = \beta_1 * (\mathbf{size} \geq \beta_2 \wedge \mathbf{size} \geq \beta_2 + 1) \Rightarrow \mathbf{size} \geq \beta_1 + \beta_2 + 1$	$(\Rightarrow \mathbf{TR}), 7, 8$
10	$\mathbf{size} = \beta_1 * ((\mathbf{size} \geq \beta_2 \wedge \mathbf{size} \geq \beta_2 + 1) \vee \mathbf{size} = \beta_2) \Rightarrow$ $\mathbf{size} \geq \beta_1 + \beta_2 + 1 \vee (\mathbf{size} = \beta_1 * \mathbf{size} = \beta_2)$	PC, 5, 9
11	$\mathbf{size} = \beta_1 + \beta_2 \Rightarrow \mathbf{size} \geq \beta_1 + \beta_2 + 1 \vee (\mathbf{size} = \beta_1 * \mathbf{size} = \beta_2)$	$(\Rightarrow \mathbf{TR}), 1, 2, 4, 10$
12	$\mathbf{size} = \beta_1 + \beta_2 \Rightarrow \neg \mathbf{size} \geq \beta_1 + \beta_2 + 1$	PC, def. of $\mathbf{size} = \beta$
13	$\mathbf{size} = \beta_1 + \beta_2 \Rightarrow \mathbf{size} = \beta_1 * \mathbf{size} = \beta_2$	PC, 11, 12

Thanks to the case analysis above, we conclude that $\vdash_{\mathcal{H}_C(*)} \mathbf{GARB} \Rightarrow \mathbf{GARB}(\varphi) * \mathbf{GARB}(\psi)$. Thus, $\vdash_{\mathcal{H}_C(*)} \mathbf{ALLOC} * \mathbf{GARB} \Rightarrow (\mathbf{ALLOC}(\varphi) * \mathbf{GARB}(\varphi)) * (\mathbf{ALLOC}(\psi) * \mathbf{GARB}(\psi))$. Indeed,

1	$\mathbf{ALLOC} \Rightarrow \mathbf{ALLOC}(\varphi) * \mathbf{ALLOC}(\psi)$	Previously derived
2	$\mathbf{GARB} \Rightarrow \mathbf{GARB}(\varphi) * \mathbf{GARB}(\psi)$	Previously derived
3	$\mathbf{ALLOC} * \mathbf{GARB} \Rightarrow (\mathbf{ALLOC}(\varphi) * \mathbf{ALLOC}(\psi)) * (\mathbf{GARB}(\varphi) * \mathbf{GARB}(\psi))$	$(\mathbf{I}_{LR}^*), 1, 2$
4	$(\mathbf{ALLOC}(\varphi) * \mathbf{ALLOC}(\psi)) * (\mathbf{GARB}(\varphi) * \mathbf{GARB}(\psi)) \Rightarrow$ $(\mathbf{ALLOC}(\varphi) * \mathbf{GARB}(\varphi)) * (\mathbf{ALLOC}(\psi) * \mathbf{GARB}(\psi))$	$(\mathbf{COM}^*), (\mathbf{ASSOC}^*)$
5	$\mathbf{ALLOC} * \mathbf{GARB} \Rightarrow (\mathbf{ALLOC}(\varphi) * \mathbf{GARB}(\varphi)) * (\mathbf{ALLOC}(\psi) * \mathbf{GARB}(\psi))$	$(\Rightarrow \mathbf{TR}), 3, 4$

To conclude this step of the proof, it is sufficient to show $\vdash_{\mathcal{H}_C(*)} \mathbf{ALLOC}(\varphi) * \mathbf{GARB}(\varphi) \Rightarrow \varphi^{(1)}$ and $\vdash_{\mathcal{H}_C(*)} \mathbf{ALLOC}(\psi) * \mathbf{GARB}(\psi) \Rightarrow \psi^{(1)}$. Indeed, by relying on the rule (\mathbf{I}_{LR}^*) , we then obtain $\vdash_{\mathcal{H}_C(*)} \mathbf{ALLOC} * \mathbf{GARB} \Rightarrow \varphi^{(1)} * \psi^{(1)}$. Below, we show $\vdash_{\mathcal{H}_C(*)} \mathbf{ALLOC}(\varphi) * \mathbf{GARB}(\varphi) \Rightarrow \varphi^{(1)}$. The developments of $\vdash_{\mathcal{H}_C(*)} \mathbf{ALLOC}(\psi) * \mathbf{GARB}(\psi) \Rightarrow \psi^{(1)}$ are analogous. We recall that the formula $\mathbf{ALLOC}(\varphi)$ is defined as

$$\mathbf{ALLOC}(\varphi) = \mathbf{*}\{\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1 \mid \mathbf{x}_i \hookrightarrow _ \subseteq_{\mathbf{LIT}} \varphi\}.$$

First of all, let us show that $\vdash_{\mathcal{H}_C(*)} \mathbf{ALLOC}(\varphi) * \top \Rightarrow \bigwedge \{\mathbf{x}_i \hookrightarrow _ \subseteq_{\mathbf{LIT}} \varphi \mid i \in [1, n]\}$. The proof is divided in three cases:

case: $\{\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1 \mid \mathbf{x}_i \hookrightarrow _ \subseteq_{\mathbf{LIT}} \varphi\} = \emptyset$. In this case, the formula we want to derive syntactically equal to is $\top * \top \Rightarrow \top$, which is derivable by propositional reasoning.

case: $\text{card}(\{\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1 \mid \mathbf{x}_i \hookrightarrow _ \subseteq_{\mathbf{LIT}} \varphi\}) = 1$. In this case, the formula we want to derive is syntactically equal to $(\mathbf{x} \hookrightarrow _ \wedge \mathbf{size} = 1) * \top \Rightarrow \mathbf{x} \hookrightarrow _$. Therefore, it is derivable in $\mathcal{H}_C(*)$ by (I_5^*) and $(*)$.

case: $\text{card}(\{\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1 \mid \mathbf{x}_i \hookrightarrow _ \subseteq_{\mathbf{LIT}} \varphi\}) \geq 2$. In the derivation below, $\mathbf{ALLOC}(\varphi)^{-i}$ stands for $\mathbf{*}\{\mathbf{x}_j \hookrightarrow _ \wedge \mathbf{size} = 1 \mid j \in [1, n] \setminus \{i\}, \mathbf{x}_j \hookrightarrow _ \subseteq_{\mathbf{LIT}} \varphi\}$. Roughly speaking, $\mathbf{ALLOC}(\varphi)^{-i}$ is obtained from $\mathbf{ALLOC}(\varphi)$ by removing the subformula $\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1$. Since $\text{card}(\{\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1 \mid \mathbf{x}_i \hookrightarrow _ \subseteq_{\mathbf{LIT}} \varphi\}) \geq 2$, the formula $\mathbf{ALLOC}(\varphi)^{-i}$ is different from \top . We have,

1	$\mathbf{ALLOC}(\varphi) * \top \Rightarrow (\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1) * (\mathbf{ALLOC}(\varphi)^{-i} * \top)$	$(\mathbf{COM}^*), (\mathbf{ASSOC}^*), \text{def. of } \mathbf{ALLOC}(\varphi)$ where $\mathbf{x}_i \hookrightarrow _ \subseteq_{\mathbf{LIT}} \varphi$ and $i \in [1, n]$
2	$\mathbf{ALLOC}(\varphi)^{-i} * \top \Rightarrow \top$	PC

3	$\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1 \Rightarrow \mathbf{x}_i \hookrightarrow _$	PC
4	$(\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1) * (\text{ALLOC}(\varphi)^{-i} * \top) \Rightarrow$ $\mathbf{x}_i \hookrightarrow _ * \top$	(*I _{LR}), 2, 3
5	$\mathbf{x}_i \hookrightarrow _ * \top \Rightarrow \mathbf{x}_i \hookrightarrow _$	(I ₅ [*])
6	$\text{ALLOC}(\varphi) * \top \Rightarrow \mathbf{x}_i \hookrightarrow _$	(⇒TR), 1, 4, 5
7	$\text{ALLOC}(\varphi) * \top \Rightarrow \bigwedge \{\mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\}$	PC, repeating 6 for all $i \in [1, n]$ s.t. $\mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$

So, we have $\vdash_{\mathcal{H}_C(*)} \text{ALLOC}(\varphi) * \top \Rightarrow \bigwedge \{\mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\}$.

Recall that $\text{card}(\{i \in [1, n] \mid \mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi\}) = n_\varphi$. At the beginning of the proof, we have shown a derivation of $\vdash_{\mathcal{H}_C(*)} \text{ALLOC} \Rightarrow \mathbf{size} = n$, where ALLOC is defined as $*\{\mathbf{x}_i \hookrightarrow _ \wedge \mathbf{size} = 1 \mid i \in [1, n]\}$. Replacing ALLOC by $\text{ALLOC}(\varphi)$ and n by n_φ in the derivation of $\text{ALLOC} \Rightarrow \mathbf{size} = n$ leads to a derivation in $\mathcal{H}_C(*)$ of $\text{ALLOC}(\varphi) \Rightarrow \mathbf{size} = n_\varphi$.

To show $\vdash_{\mathcal{H}_C(*)} \text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \varphi^{(1)}$, we split the proof in two cases:

case: $\max_{\mathbf{size}}(\varphi) = \alpha$. By definition of $\varphi^{(1)}$ and GARB(φ), we have:

- $\varphi^{(1)} = \mathbf{size} \geq \max_{\mathbf{size}}(\varphi) \wedge \bigwedge \{\mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\}$,
- $\text{GARB}(\varphi) = \mathbf{size} \geq \max_{\mathbf{size}}(\varphi) - n_\varphi$,

Then,

1	$\text{ALLOC}(\varphi) * \top \Rightarrow \bigwedge \{\mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\}$	Previously derived
2	$\text{GARB}(\varphi) \Rightarrow \top$	PC
3	$\text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \text{ALLOC}(\varphi) * \top$	(*), (* _{COM}), 2
4	$\text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \bigwedge \{\mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\}$	(⇒TR), 1, 3
5	$\text{ALLOC}(\varphi) \Rightarrow \mathbf{size} = n_\varphi$	See above
6	$\mathbf{size} = n_\varphi \Rightarrow \mathbf{size} \geq n_\varphi$	PC, def. of $\mathbf{size} = n_\varphi$
7	$\text{ALLOC}(\varphi) \Rightarrow \mathbf{size} \geq n_\varphi$	
8	$\text{GARB}(\varphi) \Rightarrow \mathbf{size} \geq \max_{\mathbf{size}}(\varphi) - n_\varphi$	PC, def. of GARB(φ)
9	$\text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow$ $\mathbf{size} \geq n_\varphi * \mathbf{size} \geq \max_{\mathbf{size}}(\varphi) - n_\varphi$	(*I _{LR}), 7, 8
10	$\mathbf{size} \geq n_\varphi * \mathbf{size} \geq \max_{\mathbf{size}}(\varphi) - n_\varphi \Rightarrow$ $\mathbf{size} \geq \max_{\mathbf{size}}(\varphi)$	(* _{ASSOC}), (* _{COM}), def. of $\mathbf{size} \geq \beta$
11	$\text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \mathbf{size} \geq \max_{\mathbf{size}}(\varphi)$	(⇒TR), 9, 10
12	$\text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \varphi^{(1)}$	PC, 4, 11, def. of $\varphi^{(1)}$

case: $\max_{\mathbf{size}}(\varphi) \neq \alpha$. In this case, $\max_{\mathbf{size}}(\varphi) < \alpha$ and so we have:

- $\varphi^{(1)} = \mathbf{size} = \max_{\mathbf{size}}(\varphi) \wedge \bigwedge \{\mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\}$,
- $\text{GARB}(\varphi) = \mathbf{size} = \max_{\mathbf{size}}(\varphi) - n_\varphi$,

We can rely on the previous case of the proof in order to show that

$$\vdash_{\mathcal{H}_C(*)} \text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \text{size} \geq \max_{\text{size}}(\varphi) \wedge \bigwedge \{\mathbf{x}_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi \mid i \in [1, n]\}.$$

By propositional reasoning, we can derive $\vdash_{\mathcal{H}_C(*)} \text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \varphi^{(1)}$ as soon as we show that $\vdash_{\mathcal{H}_C(*)} \text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \neg \text{size} \geq \max_{\text{size}}(\varphi) + 1$, as we do now:

1	$\text{ALLOC}(\varphi) \Rightarrow \text{size} = n_\varphi$	Already discussed above
2	$\text{size} = n_\varphi \Rightarrow \neg \text{size} \geq n_\varphi + 1$	PC, def. of $\text{size} = n_\varphi$
3	$\text{ALLOC}(\varphi) \Rightarrow \neg \text{size} = n_\varphi + 1$	PC, ($\Rightarrow \text{TR}$), 1, 2
4	$\text{GARB}(\varphi) \Rightarrow \neg \text{size} \geq \max_{\text{size}}(\varphi) - n_\varphi + 1$	PC, def. of $\text{size} = \beta$
5	$\text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow$ $\neg \text{size} \geq n_\varphi + 1 * \neg \text{size} \geq \max_{\text{size}}(\varphi) - n_\varphi + 1$	(*I _{LR}), 3, 4
6	$\neg \text{size} \geq n_\varphi + 1 * \neg \text{size} \geq \max_{\text{size}}(\varphi) - n_\varphi + 1 \Rightarrow$ $\neg \text{size} \geq \max_{\text{size}}(\varphi) + 1$	($\neg \text{size}$)
7	$\text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \neg \text{size} \geq \max_{\text{size}}(\varphi) + 1$	($\Rightarrow \text{TR}$), 5, 6

This concludes the proof of $\vdash_{\mathcal{H}_C(*)} \text{ALLOC}(\varphi) * \text{GARB}(\varphi) \Rightarrow \varphi^{(1)}$. As already stated, one can analogously show that $\vdash_{\mathcal{H}_C(*)} \text{ALLOC}(\psi) * \text{GARB}(\psi) \Rightarrow \psi^{(1)}$. Afterwards, by (*I_{LR}) and from $\vdash_{\mathcal{H}_C(*)} \text{ALLOC} * \text{GARB} \Rightarrow (\text{ALLOC}(\varphi) * \text{GARB}(\varphi)) * (\text{ALLOC}(\psi) * \text{GARB}(\psi))$, we conclude that

$$\vdash_{\mathcal{H}_C(*)} \text{ALLOC} * \text{GARB} \Rightarrow \varphi^{(1)} * \psi^{(1)}.$$

Step 3, add the missing literals. From the first and second step of the proof, and by propositional reasoning, $\vdash_{\mathcal{H}_C(*)} \langle \ast \rangle(\varphi, \psi) \Rightarrow \varphi^{(1)} * \psi^{(1)}$. We now rely on $\langle \ast \rangle(\varphi, \psi)$ to add to $\varphi^{(1)}$ and $\psi^{(1)}$ missing literals from φ and ψ , respectively. We add the literals progressively, building a sequence of formulae $\varphi^{(1)} * \psi^{(1)}, \varphi^{(2)} * \psi^{(2)}, \dots, \varphi^{(k)} * \psi^{(k)}$, where for all $i \in [1, k]$, $\varphi^{(i)}$ and $\psi^{(i)}$ are conjunctions of core formulae such that $\vdash_{\mathcal{H}_C(*)} \langle \ast \rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$, and for all $j \in [1, i]$, $\varphi^{(j)} \subseteq_{\text{LIT}} \varphi^{(i)}$ and $\psi^{(j)} \subseteq_{\text{LIT}} \psi^{(i)}$. Fundamentally, we obtain $\varphi = \varphi^{(k)}$ and $\psi = \psi^{(k)}$ (modulo associativity and commutativity of the classical conjunction), which allows us to derive $\vdash_{\mathcal{H}_C(*)} \langle \ast \rangle(\varphi, \psi) \Rightarrow \varphi * \psi$, ending the proof. Below, we focus on the formula $\varphi^{(i)}$ and φ . Since $\langle \ast \rangle(\varphi, \psi)$ is equal to $\langle \ast \rangle(\psi, \varphi)$ (by definition) and the separating conjunction is commutative (axiom (*_{COM})), a similar analysis can be done for $\psi^{(i)}$ and ψ . Thus, we assume that $\vdash_{\mathcal{H}_C(*)} \langle \ast \rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$ holds, where in particular $\varphi^{(1)} \subseteq_{\text{LIT}} \varphi^{(i)}$ and $\psi^{(1)} \subseteq_{\text{LIT}} \psi^{(i)}$, and that there is a literal $L \subseteq_{\text{LIT}} \varphi$ that does not appear in $\varphi^{(i)}$. By relying on the theorems in Lemma 6.12, we show that $\vdash_{\mathcal{H}_C(*)} \langle \ast \rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge L) * \psi^{(i)}$ by a case analysis on L .

case: $L = \mathbf{x} \sim \mathbf{y}$, **where** $\sim \in \{=, \neq\}$. By definition of $\langle \ast \rangle(\varphi, \psi)$, $\mathbf{x} \sim \mathbf{y} \subseteq_{\text{LIT}} \langle \ast \rangle(\varphi, \psi)$.

1	$\langle \ast \rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
2	$\langle \ast \rangle(\varphi, \psi) \Rightarrow \mathbf{x} \sim \mathbf{y}$	PC, def. of $\langle \ast \rangle(\varphi, \psi)$, see above
3	$\langle \ast \rangle(\varphi, \psi) \Rightarrow \mathbf{x} \sim \mathbf{y} \wedge (\varphi^{(i)} * \psi^{(i)})$	PC, 1, 2
4	$\mathbf{x} \sim \mathbf{y} \wedge (\varphi^{(i)} * \psi^{(i)}) \Rightarrow (\varphi^{(i)} \wedge \mathbf{x} \sim \mathbf{y}) * \psi^{(i)}$	(I _{6.12.1} [*])
5	$\langle \ast \rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge \mathbf{x} \sim \mathbf{y}) * \psi^{(i)}$	($\Rightarrow \text{TR}$), 3, 4

case: $L = \mathbf{x} \hookrightarrow __$. Since $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi$, by definition, $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \langle * \rangle(\varphi, \psi)$. By definition of $\mathbf{x}_1, \dots, \mathbf{x}_n$, there is $j \in [1, n]$ such that $\mathbf{x}_j = \mathbf{x} \subseteq_{\text{LIT}} \langle * \rangle(\varphi, \psi)$. Since φ is a core type, $\mathbf{x}_j \hookrightarrow __ \subseteq_{\text{LIT}} \varphi$. By definition of $\varphi^{(1)}$, $\mathbf{x}_j \hookrightarrow __ \subseteq_{\text{LIT}} \varphi^{(1)}$. From $\varphi^{(1)} \subseteq_{\text{LIT}} \varphi^{(i)}$, we have $\mathbf{x}_j \hookrightarrow __ \subseteq_{\text{LIT}} \varphi^{(i)}$. Then,

1	$\varphi^{(i)} \Rightarrow \varphi^{(i)} \wedge \mathbf{x}_j \hookrightarrow __$	PC, see above
2	$\langle * \rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
3	$\varphi^{(i)} * \psi^{(i)} \Rightarrow (\varphi^{(i)} \wedge \mathbf{x}_j \hookrightarrow __) * \psi^{(i)}$	(*) ₁
4	$\langle * \rangle(\varphi, \psi) \Rightarrow \mathbf{x}_j = \mathbf{x}$	PC, see above
5	$\langle * \rangle(\varphi, \psi) \Rightarrow \mathbf{x}_j = \mathbf{x} \wedge ((\varphi^{(i)} \wedge \mathbf{x}_j \hookrightarrow __) * \psi^{(i)})$	PC, 2, 3, 4
6	$\mathbf{x}_j = \mathbf{x} \wedge ((\varphi^{(i)} \wedge \mathbf{x}_j \hookrightarrow __) * \psi^{(i)}) \Rightarrow (\varphi^{(i)} \wedge \mathbf{x} \hookrightarrow __) * \psi^{(i)}$	(I _{6.12.2} [*])
7	$\langle * \rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge \mathbf{x} \hookrightarrow __) * \psi^{(i)}$	(⇒TR), 5, 6

Without loss of generality, thanks to the derivation above dealing with $\mathbf{x} \hookrightarrow __$ literals, we assume that for all $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi$ and all $\mathbf{y} \hookrightarrow __ \subseteq_{\text{LIT}} \psi$, we have $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi^{(i)}$ and $\mathbf{y} \hookrightarrow __ \subseteq_{\text{LIT}} \psi^{(i)}$.

case: $L = \neg \mathbf{x} \hookrightarrow __$. We distinguish two main subcases.

- First, assume $\neg \mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \psi$. By definition of $\langle * \rangle(\varphi, \psi)$, $\neg \mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \langle * \rangle(\varphi, \psi)$.

1	$\langle * \rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
2	$\langle * \rangle(\varphi, \psi) \Rightarrow \neg \mathbf{x} \hookrightarrow __$	PC, def. of $\langle * \rangle(\varphi, \psi)$, see above
3	$\langle * \rangle(\varphi, \psi) \Rightarrow \neg \mathbf{x} \hookrightarrow __ \wedge (\varphi^{(i)} * \psi^{(i)})$	PC, 1, 2
4	$\neg \mathbf{x} \hookrightarrow __ \wedge (\varphi^{(i)} * \psi^{(i)}) \Rightarrow (\varphi^{(i)} \wedge \neg \mathbf{x} \hookrightarrow __) * \psi^{(i)}$	(I _{6.12.4} [*])
5	$\langle * \rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge \neg \mathbf{x} \hookrightarrow __) * \psi^{(i)}$	(⇒TR), 3, 4

- Otherwise, $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \psi$. By assumption, $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \psi^{(i)}$.

1	$\psi^{(i)} \Rightarrow \psi^{(i)} \wedge \mathbf{x} \hookrightarrow __$	PC, see above
2	$\langle * \rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
3	$\varphi^{(i)} * \psi^{(i)} \Rightarrow (\psi^{(i)} \wedge \mathbf{x} \hookrightarrow __) * \varphi^{(i)}$	(_{COM} [*]), (*), 1
4	$(\psi^{(i)} \wedge \mathbf{x} \hookrightarrow __) * \varphi^{(i)} \Rightarrow \psi^{(i)} * (\varphi^{(i)} \wedge \neg \mathbf{x} \hookrightarrow __)$	(I _{6.12.3} [*])
5	$\psi^{(i)} * (\varphi^{(i)} \wedge \neg \mathbf{x} \hookrightarrow __) \Rightarrow (\varphi^{(i)} \wedge \neg \mathbf{x} \hookrightarrow __) * \psi^{(i)}$	(_{COM} [*])
6	$\langle * \rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge \neg \mathbf{x} \hookrightarrow __) * \psi^{(i)}$	(⇒TR), 2, 3, 4, 5

case: $L = \mathbf{x} \hookrightarrow \mathbf{y}$. Similar to the case $L = \mathbf{x} \hookrightarrow __$. Since φ is a satisfiable core type, we have $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi$ (see axiom (_{WEAK}[→])). By assumption, $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi^{(i)}$. By definition of $\langle * \rangle(\varphi, \psi)$, we have $\mathbf{x} \hookrightarrow \mathbf{y} \subseteq_{\text{LIT}} \langle * \rangle(\varphi, \psi)$.

1	$\varphi^{(i)} \Rightarrow \varphi^{(i)} \wedge \mathbf{x} \hookrightarrow __$	PC, see above
---	--	---------------

2	$\langle *\rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
3	$\langle *\rangle(\varphi, \psi) \Rightarrow x \leftrightarrow y$	PC, see above
4	$\varphi^{(i)} * \psi^{(i)} \Rightarrow (\varphi^{(i)} \wedge x \leftrightarrow _) * \psi^{(i)}$	($*$), 1
5	$\langle *\rangle(\varphi, \psi) \Rightarrow x \leftrightarrow y \wedge ((\varphi^{(i)} \wedge x \leftrightarrow _) * \psi^{(i)})$	PC, 3, 4
6	$x \leftrightarrow y \wedge ((\varphi^{(i)} \wedge x \leftrightarrow _) * \psi^{(i)}) \Rightarrow (\varphi^{(i)} \wedge x \leftrightarrow y) * \psi^{(i)}$	($I_{6.12.6}^*$)
7	$\langle *\rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge x \leftrightarrow y) * \psi^{(i)}$	($*$), 5, 6

Without loss of generality, thanks to the previous cases dealing with $\neg x \leftrightarrow _$ literals, from now on we assume that for every $\neg x \leftrightarrow _ \subseteq_{\text{LIT}} \varphi$ and every $\neg y \leftrightarrow _ \subseteq_{\text{LIT}} \psi$, we have $\neg x \leftrightarrow _ \subseteq_{\text{LIT}} \varphi^{(i)}$ and $\neg y \leftrightarrow _ \subseteq_{\text{LIT}} \psi^{(i)}$.

case: $L = \neg x \leftrightarrow y$. We distinguish two main subcases

- First, suppose $x \leftrightarrow _ \subseteq_{\text{LIT}} \varphi$. In this case, by definition of $\langle *\rangle(\varphi, \psi)$, we have $\neg x \leftrightarrow y \subseteq_{\text{LIT}} \langle *\rangle(\varphi, \psi)$. Therefore,

1	$\langle *\rangle(\varphi, \psi) \Rightarrow \neg x \leftrightarrow y$	PC, see above
2	$\langle *\rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
3	$\langle *\rangle(\varphi, \psi) \Rightarrow \neg x \leftrightarrow y \wedge (\varphi^{(i)} * \psi^{(i)})$	PC, 1, 2
4	$\neg x \leftrightarrow y \wedge (\varphi^{(i)} * \psi^{(i)}) \Rightarrow (\varphi^{(i)} \wedge \neg x \leftrightarrow y) * \psi^{(i)}$	($I_{6.12.7}^*$)

- Otherwise, we have $\neg x \leftrightarrow _ \subseteq_{\text{LIT}} \varphi$. By assumption, $\neg x \leftrightarrow _ \subseteq_{\text{LIT}} \varphi^{(i)}$, and thus

1	$\varphi^{(i)} \Rightarrow \neg x \leftrightarrow _$	PC, see above
2	$\neg x \leftrightarrow _ \Rightarrow \neg x \leftrightarrow y$	($\xleftrightarrow{\text{WEAK}}$), PC
3	$\varphi^{(i)} \Rightarrow \neg x \leftrightarrow y$	($\Rightarrow \text{TR}$), 1, 2
4	$\varphi^{(i)} \Rightarrow \varphi^{(i)} \wedge \neg x \leftrightarrow y$	PC, 3
5	$\langle *\rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
6	$\varphi^{(i)} * \psi^{(i)} \Rightarrow (\varphi^{(i)} \wedge \neg x \leftrightarrow y) * \psi^{(i)}$	($*$), 4
7	$\langle *\rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge \neg x \leftrightarrow y) * \psi^{(i)}$	($\Rightarrow \text{TR}$), 5, 6

case: $L = \text{size} \geq \beta$. By definition of $\text{max}_{\text{size}}(\cdot)$, we have $\beta \leq \text{max}_{\text{size}}(\varphi)$. By definition of $\varphi^{(1)}$, $\text{size} \geq \text{max}_{\text{size}}(\varphi) \subseteq_{\text{LIT}} \varphi^{(1)}$. From $\varphi^{(1)} \subseteq_{\text{LIT}} \varphi^{(i)}$, we get $\text{size} \geq \text{max}_{\text{size}}(\varphi) \subseteq_{\text{LIT}} \varphi^{(i)}$.

1	$\varphi^{(i)} \Rightarrow \text{size} \geq \text{max}_{\text{size}}(\varphi)$	PC, see above
2	$\text{size} \geq \text{max}_{\text{size}}(\varphi) \Rightarrow \text{size} \geq \beta$	repeated (I_1^C), PC, as $\beta \leq \text{max}_{\text{size}}(\varphi)$
3	$\varphi^{(i)} \Rightarrow \varphi^{(i)} \wedge \text{size} \geq \beta$	PC, 1, 2
4	$\langle *\rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
5	$\varphi^{(i)} * \psi^{(i)} \Rightarrow (\varphi^{(i)} \wedge \text{size} \geq \beta) * \psi^{(i)}$	($*$), 3

$$6 \mid \langle * \rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge \text{size} \geq \beta) * \psi^{(i)} \quad (\Rightarrow \text{TR}), 4, 5$$

case: $L = \neg \text{size} \geq \beta$. In this case, $\max_{\text{size}}(\varphi) < \alpha$. Since φ is a satisfiable core type, we have $\beta > \max_{\text{size}}(\varphi)$. Moreover, by definition of $\varphi^{(1)}$, $\neg \text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi^{(1)}$. From $\varphi^{(1)} \subseteq_{\text{LIT}} \varphi^{(i)}$, we have $\neg \text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi^{(i)}$.

1	$\varphi^{(i)} \Rightarrow \neg \text{size} \geq \max_{\text{size}}(\varphi) + 1$	PC, see above
2	$\neg \text{size} \geq \max_{\text{size}}(\varphi) + 1 \Rightarrow \neg \text{size} \geq \beta$	repeated (I_1^C) , PC, as $\beta > \max_{\text{size}}(\varphi)$
3	$\varphi^{(i)} \Rightarrow \varphi^{(i)} \wedge \neg \text{size} \geq \beta$	by PC, the contrapositive of (I_1^C) is derivable
4	$\langle * \rangle(\varphi, \psi) \Rightarrow \varphi^{(i)} * \psi^{(i)}$	Hypothesis
5	$\varphi^{(i)} * \psi^{(i)} \Rightarrow (\varphi^{(i)} \wedge \neg \text{size} \geq \beta) * \psi^{(i)}$	$(*)$, 3
6	$\langle * \rangle(\varphi, \psi) \Rightarrow (\varphi^{(i)} \wedge \neg \text{size} \geq \beta) * \psi^{(i)}$	$(\Rightarrow \text{TR})$, 4, 5

□

Corollary 6.15 (*-simulation). Let $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq \text{card}(X)$. Let $\varphi, \psi \in \text{CoreTypes}(X, \alpha)$. There is χ in $\text{Conj}(\text{Core}(X, 2\alpha))$ such that $\vdash_{\mathcal{H}_C(*)} \varphi * \psi \Leftrightarrow \chi$.

Proof. If both φ and ψ are satisfiable, the results holds directly by Lemma 6.14, as $\langle * \rangle(\varphi, \psi)$ is in $\text{Conj}(\text{Core}(X, 2\alpha))$. Otherwise, let us treat the case where one of the two formulas is unsatisfiable. For instance, assume that φ is unsatisfiable. Then $\vdash_{\mathcal{H}_C} \varphi \Rightarrow \perp$ by completeness of \mathcal{H}_C (Lemma 6.8) and, ad $\mathcal{H}_C(*)$ includes \mathcal{H}_C , $\vdash_{\mathcal{H}_C(*)} \varphi \Rightarrow \perp$. By the rule $(*)$ and by the axiom (I_4^*) , we get $\vdash_{\mathcal{H}_C(*)} \varphi * \psi \Rightarrow \perp$. Thus χ can take the value $\neg(x = x)$. The case where ψ is not satisfiable is analogous, thanks to (COM^*) . □

By the distributivity axiom (I_3^*) , Corollary 6.15 is extended from core types to arbitrary Boolean combinations of core formulae. $\mathcal{H}_C(*)$ is therefore complete for $\text{SL}(*, x \hookrightarrow _)$. In order to derive a valid formula $\varphi \in \text{SL}(*, x \hookrightarrow _)$, we repeatedly apply the *-simulation in a bottom-up fashion, starting from the leaves of φ (which are Boolean combinations of core formulae) and obtaining a Boolean combination of core formulae ψ that is equivalent to φ . Then, we rely on the completeness of \mathcal{H}_C (Theorem 6.9) to prove that ψ is derivable.

Theorem 6.16. $\mathcal{H}_C(*)$ is an adequate proof system for $\text{SL}(*, x \hookrightarrow _)$.

In order to prove Theorem 6.16, we first show that the substitution of equivalent formulae in holds true. Formally, we show that the following rule is admissible in $\mathcal{H}_C(*)$:

$$(S_*) \quad \frac{\psi \Leftrightarrow \chi}{\varphi[\psi]_\rho \Leftrightarrow \varphi[\chi]_\rho}$$

where φ, ψ, χ are in $\text{SL}(*, x \hookrightarrow _)$, and $\varphi[\psi]_\rho$ refers to the formula φ in which the subformula at the occurrence ρ is replaced by ψ (see Definition 1.4).

Proof of (S_) .* The admissibility of (S_*) is shown with a standard structural induction on φ .

base case: φ atomic formula. In this case, the only position ρ of φ is ϵ (see Definition 1.4), and thus $\varphi[\psi]_\rho = \psi$ and $\varphi[\chi]_\rho = \chi$. From $\vdash_{\mathcal{H}_C(*)} \psi \Leftrightarrow \chi$, we derive $\vdash_{\mathcal{H}_C(*)} \varphi[\psi]_\rho \Leftrightarrow \varphi[\chi]_\rho$.

induction step: $\varphi = \neg\varphi'$. If $\rho = \epsilon$, then the result follows as in the base case. Otherwise, $\rho = 1\rho'$, for some position ρ' of φ' . By induction hypothesis, $\vdash_{\mathcal{H}_C(*)} \varphi'[\psi]_{\rho'} \Leftrightarrow \varphi'[\chi]_{\rho'}$. By propositional calculus, the following rule is admissible in $\mathcal{H}_C(*)$:

$$\frac{\psi \Leftrightarrow \chi}{\neg\psi \Leftrightarrow \neg\chi}$$

So, $\vdash_{\mathcal{H}_C(*)} \neg(\varphi'[\psi]_{\rho'}) \Leftrightarrow \neg(\varphi'[\chi]_{\rho'})$. By definition, $\varphi[\psi]_{\rho} = \neg(\varphi'[\psi]_{\rho'})$ and $\varphi[\chi]_{\rho} = \neg(\varphi'[\chi]_{\rho'})$.

induction step: $\varphi = \varphi' \Rightarrow \varphi''$. Analogous to the previous case, by relying on the two following rules, that are admissible in $\mathcal{H}_C(*)$ directly from the presence of axioms and modus ponens of propositional calculus:

$$\frac{\psi \Leftrightarrow \chi}{(\psi \Rightarrow \varphi) \Leftrightarrow (\chi \Rightarrow \varphi)} \quad \frac{\psi \Leftrightarrow \chi}{(\varphi \Rightarrow \psi) \Leftrightarrow (\varphi \Rightarrow \chi)}$$

induction step: $\varphi = \varphi' * \varphi''$. Again, if $\rho = \epsilon$, then the result follows as in the base case. Otherwise, either $\rho = 1\rho'$ or $\rho = 2\rho''$, for some positions ρ' and ρ'' of φ' and φ'' , respectively. Below, we consider the case where $\rho = 1\rho'$. The other case is analogous, by considering φ'' instead of φ' , and relying on the commutativity of $*$ (axiom $(\text{COM})^*$). By induction hypothesis, $\vdash_{\mathcal{H}_C(*)} \varphi'[\psi]_{\rho'} \Leftrightarrow \varphi'[\chi]_{\rho'}$. Directly from the presence of the rule $(*)$ in $\mathcal{H}_C(*)$, the following rule is admissible:

$$\frac{\psi \Leftrightarrow \chi}{\psi * \varphi \Leftrightarrow \chi * \varphi}$$

Therefore, $\vdash_{\mathcal{H}_C(*)} (\varphi'[\psi]_{\rho'}) * \varphi'' \Leftrightarrow (\varphi'[\chi]_{\rho'}) * \varphi''$. By definition, $\varphi[\psi]_{\rho} = (\varphi'[\psi]_{\rho'}) * \varphi''$ and $\varphi[\chi]_{\rho} = (\varphi'[\chi]_{\rho'}) * \varphi''$, concluding the proof. \square

Proof of Theorem 6.16. The soundness of $\mathcal{H}_C(*)$ has been established in Lemma 6.11. As far as the completeness proof is concerned, we need to show that for every formula φ in $\text{SL}(*, \text{x} \hookrightarrow -)$, there is a Boolean combination of core formulae ψ such that $\vdash_{\mathcal{H}_C(*)} \varphi \Leftrightarrow \psi$. This is enough to conclude the proof. Indeed, when φ is valid for $\text{SL}(*, \text{x} \hookrightarrow -)$, by soundness of $\mathcal{H}_C(*)$, we obtain that ψ is valid too. Since ψ is a Boolean combination of core formulae and $\mathcal{H}_C(*)$ includes \mathcal{H}_C , by Theorem 6.9 we derive $\vdash_{\mathcal{H}_C(*)} \psi$. Together with $\vdash_{\mathcal{H}_C(*)} \varphi \Leftrightarrow \psi$ and by propositional reasoning, this implies $\vdash_{\mathcal{H}_C(*)} \varphi$.

To show that every formula φ has a provably equivalent Boolean combination of core formulae, we heavily rely on Corollary 6.15. The proof is by simple structural induction on the number of occurrences of the separating conjunction in φ that are not involved in the definition of some core formula of the form $\text{size} \geq \beta$.

base case: φ without occurrences of $*$ (excluding those in $\text{size} \geq \beta$ formulae). In this case, the formula φ is a Boolean combination of core formulae plus the atomic formula emp (recall, that \top is the core formula $\text{size} \geq 0$). By propositional reasoning $\vdash_{\mathcal{H}_C(*)} \text{emp} \Leftrightarrow \neg\neg\text{emp}$, where $\neg\neg\text{emp}$ is syntactically equivalent to $\text{size} \geq 1$, by definition of $\text{size} \geq 1$. Therefore, by relying on (S_*) , all subformulae emp can be replaced with $\text{size} \geq 1$, leading to a Boolean combination of core formulae ψ such that $\vdash_{\mathcal{H}_C(*)} \varphi \Leftrightarrow \psi$.

induction step: φ with $n \geq 1$ occurrences of $*$ (excluding $\text{size} \geq \beta$ formulae). Below, we let X be the set of variables appearing in φ . Let $\varphi_1 * \varphi_2$ be a subformula of φ , say at position ρ , such that φ_1 in $\text{Bool}(\text{Core}(X_1, \alpha_1))$ and φ_2 in $\text{Bool}(\text{Core}(X_2, \alpha_2))$. Let $\alpha = \max(\alpha_1, \alpha_2)$. As X_1 and X_2 are subsets of X , by definition, φ_1 and φ_2 belong to $\text{Bool}(\text{Core}(X, \alpha))$. By Lemma 6.10 and since $\mathcal{H}_C(*)$ includes \mathcal{H}_C , there are two formulae of

Axioms and rules from $\mathcal{H}_C(*)$ (Figure 6.6).

Axioms of the separating implication:

$$\begin{aligned} (\textcolor{red}{\rightarrow\!\!\!}\infty^*) \quad & (\mathbf{size} = 1 \wedge \bigwedge_{x \in X} \neg x \hookrightarrow _) \dashv\ast \top \quad \blacktriangleleft [X \subseteq_{\text{fin}} \text{VAR}] \\ (\textcolor{red}{\rightarrow\!\!\!}\infty^*) \quad & \neg x \hookrightarrow _ \Rightarrow ((x \hookrightarrow y \wedge \mathbf{size} = 1) \dashv\ast \top) \\ (\textcolor{red}{\rightarrow\!\!\!}\infty_{\text{ALLOC}}^*) \quad & \neg x \hookrightarrow _ \Rightarrow ((x \hookrightarrow _ \wedge \mathbf{size} = 1 \wedge \bigwedge_{y \in X} \neg x \hookrightarrow y) \dashv\ast \top) \quad \blacktriangleleft [X \subseteq_{\text{fin}} \text{VAR}] \end{aligned}$$

Rules of inference for the separating implication:

$$\begin{array}{c} (\textcolor{red}{\rightarrow\!\!\!}\infty_1) \quad \frac{\varphi \Rightarrow (\psi \dashv\ast \chi)}{\varphi \ast \psi \Rightarrow \chi} \qquad (\textcolor{red}{\rightarrow\!\!\!}\infty_2) \quad \frac{\varphi \ast \psi \Rightarrow \chi}{\varphi \Rightarrow (\psi \dashv\ast \chi)} \end{array}$$

Figure 6.8: The Hilbert-style proof system $\mathcal{H}_C(*, \dashv\ast)$ (again).

the form $\varphi_1^1 \vee \dots \vee \varphi_1^{n_1}$ and $\varphi_2^1 \vee \dots \vee \varphi_2^{n_2}$ such that $\vdash_{\mathcal{H}_C(*)} \varphi_i \Leftrightarrow \varphi_i^1 \vee \dots \vee \varphi_i^{n_i}$ for $i \in \{1, 2\}$ and moreover, all the φ_i^j 's are core types in $\text{CoreTypes}(X, \max(\text{card}(X), \alpha))$. By (S_*) ,

$$\vdash_{\mathcal{H}_C(*)} \varphi_1 \ast \varphi_2 \Leftrightarrow (\varphi_1^1 \vee \dots \vee \varphi_1^{n_1}) \ast (\varphi_2^1 \vee \dots \vee \varphi_2^{n_2}).$$

Again by propositional reasoning, together with the axiom (I_3^*) for distributivity and the theorem $(\varphi \ast \chi) \vee (\psi \ast \chi) \Rightarrow (\varphi \vee \psi) \ast \chi$ derived in page 296, we have

$$\vdash_{\mathcal{H}_C(*)} \varphi_1 \ast \varphi_2 \Leftrightarrow \bigvee_{j_1 \in [1, n_1], j_2 \in [1, n_2]} \varphi_1^{j_1} \ast \varphi_2^{j_2}.$$

For all $j_1 \in [1, n_1]$ and $j_2 \in [1, n_2]$, by Corollary 6.15, we derive that there is a formula ψ^{j_1, j_2} in $\text{Conj}(\text{Core}(X, 2 \max(\text{card}(X), \alpha)))$ such that $\vdash_{\mathcal{H}_C(*)} \varphi_1^{j_1} \ast \varphi_2^{j_2} \Leftrightarrow \psi^{j_1, j_2}$. By propositional reasoning, we conclude that $\vdash_{\mathcal{H}_C(*)} \varphi_1 \ast \varphi_2 \Leftrightarrow \bigvee_{j_1 \in [1, n_1], j_2 \in [1, n_2]} \psi^{j_1, j_2}$. Consequently (thanks to the rule (S_*)), we obtain

$$\vdash_{\mathcal{H}_C(*)} \varphi \Leftrightarrow \varphi [\bigvee_{j_1 \in [1, n_1], j_2 \in [1, n_2]} \psi^{j_1, j_2}]_\rho.$$

The right-hand side formula of the double implication above has $n - 1$ occurrences of the separating conjunction that are not involved in the definition of some core formula of the form $\mathbf{size} \geq \beta$. The induction hypothesis applies, allowing us to derive that there is a Boolean combination of core formulae ψ such that $\vdash_{\mathcal{H}_C(*)} \varphi [\bigvee_{j_1 \in [1, n_1], j_2 \in [1, n_2]} \psi^{j_1, j_2}]_\rho \Leftrightarrow \psi$, which leads to $\vdash_{\mathcal{H}_C(*)} \varphi \Leftrightarrow \psi$, by propositional reasoning. \square

6.6 SYNTACTICAL ELIMINATION OF THE SEPARATING IMPLICATION

In order to obtain the final proof system $\mathcal{H}_C(*, \dashv\ast)$, we add the axioms and rules from Figure 6.8 to the proof system $\mathcal{H}_C(*)$. These new axioms and rules are dedicated to the separating implication. The axioms $(\textcolor{red}{\rightarrow\!\!\!}\infty^*)$ – $(\textcolor{red}{\rightarrow\!\!\!}\infty_{\text{ALLOC}}^*)$ involve the subtraction $\dashv\ast$, and express that it is always possible to extend a given heap with an extra memory cell, and that the address and the content of this cell can be fixed arbitrarily (provided it is not already allocated). The adjunction rules $(\textcolor{red}{\rightarrow\!\!\!}\infty_2)$ and $(\textcolor{red}{\rightarrow\!\!\!}\infty_1)$ are from BBI (Section 2.3.3). One can observe that, in $\mathcal{H}_C(*, \dashv\ast)$, the axioms (I_3^*) , (I_4^*) and (I_5^*) of $\mathcal{H}_C(*)$ are derivable.

Lemma 6.17. The axioms (I_3^*) , (I_4^*) and (I_5^*) are derivable in $\mathcal{H}_C(*, \neg)$.

Proof of (I_3^) .*

1	$(\varphi * \chi) \Rightarrow (\varphi * \chi) \vee (\psi * \chi)$	PC
2	$(\psi * \chi) \Rightarrow (\varphi * \chi) \vee (\psi * \chi)$	PC
3	$\varphi \Rightarrow (\chi \neg (\varphi * \chi) \vee (\psi * \chi))$	$(\neg \star_2)$, 1
4	$\psi \Rightarrow (\chi \neg (\varphi * \chi) \vee (\psi * \chi))$	$(\neg \star_2)$, 2
5	$\varphi \vee \psi \Rightarrow (\chi \neg (\varphi * \chi) \vee (\psi * \chi))$	PC, 3, 4
6	$(\varphi \vee \psi) * \chi \Rightarrow (\varphi * \chi) \vee (\psi * \chi)$	$(\neg \star_1)$, 5 \square

Proof of (I_4^) .* The axiom (I_4^*) is provable by $(\neg \star_2)$. Indeed, proving $(\perp * \varphi) \Rightarrow \perp$ reduces to proving $\perp \Rightarrow (\varphi \neg \perp)$. The latter is a tautology by propositional reasoning. \square

Proof of (I_5^) .*

1	$\perp * \top \Rightarrow \perp$	(I_4^*)
2	$(x \hookrightarrow x \neg \perp) \Rightarrow (x \hookrightarrow x \neg \perp)$	PC
3	$(x \hookrightarrow x \neg \perp) * x \hookrightarrow x \Rightarrow \perp$	$(\neg \star_1)$, 2
4	$x \hookrightarrow x * (x \hookrightarrow x \neg \perp) \Rightarrow (x \hookrightarrow x \neg \perp) * x \hookrightarrow x$	$(\text{COM})^*$
5	$x \hookrightarrow x * (x \hookrightarrow x \neg \perp) \Rightarrow \perp$	$(\Rightarrow \text{TR})$, 4, 3
6	$(x \hookrightarrow x * (x \hookrightarrow x \neg \perp)) * \top \Rightarrow \perp * \top$	(\star) , 5
7	$((x \hookrightarrow x \neg \perp) * \top) * (x \hookrightarrow x) \Rightarrow (x \hookrightarrow x * (x \hookrightarrow x \neg \perp)) * \top$	$(\text{COM})^*, (\text{ASSOC})^*$
8	$((x \hookrightarrow x \neg \perp) * \top) * (x \hookrightarrow x) \Rightarrow \perp$	$(\Rightarrow \text{TR})$, 7, 6, 1
9	$(x \hookrightarrow x \neg \perp) * \top \Rightarrow (x \hookrightarrow x \neg \perp)$	$(\neg \star_2)$, 8
10	$x \hookrightarrow _ * \top \Rightarrow x \hookrightarrow _$	Def. $x \hookrightarrow _$, 9 \square

Lemma 6.17 allows us to remove the axioms (I_3^*) , (I_4^*) and (I_5^*) from the $\mathcal{H}_C(*, \neg)$, obtaining the proof system as defined at the beginning of the chapter (Figure 6.2). Fundamentally, $\mathcal{H}_C(*, \neg)$ enjoys the \neg -simulation property, as formalised in the following lemma. Actually, we state the property with the help of the subtraction \neg , as we find the related statements and developments more intuitive. Recall that $\varphi \neg \psi \stackrel{\text{def}}{=} \neg(\varphi \neg \neg \psi)$.

Lemma 6.18 (\neg -simulation). Let $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq \text{card}(X)$. Let φ and ψ in $\text{CoreTypes}(X, \alpha)$. There is a conjunction $\chi \in \text{Conj}(\text{Core}(X, \alpha))$ such that $\vdash_{\mathcal{H}_C(*, \neg)} (\varphi \neg \psi) \Leftrightarrow \chi$.

In the proof of Lemma 6.18, the formula χ is explicitly constructed from φ and ψ , following a pattern analogous to the construction of $\langle *\rangle(\dots)$ in Figure 6.7. The derivation of the equivalence $(\varphi \neg \psi) \Leftrightarrow \chi$ is shown as follows. First, the formulae $\chi * \varphi \Rightarrow \psi$ and $\neg \chi * \varphi \Rightarrow \neg \psi$ are shown valid (by using semantical means). As $\mathcal{H}_C(*)$ is complete for $\text{SL}(*, x \hookrightarrow _)$, it is a subsystem of $\mathcal{H}_C(*, \neg)$, and the formulae φ , ψ and χ are Boolean combinations of core formulae, we get $\vdash_{\mathcal{H}_C(*, \neg)} \chi * \varphi \Rightarrow \psi$ and $\vdash_{\mathcal{H}_C(*, \neg)} \neg \chi * \varphi \Rightarrow \neg \psi$. The latter theorem leads to $\vdash_{\mathcal{H}_C(*, \neg)} (\varphi \neg \psi) \Rightarrow \chi$ by using the definition of \neg and the rule $(\neg \star_2)$. In order to show that $\vdash_{\mathcal{H}_C(*, \neg)} \chi \Rightarrow (\varphi \neg \psi)$ holds, we take advantage of the admissibility of the theorem $(I_{6.19.9}^*)$ (see Lemma 6.19) for

which an instance is $(\varphi \multimap \top) \wedge (\varphi \multimap \psi) \Rightarrow (\varphi \multimap (\top \wedge \psi))$. From $\vdash_{\mathcal{H}_C(*, \multimap)} \chi * \varphi \Rightarrow \psi$ and by (\multimap_2) we have $\vdash_{\mathcal{H}_C(*, \multimap)} \chi \Rightarrow (\varphi \multimap \psi)$. Therefore, the main technical development lies in the proof of $\vdash_{\mathcal{H}_C(*, \multimap)} \chi \Rightarrow (\varphi \multimap \top)$, which allows us to take advantage of $(I_{6.19.9}^*)$, and leads to $\vdash_{\mathcal{H}_C(*, \multimap)} \chi \Rightarrow (\varphi \multimap \psi)$ by propositional reasoning.

In order to formalise the proof of Lemma 6.18 sketched above, we start by establishing several admissible axioms and rules (Lemma 6.19). Afterwards, we define the formula χ and show the validity of $\chi * \varphi \Rightarrow \psi$ and $\neg \chi * \varphi \Rightarrow \neg \psi$ (Lemma 6.20). Then, come the final bits of the proof of Lemma 6.18.

Lemma 6.19. The following axioms and rules are admissible in $\mathcal{H}_C(*, \multimap)$:

$$\begin{array}{ll}
 (I_{6.19.1}^*) \quad (\perp \multimap \varphi) \Rightarrow \perp & (I_{6.19.6}^*) \quad (\varphi \vee \psi) \multimap \chi \Leftrightarrow (\varphi \multimap \chi) \vee (\psi \multimap \chi) \\
 (I_{6.19.2}^*) \quad (\varphi \multimap \perp) \Rightarrow \perp & (I_{6.19.7}^*) \quad \chi \multimap (\varphi \vee \psi) \Leftrightarrow (\chi \multimap \varphi) \vee (\chi \multimap \psi) \\
 (I_{6.19.3}^*) \quad \varphi * (\varphi \multimap \psi) \Rightarrow \psi & (I_{6.19.8}^*) \quad \varphi \multimap (\psi \multimap \chi) \Leftrightarrow (\varphi * \psi) \multimap \chi \\
 (I_{6.19.4}^*) \quad \frac{\varphi \Rightarrow \psi}{(\varphi \multimap \chi) \Rightarrow (\psi \multimap \chi)} & (I_{6.19.9}^*) \quad (\varphi \multimap \psi) \wedge (\varphi \multimap \chi) \Rightarrow (\varphi \multimap \psi \wedge \chi) \\
 (I_{6.19.5}^*) \quad \frac{\varphi \Rightarrow \psi}{(\chi \multimap \varphi) \Rightarrow (\chi \multimap \psi)} & (I_{6.19.10}^*) \quad \mathbf{x} \sim \mathbf{y} \wedge (\varphi \multimap \psi) \Rightarrow \\
 & \quad (\varphi \wedge \mathbf{x} \sim \mathbf{y} \multimap \psi) \quad \blacktriangleleft \sim \in \{=, \neq\}.
 \end{array}$$

With the exception of the theorem $(I_{6.19.10}^*)$, all the theorems in Lemma 6.19 are from BBI. Below, we show the proofs of $(I_{6.19.3}^*)$, $(I_{6.19.4}^*)$, $(I_{6.19.6}^*)$ and $(I_{6.19.10}^*)$. The proof of the other theorems is left in Appendix D.

Proof of $(I_{6.19.3}^)$.*

1	$(\varphi \multimap \psi) \Rightarrow (\varphi \multimap \psi)$	PC
2	$(\varphi \multimap \psi) * \varphi \Rightarrow \psi$	(\multimap_1) , 1
3	$\varphi * (\varphi \multimap \psi) \Rightarrow (\varphi \multimap \psi) * \varphi$	$({}^*_{\text{COM}})$
4	$\varphi * (\varphi \multimap \psi) \Rightarrow \psi$	$(\Rightarrow \text{TR})$, 3, 2 \square

Proof of $(I_{6.19.4}^)$.*

1	$\varphi \Rightarrow \psi$	Hypothesis
2	$\psi * (\psi \multimap \neg \chi) \Rightarrow \neg \chi$	$(I_{6.19.3}^*)$
3	$(\psi \multimap \neg \chi) * \varphi \Rightarrow \varphi * (\psi \multimap \neg \chi)$	$({}^*_{\text{COM}})$
4	$\varphi * (\psi \multimap \neg \chi) \Rightarrow \psi * (\psi \multimap \neg \chi)$	(\multimap_1) , 1
5	$\varphi * (\psi \multimap \neg \chi) \Rightarrow \neg \chi$	$(\Rightarrow \text{TR})$, 2, 4
6	$(\psi \multimap \neg \chi) * \varphi \Rightarrow \neg \chi$	$(\Rightarrow \text{TR})$, 3, 5
7	$(\psi \multimap \neg \chi) \Rightarrow (\varphi \multimap \neg \chi)$	(\multimap_2) , 6
8	$\neg(\varphi \multimap \neg \chi) \Rightarrow \neg(\psi \multimap \neg \chi)$	PC, 7
9	$(\varphi \multimap \chi) \Rightarrow (\psi \multimap \chi)$	Def. \multimap , 8 \square

Proof of $(I_{6.19.6}^)$.* We derive each implication separately.

1	$(\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi) \Rightarrow (\psi \multimap \neg\chi)$	PC
2	$\psi * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) \Rightarrow \psi * (\psi \multimap \neg\chi)$	$(*\text{I}_{LR})$, 1
3	$(\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi) \Rightarrow (\varphi \multimap \neg\chi)$	PC
4	$\varphi * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) \Rightarrow \varphi * (\varphi \multimap \neg\chi)$	$(*\text{I}_{LR})$, 3
5	$\varphi * (\varphi \multimap \neg\chi) \Rightarrow \neg\chi$	$(I_{6.19.3}^*)$
6	$\psi * (\psi \multimap \neg\chi) \Rightarrow \neg\chi$	$(I_{6.19.3}^*)$
7	$\psi * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) \Rightarrow \neg\chi$	$(\Rightarrow \text{TR})$, 2, 6
8	$\varphi * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) \Rightarrow \neg\chi$	$(\Rightarrow \text{TR})$, 4, 5
9	$(\varphi \vee \psi) * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) \Rightarrow$ $\varphi * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) \vee \psi * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi))$	(I_3^*)
10	$(\varphi \vee \psi) * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) \Rightarrow \neg\chi$	PC, 7, 8, 9
11	$((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) * (\varphi \vee \psi) \Rightarrow (\varphi \vee \psi) * ((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi))$	(COM^*)
12	$((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) * (\varphi \vee \psi) \Rightarrow \neg\chi$	$(\Rightarrow \text{TR})$, 12, 10
13	$((\varphi \multimap \neg\chi) \wedge (\psi \multimap \neg\chi)) \Rightarrow ((\varphi \vee \psi) \multimap \neg\chi)$	(\multimap_2) , 12
14	$\neg(\varphi \vee \psi \multimap \neg\chi) \Rightarrow \neg(\varphi \multimap \neg\chi) \vee \neg(\psi \multimap \neg\chi)$	PC, 13
15	$((\varphi \vee \psi) \multimap \chi) \Rightarrow (\varphi \multimap \chi) \vee (\psi \multimap \chi)$	Def. \multimap , 14

The derivation of the other implication can be found below.

1	$\varphi \Rightarrow \varphi \vee \psi$	PC
2	$\psi \Rightarrow \varphi \vee \psi$	PC
3	$(\varphi \multimap \chi) \Rightarrow (\varphi \vee \psi \multimap \chi)$	$(I_{6.19.4}^*)$, 1
4	$(\psi \multimap \chi) \Rightarrow (\varphi \vee \psi \multimap \chi)$	$(I_{6.19.4}^*)$, 2
5	$((\psi \multimap \chi) \vee (\varphi \multimap \chi)) \Rightarrow (\varphi \vee \psi \multimap \chi)$	PC, 3, 4 \square

Proof of $(I_{6.19.10}^)$.* Let $\sim \in \{=, \neq\}$.

1	$\varphi \Rightarrow (\varphi \wedge x \sim y) \vee (\varphi \wedge \neg x \sim y)$	PC
2	$(\varphi \multimap \top) \Rightarrow ((\varphi \wedge x \sim y) \vee (\varphi \wedge \neg x \sim y)) \multimap \top$	$(I_{6.19.4}^*)$, 1
3	$(\varphi \multimap \top) \Rightarrow ((\varphi \wedge x \sim y) \multimap \top) \vee ((\varphi \wedge \neg x \sim y) \multimap \top)$	$(I_{6.19.6}^*)$, $(\Rightarrow \text{TR})$, 2
4	$x \sim y * \neg x \sim y \Rightarrow x \sim y$	(MONO^*) , $(*\text{I}_{LR})$
5	$\neg x \sim y * x \sim y \Rightarrow \neg x \sim y$	(MONO^*) , $(*\text{I}_{LR})$
6	$x \sim y * \neg x \sim y \Rightarrow x \sim y \wedge \neg x \sim y$	(COM^*) , $(\Rightarrow \text{TR})$, PC, 4, 5

$$\begin{aligned}
& \wedge \{x \sim y \subseteq_{\text{LIT}} \{\varphi \mid \psi\} \mid \sim \in \{=, \neq\}\} \quad \wedge \wedge \left\{ x \hookrightarrow - \mid \begin{array}{l} \neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi \\ x \hookrightarrow - \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
& \wedge \wedge \{\neg x \hookrightarrow - \subseteq_{\text{LIT}} \psi\} \quad \wedge \wedge \{\neg x \hookrightarrow - \mid x \hookrightarrow - \subseteq_{\text{LIT}} \varphi\} \\
& \wedge \wedge \{\neg x \hookrightarrow y \subseteq_{\text{LIT}} \psi\} \quad \wedge \wedge \left\{ x \hookrightarrow y \mid \begin{array}{l} \neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi \\ x \hookrightarrow y \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
& \wedge \wedge \left\{ x \neq x \mid \begin{array}{l} x \hookrightarrow - \wedge \neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi \\ x \hookrightarrow y \subseteq_{\text{LIT}} \psi \end{array} \right\} \quad \wedge \wedge \left\{ \text{size} \geq \beta_2 + 1 - \beta_1 \mid \begin{array}{l} \neg \text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi \\ \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
& \wedge \wedge \left\{ x \neq x \mid \begin{array}{l} x \hookrightarrow y \subseteq_{\text{LIT}} \varphi \\ \neg x \hookrightarrow y \subseteq_{\text{LIT}} \psi \end{array} \right\} \quad \wedge \wedge \left\{ \neg \text{size} \geq \beta_2 - \beta_1 \mid \begin{array}{l} \text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi \\ \neg \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
& \wedge \wedge \left\{ x \neq x \mid \begin{array}{l} x \hookrightarrow - \subseteq_{\text{LIT}} \varphi \\ \neg x \hookrightarrow - \subseteq_{\text{LIT}} \psi \end{array} \right\}
\end{aligned}$$

Figure 6.9: The formula $\langle \circledast \rangle(\varphi, \psi)$.

7	$x \sim y * \neg x \sim y \Rightarrow \neg \top$	PC, 6
8	$x \sim y \Rightarrow (\neg x \sim y * \neg \top)$	($\neg \circledast_2$), 7
9	$\neg(\neg x \sim y * \neg \top) \Rightarrow \neg x \sim y$	PC, 8
10	$(\neg x \sim y \circledast \top) \Rightarrow \neg x \sim y$	Def. \circledast , 9
11	$\varphi \wedge \neg x \sim y \Rightarrow \neg x \sim y$	PC
12	$((\varphi \wedge \neg x \sim y) \circledast \top) \Rightarrow (\neg x \sim y \circledast \top)$	($I_{6.19.6}^*$), 11
13	$((\varphi \wedge \neg x \sim y) \circledast \top) \Rightarrow \neg x \sim y$	($\Rightarrow \text{TR}$), 10, 12
14	$(\varphi \circledast \top) \Rightarrow ((\varphi \wedge x \sim y) \circledast \top) \vee \neg x \sim y$	PC, 3, 13
15	$x \sim y \wedge (\varphi \circledast \top) \Rightarrow (\varphi \wedge x \sim y) \circledast \top$	PC, 14 \square

Let φ and ψ be two satisfiable core types in $\text{Conj}(\text{Core}(X, \alpha))$. Following the developments of Section 6.5, we define a formula $\langle \circledast \rangle(\varphi, \psi)$ in $\text{Conj}(\text{Core}(X, \alpha))$, for which we show that $(\varphi \circledast \psi) \Leftrightarrow \langle \circledast \rangle(\varphi, \psi)$ is provable in $\mathcal{H}_C(*, \neg*)$. The formula $\langle \circledast \rangle(\varphi, \psi)$ is defined in Figure 6.9.

Lemma 6.20. Let $X \subseteq_{\text{fin}} \text{VAR}$, $\alpha \geq \text{card}(X)$ and φ, ψ be satisfiable core types in $\text{CoreTypes}(X, \alpha)$. The formulae $\langle \circledast \rangle(\varphi, \psi) * \varphi \Rightarrow \psi$ and $(\neg \langle \circledast \rangle(\varphi, \psi)) * \varphi \Rightarrow \neg \psi$ are valid.

Since we aim at proving the derivability of $(\varphi \circledast \psi) \Leftrightarrow \langle \circledast \rangle(\varphi, \psi)$ in $\mathcal{H}_C(*, \neg*)$, the validity of the formula $(\neg \langle \circledast \rangle(\varphi, \psi)) * \varphi \Rightarrow \neg \psi$ should not surprise the reader. Indeed, by replacing $\langle \circledast \rangle(\varphi, \psi)$ with $\varphi \circledast \psi$ we obtain $(\neg(\varphi \circledast \psi)) * \varphi \Rightarrow \neg \psi$ which, unfolding the definition of \circledast , is equivalent to the valid formula $(\varphi * \neg \psi) * \varphi \Rightarrow \neg \psi$ (see ($I_{6.19.3}^*$)). On the other hand, the fact that $\langle \circledast \rangle(\varphi, \psi) * \varphi \Rightarrow \psi$ is valid can be puzzling at first, as the formula $(\varphi \circledast \psi) * \varphi \Rightarrow \psi$ is not valid (in general). In its essence, Lemma 6.20 shows that $(\varphi \circledast \psi) * \varphi \Rightarrow \psi$ is valid whenever φ and ψ are restricted to core types.

Proof. Notice that the proof of lemma only requires semantical arguments. Since φ , ψ and $\langle \circledast \rangle(\varphi, \psi)$ are conjunctions of literals built from core formulae, derivability of these two tautologies in $\mathcal{H}_C(*, \neg*)$ follows from the completeness of $\mathcal{H}_C(*)$ (Theorem 6.16).

Validity of $\langle \circledast \rangle(\varphi, \psi) * \varphi \Rightarrow \psi$. If $\langle \circledast \rangle(\varphi, \psi) * \varphi$ is inconsistent, then $\langle \circledast \rangle(\varphi, \psi) * \varphi \Rightarrow \psi$ is straightforwardly valid. Below, we assume that $\langle \circledast \rangle(\varphi, \psi) * \varphi$ is satisfiable. In particular, none of the conditions depicted in Figure 6.9 that result in $\langle \circledast \rangle(\varphi, \psi)$ having a literal $x \neq x$ applies. Let $(s, h) \models \langle \circledast \rangle(\varphi, \psi) * \varphi$. Therefore, there are two disjoint heaps h_1 and h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \langle \circledast \rangle(\varphi, \psi)$ and $(s, h_2) \models \varphi$. We show that (s, h) satisfies each literal L in ψ . We perform a simple case analysis on the shape of L . Notice that, below, we have $x, y \in X$ and $\beta_2 \in [0, \alpha]$, as ψ is a core type in $\text{CoreTypes}(X, \alpha)$.

case: $L = x \sim y$, where $\sim \in \{=, \neq\}$. By definition of $\langle \circledast \rangle(\varphi, \psi)$, $x \sim y \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$ and so $(s, h_1) \models x \sim y$. We conclude that $s(x) \sim s(y)$, and thus $(s, h) \models x \sim y$.

case: $L = x \leftrightarrow __$. If $x \leftrightarrow __ \subseteq_{\text{LIT}} \varphi$, then $(s, h_2) \models x \leftrightarrow __$, which implies $s(x) \in \text{dom}(h)$ directly from $h_2 \subseteq h$. Thus, $(s, h) \models x \leftrightarrow __$. Otherwise, if $x \leftrightarrow __ \not\subseteq_{\text{LIT}} \varphi$ then, since φ is a core type in $\text{CoreTypes}(X, \alpha)$, we have $\neg x \leftrightarrow __ \subseteq_{\text{LIT}} \varphi$. By definition of $\langle \circledast \rangle(\varphi, \psi)$, we derive that $x \leftrightarrow __ \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. So, $(s, h_2) \models x \leftrightarrow __$ and thus, by $h_2 \subseteq h$, $s(x) \in \text{dom}(h)$. We conclude that $(s, h) \models x \leftrightarrow __$.

case: $L = \neg x \leftrightarrow __$. In this case, by definition of $\langle \circledast \rangle(\varphi, \psi)$, we have $\neg x \leftrightarrow __ \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, which implies $(s, h_1) \models \neg x \leftrightarrow __$. *Ad absurdum*, suppose $(s, h_2) \models x \leftrightarrow __$. Since φ is a core type in $\text{CoreTypes}(X, \alpha)$, we conclude that $x \leftrightarrow __ \subseteq_{\text{LIT}} \varphi$. However, by definition of $\langle \circledast \rangle(\varphi, \psi)$, this implies $x \neq x \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, which contradicts the fact that $\langle \circledast \rangle(\varphi, \psi)$ is satisfiable. Thus, $(s, h_2) \models \neg x \leftrightarrow __$, which implies $s(x) \notin \text{dom}(h_2)$. From $h = h_1 + h_2$ and $s(x) \notin \text{dom}(h_1)$ we conclude that $s(x) \notin \text{dom}(h)$. So, $(s, h) \models \neg x \leftrightarrow __$.

case: $L = x \leftrightarrow y$. If $\neg x \leftrightarrow __ \subseteq_{\text{LIT}} \varphi$, then $x \leftrightarrow y \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, by definition of $\langle \circledast \rangle(\varphi, \psi)$. So, $h_1(s(x)) = s(y)$ and, from $h_1 \subseteq h$ we conclude that $(s, h) \models x \leftrightarrow y$. Otherwise, let us assume that $x \leftrightarrow __ \subseteq_{\text{LIT}} \varphi$. *Ad absurdum*, suppose $\neg x \leftrightarrow y \subseteq_{\text{LIT}} \varphi$. Then, by definition of $\langle \circledast \rangle(\varphi, \psi)$, we derive $x \neq x \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. However, this contradicts the satisfiability of $\langle \circledast \rangle(\varphi, \psi)$. Therefore, $\neg x \leftrightarrow y \not\subseteq_{\text{LIT}} \varphi$. Since φ is a core type, this implies $x \leftrightarrow y \subseteq_{\text{LIT}} \varphi$, and therefore $h_2(s(x)) = s(y)$. From $h_2 \subseteq h$ we conclude that $(s, h) \models x \leftrightarrow y$.

case: $L = \neg x \leftrightarrow y$. By definition of $\langle \circledast \rangle(x, y)$, we have $\neg x \leftrightarrow y \subseteq_{\text{LIT}} \langle \circledast \rangle(x, y)$, which implies that if $s(x) \in \text{dom}(h_1)$ then $h_1(s(x)) \neq s(y)$. *Ad absurdum*, suppose $x \leftrightarrow y \subseteq_{\text{LIT}} \varphi$. Then, by definition of $\langle \circledast \rangle(\varphi, \psi)$, we derive $x \neq x \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. However, this contradicts the satisfiability of $\langle \circledast \rangle(\varphi, \psi)$. Therefore $x \leftrightarrow y \not\subseteq_{\text{LIT}} \varphi$ and, since φ is a core type, $\neg x \leftrightarrow y \subseteq_{\text{LIT}} \varphi$. So, if $s(x) \in \text{dom}(h_2)$ then $h_2(s(x)) \neq s(y)$. By $h = h_1 + h_2$ and the fact that $h_1(s(x)) \neq s(y)$, we conclude that $(s, h) \models x \leftrightarrow y$.

case: $L = \text{size} \geq \beta_2$. If $\text{size} \geq \alpha \subseteq_{\text{LIT}} \varphi$, then $\text{card}(h) \geq \text{card}(h_2) \geq \alpha$, by $h_2 \subseteq h$. As $\beta_2 \in [0, \alpha]$, this implies $(s, h) \models \text{size} \geq \beta_2$. Else, assume $\text{size} \geq \alpha \not\subseteq_{\text{LIT}} \varphi$. In particular, since φ is in $\text{CoreTypes}(X, \alpha)$, this implies that $\max_{\text{size}}(\varphi) < \alpha$ and

$$\text{size} \geq \max_{\text{size}}(\varphi) \wedge \neg \text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi.$$

We have $\text{card}(h_2) = \max_{\text{size}}(\varphi)$. If $\max_{\text{size}}(\varphi) \geq \beta_2$, then from $h_2 \subseteq h$ we conclude that $(s, h) \models \text{size} \geq \beta_2$. Otherwise, let us assume $\beta_2 > \max_{\text{size}}(\varphi)$. By definition of $\langle \circledast \rangle(\varphi, \psi)$, we conclude that $\text{size} \geq \beta_2 + 1 \doteq (\max_{\text{size}}(\varphi) + 1) \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. Together with $\beta_2 > \max_{\text{size}}(\varphi)$, this implies $\text{card}(h_1) \geq \beta_2 - \max_{\text{size}}(\varphi)$. With $\text{card}(h_2) = \max_{\text{size}}(\varphi)$ and $h = h_1 + h_2$, this implies $(s, h) \models \text{size} \geq \beta_2$.

case: $L = \neg \text{size} \geq \beta_2$. *Ad absurdum*, suppose that $\text{size} \geq \alpha \subseteq_{\text{LIT}} \varphi$. Then, by definition of $\langle \circledast \rangle(\varphi, \psi)$ we have $\neg \text{size} \geq \beta_2 \doteq \alpha \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. However, since $\beta_2 \in [0, \alpha]$, this

means that $\neg\text{size} \geq 0 \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, which contradicts the satisfiability of $\langle \circledast \rangle(\varphi, \psi)$. Therefore, $\text{size} \geq \alpha \not\subseteq_{\text{LIT}} \varphi$. As φ is in $\text{CoreTypes}(X, \alpha)$, we derive $\max_{\text{size}}(\varphi) < \alpha$ and

$$\text{size} \geq \max_{\text{size}}(\varphi) \wedge \neg\text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi.$$

We conclude that $\text{card}(h_2) \leq \max_{\text{size}}(\varphi)$. From $\text{size} \geq \max_{\text{size}}(\varphi) \subseteq_{\text{LIT}} \varphi$ and by definition of $\langle \circledast \rangle(\varphi, \psi)$, we conclude that

$$\neg\text{size} \geq \beta_2 \doteq \max_{\text{size}}(\varphi) \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi).$$

If $\beta_2 \leq \max_{\text{size}}(\varphi)$, then $\neg\text{size} \geq 0 \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, which contradicts the satisfiability of $\langle \circledast \rangle(\varphi, \psi)$. Therefore, $\beta_2 > \max_{\text{size}}(\varphi)$. So, $\text{card}(h_1) < \beta_2 - \max_{\text{size}}(\varphi)$. Together with $\text{card}(h_2) \leq \max_{\text{size}}(\varphi)$ and $h = h_1 + h_2$, we conclude that $\text{card}(h) < \beta_2$, and thus $(s, h) \models \neg\text{size} \geq \beta_2$.

Validity of $(\neg\langle \circledast \rangle(\varphi, \psi)) * \varphi \Rightarrow \neg\psi$. Let us assume $(s, h) \models (\neg\langle \circledast \rangle(\varphi, \psi)) * \varphi$. Consequently, there is a literal L of $\langle \circledast \rangle(\varphi, \psi)$ such that $(s, h) \models (\neg L) * \varphi$ holds. We show that $(s, h) \models \neg\psi$. Let h_1 and h_2 be two disjoint heaps such that $h = h_1 + h_2$, $(s, h_1) \models \neg L$ and $(s, h_2) \models \varphi$. We perform a case analysis on the shape of L . As in the previous part of the proof, recall that $x, y \in X$ and $\beta_1, \beta_2 \in [0, \alpha]$.

case: $L = x \neq x$. As φ and ψ are satisfiable, by definition of $\langle \circledast \rangle(\varphi, \psi)$, the fact that $x \neq x \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$ implies that one of the following three cases holds:

1. $x \hookrightarrow _ \wedge \neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ and $x \hookrightarrow y \subseteq_{\text{LIT}} \psi$.

From $x \hookrightarrow _ \wedge \neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ and $h_2 \subseteq h$, we have $s(x) \in \text{dom}(h)$ and $h(s(x)) \neq s(y)$. Thus $(s, h) \not\models x \hookrightarrow y$, and so, by $x \hookrightarrow y \subseteq_{\text{LIT}} \psi$, $(s, h) \models \neg\psi$.

2. $x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ and $\neg x \hookrightarrow y \subseteq_{\text{LIT}} \psi$.

From $x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ and $h_2 \subseteq h$, $h(s(x)) = s(y)$. Thus $(s, h) \models x \hookrightarrow y$ and so, by $\neg x \hookrightarrow y \subseteq_{\text{LIT}} \psi$, $(s, h) \models \neg\psi$.

3. $x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$ and $\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \psi$.

From $x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$ and $h_2 \subseteq h$, $s(x) \in \text{dom}(h)$. Thus $(s, h) \models x \hookrightarrow _$ and so, by $\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \psi$, $(s, h) \models \neg\psi$.

case: $L = x \sim y$, where $\sim \in \{=, \neq\}$. In this case, since $(s, h_1) \models \neg L$, we have $(s, h) \models \neg L$. Now, it cannot be that $L \subseteq_{\text{LIT}} \varphi$, as it would imply $(s, h) \models L$, which is contradictory. Therefore, by definition of $\langle \circledast \rangle(\varphi, \psi)$, we must have $L \subseteq_{\text{LIT}} \psi$. This implies $(s, h) \models \neg\psi$.

case: $L = x \hookrightarrow _$. By definition of $\langle \circledast \rangle(\varphi, \psi)$, both $\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$ and $x \hookrightarrow _ \subseteq_{\text{LIT}} \psi$ hold. From $(s, h_1) \models \neg x \hookrightarrow _$, we derive $s(x) \notin \text{dom}(h_1)$. By $\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$, $s(x) \notin \text{dom}(h_2)$. By $h = h_1 + h_2$, $s(x) \notin \text{dom}(h)$. As $x \hookrightarrow _ \subseteq_{\text{LIT}} \psi$, $(s, h) \models \neg\psi$.

case: $L = \neg x \hookrightarrow _$. As $(s, h) \models \neg L$, we have $s(x) \in \text{dom}(h_1)$. According to the definition of $\langle \circledast \rangle(\varphi, \psi)$, either $x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$ or $\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \psi$. The first case cannot hold, as it implies $s(x) \in \text{dom}(h_2)$ which contradicts the fact that h_1 and h_2 are disjoint. In the second case, from $s(x) \in \text{dom}(h_1)$ and $h_1 \subseteq h$, we have $(s, h) \models x \hookrightarrow _$. So, $(s, h) \models \neg\psi$.

case: $L = x \hookrightarrow y$. Then by definition of $\langle \circledast \rangle(\varphi, \psi)$, $\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$ and $x \hookrightarrow y \subseteq_{\text{LIT}} \psi$. From $(s, h_1) \models \neg L$, if $s(x) \in \text{dom}(h_1)$ then $h_1(s(x)) \neq s(y)$. As $\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$, $s(x) \notin \text{dom}(h_2)$ and therefore, by $h = h_1 + h_2$, $h(s(x)) \neq s(y)$. By $x \hookrightarrow y \subseteq_{\text{LIT}} \psi$, we derive $(s, h) \models \neg\psi$.

case: $L = \neg x \leftrightarrow y$. Then, by definition of $\langle \circledast \rangle(\varphi, \psi)$, $\neg x \leftrightarrow y \subseteq_{\text{LIT}} \psi$. From $(s, h_1) \models \neg L$ and $h_1 \subseteq h$, we derive $h(s(x)) = s(y)$. From $\neg x \leftrightarrow y \subseteq_{\text{LIT}} \psi$, we derive $(s, h) \models \neg \psi$.

case: $L = \text{size} \geq \beta_2 + 1 \doteq \beta_1$, where $\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$ and $\neg \text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$. Since we have $(s, h_1) \models \neg L$ and $(s, h_2) \models \varphi$, we derive (respectively) $\text{card}(h_1) \leq \beta_2 \doteq \beta_1$ and $\text{card}(h_2) < \beta_1$. From $h = h_1 + h_2$, we conclude that $\text{card}(h) < \beta_2$. From $\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$, we derive $(s, h) \models \neg \psi$.

case: $L = \neg \text{size} \geq \beta_2 \doteq \beta_1$, where $\neg \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$ and $\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$. From the fact that $(s, h_1) \models \neg L$ and $(s, h_2) \models \varphi$, we derive $\text{card}(h_1) \geq \beta_2 \doteq \beta_1$ and $\text{card}(h_2) \geq \beta_1$. So, $h = h_1 + h_2$ implies $\text{card}(h) \geq \beta_2$. By $\neg \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$, we derive $(s, h) \models \neg \psi$. \square

Before providing the proof for Lemma 6.18, we establish the existence of further derivations.

Lemma 6.21. Let $X \subseteq_{\text{fin}} \text{VAR}$ and let φ_{size} be a satisfiable conjunction of literals of the form $\text{size} \geq \beta_1$ or $\neg \text{size} \geq \beta_2$. The following axiom schema is admissible in $\mathcal{H}_C(*, \circledast)$:

$$(I_{6.21.1}^*) (\varphi_{\text{size}} \wedge \bigwedge_{x \in X} \neg x \leftrightarrow _) \circledast \top.$$

Proof. Notice that, since φ_{size} is satisfiable, for all $\beta_1, \beta_2 \in \mathbb{N}$ such that $\text{size} \geq \beta_1 \wedge \neg \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \varphi$, we must have $\beta_1 < \beta_2$. Moreover, thanks to (I_1^C) and $(I_{6.19.4}^*)$, without loss of generality, we can restrict ourselves to φ_{size} of the form:

- (1) $\varphi_{\text{size}} = \text{size} \geq \beta$ for some $\beta \geq 0$,
- (2) $\varphi_{\text{size}} = \neg(\text{size} \geq \beta)$ for some $\beta > 0$,
- (3) $\varphi_{\text{size}} = \text{size} \geq \beta_1 \wedge \neg(\text{size} \geq \beta_2)$ for some $\beta_2 > \beta_1$.

Indeed, given an arbitrary φ_{size} , every positive literal $\text{size} \geq \beta$ such that $\beta < \max_{\text{size}}(\varphi_{\text{size}})$ can be derived starting from $\text{size} \geq \max_{\text{size}}(\varphi_{\text{size}})$, by repeated applications of (I_1^C) . Similarly, let $\bar{\beta}$ be the smallest natural number such that $\neg \text{size} \geq \bar{\beta} \subseteq_{\text{LIT}} \varphi$, if any. Every literal $\neg \text{size} \geq \beta' \subseteq_{\text{LIT}} \varphi$ with $\beta' \geq \bar{\beta}$ can be derived from $\neg \text{size} \geq \bar{\beta}$, by repeated applications of the axiom (I_1^C) (taken in contrapositive form i.e. $\neg \text{size} \geq \beta \Rightarrow \neg \text{size} \geq \beta + 1$, which is derivable in \mathcal{H}_C by propositional reasoning).

We write $U(X)$ to denote the conjunction $\bigwedge_{x \in X} \neg x \leftrightarrow _$. Below, given $\beta \in \mathbb{N}$, we aim at deriving the formula $(\text{size} = \beta \wedge U(X)) \circledast \top$. Notice that this implies that $(I_{6.21.1}^*)$ is derivable in its instances (1)–(3):

case (1). Let $\varphi_{\text{size}} = \text{size} \geq \beta$.

1	$\text{size} = \beta \wedge U(X) \circledast \top$	Hypothesis
2	$\text{size} = \beta \wedge U(X) \Rightarrow \text{size} \geq \beta \wedge U(X)$	PC, def. of $\text{size} = \beta$
3	$(\text{size} = \beta \wedge U(X) \circledast \top) \Rightarrow (\text{size} \geq \beta \wedge U(X) \circledast \top)$	$(I_{6.19.4}^*), 2$
4	$\text{size} \geq \beta \wedge U(X) \circledast \top$	Modus Ponens, 1, 3

case (2). Let $\varphi_{\text{size}} = \neg \text{size} \geq \beta$. Since φ_{size} is satisfiable, we have $\beta \geq 1$.

1	$\text{size} = \beta - 1 \wedge U(X) \circledast \top$	Hypothesis
2	$\text{size} = \beta - 1 \wedge U(X) \Rightarrow \neg \text{size} \geq \beta \wedge U(X)$	PC, def. of $\text{size} = \beta - 1$
3	$(\text{size} = \beta - 1 \wedge U(X) \circledast \top) \Rightarrow (\neg \text{size} \geq \beta \wedge U(X) \circledast \top)$	$(I_{6.19.4}^*), 2$

4	$(\neg \text{size} \geq \beta \wedge U(X)) \multimap \top$	Modus Ponens, 1, 3
---	--	--------------------

case (3). Let $\varphi_{\text{size}} = \text{size} \geq \beta_1 \wedge \neg \text{size} \geq \beta_2$. Since φ_{size} is satisfiable, $\beta_2 > \beta_1$.

1	$\text{size} = \beta_2 - 1 \wedge U(X) \multimap \top$	Hypothesis
2	$\text{size} = \beta_2 - 1 \Rightarrow \text{size} \geq \beta_1$	repeated (I_1^C) , as $\beta_2 > \beta_1$
3	$\text{size} = \beta_2 - 1 \Rightarrow \neg \text{size} \geq \beta_2$	PC, def. of $\text{size} = \beta - 1$
4	$\text{size} = \beta_2 - 1 \wedge U(X) \Rightarrow \text{size} \geq \beta_1 \wedge \neg \text{size} \geq \beta_2 \wedge U(X)$	PC, 2, 3
5	$(\text{size} = \beta_2 - 1 \wedge U(X) \multimap \top) \Rightarrow$ $(\text{size} \geq \beta_1 \wedge \neg \text{size} \geq \beta_2 \wedge U(X) \multimap \top)$	$(I_{6.19.4}^*)$, 4
6	$\text{size} \geq \beta_1 \wedge \neg \text{size} \geq \beta_2 \wedge U(X) \multimap \top$	Modus Ponens, 1, 5

To conclude the proof, let us show that $\text{size} = \beta \wedge U(X) \multimap \top$ is derivable in $\mathcal{H}_C(*, \multimap)$. The proof is by induction on β , with two base cases, for $\beta = 0$ and $\beta = 1$.

base case: $\beta = 0$. In this case, $\text{size} = 0 = \text{size} \geq 0 \wedge \neg \text{size} \geq 1$. We have,

1	$(\text{emp} * \perp) \Rightarrow \text{emp} * (\text{emp} * \perp)$	$(*)_{ID}$
2	$\text{emp} * (\text{emp} * \perp) \Rightarrow \perp$	$(I_{6.19.3}^*)$
3	$(\text{emp} * \perp) \Rightarrow \perp$	$(\Rightarrow TR)$, 1, 2
4	$\text{emp} \multimap \top$	PC, 3, def. of \multimap
5	$x \hookrightarrow _ \Rightarrow \text{size} \geq 1$	(I_2^C)
6	$\text{emp} \Rightarrow \neg x \hookrightarrow _$	PC, 5, as $\text{size} \geq 1 = \neg \text{emp}$
7	$\text{emp} \Rightarrow U(X)$	PC, 6 used for all $x \in X$
8	$\text{emp} \Rightarrow \text{size} \geq 0 \wedge \neg(\text{size} \geq 1)$	PC, def. of $\text{size} \geq \beta$
9	$\text{emp} \Rightarrow \text{size} \geq 0 \wedge \neg(\text{size} \geq 1) \wedge U(X)$	PC, 7, 8
10	$(\text{emp} \multimap \top) \Rightarrow (\text{size} \geq 0 \wedge \neg(\text{size} \geq 1) \wedge U(X) \multimap \top)$	$(I_{6.19.4}^*)$, 9
11	$\text{size} \geq 0 \wedge \neg \text{size} \geq 1 \wedge U(X) \multimap \top$	Modus Ponens, 4, 10

base case: $\beta = 1$ This case corresponds exactly to the axiom (\multimap_∞^*) .

induction step: $\beta \geq 2$ First of all, we notice that the following formula is valid:

$$(\text{size} = 1 \wedge U(X)) * (\text{size} = \beta - 1 \wedge U(X)) \Rightarrow \text{size} = \beta \wedge U(X). \quad (\dagger)$$

Indeed, let (s, h) be a memory state satisfying the antecedent of the implication above. So, there are disjoint heaps h_1 and h_2 such that $h = h_1 + h_2$, $\text{card}(h_1) = 1$, $\text{card}(h_2) = \beta - 1$, and for every $x \in X$, $s(x) \notin \text{dom}(h_1)$ and $s(x) \notin \text{dom}(h_2)$. By $h = h_1 + h_2$, $\text{card}(h) = \text{card}(h_1) + \text{card}(h_2) = \beta$, and for every $x \in X$, $s(x) \notin \text{dom}(h)$. Thus, $(s, h) \models \text{size} = \beta \wedge U(X)$.

As (\dagger) can be seen as a formula in $\text{SL}(*, x \hookrightarrow _)$, by Theorem 6.16 it is derivable in $\mathcal{H}_C(*)$ and thus in $\mathcal{H}_C(*, \multimap)$. Let us derive $(\text{size} = \beta \wedge U(X)) \multimap \top$. Let us consider as induction

hypothesis the derivability of $(\text{size} = \beta - 1 \wedge U(X)) \multimap \top$. Therefore,

1	$\text{size} = \beta - 1 \wedge U(X) \multimap \top$	Induction Hypothesis
2	$(\text{size} = 1 \wedge U(X)) * (\text{size} = \beta - 1 \wedge U(X)) \Rightarrow \text{size} = \beta \wedge U(X)$	$(\dagger)_S$, see above
3	$\text{size} = 1 \wedge U(X) \multimap \top$	$(\overline{*})_{\infty}$
4	$\top \Rightarrow (\text{size} = \beta - 1 \wedge U(X) \multimap \top)$	PC, 1
5	$(\text{size} = 1 \wedge U(X) \multimap \top) \Rightarrow$ $(\text{size} = 1 \wedge U(X) \multimap (\text{size} = \beta - 1 \wedge U(X) \multimap \top))$	$(I_{6.19.5}^*)$, 4
6	$(\text{size} = 1 \wedge U(X) \multimap (\text{size} = \beta - 1 \wedge U(X) \multimap \top)) \Rightarrow$ $((\text{size} = 1 \wedge U(X)) * (\text{size} = \beta - 1 \wedge U(X)) \multimap \top)$	$(I_{6.19.8}^*)$
7	$((\text{size} = 1 \wedge U(X)) * (\text{size} = \beta - 1 \wedge U(X)) \multimap \top) \Rightarrow$ $(\text{size} = \beta \wedge U(X) \multimap \top)$	$(I_{6.19.4}^*)$, 2
8	$(\text{size} = 1 \wedge U(X) \multimap \top) \Rightarrow (\text{size} = \beta \wedge U(X) \multimap \top)$	$(\Rightarrow TR)$, 5, 6, 7
9	$\text{size} = \beta \wedge U(X) \multimap \top$	Modus Ponens, 3, 8 \square

Proof of Lemma 6.18. As in the statement of the lemma, let us consider $X \subseteq_{\text{fin}} \text{VAR}$ and $\alpha \geq \text{card}(X)$, and two core types φ and ψ in $\text{CoreTypes}(X, \alpha)$. We want to show that there is a conjunction $\chi \in \text{Conj}(\text{Core}(X, \alpha))$ such that $\vdash_{\mathcal{H}_C(*, *)} (\varphi \multimap \psi) \Leftrightarrow \chi$.

First of all, if φ or ψ is unsatisfiable, then $\vdash_{\mathcal{H}_C(*, *)} (\varphi \multimap \psi) \Rightarrow \perp$ by using Lemma 6.8 and the admissible axioms $(I_{6.19.4}^*)$ and $(I_{6.19.5}^*)$ from Lemma 6.19. Therefore, in this case, it is enough to take χ equal to $x \neq x$ to complete the proof. Otherwise, let us assume that φ and ψ are satisfiable. We consider $\chi \stackrel{\text{def}}{=} \langle \multimap \rangle(\varphi, \psi)$ (see Figure 6.9), and show that $\vdash_{\mathcal{H}_C(*, *)} (\varphi \multimap \psi) \Leftrightarrow \langle \multimap \rangle(\varphi, \psi)$. We derive each implication separately.

(\Rightarrow) : Given Lemma 6.20, the proof of $\vdash_{\mathcal{H}_C(*, *)} \varphi \multimap \psi \Rightarrow \langle \multimap \rangle(\varphi, \psi)$ is straightforward:

1	$\neg \langle \multimap \rangle(\varphi, \psi) * \varphi \Rightarrow \neg \psi$	Lemma 6.20, Theorem 6.16
2	$\neg \langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi * \neg \psi)$	(\neg_2) , 1
3	$\neg (\varphi * \neg \psi) \Rightarrow \langle \multimap \rangle(\varphi, \psi)$	PC, 2
4	$(\varphi \multimap \psi) \Rightarrow \langle \multimap \rangle(\varphi, \psi)$	Def. of \multimap , 3

(\Leftarrow) : We show that $\vdash_{\mathcal{H}_C(*, *)} \langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi \multimap \psi)$. First, notice that since $\langle \multimap \rangle(\varphi, \psi) * \varphi \Rightarrow \psi$ is valid (Lemma 6.20), it is derivable in $\mathcal{H}_C(*)$ (Theorem 6.16), and therefore, by the rule (\neg_2) , $\vdash_{\mathcal{H}_C(*, *)} \langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi * \psi)$. It follows that it is enough to show that $\langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi \multimap \top)$ is derivable in $\mathcal{H}_C(*, *)$. Indeed, from $\langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi \multimap \top)$ and $\langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi * \psi)$, we get, by $(I_{6.19.9}^*)$, that $\langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi \multimap \psi)$ is derivable too.

Thus, let us prove that $\langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi \multimap \top)$ is derivable. If $\langle \multimap \rangle(\varphi, \psi)$ is unsatisfiable, then from the completeness of \mathcal{H}_C with respect to Boolean combinations of core formulae (Theorem 6.9), we conclude that $\vdash_{\mathcal{H}_C} \langle \multimap \rangle(\varphi, \psi) \Rightarrow \perp$. Since $\mathcal{H}_C(*, *)$ extends \mathcal{H}_C , we have $\vdash_{\mathcal{H}_C(*, *)} \langle \multimap \rangle(\varphi, \psi) \Rightarrow \perp$. By propositional reasoning, $\vdash_{\mathcal{H}_C(*, *)} \langle \multimap \rangle(\varphi, \psi) \Rightarrow (\varphi \multimap \top)$. Other-

wise, let us assume that $\langle \circledast \rangle(\varphi, \psi)$ is satisfiable. Directly from the definition of $\langle \circledast \rangle(\varphi, \psi)$, the following simple facts hold.

1. φ, ψ and $\langle \circledast \rangle(\varphi, \psi)$ have exactly the same equalities and inequalities.
2. $\neg \text{size} \geq 0$ is not part of $\langle \circledast \rangle(\varphi, \psi)$, and therefore, following the definition of $\langle \circledast \rangle(\varphi, \psi)$, there are no $\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$ and $\neg \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$ with $\beta_1 \geq \beta_2$.
3. $x \neq x$ does not belong to $\langle \circledast \rangle(\varphi, \psi)$. In particular, by definition of $\langle \circledast \rangle(\varphi, \psi)$, none of the following conditions apply:
 - there is $x \in X$ such that $x \hookrightarrow_- \subseteq_{\text{LIT}} \varphi$ and $\neg x \hookrightarrow_- \subseteq_{\text{LIT}} \psi$,
 - there are $x, y \in X$ such that $x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ and $\neg x \hookrightarrow y \subseteq_{\text{LIT}} \psi$,
 - there are $x, y \in X$ such that $x \hookrightarrow_- \wedge \neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ and $x \hookrightarrow y \subseteq_{\text{LIT}} \psi$.

From (1), we know that $\langle \circledast \rangle(\varphi, \psi)$ and φ satisfy the same (in)equalities. Similarly to the proof of Lemma 6.14, let x_1, \dots, x_n be a maximal enumeration of representatives of the equivalence classes (one per equivalence class) such that $x_i \hookrightarrow_-$ occurs in φ . As it is maximal, for every $x \hookrightarrow_-$ in $\text{LIT}(\varphi)$ there is $i \in [1, n]$ such that x_i is syntactically equal to x . Moreover, by definition of $\langle \circledast \rangle(\varphi, \psi)$, for every $i \in [1, n]$, $\neg x_i \hookrightarrow_- \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. We show $\vdash_{\mathcal{H}_C(*, *)} \langle \circledast \rangle(\varphi, \psi) \Rightarrow (\varphi \circledast \top)$ by induction on the number j of variables $x \in X$ for which $x \hookrightarrow_- \subseteq_{\text{LIT}} \varphi$ holds.

base case: $j = 0$. In the base case, no formula $x \hookrightarrow_-$ occurs positively in φ . Since φ is a core type, this implies that for every $x \in X$, $\neg x \hookrightarrow_- \subseteq_{\text{LIT}} \varphi$. Moreover, since φ is satisfiable, for every $x, y \in X$, $\neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ (see (WEAK)). Therefore, the core type φ is syntactically equivalent (up to associativity and commutativity of conjunction) to the formula

$$\varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \wedge \varphi_{\not\hookrightarrow} \wedge \varphi_{(\text{in})\text{eq}},$$

where

- $\varphi_{\text{size}} \stackrel{\text{def}}{=} \wedge (\{\text{size} \geq \beta \subseteq_{\text{LIT}} \varphi\} \cup \{\neg \text{size} \geq \beta \subseteq_{\text{LIT}} \varphi\})$,
- $\varphi_{\neg \text{alloc}} \stackrel{\text{def}}{=} \wedge_{x \in X} \neg x \hookrightarrow_-$,
- $\varphi_{\not\hookrightarrow} \stackrel{\text{def}}{=} \wedge_{x, y \in X} \neg x \hookrightarrow y$,
- $\varphi_{(\text{in})\text{eq}} \stackrel{\text{def}}{=} \wedge \{x \sim y \subseteq_{\text{LIT}} \varphi \mid \sim \in \{=, \neq\}\}$.

Since φ is satisfiable, so is φ_{size} . We show that $\vdash_{\mathcal{H}_C(*, *)} \varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \wedge \varphi_{\not\hookrightarrow} \circledast \top$:

1	$\varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \circledast \top$	$(I_{6.21.1}^*)$
2	$\neg x \hookrightarrow_- \Rightarrow \neg x \hookrightarrow y$	(WEAK) , PC
3	$\varphi_{\neg \text{alloc}} \Rightarrow \varphi_{\not\hookrightarrow}$	PC, repeated 2
4	$\varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \Rightarrow \varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \wedge \varphi_{\not\hookrightarrow}$	PC, 3
5	$(\varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \circledast \top) \Rightarrow (\varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \wedge \varphi_{\not\hookrightarrow} \circledast \top)$	$(I_{6.19.4}^*)$, 4
6	$\varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \wedge \varphi_{\not\hookrightarrow} \circledast \top$	Modus Ponens, 1, 5

Now, let us treat the formula $\varphi_{(\text{in})\text{eq}}$. From the definition of $\langle \circledast \rangle(\varphi, \psi)$, we have $\varphi_{(\text{in})\text{eq}} \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, and so by propositional reasoning, $\vdash_{\mathcal{H}_C(*, *)} \langle \circledast \rangle(\varphi, \psi) \Rightarrow \varphi_{(\text{in})\text{eq}}$. This allows us to conclude that

$$\vdash_{\mathcal{H}_C(*, *)} \langle \circledast \rangle(\varphi, \psi) \Rightarrow (\varphi_{\text{size}} \wedge \varphi_{\neg \text{alloc}} \wedge \varphi_{\not\hookrightarrow} \wedge \varphi_{(\text{in})\text{eq}} \circledast \top), \quad (\dagger)$$

by induction on the number of literals $x \sim y$ appearing in $\varphi_{(in)eq}$, and by relying on the two theorem ($I_{6.19.10}^*$). In the base case, $\varphi_{(in)eq} = \top$, and so

7	$\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \Rightarrow \varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi_{(in)eq}$	PC
8	$(\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim}) \Rightarrow (\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi_{(in)eq} \rightsquigarrow \top)$	$(I_{6.19.4}^*)$, 7
9	$\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi_{(in)eq} \rightsquigarrow \top$	Modus Ponens, 6, 8
10	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi_{(in)eq} \rightsquigarrow \top)$	PC, 9

In the induction step, let $\varphi_{(in)eq} = \varphi'_{(in)eq} \wedge x \sim y$, where $x \sim y \not\subseteq_{LIT} \varphi'_{(in)eq}$. We have,

1	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi'_{(in)eq} \rightsquigarrow \top)$	Induction Hypothesis
2	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow x \sim y$	PC, as $\varphi_{(in)eq} \subseteq_{LIT} \langle \rightsquigarrow \rangle(\varphi, \psi)$
3	$x \sim y \wedge (\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi'_{(in)eq} \rightsquigarrow \top) \Rightarrow$ $(\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi'_{(in)eq} \wedge x \sim y \rightsquigarrow \top)$	$(I_{6.19.10}^*)$
4	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi'_{(in)eq} \wedge x \sim y \rightsquigarrow \top)$	PC, 1, 2, 3
5	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi_{(in)eq} \rightsquigarrow \top)$	Def. of $\varphi'_{(in)eq}$, 4

Since $\varphi_{size} \wedge \varphi_{\neg alloc} \wedge \varphi_{\not\sim} \wedge \varphi_{(in)eq}$ is equivalent to φ , from (\dagger) and by $(I_{6.19.4}^*)$, we conclude that $\vdash_{H_C(*, \rightsquigarrow)} \langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\varphi \rightsquigarrow \top)$.

induction step: $j \geq 1$. In this case, let $i \in [1, n]$ such that $x_i \hookrightarrow_- \subseteq_{LIT} \varphi$ and thus, by definition of $\langle \rightsquigarrow \rangle(\varphi, \psi)$, $\neg x_i \hookrightarrow_- \subseteq_{LIT} \langle \rightsquigarrow \rangle(\varphi, \psi)$. We define the formula:

$$\text{ATOM}(x_i) \stackrel{\text{def}}{=} \begin{cases} x_i \hookrightarrow y \wedge \text{size} = 1 & \text{if } x_i \hookrightarrow y \subseteq_{LIT} \varphi, \text{ for some } y \in X \\ x_i \hookrightarrow_- \wedge \text{size} = 1 \wedge \bigwedge_{y \in X} \neg x_i \hookrightarrow y & \text{otherwise} \end{cases}$$

Notice that, if there is $y \in X$ such that $x_i \hookrightarrow y \subseteq_{LIT} \varphi$, then the axiom schema (\hookrightarrow^*) can be instantiated to $\neg x_i \hookrightarrow_- \Rightarrow (\text{ATOM}(x_i) \rightsquigarrow \top)$. Otherwise (for all $y \in X$, $x_i \hookrightarrow y \not\subseteq_{LIT} \varphi$) this formula is an instantiation of the axiom schema (ALLOC^*) . This allows us to show the following theorem:

$$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\text{ATOM}(x_i) \rightsquigarrow \langle \rightsquigarrow \rangle(\varphi, \psi) * \text{ATOM}(x_i)) \quad (\ddagger)$$

1	$\neg x_i \hookrightarrow_- \Rightarrow (\text{ATOM}(x_i) \rightsquigarrow \top)$	$(\hookrightarrow^*) / (\text{ALLOC}^*)$
2	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow \neg x_i \hookrightarrow_-$	Def. of $\langle \rightsquigarrow \rangle(\varphi, \psi)$, PC
3	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\text{ATOM}(x_i) \rightsquigarrow \top)$	$(\Rightarrow \text{TR})$, 1, 2
4	$\langle \rightsquigarrow \rangle(\varphi, \psi) * \text{ATOM}(x_i) \Rightarrow \langle \rightsquigarrow \rangle(\varphi, \psi) * \text{ATOM}(x_i)$	PC
5	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\text{ATOM}(x_i) \rightsquigarrow \langle \rightsquigarrow \rangle(\varphi, \psi) * \text{ATOM}(x_i))$	(\rightsquigarrow_2) , 4
6	$\langle \rightsquigarrow \rangle(\varphi, \psi) \Rightarrow (\text{ATOM}(x_i) \rightsquigarrow \langle \rightsquigarrow \rangle(\varphi, \psi) * \text{ATOM}(x_i))$	$(I_{6.19.9}^*)$, 3, 5, PC

From the hypothesis $\text{card}(X) \leq \alpha$, together with $x_i \hookrightarrow_- \subseteq_{LIT} \varphi$ and the fact that φ is

satisfiable, we have $\text{max_size}(\varphi) \geq 1$ (see (I_2^C) , instantiated with $X = \{x_i\}$). In order to show that $\vdash_{\mathcal{H}_{C^*,-*}} \langle \rightarrow \ast \rangle(\varphi, \psi) \Rightarrow (\varphi \rightarrow \ast \top)$, we split the proof depending on whether $\text{max_size}(\varphi) < \alpha$ holds.

case: $\text{max_size}(\varphi) < \alpha$. Since φ is a satisfiable core type in $\text{CoreTypes}(X, \alpha)$, by definition of $\text{max_size}(\cdot)$, we have $\text{size} \geq \text{max_size}(\varphi) \wedge \neg\text{size} \geq \text{max_size}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi$. Below, we consider the formula φ' obtained from φ by:

- replacing $\text{size} \geq \text{max_size}(\varphi) \subseteq_{\text{LIT}} \varphi$ with $\neg\text{size} \geq \text{max_size}(\varphi)$,
- for every $x \in X$ such that $x = x_i \subseteq_{\text{LIT}} \varphi$, replacing every literal $x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$ with $\neg x \hookrightarrow _$, and every literal $x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ with $\neg x \hookrightarrow y$, where $y \in X$.

Explicitly,

$$\begin{aligned} \varphi' \stackrel{\text{def}}{=} & \bigwedge \{x \sim y \subseteq_{\text{LIT}} \varphi \mid \sim \in \{=, \neq\}\} \wedge \bigwedge \{x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi \mid x \neq x_i \subseteq_{\text{LIT}} \varphi\} \wedge \\ & \bigwedge \{\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{\neg x \hookrightarrow _ \mid x = x_i \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{x \hookrightarrow y \subseteq_{\text{LIT}} \varphi \mid x \neq x_i \subseteq_{\text{LIT}} \varphi\} \wedge \\ & \bigwedge \{\neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{\neg x \hookrightarrow y \mid x = x_i \wedge x \hookrightarrow y \subseteq_{\text{LIT}} \varphi\} \wedge \neg\text{size} \geq \text{max_size}(\varphi) \wedge \\ & \bigwedge \{\text{size} \geq \beta \subseteq_{\text{LIT}} \varphi \mid \beta < \text{max_size}(\varphi)\} \wedge \bigwedge \{\neg\text{size} \geq \beta \subseteq_{\text{LIT}} \varphi\}. \end{aligned}$$

The formula φ' enjoys the two following properties:

- A. φ' is a satisfiable core type in $\text{CoreTypes}(X, \alpha)$.
- B. $\text{ATOM}(x_i) * \varphi' \Rightarrow \varphi$ is valid.

Proof of (A). Since φ' is obtained from φ simply by changing the polarity of some of the literals in $\text{LIT}(\varphi)$, clearly φ' is in $\text{CoreTypes}(X, \alpha)$. To show that φ' is satisfiable, we rely on the fact that φ is satisfiable. Let (s, h) be a memory state satisfying φ . Since $x_i \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$, we conclude that $s(x_i) \in \text{dom}(h)$. Let us consider the disjoint heaps h_1 and h_2 such that $h = h_1 + h_2$ and $\text{dom}(h_1) = \{s(x_i)\}$. We show that $(s, h_2) \models \varphi'$ by considering every $L \in \text{LIT}(\varphi')$ and showing that $(s, h_2) \models L$.

case: $L = x \sim y$, where $\sim \in \{=, \neq\}$. By definition of φ' , $(s, h) \models L$ and therefore $s(x) \sim s(y)$. Thus, $(s, h_2) \models L$.

case: $L = \neg x \hookrightarrow _$. If $x = x_i \subseteq_{\text{LIT}} \varphi$ then $s(x) \in \text{dom}(h_1)$, and therefore, by $h_1 \perp h_2$, $s(x) \notin \text{dom}(h_2)$. So, $(s, h_2) \models \neg x \hookrightarrow _$. Otherwise ($x \neq x_i \subseteq_{\text{LIT}} \varphi$), by definition of φ' , we have $\neg x \hookrightarrow _ \subseteq_{\text{LIT}} \varphi$. So $s(x) \notin \text{dom}(h)$ and, from $h_2 \subseteq h$, we conclude that $(s, h_2) \models \neg x \hookrightarrow _$.

case: $L = \neg x \hookrightarrow y$. Similar to the previous case. Briefly, if $x = x_i \subseteq_{\text{LIT}} \varphi$ then, by definition of $\text{ATOM}(x_i)$, $(s, h_2) \not\models x \hookrightarrow _$, which implies $(s, h_2) \models \neg x \hookrightarrow y$. Otherwise, by definition of φ' , $\neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ and thus $(s, h) \models \neg x \hookrightarrow y$. From $h_2 \subseteq h$, we conclude that $(s, h_2) \models \neg x \hookrightarrow y$.

case: $L = x \hookrightarrow _$. By definition of φ' , $x \hookrightarrow _ \wedge x \neq x_i \subseteq_{\text{LIT}} \varphi$. Therefore $s(x) \in \text{dom}(h)$ and, by definition of $\text{ATOM}(x_i)$, $s(x) \notin \text{dom}(h_1)$. Since $h = h_1 + h_2$, we conclude that $(s, h_2) \models x \hookrightarrow _$.

case: $L = x \hookrightarrow y$. Similar to the previous case. By definition of φ' , we have $x \hookrightarrow y \wedge x \neq x_i \subseteq_{\text{LIT}} \varphi$. Thus, $h(s(x)) = s(y)$. By definition of $\text{ATOM}(x_i)$, $s(x) \in \text{dom}(h_2)$ and thus $h_2(s(x)) = s(y)$. So, $(s, h_2) \models x \hookrightarrow y$.

case: $L = \text{size} \geq \beta$. By definition of φ' , $\beta < \text{max_size}(\varphi)$. Since $(s, h) \models \varphi$, we have $\text{card}(h) \geq \text{max_size}(\varphi)$. By definition of $\text{ATOM}(x_i)$ and from $h = h_1 + h_2$, $\text{card}(h_2) = \text{card}(h) - 1 \geq \text{max_size}(\varphi) - 1 \geq \beta$. So, $(s, h_2) \models \text{size} \geq \beta$.

case: $L = \neg\text{size} \geq \beta$. By definition of φ' , $\neg\text{size} \geq \beta \subseteq_{\text{LIT}} \varphi$ or $\beta = \max_{\text{size}}(\varphi)$. In the former case, since φ is satisfiable, we know that $\beta > \max_{\text{size}}(\varphi)$. In both cases, $\beta \geq \max_{\text{size}}(\varphi)$. Moreover, as $(s, h) \models \varphi$ and $\neg\text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi$, we have $\text{card}(h) \leq \max_{\text{size}}(\varphi)$. Since $\text{card}(h_1) = 1$, by $h = h_1 + h_2$ we conclude that $\text{card}(h_2) < \max_{\text{size}}(\varphi) \leq \beta$. Therefore, $(s, h_2) \models \neg\text{size} \geq \beta$.

Proof of (B). Let $(s, h) \models \text{ATOM}(\mathbf{x}_i) * \varphi'$. There are h_1 and h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \text{ATOM}(\mathbf{x}_i)$ and $(s, h_2) \models \varphi'$. By definition of $\text{ATOM}(\mathbf{x}_i)$, $\text{dom}(h_1) = \{s(\mathbf{x}_i)\}$. To prove (B), we show that $(s, h) \models L$, for every literal $L \in \text{LIT}(\varphi)$.

case: $L = \mathbf{x} \sim \mathbf{y}$, **where** $\sim \in \{=, \neq\}$. By definition of φ' , $(s, h_2) \models L$ and therefore $s(\mathbf{x}) \sim s(\mathbf{y})$. Thus, $(s, h) \models L$.

case: $L = \neg\mathbf{x} \hookrightarrow __$. By definition of $\text{ATOM}(\mathbf{x}_i)$, $\mathbf{x}_i \hookrightarrow __ \subseteq_{\text{LIT}} \varphi$ and therefore $s(\mathbf{x}) \notin \text{dom}(h_1)$. By definition of φ' , for every $\mathbf{y} \in \mathbf{X}$, $\mathbf{y} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi'$ implies $\mathbf{y} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi$. Therefore, $s(\mathbf{x}) \notin \text{dom}(h_2)$. We have $s(\mathbf{x}) \notin \text{dom}(h)$, and so $(s, h) \models \neg\mathbf{x} \hookrightarrow __$.

case: $L = \neg\mathbf{x} \hookrightarrow \mathbf{y}$. Similar to the previous case. Briefly, by definition of $\text{ATOM}(\mathbf{x}_i)$, $(s, h_1) \models \neg\mathbf{x} \hookrightarrow \mathbf{y}$. By definition of φ' , $(s, h_2) \models \neg\mathbf{x} \hookrightarrow \mathbf{y}$. So, $(s, h) \models \neg\mathbf{x} \hookrightarrow \mathbf{y}$.

case: $L = \mathbf{x} \hookrightarrow __$. If $\mathbf{x} = \mathbf{x}_i \subseteq_{\text{LIT}} \varphi$, then $s(\mathbf{x}) = s(\mathbf{x}_i)$ (first case of the proof), and by definition of $\text{ATOM}(\mathbf{x}_i)$, $s(\mathbf{x}) \in \text{dom}(h_1)$. As $h_1 \subseteq h$, we conclude that $(s, h) \models \mathbf{x} \hookrightarrow __$. Otherwise, if $\mathbf{x} \neq \mathbf{x}_i \subseteq_{\text{LIT}} \varphi$, then by definition of φ' we have $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi'$. This implies $s(\mathbf{x}) \in \text{dom}(h_2)$. As $h_2 \subseteq h$, we derive $(s, h) \models \mathbf{x} \hookrightarrow __$.

case: $L = \mathbf{x} \hookrightarrow \mathbf{y}$. Similar to the previous case. Briefly, if $\mathbf{x} = \mathbf{x}_i \subseteq_{\text{LIT}} \varphi$ then, by definition of $\text{ATOM}(\mathbf{x}_i)$, $(s, h_1) \models \mathbf{x} \hookrightarrow \mathbf{y}$ and so $(s, h) \models \mathbf{x} \hookrightarrow \mathbf{y}$. Otherwise ($\mathbf{x} \neq \mathbf{x}_i \subseteq_{\text{LIT}} \varphi$), $\mathbf{x} \hookrightarrow \mathbf{y} \subseteq_{\text{LIT}} \varphi'$ and therefore $(s, h_2) \models \mathbf{x} \hookrightarrow \mathbf{y}$. So, $(s, h) \models \mathbf{x} \hookrightarrow \mathbf{y}$.

case: $L = \text{size} \geq \beta$. If $\beta < \max_{\text{size}}(\varphi)$, then directly by definition of φ' , we have $(s, h_2) \models \text{size} \geq \beta$. From $h_2 \subseteq h$, we conclude that $(s, h) \models \text{size} \geq \beta$. Otherwise, $\beta = \max_{\text{size}}(\varphi)$. Recall that $\max_{\text{size}}(\varphi) \geq 1$ and so, by definition of φ' , $\text{size} \geq \max_{\text{size}}(\varphi) - 1 \subseteq_{\text{LIT}} \varphi'$. Thus, $\text{card}(h_1) \geq \max_{\text{size}}(\varphi) - 1$. By definition of $\text{ATOM}(\mathbf{x}_i)$ we have $\text{card}(h_1) = 1$. As $h = h_1 + h_2$, we conclude that $(s, h) \models \text{size} \geq \max_{\text{size}}(\varphi)$.

case: $L = \neg\text{size} \geq \beta$. As φ is satisfiable, $\beta > \max_{\text{size}}(\varphi)$. By definition of φ' , $\neg\text{size} \geq \max_{\text{size}}(\varphi) \subseteq_{\text{LIT}} \varphi'$, and so $\text{card}(h_2) < \max_{\text{size}}(\varphi)$. Since we have $\text{card}(h_1) = 1$, $\text{card}(h) \leq \max_{\text{size}}(\varphi) < \beta$. So, $(s, h) \models \neg\text{size} \geq \beta$.

The property (A) allows us to consider the formula $\langle \circledast \rangle(\varphi', \psi)$, and show that

C. $\langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(\mathbf{x}_i) \Rightarrow \langle \circledast \rangle(\varphi', \psi)$ is valid.

Proof of (C). Figure 6.10 recalls the definition of $\langle \circledast \rangle(\varphi', \psi)$. First of all, notice that it cannot be that there is $\mathbf{x} \in \mathbf{X}$ such that $\mathbf{x} \neq \mathbf{x} \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi', \psi)$. Indeed, *ad absurdum*, suppose the opposite. By definition of $\langle \circledast \rangle(\varphi', \psi)$, this implies that (1) $\mathbf{x} \hookrightarrow __ \wedge \neg\mathbf{x} \hookrightarrow \mathbf{y} \subseteq_{\text{LIT}} \varphi'$ and $\mathbf{x} \hookrightarrow \mathbf{y} \subseteq_{\text{LIT}} \psi$, (2) $\mathbf{x} \hookrightarrow \mathbf{y} \subseteq_{\text{LIT}} \varphi'$ and $\neg\mathbf{x} \hookrightarrow \mathbf{y} \subseteq_{\text{LIT}} \psi$, or (3) $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi'$ and $\neg\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \psi$. By definition of φ' , this implies that (1) $\mathbf{x} \hookrightarrow __ \wedge \neg\mathbf{x} \hookrightarrow \mathbf{y} \subseteq_{\text{LIT}} \varphi$, (2) $\mathbf{x} \hookrightarrow \mathbf{y} \subseteq_{\text{LIT}} \varphi$ or (3) $\mathbf{x} \hookrightarrow __ \subseteq_{\text{LIT}} \varphi$. However, by definition of $\langle \circledast \rangle(\varphi, \psi)$, this implies that $\mathbf{x} \neq \mathbf{x} \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, in contradiction with the satisfiability of $\langle \circledast \rangle(\varphi, \psi)$. Therefore, below we assume that for all $\mathbf{x} \in \mathbf{X}$, $\mathbf{x} \neq \mathbf{x} \not\subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi', \psi)$.

Let $(s, h) \models \langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(\mathbf{x}_i)$. Thus, $(s, h_1) \models \langle \circledast \rangle(\varphi, \psi)$ and $(s, h_2) \models \text{ATOM}(\mathbf{x}_i)$ for some h_1 and h_2 such that $h = h_1 + h_2$. By definition of $\text{ATOM}(\mathbf{x}_i)$, $\text{dom}(h_2) = \{s(\mathbf{x}_i)\}$. To prove (C), we show that $(s, h) \models L$, for every literal $L \in \text{LIT}(\langle \circledast \rangle(\varphi', \psi))$.

$$\begin{aligned}
& \wedge \{x \sim y \subseteq_{\text{LIT}} \{\varphi' \mid \psi\} \mid \sim \in \{=, \neq\}\} \quad \wedge \wedge \left\{ x \hookrightarrow - \mid \begin{array}{l} \neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi' \\ x \hookrightarrow - \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
& \wedge \wedge \{\neg x \hookrightarrow - \subseteq_{\text{LIT}} \psi\} \quad \wedge \wedge \{\neg x \hookrightarrow - \mid x \hookrightarrow - \subseteq_{\text{LIT}} \varphi'\} \\
& \wedge \wedge \{\neg x \hookrightarrow y \subseteq_{\text{LIT}} \psi\} \quad \wedge \wedge \left\{ x \hookrightarrow y \mid \begin{array}{l} \neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi' \\ x \hookrightarrow y \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
& \wedge \wedge \left\{ x \neq x \mid \begin{array}{l} x \hookrightarrow - \wedge \neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi' \\ x \hookrightarrow y \subseteq_{\text{LIT}} \psi \end{array} \right\} \quad \wedge \wedge \left\{ \text{size} \geq \beta_2 + 1 - \beta_1 \mid \begin{array}{l} \neg \text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi' \\ \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
& \wedge \wedge \left\{ x \neq x \mid \begin{array}{l} x \hookrightarrow y \subseteq_{\text{LIT}} \varphi' \\ \neg x \hookrightarrow y \subseteq_{\text{LIT}} \psi \end{array} \right\} \quad \wedge \wedge \left\{ \neg \text{size} \geq \beta_2 - \beta_1 \mid \begin{array}{l} \text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi' \\ \neg \text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi \end{array} \right\} \\
& \wedge \wedge \left\{ x \neq x \mid \begin{array}{l} x \hookrightarrow - \subseteq_{\text{LIT}} \varphi' \\ \neg x \hookrightarrow - \subseteq_{\text{LIT}} \psi \end{array} \right\}
\end{aligned}$$

Figure 6.10: The formula $\langle \circledast \rangle(\varphi', \psi)$.

case: $L = x \sim y$, where $\sim \in \{=, \neq\}$. By definition of $\langle \circledast \rangle(\varphi', \psi)$, $L \subseteq_{\text{LIT}} \{\varphi' \mid \psi\}$ and so, by definition of φ' , $L \subseteq_{\text{LIT}} \{\varphi \mid \psi\}$. By definition of $\langle \circledast \rangle(\varphi, \psi)$, $L \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. From $(s, h_1) \models \langle \circledast \rangle(\varphi, \psi)$ we derive $s(x) \sim s(y)$. So, $(s, h) \models L$.

case: $L = \neg x \hookrightarrow -$. By definition of $\langle \circledast \rangle(\varphi', \psi)$, either $\neg x \hookrightarrow - \subseteq_{\text{LIT}} \psi$ or $x \hookrightarrow - \subseteq_{\text{LIT}} \varphi'$. In the first case, by definition of $\langle \circledast \rangle(\varphi, \psi)$, $\neg x \hookrightarrow - \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, and therefore $s(x) \notin \text{dom}(h_1)$. Moreover, since $\langle \circledast \rangle(\varphi, \psi)$ is satisfiable, $x \hookrightarrow - \not\subseteq_{\text{LIT}} \varphi$ (otherwise we would have $x \neq x \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$). Therefore, by definition of $\text{ATOM}(x_i)$, we conclude that $s(x) \notin \text{dom}(h_2)$. From $h = h_1 + h_2$, we derive $s(x) \notin \text{dom}(h)$, and thus $(s, h) \models \neg x \hookrightarrow -$.

In the second case, $(x \hookrightarrow - \subseteq_{\text{LIT}} \varphi')$, by definition of φ' we have $x \hookrightarrow - \subseteq_{\text{LIT}} \varphi$ and $x \neq x_i \subseteq_{\text{LIT}} \varphi$. By definition of $\text{ATOM}(x_i)$, $s(x) \notin \text{dom}(h_2)$. By definition of $\langle \circledast \rangle(\varphi, \psi)$, $\neg x \hookrightarrow - \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, and therefore $s(x) \notin \text{dom}(h_1)$. Again, by $h = h_1 + h_2$, we have $(s, h) \models \neg x \hookrightarrow -$.

case: $L = \neg x \hookrightarrow y$. Following the definition of $\langle \circledast \rangle(\varphi', \psi)$, $\neg x \hookrightarrow y \subseteq_{\text{LIT}} \psi$ and therefore $\neg x \hookrightarrow y \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. Therefore, $(s, h_1) \models \neg x \hookrightarrow y$. Since $\langle \circledast \rangle(\varphi, \psi)$ is satisfiable, $\neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$. By definition of $\text{ATOM}(x_i)$, we derive $(s, h_2) \models \neg x \hookrightarrow y$. From $h = h_1 + h_2$, $(s, h) \models \neg x \hookrightarrow y$.

case: $L = x \hookrightarrow -$. By definition of $\langle \circledast \rangle(\varphi', \psi)$, $\neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi'$ and $x \hookrightarrow - \subseteq_{\text{LIT}} \psi$. First, let us suppose $x \hookrightarrow - \subseteq_{\text{LIT}} \varphi$. By definition of φ' , $x = x_i \subseteq_{\text{LIT}} \varphi$ and so, by definition of $\text{ATOM}(x_i)$, $s(x) \in \text{dom}(h_2)$. From $h_2 \subseteq h$, $(s, h) \models x \hookrightarrow -$. Otherwise ($\neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi$), by definition of $\langle \circledast \rangle(\varphi, \psi)$, $x \hookrightarrow - \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. So, $s(x) \in \text{dom}(h_1)$, and by $h_1 \subseteq h$, $(s, h) \models x \hookrightarrow -$.

case: $L = x \hookrightarrow y$. Similar to the previous case. By definition of $\langle \circledast \rangle(\varphi', \psi)$, we have $\neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi'$ and $x \hookrightarrow y \subseteq_{\text{LIT}} \psi$. First, let us assume $x \hookrightarrow - \subseteq_{\text{LIT}} \varphi$. By definition of φ' , $x = x_i \subseteq_{\text{LIT}} \varphi$. By definition of $\text{ATOM}(x_i)$, $s(x) \in \text{dom}(h_2)$. *Ad absurdum*, suppose $h(s(x)) \neq s(y)$. By definition of $\text{ATOM}(x_i)$, we have that $x \hookrightarrow - \wedge \neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$. However, from $x \hookrightarrow y \subseteq_{\text{LIT}} \psi$, this implies $x \neq x \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$, which contradicts the satisfiability of $\langle \circledast \rangle(\varphi, \psi)$. Thus, $h(s(x)) = s(y)$ and, from $h_2 \subseteq h$, we conclude that $(s, h) \models x \hookrightarrow y$. Otherwise ($\neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi$), by definition of $\langle \circledast \rangle(\varphi, \psi)$, $x \hookrightarrow y \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. So, $h_1(s(x)) = s(y)$, and by $h_1 \subseteq h$, we derive $(s, h) \models x \hookrightarrow y$.

case: $L = \text{size} \geq \beta_2 + 1 - \beta_1$, where $\neg\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi'$ and $\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$.

By definition of φ' , $\neg\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi$, and so $\beta_1 > \max_{\text{size}}(\varphi)$, since φ is satisfiable. By definition of $\langle \circledast \rangle(\varphi, \psi)$ and as $\neg\text{size} \geq \max_{\text{size}}(\varphi) + 1 \subseteq_{\text{LIT}} \varphi$,

$$\text{size} \geq \beta_2 + 1 - (\max_{\text{size}}(\varphi) + 1) \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi),$$

which in turn implies $\text{card}(h_1) \geq \beta_2 - \max_{\text{size}}(\varphi)$. By definition of $\text{ATOM}(x_i)$, $\text{card}(h_2) \geq 1$. By $h = h_1 + h_2$, $\text{card}(h) \geq (\beta_2 - \max_{\text{size}}(\varphi)) + 1 \geq (\beta_2 + 1) - \max_{\text{size}}(\varphi)$. As $\beta_1 > \max_{\text{size}}(\varphi)$, $(s, h) \models \text{size} \geq \beta_2 + 1 - \beta_1$.

case: $L = \neg\text{size} \geq \beta_2 - \beta_1$, where $\text{size} \geq \beta_1 \subseteq_{\text{LIT}} \varphi'$ and $\neg\text{size} \geq \beta_2 \subseteq_{\text{LIT}} \psi$.

By definition of φ' , $\beta_1 < \max_{\text{size}}(\varphi)$. By definition of $\langle \circledast \rangle(\varphi, \psi)$,

$$\neg\text{size} \geq \beta_2 - \max_{\text{size}}(\varphi) \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi).$$

As $\langle \circledast \rangle(\varphi, \psi)$ is satisfiable, $\beta_2 > \max_{\text{size}}(\varphi)$. So, $\text{card}(h_1) < \beta_2 - \max_{\text{size}}(\varphi)$. By definition of $\text{ATOM}(x_i)$, $\text{card}(h_2) \leq 1$. From $h = h_1 + h_2$, we conclude that $\text{card}(h) < (\beta_2 - \max_{\text{size}}(\varphi)) + 1$. As $\beta_1 < \max_{\text{size}}(\varphi)$, we have $\beta_2 - \max_{\text{size}}(\varphi) + 1 \leq \beta_2 - \beta_1$. Thus, $(s, h) \models \neg\text{size} \geq \beta_2 - \beta_1$.

We are now ready to prove that $\langle \circledast \rangle(\varphi, \psi) \Rightarrow (\varphi \circledast \top)$. Notice that, by completeness of $\mathcal{H}_C(*)$ (Theorem 6.16), we conclude that the tautologies in (B) and (C) are derivable in $\mathcal{H}_C(*, \neg)$. Moreover, notice that $\neg x_i \hookrightarrow_- \subseteq_{\text{LIT}} \varphi'$ and, for every $y \in X$, $\neg y \hookrightarrow_- \subseteq_{\text{LIT}} \varphi$ implies $\neg y \hookrightarrow_- \subseteq_{\text{LIT}} \varphi'$. This allows us to rely on the induction hypothesis, and derive $\vdash_{\mathcal{H}_C(*, \neg)} \langle \circledast \rangle(\varphi', \psi) \Rightarrow (\varphi' \circledast \top)$. The derivation of $\langle \circledast \rangle(\varphi, \psi) \Rightarrow (\varphi \circledast \top)$ is given below:

1	$\langle \circledast \rangle(\varphi', \psi) \Rightarrow (\varphi' \circledast \top)$	Induction hypothesis
2	$(\text{ATOM}(x_i) * \varphi') \Rightarrow \varphi$	(B), Theorem 6.16
3	$((\langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(x_i)) \Rightarrow \langle \circledast \rangle(\varphi', \psi))$	(C), Theorem 6.16
4	$((\langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(x_i)) \Rightarrow (\varphi' \circledast \top))$	($\Rightarrow \text{TR}$), 1, 3
5	$\langle \circledast \rangle(\varphi, \psi) \Rightarrow (\text{ATOM}(x_i) \circledast ((\langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(x_i))))$	(\ddagger)
6	$(\text{ATOM}(x_i) \circledast ((\langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(x_i)))) \Rightarrow (\text{ATOM}(x_i) \circledast (\varphi' \circledast \top))$	$(I_{6.19.5})^*$, 4
7	$(\text{ATOM}(x_i) \circledast (\varphi' \circledast \top)) \Rightarrow ((\text{ATOM}(x_i) * \varphi') \circledast \top)$	$(I_{6.19.8})^*$
8	$((\text{ATOM}(x_i) * \varphi') \circledast \top) \Rightarrow (\varphi \circledast \top)$	$(I_{6.19.4})^*$, 2
9	$\langle \circledast \rangle(\varphi, \psi) \Rightarrow (\varphi \circledast \top)$	$(\Rightarrow \text{TR})$, 5, 6, 7, 8

case: $\max_{\text{size}}(\varphi) = \alpha$. In this case, we have $\text{size} \geq \alpha \subseteq_{\text{LIT}} \varphi$, where we recall that $\alpha = \max_{\text{size}}(\varphi) \geq 1$. Following the developments of the previous case, we would like to define a formula φ' for which the formula $\varphi' * \text{ATOM}(x_i) \Leftrightarrow \varphi$ is valid. However, since φ is in $\text{CoreTypes}(X, \alpha)$, we cannot hope for φ' to be a core type in $\text{CoreTypes}(X, \alpha)$. Indeed, because of $\text{size} \geq \alpha \subseteq_{\text{LIT}} \varphi$, in order to achieve the valid formula above we must differentiate between the case where φ is satisfied by a memory state (s, h) such that $\text{card}(h) > \alpha$, to the case where $\text{card}(h) = \alpha$. Therefore, below we introduce two core types φ'_α and $\varphi'_{\alpha-1}$, and define φ' as $\varphi'_\alpha \vee \varphi'_{\alpha-1}$. Since the separating conjunction distributes over disjunctions, after defining these two core types, we can easily adapt the arguments of the previous case to prove that $\langle \circledast \rangle(\varphi, \psi) \Rightarrow (\varphi \circledast \top)$.

The formula φ'_α is obtained from φ by replacing, for every $x \in X$ such that $x = x_i \subseteq_{\text{LIT}} \varphi$, every literal $x \hookrightarrow_- \subseteq_{\text{LIT}} \varphi$ with $\neg x \hookrightarrow_-$, and every $x \hookrightarrow y \subseteq_{\text{LIT}} \varphi$ with $\neg x \hookrightarrow y$, where

$y \in X$. Notice that φ'_α is defined similarly to φ' (in the previous case of the proof), with the exception that we do not modify the polarity of size literals. Explicitly, φ'_α is defined as follows.

$$\begin{aligned} \varphi'_\alpha \stackrel{\text{def}}{=} & \bigwedge \{x \sim y \subseteq_{\text{LIT}} \varphi \mid \sim \in \{=, \neq\}\} \wedge \bigwedge \{x \hookrightarrow - \subseteq_{\text{LIT}} \varphi \mid x \neq x_i \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{\neg x \hookrightarrow - \subseteq_{\text{LIT}} \varphi\} \wedge \\ & \bigwedge \{\neg x \hookrightarrow - \mid x = x_i \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{x \hookrightarrow y \subseteq_{\text{LIT}} \varphi \mid x \neq x_i \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{\neg x \hookrightarrow y \subseteq_{\text{LIT}} \varphi\} \wedge \\ & \bigwedge \{\neg x \hookrightarrow y \mid x = x_i \wedge x \hookrightarrow y \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{\text{size} \geq \beta \mid \beta \in [0, \alpha - 1]\} \wedge \underline{\text{size} \geq \alpha}. \end{aligned}$$

The formula $\varphi'_{\alpha-1}$ is obtained from φ'_α by replacing $\text{size} \geq \alpha$ (highlighted in the definition of φ'_α above), by $\neg \text{size} \geq \alpha$. The two following properties are satisfied:

- D. φ'_α and $\varphi'_{\alpha-1}$ are satisfiable core types in $\text{CoreTypes}(X, \alpha)$,
- E. $(\text{ATOM}(x_i) * (\varphi'_\alpha \vee \varphi'_{\alpha-1})) \Rightarrow \varphi$ is valid.

Proof of (D). The proof is very similar to the one of the property (A). Here, we pinpoint the main differences. First of all, since both φ'_α and $\varphi'_{\alpha-1}$ are obtained from φ by changing the polarity of some of the literals in $\text{LIT}(\varphi)$, they are both in $\text{CoreTypes}(X, \alpha)$. To show that φ'_α and $\varphi'_{\alpha-1}$ are satisfiable, we rely on the fact that φ is satisfiable. Let (s, h) be a memory state satisfying φ . Since $\text{size} \geq \alpha \subseteq_{\text{LIT}} \varphi$, $\text{card}(h) \geq \alpha$. Without loss of generality, we can assume $\text{card}(h) > \alpha$. Indeed, if $\text{card}(h) = \alpha$ it is sufficient to add a memory cell (ℓ, ℓ) to h , such that ℓ does not correspond to a program variable $x \in X$. It is straightforward to check that the resulting memory state still satisfies φ . We introduce a second heap h' . Let $L = \text{dom}(h) \cap \{s(x) \mid x \in X\}$ be the set of locations in $\text{dom}(h)$ that corresponds to variables in X . Since $\text{card}(X) \leq \alpha$, $\text{card}(L) \leq \alpha$. Let $h' \subseteq h$ such that $L \subseteq \text{dom}(h')$ and $\text{card}(h') = \alpha$. Again, it is straightforward to see that (s, h') satisfies φ . Intuitively, we rely on (s, h) to show that φ'_α is satisfiable, and on (s, h') to show that $\varphi'_{\alpha-1}$ is satisfiable. As $x_i \hookrightarrow - \subseteq_{\text{LIT}} \varphi$, we have $s(x) \in \text{dom}(h)$ and $s(x) \in \text{dom}(h')$. We consider heaps h_1 and h_2 such that $h = h_1 + h_2$ and $\text{dom}(h_1) = \{s(x_i)\}$. Similarly, we consider heaps h'_1 and h'_2 such that $h' = h'_1 + h'_2$ and $\text{dom}(h'_1) = \{s(x_i)\}$. We show that $(s, h_2) \models \varphi'_\alpha$ and $(s, h'_2) \models \varphi'_{\alpha-1}$. Let us first discuss the former result. Let $L \in \text{LIT}(\varphi'_\alpha)$. If L is not of the form $\text{size} \geq \beta$ or $\neg \text{size} \geq \beta$, then $(s, h_2) \models L$ follows exactly as in the proof of (A). Otherwise,

case: $L = \text{size} \geq \beta$. By definition of h_2 , $\text{card}(h_2) = \text{card}(h) - 1 \geq \alpha$. Since $\beta \leq \alpha$ (as φ'_α is in $\text{CoreTypes}(X, \alpha)$), we conclude that $(s, h_2) \models \text{size} \geq \beta$.

case: $L = \neg \text{size} \geq \beta$. By definition of φ'_α , no literal of the form $\neg \text{size} \geq \beta$ belongs to $\text{LIT}(\varphi'_\alpha)$. Therefore, this case does not occur.

This concludes the proof of $(s, h_2) \models \varphi'_\alpha$. For the proof of $(s, h'_2) \models \varphi'_{\alpha-1}$, let us consider $L \in \text{LIT}(\varphi'_{\alpha-1})$. Again, L is not of the form $\text{size} \geq \beta$ or $\neg \text{size} \geq \alpha$, then $(s, h'_2) \models L$ follows exactly as in the proof of (A) (replacing h by h' and h_2 by h'_2). Otherwise,

case: $L = \text{size} \geq \beta$. By definition of $\varphi'_{\alpha-1}$, we have $\beta < \alpha$. By definition of h'_2 , $\text{card}(h'_2) = \text{card}(h') - 1 = \alpha - 1$. Therefore, $(s, h'_2) \models \text{size} \geq \beta$.

case: $L = \neg \text{size} \geq \beta$. By definition of $\varphi'_{\alpha-1}$, $\beta = \alpha$. Since $\text{card}(h'_2) = \alpha - 1$, we conclude that $(s, h'_2) \models \neg \text{size} \geq \beta$.

Proof of (E). The proof is very similar to the one of the property (B). We show that $(\text{ATOM}(x_i) * \varphi'_\alpha) \Rightarrow \varphi$ and $(\text{ATOM}(x_i) * \varphi'_\alpha) \Rightarrow \varphi$. Then, (E) follows as the separating

conjunction distributes over disjunction. First, let us consider $(\text{ATOM}(\mathbf{x}_i) * \varphi'_\alpha) \Rightarrow \varphi$, and a memory state (s, h) satisfying $\text{ATOM}(\mathbf{x}_i) * \varphi'_\alpha$. There are h_1 and h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \text{ATOM}(\mathbf{x}_i)$ and $(s, h_2) \models \varphi'_\alpha$. Let $L \in \text{LIT}(\varphi)$. Notice that φ does not contain negated $\text{size} \geq \beta$ literals. If L is not $\text{size} \geq \beta$, for some $\beta \in [0, \alpha]$, then $(s, h) \models L$ follows exactly as it is shown in the proof of (B). Otherwise, suppose $L = \text{size} \geq \beta$, where $\beta \in [0, \alpha]$. By definition of φ'_α , $\text{size} \geq \alpha \subseteq_{\text{LIT}} \varphi'_\alpha$. Therefore, $\text{card}(h_2) \geq \alpha$ and, from $h_2 \subseteq h$, we conclude that $(s, h) \models \text{size} \geq \beta$. So, $(s, h) \models \varphi$. Let us now consider $(\text{ATOM}(\mathbf{x}_i) * \varphi'_{\alpha-1}) \Rightarrow \varphi$ and a memory state (s, h) satisfying $\text{ATOM}(\mathbf{x}_i) * \varphi'_{\alpha-1}$. There are h_1 and h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \text{ATOM}(\mathbf{x}_i)$ and $(s, h_2) \models \varphi'_{\alpha-1}$. Let $L \in \text{LIT}(\varphi)$. Again, φ does not contain negated $\text{size} \geq \beta$ literals, and if L is not $\text{size} \geq \beta$, for some $\beta \in [0, \alpha]$, then $(s, h) \models L$ follows exactly as is shown in the proof of (B). Otherwise, suppose $L = \text{size} \geq \beta$, where $\beta \in [0, \alpha]$. By definition of $\varphi'_{\alpha-1}$, $\text{size} \geq \alpha - 1 \subseteq_{\text{LIT}} \varphi'_{\alpha-1}$. Therefore, $\text{card}(h_2) \geq \alpha - 1$. By definition of $\text{ATOM}(\mathbf{x}_i)$, $\text{card}(h_1) = 1$. From $h = h_1 + h_2$, we conclude that $\text{card}(h) \geq \alpha$ and thus $(s, h) \models \text{size} \geq \beta$. Therefore, $(s, h) \models \varphi$.

As in the previous case of the proof, (D) allows us to consider the formulae $\langle \circledast \rangle(\varphi'_\alpha, \psi)$ and $\langle \circledast \rangle(\varphi'_{\alpha-1}, \psi)$ and show that

F. $\langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(\mathbf{x}_i) \Rightarrow \langle \circledast \rangle(\varphi'_\alpha, \psi) \vee \langle \circledast \rangle(\varphi'_{\alpha-1}, \psi)$ is valid.

Proof of (F). We recall that $\langle \circledast \rangle(\varphi, \psi)$ is satisfiable. In particular, from its definition together with $\text{size} \geq \alpha \subseteq_{\text{LIT}} \varphi$, this implies that $\text{size} \geq \alpha \subseteq_{\text{LIT}} \psi$, as otherwise we would have $\neg \text{size} \geq 0 \subseteq_{\text{LIT}} \langle \circledast \rangle(\varphi, \psi)$. So, as ψ is a satisfiable core type in $\text{CoreTypes}(\mathbf{x}, \alpha)$, for all $\beta \in [0, \alpha]$, $\text{size} \geq \beta \subseteq_{\text{LIT}} \psi$. Alternatively, ψ does not contain $\neg \text{size} \geq \beta$ literals. We look at the definitions of $\langle \circledast \rangle(\varphi'_\alpha, \psi)$ and $\langle \circledast \rangle(\varphi'_{\alpha-1}, \psi)$.

- a. Since for all $\beta \in [0, \alpha]$, $\text{size} \geq \beta \subseteq_{\text{LIT}} \varphi'_\alpha$ and $\text{size} \geq \beta \subseteq_{\text{LIT}} \psi$, we derive that $\langle \circledast \rangle(\varphi'_\alpha, \psi)$ does not contain $\text{size} \geq \beta$ nor $\neg \text{size} \geq \beta$ literals (for all $\beta \in [0, \alpha]$). This holds directly by definition of $\langle \circledast \rangle(\varphi'_\alpha, \psi)$, which can be retrieved by substituting φ' by φ'_α in Figure 6.10.
- b. Analogously, we know that $\neg \text{size} \geq \alpha \subseteq_{\text{LIT}} \varphi'_{\alpha-1}$ whereas for every $\beta \in [0, \alpha - 1]$, $\text{size} \geq \beta \subseteq_{\text{LIT}} \varphi'_{\alpha-1}$, and therefore among all the literals $\text{size} \geq \beta$ or $\neg \text{size} \geq \beta$ ($\beta \in [0, \alpha]$), $\langle \circledast \rangle(\varphi'_{\alpha-1}, \psi)$ only contains $\text{size} \geq 1$ (occurring positively).

By definition and with the sole exception of the polarity of the formula $\text{size} \geq \alpha$ (occurring positively in φ'_α and negatively in $\varphi'_{\alpha-1}$), the two core types $\varphi'_{\alpha-1}$ and φ'_α are equal. Directly by definition of $\langle \circledast \rangle(\varphi'_\alpha, \psi)$ and $\langle \circledast \rangle(\varphi'_{\alpha-1}, \psi)$, together with (a) and (b), this implies that $\langle \circledast \rangle(\varphi'_{\alpha-1}, \psi)$ is syntactically equal to $\langle \circledast \rangle(\varphi'_\alpha, \psi) \wedge \text{size} \geq 1$ (up to commutativity and associativity of conjunction). This means that the formula $\langle \circledast \rangle(\varphi'_{\alpha-1}, \psi) \Rightarrow \langle \circledast \rangle(\varphi'_\alpha, \psi)$ is valid, and suggests us that, in order to show (F), we can simply establish that $\langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(\mathbf{x}_i) \Rightarrow \langle \circledast \rangle(\varphi'_\alpha, \psi)$ is valid. As we already stated, φ'_α is defined as φ' (in the previous step of the proof), with the exception that we do not modify the polarity of $\text{size} \geq \beta$ literals. Because of this, we can rely on the proof of (C). Briefly, we consider a memory state (s, h) satisfying $\langle \circledast \rangle(\varphi, \psi) * \text{ATOM}(\mathbf{x}_i)$. There are h_1 and h_2 such that $h = h_1 + h_2$, $(s, h_1) \models \langle \circledast \rangle(\varphi, \psi)$ and $(s, h_2) \models \text{ATOM}(\mathbf{x}_i)$. Let $L \in \text{LIT}(\langle \circledast \rangle(\varphi'_\alpha, \psi))$. By (a), L is neither of the form $\text{size} \geq \beta$ nor of the form $\neg \text{size} \geq \beta$. Therefore, $(s, h) \models L$ follows exactly as shown in the proof of (C).

We are now ready to prove that $\langle \circledast \rangle(\varphi, \psi) \Rightarrow (\varphi \circledast \top)$. By Theorem 6.16, the tautologies in (D) and (F) are derivable in $\mathcal{H}_C(*, \circledast)$. Moreover, since $\neg \mathbf{x}_i \hookrightarrow \neg \subseteq_{\text{LIT}} \{\varphi'_\alpha; \varphi'_{\alpha-1}\}$ and,

for every $y \in X$, $\neg y \hookrightarrow \perp \subseteq_{\text{LIT}} \varphi$ implies $\neg y \hookrightarrow \perp \subseteq_{\text{LIT}} \{\varphi'_\alpha; \varphi'_{\alpha-1}\}$, we rely on the induction hypothesis to derive

$$\vdash_{\mathcal{H}_C(*, \neg)} \langle \neg \rangle(\varphi'_\alpha, \psi) \Rightarrow (\varphi'_\alpha \dashv \top), \quad \vdash_{\mathcal{H}_C(*, \neg)} \langle \neg \rangle(\varphi'_{\alpha-1}, \psi) \Rightarrow (\varphi'_{\alpha-1} \dashv \top).$$

We are ready to derive $\langle \neg \rangle(\varphi, \psi) \Rightarrow (\varphi \dashv \top)$, concluding the proof:

1	$\langle \neg \rangle(\varphi'_\alpha, \psi) \Rightarrow (\varphi'_\alpha \dashv \top)$	Induction hypothesis
2	$\langle \neg \rangle(\varphi'_{\alpha-1}, \psi) \Rightarrow (\varphi'_{\alpha-1} \dashv \top)$	Induction hypothesis
3	$(\text{ATOM}(x_i) * (\varphi'_\alpha \vee \varphi'_{\alpha-1})) \Rightarrow \varphi$	(E), Theorem 6.16
4	$((\langle \neg \rangle(\varphi, \psi) * \text{ATOM}(x_i)) \Rightarrow \langle \neg \rangle(\varphi'_\alpha, \psi) \vee \langle \neg \rangle(\varphi'_{\alpha-1}, \psi))$	(F), Theorem 6.16
5	$\langle \neg \rangle(\varphi'_\alpha, \psi) \vee \langle \neg \rangle(\varphi'_{\alpha-1}, \psi) \Rightarrow (\varphi'_\alpha \dashv \top) \vee (\varphi'_{\alpha-1} \dashv \top)$	PC, 1, 2
6	$(\varphi'_\alpha \dashv \top) \vee (\varphi'_{\alpha-1} \dashv \top) \Rightarrow ((\varphi'_\alpha \vee \varphi'_{\alpha-1}) \dashv \top)$	(I _{6.19.6} [*])
7	$((\langle \neg \rangle(\varphi, \psi) * \text{ATOM}(x_i)) \Rightarrow ((\varphi'_\alpha \vee \varphi'_{\alpha-1}) \dashv \top))$	($\Rightarrow \text{TR}$), 4, 5, 6
8	$\langle \neg \rangle(\varphi, \psi) \Rightarrow (\text{ATOM}(x_i) \dashv ((\langle \neg \rangle(\varphi, \psi) * \text{ATOM}(x_i))))$	(‡)
9	$(\text{ATOM}(x_i) \dashv ((\langle \neg \rangle(\varphi, \psi) * \text{ATOM}(x_i)))) \Rightarrow$ $(\text{ATOM}(x_i) \dashv ((\varphi'_\alpha \vee \varphi'_{\alpha-1}) \dashv \top))$	(I _{6.19.5} [*]), 4
10	$(\text{ATOM}(x_i) \dashv ((\varphi'_\alpha \vee \varphi'_{\alpha-1}) \dashv \top)) \Rightarrow$ $((\text{ATOM}(x_i) * (\varphi'_\alpha \vee \varphi'_{\alpha-1})) \dashv \top)$	(I _{6.19.8} [*])
11	$((\text{ATOM}(x_i) * (\varphi'_\alpha \vee \varphi'_{\alpha-1})) \dashv \top) \Rightarrow (\varphi \dashv \top)$	(I _{6.19.4} [*]), 3
12	$\langle \neg \rangle(\varphi, \psi) \Rightarrow (\varphi \dashv \top)$	($\Rightarrow \text{TR}$), 8, 9, 10, 11 \square

Lemma 6.18 for core types can be extended to arbitrary Boolean combinations of core formulae, as we show that the distributivity of \dashv over disjunctions is provable in $\mathcal{H}_C(*, \neg)$. As a consequence of this development, we achieve the main result of the chapter.

Theorem 6.22. $\mathcal{H}_C(*, \neg)$ is an adequate proof system for $\text{SL}(*, \neg)$.

Soundness of the proof system $\mathcal{H}_C(*, \neg)$ has been already in Lemma 6.5. The structure of the completeness proof is very similar to the proof of Theorem 6.16 except that we have to be able to handle the separating implication. In order to be self-contained, we reproduce some of the arguments in Theorem 6.16, albeit adapted to $\mathcal{H}_C(*, \neg)$.

First of all, we show that the substitution of equivalent formulae in holds true in $\mathcal{H}_C(*, \neg)$, that is the following rule is admissible:

$$(S_{\neg}) \quad \frac{\psi \Leftrightarrow \chi}{\varphi[\psi]_\rho \Leftrightarrow \varphi[\chi]_\rho}$$

where φ, ψ, χ are in $\text{SL}(*, \neg)$, and $\varphi[\psi]_\rho$ refers to the formula φ in which the subformula at the occurrence ρ is replaced by ψ (see Definition 1.4).

Proof of (S_{neg}). The admissibility of (S_{neg}) is proved as for the admissibility of (S_{*}), i.e. with a standard structural induction on φ . The cases where φ is either an atomic formula or a formula

of the form $\neg\psi'$, $\varphi' \Rightarrow \varphi''$ or $\varphi' * \varphi''$ are exactly as in the proof of (S_*) . For the only remaining case, where $\varphi = \varphi' * \varphi''$, the proof carries out as shown below. Let φ , ψ and χ being three formulae in $\text{SL}(*, \neg)$ and let ρ be a position in φ . Suppose $\vdash_{\mathcal{H}_C(*, \neg)} \psi \Leftrightarrow \chi$.

induction step: $\varphi = \varphi' * \varphi''$. If $\rho = \epsilon$, then the result follows as in the base case where φ is an atomic formula (see proof of (S_*) , page 314). Otherwise, either $\rho = 1\rho'$ or $\rho = 2\rho''$, for some positions ρ' and ρ'' of φ' and φ'' , respectively. We split the proof depending on whether $\rho = 1\rho'$ or $\rho = 2\rho''$.

case: $\rho = 1\rho'$. By induction hypothesis, $\vdash_{\mathcal{H}_C(*)} \varphi'[\psi]_{\rho'} \Leftrightarrow \varphi'[\chi]_{\rho'}$. As a direct consequence of the admissibility of the rule $(I_{6.19.4}^*)$ from Lemma 6.19, together with propositional reasoning, the following rule is admissible:

$$\frac{\varphi_1 \Leftrightarrow \varphi_2}{(\varphi_1 * \psi_3) \Leftrightarrow (\varphi_2 * \psi_3)}$$

Therefore, $\vdash_{\mathcal{H}_C(*)} ((\varphi'[\psi]_{\rho'}) * \varphi'') \Leftrightarrow ((\varphi'[\chi]_{\rho'}) * \varphi'')$. By definition, $\varphi[\psi]_{\rho} = (\varphi'[\psi]_{\rho'}) * \varphi''$ and $\varphi[\chi]_{\rho} = (\varphi'[\chi]_{\rho'}) * \varphi''$, concluding the proof.

case: $\rho = 2\rho''$. Similarly to the other case, $\vdash_{\mathcal{H}_C(*)} \varphi''[\psi]_{\rho''} \Leftrightarrow \varphi''[\chi]_{\rho''}$, by induction hypothesis. Thanks to the rule $(I_{6.19.5}^*)$ from Lemma 6.19, the following rule is admissible:

$$\frac{\varphi_3 \Leftrightarrow \varphi_2}{(\varphi_1 * \varphi_3) \Leftrightarrow (\varphi_1 * \varphi_2)}$$

Thus, $\vdash_{\mathcal{H}_C(*)} (\varphi' * (\varphi''[\psi]_{\rho''})) \Leftrightarrow (\varphi' * (\varphi''[\chi]_{\rho''}))$. By definition, $\varphi[\psi]_{\rho} = \varphi' * (\varphi''[\psi]_{\rho''})$ and $\varphi[\chi]_{\rho} = \varphi' * (\varphi''[\chi]_{\rho''})$, concluding the proof. \square

Proof of Theorem 6.22 (completeness). We need to show that for every formula φ in $\text{SL}(*, \neg)$, there is a Boolean combination of core formulae ψ such that $\vdash_{\mathcal{H}_C(*, \neg)} \varphi \Leftrightarrow \psi$. This is enough to conclude the proof. Indeed, when φ is valid for $\text{SL}(*, \neg)$, by soundness of $\mathcal{H}_C(*, \neg)$, we obtain that ψ is valid too. Therefore, $\vdash_{\mathcal{H}_C(*, \neg)} \psi$ as \mathcal{H}_C is a subsystem of $\mathcal{H}_C(*, \neg)$ and it is complete for Boolean combinations of core formulae (Theorem 6.9). By propositional reasoning, we get that $\vdash_{\mathcal{H}_C(*, \neg)} \varphi$.

In order to show that every formula φ has a provably equivalent Boolean combination of core formulae, we heavily rely on the $*$ - and \neg -simulation properties given by Corollary 6.15 and Lemma 6.18. For convenience, we consider φ such that every \neg is either used inside the definition of a core formula of the form $x \hookrightarrow \perp$ or it is involved in the definition of the subtraction $\neg\otimes$. This can be done without loss of generality. Indeed, it is easy to show that $(\varphi \neg \psi) \Leftrightarrow \neg(\varphi \otimes \neg\psi)$ is derivable in $\mathcal{H}_C(*, \neg)$.

1	$\psi \Leftrightarrow \neg\neg\psi$	PC
2	$(\varphi \neg \psi) \Leftrightarrow (\varphi \neg \neg\neg\psi)$	(S_*) , 1
3	$(\varphi \neg \neg\neg\psi) \Leftrightarrow \neg\neg(\varphi \neg \neg\psi)$	PC
4	$(\varphi \neg \psi) \Leftrightarrow \neg(\varphi \otimes \neg\psi)$	$(\Rightarrow \text{TR})$, 2, 3, def. of $\neg\otimes$

Therefore, thanks to (S_*) , we can replace all occurrences of $\varphi' * \varphi''$ in φ , that are not involved in the definition of a core formula of the form $x \hookrightarrow \perp$, by $\neg(\varphi' \otimes \neg\varphi'')$, leading to a formula of the wanted shape.

Now, similarly to Theorem 6.16, the proof is by simple induction on the number of occur-

rences of $*$ or $\text{-}\otimes$ in φ that are not involved in the definition of some core formula of the form $\text{size} \geq \beta$ or $x \hookrightarrow _$.

base case: φ without occurrences of $*$ or $\text{-}\otimes$ (excluding those appearing in core formulae). This case is exactly as in Theorem 6.16. The formula φ is a Boolean combination of core formulae plus the atomic formula emp . Since $\vdash_{\mathcal{H}_C(*, \text{-}\otimes)} \text{emp} \Leftrightarrow \neg \text{size} \geq 1$ holds by propositional reasoning and definition of $\text{size} \geq 1$, by $(S_{\text{-}\otimes})$ we can replace every subformula emp by $\neg \text{size} \geq 1$, leading to a Boolean combination of core formulae ψ such that $\vdash_{\mathcal{H}_C(*, \text{-}\otimes)} \varphi \Leftrightarrow \psi$.

induction step: φ with $n \geq 1$ occurrences of $*$ or $\text{-}\otimes$ (excluding those appearing in core formulae). Below, we let X be the set of variables appearing in φ . Let $\varphi_1 \otimes \varphi_2$ be a subformula of φ , say at position ρ , such that $\otimes \in \{*, \text{-}\otimes\}$, φ_1 in $\text{Bool}(\text{Core}(X_1, \alpha_1))$ and φ_2 in $\text{Bool}(\text{Core}(X_2, \alpha_2))$. Let $\alpha = \max(\alpha_1, \alpha_2)$. As X_1 and X_2 are subsets of X , by definition, φ_1 and φ_2 belong to $\text{Bool}(\text{Core}(X, \alpha))$. By Lemma 6.10 and since $\mathcal{H}_C(*)$ includes \mathcal{H}_C , there are two formulae of the form $\varphi_1^1 \vee \dots \vee \varphi_1^{n_1}$ and $\varphi_2^1 \vee \dots \vee \varphi_2^{n_2}$ such that $\vdash_{\mathcal{H}_C(*)} \varphi_i \Leftrightarrow \varphi_i^1 \vee \dots \vee \varphi_i^{n_i}$ for $i \in \{1, 2\}$ and moreover, all the φ_i^j 's are core types in $\text{CoreTypes}(X, \max(\text{card}(X), \alpha))$. By (S_*) ,

$$\vdash_{\mathcal{H}_C(*)} \varphi_1 * \varphi_2 \Leftrightarrow (\varphi_1^1 \vee \dots \vee \varphi_1^{n_1}) \otimes (\varphi_2^1 \vee \dots \vee \varphi_2^{n_2}).$$

We now derive the following double implication, by cases on \otimes :

$$\vdash_{\mathcal{H}_C(*)} \varphi_1 \otimes \varphi_2 \Leftrightarrow \bigvee_{j_1 \in [1, n_1], j_2 \in [1, n_2]} \varphi_1^{j_1} \otimes \varphi_2^{j_2}. \quad (\dagger)$$

case: $\otimes = *$. As in Theorem 6.16, (\dagger) follows by propositional reasoning, together with the axiom (I_3^*) for distributivity and the theorem $(\varphi * \chi) \vee (\psi * \chi) \Rightarrow (\varphi \vee \psi) * \chi$ derived in page 296.

case: $\otimes = \text{-}\otimes$. In this case, (\dagger) follows directly from $(I_{6.19.6}^*)$ and $(I_{6.19.7}^*)$ (which tell us that the subtraction $\text{-}\otimes$ distributes over \vee), and propositional reasoning.

We now rely on Corollary 6.15 and Lemma 6.18 (depending on whether \otimes is $*$ or $\text{-}\otimes$). We conclude that for all $j_1 \in [1, n_1]$ and $j_2 \in [1, n_2]$, there is a conjunction of core formulae ψ^{j_1, j_2} such that $\vdash_{\mathcal{H}_C(*)} \varphi_1^{j_1} \otimes \varphi_2^{j_2} \Leftrightarrow \psi^{j_1, j_2}$. By propositional reasoning, we conclude that $\vdash_{\mathcal{H}_C(*)} \varphi_1 \otimes \varphi_2 \Leftrightarrow \bigvee_{j_1 \in [1, n_1], j_2 \in [1, n_2]} \psi^{j_1, j_2}$. Consequently (thanks to the rule $(S_{\text{-}\otimes})$), we obtain

$$\vdash_{\mathcal{H}_C(*)} \varphi \Leftrightarrow \varphi[\bigvee_{j_1 \in [1, n_1], j_2 \in [1, n_2]} \psi^{j_1, j_2}]_\rho.$$

The right-hand side formula of the double implication above has $n - 1$ occurrences of $*$ or $\text{-}\otimes$ that are not involved in the definition of core formulae. The induction hypothesis applies: there is a Boolean combination of core formulae ψ such that $\vdash_{\mathcal{H}_C(*)} \varphi[\bigvee_{j_1 \in [1, n_1], j_2 \in [1, n_2]} \psi^{j_1, j_2}]_\rho \Leftrightarrow \psi$. By propositional reasoning, $\vdash_{\mathcal{H}_C(*)} \varphi \Leftrightarrow \psi$. \square

Axiomatising a Modal Logic Featuring Ambient-like Composition

Contents

7.1	A Taste of Ambient Logic	341
7.2	The Modal Logic $\text{ML}(\mathbb{I})$	342
7.2.1	Kripke-style finite forests.	342
7.2.2	$\text{ML}(\mathbb{I})$: Syntax and Semantics.	344
7.2.3	$\text{ML}(\mathbb{I})$ as a Graded Modal Logic.	344
7.3	Towards an Hilbert-style proof system for $\text{ML}(\mathbb{I})$	346
7.4	Graded Modalities as Core Formulae	348
7.5	Syntactical Elimination of the Composition Operator	352

In this chapter

We move away from separation logic and introduce another instantiation of BBI, called $\text{ML}(\mathbb{I})$, which extends the standard modal logic ML with the composition operator from *ambient logic*, a logic introduced by L. Cardelli and A. D. Gordon to verify properties of distributed systems. After introducing ambient logic and $\text{ML}(\mathbb{I})$, we design an Hilbert-style proof system for $\text{ML}(\mathbb{I})$, again by relying on the core formulae technique used for the axiomatisation of $\text{SL}(*, \neg*)$. In the case of $\text{ML}(\mathbb{I})$, the core formulae are given by *graded modal logic*, a well-known extension of ML . The aim of the chapter is twofold. On one side, we want to show another example of very natural Hilbert-style proof system designed with the help of core formulae. On the other side, the Hilbert-style proof system for $\text{ML}(\mathbb{I})$ reveals interesting connections between separation logic and ambient logic, which we later analyse in the last part of the thesis (Chapters 8 and 9).

Here is a roadmap of the chapter, which follows quite closely Chapter 6.

Section 7.1. To better contextualise the chapter, we start with a short introduction on the calculus of Mobile Ambients [40] and ambient logic. Broadly speaking, ambient logic is interpreted on a class of syntactical trees called *information trees*, and its composition operator $\varphi \parallel \psi$ splits the tree into two components, similarly to the operator $*$ from separation logic.

Section 7.2. We introduce $\text{ML}(\mathbb{I})$. This modal logic is interpreted on a class of Kripke structures whose underlying accessibility relation is a finite forest. As already stated, $\text{ML}(\mathbb{I})$ extends the language of the standard modal logic ML with the composition from ambient logic, opportunely redefined on Kripke-style finite forests. For instance, the $\text{ML}(\mathbb{I})$ formula $\Diamond\varphi \parallel \Diamond\varphi$ states that the current world has at least two children satisfying the formula φ . We notice that $\Diamond\varphi \parallel \Diamond\varphi$ corresponds to the formula $\Diamond_{\geq 2}\varphi$ from graded modal logic (GML), paving a way of axiomatising $\text{ML}(\mathbb{I})$ by relying on GML as a family of core formulae.

Section 7.3. We define the proof system $\mathcal{H}_{\text{GML}}(\mathbb{I})$ that shall be proven adequate for $\text{ML}(\mathbb{I})$. This proof system extends the axiomatisation of GML introduced in [49] with axioms and rules that are able to transform every formula of $\text{ML}(\mathbb{I})$ into an equivalent formula from GML . Since the graded modalities $\Diamond_{\geq k}\varphi$ can be seen as abbreviations of formulae from $\text{ML}(\mathbb{I})$, the proof system $\mathcal{H}_{\text{GML}}(\mathbb{I})$ is internal.

Section 7.4. We introduce technical notions needed to use GML as a family of core formulae.

Section 7.5. We show that $\mathcal{H}_{\text{GML}}(\mathbb{I})$ is sound and complete for $\text{ML}(\mathbb{I})$. As usual, soundness is proved with a simple semantical analysis on the axioms and rules of the proof system. To establish the completeness of $\mathcal{H}_{\text{GML}}(\mathbb{I})$, we rely on the following result.

Lemma 7.14. Let Φ be a set of disjoint formulae in GML , $k_1, k_2 \in \mathbb{N}$, and $P_1, P_2 \subseteq_{\text{fin}} \text{AP}$. For every two satisfiable formulae φ in $\text{Conj}(\text{Core}(\Phi, k_1, P_1))$ and ψ in $\text{Conj}(\text{Core}(\Phi, k_2, P_2))$. Then,

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbb{I})} \varphi \parallel \psi \Leftrightarrow \langle \mathbb{I} \rangle(\varphi, \psi).$$

Here, φ , ψ and $\langle \mathbb{I} \rangle(\varphi, \psi)$ are formulae from GML . This lemma implies that, in $\mathcal{H}_{\text{GML}}(\mathbb{I})$, every formula of $\text{ML}(\mathbb{I})$ can be shown equivalent to a formula of GML . Thus, completeness of $\mathcal{H}_{\text{GML}}(\mathbb{I})$ stems directly from the fact that the proof system extends a complete axiomatisation for GML .

7.1 A TASTE OF AMBIENT LOGIC

In 1998, L. Cardelli and A. D. Gordon introduced the calculus of Mobile Ambients (MA) [40], a.k.a. *ambient calculus*: a process calculus focused on spatial configurations of distributed agents, where computations carry out by means of spatial reconfigurations. The fundamental entity of MA is the *ambient*, which can be informally described as a local object in which computation might occur. For instance, the term of MA

$$m[0] \mid n[\text{in}(m).0]$$

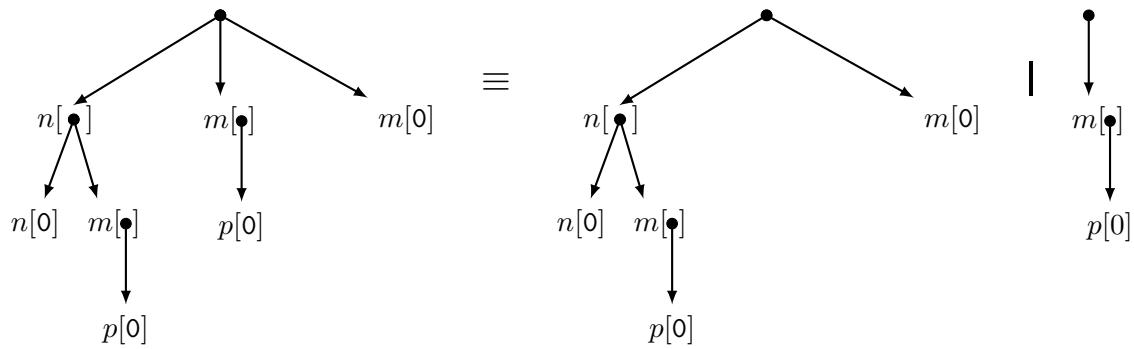
features two ambients $m[0]$ and $n[\text{in}(m).0]$, which belongs to the same space. The notion of *space* is defined through the parallel composition $P \mid Q$ which essentially operates as a union on multisets (elements of the sets being ambients in the spaces P and Q), together with the empty space 0. So, the ambient $m[0]$ stands for an ambient named m that does not contain anything. Instead, the ambient $n[\text{in}(m).0]$ contains an *action* $\text{in}(m).0$ which allows n to enter in the space of the ambient m . Formally, this interaction is achieved via the following rewriting rule:

$$m[P] \mid n[\text{in}(m).Q \mid R] \rightarrow m[n[Q \mid R] \mid P],$$

where P , Q and R are metavariables. Notice that the term $P \mid 0$ describes a space obtained by composing the space P with the empty space. Because of this, $P \mid 0$ is equivalent to P (written $P \mid 0 \equiv P$), which implies that the ambient $n[\text{in}(m).0]$ is equivalent to $n[\text{in}(m).0 \mid 0]$. This allows us to apply the rule above on $m[0] \mid n[\text{in}(m).0]$, deducing $m[n[0 \mid 0] \mid 0]$. Again from the equivalence $P \mid 0 \equiv P$, the latter term is equivalent to $m[n[0]]$. Fundamentally, the interaction between ambients changes the shape of the space: in our example, from two ambients m and n belonging to the same space, we have obtained an ambient m that contains the ambient n .

To verify the complex interactions between ambients, in [39] L. Cardelli and A. D. Gordon introduce a modal logic that combines spatial and temporal operators, called *Ambient Logic* (AL). While the temporal operators of AL reason on the sequence of terms obtained via rewriting rules, the spatial operators analyse the structure of the ambients. Interestingly, even though independently formalised, the fragment of AL without temporal operators, a.k.a. *Static Ambient Logic* (SAL) [103], is an instantiation of BBI (Section 2.3.3). In particular, AL and SAL feature a composition operator $\varphi \mid \psi$ which, with respect to the terms of MA, corresponds to the parallel composition $P \mid Q$ between spaces P and Q . As required by the notion of non-deterministic monoid of BBI, the parallel composition \mid is associative and commutative, with 0 being its identity element. This makes the models of SAL, called *information trees*, finite unordered trees with labelled nodes that represent the names of the ambients. Instead of giving the formal definition of information trees, let us look at the ones depicted in Figure 7.1. In the tree on the left (i.e. T_3), the root consists of a space made of three ambients, which are depicted as children of the root. In this representation, the parallel composition $T_1 \mid T_2$ of the two information trees T_1 and T_2 on the right merge the spaces represented by the roots of T_1 and T_2 , leading to T_3 .

The connections between ambient logic and BBI make SAL and separation logic quite close. To the best of our knowledge, the first work that analyses together these two logics is [104]: the paper that led to the development of the core formulae technique for separation logic (see Chapter 5). In particular, in [104], E. Lozes uses simulation arguments that are similar to the $*$ -simulation property of separation logic (Lemma 5.6) in order to show that several operators of SAL can be removed from the logic without modifying its expressive power. Following [104] and in view of the Hilbert-style proof system for SL($*$, \rightarrow) defined in Chapter 6, it is natural to

Figure 7.1: Three information trees T_3, T_1, T_2 (from left to right), such that $T_3 \equiv T_1 | T_2$.

ask ourselves if we can derive axiomatisations of SAL and similar ambient logics. At the same time, we would like to find a common ground between ambient logic and separation logic, as it could allow us to deepen our understanding on the connections between these two formalisms.

To keep things simple, in this chapter we introduce the modal logic $\text{ML}(|)$ that extends standard modal logic ML (a.k.a. K) [15] with the composition operator $|$ from SAL. Instead of information trees, we interpret $\text{ML}(|)$ on a class of Kripke structures that represent finite forests. On one hand, these Kripke-style finite forests do not diverge too much from information trees, allowing us to transfer results from $\text{ML}(|)$ to SAL, as we will see in Chapter 8. On the other hand, relying on the standard framework of modal logic allows us to draw new connections between ambient logics and (modal) separation logics, as we show in Chapter 9. Thus, the main purpose of this chapter is to begin the study on $\text{ML}(|)$, which progresses throughout the remaining part of the thesis. We present an Hilbert-style proof system of $\text{ML}(|)$, which is designed thanks to the core formulae technique. The axioms of the system already reveal interesting connections between $\text{ML}(|)$ and $\text{SL}(*, *)$, which motivates us to pursue the comparison between separation logic and ambient logic carried out in Chapters 8 and 9.

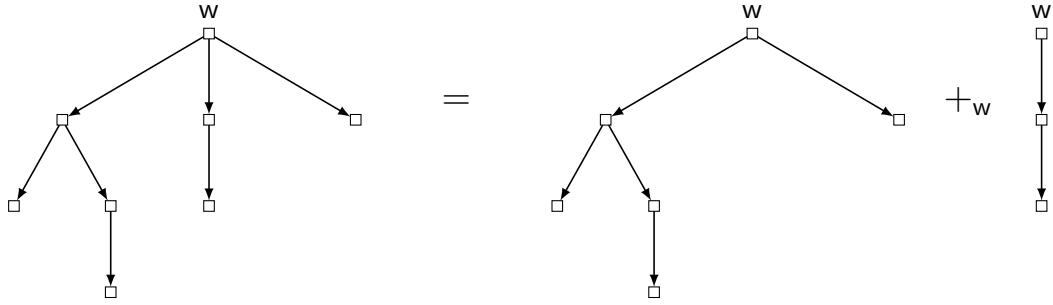
7.2 THE MODAL LOGIC $\text{ML}(|)$

In this section, we introduce the modal logic $\text{ML}(|)$ that extends the standard modal logic ML with the composition operator from ambient logic. The main contribution of this chapter is the design of an Hilbert-style proof system for $\text{ML}(|)$. We refer the reader to Section 6.1.2 for a quick introduction on Hilbert-style proof systems.

7.2.1 Kripke-style finite forests.

As already stated, in order to keep $\text{ML}(|)$ close to ambient logics while taking advantage of the framework of modal logics, we consider the class of Kripke-like finite forests defined below. As done in previous chapters, we write AP for a countably infinite set of *atomic propositions*.

Definition 7.1 (Kripke-style finite forest). A (*Kripke-style*) *finite forest* $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a triple where \mathcal{W} is a non-empty finite set of *worlds* (i.e. a *universe*), $\mathcal{V} : \text{AP} \rightarrow 2^{\mathcal{W}}$ is a *valuation* and $R \subseteq \mathcal{W} \times \mathcal{W}$ is a finite binary relation whose inverse R^{-1} is functional and acyclic.

Figure 7.2: Three finite forests $\mathcal{K}_3, \mathcal{K}_1, \mathcal{K}_2$ (from left to right), such that $\mathcal{K}_3 = \mathcal{K}_1 +_w \mathcal{K}_2$.

We define $R(w) \stackrel{\text{def}}{=} \{w' \in \mathcal{W} \mid (w, w') \in R\}$. Worlds in $R(w)$ are understood as *children* of w . As usual, we write R^δ for the δ th composition of R , R^+ for its transitive closure (a.k.a. Kleene plus) and R^* for its reflexive and transitive closure (a.k.a. Kleene star).

To mimic the composition from ambient logic, we introduce a suitable union of finite forests.

Definition 7.2 ($+_w$: the Ambient-like union). Let w be a world from a finite set \mathcal{W} . We introduce the ternary relation $+_w$ on Kripke-style finite forests, that given $\mathcal{K}_i = (\mathcal{W}_i, R_i, \mathcal{V}_i)$ ($i \in \{1, 2, 3\}$) is characterised as follows:

$$(\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3) \in +_w \text{ iff } \mathcal{W}_1 = \mathcal{W}_2 = \mathcal{W}_3 = \mathcal{W}, \mathcal{V}_1 = \mathcal{V}_2 = \mathcal{V}_3, R_1 \cap R_2 = \emptyset, R_1 \cup R_2 = R_3, \text{ and for all } i \in \{1, 2\} \text{ and } w' \in R_i(w), R_i^+(w') = R_3^+(w').$$

Thanks to the constraints $R_1 \cap R_2 = \emptyset$ and $R_1 \cup R_2 = R_3$, the relation $+_w$ is a partial function on its third component; that is, given \mathcal{K}_1 and \mathcal{K}_2 , there is at most one \mathcal{K}_3 such that $(\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3) \in +_w$. Because of this, we often see $+_w$ as a binary partial function, and write $\mathcal{K}_1 +_w \mathcal{K}_2$ to denote the only Kripke-style finite forest \mathcal{K}_3 such that $(\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3) \in +_w$, if any. In this way, the expression $\mathcal{K}_3 = \mathcal{K}_1 +_w \mathcal{K}_2$ is equivalent to $(\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3) \in +_w$. We say that \mathcal{K}_1 a *w-subforest* (or more simply, a *subforest*) of \mathcal{K}_3 , written $\mathcal{K}_1 \subseteq_w \mathcal{K}_3$, whenever there is a finite forest \mathcal{K}_2 such that $\mathcal{K}_3 = \mathcal{K}_1 +_w \mathcal{K}_2$. Figure 7.2 shows possible instances for $\mathcal{K}_1, \mathcal{K}_2$ and \mathcal{K}_3 such that $\mathcal{K}_3 = \mathcal{K}_1 +_w \mathcal{K}_2$. Notice the similarities with the information trees of Figure 7.1. Indeed, exactly as in the case of the parallel composition of MA, the union $+_w$ preserves the subtrees rooted at a child of w , which must occur in either \mathcal{K}_1 or \mathcal{K}_2 . Intuitively, this means that every possible split of the subtree rooted at w is completely determined by how the children of w are partitioned in the two structures \mathcal{K}_1 and \mathcal{K}_2 .

The modal logic $\text{ML}(\mathbf{I})$ we are about to define instantiate the framework of BBI introduced in Section 2.3.3. In particular, when fixing the universe and valuation, the class of Kripke-style finite forests forms a non-deterministic monoid with the union $+_w$ (seen as a partial function).

Proposition 7.3. Let $\mathbb{K}_{(\mathcal{W}, \mathcal{V})}$ be the class of Kripke-style finite forests with universe \mathcal{W} and valuation \mathcal{V} . Let $w \in \mathcal{W}$ and let $\epsilon = (\mathcal{W}, \emptyset, \mathcal{V})$. $(\mathbb{K}_{(\mathcal{W}, \mathcal{V})}, +_w, \epsilon)$ is a non-deterministic monoid.

We recall that Proposition 7.3 above (whose proof is straightforward) states that, whenever defined, $+_w$ is an associative and commutative operator, with ϵ being its identity element.

-
- $(\mathcal{K}, w) \models p$ iff $w \in \mathcal{V}(p)$,
 $(\mathcal{K}, w) \models \Diamond\varphi$ iff there is $w' \in \mathcal{W}$ such that $w' \in R(w)$ and $(\mathcal{K}, w') \models \varphi$,
 $(\mathcal{K}, w) \models \varphi \mid \psi$ iff there are \mathcal{K}_1 and \mathcal{K}_2 s.t. $\mathcal{K}_1 +_w \mathcal{K}_2 = \mathcal{K}$, $(\mathcal{K}_1, w) \models \varphi$ and $(\mathcal{K}_2, w) \models \psi$.

Figure 7.3: Satisfaction relation for $\text{ML}(\mid)$, with respect to (\mathcal{K}, w) where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

7.2.2 $\text{ML}(\mid)$: Syntax and Semantics.

The logic $\text{ML}(\mid)$ enriches the standard modal logic ML with the *composition operator* \mid that allows for submodel reasoning via the union $+_w$. The formulae of $\text{ML}(\mid)$ are built from the grammar below (where $p \in \text{AP}$):

$$\begin{array}{lll}
 \pi := & \top & \text{(true)} \\
 & | & \\
 & p & \text{(propositional symbol)} \\
 & | & \\
 & \varphi & \text{(atomic formulae)} \\
 & | & \\
 & \varphi \Rightarrow \varphi & \text{(Boolean connectives)} \\
 & | & \\
 & \Diamond\varphi & \text{(modality of possibility)} \\
 & | & \\
 & \varphi \mid \varphi & \text{(composition)}
 \end{array}$$

Given a *pointed forest* (\mathcal{K}, w) , that is a Kripke-style finite forest $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ together with a world $w \in \mathcal{W}$, the satisfaction relation \models for formulae in $\text{ML}(\mid)$ is given in Figure 7.3, omitting the standard clauses for \top and the Boolean connectives \Rightarrow and \neg .

As in the previous chapter, the formulae $\varphi \wedge \psi$ and $\varphi \vee \psi$ are syntactical abbreviations for $\neg(\varphi \Rightarrow \neg\psi)$ and $\neg\varphi \Rightarrow \psi$, respectively. Similarly, \perp and $\varphi \Leftrightarrow \psi$ stand for $\neg\top$ and $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$, respectively. The *modality of necessity* $\Box\varphi$ is defined as $\neg\Diamond\neg\varphi$. It is the dual of the modality of possibility \Diamond , and its semantics is as follows:

$$(\mathcal{K}, w) \models \Box\varphi \text{ if and only if for every } w' \in R(w), (\mathcal{K}, w') \models \varphi.$$

We follow the precedence $\{\neg, \Diamond, \Box\} > \{\wedge, \vee, \mid\} > \{\Rightarrow, \Leftrightarrow\}$ for the various connectives of $\text{ML}(\mid)$.

As we will see throughout the chapter, despite the addition of the composition operator \mid , the ability to reason within $\text{ML}(\mid)$ comes quite naturally. Arguably, this is mainly due to the following monotonicity property induced by the union $+_w$:

Lemma 7.4. Let $w \in \mathcal{W}$. Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ be two Kripke-style finite forests such that $\mathcal{K}' \subseteq_w \mathcal{K}$. For every $w' \in R'(w)$ and every formula φ in $\text{ML}(\mid)$ we have

$$(\mathcal{K}, w') \models \varphi \text{ if and only if } (\mathcal{K}', w') \models \varphi.$$

Intuitively, this property state that every property of a child of w sill holds when splitting the finite forest with the union $+_w$. Inside $\text{ML}(\mid)$, this translates to the valid formula $\Diamond\varphi \Leftrightarrow \Diamond\varphi \mid \top$. Lemma 7.4 follows directly from the fact that the union $+_w$ guarantees the subtree rooted at w' to be the same in both \mathcal{K} and \mathcal{K}' . Its proof can be found in Appendix E.

7.2.3 $\text{ML}(\mid)$ as a Graded Modal Logic.

Reasoning in $\text{ML}(\mid)$ is quite intuitive, but it is far from being as immediate as for ML . Indeed, in addition to be able to express typical sentences of ML , such as the formula $\Box\varphi \wedge \Diamond\psi$ stating all children satisfy φ and at least one satisfies ψ , the addition of the composition operator leads

to interesting interactions with the \Diamond . For instance, let us consider the formula $(\Box\varphi \wedge \Diamond\psi) \mathbin{|} \top$ obtained by placing $\Box\varphi \wedge \Diamond\psi$ under the context $_ \mathbin{|} \top$. Instead of stating that all children satisfy φ and at least one satisfies ψ , this formula simply states that there is a child satisfying both φ and ψ , i.e. $\Diamond(\varphi \wedge \psi)$. Indeed, $\Box\varphi \wedge \Diamond\psi$ entails the existence of a child satisfying both φ and ψ , whereas the context $_ \mathbin{|} \top$ allows us to forget the properties satisfied by other children. Because of the interplays between the operators \Diamond and \mathbf{I} , deriving an Hilbert-style axiomatisation of $\text{ML}(\mathbf{I})$ becomes certainly an interesting task. This is heightened by the fact that the composition operator \mathbf{I} captures a very natural form of counting that has largely been studied by the modal logic community. Consider the formula $\Diamond_{\geq k}\varphi$ defined below, where $k \geq 1$ is a natural number:

$$\Diamond_{\geq k}\varphi \stackrel{\text{def}}{=} \underbrace{\Diamond\varphi \mathbin{|} \Diamond\varphi \mathbin{|} \cdots \mathbin{|} \Diamond\varphi}_{k-1 \text{ occurrences of } |}$$

Given a finite forest (\mathcal{K}, w) where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, this formula states that it is possible to split \mathcal{K} into k distinct subforests, in all of which the world w has a child satisfying φ . Thanks to the monotonicity property given by Lemma 7.4, this leads to the following semantics:

$$(\mathcal{K}, w) \models \Diamond_{\geq k}\varphi \text{ if and only if } \text{card}(\{w' \in R(w) \mid (\mathcal{K}, w') \models \varphi\}) \geq k,$$

stating that, in \mathcal{K} , at least k distinct children of w verify φ . Now, by enriching the standard modal logic ML with the suites of modalities $\Diamond_{\geq k}\varphi$ ($k \in \mathbb{N} \setminus \{0\}$), where $\Diamond\varphi$ becomes a shortcut for $\Diamond_{\geq 1}\varphi$, we obtain the well-known *graded modal logic* (GML) that was initially introduced during the seventies [79, 70], extensively studied in the eighties [66], with important contributions in the nineties [51]. The syntax of GML is recalled below:

$$\varphi := \top \mathbin{|} p \mathbin{|} \varphi \Rightarrow \varphi \mathbin{|} \neg\varphi \mathbin{|} \Diamond_{\geq k}\varphi, \quad \text{where } k \in \mathbb{N} \setminus \{0\} \text{ and } p \in \text{AP}.$$

From the definition of the *graded modality* $\Diamond_{\geq k}\varphi$ given above, GML can be seen as a syntactical fragment of $\text{ML}(\mathbf{I})$. To be completely precise, we should clarify that GML is usually interpreted on arbitrary Kripke structures (see Section 4.4.2), and not just Kripke-style finite forests. However, it is well-known that GML admits a finite tree model properties [51], i.e. all its satisfiable formulae are satisfied by a finite tree-like Kripke structure. This implies that GML cannot distinguish between arbitrary Kripke structures and Kripke-style finite forests, which allows us to reason about this logic with respect to the latter class of models, while taking advantage of all the known results for Kripke structures.

Interestingly, one can show several instances of formulae in GML that can be characterised in $\text{ML}(\mathbf{I})$ in a more natural way. For instance, let us look at the formula $\Diamond\varphi \mathbin{|} \Diamond\psi$ of $\text{ML}(\mathbf{I})$. Informally, given a pointed finite forest (\mathcal{K}, w) , this formula states that the world w has two distinct children, one satisfying φ and the other satisfying ψ . This property can be captured in GML , but enforcing the two children to be distinct results in a clumsy formula:

$$\Diamond_{\geq 1}\varphi \wedge \Diamond_{\geq 1}\psi \wedge (\Diamond_{\geq 1}(\varphi \wedge \psi) \Rightarrow \Diamond_{\geq 2}\varphi \vee \Diamond_{\geq 2}\psi).$$

Indeed, the formula $\Diamond_{\geq 1}\varphi \wedge \Diamond_{\geq 1}\psi$ is not enough to characterise $\Diamond\varphi \mathbin{|} \Diamond\psi$, as it is satisfied by a pointed forest where w has one child satisfying both φ and ψ . To avoid this case, we take the conjunction of the formula $\Diamond_{\geq 1}\varphi \wedge \Diamond_{\geq 1}\psi$ with the formula $\Diamond_{\geq 1}(\varphi \wedge \psi) \Rightarrow \Diamond_{\geq 2}\varphi \vee \Diamond_{\geq 2}\psi$, as done above. The latter formula states that w has a child satisfying both φ and ψ , there must be a second child satisfying either φ or ψ , as required by $\Diamond\varphi \mathbin{|} \Diamond\psi$.

The example we just discussed highlights quite well the conciseness of $\text{ML}(\mathbf{I})$ with respect of GML . This conciseness comes with a computational price (as we will see in Chapter 8, where

we analyse the complexity of $\text{ML}(\mathbf{|})$). Indeed, $\text{ML}(\mathbf{|})$ is exponentially more succinct than GML , even on very natural queries. For instance, let us consider a finite set of atomic propositions $P \subseteq_{\text{fin}} \text{AP}$. We aim at defining equivalent formulae in GML and $\text{ML}(\mathbf{|})$, that are satisfied by a pointed forest (\mathcal{K}, w) whenever w does not witness two children satisfying exactly the same atomic propositions from P . In GML , the only way of expressing this property is to essentially quantify over every possible truth assignment for atomic propositions, as done in the formula:

$$\text{alldiff}_{\text{GML}}(P) \stackrel{\text{def}}{=} \bigwedge_{f:P \rightarrow \{\top, \perp\}} \neg \Diamond_{\geq 2} (\bigwedge_{p \in P} f(p)=\top p \wedge \bigwedge_{p \in P} f(p)=\perp \neg p).$$

Informally, this formula states that for every possible truth assignment f for the atomic propositions in P , it cannot be that there are two children of w agreeing with f . The formula $\text{alldiff}_{\text{GML}}(P)$ express the right property, but its size is exponential in $\text{card}(P)$. The same property can be expressed in $\text{ML}(\mathbf{|})$ with a formula that is only linear in the size of P . First, we introduce a formula $\text{alleq}(P)$ stating that all the children of w agree on the truth of the atomic propositions in P . This formula is already definable in ML , and corresponds to $\text{alleq}(P) \stackrel{\text{def}}{=} \bigwedge_{p \in P} (\Diamond p \Rightarrow \Box p)$. To achieve the right property, we can now exploit the compositionality of $\text{ML}(\mathbf{|})$ given by the operator $\mathbf{|}$, and state that whenever two children of w are singled out, they do not satisfy the same atomic propositions. The following formula, of size linear in $\text{card}(P)$, does the job:

$$\text{alldiff}(P) \stackrel{\text{def}}{=} \neg (\top \mathbf{|} (\Diamond_{=2} \top \wedge \text{alleq}(P))).$$

Here, given $k \in \mathbb{N}$, we write $\Diamond_{=k}\varphi$ as a shortcut for $\Diamond_{\geq k}\varphi \wedge \neg \Diamond_{\geq k+1}\varphi$, where $\Diamond_{\geq 0}\varphi \stackrel{\text{def}}{=} \top$.

7.3 TOWARDS AN HILBERT-STYLE PROOF SYSTEM FOR $\text{ML}(\mathbf{|})$

With the previous examples, we hope to have given some flavour of the differences between $\text{ML}(\mathbf{|})$ and graded modal logic. As a corollary of the Hilbert-style axiomatisation of $\text{ML}(\mathbf{|})$ defined in this chapter, we show that these differences do not translate to a greater expressive power for $\text{ML}(\mathbf{|})$: every formula of $\text{ML}(\mathbf{|})$ is equivalent to a formula in GML , and vice versa.

Claim 7.5. $\text{ML}(\mathbf{|})$ is as expressive as GML .

Knowing this result beforehand (e.g. through semantical meanings) is incredibly helpful, as it suggest us to design the Hilbert-style proof system by relying on the core formula technique used in Chapter 6 in order to axiomatise the separation logic $\text{SL}(*, -*)$ (see Section 6.3 for a summary of the technique). Graded modal logic provides the core formulae, leaving us with the task of designing the axioms for the composition operator. In particular, we rely on an adequate proof system for GML defined in [49] and denoted by \mathcal{H}_{GML} (Section 7.4), and we introduce $\mathcal{H}_{\text{GML}}(\mathbf{|})$: an extension of \mathcal{H}_{GML} that is adequate for the logic $\text{ML}(\mathbf{|})$ (Section 7.5).

The final proof system $\mathcal{H}_{\text{GML}}(\mathbf{|})$ is given in Figure 7.4. Even though it is quite early to appreciate completely its axioms and rules, we can already spot some interesting analogies between $\mathcal{H}_{\text{GML}}(\mathbf{|})$ and the Hilbert-style proof system $\mathcal{H}_C(*)$ introduced in Chapter 6 to axiomatise the separation logic $\text{SL}(*, x \hookleftarrow _)$ (Figure 6.6). Unsurprisingly, $\mathcal{H}_{\text{GML}}(\mathbf{|})$ features the axioms of non-deterministic monoid (ID^C) , (ASSOC^C) and (COM^C) from BBI , and the inference rule (C) for the composition operator. Perhaps more interesting is the treatment of the graded modalities appearing inside the $\mathbf{|}$. In particular, the axiom (SPLIT^C) states that children satisfying a formula φ can always be separated from the ones not satisfying φ . Interestingly, the axioms (GRAD^C) and (ATOM^C) are analogous to the axioms (SIZE^*) and (ATOM^1) of $\mathcal{H}_C(*)$, recalled below:

Propositional Calculus:

$$(\text{L}_1) \quad (\neg\varphi \Rightarrow \varphi) \Rightarrow \varphi$$

$$(\text{L}_2) \quad \varphi \Rightarrow (\neg\varphi \Rightarrow \psi)$$

$$(\text{L}_3) \quad (\varphi \Rightarrow \psi) \Rightarrow ((\psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \chi))$$

$$(\text{MP}) \quad \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$$

Axioms and rules of the graded modalities:

$$(\square_{\text{INF}}) \quad \square(\varphi \Rightarrow \psi) \Rightarrow (\diamondsuit_{\geq k}\varphi \Rightarrow \diamondsuit_{\geq k}\psi)$$

$$(\diamondsuit_{\text{JOIN}}) \quad \diamondsuit_0(\varphi \wedge \psi) \Rightarrow (\diamondsuit_{=k_1}\varphi \wedge \diamondsuit_{=k_2}\psi \Rightarrow \diamondsuit_{=k_1+k_2}(\varphi \vee \psi))$$

$$(\text{N}) \quad \frac{\varphi}{\square\varphi}$$

Axioms of the composition operator:

$$(\text{C}_{\text{ID}}) \quad \varphi \Leftrightarrow \varphi \mid \square \perp$$

$$(\text{C}_{\text{MONO}}) \quad e \mid \top \Rightarrow e \quad \blacktriangleleft [e \in \{p, \neg p, \diamondsuit \varphi\} \mid p \in \text{AP}, \varphi \text{ in GML}]$$

$$(\text{C}_{\text{ASSOC}}) \quad (\varphi \mid \psi) \mid \chi \Leftrightarrow \varphi \mid (\psi \mid \chi)$$

$$(\text{C}_{\text{GRAD}}) \quad \neg\diamondsuit_{\geq \beta_1}\varphi \mid \neg\diamondsuit_{\geq \beta_2}\varphi \Rightarrow \neg\diamondsuit_{\geq \beta_1 + \beta_2 - 1}\varphi$$

$$(\text{C}_{\text{COM}}) \quad \varphi \mid \psi \Rightarrow \psi \mid \varphi$$

$$(\text{C}_{\text{ATOM}}) \quad \diamondsuit\varphi \Rightarrow \diamondsuit_{=1}\varphi \mid \top$$

$$(\text{C}_{\text{ZERO}}) \quad (\perp \mid \varphi) \Leftrightarrow \perp$$

$$(\text{C}_{\text{SPLIT}}) \quad \square\varphi \mid \square\neg\varphi$$

$$(\text{C}_{\text{DIST}}) \quad (\varphi \vee \psi) \mid \chi \Rightarrow (\varphi \mid \chi) \vee (\psi \mid \chi)$$

Rules of inference for the composition operator:

$$(\text{C}) \quad \frac{\varphi \Rightarrow \chi}{\varphi \mid \psi \Rightarrow \chi \mid \psi}$$

Figure 7.4: The Hilbert-style proof system $\mathcal{H}_{\text{GML}}(\mathbf{I})$.

$$(\text{SIZE}^*) \quad \neg\text{size} \geq \beta_1 * \neg\text{size} \geq \beta_2 \Rightarrow \neg\text{size} \geq \beta_1 + \beta_2 - 1 \quad (\text{ATOM}^{*1}) \quad \neg\text{emp} \Rightarrow \text{size} = 1 * \top,$$

where we remind the reader that $i \div j = \max(0, i - j)$ and that $\neg\text{emp}$ is $\text{size} \geq 1$. These correspondences between axioms of the two proof systems are quite revealing, and essentially summarise how we think of graded modalities throughout the chapter. Very roughly, we see two distinct formulae $\diamondsuit_{\geq k}\varphi$ and $\diamondsuit_{\geq j}\psi$ as size predicates that are checked with respect to disjoint heaps, say h_φ and h_ψ . So, $\diamondsuit_{\geq k}\varphi$ could in a sense be seen as a query stating that the heap h_φ satisfies $\text{size} \geq k$. With this in mind, many of the syntactical proof we did in Chapter 6 transfer to $\mathcal{H}_{\text{GML}}(\mathbf{I})$ with a simple change of notation (e.g. $\text{size} \geq \beta$ becomes $\diamondsuit_{\geq \beta}\varphi$) and axioms (e.g. we use (C_{GRAD}) instead of (SIZE^*)). The disjointness of h_φ and h_ψ translates back to $\diamondsuit_{\geq k}\varphi$ and $\diamondsuit_{\geq j}\psi$ as we ask $\varphi \wedge \psi$ to be unsatisfiable. As we later formally discuss (see Lemma 7.11), every formula in GML can be manipulated to obtain an equivalent formula satisfying this auxiliary constraint. To give some flavour on how this is achieved within the proof system $\mathcal{H}_{\text{GML}}(\mathbf{I})$, let us consider the formula $\diamondsuit_{\geq k}\varphi \mid \neg\diamondsuit_{\geq j}\psi$, where we assume φ and ψ to be non-equivalent (otherwise we can simply substitute ψ by φ). By relying on the two following theorems of propositional calculus

$$\varphi \Leftrightarrow (\varphi \wedge \psi) \vee (\varphi \wedge \neg\psi), \quad \psi \Leftrightarrow (\varphi \wedge \psi) \vee (\neg\varphi \wedge \psi),$$

together with the fact that the replacement of equivalent formulae is admissible in $\mathcal{H}_{\text{GML}}(\mathbf{I})$ (see the rule (S) in the proof of Theorem 7.16), one can show within $\mathcal{H}_{\text{GML}}(\mathbf{I})$ that $\Diamond_k \varphi \Vdash \neg\Diamond_j \psi$ is equivalent to

$$\Diamond_{\geq k}((\varphi \wedge \psi) \vee (\varphi \wedge \neg\psi)) \Vdash \neg\Diamond_{\geq j}((\varphi \wedge \psi) \vee (\neg\varphi \wedge \psi)). \quad (\dagger)$$

Now, when looking at the left conjunct $\Diamond_{\geq k}((\varphi \wedge \psi) \vee (\varphi \wedge \neg\psi))$, we notice that the disjuncts $\varphi \wedge \psi$ and $\varphi \wedge \neg\psi$ cannot be simultaneously satisfied, which implies that $\Diamond_{=0}((\varphi \wedge \psi) \wedge (\varphi \wedge \neg\psi))$ is valid. By relying on the axiom (\Diamond_{JOIN}) , one can prove the following theorem of GML

$$\Diamond_{=0}(\varphi_1 \wedge \varphi_2) \Rightarrow (\Diamond_{\geq k}(\varphi_1 \vee \varphi_2) \Leftrightarrow \bigvee_{\substack{k_1, k_2 \in \mathbb{N} \\ k=k_1+k_2}} (\Diamond_{\geq k_1} \varphi_1 \wedge \Diamond_{\geq k_2} \varphi_2)),$$

which shows that $\Diamond_{\geq k}((\varphi \wedge \psi) \vee (\varphi \wedge \neg\psi))$, is equivalent to $\bigvee_{k=k_1+k_2} (\Diamond_{\geq k_1}(\varphi \wedge \psi) \wedge \Diamond_{\geq k_2}(\varphi \wedge \neg\psi))$. Similarly, $\Diamond_{\geq j}((\varphi \wedge \psi) \vee (\neg\varphi \wedge \psi))$ is equivalent to $\bigvee_{j=j_1+j_2} (\Diamond_{\geq j_1}(\varphi \wedge \psi) \wedge \Diamond_{\geq j_2}(\neg\varphi \wedge \psi))$. From (\dagger) , we conclude that $\Diamond_{\geq k} \varphi \Vdash \neg\Diamond_{\geq j} \psi$ is equivalent to

$$\left(\bigvee_{k=k_1+k_2} (\Diamond_{\geq k_1}(\varphi \wedge \psi) \wedge \Diamond_{\geq k_2}(\varphi \wedge \neg\psi)) \right) \Vdash \neg \left(\bigvee_{j=j_1+j_2} (\Diamond_{\geq j_1}(\varphi \wedge \psi) \wedge \Diamond_{\geq j_2}(\neg\varphi \wedge \psi)) \right)$$

In this formula, the subformulae $\Diamond_{\geq k_1}(\varphi \wedge \psi)$, $\Diamond_{\geq k_2}(\varphi \wedge \neg\psi)$, $\Diamond_{\geq j_1}(\varphi \wedge \psi)$ and $\Diamond_{\geq j_2}(\neg\varphi \wedge \psi)$ satisfy the required ‘‘disjointness’’ property: all conjunctions of distinct formulae appearing inside the graded modality (i.e. conjunctions of $\varphi \wedge \psi$, $\varphi \wedge \neg\psi$ and $\neg\varphi \wedge \psi$), are unsatisfiable.

In the next section, we recall and briefly analyse the axiom system of GML, from [49]. Afterwards, we establish the role of GML as a set of core formulae of $\text{ML}(\mathbf{I})$, as well as formalising the idea of disjoint formulae introduced above (Definition 7.9).

7.4 GRADED MODALITIES AS CORE FORMULAE

In Figure 7.5, we recall the Hilbert-style proof system \mathcal{H}_{GML} introduced in [49], where it is proved to be sound and complete for graded modal logic.

Theorem 7.6 (Adequacy, [49]). A formula φ in GML is valid if and only if $\vdash_{\mathcal{H}_{\text{GML}}} \varphi$.

Let us look more closely at \mathcal{H}_{GML} . As we can see, the proof system extends propositional calculus with three more axioms and one rule. Let (\mathcal{K}, w) be a pointed forest, and let us write S_φ for the set of children of w satisfying a formula φ . Both the axioms (\Box_{INF}) and (\Diamond_{JOIN}) are quite natural. The axiom (\Box_{INF}) essentially state that if the set S_φ is a subset of S_ψ (i.e. $\Box(\varphi \Rightarrow \psi)$), then $\text{card}(S_\varphi) \leq \text{card}(S_\psi)$. This axiom pairs well with the necessitation rule (N) from ML, as it allows us to derive the following inference rule for graded modalities:

Proof of (G).

$(G) \quad \frac{\varphi \Rightarrow \psi}{\Diamond_{\geq k} \varphi \Rightarrow \Diamond_{\geq k} \psi}$	$1 \quad \varphi \Rightarrow \psi$ $2 \quad \Box(\varphi \Rightarrow \psi)$ $3 \quad \Box(\varphi \Rightarrow \psi) \Rightarrow (\Diamond_{\geq k} \varphi \Rightarrow \Diamond_{\geq k} \psi)$ $4 \quad \Diamond_{\geq k} \varphi \Rightarrow \Diamond_{\geq k} \psi$	Hypothesis $(N), 1$ (\Box_{INF}) $(MP), 2, 3$
---	---	---

Propositional Calculus:

$$(L_1) \quad (\neg\varphi \Rightarrow \varphi) \Rightarrow \varphi$$

$$(L_2) \quad \varphi \Rightarrow (\neg\varphi \Rightarrow \psi)$$

$$(L_3) \quad (\varphi \Rightarrow \psi) \Rightarrow ((\psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \chi))$$

$$(MP) \quad \frac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$$

Axioms and rules of the graded modalities:

$$(I_{\text{INF}}^{\square}) \quad \square(\varphi \Rightarrow \psi) \Rightarrow (\diamondsuit_{\geq k}\varphi \Rightarrow \diamondsuit_{\geq k}\psi)$$

$$(J_{\text{JOIN}}^{\diamondsuit}) \quad \diamondsuit_{=0}(\varphi \wedge \psi) \Rightarrow (\diamondsuit_{=k_1}\varphi \wedge \diamondsuit_{=k_2}\psi \Rightarrow \diamondsuit_{=k_1+k_2}(\varphi \vee \psi))$$

$$(N) \quad \frac{\varphi}{\square\varphi}$$

Intermediate axiom:

$$(I_1^{\text{GML}}) \quad \diamondsuit_{\geq k+1}\varphi \Rightarrow \diamondsuit_{\geq k}\varphi$$

Figure 7.5: The Hilbert-style proof system \mathcal{H}_{GML} .

If $(I_{\text{INF}}^{\square})$ talks about inclusion between sets of children of w , the axiom $(J_{\text{JOIN}}^{\diamondsuit})$ talks about empty intersections. In particular, it states that if $S_\varphi \cap S_\psi = \emptyset$ (i.e. $\diamondsuit_{=0}(\varphi \wedge \psi)$), $\text{card}(S_\varphi) = k_1$ and $\text{card}(S_\psi) = k_2$, then $\text{card}(S_\varphi \cup S_\psi) = k_1 + k_2$. Lastly, \mathcal{H}_{GML} features the axiom (I_1^{GML}) which essentially states that if $\text{card}(S_\varphi) \geq k + 1$ then $\text{card}(S_\varphi) \geq k$. This axiom is reminiscent of the axiom (I_1^C) introduced to axiomatise the core formulae of $\text{SL}(*, x \hookrightarrow -)$, and recalled below:

$$\text{size} \geq \beta + 1 \Rightarrow \text{size} \geq \beta.$$

The similarity does not end there, as both (I_1^C) and (I_1^{GML}) are derivable in the proof systems $\mathcal{H}_C(*)$ and $\mathcal{H}_{\text{GML}}(\mathbf{I})$, respectively. Following the terminology and notation $I_i^?$ introduced in the previous chapter, this makes (I_1^{GML}) an intermediate axiom: an axiom that must be considered in order for \mathcal{H}_{GML} to be complete, but is superfluous when \mathcal{H}_{GML} is extended in order to capture the composition operator.

Using GML as core formulae. When using GML as core formulae of $\text{ML}(\mathbf{I})$, we want to restrict ourselves to formulae of a particular shape that facilitate reasoning about the composition operator. As already formalised with Lemma 7.4, the fundamental property of $\text{ML}(\mathbf{I})$ is that, given a pointed forest (\mathcal{K}, w) , the operator \mathbf{I} does not modify the subtrees rooted at the children of w , which therefore keep satisfying the same formulae. This monotonicity property allows us to be often unconcerned by the shape of the formulae appearing inside a (graded) modality, and to treat them almost as atomic propositions. This leads to the following definition of core formulae, that is parametric on a (fixed) set Φ of GML formulae.

Definition 7.7 (Core formulae). Let Φ be a finite set of formulae in GML. Consider a natural number $k \in \mathbb{N}$ and let P be a finite set of atomic propositions. We denote with $\text{Core}(\Phi, k, P)$ the following set of formulae in GML:

$$\mathbf{Core}(\Phi, k, P) \stackrel{\text{def}}{=} \{\Diamond_{\geq j} \varphi, p, \top \mid j \in [1, k], \varphi \in \Phi, p \in P\}.$$

Essentially, given a pointed forest (\mathcal{K}, w) , the formulae in $\mathbf{Core}(\Phi, k, P)$ express which atomic propositions among P are verified in w , as well as how many children of w satisfy each formula in Φ , up to the bound k .

Similarly to Chapter 6, given a set of core formulae Γ , we write $\mathbf{Conj}(\Gamma)$ for the set of (finite) conjunctions of literals built upon Γ , where a *literal* is understood as a formula of Γ or its negation. Given a conjunction $\varphi = L_1 \wedge \dots \wedge L_n$ of literals L_1, \dots, L_n , we write $\mathbf{LIT}(\varphi)$ to denote $\{L_1, \dots, L_n\}$. Given two conjunctions of formulae φ and ψ , $\psi \subseteq_{\mathbf{LIT}} \varphi$ stands for $\mathbf{LIT}(\psi) \subseteq \mathbf{LIT}(\varphi)$. We also use the following shortcuts from Chapter 6:

- $\chi \subseteq_{\mathbf{LIT}} \{\varphi \mid \psi\}$ for “ $\chi \subseteq_{\mathbf{LIT}} \varphi$ or $\chi \subseteq_{\mathbf{LIT}} \psi$ ”,
- $\{\varphi \mid \psi\} \subseteq_{\mathbf{LIT}} \chi$ for “ $\varphi \subseteq_{\mathbf{LIT}} \chi$ or $\psi \subseteq_{\mathbf{LIT}} \chi$ ”,
- $\chi \subseteq_{\mathbf{LIT}} \{\varphi ; \psi\}$ for “ $\chi \subseteq_{\mathbf{LIT}} \varphi$ and $\chi \subseteq_{\mathbf{LIT}} \psi$ ”.

Given a set of formulae $\Gamma = \{\varphi_1, \dots, \varphi_n\}$, we write $\bigwedge \Gamma$ for $\varphi_1 \wedge \dots \wedge \varphi_n$.

We introduce further new notions of **GML**. Given a formula φ in **GML**, we write $\mathbf{top}_{\mathbf{gm}}(\varphi)$ for the set of subformulae ψ of φ such that ψ is of the form $\Diamond_{\geq j} \chi$ and one of its occurrences in φ is not in the scope of the graded modalities $\Diamond_{\geq k}$. Formally, $\mathbf{top}_{\mathbf{gm}}(\varphi)$ is inductively defined as:

$$\begin{aligned} \mathbf{top}_{\mathbf{gm}}(\top) &\stackrel{\text{def}}{=} \mathbf{top}_{\mathbf{gm}}(p) \stackrel{\text{def}}{=} \emptyset, \\ \mathbf{top}_{\mathbf{gm}}(\Diamond_{\geq k} \varphi) &\stackrel{\text{def}}{=} \{\Diamond_{\geq k} \varphi\}, \\ \mathbf{top}_{\mathbf{gm}}(\neg \varphi) &\stackrel{\text{def}}{=} \mathbf{top}_{\mathbf{gm}}(\varphi), \\ \mathbf{top}_{\mathbf{gm}}(\varphi \Rightarrow \psi) &\stackrel{\text{def}}{=} \mathbf{top}_{\mathbf{gm}}(\varphi) \cup \mathbf{top}_{\mathbf{gm}}(\psi). \end{aligned}$$

Similarly, we write $\mathbf{top}_{\mathbf{AP}}(\varphi)$ for the set of atomic propositions of φ that appear outside graded modalities. Its formal definition is analogous to the one of $\mathbf{top}_{\mathbf{gm}}(\varphi)$, the only difference being that $\mathbf{top}_{\mathbf{AP}}(p) = \{p\}$ and $\mathbf{top}_{\mathbf{AP}}(\Diamond_{\geq k} \varphi) = \emptyset$. For instance, given the formula φ defined as $(p \Rightarrow \Diamond_{\geq k} q) \wedge \neg(q \vee \Diamond_{\geq k} \Diamond_{\geq 2r})$, we have $\mathbf{top}_{\mathbf{gm}}(\varphi) = \{\Diamond_{\geq k} q, \Diamond_{\geq k} \Diamond_{\geq 2r}\}$ and $\mathbf{top}_{\mathbf{AP}}(\varphi) = \{p, q\}$. Notice that all the formulae in $\mathbf{top}_{\mathbf{gm}}(\varphi)$ and $\mathbf{top}_{\mathbf{AP}}(\varphi)$ are core formulae.

Proposition 7.8. Every formula φ in **GML** is equivalent to a disjunction of formulae belonging to $\mathbf{Conj}(\mathbf{top}_{\mathbf{gm}}(\varphi) \cup \mathbf{top}_{\mathbf{AP}}(\varphi))$.

Assuming $\mathbf{top}_{\mathbf{gm}}(\varphi) = \{\Diamond_{\geq k_1} \psi_1, \dots, \Diamond_{\geq k_n} \psi_n\}$, the proposition above implies that φ is equivalent to a disjunction of formulae belonging to $\mathbf{Conj}(\mathbf{Core}(\{\psi_1, \dots, \psi_n\}, \max(k_1, \dots, k_n), \mathbf{top}_{\mathbf{AP}}(\varphi)))$. Its proof, left to the reader, carries out by simply putting φ in disjunctive normal form while seeing the formulae in $\mathbf{top}_{\mathbf{gm}}(\varphi)$ as atomic propositions. As $\mathcal{H}_{\mathbf{GML}}$ is complete for **GML**, the equivalence of Proposition 7.8 is provable within $\mathcal{H}_{\mathbf{GML}}$.

As stressed at the end of the previous section, we often ask the formulae Φ that are involved in the definition of $\mathbf{Core}(\Phi, k, P)$ to satisfy the following *disjointness* property.

Definition 7.9 (Disjointness). Let $\Phi = \{\varphi_1, \dots, \varphi_n\}$ be a finite set of formulae in **GML**. We say that the formulae in Φ are *disjoint* whenever for every distinct $i, j \in [1, n]$, $\varphi_i \wedge \varphi_j$ is unsatisfiable.

The idea behind this notion is that, given two disjoint formulae φ and ψ , the satisfaction of a formula of the form $\Diamond_{\geq k} \varphi$ is completely independent from the truth of formulae of the form $\Diamond_{\geq j} \psi$. Provided that we can always restrict ourselves to disjoint formulae (as we show below),

this implies that we can design the Hilbert-style axiom system for $\text{ML}(\mathbf{I})$ by looking at the interaction of \mathbf{I} with formulae of the form $\Diamond_{\geq k}\varphi$ for a fixed metavariable φ .

When $\Phi = \{\varphi_1, \dots, \varphi_n\}$ is not a set of disjoint formulae, we can easily refine it so that it becomes one. To do so, we introduce the set 2^Φ defined below:

$$2^\Phi \stackrel{\text{def}}{=} \left\{ \bigwedge_{\substack{i \in [1, n] \\ f(i) = \top}} \varphi_i \wedge \bigwedge_{\substack{i \in [1, n] \\ f(i) = \perp}} \neg\varphi_i \mid \begin{array}{l} f : [1, n] \rightarrow \{\top, \perp\}, \\ f(i) = \top \text{ for some } i \in [1, n] \end{array} \right\}.$$

Informally, 2^Φ contains conjunctions of every formula in Φ , possibly negated, where at least one formula of Φ occurs positively. It is easy to see that the formulae in 2^Φ are disjoint (Lemma 7.10), and that disjunctions of formulae in $\text{Core}(2^\Phi, k, P)$ capture $\text{Core}(\Phi, k, P)$ (Lemma 7.11).

Lemma 7.10. Let Φ be a finite set of formulae in GML. 2^Φ is a set of disjoint formulae.

Proof. Directly from the definition of 2^Φ . Indeed, consider distinct formulae φ and ψ in 2^Φ . By definition of 2^Φ , both formulae are conjunctions of all the formulae in Φ (appearing positively or negatively). Since φ and ψ are distinct, there is a formula χ in Φ that appears positively in φ and negatively in ψ , or vice versa. Thus, $\varphi \wedge \psi$ is unsatisfiable, as it entails $\chi \wedge \neg\chi$. \square

Lemma 7.11. Let Φ be a finite set of GML formulae, $k \in \mathbb{N}$ and $P \subseteq_{\text{fin}} \text{AP}$. Every formula in $\text{Core}(\Phi, k, P)$ is equivalent to a disjunction of formulae in $\text{Conj}(\text{Core}(2^\Phi, k, P))$.

Proof. Let $\Phi = \{\psi_1, \dots, \psi_n\}$ and consider a formula φ in $\text{Core}(\Phi, k, P)$. Trivially, if φ is an atomic proposition in P , then it belongs to $\text{Core}(2^\Phi, k, P)$. Otherwise, $\varphi = \Diamond_{\geq k}\psi_i$, for some $i \in [1, n]$. By propositional reasoning, ψ_i is equivalent to the formula

$$\psi_i \wedge \bigwedge_{j \in [1, n]} (\psi_j \vee \neg\psi_j),$$

where we essentially applied the *tertium non datur* principle $\chi \vee \neg\chi$ to every formula among ψ_1, \dots, ψ_n . As conjunction distributes over disjunction and is associative, commutative and idempotent (i.e. $\chi \equiv \chi \wedge \chi$), this formula is equivalent to

$$\bigvee_{\substack{f : [1, n] \rightarrow \{\top, \perp\} \\ f(i) = \top}} \left(\bigwedge_{i \in [1, n]} \psi_i \wedge \bigwedge_{i \in [1, n]} \neg\psi_i \right). \quad (\dagger)$$

where the external disjunction quantifies over all possible evaluations for ψ_1, \dots, ψ_n having ψ_i verified. Every disjunct of this formula of the form $\bigwedge_{i \in [1, n]} f(i) = \top \psi_i \wedge \bigwedge_{i \in [1, n]} f(i) = \perp \neg\psi_i$ belongs to 2^Φ . Let $\{\chi_1, \dots, \chi_l\} \subseteq 2^\Phi$ be the set of (distinct) disjuncts appearing in (\dagger) , so that $\psi_i \equiv \chi_1 \vee \dots \vee \chi_l$. By Lemma 7.10, $\{\chi_1, \dots, \chi_l\}$ is a set of disjoint formulae. In order to conclude the proof, it is sufficient to show that $\Diamond_{\geq k}\psi_i \equiv \gamma$ where

$$\gamma \stackrel{\text{def}}{=} \bigvee_{\substack{k_1, \dots, k_l \in \mathbb{N} \\ k = k_1 + \dots + k_l}} (\Diamond_{\geq k_1}\chi_1 \wedge \dots \wedge \Diamond_{\geq k_l}\chi_l).$$

Indeed, by definition of γ , every formula of the form $\Diamond_{\geq j}\chi$ appearing in $\text{top}_{\text{gm}}(\gamma)$ is such that $j \leq k$ and $\chi \in 2^\Phi$, which entails that $\text{top}_{\text{gm}}(\gamma) \subseteq \text{Core}(2^\Phi, k, P)$. In the definition of γ , recall that a formula $\Diamond_{\geq 0}\tilde{\varphi}$ stands for \top . Below, let (\mathcal{K}, w) be a pointed forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. (\Rightarrow): Suppose $(\mathcal{K}, w) \models \Diamond_{\geq k}\psi_i$, and thus there are at least k distinct worlds w_1, \dots, w_k in $R(w)$ such that $(\mathcal{K}, w_j) \models \psi_i$, for every $j \in [1, k]$. Since $\psi_i \equiv \chi_1 \vee \dots \vee \chi_l$ and $\{\chi_1, \dots, \chi_l\}$ is a set of disjoint formulae, each world in $\{w_1, \dots, w_k\}$ satisfies exactly one formula in $\{\chi_1, \dots, \chi_l\}$. Given

Axioms and rules from \mathcal{H}_{GML} (Figure 7.5), excluding (I_1^{GML}) .

Axioms of the composition operator:

(C_{ID})	$\varphi \Leftrightarrow \varphi \mid \square \perp$	(C_{MONO})	$e \mid \top \Rightarrow e \triangleleft [e \in \{p, \neg p, \Diamond \varphi\} \mid p \in \text{AP}, \varphi \text{ in GML}]$
(C_{ASSOC})	$(\varphi \mid \psi) \mid \chi \Leftrightarrow \varphi \mid (\psi \mid \chi)$	$(\neg C_{\text{GRAD}})$	$\neg \Diamond_{\geq \beta_1} \varphi \mid \neg \Diamond_{\geq \beta_2} \varphi \Rightarrow \neg \Diamond_{\geq \beta_1 + \beta_2 - 1} \varphi$
(C_{COM})	$\varphi \mid \psi \Rightarrow \psi \mid \varphi$	(C_{ATOM})	$\Diamond \varphi \Rightarrow \Diamond_{=1} \varphi \mid \top$
(C_{ZERO})	$(\perp \mid \varphi) \Leftrightarrow \perp$	(C_{SPLIT})	$\Box \varphi \mid \Box \neg \varphi$
(C_{DIST})	$(\varphi \vee \psi) \mid \chi \Rightarrow (\varphi \mid \chi) \vee (\psi \mid \chi)$		

Rules of inference for the composition operator:

$$(C) \quad \frac{\varphi \Rightarrow \chi}{\varphi \mid \psi \Rightarrow \chi \mid \psi}$$

Figure 7.6: The Hilbert-style proof system $\mathcal{H}_{\text{GML}}(\mid)$ (again).

$j \in [1, l]$, let k_j be the number of worlds in $\{w_1, \dots, w_k\}$ satisfying the formula χ_j . We have $(\mathcal{K}, w) \models \Diamond_{\geq k_1} \chi_1 \wedge \dots \wedge \Diamond_{\geq k_l} \chi_l$, and thus γ is satisfied.

(\Leftarrow): Suppose $(\mathcal{K}, w) \models \gamma$, and thus there are $k_1, \dots, k_l \in \mathbb{N}$ such that $k = k_1 + \dots + k_l$ and $(\mathcal{K}, w) \models \Diamond_{\geq k_1} \chi_1 \wedge \dots \wedge \Diamond_{\geq k_l} \chi_l$. So, there are l subsets S_1, \dots, S_l of $R(w)$ such that for every $j \in [1, l]$ and $w' \in S_j$, $(\mathcal{K}, w') \models \chi_j$. Moreover, $\text{card}(S_j) \geq k_j$. Since $\{\chi_1, \dots, \chi_l\}$ is a set of disjoint formulae, the sets S_1, \dots, S_l are mutually disjoint, and thus their union T contains at least k worlds. From $\psi_i \equiv \chi_1 \vee \dots \vee \chi_l$, every world w' in T is such that $(\mathcal{K}, w') \models \psi_i$. We conclude that $(\mathcal{K}, w) \models \Diamond_{\geq k} \psi_i$. \square

7.5 SYNTACTICAL ELIMINATION OF THE COMPOSITION OPERATOR

In Figure 7.6, we recall the Hilbert-style proof system $\mathcal{H}_{\text{GML}}(\mid)$ that we show to be adequate for the modal logic $\text{ML}(\mid)$. As we can see, $\mathcal{H}_{\text{GML}}(\mid)$ is obtained by enriching \mathcal{H}_{GML} with axioms and rules that handle the composition operator \mid , and that were briefly discussed during Section 7.3. The axiom (I_1^{GML}) is removed from the system, as it is now derivable.

Lemma 7.12. The axiom (I_1^{GML}) of \mathcal{H}_{GML} is derivable in $\mathcal{H}_{\text{GML}}(\mid)$.

Proof. The proof, by induction on k , follows exactly the proof of the intermediate axiom (I_1^C) with respect to the proof system $\mathcal{H}_C(\ast)$, given in Lemma 6.13.

base case: $k = 0$. The instance of the axiom (I_1^{GML}) with $\beta = 0$ amounts to derive the formula $\Diamond_{\geq 1} \varphi \Rightarrow \Diamond_{\geq 0} \varphi$. As $\Diamond_{\geq 0} \varphi = \top$, by propositional reasoning, $\vdash_{\mathcal{H}_{\text{GML}}(\mid)} \Diamond_{\geq 1} \varphi \Rightarrow \Diamond_{\geq 0} \varphi$.

induction step: $k > 0$. By induction hypothesis, assume $\vdash_{\mathcal{H}_{\text{GML}}(\mid)} \Diamond_{\geq k} \varphi \Rightarrow \Diamond_{\geq k-1} \varphi$. The formula $\Diamond_{\geq k+1} \varphi \Rightarrow \Diamond_{\geq k} \varphi$ is derived as follows:

1	$\Diamond_{\geq k}\varphi \Rightarrow \Diamond_{\geq k-1}\varphi$	Induction Hypothesis
2	$(\Diamond_{\geq k}\varphi) \parallel \Diamond\varphi \Rightarrow (\Diamond_{\geq k-1}\varphi) \parallel \Diamond\varphi$	(C) , 1
3	$\Diamond_{\geq k+1}\varphi \Rightarrow \Diamond_{\geq k}\varphi$	2, def. of $\Diamond_{\geq j}\varphi$ \square

A quick analysis on the axioms and rules establishes the soundness of $\mathcal{H}_{\text{GML}}(\mathbf{I})$.

Lemma 7.13. $\mathcal{H}_{\text{GML}}(\mathbf{I})$ is sound.

Proof. The soundness of all the axioms from \mathcal{H}_{GML} follows from Theorem 7.6. The validity of the axioms (COM) , (ASSOC) , (DIST) and (ZERO) , as well as the admissibility of the rule (C) are inherited from BBI (see [75]). Below, let (\mathcal{K}, w) be a pointed forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

Validity of the axiom $(\text{ID})^{\text{C}}$.

(\Rightarrow) : Suppose $(\mathcal{K}, w) \models \varphi$. Let $\mathcal{K}' = (\mathcal{W}, \emptyset, \mathcal{V})$. Clearly, $\mathcal{K} = \mathcal{K} +_w \mathcal{K}'$ and $(\mathcal{K}, w) \models \Box \perp$. Therefore, $(\mathcal{K}, w) \models \varphi \parallel \Box \perp$.

(\Leftarrow) : Suppose $(\mathcal{K}, w) \models \varphi \parallel \Box \perp$. Thus, there are $\mathcal{K}_1 = (w, R_1, \mathcal{V})$ and \mathcal{K}_2 such that $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$, $(\mathcal{K}_1, w) \models \varphi$ and $(\mathcal{K}_2, w) \models \Box \perp$. From the semantics of $\Box \perp$, we conclude that $R(w) = R_1(w)$. By definition of $+_w$, $R^+(w) = R_1^+(w)$ and $R_1 \subseteq R$. This means that the subtree rooted at w in \mathcal{K} is the same as the one in \mathcal{K}_1 , which implies that (\mathcal{K}_1, w) and (\mathcal{K}, w) satisfy the same formulae of $\text{ML}(\mathbf{I})$. Thus, $(\mathcal{K}, w) \models \varphi$. For a formal proof that (\mathcal{K}_1, w) and (\mathcal{K}, w) satisfy the same formulae, see Lemma E.2 in Appendix E.

Validity of the axiom $(\text{SPLIT})^{\text{C}}$.

Consider two Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$ and for every $w' \in R(w)$, $w' \in R_1(w)$ if and only if $(\mathcal{K}, w') \models \varphi$. By Lemma 7.4, for every $w' \in R_1(w)$, $(\mathcal{K}_1, w') \models \varphi$, whereas for every $w'' \in R_2(w)$, $(\mathcal{K}_2, w'') \not\models \varphi$. So, $(\mathcal{K}_1, w) \models \Box \varphi$ and $(\mathcal{K}_2, w) \models \Box \neg \varphi$, which entails $(\mathcal{K}, w) \models \Box \varphi \parallel \Box \neg \varphi$.

Validity of the axiom $(\text{MONO})^{\text{C}}$.

The case where $e = \Diamond \varphi$ follows directly from Lemma 7.4. Suppose $(\mathcal{K}, w) \models e \parallel \top$, where e is either p or $\neg p$, for some $p \in \text{AP}$. There is $\mathcal{K}_1 \subseteq_w \mathcal{K}$ such that $(\mathcal{K}_1, w) \models e$. By definition, if $e = p$ then $w \in \mathcal{V}(p)$, otherwise ($e = \neg p$) $w \notin \mathcal{V}(p)$. We conclude that $(\mathcal{K}, w) \models e$.

Validity of the axiom $(\text{GRAD})^{\text{C}}$.

Let $k_1, k_2 \geq 0$. Suppose $(s, h) \models \neg \Diamond_{\geq k_1} \varphi \parallel \neg \Diamond_{\geq k_2} \varphi$. Then, there are two finite forests \mathcal{K}_1 and \mathcal{K}_2 such that $\mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2$, $(\mathcal{K}_1, w) \models \neg \Diamond_{\geq k_1} \varphi$ and $(\mathcal{K}_2, w) \models \neg \Diamond_{\geq k_2} \varphi$. From the semantics of the graded modality,

$$\text{card}(\{w' \in R_1(w) \mid (\mathcal{K}_1, w') \models \varphi\}) < k_1, \quad \text{card}(\{w' \in R_2(w) \mid (\mathcal{K}_2, w') \models \varphi\}) < k_2.$$

By definition of $+_w$, $R_1(w) \cap R_2(w) = \emptyset$ and $R(w) = R_1(w) \cup R_2(w)$. By Lemma 7.4,

$$\{w' \in R_1(w) \mid (\mathcal{K}_1, w') \models \varphi\} \cup \{w' \in R_2(w) \mid (\mathcal{K}_2, w') \models \varphi\} = \{w' \in R(w) \mid (\mathcal{K}, w') \models \varphi\}.$$

We have $\text{card}(\{w' \in R(w) \mid (\mathcal{K}, w') \models \varphi\}) < k_1 + k_2 - 1$, and thus $(\mathcal{K}, w) \models \neg \Diamond_{\geq k_1 + k_2 - 1} \varphi$.

Validity of the axiom $(\text{ATOM})^{\text{C}}$.

Suppose $(\mathcal{K}, w) \models \Diamond_{\geq 1} \varphi$. Then, there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models \varphi$. Let \mathcal{K}' be the finite forest obtained from \mathcal{K} by removing all elements of R that do not involve worlds in $R^*(w')$. Formally, $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ where $R' = \{(w_1, w_2) \in R \mid w_2 \in R^*(w')\}$. Clearly, $\mathcal{K}' \subseteq_w \mathcal{K}$ and $R'(w) = \{w'\}$. By Lemma 7.4, $(\mathcal{K}', w') \models \varphi$. From $R'(w) = \{w'\}$, we conclude that $(\mathcal{K}', w) \models \Diamond_{=1} \varphi$. From $\mathcal{K}' \subseteq_w \mathcal{K}$, $(\mathcal{K}, w) \models \Diamond_{=1} \varphi \parallel \top$. \square

$$\begin{aligned}
& \bigwedge \{\Diamond_{\geq j} \chi \mid \chi \in \Phi, \Diamond_{\geq j} \chi \subseteq_{\text{LIT}} \{\varphi \mid \psi\}\} \\
& \wedge \bigwedge \{\Diamond_{\geq j_1+j_2} \chi \mid \chi \in \Phi, \Diamond_{\geq j_1} \chi \subseteq_{\text{LIT}} \varphi, \Diamond_{\geq j_2} \chi \subseteq_{\text{LIT}} \psi\} \\
& \wedge \bigwedge \{\neg \Diamond_{\geq j_1+j_2+1} \chi \mid \chi \in \Phi, \neg \Diamond_{\geq j_1} \chi \subseteq_{\text{LIT}} \varphi, \neg \Diamond_{\geq j_2} \chi \subseteq_{\text{LIT}} \psi\} \\
& \wedge \bigwedge \{p \in \text{AP} \mid p \subseteq_{\text{LIT}} \{\varphi \mid \psi\}\} \wedge \bigwedge \{\neg p \mid p \in \text{AP}, \neg p \subseteq_{\text{LIT}} \{\varphi \mid \psi\}\}
\end{aligned}$$

Figure 7.7: the GML formula $\langle \parallel \rangle(\varphi, \psi)$, where φ in $\text{Core}(\Phi, k_1, P_1)$ and ψ in $\text{Core}(\Phi, k_2, P_2)$.

In order to show the completeness of $\mathcal{H}_{\text{GML}}(\parallel)$, we prove that the proof system enjoys the \vdash -simulation property. That is, given two formulae φ and ψ in GML, there is a formula χ in GML such that $\vdash_{\mathcal{H}_{\text{GML}}(\parallel)} \varphi \parallel \psi \Leftrightarrow \chi$. This property is first shown for the case where φ and ψ are satisfiable conjunctions of core formulae, and then extended to arbitrary formulae of GML, thanks to Proposition 7.8.

Lemma 7.14. Let Φ be a set of disjoint formulae in GML, $k_1, k_2 \in \mathbb{N}$, and $P_1, P_2 \subseteq_{\text{fin}} \text{AP}$. For every two satisfiable formulae φ in $\text{Conj}(\text{Core}(\Phi, k_1, P_1))$ and ψ in $\text{Conj}(\text{Core}(\Phi, k_2, P_2))$. Then,

$$\vdash_{\mathcal{H}_{\text{GML}}(\parallel)} \varphi \parallel \psi \Leftrightarrow \langle \parallel \rangle(\varphi, \psi).$$

The GML formula $\langle \parallel \rangle(\varphi, \psi)$ of Lemma 7.14 is defined in Figure 7.7. From its definition, it is easy to see that $\langle \parallel \rangle(\varphi, \psi)$ belongs to $\text{Conj}(\text{Core}(\Phi, k_1 + k_2, P_1 \cup P_2))$. In order to prove Lemma 7.14 we rely on the following theorems and admissible rules of $\mathcal{H}_{\text{GML}}(\parallel)$.

Lemma 7.15. The following axioms and rules are admissible in $\mathcal{H}_{\text{GML}}(\parallel)$:

$$\begin{array}{ll}
(\Rightarrow \text{TR}) \quad \frac{\varphi \Rightarrow \chi \quad \chi \Rightarrow \psi}{\varphi \Rightarrow \psi} & (I_{7.15.1}^*) \quad \Diamond_{\geq k_1} \varphi \parallel \Diamond_{\geq k_2} \varphi \Rightarrow \Diamond_{\geq k_1+k_2} \varphi \\
& (I_{7.15.2}^*) \quad \Diamond_{\geq k} \varphi \Rightarrow \Diamond_{=k} \varphi \parallel \top \\
(\parallel \text{LR}) \quad \frac{\varphi \Rightarrow \varphi' \quad \psi \Rightarrow \psi'}{\varphi \parallel \psi \Rightarrow \varphi' \parallel \psi'} & (I_{7.15.3}^*) \quad \Box \varphi \wedge (\psi \parallel \chi) \Rightarrow (\psi \wedge \Box \varphi) \parallel (\chi \wedge \Box \varphi) \\
& (I_{7.15.4}^*) \quad \Box \varphi_1 \parallel \cdots \parallel \Box \varphi_n \parallel (\Box \neg \varphi_1 \wedge \cdots \wedge \Box \neg \varphi_n)
\end{array}$$

The proof of Lemma 7.15 can be found in Appendix E.

Proof of Lemma 7.14. (\Rightarrow): In order to prove that $\vdash_{\mathcal{H}_{\text{GML}}(\parallel)} \varphi \parallel \psi \Rightarrow \langle \parallel \rangle(\varphi, \psi)$, we establish that $\vdash_{\mathcal{H}_{\text{GML}}(\parallel)} \varphi \parallel \psi \Rightarrow L$ holds for every literal L of $\langle \parallel \rangle(\varphi, \psi)$, by case analysis on L .

case: L is either p or $\neg p$. By definition of $\langle \parallel \rangle(\varphi, \psi)$, either $L \subseteq_{\text{LIT}} \varphi$ or $L \subseteq_{\text{LIT}} \psi$. Let us assume that $L \subseteq_{\text{LIT}} \varphi$. Then,

$$\begin{array}{llll}
1 \quad \varphi \Rightarrow L & \text{PC} & 4 \quad L \parallel \top \Rightarrow L & (\text{MONO}) \\
2 \quad \psi \Rightarrow \top & \text{PC} & 5 \quad \varphi \parallel \psi \Rightarrow L & (\Rightarrow \text{TR}), 3, 4 \\
3 \quad \varphi \parallel \psi \Rightarrow L \parallel \top & (\parallel \text{LR}), 1, 2 & &
\end{array}$$

In the case where $L \subseteq_{\text{LIT}} \psi$, we first use the commutativity of \parallel (axiom $(\text{COM})^C$) to derive $\vdash_{\mathcal{H}_{\text{GML}}(\parallel)} \varphi \parallel \psi \Rightarrow \psi \parallel \varphi$. We then rely on the derivation above, swapping φ and ψ , to conclude that $\vdash_{\mathcal{H}_{\text{GML}}(\parallel)} \psi \parallel \varphi \Rightarrow L$. Lastly, by $(\Rightarrow \text{TR})$, we get $\vdash_{\mathcal{H}_{\text{GML}}(\parallel)} \varphi \parallel \psi \Rightarrow L$.

case: $L = \Diamond_{\geq j} \chi$, where $\Diamond_{\geq j} \chi \subseteq_{\text{LIT}} \{\varphi \mid \psi\}$. The proof is divided in three cases, depending on whether $j = 0$, $j = 1$ or $j \geq 2$.

case: $j = 0$. Since $\Diamond_{\geq 0} \chi = \top$, $\vdash_{\mathcal{H}_{\text{GML}}(\|)} \varphi \|\psi \Rightarrow L$ follows by propositional reasoning.

case: $j = 1$. In this case we have $\Diamond_{\geq j} \chi = \Diamond \chi$. If $\Diamond \chi \subseteq_{\text{LIT}} \varphi$, then the proof of $\varphi \|\psi \Rightarrow L$ is exactly as the one shown above for the case where L is an atomic proposition or its negation. The other case ($\Diamond \chi \subseteq_{\text{LIT}} \psi$) is analogous, thanks to the axiom $(\text{COM})^C$.

case: $j \geq 2$. By definition of $\Diamond_{\geq j} \chi$ and from the associativity and commutativity of $\|$ (axioms $(\text{ASSOC})^C$ and $(\text{COM})^C$), $\vdash_{\mathcal{H}_{\text{GML}}(\|)} \Diamond_{\geq j} \chi \Rightarrow (\Diamond_{\geq j-1} \chi) \|\Diamond \chi$. If $\Diamond_{\geq j} \chi \subseteq_{\text{LIT}} \varphi$, then

1	$\Diamond_{\geq j} \chi \Rightarrow (\Diamond_{\geq j-1} \chi) \ \Diamond \chi$	See above
2	$\varphi \Rightarrow \Diamond_{\geq j} \chi$	PC
3	$\psi \Rightarrow \top$	PC
4	$\varphi \ \psi \Rightarrow ((\Diamond_{\geq j-1} \chi) \ \Diamond \chi) \ \top$	$(\Rightarrow \text{TR}), (\text{II}_{LR})$, 1, 2, 3
5	$((\Diamond_{\geq j-1} \chi) \ \Diamond \chi) \ \top \Rightarrow (\Diamond \chi \ \top) \ (\Diamond_{\geq j-1} \chi)$	$(\text{ASSOC})^C, (\text{COM})^C$
6	$\Diamond \chi \ \top \Rightarrow \Diamond \chi$	$(\text{MONO})^C$
7	$(\Diamond \chi \ \top) \ (\Diamond_{\geq j-1} \chi) \Rightarrow \Diamond \chi \ (\Diamond_{\geq j-1} \chi)$	$(\text{C}), 6$
8	$\Diamond \chi \ (\Diamond_{\geq j-1} \chi) \Rightarrow \Diamond_{\geq j} \chi$	$(\text{ASSOC})^C, (\text{COM})^C$, def. of. $\Diamond_{\geq j} \chi$
9	$\varphi \ \psi \Rightarrow \Diamond_{\geq j} \chi$	$(\Rightarrow \text{TR})$, 4, 5, 7, 8

Again, if instead $\Diamond_{\geq j} \chi \subseteq_{\text{LIT}} \psi$, we simply rely on the commutativity of $\|$ and then carry out the proof as for the case where $\Diamond_{\geq j} \chi \subseteq_{\text{LIT}} \varphi$.

case: $L = \Diamond_{\geq j_1+j_2} \chi$, **where** $\Diamond_{\geq j_1} \chi \subseteq_{\text{LIT}} \varphi$ **and** $\Diamond_{\geq j_2} \chi \subseteq_{\text{LIT}} \psi$. We divide the proof in two cases, depending on whether $j_1, j_2 \geq 1$ holds.

case: $j_1 = 0$ **or** $j_2 = 0$. In this case, either $\Diamond_{\geq j_1+j_2} \chi \subseteq_{\text{LIT}} \varphi$ or $\Diamond_{\geq j_1+j_2} \chi \subseteq_{\text{LIT}} \psi$, and thus we derive $\vdash_{\mathcal{H}_{\text{GML}}(\|)} \varphi \|\chi \Rightarrow L$ by relying on the previous case of the proof.

case: $j_1 \geq 1$ **and** $j_2 \geq 1$. The proof of this case is straightforward:

1	$\varphi \Rightarrow \Diamond_{\geq j_1} \chi$	PC
2	$\varphi \Rightarrow \Diamond_{\geq j_2} \chi$	PC
3	$\varphi \ \psi \Rightarrow \Diamond_{\geq j_1} \chi \ \Diamond_{\geq j_2} \chi$	(II_{LR}) , 1, 2
4	$\Diamond_{\geq j_1} \chi \ \Diamond_{\geq j_2} \chi \Rightarrow \Diamond_{\geq j_1+j_2} \chi$	$(\text{ASSOC})^C, (\text{COM})^C$, def. of. $\Diamond_{\geq j_1+j_2} \chi$
5	$\varphi \ \psi \Rightarrow \Diamond_{\geq j_1+j_2} \chi$	$(\Rightarrow \text{TR})$, 3, 4

case: $L = \neg \Diamond_{\geq j_1+j_2+1}$, **where** $\neg \Diamond_{\geq j_1} \subseteq_{\text{LIT}} \varphi$ **and** $\neg \Diamond_{\geq j_2} \subseteq_{\text{LIT}} \psi$. Directly from $(\neg \text{GRAD})^C$:

1	$\varphi \Rightarrow \neg \Diamond_{\geq j_1} \chi$	PC	4	$\neg \Diamond_{\geq j_1} \chi \ \neg \Diamond_{\geq j_2} \chi \Rightarrow \neg \Diamond_{\geq j_1+j_2+1} \chi$	$(\neg \text{GRAD})^C$
2	$\varphi \Rightarrow \neg \Diamond_{\geq j_2} \chi$	PC	5	$\varphi \ \psi \Rightarrow \neg \Diamond_{\geq j_1+j_2+1} \chi$	$(\Rightarrow \text{TR})$, 3, 4
3	$\varphi \ \psi \Rightarrow \neg \Diamond_{\geq j_1} \chi \ \neg \Diamond_{\geq j_2} \chi$	(II_{LR}) , 1, 2			

(\Leftarrow) : Let us show that $\vdash_{\mathcal{H}_{\text{GML}}(\|)} (\|)(\varphi, \psi) \Rightarrow \varphi \|\psi$. This direction is quite similar to the right-to-left direction of Lemma 6.14, where we showed that $\mathcal{H}_C(*)$ (i.e. the proof system of $\text{SL}(*, \mathbf{x} \hookrightarrow _)$) enjoy the $*$ -simulation property. The central property that allows us to carry out the proof is the one of disjoint formulae. To depict the usefulness of this property, let us pick two disjoint formulae χ_1 and χ_2 , and let us suppose that $\varphi = \Diamond_{=1} \chi_1 \wedge \Diamond_{\geq 2} \chi_2$ whereas $\psi = \Diamond_{=2} \chi_1$. So, by definition, $(\|)(\varphi, \psi)$ is equivalent to the formula $\Diamond_{=3} \chi_1 \wedge \Diamond_{\geq 2} \chi_2$. As a first step, we rely

on $(I_{7.15.4}^*)$ to derive $\square\chi_1 \parallel \square\chi_2 \parallel (\square\neg\chi_1 \wedge \square\neg\chi_2)$, which in a sense allows us to separate worlds satisfying χ_1 from the ones satisfying χ_2 . Since χ_1 and χ_2 are disjoint formulae, $\square\chi_2 \Rightarrow \square\neg\chi_1$ is valid, which allows us to show that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \square\chi_2 \parallel (\square\neg\chi_1 \wedge \square\neg\chi_2) \Rightarrow \square\neg\chi_1$. This allows us enrich the left hand side of $\square\chi_1 \parallel \square\chi_2 \parallel (\square\neg\chi_1 \wedge \square\neg\chi_2)$ with all the literals of $\langle \parallel \rangle(\varphi, \psi)$ of the form $\Diamond_{\geq k}\chi_1$ or $\neg\Diamond_{\geq k}\chi_1$. We can do the same with χ_2 , and obtain a derivation in $\mathcal{H}_{\text{GML}}(\mathbf{I})$ of

$$\langle \parallel \rangle(\varphi, \psi) \Rightarrow (\square\chi_1 \wedge \Diamond_{=3}\chi_1) \parallel (\square\chi_2 \wedge \Diamond_{\geq 2}\chi_2) \parallel (\square\neg\chi_1 \wedge \square\neg\chi_2).$$

We can now apply the axiom $(I_{7.15.2}^*)$ locally, on the subformulae $\square\chi_1 \wedge \Diamond_{=3}\chi_1$ and $\square\chi_2 \wedge \Diamond_{\geq 2}\chi_2$, in order to derive formulae that follow the definitions of φ and ψ . More precisely, from $\square\chi_1 \wedge \Diamond_{=3}\chi_1$ we derive the formula $(\square\chi_1 \wedge \Diamond_{=1}\chi_1) \parallel (\square\chi_1 \wedge \Diamond_{=2}\chi_1)$, where we notice that the left hand side contains $\Diamond_{=1}\chi_1$ from φ , whereas the right hand side contains $\Diamond_{=2}\chi_1$ from ψ . By relying on the commutativity and associativity of \parallel , this allows us to derive:

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \parallel \rangle(\varphi, \psi) \Rightarrow ((\square\chi_1 \wedge \Diamond_{=1}\chi_1) \parallel (\square\chi_2 \wedge \Diamond_{\geq 2}\chi_2)) \parallel ((\square\chi_1 \wedge \Diamond_{=2}\chi_1) \parallel (\square\neg\chi_1 \wedge \square\neg\chi_2))$$

Again thanks to the disjointness of χ_1 and χ_2 , we show that the $(\square\chi_1 \wedge \Diamond_{=1}\chi_1) \parallel (\square\chi_2 \wedge \Diamond_{\geq 2}\chi_2)$ entails φ , whereas $(\square\chi_1 \wedge \Diamond_{=2}\chi_1) \parallel (\square\neg\chi_1 \wedge \square\neg\chi_2)$ entails ψ , allowing us to derive $\langle \parallel \rangle(\varphi, \psi) \Rightarrow \varphi \parallel \psi$.

Below, we generalise these manipulations for arbitrary formulae φ and ψ . We assume $\langle \parallel \rangle(\varphi, \psi)$ to be satisfiable, as otherwise from the completeness of \mathcal{H}_{GML} with respect to **GML** we are able to derive $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \parallel \rangle(\varphi, \psi) \Rightarrow \perp$, which then allows us to show $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \parallel \rangle(\varphi, \psi) \Rightarrow \varphi \parallel \psi$ by pure propositional reasoning. We also remind the reader that φ and ψ are assumed satisfiable.

Step 1, partitioning the formula. Given χ in Φ , we write $\text{ATOM}(\chi)$ for the formula:

$$\text{ATOM}(\chi) \stackrel{\text{def}}{=} \square\chi \wedge \bigwedge \{\Diamond_{\geq j}\chi \mid \Diamond_{\geq j}\chi \subseteq_{\text{LIT}} \langle \parallel \rangle(\varphi, \psi)\} \wedge \bigwedge \{\neg\Diamond_{\geq j}\chi \mid \neg\Diamond_{\geq j}\chi \subseteq_{\text{LIT}} \langle \parallel \rangle(\varphi, \psi)\}.$$

Roughly speaking, $\text{ATOM}(\chi)$ contains all the literals of $\langle \parallel \rangle(\varphi, \psi)$ featuring the formula χ , together with the formula $\square\chi$. Notice that $\text{ATOM}(\chi)$ can be equal to $\square\chi$, for instance in the case where χ does not appear in φ or ψ , and thus neither in $\langle \parallel \rangle(\varphi, \psi)$. Let $\Phi = \{\chi_1, \dots, \chi_n\}$. In this step of the proof, we aim at showing that the following formula is derivable in $\mathcal{H}_{\text{GML}}(\mathbf{I})$:

$$\langle \parallel \rangle(\varphi, \psi) \Rightarrow \text{ATOM}(\chi_1) \parallel \dots \parallel \text{ATOM}(\chi_n) \parallel (\square\neg\chi_1 \wedge \dots \wedge \square\neg\chi_n). \quad (\dagger)$$

Given $i \in [1, n]$, we write $\gamma_i^{(1)}$ for $\square\chi_i$. By $(I_{7.15.4}^*)$, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \gamma_1^{(1)} \parallel \dots \parallel \gamma_n^{(1)} \parallel (\square\neg\chi_1 \wedge \dots \wedge \square\neg\chi_n)$, and thus by propositional reasoning we have

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \parallel \rangle(\varphi, \psi) \Rightarrow \gamma_1^{(1)} \parallel \dots \parallel \gamma_n^{(1)} \parallel (\square\neg\chi_1 \wedge \dots \wedge \square\neg\chi_n).$$

We now rely on $\langle \parallel \rangle(\varphi, \psi)$ to add to every $\gamma_i^{(1)}$ all the missing literals appearing in $\text{ATOM}(\chi_i)$. We add the literals progressively, building a sequence of formulae

$$\begin{aligned} & \gamma_1^{(1)} \parallel \dots \parallel \gamma_n^{(1)} \parallel (\square\neg\chi_1 \wedge \dots \wedge \square\neg\chi_n), \\ & \gamma_1^{(2)} \parallel \dots \parallel \gamma_n^{(2)} \parallel (\square\neg\chi_1 \wedge \dots \wedge \square\neg\chi_n), \\ & \dots \\ & \gamma_1^{(k)} \parallel \dots \parallel \gamma_n^{(k)} \parallel (\square\neg\chi_1 \wedge \dots \wedge \square\neg\chi_n), \end{aligned}$$

where for all $j \in [1, k]$ we have $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \parallel \rangle(\varphi, \psi) \Rightarrow \gamma_1^{(j)} \parallel \dots \parallel \gamma_n^{(j)} \parallel (\square\neg\chi_1 \wedge \dots \wedge \square\neg\chi_n)$, and for all $i \in [1, n]$ and $j' \in [1, j]$, $\gamma_i^{(j')} \subseteq_{\text{LIT}} \gamma_i^{(j)}$. As we obtain $\gamma_i^k = \text{ATOM}(\chi_i)$ (up to associativity and commutativity of \wedge), this allows us to prove (\dagger) within $\mathcal{H}_{\text{GML}}(\mathbf{I})$. Below, we assume that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \parallel \rangle(\varphi, \psi) \Rightarrow \gamma_1^{(j)} \parallel \dots \parallel \gamma_n^{(j)} \parallel (\square\neg\chi_1 \wedge \dots \wedge \square\neg\chi_n)$$

holds, and consider $i \in [1, n]$ such that $\gamma_i^{(j)}$ is missing a literal L of $\text{ATOM}(\chi_i)$. We show that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \gamma_1^{(j)} | \cdots | \gamma_n^{(j)} | (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n)$$

by case analysis on L , concluding the proof of (\dagger) .

case: $L = \neg \diamond \geq_k \chi_i$. By definiton of $\text{ATOM}(\chi_i)$, $L \subseteq_{\text{LIT}} \langle \mathbf{I} \rangle(\varphi, \psi)$. So,

1	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \gamma_1^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n)$	Hypothesis
2	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \neg \diamond \geq_k \chi_i$	PC
3	$\gamma_i^{(j)} \Rightarrow (\gamma_i^{(j)} \wedge \diamond \geq_k \chi_i) \vee (\gamma_i^{(j)} \wedge \neg \diamond \geq_k \chi_i)$	PC
4	$\gamma_1^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow$ $\gamma_i^{(j)} \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n)$	(ASSOC) , (COM)
5	$\gamma_i^{(j)} \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow$ $((\gamma_i^{(j)} \wedge \diamond \geq_k \chi_i) \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$ $\vee ((\gamma_i^{(j)} \wedge \neg \diamond \geq_k \chi_i) \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$	(C) , 3, (DIST)
6	$\gamma_i^{(j)} \wedge \diamond \geq_k \chi_i \Rightarrow \diamond \geq_k \chi_i$	PC
7	$\gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \top$	PC
8	$(\gamma_i^{(j)} \wedge \diamond \geq_k \chi_i) \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \diamond \geq_k \chi_i \mid \top (\text{I}_{LR})$, 6, 7	
9	$\diamond \geq_k \chi_i \mid \top \Rightarrow \diamond \geq_k \chi_i$	$(\text{I}_{7.15.1})$, $\diamond \geq_0 \chi_i = \top$
10	$(\gamma_i^{(j)} \wedge \neg \diamond \geq_k \chi_i) \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \neg \diamond \geq_k \chi_i$	$(\Rightarrow \text{TR})$, 8, 9
11	$\gamma_i^{(j)} \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow$ $\diamond \geq_k \chi_i$ $\vee ((\gamma_i^{(j)} \wedge \neg \diamond \geq_k \chi_i) \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$	PC, 5, 10
12	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow$ $((\gamma_i^{(j)} \wedge \neg \diamond \geq_k \chi_i) \gamma_1^{(j)} \cdots \gamma_{i-1}^{(j)} \gamma_{j+1}^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$	PC, 1, 2, 4, 11
13	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \gamma_1^{(j)} \cdots (\gamma_i^{(j)} \wedge L) \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n)$	PC, (ASSOC) , (COM) , 12

case: $L = \diamond \geq_k \chi_i$. In this case, $\diamond \geq_k \chi_i \subseteq_{\text{LIT}} \langle \mathbf{I} \rangle(\varphi, \psi)$. We rely on the disjointness of the formulae in Φ . Recall that for every $l \in [1, n]$ different from i , $\chi_i \wedge \chi_l$ is unsatisfiable, which entails the validity of the formula $\square \chi_l \Rightarrow \square \neg \chi_i$ in GML. As χ_i and χ_l are formulae in GML, and $\square \chi_l = \gamma_l^{(1)} \subseteq_{\text{LIT}} \gamma_l^{(j)}$, this allows us to derive $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \gamma_l^{(j)} \Rightarrow \square \neg \chi_i$ directly from the completeness of \mathcal{H}_{GML} (Theorem 7.6). Moreover, by definition of \square and axiom (GRAD) , $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \square \neg \chi_i \mid \square \neg \chi_i \Rightarrow \square \neg \chi_i$. This allows us to conclude that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \gamma_1^{(j)} | \cdots | \gamma_{i-1}^{(j)} | \gamma_{j+1}^{(j)} | \cdots | \gamma_n^{(j)} | (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \square \neg \chi_i.$$

The proof proceeds as follows:

1	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \gamma_1^{(j)} \cdots \gamma_n^{(j)} (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n)$	Hypothesis
2	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \diamond \geq_k \chi_i$	PC

3	$\gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \square \neg \chi_i$	See above
4	$\gamma_i^{(j)} \Rightarrow (\gamma_i^{(j)} \wedge \diamond_{\geq k} \chi_i) \vee (\gamma_i^{(j)} \wedge \neg \diamond_{\geq k} \chi_i)$	PC
5	$\gamma_1^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow$ $\gamma_i^{(j)} \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n)$	$(\text{ASSOC})^C, (\text{COM})^C$
6	$\gamma_i^{(j)} \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow$ $((\gamma_i^{(j)} \wedge \diamond_{\geq k} \chi_i) \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$ $\vee ((\gamma_i^{(j)} \wedge \neg \diamond_{\geq k} \chi_i) \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$	$(\text{C}), 4, (\text{DIST})^C$
7	$\gamma_i^{(j)} \wedge \neg \diamond_{\geq k} \chi_i \Rightarrow \neg \diamond_{\geq k} \chi_i$	PC
8	$(\gamma_i^{(j)} \wedge \diamond_{\geq k} \chi_i) \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow$ $\neg \diamond_{\geq k} \chi_i \mid \square \neg \chi_i$	$(\text{IR}_{LR}), 3, 7$
9	$\neg \diamond_{\geq k} \chi_i \mid \square \neg \chi_i \Rightarrow \neg \diamond_{\geq k} \chi_i$	$(\text{GRAD})^C$, def. of \square
10	$(\gamma_i^{(j)} \wedge \neg \diamond_{\geq k} \chi_i) \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow$ $\neg \diamond_{\geq k} \chi_i$	$(\Rightarrow \text{TR}), 8, 9$
11	$\gamma_i^{(j)} \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow$ $((\gamma_i^{(j)} \wedge \diamond_{\geq k} \chi_i) \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$ $\vee \neg \diamond_{\geq k} \chi_i$	PC, 6, 10
12	$\langle \mid \rangle(\varphi, \psi) \Rightarrow$ $((\gamma_i^{(j)} \wedge \diamond_{\geq k} \chi_i) \mid \gamma_1^{(j)} \mid \cdots \mid \gamma_{i-1}^{(j)} \mid \gamma_{j+1}^{(j)} \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$	PC, 1, 2, 5, 11
13	$\langle \mid \rangle(\varphi, \psi) \Rightarrow \gamma_1^{(j)} \mid \cdots \mid (\gamma_i^{(j)} \wedge L) \mid \cdots \mid \gamma_n^{(j)} \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n)$	PC, $(\text{ASSOC})^C, (\text{COM})^C$, 12

Step 2, splitting the atoms. In the previous step, we have shown that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mid)} \langle \mid \rangle(\varphi, \psi) \Rightarrow \text{ATOM}(\chi_1) \mid \cdots \mid \text{ATOM}(\chi_n) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n).$$

In this step, we focus on a single formula $\text{ATOM}(\chi_i)$ ($i \in [1, n]$), and show that the following formula is derivable in $\mathcal{H}_{\text{GML}}(\mid)$:

$$\text{ATOM}(\chi_i) \Rightarrow \mathsf{A}_\varphi(\chi_i) \mid \mathsf{A}_\psi(\chi_i), \quad (\ddagger)$$

where, given γ among φ and ψ , $\mathsf{A}_\gamma(\chi_i)$ is defined as the formula

$$\mathsf{A}_\gamma(\chi_i) \stackrel{\text{def}}{=} \square \chi_i \wedge \bigwedge \{\diamond_{\geq j} \chi_i \mid \diamond_{\geq j} \chi_i \subseteq_{\text{LIT}} \gamma\} \wedge \bigwedge \{\neg \diamond_{\geq j} \chi_i \mid \neg \diamond_{\geq j} \chi_i \subseteq_{\text{LIT}} \gamma\}.$$

Essentially, $\mathsf{A}_\gamma(\chi_i)$ contains all the literals of $\text{LIT}(\gamma)$ featuring the formula $\diamond_{\geq j} \chi_i$, together with the formula $\square \chi_i$. In order to show (\ddagger) , we first establish an easier entailment. Let us introduce the values k_φ and $\overline{k_\varphi}$ defined as follows:

$$k_\varphi = \max(\{k \mid \diamond_{\geq k} \chi_i \subseteq_{\text{LIT}} \varphi\} \cup \{0\}), \quad \overline{k_\varphi} = \min(\{k \mid \diamond_{\geq k} \chi_i \subseteq_{\text{LIT}} \varphi\} \cup \{k_1 + 1\}).$$

Informally, the value k_φ represent the greatest coefficient for graded modalities such that $\diamond_{\geq k_\varphi} \chi_i$ occurs positively in φ . If no such coefficient exists, $k_\varphi = 0$. Similarly, $\overline{k_\varphi}$ is the smallest coefficient for a graded modality such that $\neg \diamond_{\geq \overline{k_\varphi}} \chi_i \subseteq_{\text{LIT}} \varphi$. If φ does not have any subformula of the form $\neg \diamond_{\geq k} \chi_i$, then $\overline{k_\varphi} = k_1 + 1$, which is an upper bound to all the coefficients of graded modalities

appearing at the top level of φ , since φ is a conjunction of core formulae from $\text{Core}(\Phi, k_1, P)$. We do the same for the formula ψ , and introduce:

$$k_\psi = \max(\{k \mid \Diamond_{\geq k} \chi_i \subseteq_{\text{LIT}} \psi\} \cup \{0\}), \quad \bar{k}_\psi = \min(\{k \mid \Diamond_{\geq k} \chi_i \subseteq_{\text{LIT}} \psi\} \cup \{k_2 + 1\}).$$

It is quite easy to see that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ATOM}(\chi_i) \Rightarrow \Diamond_{\geq k_\varphi + k_\psi} \chi_i. \quad (\eta)$$

Indeed, if $k_\varphi = 0$ and $k_\psi = 0$, then the proof follows by propositional reasoning, as $\Diamond_{\geq k_\varphi + k_\psi} \chi_i$ is defined as \top . Otherwise, suppose $k_\varphi + k_\psi \geq 1$. In this case, we have $\Diamond_{\geq k_\varphi + k_\psi} \chi_i \subseteq_{\text{LIT}} \langle \mathbf{I} \rangle(\varphi, \psi)$ (see first two lines of the definition of $\langle \mathbf{I} \rangle(\varphi, \psi)$, Figure 7.7). By definition of $\text{ATOM}(\chi_i)$, we derive $\Diamond_{\geq k_\varphi + k_\psi} \chi_i \subseteq_{\text{LIT}} \text{ATOM}(\chi_i)$. By propositional reasoning, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ATOM}(\chi_i) \Rightarrow \Diamond_{\geq k_\varphi + k_\psi} \chi_i$.

Now, let us define the formula $U(\varphi)$ which is defined as follows:

$$U(\varphi) \stackrel{\text{def}}{=} \begin{cases} \Diamond_{\geq k_\varphi} \chi_i & \text{if there is no } k \in \mathbb{N} \text{ such that } \neg \Diamond_{\geq k} \chi_i \subseteq_{\text{LIT}} \varphi \\ \Diamond_{\geq k_\varphi} \chi_i \wedge \neg \Diamond_{\geq \bar{k}_\varphi} \chi_i & \text{otherwise} \end{cases}$$

The formula $U(\psi)$ is defined in a similar way. Before tackling the derivation of (\ddagger) , we show that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ATOM}(\chi_i) \Rightarrow U(\varphi) \parallel U(\psi)$. The proof is by cases on the shape of $U(\varphi)$ and $U(\psi)$.

case: $U(\varphi) = \Diamond_{\geq k_\varphi} \chi_i$ and $U(\psi) = \Diamond_{\geq k_\psi} \chi_i$. We have,

1	$\text{ATOM}(\chi_i) \Rightarrow \Diamond_{\geq k_\varphi + k_\psi} \chi_i$	(η)
2	$\Diamond_{\geq k_\varphi + k_\psi} \chi_i \Rightarrow \Diamond_{\geq k_\varphi} \chi_i$	PC, repeated (I_1^{GML})
3	$\Diamond_{\geq k_\varphi} \chi_i \Rightarrow \Diamond_{= k_\varphi} \chi_i \parallel \top$	$(I_{7.15.2}^*)$
4	$\top \Rightarrow \Diamond_{\geq k_\psi} \chi_i \vee \neg \Diamond_{\geq k_\psi} \chi_i$	PC
5	$\Diamond_{= k_\varphi} \chi_i \parallel \top \Rightarrow (\Diamond_{\geq k_\psi} \chi_i \vee \neg \Diamond_{\geq k_\psi} \chi_i) \parallel \Diamond_{= k_\varphi} \chi_i$	$(\text{C}_{\text{COM}}), (\text{C}), 4$
6	$(\Diamond_{\geq k_\psi} \chi_i \vee \neg \Diamond_{\geq k_\psi} \chi_i) \parallel \Diamond_{= k_\varphi} \chi_i \Rightarrow (\Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i) \vee (\neg \Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i)$	(C_{DIST})
7	$\Diamond_{= k_\varphi} \chi_i \Rightarrow \neg \Diamond_{\geq k_\varphi + 1} \chi_i$	PC, def. of $\Diamond_{= k_\varphi} \chi_i$
8	$\neg \Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i \Rightarrow \neg \Diamond_{\geq k_\varphi + 1} \chi_i \parallel \neg \Diamond_{\geq k_\psi} \chi_i$	$(\text{C}_{\text{COM}}), (\text{C}), 7$
9	$\neg \Diamond_{\geq k_\varphi + 1} \chi_i \parallel \neg \Diamond_{\geq k_\psi} \chi_i \Rightarrow \neg \Diamond_{\geq k_\varphi + k_\psi} \chi_i$	$(\neg \text{GRAD})$
10	$\neg \Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i \Rightarrow \neg \Diamond_{\geq k_\varphi + k_\psi} \chi_i$	$(\Rightarrow \text{TR}), 8, 9$
11	$(\Diamond_{\geq k_\psi} \chi_i \vee \neg \Diamond_{\geq k_\psi} \chi_i) \parallel \Diamond_{= k_\varphi} \chi_i \Rightarrow (\Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i) \vee \neg \Diamond_{\geq k_\varphi + k_\psi} \chi_i$	PC, 6, 10
12	$\Diamond_{\geq k_\varphi} \chi_i \Rightarrow (\Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i) \vee \neg \Diamond_{\geq k_\varphi + k_\psi} \chi_i$	$(\Rightarrow \text{TR}), 3, 5, 11$
13	$\Diamond_{\geq k_\varphi + k_\psi} \chi_i \Rightarrow \Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i$	PC, 2, 12
14	$\Diamond_{= k_\varphi} \chi_i \Rightarrow \Diamond_{k_\varphi} \chi_i$	PC, def. of $\Diamond_{= k_\varphi} \chi_i$
15	$\Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i \Rightarrow \Diamond_{\geq k_\varphi} \chi_i \parallel \Diamond_{\geq k_\psi} \chi_i$	$(\text{C}_{\text{COM}}), (\text{C}), 14$
16	$\text{ATOM}(\chi_i) \Rightarrow \underbrace{\Diamond_{\geq k_\varphi} \chi_i \parallel \Diamond_{\geq k_\psi} \chi_i}_{U(\varphi) \parallel U(\psi)}$	$(\Rightarrow \text{TR}), 1, 13, 15$

case: $U(\varphi) = \Diamond_{\geq k_\varphi} \chi_i \wedge \neg \Diamond_{\geq \bar{k}_\varphi} \chi_i$ and $U(\psi) = \Diamond_{\geq k_\psi} \chi_i$. In the previous case of the proof, we have shown that $\Diamond_{\geq k_\varphi + k_\psi} \chi_i \Rightarrow \Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{= k_\varphi} \chi_i$ is a theorem of $\mathcal{H}_{\text{GML}}(\mathbf{I})$ (line 13). Since φ is satisfiable, we have $k_\varphi < \bar{k}_\varphi$. By repeated applications of (I_1^{GML}) and propositional reason-

ing, this allows us conclude that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Diamond_{=k_\varphi} \chi_i \Rightarrow \Diamond_{\geq k_\varphi} \chi_i \wedge \neg \Diamond_{\geq \overline{k}_\varphi} \chi_i$. The proof then carries out as follows:

1	$\text{ATOM}(\chi_i) \Rightarrow \Diamond_{\geq k_\varphi + k_\psi} \chi_i$	(η)
2	$\Diamond_{\geq k_\varphi + k_\psi} \chi_i \Rightarrow \Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{=k_\varphi} \chi_i$	See above
3	$\Diamond_{=k_\varphi} \chi_i \Rightarrow \Diamond_{\geq k_\varphi} \chi_i \wedge \neg \Diamond_{\geq \overline{k}_\varphi} \chi_i$	See above
4	$\Diamond_{\geq k_\psi} \chi_i \parallel \Diamond_{=k_\varphi} \chi_i \Rightarrow (\Diamond_{\geq k_\varphi} \chi_i \wedge \neg \Diamond_{\geq \overline{k}_\varphi} \chi_i) \parallel \Diamond_{\geq k_\psi} \chi_i$	$(\text{COM}^{\text{C}}), (\text{C}), 3$
5	$\text{ATOM}(\chi_i) \Rightarrow \underbrace{(\Diamond_{\geq k_\varphi} \chi_i \wedge \neg \Diamond_{\geq \overline{k}_\varphi} \chi_i)}_{\mathbf{U}(\varphi)} \parallel \underbrace{\Diamond_{\geq k_\psi} \chi_i}_{\mathbf{U}(\psi)}$	$(\Rightarrow \text{TR}), 1, 2, 4$

case: $\mathbf{U}(\psi) = \Diamond_{\geq k_\psi} \chi_i \wedge \neg \Diamond_{\geq \overline{k}_\psi} \chi_i$ **and** $\mathbf{U}(\varphi) = \Diamond_{\geq k_\varphi} \chi_i$. Symmetrical to the previous case. We rely on the commutativity of the composition operator (axiom (COM^{C})).

case: $\mathbf{U}(\varphi) = \Diamond_{\geq k_\varphi} \chi_i \wedge \neg \Diamond_{\geq \overline{k}_\varphi} \chi_i$ **and** $\mathbf{U}(\psi) = \Diamond_{\geq k_\psi} \chi_i \wedge \neg \Diamond_{\geq \overline{k}_\psi} \chi_i$. In this case, by definition of $\langle \mathbf{I} \rangle(\varphi, \psi)$, we have $\neg \Diamond_{\geq \overline{k}_\varphi + \overline{k}_\psi - 1} \chi_i \subseteq_{\text{LIT}} \langle \mathbf{I} \rangle(\varphi, \psi)$. Thus, by definition of $\text{ATOM}(\chi_i)$, we derive $\neg \Diamond_{\geq \overline{k}_\varphi + \overline{k}_\psi - 1} \chi_i \subseteq_{\text{LIT}} \text{ATOM}(\chi_i)$. Together with (η) , this allows us to conclude that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ATOM}(\chi_i) \Rightarrow \Diamond_{\geq k_\varphi + k_\psi} \chi_i \wedge \neg \Diamond_{\geq k_\varphi + k_\psi - 1} \chi_i.$$

Since φ and ψ are satisfiable, necessarily $k_\varphi < \overline{k}_\varphi$ and $k_\psi < \overline{k}_\psi$. Therefore, $k_\varphi + k_\psi < \overline{k}_\varphi + \overline{k}_\psi - 1$. Let us consider a pointed forest $(\mathcal{K}, \mathbf{w})$, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. We have,

$$(\mathcal{K}, \mathbf{w}) \models \Diamond_{\geq k_\varphi + k_\psi} \chi_i \wedge \neg \Diamond_{\geq k_\varphi + k_\psi - 1} \chi_i \text{ iff } \text{card}(\{w' \in R(\mathbf{w}) \mid (\mathcal{K}, \mathbf{w}') \models \chi_i\}) \in [k_\varphi + k_\psi, \overline{k}_\varphi + \overline{k}_\psi - 2].$$

This implies that $\Diamond_{\geq k_\varphi + k_\psi} \chi_i \wedge \neg \Diamond_{\geq k_\varphi + k_\psi - 1} \chi_i$ is equivalent to $\bigvee_{j \in [k_\varphi + k_\psi, \overline{k}_\varphi + \overline{k}_\psi - 2]} \Diamond_{=j} \chi_i$. By completeness of \mathcal{H}_{GML} with respect to GML , we conclude that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} (\Diamond_{\geq k_\varphi + k_\psi} \chi_i \wedge \neg \Diamond_{\geq k_\varphi + k_\psi - 1} \chi_i) \Leftrightarrow \bigvee_{j \in [k_\varphi + k_\psi, \overline{k}_\varphi + \overline{k}_\psi - 2]} \Diamond_{=j} \chi_i.$$

Therefore, in order to conclude that $\text{ATOM}(\chi_i) \Rightarrow \mathbf{U}(\varphi) \parallel \mathbf{U}(\psi)$, it is sufficient to show that for every $j \in [k_\varphi + k_\psi, \overline{k}_\varphi + \overline{k}_\psi - 2]$, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Diamond_{=j} \chi_i \Rightarrow \mathbf{U}(\varphi) \parallel \mathbf{U}(\psi)$. Let $j \in [k_\varphi + k_\psi, \overline{k}_\varphi + \overline{k}_\psi - 2]$. As $k_\varphi < \overline{k}_\varphi$ and $k_\psi < \overline{k}_\psi$, there are $j_1 \in [k_\varphi, \overline{k}_\varphi - 1]$ and $j_2 \in [k_\psi, \overline{k}_\psi - 1]$ such that $j_1 + j_2 = j$. Let us show that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Diamond_{=j_1 + j_2} \chi_i \Rightarrow \Diamond_{=j_1} \chi_i \parallel \Diamond_{=j_2} \chi_i$. To do so, we rely on the theorem $\Diamond_{\geq j_1 + j_2} \chi_i \Rightarrow \Diamond_{\geq j_1} \chi_i \parallel \Diamond_{=j_2} \chi_i$ of $\mathcal{H}_{\text{GML}}(\mathbf{I})$, showed in the first case of the proof (where it was derived for $j_1 = k_\varphi$ and $j_2 = k_\psi$, but its derivation is analogous for all natural numbers).

1	$\Diamond_{=j_1 + j_2} \chi_i \Rightarrow \Diamond_{\geq j_1 + j_2} \chi_i$	PC, def. of $\Diamond_{=j_1 + j_2} \chi$
2	$\Diamond_{\geq j_1 + j_2} \chi_i \Rightarrow \Diamond_{\geq j_1} \chi_i \parallel \Diamond_{=j_2} \chi_i$	See above
3	$\Diamond_{=j_1 + j_2} \chi_i \Rightarrow \Diamond_{\geq j_1} \chi_i \parallel \Diamond_{=j_2} \chi_i$	$(\Rightarrow \text{TR}), 1, 2$
4	$\Diamond_{\geq j_1} \chi_i \Rightarrow (\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1 + 1} \chi_i) \vee (\Diamond_{\geq j_1} \chi_i \wedge \Diamond_{\geq j_1 + 1} \chi_i)$	PC
5	$\Diamond_{\geq j_1} \chi_i \wedge \Diamond_{\geq j_1 + 1} \chi_i \Rightarrow \Diamond_{\geq j_1 + 1} \chi_i$	PC
6	$\Diamond_{\geq j_1} \chi_i \Rightarrow (\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1 + 1} \chi_i) \vee \Diamond_{\geq j_1 + 1} \chi_i$	PC, 5
7	$\Diamond_{\geq j_1} \chi_i \parallel \Diamond_{=j_2} \chi_i \Rightarrow ((\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1 + 1} \chi_i) \vee \Diamond_{\geq j_1 + 1} \chi_i) \parallel \Diamond_{=j_2} \chi_i$	$(\text{C}), 6$
8	$((\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1 + 1} \chi_i) \vee \Diamond_{\geq j_1 + 1} \chi_i) \parallel \Diamond_{=j_2} \chi_i \Rightarrow$ $((\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1 + 1} \chi_i) \parallel \Diamond_{=j_2} \chi_i) \vee (\Diamond_{\geq j_1 + 1} \chi_i \parallel \Diamond_{=j_2} \chi_i)$	(DIST)

9	$\Diamond_{=j_2} \chi_i \Rightarrow \Diamond_{\geq j_2} \chi_i$	PC, def. of $\Diamond_{=j_2} \chi$
10	$\Diamond_{\geq j_1+1} \chi_i \mid \Diamond_{=j_2} \chi_i \Rightarrow \Diamond_{\geq j_2} \chi_i \mid \Diamond_{\geq j_1+1} \chi_i$	(C_{COM}), ($\Rightarrow \text{TR}$), 9
11	$\Diamond_{\geq j_2} \chi_i \mid \Diamond_{\geq j_1+1} \chi_i \Rightarrow \Diamond_{\geq j_1+j_2+1} \chi_i$	($I_{7.15.1}^*$)
12	$\Diamond_{\geq j_1+1} \chi_i \mid \Diamond_{=j_2} \chi_i \Rightarrow \Diamond_{\geq j_1+j_2+1} \chi_i$	($\Rightarrow \text{TR}$), 10, 11
13	$((\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1+1} \chi_i) \vee \Diamond_{\geq j_1+1} \chi_i) \mid \Diamond_{=j_2} \chi_i \Rightarrow$ $((\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1+1} \chi_i) \mid \Diamond_{=j_2} \chi_i) \vee \Diamond_{\geq j_1+j_2+1} \chi_i$	PC, 12
14	$\Diamond_{=j_1+j_2} \chi_i \Rightarrow ((\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1+1} \chi_i) \mid \Diamond_{=j_2} \chi_i) \vee \Diamond_{\geq j_1+j_2+1} \chi_i$	($\Rightarrow \text{TR}$), 3, 7, 13
15	$\Diamond_{=j_1+j_2} \chi_i \Rightarrow \neg \Diamond_{\geq j_1+j_2+1} \chi_i$	PC, def. of $\Diamond_{=j_1+j_2} \chi$
16	$\Diamond_{=j_1+j_2} \chi_i \Rightarrow (\underbrace{\Diamond_{\geq j_1} \chi_i \wedge \neg \Diamond_{\geq j_1+1} \chi_i}_{\Diamond_{=j_1} \chi_i}) \mid \Diamond_{=j_2} \chi_i$	PC, 14, 15

Since $j_1 \in [k_\varphi, \overline{k_\varphi} - 1]$, by propositional reasoning and repeated use of (C_{GRAD}) we derive

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Diamond_{=j_1} \chi_i \Rightarrow \underbrace{\Diamond_{\geq k_\varphi} \chi_i \wedge \neg \Diamond_{\geq \overline{k_\varphi}} \chi_i}_{\mathbf{U}(\varphi)}.$$

Similarly, from $j_2 \in [k_\psi, \overline{k_\psi} - 1]$, we have $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Diamond_{=j_2} \chi_i \Rightarrow \mathbf{U}(\psi)$. By (\mathbf{II}_{LR}), we conclude that $\Diamond_{=j_1} \chi_i \mid \Diamond_{=j_2} \chi_i \Rightarrow \mathbf{U}(\varphi) \mid \mathbf{U}(\psi)$, which together with $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Diamond_{=j} \chi_i \Rightarrow \chi_i \mid \Diamond_{=j_2} \chi_i$ allows us to derive $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Diamond_{=j} \chi_i \Rightarrow \mathbf{U}(\varphi) \mid \mathbf{U}(\psi)$, by ($\Rightarrow \text{TR}$).

This concludes the proof that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ATOM}(\chi_i) \Rightarrow \mathbf{U}(\varphi) \mid \mathbf{U}(\psi)$. Since $\Box \chi_i \subseteq_{\text{LIT}} \text{ATOM}(\chi_i)$, by propositional reasoning together with the axiom ($I_{7.15.3}^*$) we have,

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ATOM}(\chi_i) \Rightarrow (\Box \chi_i \wedge \mathbf{U}(\varphi)) \mid (\Box \chi_i \wedge \mathbf{U}(\psi)). \quad (\chi)$$

To conclude the proof of (\ddagger), it is now sufficient to show $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Box \chi_i \wedge \mathbf{U}(\varphi) \Rightarrow \mathbf{A}_\varphi(\chi_i)$ and $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Box \chi_i \wedge \mathbf{U}(\psi) \Rightarrow \mathbf{A}_\psi(\chi_i)$. Indeed, (\ddagger) then follows by (\mathbf{II}_{LR}) and ($\Rightarrow \text{TR}$), from (χ). We show that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Box \chi_i \wedge \mathbf{U}(\varphi) \Rightarrow \mathbf{A}_\varphi(\chi_i)$ by proving $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Box \chi_i \wedge \mathbf{U}(\varphi) \Rightarrow L$ for every literal L in $\text{LIT}(\mathbf{A}_\varphi(\chi_i))$. We reason by cases on L .

case: $L = \Box \chi_i$. Straightforward.

case: $L = \Diamond_{\geq k} \chi_i$. By definition of $\mathbf{U}(\varphi)$, we have $\Diamond_{\geq k_\varphi} \chi_i \subseteq_{\text{LIT}} \mathbf{U}(\varphi)$, where $k_\varphi \geq k$. Then, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Box \chi_i \wedge \mathbf{U}(\varphi) \Rightarrow L$ follows by propositional reasoning together with repeated application of the axiom (I_1^{GML}).

case: $L = \neg \Diamond_{\geq k} \chi_i$. In this case, by definition of $\mathbf{U}(\varphi)$, $\neg \Diamond_{\geq \overline{k_\varphi}} \chi_i \subseteq_{\text{LIT}} \mathbf{U}(\varphi)$, where $k \geq \overline{k_\varphi}$. Again, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Box \chi_i \wedge \mathbf{U}(\varphi) \Rightarrow L$ follows by propositional reasoning together with repeated applications of the axiom (I_1^{GML}) (more precisely its contrapositive, i.e. $\neg \Diamond_{\geq k} \varphi \Rightarrow \neg \Diamond_{\geq k+1} \varphi$). The proof of $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Box \chi_i \wedge \mathbf{U}(\psi) \Rightarrow \mathbf{A}_\psi(\chi_i)$ is analogous.

Step 3, building φ and ψ . Recapitulating the previous two steps, we first showed that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \text{ATOM}(\chi_1) \mid \dots \mid \text{ATOM}(\chi_n) \mid (\Box \neg \chi_1 \wedge \dots \wedge \Box \neg \chi_n)$ (i.e. (\dagger)) and later discussed how, for all $i \in [1, n]$, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ATOM}(\chi_i) \Rightarrow \mathbf{A}_\varphi(\chi_i) \mid \mathbf{A}_\psi(\chi_i)$ (i.e. (\ddagger)). Together with the commutativity and associativity of the composition operator (axioms (C_{ASSOC}) and (C_{COM})), and the inference rule (C), this allows us to conclude that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow (\mathbf{A}_\varphi(\chi_1) \mid \dots \mid \mathbf{A}_\varphi(\chi_n)) \mid (\mathbf{A}_\psi(\chi_1) \mid \dots \mid \mathbf{A}_\psi(\chi_n)) \mid (\Box \neg \chi_1 \wedge \dots \wedge \Box \neg \chi_n). \quad (\delta)$$

We now complete the derivation of $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \varphi \mid \psi$. First, let us define the formula

$$\text{ALMOST}(\varphi) = \bigwedge \{\Diamond_{\geq k} \chi \mid \Diamond_{\geq k} \chi \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{\neg \Diamond_{\geq k} \chi \mid \neg \Diamond_{\geq k} \chi \subseteq_{\text{LIT}} \varphi\}.$$

Essentially, $\text{ALMOST}(\varphi)$ is the formula obtained from φ by removing every occurrence of atomic propositions non appearing inside a graded modality. We also define $\text{ALMOST}(\psi)$, in a similar way. We show that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow \text{ALMOST}(\varphi).$$

As usual, we prove this result by showing that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow L$ holds for every literal $L \subseteq_{\text{LIT}} \text{ALMOST}(\varphi)$, by cases on L . Yet again, the notion of disjoint formulae is essential.

case: $L = \Diamond_{\geq k} \chi$. By definition of $\text{ALMOST}(\varphi)$ and φ , there is $i \in [1, n]$ such that $\chi = \chi_i$.

Moreover, by definition of $A_\varphi(\chi_i)$, we have $\Diamond_{\geq k} \chi_i \subseteq_{\text{LIT}} A_\varphi(\chi_i)$. Then,

1	$A_\varphi(\chi_i) \Rightarrow \Diamond_{\geq k} \chi_i$	PC, see above
2	$A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow$	
	$A_\varphi(\chi_i) \mid (A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_{i-1}) \mid A_\varphi(\chi_{i+1}) \mid \cdots \mid A_\varphi(\chi_n))$	$(\text{ASSOC}), (\text{COM})$
3	$A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_{i-1}) \mid A_\varphi(\chi_{i+1}) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow \top$	PC
4	$A_\varphi(\chi_i) \mid (A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_{i-1}) \mid A_\varphi(\chi_{i+1}) \mid \cdots \mid A_\varphi(\chi_n)) \Rightarrow \Diamond_{\geq k} \chi_i \mid \top$	(II_{LR})
5	$\Diamond_{\geq k} \chi_i \mid \top \Rightarrow \Diamond_{\geq k} \chi_i$	$(I_{7.15.1}^*), \text{ as } \Diamond_{\geq 0} \chi_i \text{ is } \top$
6	$A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow \Diamond_{\geq k} \chi_i$	$(\Rightarrow \text{TR}), 2, 4, 5$

case: $L = \neg \Diamond_{\geq k} \chi$. Again, by definition of $\text{ALMOST}(\varphi)$ and φ , there is $i \in [1, n]$ such that $\chi = \chi_i$. Moreover, by definition of $A_\varphi(\chi_i)$, we have $\neg \Diamond_{\geq k} \chi_i \subseteq_{\text{LIT}} A_\varphi(\chi_i)$. In this case, we rely on the disjointness of the formulae in Φ . Recall that for every $l \in [1, n]$ different from i , $\chi_i \wedge \chi_l$ is unsatisfiable, and moreover $\Box \chi_l \subseteq_{\text{LIT}} A_\varphi(\chi_l)$. As χ_i and χ_l are formulae in GML , this allows us to derive $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} A_\varphi(\chi_l) \Rightarrow \Box \neg \chi_i$ directly from the completeness of \mathcal{H}_{GML} (Theorem 7.6). Moreover, by definition of \Box together with the axiom $(\neg \text{GRAD})$, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Box \neg \chi_i \mid \Box \neg \chi_i \Rightarrow \Box \neg \chi_i$. This allows us to conclude that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_{i-1}) \mid A_\varphi(\chi_{i+1}) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow \Box \neg \chi_i.$$

The proof proceeds as follows:

1	$A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_{i-1}) \mid A_\varphi(\chi_{i+1}) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow \Box \neg \chi_i$	See above
2	$A_\varphi(\chi_i) \Rightarrow \neg \Diamond_{\geq k} \chi_i$	PC, see above
3	$A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow$	
	$A_\varphi(\chi_i) \mid (A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_{i-1}) \mid A_\varphi(\chi_{i+1}) \mid \cdots \mid A_\varphi(\chi_n))$	$(\text{ASSOC}), (\text{COM})$
4	$A_\varphi(\chi_i) \mid (A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_{i-1}) \mid A_\varphi(\chi_{i+1}) \mid \cdots \mid A_\varphi(\chi_n)) \Rightarrow$	
	$\neg \Diamond_{\geq k} \chi_i \mid \Box \neg \chi_i$	$(\text{II}_{LR}), 1, 2$
5	$\neg \Diamond_{\geq k} \chi_i \mid \Box \neg \chi_i \Rightarrow \neg \Diamond_{\geq k} \chi_i$	$(\neg \text{GRAD})$ PC, def. of $\Box \neg \chi_i$
6	$A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow \neg \Diamond_{\geq k} \chi_i$	$(\Rightarrow \text{TR}), 3, 4, 5$

This concludes the proof of $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow \text{ALMOST}(\varphi)$. In a similar way one can show that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} A_\psi(\chi_1) \mid \cdots \mid A_\psi(\chi_n) \Rightarrow \text{ALMOST}(\psi)$. Thanks to (\mathbf{I}_{LR}) , we conclude that the following formula is derivable within $\mathcal{H}_{\text{GML}}(\mathbf{I})$:

$$(A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n)) \mid (A_\psi(\chi_1) \mid \cdots \mid A_\psi(\chi_n)) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \\ \text{ALMOST}(\varphi) \mid (\text{ALMOST}(\psi) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n))$$

We show that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ALMOST}(\psi) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \text{ALMOST}(\psi)$ which, directly from the formula above and (δ) , allows us to conclude (by propositional reasoning and (\mathbf{I}_{LR})) that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \text{ALMOST}(\varphi) \mid \text{ALMOST}(\psi).$$

The proof of $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ALMOST}(\psi) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \text{ALMOST}(\psi)$ is quite similar to the proof of $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} A_\varphi(\chi_1) \mid \cdots \mid A_\varphi(\chi_n) \Rightarrow \text{ALMOST}(\varphi)$. Given $L \subseteq_{\text{LIT}} \text{ALMOST}(\psi)$, we show that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \text{ALMOST}(\psi) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow L$, by cases on L .

case: $L = \diamondsuit_{\geq k} \chi$. We have,

1	$\text{ALMOST}(\psi) \Rightarrow \diamondsuit_{\geq k} \chi$	PC
2	$\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n \Rightarrow \top$	PC
3	$\text{ALMOST}(\psi) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \diamondsuit_{\geq k} \chi \mid \top$	$(\Rightarrow \text{TR})$, 1, 2
4	$\diamondsuit_{\geq k} \chi \mid \top \Rightarrow \diamondsuit_{\geq k} \chi$	$(I_{7.15.1}^*)$, as $\diamondsuit_{\geq 0} \chi$ is \top
5	$\text{ALMOST}(\psi) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \diamondsuit_{\geq k} \chi$	$(\Rightarrow \text{TR})$, 3, 4

case: $L = \neg \diamondsuit_{\geq k} \chi$. By definition of $\text{ALMOST}(\psi)$ and ψ , there is $i \in [1, n]$ such that $\chi = \chi_i$. Thus, the proof follows thanks to (\neg_{GRAD}^C) , as shown below:

1	$\text{ALMOST}(\psi) \Rightarrow \neg \diamondsuit_{\geq k} \chi_i$	PC
2	$\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n \Rightarrow \square \neg \chi_i$	PC
3	$\text{ALMOST}(\psi) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \neg \diamondsuit_{\geq k} \chi_i \mid \square \neg \chi_i$	$(\Rightarrow \text{TR})$, 1, 2
4	$\neg \diamondsuit_{\geq k} \chi_i \mid \square \neg \chi_i \Rightarrow \neg \diamondsuit_{\geq k} \chi_i$	(\neg_{GRAD}^C) PC, def. of $\square \neg \chi_i$
5	$\text{ALMOST}(\psi) \mid (\square \neg \chi_1 \wedge \cdots \wedge \square \neg \chi_n) \Rightarrow \neg \diamondsuit_{\geq k} \chi_i$	$(\Rightarrow \text{TR})$, 3, 4

We conclude that the formula $\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \text{ALMOST}(\varphi) \mid \text{ALMOST}(\psi)$ is derivable in $\mathcal{H}_{\text{GML}}(\mathbf{I})$.

In view of the definitions of $\text{ALMOST}(\varphi)$ and $\text{ALMOST}(\psi)$, the formulae φ and ψ are respectively equal (up to commutativity and associativity of \wedge) to two formulae $\text{ALMOST}(\varphi) \wedge \varphi_{\text{AP}}$ and $\text{ALMOST}(\psi) \wedge \psi_{\text{AP}}$, where the formulae φ_{AP} and ψ_{AP} are conjunctions of possibly negated atomic propositions. More precisely,

$$\varphi_{\text{AP}} \stackrel{\text{def}}{=} \bigwedge \{p \mid p \subseteq_{\text{LIT}} \varphi\} \wedge \bigwedge \{\neg p \mid \neg p \subseteq_{\text{LIT}} \varphi\},$$

with ψ_{AP} being similarly defined. By definition of $\langle \mathbf{I} \rangle(\varphi, \psi)$ we have $\varphi_{\text{AP}} \wedge \psi_{\text{AP}} \subseteq_{\text{LIT}} \langle \mathbf{I} \rangle(\varphi, \psi)$. Below, let $\varphi^{(1)} = \text{ALMOST}(\varphi)$ and $\psi^{(1)} = \text{ALMOST}(\psi)$. So, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \varphi^{(1)} \mid \psi^{(1)}$. Below, we add to $\varphi^{(1)}$ and $\psi^{(1)}$ every literal from φ_{AP} and ψ_{AP} , respectively. We add the literal progressively, building a sequence of formulae $\varphi^{(1)} \mid \psi^{(1)}, \dots, \varphi^{(k)} \mid \psi^{(k)}$, where for all $j \in [1, k]$ we have $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \varphi^{(j)} \mid \psi^{(j)}$ and for all $i \in [1, j]$, $\varphi^{(i)} \subseteq_{\text{LIT}} \varphi^{(j)}$ and $\psi^{(i)} \subseteq_{\text{LIT}} \psi^{(j)}$. As

we obtain $\varphi^{(k)} = \text{ALMOST}(\varphi) \wedge \varphi_{\text{AP}}$ and $\psi^{(k)} = \text{ALMOST}(\psi) \wedge \psi_{\text{AP}}$, from the associativity and commutativity of \wedge , we derive that

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \varphi \mathbin{\|} \psi$$

concluding the proof. Thus, let us assume that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \varphi^{(j)} \mathbin{\|} \psi^{(j)}$ and consider a literal $L \subseteq_{\text{LIT}} \varphi_{\text{AP}}$ that does not appear in $\varphi^{(j)}$. We show $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow (\varphi^{(j)} \wedge L) \mathbin{\|} \psi^{(j)}$, by case analysis on L .

case: $L = p$. We have $p \subseteq_{\text{LIT}} \langle \mathbf{I} \rangle(\varphi, \psi)$. So,

1	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \varphi^{(j)} \mathbin{\ } \psi^{(j)}$	Hypothesis
2	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow p$	PC, see above
3	$\varphi^{(j)} \Rightarrow (\varphi^{(j)} \wedge p) \vee (\varphi^{(j)} \wedge \neg p)$	PC
4	$\varphi^{(j)} \mathbin{\ } \psi^{(j)} \Rightarrow ((\varphi^{(j)} \wedge p) \vee (\varphi^{(j)} \wedge \neg p)) \mathbin{\ } \psi^{(j)}$	(C), 3
5	$((\varphi^{(j)} \wedge p) \vee (\varphi^{(j)} \wedge \neg p)) \mathbin{\ } \psi^{(j)} \Rightarrow ((\varphi^{(j)} \wedge p) \mathbin{\ } \psi^{(j)}) \vee ((\varphi^{(j)} \wedge \neg p) \mathbin{\ } \psi^{(j)})$	(DIST)
6	$\varphi^{(j)} \mathbin{\ } \psi^{(j)} \Rightarrow ((\varphi^{(j)} \wedge p) \mathbin{\ } \psi^{(j)}) \vee ((\varphi^{(j)} \wedge \neg p) \mathbin{\ } \psi^{(j)})$	($\Rightarrow \text{TR}$), 4, 5
7	$\varphi^{(j)} \wedge \neg p \Rightarrow \neg p$	PC
8	$\psi^{(j)} \Rightarrow \top$	PC
9	$(\varphi^{(j)} \wedge \neg p) \mathbin{\ } \psi^{(j)} \Rightarrow \neg p \mathbin{\ } \top$	(I_{LR}), 7, 8
10	$\neg p \mathbin{\ } \top \Rightarrow \neg p$	(MONO)
11	$(\varphi^{(j)} \wedge \neg p) \mathbin{\ } \psi^{(j)} \Rightarrow \neg p$	($\Rightarrow \text{TR}$), 9, 10
12	$\varphi^{(j)} \mathbin{\ } \psi^{(j)} \Rightarrow ((\varphi^{(j)} \wedge p) \mathbin{\ } \psi^{(j)}) \vee \neg p$	PC, 6, 11
13	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow (\varphi^{(j)} \mathbin{\ } \psi^{(j)}) \wedge p$	PC, 1, 2
14	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow (\varphi^{(j)} \wedge p) \mathbin{\ } \psi^{(j)}$	PC, 12, 13

case: $L = \neg p$. We have $\neg p \subseteq_{\text{LIT}} \langle \mathbf{I} \rangle(\varphi, \psi)$. This case is analogous to the previous one, by swapping p and $\neg p$.

1	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \varphi^{(j)} \mathbin{\ } \psi^{(j)}$	Hypothesis
2	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \neg p$	PC, see above
3	$\varphi^{(j)} \mathbin{\ } \psi^{(j)} \Rightarrow ((\varphi^{(j)} \wedge p) \mathbin{\ } \psi^{(j)}) \vee ((\varphi^{(j)} \wedge \neg p) \mathbin{\ } \psi^{(j)})$	Previous case of the proof, line 6
4	$\varphi^{(j)} \wedge p \Rightarrow p$	PC
5	$\psi^{(j)} \Rightarrow \top$	PC
6	$(\varphi^{(j)} \wedge p) \mathbin{\ } \psi^{(j)} \Rightarrow p \mathbin{\ } \top$	(I_{LR}), 4, 5
7	$p \mathbin{\ } \top \Rightarrow p$	(MONO)
8	$(\varphi^{(j)} \wedge p) \mathbin{\ } \psi^{(j)} \Rightarrow p$	($\Rightarrow \text{TR}$), 6, 7
9	$\varphi^{(j)} \mathbin{\ } \psi^{(j)} \Rightarrow p \vee ((\varphi^{(j)} \wedge \neg p) \mathbin{\ } \psi^{(j)})$	PC, 3, 8
10	$\langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow (\varphi^{(j)} \mathbin{\ } \psi^{(j)}) \wedge \neg p$	PC, 1, 2

$$11 \quad \left| \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow (\varphi^{(j)} \wedge \neg p) \mid \psi^{(j)} \right. \quad \text{PC, 9, 10}$$

Thanks to the commutativity of the composition operator, with analogous derivations one can show that for every $L \subseteq_{\text{LIT}} \psi_{\text{AP}}$ not appearing in $\psi^{(j)}$, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \langle \mathbf{I} \rangle(\varphi, \psi) \Rightarrow \varphi^{(j)} \mid (\psi^{(j)} \wedge L)$. \square

We are now ready to state the adequateness of $\mathcal{H}_{\text{GML}}(\mathbf{I})$.

Theorem 7.16 (Adequacy). A formula φ in $\text{ML}(\mathbf{I})$ is valid if and only if $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi$.

To prove this theorem, we first extend Lemma 7.14 to Boolean combinations of core formulae.

Lemma 7.17. Let φ and ψ be two GML formulae with $\text{top}_{\text{gm}}(\varphi) = \{\Diamond_{\geq j_1} \varphi_1, \dots, \Diamond_{\geq j_n} \varphi_n\}$, $\text{top}_{\text{gm}}(\psi) = \{\Diamond_{\geq k_1} \psi_1, \dots, \Diamond_{\geq k_m} \psi_m\}$ and where $\Phi = \{\varphi_1, \dots, \varphi_n\} \cup \{\psi_1, \dots, \psi_m\}$ is a set of disjoint formulae. We have $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \mid \psi \Leftrightarrow \chi$, where χ is either \perp or a disjunction of formulae from $\text{Conj}(\text{Core}(\Phi, \max(j_1, \dots, j_n) + \max(k_1, \dots, k_n), \text{top}_{\text{AP}}(\varphi) \cup \text{top}_{\text{AP}}(\psi)))$.

Proof. If either φ or ψ are unsatisfiable, then $\vdash_{\mathcal{H}_C} \varphi \mid \psi \Leftrightarrow \perp$. Indeed, suppose φ unsatisfiable (the case where ψ is unsatisfiable is analogous). From the completeness of \mathcal{H}_{GML} , we have $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \Rightarrow \perp$. We show that $\vdash_{\mathcal{H}_C} \varphi \mid \psi \Rightarrow \perp$ (the right to left direction is straightforward, by propositional reasoning).

$$\begin{array}{llll} 1 \quad \left| \varphi \Rightarrow \perp \right. & \text{See above} & 3 \quad \left| \perp \mid \psi \Rightarrow \perp \right. & (\text{C}_{\text{ZERO}}) \\ 2 \quad \left| \psi \mid \psi \Rightarrow \perp \mid \psi \right. & (\text{C}), 1 & 4 \quad \left| \varphi \mid \psi \Rightarrow \perp \right. & (\Rightarrow \text{TR}), 2, 3 \end{array}$$

Otherwise, let us assume φ and ψ satisfiable. By proposition 7.8, φ is equivalent to a formula $\bigvee_{i \in I_1} \varphi^{(i)}$ where all $\varphi^{(i)}$ belong to $\text{Conj}(\text{Core}(\{\varphi_1, \dots, \varphi_n\}, \max(j_1, \dots, j_n), \text{top}_{\text{AP}}(\varphi)))$. Similarly, $\psi \equiv \bigvee_{i \in I_2} \psi^{(i)}$ where all $\psi^{(i)}$ are in $\text{Conj}(\text{Core}(\{\psi_1, \dots, \psi_m\}, \max(k_1, \dots, k_m), \text{top}_{\text{AP}}(\psi)))$. By completeness of \mathcal{H}_{GML} , $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \Leftrightarrow \bigvee_{i \in I_1} \varphi^{(i)}$ and $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \psi \Leftrightarrow \bigvee_{i \in I_2} \psi^{(i)}$. Thanks to the axioms (DIST) and (COM) , we can distribute \mid over disjunctions, leading to

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \mid \psi \Leftrightarrow \bigvee_{i_1 \in I_1, i_2 \in I_2} (\varphi^{(i_1)} \mid \psi^{(i_2)}).$$

As Φ is a set of disjoint formulae that includes $\varphi_1, \dots, \varphi_n$ and ψ_1, \dots, ψ_m , given $i_1 \in I_1$ and $i_2 \in I_2$, we can apply Lemma 7.14 to conclude that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi^{(i_1)} \mid \psi^{(i_2)} \Leftrightarrow \langle \mathbf{I} \rangle(\varphi^{(i_1)}, \psi^{(i_2)})$. By propositional reasoning, we conclude: $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \mid \psi \Leftrightarrow \bigvee_{i_1 \in I_1, i_2 \in I_2} \langle \mathbf{I} \rangle(\varphi^{(i_1)}, \psi^{(i_2)})$. \square

Proof of Theorem 7.16. The soundness of $\mathcal{H}_{\text{GML}}(\mathbf{I})$ as already been established in Lemma 7.13. As far as the completeness proof is concerned, we need to show that for every formula φ in $\text{ML}(\mathbf{I})$ there is a formula ψ in GML such that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \Leftrightarrow \psi$. This is enough to conclude the proof: if φ is valid then from the soundness of $\mathcal{H}_{\text{GML}}(\mathbf{I})$ we obtain that ψ is valid. Since $\mathcal{H}_{\text{GML}}(\mathbf{I})$ extends \mathcal{H}_{GML} , which is complete for GML, we have $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \psi$. By propositional reasoning, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi$.

To show that every formula φ has a provably equivalent formula in GML, we first notice that the substitution of equivalent formulae holds true in $\mathcal{H}_{\text{GML}}(\mathbf{I})$, i.e. the rule below is admissible:

$$(S_{\mid}) \quad \frac{\psi \Leftrightarrow \chi}{\varphi[\psi]_{\rho} \Leftrightarrow \varphi[\chi]_{\rho}}$$

We already saw how to derive similar rules during Chapter 6 (e.g. at the end of both Sections 6.5 and 6.6). Briefly, the proof of admissibility carries out with a simple structural induction on φ , relying on the following rules:

$$\frac{\varphi \Leftrightarrow \psi}{\neg\varphi \Leftrightarrow \neg\psi} \quad \frac{\varphi \Leftrightarrow \psi}{(\varphi \Rightarrow \chi) \Leftrightarrow (\psi \Rightarrow \chi)} \quad \frac{\varphi \Leftrightarrow \psi}{(\chi \Rightarrow \varphi) \Leftrightarrow (\chi \Rightarrow \psi)} \quad \frac{\varphi \Leftrightarrow \psi}{\Diamond_{\geq k}\varphi \Leftrightarrow \Diamond_{\geq k}\psi} \quad \frac{\varphi \Leftrightarrow \psi}{\varphi | \chi \Leftrightarrow \psi | \chi}.$$

The first three rules are from propositional reasoning, whereas the third one is from graded modal logic (see rule (G), page 348). The last rule is admissible in $\mathcal{H}_{\text{GML}}(\mathbf{I})$ thanks to (C). The details of the proof of (S_I) are left to the reader (hint: see the proof of (S_{*}), Theorem 6.16).

Exactly as done in the proof of Theorem 6.16, the rule (S_I) allows us to prove that φ is equivalent to a formula in GML by induction on the number of occurrences of the separating conjunction φ that are not involved in the definition of a graded modality $\Diamond_{\geq k}$ (we recall that $\Diamond_{\geq k}\psi$ is a shortcut for $\Diamond\psi| \cdots | \Diamond\psi$, where $|$ repeats $k - 1$ times).

base case: φ without occurrences of $|$ (excluding those used to define $\Diamond_{\geq k}$).

In this case, φ is already in GML.

induction step: φ with $a > 1$ occurrences of $|$ (excluding those used for $\Diamond_{\geq k}$).

Let $\varphi_1 | \varphi_2$ be a subformula of φ , say at position ρ , such that φ_1 and φ_2 are formulae in GML. Let $\text{top}_{\text{gm}}(\varphi_1) = \{\Diamond_{\geq j_1}\chi_1, \dots, \Diamond_{\geq j_n}\chi_n\}$ and $\text{top}_{\text{gm}}(\varphi_2) = \{\Diamond_{\geq k_1}\chi'_1, \dots, \Diamond_{\geq k_m}\chi'_m\}$. Let $\Phi = \{\chi_1, \dots, \chi_n\} \cup \{\chi'_1, \dots, \chi'_m\}$. All the formulae in $\text{top}_{\text{gm}}(\varphi_1) \cup \text{top}_{\text{gm}}(\varphi_2)$ belong to the set $\text{Core}(\Phi, \max(j_1, \dots, j_n, k_1, \dots, k_m), \emptyset)$, and thus by Lemma 7.11 they are equivalent to a disjunction of formulae in $\text{Conj}(\text{Core}(2^\Phi, \max(j_1, \dots, j_n, k_1, \dots, k_m), \emptyset))$. Thanks to (S_I), we can derive $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi_1 \Leftrightarrow \varphi'_1$ where φ'_1 is the formula obtained from φ_1 by substituting every occurrence of formulae in $\text{top}_{\text{gm}}(\varphi_1)$ not appearing under the scope of a graded modality with the equivalent disjunction of formulae from $\text{Conj}(\text{Core}(2^\Phi, \max(j_1, \dots, j_n, k_1, \dots, k_m), \emptyset))$. In particular, $\text{top}_{\text{gm}}(\varphi'_1)$ is a subset of $\text{Core}(2^\Phi, \max(j_1, \dots, j_n, k_1, \dots, k_m), \emptyset)$. Similarly, we derive $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi_2 \Leftrightarrow \varphi'_2$ where φ'_2 is a formula in GML such that $\text{top}_{\text{gm}}(\varphi'_2)$ is a subset of $\text{Core}(2^\Phi, \max(j_1, \dots, j_n, k_1, \dots, k_m), \emptyset)$. By (S_I), we have $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi_1 | \varphi_2 \Leftrightarrow \varphi'_1 | \varphi'_2$. Fundamentally, since 2^Φ is a set of disjoint formulae (Lemma 7.10) and $\text{top}_{\text{gm}}(\varphi'_1) \cup \text{top}_{\text{gm}}(\varphi'_2) \subseteq \text{Core}(2^\Phi, \max(j_1, \dots, j_n, k_1, \dots, k_m), \emptyset)$, we can apply Lemma 7.17. We conclude that $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi'_1 | \varphi'_2 \Leftrightarrow \chi$ for some formula χ in GML. By propositional reasoning, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi_1 | \varphi_2 \Leftrightarrow \chi$. Recall that $\varphi_1 | \varphi_2$ is the subformula of φ occurring at position ρ . Thanks to the substitution rule (S_I), we derive:

$$\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \Leftrightarrow \varphi[\chi]_\rho.$$

The right-hand side of the double implication above has $a - 1$ occurrences of the composition operator that are not involved in the definition of graded modalities. The induction hypothesis applies, allowing us to derive $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi[\chi]_\rho \Leftrightarrow \gamma$, where γ is a GML formula. By propositional reasoning, $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \Leftrightarrow \gamma$. \square

As a by-product of the transformation into GML formulae we have just carried out, we conclude that $\text{ML}(\mathbf{I})$ and GML have the same expressive power, as already stated in Claim 7.5.

Corollary 7.18. $\text{ML}(\mathbf{I})$ is as expressive as GML.

Conclusion

In Chapters 6 and 7, we presented a method to axiomatise internally spatial logics, and applied this method to derive Hilbert-style proof systems for $\text{SL}(*, \neg*)$ and $\text{ML}(\mathbf{I})$. The central object that allows us to show the completeness of our proof system is given by the notion of core formulae, introduced in Chapter 5. Obviously, the internal proof systems presented here are of theoretical interest, at least to grasp what are the essential features of $\text{SL}(*, \neg*)$ and $\text{ML}(\mathbf{I})$. Hilbert-style proof systems are known to offering very little guidance on which axiom to choose and which substitution to apply during the proof-search process, making them hard to automate. However, our hope with the proof system of $\text{SL}(*, \neg*)$ is to give new insights on the logic, that could in the future be applied to proof calculi that are more geared for automation.

To provide further evidence that our method based on core formulae is robust, it is desirable to apply it to axiomatise other spatial logics, for instance by considering separation logics featuring the list segment predicate \mathbf{ls} or first-order quantification. A key step in our approach is first to show that the logic admits a characterisation in terms of core formulae and such formulae need to be designed adequately. Of course, it is required that the set of valid formulae is recursively enumerable, which discards any attempt with $\text{SL}(*, \neg*, \mathbf{ls})$ or $\text{SL}(\exists, *, \neg*)$ (see Theorem 2.13 and Corollary 3.19). To this end, the second part of the paper [58], co-authored with S. Demri and E. Lozes, introduces an extension of $\text{SL}(*, \rightarrow^+)$ featuring a guarded form of quantification, and axiomatise it by relying on the core formulae technique. The paper [57], co-authored with S. Demri and R. Fervari, applies the method to axiomatise modal separation logics.

More separation logics could be axiomatised by relying on the core formulae approach, other good candidates being the one-quantified variable extension of $\text{SL}(*, \neg*)$ [55] and the Bernays-Schönfinkel-Ramsey class of separation logic [63]. On the ambient logic side, it is known from [104] that the guarantee operator \triangleright , i.e. the right-adjoint of the composition operator \mathbf{I} , does not increase the expressive power of the logic. We conjecture the same to be true for $\text{ML}(\mathbf{I}, \triangleright)$, i.e. $\text{ML}(\mathbf{I})$ enriched with \triangleright , which should easily lead to a proof system for $\text{ML}(\mathbf{I}, \triangleright)$, again using graded modal logic as a family of core formulae.

Other proof systems for $\text{SL}(*, \neg*)$ and ambient logic.

Surprisingly, as far as we know, sound and complete proof systems for $\text{SL}(*, \neg*)$ are very rare and the only system we are aware of is a tableaux-based calculus from [74] with labelled formulae (each formula is enriched with a label to be interpreted by some heap) and with resource graphs to encode symbolically constraints between heap expressions (i.e. labels). Of course, translations from separation logics into logics or theories have been designed, see e.g. [35, 123], but the finding of proof systems for $\text{SL}(*, \neg*)$ with all Boolean connectives and the separating connectives $*$ and $\neg*$ has been quite challenging. Unlike [74], our proof system uses only $\text{SL}(*, \neg*)$ formulae and

therefore can be viewed as a quite orthodox Hilbert-style calculus with no extra syntactic objects. In particular, $\mathcal{H}_C(*, \neg*)$ has no syntactic machinery to refer to heaps or to other semantical objects related to $\mathbf{SL}(*, \neg*)$. In [74], the resource graphs attached to the tableaux are designed to reason about heap constraints, and to provide control for designing strategies that lead to termination. Interestingly, the calculus in [74] is intended to be helpful to synthesize counter-models (which is a standard feature for labelled deduction systems [72]) or to be extended to the first-order case, which is partly done in [74] but we know that completeness is theoretically impossible. Besides, a sound labelled sequent calculus for the first-order extension of $\mathbf{SL}(*, \neg*)$ is presented in [90] but completeness for the sublogic $\mathbf{SL}(*, \neg*)$ is not established. The calculus in [90] has also labels, which differs from our puristic approach. A complete sequent-style calculus for the symbolic heap fragment has been designed quite early in [10] but does not deal with full $\mathbf{SL}(*, \neg*)$ (in particular it is not closed under Boolean connectives and does not contain the separating implication). A complexity-wise optimal decision procedure for the symbolic heap fragment is designed in [44] based on a characterisation in terms of homomorphisms.

Compared to separation logic, ambient logics received less attention when it comes to proof systems, as numerous of its interesting components, from ambient names restrictions to temporal operators, make the logic non-recursively enumerable. When these components are removed from the logic, the only proof system we know of is the sequent calculus from [34]. However, in view of the differences between the logic in [34] and $\mathbf{ML}(\mathsf{I})$, it is not possible to carry out a proper comparison between this system and the axiomatisation defined in Chapter 7.

Axiomatising knowledge logics with reduction axioms.

Let us recall that the proof systems of $\mathbf{SL}(*, \neg*)$ and $\mathbf{ML}(\mathsf{I})$ essentially simulate a bottom-up elimination of the multiplicative connectives ($*$ and $\neg*$ for $\mathbf{SL}(*, \neg*)$, I for $\mathbf{ML}(\mathsf{I})$), leading to Boolean combinations of core formulae for which the proof systems are also complete. As the core formulae are (simple) formulae in $\mathbf{SL}(*, \neg*)/\mathbf{ML}(\mathsf{I})$, the axiomatisations use only formulae of the respective logic. As a by-product of the completeness proofs, we get expressive completeness of the two logics with respect to the corresponding Boolean combinations of core formulae. In particular, we obtain that $\mathbf{ML}(\mathsf{I})$ is as expressive as graded modal logic (Corollary 7.18).

This general principle described above is similar to the one used for axiomatising dynamic epistemic logics, in which dynamic connectives might be eliminated with the help of so-called *reduction axioms*, see e.g. standard examples in [139, 138, 143, 68]. In a nutshell, every formula containing a dynamic operator is provably reduced to a formula without such an operator. Completeness is then established thanks to the completeness of the underlying ‘basic’ language. A similar approach for the linear μ -calculus is recently presented in [61] for which a form of constructive completeness is advocated, see also [105].

Part III

Mixing Multiplicative Connectives and Modalities

Two Ways to Chop a Tree

During the second part of the thesis (Chapters 6 and 7), we observed interesting connections between separation logics and ambient logics, which we made explicit through the Hilbert-style proof systems of $\text{SL}(*, -*)$ and $\text{ML}(\mathsf{I})$. Despite the novelty of the two proof systems, connections between separation logics and ambient logics were already established in the early 2000s, when the two logics were firstly introduced. For instance, the decidability of the static ambient logic SAL considered in [34] is based on the proof technique firstly used by C. Calcagno, H. Yang and P. W. O’Hearn in order to show the decidability of $\text{SL}(*, -*)$ [33]. Similarly, in [104] E. Lozes relies on the core formulae technique in order to study the expressiveness and minimality of both separation logics and ambient logics.

Even though the aforementioned works rely on the same proof techniques to tackle problems for both separation logics and ambient logics, a direct comparison between these logics is missing or limited to a superficial analysis on the different classes of models and spatial connectives. In this regard, we recall that the composition operator I in ambient logics decomposes a tree into two disjoint pieces such that once a node has been assigned to one submodel, all its descendants belong to the same submodel. Instead, the separating conjunction $*$ from separation logics simply decomposes the structures into two disjoint pieces, with no additional constraints.

Concerning their computational complexity, these types of logics can often be encoded in monadic second-order logic MSO interpreted on tree-like structures (see Section 2.2), leading to decidability by Rabin’s theorem [122]. However, most likely, this does not produce the best decision procedures when it comes to solving simple reasoning tasks (recall that the satisfiability problem of MSO interpreted on trees is TOWER-complete [128]). Thus, relying on MSO as a common umbrella to capture and understand the differences between those logical formalisms is often not satisfactory. Of course, we should also not forget that these logics instantiate the framework of BBI (Section 2.3.3), but following this direction will most likely lead to a limited comparison, bounded to the abstract level of BBI . All things considered, no uniform framework investigates exhaustively the relationships between ambient logics and separation logics.

Motivations.

In Chapters 8 and 9, we aim for an in-depth comparison between the composition operator I from static ambient logic and the separating conjunction $*$ from separation logics by identifying a common ground in terms of logical languages and models. As a consequence, we are able to study the effects of having these operators as far as expressivity and complexity are concerned. To carry out our comparison, we consider $\text{ML}(\mathsf{I})$ as an ambient logic and introduce the modal logic $\text{ML}(*)$ which is obtained from $\text{ML}(\mathsf{I})$ by replacing the composition operator with the separating conjunction. This framework is sufficiently fundamental to give us the possibility to take

advantage of model theoretical tools from modal logics [51, 15, 66], and sets a common ground for comparison that may lead to further connections with other logics.

Contribution of Chapter 8.

We continue the analysis of $\text{ML}(\|)$ started in Chapter 7, by looking at the computational complexity of its satisfiability problem. We show that this problem is AEXP_{POL} -complete, where AEXP_{POL} is the class of decision problems solvable by an alternating Turing machine with exponential runtime and polynomial number of alternations between existential and universal states. The AEXP_{POL} -hardness is shown by reduction from the satisfiability problem of propositional team logic [85], another logic instantiating the framework of BBI. To solve the satisfiability problem of $\text{ML}(\|)$ in AEXP_{POL} , the essential step is to show that $\text{ML}(\|)$ enjoys an exponential-size small model property. This is done thanks to a refined translation from $\text{ML}(\|)$ to graded modal logic (GML), which we remind the reader being as expressive as $\text{ML}(\|)$ (Chapter 7). Afterwards, the AEXP_{POL} -hardness follows as we show that the semantics of $\text{ML}(\|)$ can be internalised in the second-order modal logic QK. Lastly, we are able to relate $\text{ML}(\|)$ to the intensional fragment of static ambient logic SAL($\|$) from [34] by providing polynomial-time reductions between their satisfiability problems. Consequently, we establish AEXP_{POL} -completeness of SAL($\|$) and the AEXP_{POL} -hardness of SAL, refuting hints from [34, Section 6].

Contribution of Chapter 9.

We introduce $\text{ML}(*)$, the logic obtained from $\text{ML}(\|)$ by replacing the composition operator $\|$ by the separating conjunction $*$. As done for $\text{ML}(\|)$, we interpret $\text{ML}(*)$ on Kripke-style finite forests, which allows us to compare the two logics in terms of expressive power and complexity, obtaining surprising results.

We show that $\text{ML}(*)$ is strictly less expressive than $\text{ML}(\|)$ and GML. Interestingly, this development partially reuses the result for $\text{ML}(\|)$, hence showing that our framework allows us to transpose results between the two logics. To show that GML is strictly more expressive than $\text{ML}(*)$, we define an ad-hoc notion of Ehrenfeucht-Fraïssé games for the logic. Very surprisingly, although $\text{ML}(*)$ is strictly less expressive than $\text{ML}(\|)$, its complexity is much higher (not even elementary). More precisely, we show that the satisfiability problem for $\text{ML}(*)$ is TOWER-complete. The TOWER upper bound is a consequence of Rabin's theorem [122], whereas hardness is shown by reduction from a TOWER-complete tiling problem, adapting substantially the TOWER-hardness proof from [8] for the second-order modal logic on finite trees QK^t. Lastly, we relate $\text{ML}(*)$ with the modal separation logic MSL(*, \Diamond^{-1}) from [54], and conclude that the satisfiability problem of the latter logic is TOWER-complete (the previous best lower bound was PSPACE, from [54]).

8

The Complexity of the Modal Logic $\text{ML}(\Box)$

Contents

8.1	$\text{ML}(\Box)$ and GML as fragments of second-order ML	377
8.1.1	Second-order modal logic.	377
8.1.2	On alternating time.	379
8.2	Checking satisfiability for $\text{ML}(\Box)$, in AEXP_{POL}	380
8.2.1	GML formulae in good shape.	381
8.2.2	The exponential-size small model property of $\text{ML}(\Box)$.	383
8.3	$\text{ML}(\Box)$ is AEXP_{POL} -complete	390
8.3.1	Propositional logic in team semantics.	390
8.3.2	From Propositional Team Logic to $\text{ML}(\Box)$.	391
8.4	An AEXP_{POL} -complete Static Ambient Logic	397
8.4.1	From $\text{SAL}(\Box)$ to $\text{ML}(\Box)$.	399
8.4.2	From $\text{ML}(\Box)$ to $\text{SAL}(\Box)$.	400

In this chapter

We conclude the study of $\text{ML}(\mathbf{I})$ started in Chapter 7 and show that its satisfiability problem is complete for AEXP_{POL} , an alternating complexity class that sits between NEXPTIME and EXPSPACE . The upper bound is based on an exponential-size small model property, whereas the AEXP_{POL} -hardness follows by reduction from the satisfiability problem of propositional logic in team semantics: another logic instantiating the framework of BBI.

The chapter ends with a brief section that formalises the connections between $\text{ML}(\mathbf{I})$ and ambient logic, transferring complexity results to the latter logic.

Here is a roadmap of the chapter.

Section 8.1. Roughly speaking, the algorithm that checks for the satisfiability of a formula φ in $\text{ML}(\mathbf{I})$ works by guessing a pointed forest $(\mathcal{K}, \mathbf{w})$ of size bounded by $\mathcal{B}(|\varphi|)$ for some function \mathcal{B} , and applying a model-checking procedure on inputs $(\mathcal{K}, \mathbf{w})$ and φ . With this in mind, the chapter starts by looking at the model checking problem for $\text{ML}(\mathbf{I})$. We show that $\text{ML}(\mathbf{I})$ is a fragment of the second-order modal logic QK, which in turn is known to be a fragment of monadic second-order logic (MSO). We recall the complexity of the model checking problem for MSO with respect to $\text{ATIME}(\mathbf{f}(n), \mathbf{g}(m))$, i.e. the class of languages accepted by an alternating Turing machine with at most $\mathbf{g}(m)$ alternations and runtime at most $\mathbf{f}(n)$.

Proposition 8.4 (From [132]). There are polynomials \mathbf{f} and \mathbf{g} such that the model checking problem of MSO (alternatively, QK or $\text{ML}(\mathbf{I})$) is in $\text{ATIME}(\mathbf{f}(n), \mathbf{g}(m))$, where n is the input size and m is the formula size.

The complexity class AEXP_{POL} is defined as the union of all $\text{ATIME}(\mathbf{f}(n), \mathbf{g}(n))$ (where n is the size of the input), over all exponential functions \mathbf{f} and polynomial functions \mathbf{g} . Following Proposition 8.4, to prove that the satisfiability problem of $\text{ML}(\mathbf{I})$ is decidable in AEXP_{POL} it is sufficient to show that the function \mathcal{B} is at most exponential in $|\varphi|$.

Section 8.2. We show that $\text{ML}(\mathbf{I})$ has the aforementioned exponential-size small model property. The proof relies on the definition of a normal form for GML formulae (Definition 8.5). Fundamentally, the size of the smallest model satisfying a formula φ in normal form is bounded by the *branching degree* of φ (Definition 8.6), which depends on the coefficients k appearing in the graded modalities $\Diamond_{\geq k}$. We rely on properties of the axiom system $\mathcal{H}_{\text{GML}}(\mathbf{I})$ introduced in Chapter 7 to show that every formula φ of $\text{ML}(\mathbf{I})$ can be translated into a GML formula ψ in normal form. The branching degree of ψ is exponential in the size of φ , allowing us to conclude.

Lemma 8.10. There is a polynomial \mathcal{Q} such that every satisfiable φ in $\text{ML}(\mathbf{I})$ is satisfied by a pointed forest of size bounded by $2^{\mathcal{Q}(|\varphi|)}$.

Section 8.3. We show that the satisfiability problem of $\text{ML}(\mathbf{I})$ is AEXP_{POL} -hard, by reduction from the satisfiability problem of propositional logic interpreted in team semantics [85] ($\text{PL}(\sim)$). A *team* is a set of Boolean valuations $\mathbf{v} : P \rightarrow \{\top, \perp\}$, where $P \subseteq_{\text{fin}} \text{AP}$. Under team semantics, propositional formulae are interpreted on a team \mathfrak{T} , and $\mathfrak{T} \models \varphi \vee \psi$ is satisfied whenever \mathfrak{T} can be partitioned into two disjoint sets respectively satisfying φ and ψ . As such, $\text{PL}(\sim)$ can be seen as an instantiation of BBI. The reduction from $\text{PL}(\sim)$ to $\text{ML}(\mathbf{I})$ shows that the former logic corresponds to the restriction of $\text{ML}(\mathbf{I})$ to formulae of modal depth 1.

Theorem 8.17. The satisfiability problem for $\text{ML}(\mathbf{I})$ is AEXP_{POL} -complete. It is already AEXP_{POL} -hard for the fragment of $\text{ML}(\mathbf{I})$ formulae with modal depth at most 1.

Section 8.4. Section 8.3 concludes the study of $\text{ML}(\mathbf{I})$. We end the chapter by going back to ambient logic, and design semantically faithful reductions (in both directions) between the satisfiability problem for $\text{ML}(\mathbf{I})$ and the one for $\text{SAL}(\mathbf{I})$, i.e. the intensional fragment of static ambient logic. This implies the AEXP_{POL} -completeness of the satisfiability problem for $\text{SAL}(\mathbf{I})$, instead of the conjectured PSPACE -completeness [34].

8.1 $\text{ML}(\mathbf{|})$ AND GML AS FRAGMENTS OF SECOND-ORDER ML

Chapter 7 introduces the modal logic $\text{ML}(\mathbf{|})$, that extends the modal logic ML with the composition operator $\mathbf{|}$ from ambient logic. The logic is interpreted on Kripke-style finite forests (Definition 7.1) and it is as expressive as graded modal logic (GML , see Corollary 7.18).

The objective of this chapter is quite simple: we continue the study of $\text{ML}(\mathbf{|})$, as well as its comparison with GML , by looking at the computational complexity of its satisfiability problem. While we refer the reader to Chapter 7 for the proper definitions of $\text{ML}(\mathbf{|})$ and GML , we recall below the grammar of the two logics.

$\text{ML}(\mathbf{ })$:	GML :
$\varphi ::= \top \mid p \mid \varphi \Rightarrow \varphi \mid \neg \varphi \mid \Diamond \varphi \mid \varphi \mathbf{ } \varphi,$	$\varphi ::= \top \mid p \mid \varphi \Rightarrow \varphi \mid \neg \varphi \mid \Diamond_{\geq k} \varphi,$

where $k \in \mathbb{N} \setminus \{0\}$ and p is an atomic proposition taken from a countably infinite set AP .

On standard Kripke structures (Definition 4.34), it is well known from [133] that the satisfiability problem of GML is PSPACE-complete. Since GML admits a finite tree model properties [51], this complexity result transfers to the class of Kripke-style finite forests. Moreover, the PSPACE-completeness of GML holds regardless of whether the coefficients k of graded modalities $\Diamond_{\geq k}$ are encoded in unary or binary. For simplicity, during the chapter we assume the former encoding.

The main result of the chapter is given by an exponential-size small model property for $\text{ML}(\mathbf{|})$, i.e. every satisfiable formula φ is satisfied by a pointed forest of size exponential in the size of φ . Thanks to this result, we show that the satisfiability problem of $\text{ML}(\mathbf{|})$ belongs to the complexity class AEXP_{POL} , which is the class of decision problems solvable by an alternating Turing machine running in exponential time and alternating between existential and universal states a polynomial amount of times, with respect to the size of the input. In terms of complexity classes for (non)deterministic machines, AEXP_{POL} includes NEXPTIME , and it is included in EXPSPACE . AEXP_{POL} captures the precise complexity of several natural decision problems, such as the satisfiability problems of the first-order theory of real addition with order [67], of one-agent refinement modal logic [20], of quantified computation tree logic interpreted on trees with fixed branching degree [8], of propositional logic in team semantics [85]. In our case, we are particularly interested in the latter result, as it will give us a direct way of proving that the satisfiability problem of $\text{ML}(\mathbf{|})$ is in fact complete for AEXP_{POL} (Section 8.3).

8.1.1 Second-order modal logic.

In order to analyse the complexity of $\text{ML}(\mathbf{|})$, it is first helpful to frame the logic in terms of second-order theories. Both $\text{ML}(\mathbf{|})$ and GML can be seen as syntactical fragments of second-order modal logic QK , which is the extension of the standard modal logic ML with second-order propositional quantification. The syntax of the formulae in QK is as follows:

$$\begin{array}{lll} \pi ::= & \top \quad (\text{true}) & \varphi ::= \pi \quad (\text{atomic formulae}) \\ & \mid p \quad (\text{propositional symbol}) & \mid \varphi \Rightarrow \varphi \mid \neg \varphi \quad (\text{Boolean connectives}) \\ & & \mid \Diamond \varphi \quad (\text{modality of possibility}) \\ & & \mid \exists p \varphi \quad (\text{propositional quantification}) \end{array}$$

We have already introduced propositional quantifiers in Section 4.4.2, where we studied the relationship between ALT and quantified computation tree logic (QCTL). In fact, QK can be seen

$$\begin{aligned}
 (\mathcal{K}, w) \models p &\quad \text{iff } w \in \mathcal{V}(p), \\
 (\mathcal{K}, w) \models \Diamond\varphi &\quad \text{iff } \text{there is } w' \in R(w) \text{ such that } (\mathcal{K}, w') \models \varphi, \\
 (\mathcal{K}, w) \models \exists p \varphi &\quad \text{iff } \text{there is } \mathcal{W}' \subseteq \mathcal{W} \text{ such that } (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W}']) \models \varphi.
 \end{aligned}$$

Figure 8.1: Satisfaction relation for QK.

as the fragment of QCTL restricted to the temporal modality $\text{EX } \varphi$, which corresponds to $\Diamond\varphi$. By Rabin’s theorem [122] this means that, while the satisfiability problem of QK is undecidable when interpreted on Kripke structures [69], it becomes decidable on functional structures such as the class of Kripke-style finite forests [122]. Given a Kripke-style finite forest (\mathcal{K}, w) (alternatively, a Kripke structure), the satisfaction relation \models for formulae in QK is defined in Figure 8.1 (omitting standard cases for \top and Boolean connectives). Exactly as in QCTL, we recall that in the semantics of the propositional quantification $\exists p \varphi$, $\mathcal{V}[p \leftarrow \mathcal{W}']$ stands for the valuation obtained from \mathcal{V} by updating the set of worlds satisfying p , from $\mathcal{V}(p)$ to \mathcal{W}' .

To characterise $\text{ML}(\mathbf{I})$ as a fragment of QK, we rely on the second-order quantification $\exists p$ in order to simulate the composition operator. We recall that, given a pointed tree (\mathcal{K}, w) , the formula $\varphi \mid \psi$ essentially asks to partition the children of w into two sets S_1 and S_2 , so that φ is evaluated on the structure obtained from (\mathcal{K}, w) by removing the children of w that belongs to S_2 , as well as their descendants, whereas ψ is evaluated on a model where the children in S_1 are lost instead. Let us show how to encode these types of operations in QK. Assume we want to check whether a pointed forest (\mathcal{K}, w) satisfies a formula φ in $\text{ML}(\mathbf{I})$, built over the set of atomic propositions $P = \{p_1, \dots, p_n\} \subseteq \text{AP}$. In order to represent the submodels obtained from \mathcal{K} through the operator $+_w$, we consider the satisfaction of three atomic propositions $Q = \{q_1, q_2, q_3\}$ that do not belong to P . These atomic propositions are used to represent subsets of children w that must be considered when evaluating a formula of $\text{ML}(\mathbf{I})$. In particular, we use the satisfaction of one of these symbols to represent the current subset of children of w , also called *active* children, whereas the other two auxiliary atomic propositions of Q are used to capture the semantics of \mathbf{I} . The three symbols are reused when considering multiple applications of the operator \mid , since at any time only one symbol encodes the set of active children of w . To formalise this idea, given $\{q_i, q_j, q_k\} = Q$ we define the formula:

$$[q_i = q_j \mid q_k] \stackrel{\text{def}}{=} \Box((q_i \Leftrightarrow q_j \vee q_k) \wedge \neg(q_j \wedge q_k)).$$

Informally, this formula requires that, among the children of the current world w , the ones satisfying q_i are the disjoint union of the ones satisfying q_j and q_k . Given an index $i \in [1, 3]$ denoting which of the symbols in Q is currently used to represent the active children of the current world, the translation $\tau_i(\varphi)$ in QK is defined as shown in Figure 8.2. The translation is straightforward for atomic formulae and Boolean connectives. In the translation of $\varphi_1 \mid \varphi_2$, we rely on the propositional quantification in order to partition the set of active children, i.e. the ones satisfying q_i , into two subsets, depending on the satisfaction of q_j and q_k . We then evaluate φ_1 with respect to the children satisfying q_j and φ_2 with respect to the children satisfying q_k . Lastly, for the translation of $\Diamond\psi$, we essentially relativise the modality of possibility to only consider active children, as done by the $\Diamond(q_i \wedge \dots)$ part of the formula, and rely on the propositional quantification in order to “activate” all the children of the new world. In $\text{ML}(\mathbf{I})$, this corresponds to the fact that subforests obtained through the operator $+_w$ preserves the trees

$$\begin{aligned}
\tau_i(\top) &\stackrel{\text{def}}{=} \top, \\
\tau_i(p) &\stackrel{\text{def}}{=} p, \\
\tau_i(\varphi_1 \Rightarrow \varphi_2) &\stackrel{\text{def}}{=} \tau_i(\varphi_1) \Rightarrow \tau_i(\varphi_2), \\
\tau_i(\neg\varphi) &\stackrel{\text{def}}{=} \neg\tau_i(\varphi), \\
\tau_i(\Diamond\psi) &\stackrel{\text{def}}{=} \Diamond(q_i \wedge \exists q_1 (\Box q_1 \wedge \tau_1(\psi))), \\
\tau_i(\varphi_1 \parallel \varphi_2) &\stackrel{\text{def}}{=} \exists q_j \exists q_k ([q_i = q_j \parallel q_k] \wedge \tau_j(\varphi_1) \wedge \tau_k(\varphi_2)), \\
&\quad \text{where } j, k \in [1, 3], j < k \text{ and } j \neq i \neq k.
\end{aligned}$$

Figure 8.2: Translation from $\text{ML}(\mathbb{I})$ to QK.

rooted at children of w (see Section 7.2.2). The translation τ_i allows us to reduce the model checking problem of $\text{ML}(\mathbb{I})$ to the one of QK as follows.

Lemma 8.1. Let (\mathcal{K}, w) be a pointed forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, and let φ be a formula in $\text{ML}(\mathbb{I})$, written with atomic propositions from P . Let $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[q_1 \leftarrow R(w)])$. $(\mathcal{K}, w) \models \varphi$ in $\text{ML}(\mathbb{I})$ if and only if $(\mathcal{K}', w) \models \tau_1(\varphi)$ in QK.

To keep the presentation light, we leave the proof of Lemma 8.1 (which carries out by structural induction on φ) in Appendix F. Passing through the model checking problem of QK in order to solve the satisfiability problem of $\text{ML}(\mathbb{I})$ is quite promising, as the former problem can be solved in PSPACE [8] by seeing QK as a fragment of MSO [132].

Proposition 8.2 (From [132]). The model checking problem of MSO (alternatively, QK or $\text{ML}(\mathbb{I})$) can be decided in PSPACE.

In order to rely on the model checking algorithm to define a procedure for satisfiability, we must bound the size of the smallest model satisfying a formula in $\text{ML}(\mathbb{I})$. This is where the exponential-size small model property we show in Section 8.2 kicks in.

8.1.2 On alternating time.

Before moving to the exponential-size small model property of $\text{ML}(\mathbb{I})$, let us recall some landmark results on deterministic and alternating space/time complexities, which will help us better understand the complexity class AEXP_{POL}. The concept of alternating space/time complexity classes was set forth by A. K. Chandra, D. Kozen and L. J. Stockmeyer [41] in order to generalise the existential computations required by non-deterministic languages (e.g. problems in NP) together with the universal computations required by their complement (e.g. problems in coNP). The computational device introduced in [41] to achieve this generalisation is the alternating Turing machine (ATM), which extends a non-deterministic Turing machine by dividing the set of states into *existential* and *universal* ones. During the computation, existential states behave as states of a non-deterministic machine, and ask for at least one of the successors of the current state to reach an accepting state. Universal states instead ask for every successor to reach an accepting state. An *alternation* occurs whenever the control switches from an existential state to a universal one, or vice versa.

Given $n, m \in \mathbb{N}$ and functions f and g , we write $\text{ATIME}(f(n), g(m))$ for the class of languages accepted by an ATM with at most $g(m)$ alternations and runtime at most $f(n)$. With this notation, AEXP_{POL} is defined as the union of all $\text{ATIME}(f(n), g(n))$ (n size of the input), where f is an exponential function and g is a polynomial. We write $\text{ATIME}(f(n))$ as a shortcut for $\text{ATIME}(f(n), f(n))$. Similarly, $\text{DSPACE}(f(n))$ (resp. $\text{NSPACE}(f(n))$) stands for the class of languages accepted by a deterministic (resp. non-deterministic) Turing machine running in space $f(n)$. One of the many fundamental results in [41] is a complete characterisation of (non)deterministic space complexity classes in terms of alternating time complexity classes.

Theorem 8.3 (From [41]). Let $f(n) \geq n$.

$$\begin{aligned}\text{NSPACE}(f(n)) &\subseteq \bigcup_{c>0} \text{ATIME}(c \times f(n)^2), \\ \text{ATIME}(f(n)) &\subseteq \text{DSPACE}(f(n)).\end{aligned}$$

In particular, since $\text{DSPACE}(f(n)) \subseteq \text{NSPACE}(f(n))$ (by definition), Theorem 8.3 implies that PSPACE , i.e. the union of all $\text{DSPACE}(f(n))$ where f is a polynomial, is equivalent to alternating polynomial time, i.e. the union of all $\text{ATIME}(f(n))$ where f is a polynomial. This equivalence allows us to cast the PSPACE algorithm for the model checking of MSO [132] in terms of ATIME .

Proposition 8.4 (From [132]). There are polynomials f and g such that the model checking problem of MSO (alternatively, QK or $\text{ML}(\mathbb{I})$) is in $\text{ATIME}(f(n), g(m))$, where n is the input size and m is the formula size.

Fundamentally, the number of alternations required in the model checking algorithm for MSO only depends on the alternation between existential quantification (or disjunctions) and universal quantifications (or conjunctions), and it is thus bounded by the size of the formula, and independent from the size of the structure. This is why the exponential small-model property for $\text{ML}(\mathbb{I})$ leads to AEXP_{POL} : once established, given an input formula φ , the small-model property allows us to guess a pointed forest and has size exponential in $|\varphi|$. Guessing the pointed forest can be done in $\text{NEXPTIME} \subseteq \text{AEXP}_{\text{POL}}$. Afterwards, we run the model-checking algorithm of $\text{ML}(\mathbb{I})$, which takes as input the exponential-size pointed forest and the formula φ , and runs in $\text{ATIME}(f(|\varphi|), g(|\varphi|))$, for a fixed exponential function f and polynomial g , following Proposition 8.4. The whole algorithm runs in AEXP_{POL} .

8.2 CHECKING SATISFIABILITY FOR $\text{ML}(\mathbb{I})$, IN AEXP_{POL}

In order to show that $\text{ML}(\mathbb{I})$ has an exponential-size small model property, we rely on the connections between GML and $\text{ML}(\mathbb{I})$ already established in Chapter 7. In particular, we revise the proof of Theorem 7.16, where every formula φ of $\text{ML}(\mathbb{I})$ is showed to be equivalent to a formula ψ of GML , and show that whenever ψ (and thus φ) is satisfiable, it is satisfied by a pointed forest having a number of worlds that is exponential in $|\varphi|$. Unfortunately, as it is the proof of Theorem 7.16 does not lead to an exponential bound, and needs to be refined to improve the way ψ is computed. This requires a new strategy for the application of Lemma 7.17.

Let us start by recalling some of the definitions introduced in Chapter 7, as well as introducing further auxiliary notions. Given a formula φ in GML , $\text{top}_{\text{gm}}(\varphi)$ stands for the set of subformulae of φ of the form $\Diamond_{\geq k} \psi$ occurring outside the scope of graded modalities. Formally,

$$\begin{aligned}\text{top}_{\text{gm}}(\top) &\stackrel{\text{def}}{=} \text{top}_{\text{gm}}(p) \stackrel{\text{def}}{=} \emptyset, \\ \text{top}_{\text{gm}}(\Diamond_{\geq k}\varphi) &\stackrel{\text{def}}{=} \{\Diamond_{\geq k}\varphi\}, \\ \text{top}_{\text{gm}}(\neg\varphi) &\stackrel{\text{def}}{=} \text{top}_{\text{gm}}(\varphi), \\ \text{top}_{\text{gm}}(\varphi \Rightarrow \psi) &\stackrel{\text{def}}{=} \text{top}_{\text{gm}}(\varphi) \cup \text{top}_{\text{gm}}(\psi).\end{aligned}$$

Similarly, we recall that $\text{top}_{\text{AP}}(\varphi)$ stands for the set of atomic propositions of φ that appear outside graded modalities. We write $\text{md}(\varphi)$ for the modal depth of φ , that is the maximal number of nested modalities occurring in φ . We extend the notion of $\text{top}_{\text{gm}}(\varphi)$ and, given a natural number $d \in \mathbb{N}$, we write $\text{gm}(d, \varphi)$ to denote the set of subformulae of φ of the form $\Diamond_{\geq k}\psi$ occurring under the scope of exactly d nested graded modalities. Formally, we have

$$\text{gm}(0, \varphi) \stackrel{\text{def}}{=} \text{top}_{\text{gm}}(\varphi) \quad \text{gm}(d+1, \varphi) \stackrel{\text{def}}{=} \bigcup_{\Diamond_{\geq k}\psi \in \text{top}_{\text{gm}}(\varphi)} \text{gm}(d, \psi).$$

Notice that if $d \geq \text{md}(\varphi)$, then $\text{gm}(d, \varphi) = \emptyset$.

8.2.1 GML formulae in good shape.

Exactly as in Chapter 7, the technical developments required to show that the satisfiability problem of $\text{ML}(\mathbb{I})$ can be solved in AEXP_{POL} heavily relies on the concept of disjoint formulae. We remind the reader that a set $\Phi = \{\psi_1, \dots, \psi_n\}$ is said to contain *disjoint* GML formulae whenever for every distinct $i, j \in [1, n]$, $\psi_i \wedge \psi_j$ is unsatisfiable (Definition 7.9). From the disjointness property, we introduce the following notion of formulae in good shape.

Definition 8.5 (Good shape). A GML formula φ in *good shape* if for every $d \in [0, \text{md}(\varphi)-1]$, given $\text{gm}(d, \varphi) = \{\Diamond_{\geq k_1}\psi_1, \dots, \Diamond_{\geq k_n}\psi_n\}$, the set $\{\psi_1, \dots, \psi_n\}$ is of disjoint formulae.

Roughly speaking, a formula φ is in good shape if for every two distinct formulae of the form $\Diamond_{\geq j}\psi$ and $\Diamond_{\geq k}\chi$ occurring under the same number of nested graded modalities, the formula $\psi \wedge \chi$ is unsatisfiable. For instance, the formula $\varphi = \Diamond\Diamond p \wedge \Diamond\neg\Diamond p$ is in good shape. Indeed, $\text{gm}(1, \varphi)$ corresponds to the singleton set $\{\Diamond p\}$, which (as all singleton sets) is a set of disjoint formulae. The same holds true for $\text{gm}(0, \varphi) = \{\Diamond\Diamond p, \Diamond\neg\Diamond p\}$, since $\Diamond p \wedge \neg\Diamond p$ is unsatisfiable.

The set of GML formulae that are in good shape is instrumental to show the aforementioned exponential-size small model property of $\text{ML}(\mathbb{I})$, as the size of their minimal model (if any) can be bounded using the notion of branching degree introduced below (Definition 8.6). The fundamental feature of the branching degree is that it does not change when putting a formula in disjoint normal form (seeing formulae in $\text{top}_{\text{gm}}(\varphi)$ as atomic). Thanks to this property we are able to define a translation from $\text{ML}(\mathbb{I})$ to GML that produces a formula with relatively small branching degree, despite its size being non-elementary.

Definition 8.6. Let φ be a GML formula with $\text{top}_{\text{gm}}(\varphi) = \{\Diamond_{\geq k_1}\psi_1, \dots, \Diamond_{\geq k_n}\psi_n\}$. Given $d \in \mathbb{N}$, the *branching degree* at depth d of φ is recursively defined as follows:

$$\text{bd}(0, \varphi) \stackrel{\text{def}}{=} k_1 + \dots + k_n \quad \text{bd}(d+1, \varphi) \stackrel{\text{def}}{=} \max(\{\text{bd}(d, \psi) \mid \Diamond_{\geq k}\psi \in \text{top}_{\text{gm}}(\varphi)\}).$$

The *maximal branching degree* of φ is $\max_{\text{bd}}(\varphi) = \max \{\text{bd}(d, \varphi) \mid d \in [0, \text{md}(\varphi)]\}$.

Notice that $\text{bd}(d, \varphi)$ can be understood as the maximal $\text{bd}(0, \psi)$ for some subformula ψ occurring at the modal depth d within φ . When φ is a satisfiable formula in good shape, we are able to rely on $\max_{\text{bd}}(\varphi)$ to obtain a bound on the smallest model satisfying it, as stated in Lemma 8.7 below.

Lemma 8.7. Every satisfiable GML formula φ in good shape is satisfied by a pointed forest with at most $(\max_{\text{bd}}(\varphi) + 1)^{\text{md}(\varphi)}$ worlds.

Proof. The proof is by induction on the modal degree of φ .

base case: $\text{md}(\varphi) = 0$. In this case, φ is a Boolean combination of atomic propositions, and thus the satisfaction of φ can be witnessed on a pointed forest with one single world (i.e. the satisfaction of φ only depends on the atomic propositions satisfied by the current world).

induction step: $\text{md}(\varphi) = d + 1$. Let $\text{top}_{\text{gm}}(\varphi)$ and $\text{top}_{\text{AP}}(\varphi)$ be the sets $\{\Diamond_{\geq k_1} \psi_1, \dots, \Diamond_{\geq k_n} \psi_n\}$ and $\{p_1, \dots, p_m\}$, respectively. We rely on Proposition 7.8 (Chapter 7), whose statement is recalled below.

[**Proposition 7.8.** Every formula φ in GML is equivalent to a disjunction of formulae belonging to $\text{Conj}(\text{top}_{\text{gm}}(\varphi) \cup \text{top}_{\text{AP}}(\varphi))$.]

We derive that there is φ' in GML such that $\varphi \equiv \varphi'$ and φ' is a disjunction of conjunctions of possibly negated formulae from $\text{top}_{\text{gm}}(\varphi) \cup \text{top}_{\text{AP}}(\varphi)$. Since φ is satisfiable and $\varphi \equiv \varphi'$, one of the disjuncts of φ' must be satisfiable. Let χ be such a disjunct, which is a conjunction of the form (modulo associativity and commutativity of \wedge):

$$\chi = \Diamond_{\geq k_{i_1}} \psi_{i_1} \wedge \dots \wedge \Diamond_{\geq k_{i_p}} \psi_{i_p} \wedge \neg \Diamond_{\geq k_{j_1}} \psi_{j_1} \wedge \dots \wedge \neg \Diamond_{\geq k_{j_q}} \psi_{k_{j_q}} \wedge L_1 \wedge \dots \wedge L_r,$$

where L_1, \dots, L_r are literals built upon $\text{top}_{\text{AP}}(\varphi)$. By definition $\text{top}_{\text{gm}}(\chi) \subseteq \text{top}_{\text{gm}}(\varphi)$, which allows us to conclude that $\text{bd}(0, \chi)$ satisfies the following (in)equalities:

$$k_{i_1} + \dots + k_{i_p} + k_{j_1} + \dots + k_{j_q} \leq \text{bd}(0, \chi) \leq \text{bd}(0, \varphi) \leq \max_{\text{bd}}(\varphi).$$

Moreover, $\text{top}_{\text{gm}}(\chi) \subseteq \text{top}_{\text{gm}}(\varphi)$ allows us to conclude that χ is in good shape, directly from the fact that φ is in good shape.

As χ is satisfiable, let us consider a finite forest $(\mathcal{K}, \mathbf{w})$, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, such that $(\mathcal{K}, \mathbf{w}) \models \varphi$. By definition of \models , for each $i \in \{i_1, \dots, i_p\}$ there is a set $S_i = \{\mathbf{w}_{i,1}, \dots, \mathbf{w}_{i,k_i}\}$ of k_i children of \mathbf{w} such that each child in S_i satisfies φ_i . Let us consider the Kripke-style finite forest $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$ such that

- $\mathcal{W}' \stackrel{\text{def}}{=} \{\mathbf{w}\} \cup \{\mathbf{w}' \mid \mathbf{w}' \in R^*(\mathbf{w}''), \mathbf{w}'' \in S_1 \cup \dots \cup S_m\}$,
- $R' = R \cap (\mathcal{W}' \times \mathcal{W}')$,
- \mathcal{V}' is the restriction of \mathcal{V} to \mathcal{W}' .

Informally, $(\mathcal{K}', \mathbf{w})$ is obtained from $(\mathcal{K}, \mathbf{w})$ by removing the children (and their subtrees) of \mathbf{w} that do not belong to $S_1 \cup \dots \cup S_m$. It is easy to verify that $(\mathcal{K}', \mathbf{w}) \models \chi$. Moreover, $\text{card}(R'(\mathbf{w})) = k_{i_1} + \dots + k_{i_p} \leq \max_{\text{bd}}(\varphi)$. Let us now consider a world $\mathbf{w}_{i,j} \in S_i$, where $i \in \{i_1, \dots, i_p\}$ and $j \in [1, k_i]$. By definition, $(\mathcal{K}', \mathbf{w}_{i,j}) \models \psi_i$, where $\text{md}(\psi_i) < \text{md}(\varphi)$. We apply the induction hypothesis, and conclude that there is a pointed forest $(\mathcal{K}_{i,j}, \mathbf{w}'_{i,j})$, with $\mathcal{K}_{i,j} = (\mathcal{W}_{i,j}, R_{i,j}, \mathcal{V}_{i,j})$, such that $\text{card}(\mathcal{W}_{i,j}) \leq (\max_{\text{bd}}(\psi_i) + 1)^{\text{md}(\psi_i)}$ and $(\mathcal{K}_{i,j}, \mathbf{w}'_{i,j}) \models \psi_i$. Without loss of generality, let us assume that $\mathbf{w}'_{i,j} = \mathbf{w}_{i,j}$ and that $\mathbf{w}_{i,j}$ is the only world appearing in both $\mathcal{W}_{i,j}$ and \mathcal{W}' . We construct similar pointed forests $(\mathcal{K}_{i,j}, \mathbf{w}_{i,j})$ for every world in $R(\mathbf{w}) = S_1 \cup \dots \cup S_m$. Again without loss of generality, we can assume that these pointed forests feature distinct sets of worlds. Now, since χ is in good shape and $(\mathcal{K}_{i,j}, \mathbf{w}'_{i,j}) \models \psi_i$, for every $\Diamond_{\geq j} \gamma \in \text{top}_{\text{gm}}(\chi)$ such that γ is not syntactically equal to ψ_i , we have $(\mathcal{K}_{i,j}, \mathbf{w}'_{i,j}) \not\models \gamma$. This implies that replacing the subtree in \mathcal{K}' rooted at a world $\mathbf{w}_{i,j} \in S_i$, where $i \in \{i_1, \dots, i_p\}$ and $j \in [1, k_i]$, with the tree described by $\mathcal{K}_{i,j}$ does not change the satisfaction of χ . Formally, the Kripke-style finite forest $\mathcal{K}'' = (\mathcal{W}'', R'', \mathcal{V}'')$ obtained from these replacements is defined as follows

- $\mathcal{W}'' \stackrel{\text{def}}{=} \{\mathbf{w}\} \cup \bigcup_{i \in \{i_1, \dots, i_p\}, j \in [1, k_i]} \mathcal{W}_{i,j}$,
- $R'' = (\{\mathbf{w}\} \times R'(\mathbf{w})) \cup \bigcup_{i \in \{i_1, \dots, i_p\}, j \in [1, k_i]} R_{i,j}$,
- for every $p \in \text{AP}$,

$$\mathcal{V}''(p) = \left\{ \tilde{\mathbf{w}} \middle| \begin{array}{l} \tilde{\mathbf{w}} = \mathbf{w} \text{ and } \mathbf{w} \in \mathcal{V}'(p), \text{ or} \\ \text{there are } i \in \{i_1, \dots, i_p\}, j \in [1, k_i] \text{ s.t. } \tilde{\mathbf{w}} \in \mathcal{W}_{i,j} \text{ and } \tilde{\mathbf{w}} \in \mathcal{V}_{i,j}(p) \end{array} \right\}.$$

We have,

$$\begin{aligned} \text{card}(\mathcal{W}'') &= 1 + \sum_{i \in \{i_1, \dots, i_p\}, j \in [1, k_i]} \text{card}(\mathcal{W}_{i,j}) && \text{by definition} \\ &\leq 1 + \sum_{i \in \{i_1, \dots, i_p\}} k_i (\max_{\text{bd}}(\psi_i) + 1)^{\text{md}(\psi_i)} && \text{by def. of } \mathcal{W}_{i,j} \\ &\leq 1 + (k_1 + \dots + k_p) (\max_{\text{bd}}(\varphi) + 1)^{\text{md}(\varphi)-1} && \text{by } \max_{\text{bd}}(\psi_i) \leq \max_{\text{bd}}(\varphi) \\ &\quad \text{and } \text{md}(\psi_i) < \text{md}(\varphi) \\ &\leq 1 + \max_{\text{bd}}(\varphi) (\max_{\text{bd}}(\varphi) + 1)^{\text{md}(\varphi)-1} && \text{as } k_1 + \dots + k_p \leq \max_{\text{bd}}(\varphi) \\ &\leq (\max_{\text{bd}}(\varphi) + 1) (\max_{\text{bd}}(\varphi) + 1)^{\text{md}(\varphi)-1} && \text{as } (\max_{\text{bd}}(\varphi) + 1)^{\text{md}(\varphi)-1} \geq 1 \\ &= (\max_{\text{bd}}(\varphi) + 1)^{\text{md}(\varphi)} && \text{by definition.} \end{aligned}$$

Notice that $R''(\mathbf{w}) = R'(\mathbf{w})$. As $(\mathcal{K}', \mathbf{w}) \models \chi$, in order to conclude that $(\mathcal{K}'', \mathbf{w}) \models \chi$, it is sufficient to show that:

for every $\mathbf{w}' \in R'(\mathbf{w})$ and all $i \in \{i_1, \dots, i_p, j_1, \dots, j_q\}$, $(\mathcal{K}', \mathbf{w}') \models \psi_i$ iff $(\mathcal{K}'', \mathbf{w}') \models \psi_i$.

This holds from the definition of \mathcal{K}'' , together with the fact that the formula φ is in good shape.

(\Rightarrow): Suppose $(\mathcal{K}', \mathbf{w}') \models \psi_i$. By definition of \mathcal{K}' , there is $i' \in \{i_1, \dots, i_p\}$ and $j \in [1, k_{i'}]$ such that $\mathbf{w}' = \mathbf{w}_{i',j} \in S_{i'}$. By definition of $S_{i'}$, $(\mathcal{K}', \mathbf{w}') \models \psi_{i'}$. Therefore, $(\mathcal{K}', \mathbf{w}') \models \psi_i \wedge \psi_{i'}$. Since φ is in good shape and $\text{top}_{\text{gm}}(\varphi) = \{\Diamond_{\geq k_1} \psi_1, \dots, \Diamond_{\geq k_n} \psi_n\}$, for every two distinct formulae γ and γ' in $\{\psi_1, \dots, \psi_n\}$ we have that $\gamma \wedge \gamma'$ is unsatisfiable. Since ψ_i and $\psi_{i'}$ belong to $\{\psi_1, \dots, \psi_n\}$, this implies that $i = i'$. Therefore, $\mathbf{w}' = \mathbf{w}_{i,j} \in S_i$. By definition of $\mathcal{K}_{i,j}$, $(\mathcal{K}_{i,j}, \mathbf{w}') \models \psi_i$. By definition of \mathcal{K}'' , the tree rooted at \mathbf{w}' in $\mathcal{K}_{i,j}$ is the tree rooted at \mathbf{w}' in \mathcal{K}'' . This implies that $(\mathcal{K}'', \mathbf{w}') \models \psi_i$ (see Lemma E.2 in Appendix E for a formal proof of this last step).

(\Leftarrow): Suppose $(\mathcal{K}'', \mathbf{w}') \models \psi_i$. As $\mathbf{w}' \in R''(\mathbf{w}) = R'(\mathbf{w})$, there is $i' \in \{i_1, \dots, i_p\}$ and $j \in [1, k_{i'}]$ such that $\mathbf{w}' = \mathbf{w}_{i',j} \in S_{i'}$. By definition of $\mathcal{K}_{i',j}$, $(\mathcal{K}_{i',j}, \mathbf{w}') \models \psi_{i'}$. By definition of \mathcal{K}'' , the tree rooted at \mathbf{w}' in $\mathcal{K}_{i',j}$ is the tree rooted at \mathbf{w}' in \mathcal{K}'' . This implies that $(\mathcal{K}'', \mathbf{w}') \models \psi_{i'}$. As in the proof of the left to right direction, since φ is in good shape we conclude that $i = i'$ and therefore $\mathbf{w}' \in S_i$. By definition of S_i , $(\mathcal{K}', \mathbf{w}') \models \psi_i$. \square

8.2.2 The exponential-size small model property of $\text{ML}(\mathbb{I})$.

Thanks to Lemma 8.7, to show that $\text{ML}(\mathbb{I})$ has the exponential-size small model property it is sufficient to establish that given φ in $\text{ML}(\mathbb{I})$, there is a logically equivalent GML formula ψ in good shape such that $\psi \equiv \varphi$, $\text{md}(\psi) \leq \text{md}(\varphi)$ and $\max_{\text{bd}}(\psi)$ is exponential in $|\varphi|$. To highlight the essential step required in order to prove this result, we first consider the fragment \mathcal{F} of $\text{ML}(\mathbb{I})$ whose formulae are from the grammar below:

$$\varphi := \Diamond_{\geq k} \psi \mid p \mid \varphi \mid \varphi \mid \varphi \wedge \varphi \mid \neg \varphi,$$

where $p \in \text{AP}$ and $\Diamond_{\geq k} \psi$ is a formula in GML (abusively assumed in $\text{ML}(\mathbb{I})$ but we know that GML can be seen as a fragment of $\text{ML}(\mathbb{I})$, see Corollary 7.18). Given φ in $\text{ML}(\mathbb{I})$, we write

$\text{cd}(\varphi)$ to denote its *composition degree*, that is the maximal number of imbrications of \mathbf{I} in φ , not occurring under the scope of a graded modality. Formally,

$$\begin{aligned}\text{cd}(\top) &\stackrel{\text{def}}{=} \text{cd}(p) \stackrel{\text{def}}{=} \text{cd}(\Diamond_{\geq k}\varphi) \stackrel{\text{def}}{=} 0, \\ \text{cd}(\neg\varphi) &\stackrel{\text{def}}{=} \text{cd}(\varphi), \\ \text{cd}(\varphi \Rightarrow \psi) &\stackrel{\text{def}}{=} \max(\text{cd}(\varphi), \text{cd}(\psi)), \\ \text{cd}(\varphi \mathbf{I} \psi) &\stackrel{\text{def}}{=} 1 + \max(\text{cd}(\varphi), \text{cd}(\psi)).\end{aligned}$$

Notice that for formulae of the fragment \mathcal{F} , the composition degree corresponds to the maximal number of imbrication of \mathbf{I} . We extend the notion of $\text{top}_{\text{gm}}(\varphi)$ to φ in $\text{ML}(\mathbf{I})$ or in \mathcal{F} , so that $\text{top}_{\text{gm}}(\varphi) = \text{top}_{\text{gm}}(\varphi[\mathbf{I} \leftarrow \wedge])$, where $\varphi[\mathbf{I} \leftarrow \wedge]$ is the formula obtained from φ by replacing every occurrence of the operator \mathbf{I} by \wedge . Similarly, we write $\text{bd}(m, \varphi)$ for $\text{bd}(m, \varphi[\mathbf{I} \leftarrow \wedge])$.

Let φ be in \mathcal{F} such that $\text{top}_{\text{gm}}(\varphi) \subseteq \{\Diamond_{\geq k_1}\chi_1, \dots, \Diamond_{\geq k_n}\chi_n\}$. The key step to show the exponential-size model property consists in manipulating the formulae in $\text{top}_{\text{gm}}(\varphi)$ to produce a formula in good shape that witnesses an exponential blow-up on $\text{bd}(0, \varphi)$, whereas, for every $m > 1$, $\text{bd}(m, \varphi)$ remains polynomially bounded. The exact bounds obtained from this manipulation are given in the following lemma, whose proof heavily relies on Lemma 7.17 (Chapter 7).

Lemma 8.8. Let φ be a formula of the fragment \mathcal{F} such that $\text{top}_{\text{gm}}(\varphi) \subseteq \{\Diamond_{\geq k_1}\chi_1, \dots, \Diamond_{\geq k_n}\chi_n\}$, $\hat{k} = \max\{k_1, \dots, k_n\}$, and $\chi_1 \wedge \dots \wedge \chi_n$ is in good shape. There is a GML formula ψ in good shape and such that $\varphi \equiv \psi$, $\text{top}_{\text{gm}}(\psi) \subseteq \{\Diamond_{\geq j}\chi \mid j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}], \chi \in 2^{\{\chi_1, \dots, \chi_n\}}\}$ and,

1. $\text{md}(\psi) \leq 1 + \max(\{\text{md}(\chi_i) \mid i \in [1, n]\})$,
2. $\text{bd}(0, \psi) \leq (1 + (\text{cd}(\varphi) + 1) \times \hat{k})^2 \times 2^{n-1}$,
3. for every $m \geq 1$, $\text{bd}(m, \psi) \leq \text{bd}(m-1, \chi_1 \wedge \dots \wedge \chi_n)$.

We remind the reader that the set $2^{\{\chi_1, \dots, \chi_n\}}$ is defined as:

$$2^{\{\chi_1, \dots, \chi_n\}} = \left\{ \bigwedge_{\substack{i \in [1, n] \\ f(i) = \top}} \chi_i \wedge \bigwedge_{\substack{i \in [1, n] \\ f(i) = \perp}} \neg\chi_i \mid f : [1, n] \rightarrow \{\top, \perp\}, \quad \begin{array}{l} f(i) = \top \text{ for some } i \in [1, n] \end{array} \right\}.$$

Let us also recall the definition of core formulae given during Chapter 7, as it will be instrumental during the proof of Lemma 8.8.

Definition 7.7 (Core formulae). Let Φ be a finite set of formulae in GML. Consider a natural number $k \in \mathbb{N}$ and let P be a finite set of atomic propositions. We denote with $\text{Core}(\Phi, k, P)$ the following set of formulae in GML:

$$\text{Core}(\Phi, k, P) \stackrel{\text{def}}{=} \{\Diamond_{\geq j}\varphi, p, \top \mid j \in [1, k], \varphi \in \Phi, p \in P\}.$$

As already stated, Lemma 8.8 is proved by manipulating the formula φ to obtain the formula ψ satisfying the constraints (1)–(3). Notice that whereas the constraints (1) and (3) state the modal depth and branching degree at depth $m \geq 1$ of ψ are polynomially bounded by the ones of χ_1, \dots, χ_n , the constraint (2) shows an exponential blow-up on $\text{bd}(0, \psi)$. The fact that the branching degree of ψ witnesses an exponential blow-up only in the case of depth 0 is crucial to obtain the exponential-size small model property of $\text{ML}(\mathbf{I})$.

Proof of Lemma 8.8. Below, let us write Φ for the set of GML formulae $\{\chi_1, \dots, \chi_n\}$. First of all, we rely on Lemma 7.11, proved in Chapter 7 and whose statement is given below.

Lemma 7.11. Let Φ be a finite set of GML formulae, $k \in \mathbb{N}$ and $P \subseteq_{\text{fin}} \text{AP}$. Every formula in $\text{Core}(\Phi, k, P)$ is equivalent to a disjunction of formulae in $\text{Conj}(\text{Core}(2^\Phi, k, P))$.

Recall that, for a finite set of formulae S , $\text{Conj}(S)$ is the set of conjunctions built upon possibly negated formulae in S . Thanks to Lemma 7.11, every formula in $\text{top}_{\text{gm}}(\varphi)$ is equivalent to a disjunction of formulae in $\text{Conj}(\text{Core}(2^\Phi, \hat{k}, \text{top}_{\text{AP}}(\varphi)))$. Thus, let us consider the formula φ' obtained from φ by replacing every occurrence of formulae in $\text{top}_{\text{gm}}(\varphi)$ not occurring under the scope of a graded modality with the equivalent disjunction of formulae from $\text{Conj}(\text{Core}(2^\Phi, \hat{k}, \text{top}_{\text{AP}}(\varphi)))$. The formula φ' belongs to the fragment \mathcal{F} and it is such that $\text{cd}(\varphi) = \text{cd}(\varphi')$ and $\text{top}_{\text{gm}}(\varphi') \subseteq \text{Core}(2^\Phi, \hat{k}, \text{top}_{\text{AP}}(\varphi))$. We show that φ' is equivalent to a Boolean combination ψ of formulae belonging to the set $\text{Core}(2^\Phi, (\text{cd}(\varphi) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$. For this result, we prove that every formula ψ' , in the fragment \mathcal{F} and such that $\text{top}_{\text{gm}}(\psi') \subseteq \text{Core}(2^\Phi, \hat{k}, \text{top}_{\text{AP}}(\varphi))$ and $\text{cd}(\psi') = n$, is equivalent to a Boolean combination of formulae in $\text{Core}(2^\Phi, (n+1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$. The proof is by induction on n .

base case: $n = 0$. In this case, ψ' is already the required Boolean combination.

induction step: $n \geq 1$. In this case, it is sufficient to show that every subformula $\varphi_1 \parallel \varphi_2$ of ψ' not occurring under the scope of a composition operator is equivalent to a Boolean combination of formulae in $\text{Core}(2^\Phi, (n+1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$. By definition, $\text{cd}(\varphi_1) < n$ and $\text{cd}(\varphi_2) < n$. Thus, by induction hypothesis there are two formulae φ'_1 and φ'_2 such that

- $\varphi'_1 \equiv \varphi_1$ and φ'_1 is in $\text{Bool}(\text{Core}(2^\Phi, (\text{cd}(\varphi_1) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi)))$,
- $\varphi'_2 \equiv \varphi_2$ and φ'_2 is in $\text{Bool}(\text{Core}(2^\Phi, (\text{cd}(\varphi_2) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi)))$.

We now rely on Lemma 7.17, proven in Chapter 7 and whose statement is recalled below.

Lemma 7.17. Let φ and ψ be two GML formulae with $\text{top}_{\text{gm}}(\varphi) = \{\Diamond_{\geq j_1} \varphi_1, \dots, \Diamond_{\geq j_n} \varphi_n\}$, $\text{top}_{\text{gm}}(\psi) = \{\Diamond_{\geq k_1} \psi_1, \dots, \Diamond_{\geq k_m} \psi_m\}$ and where $\Phi = \{\varphi_1, \dots, \varphi_n\} \cup \{\psi_1, \dots, \psi_m\}$ is a set of disjoint formulae. We have $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \varphi \parallel \psi \Leftrightarrow \chi$, where χ is either \perp or a disjunction of formulae from $\text{Conj}(\text{Core}(\Phi, \max(j_1, \dots, j_n) + \max(k_1, \dots, k_m), \text{top}_{\text{AP}}(\varphi) \cup \text{top}_{\text{AP}}(\psi)))$.

Since the proof system $\mathcal{H}_{\text{GML}}(\mathbf{I})$ defined in Chapter 7 is sound with respect to $\text{ML}(\mathbf{I})$, by applying this lemma we conclude that $\varphi_1 \parallel \varphi_2$ is equivalent to a Boolean combination of formulae from $\text{Core}(2^\Phi, (\text{cd}(\varphi_1) + 1) \times \hat{k} + (\text{cd}(\varphi_2) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$. From $\text{cd}(\varphi_1) < n$ and $\text{cd}(\varphi_2) < n$ we conclude that $(\text{cd}(\varphi_1) + 1) \times \hat{k} + (\text{cd}(\varphi_2) + 1) \times \hat{k} \leq (n+1) \times \hat{k}$. By definition of core formulae, this implies that $\text{Core}(2^\Phi, (\text{cd}(\varphi_1) + 1) \times \hat{k} + (\text{cd}(\varphi_2) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi)) \subseteq \text{Core}(2^\Phi, (n+1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$, concluding the induction step.

Below, let ψ be the Boolean combination of formulae from $\text{Core}(2^\Phi, (\text{cd}(\varphi) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$ that is equivalent to φ' . Therefore, ψ is a formula in GML such that $\psi \equiv \varphi$ and $\text{top}_{\text{gm}}(\psi) \subseteq \{\Diamond_{\geq j} \chi \mid j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}], \chi \in 2^\Phi\}$. In order to conclude the proof, it is sufficient to show that ψ is in good shape and that its bd satisfies the required constraints.

The proof that ψ is in good shape is quite straightforward. Since ψ is a Boolean combination of formulae from $\text{Core}(2^\Phi, (\text{cd}(\varphi) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$, where $\Phi = \{\chi_1, \dots, \chi_n\}$, we have $\text{gm}(1, \psi) \subseteq \bigcup_{\chi \in 2^\Phi} \text{gm}(0, \chi) = \text{gm}(0, \chi_1 \wedge \dots \wedge \chi_n)$. Since by hypothesis $\chi_1 \wedge \dots \wedge \chi_n$ is a formula in good shape, this allows us to conclude that for every $d \in [1, \text{md}(\psi) - 1]$, given $\text{gm}(d, \psi) = \{\Diamond_{\geq j_1} \psi_1, \dots, \Diamond_{\geq j_m} \psi_m\}$, the set $\{\psi_1, \dots, \psi_m\}$ is of disjoint formulae. In order to conclude that ψ is in good shape, we just need to check that given $\text{gm}(0, \psi) = \{\Diamond_{\geq j'_1} \psi'_1, \dots, \Diamond_{\geq j'_m} \psi'_m\}$, the set $\{\psi'_1, \dots, \psi'_m\}$ is of disjoint formulae. By definition, $\text{gm}(0, \psi) = \text{top}_{\text{gm}}(\psi) \subseteq \{\Diamond_{\geq j} \chi \mid j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}], \chi \in 2^\Phi\}$. Therefore, $\{\psi'_1, \dots, \psi'_m\} \subseteq 2^\Phi$. This implies that $\{\psi'_1, \dots, \psi'_m\}$ is a set of disjoint formulae directly from Lemma 7.10, whose statement is recalled below.

[**Lemma 7.10.** Let Φ be a finite set of formulae in GML. 2^Φ is a set of disjoint formulae.]

Let us now check that the constraints on the bd of ψ are satisfied.

1. $\text{md}(\psi) \leq \max(\{\text{md}(\chi_i) \mid i \in [1, n]\}) + 1$ follows directly from the fact that ψ is a Boolean combination of formulae from $\text{Core}(2^\Phi, (\text{cd}(\varphi) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$, where $\Phi = \{\psi_1, \dots, \psi_n\}$.
2. We show that $\text{bd}(0, \psi) \leq (1 + (\text{cd}(\varphi) + 1) \times \hat{k})^2 \times 2^{n-1}$. From $\text{top}_{\text{gm}}(\psi) \subseteq \{\Diamond_{\geq j}\chi \mid j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}], \chi \in 2^\Phi\}$ we conclude that $\text{bd}(0, \psi)$ is at most the sum of every coefficient j of all the formulae $\Diamond_{\geq j}\chi$ where $j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}]$ and $\chi \in 2^\Phi$. Thus,

$$\begin{aligned} \text{bd}(0, \psi) &\leq \sum_{\chi \in 2^\Phi} \sum_{j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}]} j \\ &= \sum_{\chi \in 2^\Phi} \frac{(\text{cd}(\varphi) + 1) \times \hat{k} \times ((\text{cd}(\varphi) + 1) \times \hat{k} + 1)}{2} \\ &= \frac{(\text{cd}(\varphi) + 1) \times \hat{k} \times ((\text{cd}(\varphi) + 1) \times \hat{k} + 1)}{2} \times \text{card}(2^\Phi) \\ &\leq \frac{(\text{cd}(\varphi) + 1) \times \hat{k} \times ((\text{cd}(\varphi) + 1) \times \hat{k} + 1)}{2} \times 2^{\text{card}(\Phi)} \\ &= (\text{cd}(\varphi) + 1) \times \hat{k} \times ((\text{cd}(\varphi) + 1) \times \hat{k} + 1) \times 2^{n-1} \\ &\leq (1 + (\text{cd}(\varphi) + 1) \times \hat{k})^2 \times 2^{n-1} \end{aligned}$$

3. Let $m \geq 1$. We show that $\text{bd}(m, \psi) \leq \text{bd}(m-1, \chi_1 \wedge \dots \wedge \chi_n)$. Recall that, by definition,

$$\text{bd}(m, \psi) = \max(\{\text{bd}(m-1, \chi) \mid \Diamond_{\geq k}\chi \in \text{top}_{\text{gm}}(\psi)\}).$$

As ψ is a Boolean combination of core formulae from $\text{Core}(2^\Phi, (\text{cd}(\varphi) + 1) \times \hat{k}, \text{top}_{\text{AP}}(\varphi))$, this means that $\text{bd}(m, \psi) \leq \max(\{\text{bd}(m-1, \chi) \mid \chi \in 2^\Phi\})$. By definition, every formula χ in 2^Φ is a conjunction of possibly negated formulae in Φ , where every formula of Φ occurs exactly once. As $\Phi = \{\chi_1, \dots, \chi_n\}$, this implies that $\text{top}_{\text{gm}}(\chi) = \text{top}_{\text{gm}}(\chi_1 \wedge \dots \wedge \chi_n)$. Directly from the definition of bd , this implies that $\text{bd}(m-1, \chi) = \text{bd}(m-1, \chi_1 \wedge \dots \wedge \chi_n)$. Thus, $\max(\{\text{bd}(m-1, \chi) \mid \chi \in 2^\Phi\}) = \text{bd}(m-1, \chi_1 \wedge \dots \wedge \chi_n)$, which allows us to conclude that $\text{bd}(m, \psi) \leq \text{bd}(m-1, \chi_1 \wedge \dots \wedge \chi_n)$. \square

Applying adequately the transformation from Lemma 8.8 to a formula in $\text{ML}(\mathbf{I})$, i.e. by considering maximal subformulae of the fragment \mathcal{F} , allows us to get a logically equivalent GML formula having small models. In order to simplify the exposition of the forthcoming proof of exponential size small model property, we introduce the notion of subformulae at level $d \in \mathbb{N}$. Closely related to the set $\text{gm}(d, \varphi)$, the set of subformulae of φ at level d , written $\text{lvl}(d, \varphi)$, is defined as follows:

$$\text{lvl}(0, \varphi) \stackrel{\text{def}}{=} \{\varphi\} \quad \text{lvl}(d+1, \varphi) \stackrel{\text{def}}{=} \bigcup_{\Diamond_{\geq k}\psi \in \text{top}_{\text{gm}}(\varphi)} \text{lvl}(d, \psi).$$

Notice that for all $d \in \mathbb{N}$, if $\text{gm}(d, \varphi) = \{\Diamond_{\geq k_1}\psi_1, \dots, \Diamond_{\geq k_n}\psi_n\}$ then $\text{lvl}(d+1, \varphi) = \{\psi_1, \dots, \psi_n\}$.

Lemma 8.9. For every formula φ in $\text{ML}(\mathbf{I})$ there is an equivalent GML formula ψ in good shape and such that $\text{md}(\psi) \leq \text{md}(\varphi)$ and $\text{max}_{\text{bd}}(\psi) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$.

Proof. During the proof, we assume that every subformula ψ of φ without occurrences of the graded modalities (where we see $\Diamond\chi$ as a shortcut for $\Diamond_{\geq 1}\chi$) is a Boolean combination of atomic propositions. This assumption is without loss of generality. Indeed, a formula ψ of $\text{ML}(\mathbf{I})$ without graded modalities is a formula built upon Boolean connectives, the composition operator \mathbf{I} and atomic propositions. However, as ψ lacks graded modalities, its satisfaction with respect to a pointed forest (\mathcal{K}, w) only depends on the current world w (and on the valuation function of the Kripke-style finite forest \mathcal{K}). This implies that the formula $\psi[\mathbf{I} \leftarrow \wedge]$, obtained from ψ by

replacing every occurrence of the operator $\mathbf{|}$ by \wedge , is equivalent to ψ . Notice that this assumption implies that the formulae in $\text{lvl}(\text{md}(\varphi), \varphi)$ are Boolean combinations of atomic propositions, which means that they belong to GML and that their conjunction is in good shape.

To prove the result, we build a sequence of formulae $\gamma_0, \gamma_1, \dots, \gamma_{\text{md}(\varphi)}$, where $\gamma_0 = \varphi$ and for every $i \in [0, \text{md}(\varphi)]$ the following properties are satisfied:

1. $\gamma_i \equiv \varphi$,
2. given $\text{lvl}(\text{md}(\varphi) - i, \gamma_i) = \{\chi_1, \dots, \chi_n\}$, $\chi_1 \wedge \dots \wedge \chi_n$ is a GML formula in good shape,
3. $\text{md}(\gamma_i) \leq \text{md}(\varphi)$,
4. for every $j \in [0, \text{md}(\varphi) - (i + 1)]$ there are $k_1, \dots, k_n \in \mathbb{N}$ and formulae χ_1, \dots, χ_n and χ'_1, \dots, χ'_n such that $\text{gm}(j, \varphi) = \{\Diamond_{k_1}\chi_1, \dots, \Diamond_{k_n}\chi_n\}$, $\text{gm}(j, \gamma_i) = \{\Diamond_{k_1}\chi'_1, \dots, \Diamond_{k_n}\chi'_n\}$ and for all $k \in [1, n]$, $\text{cd}(\chi'_k) \leq \text{cd}(\chi_k)$. Moreover, $\text{cd}(\gamma_i) \leq \text{cd}(\varphi)$,
5. for every $\chi \in \text{lvl}(\text{md}(\varphi) - i, \gamma_i)$, $\text{bd}(0, \chi) \leq (|\varphi| + 1)^4 \times 2^{|\varphi|}$,
6. for every $d > \text{md}(\varphi) - i$ and every $\chi \in \text{lvl}(d, \gamma_i)$, $\text{bd}(0, \chi) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$.

Each formula γ_i ($i \geq 1$) is built from γ_{i-1} by relying on Lemma 8.8 in order to translate every subformula occurring under the scope of i nested modalities into an equivalent formula in GML. Notice that, thanks to the properties (1)–(6), the formula $\gamma_{\text{md}(\varphi)}$ is a GML formula in good shape and such that $\varphi \equiv \gamma_{\text{md}(\varphi)}$, $\text{md}(\gamma_{\text{md}(\varphi)}) \leq \text{md}(\varphi)$ and $\text{max}_{\text{bd}}(\gamma_{\text{md}(\varphi)}) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$, which allows us to conclude the proof. In particular, $\varphi \equiv \gamma_{\text{md}(\varphi)}$ and $\text{md}(\gamma_{\text{md}(\varphi)}) \leq \text{md}(\varphi)$ holds directly by (1) and (3), respectively. Since $\text{lvl}(\text{md}(\varphi) - \text{md}(\varphi), \gamma_{\text{md}(\varphi)}) = \text{lvl}(0, \gamma_{\text{md}(\varphi)}) = \{\gamma_{\text{md}(\varphi)}\}$, from (2) we conclude that $\gamma_{\text{md}(\varphi)}$ is a GML formula in good shape. Lastly, from (5) and (6) we conclude that every subformula ψ of $\gamma_{\text{md}(\varphi)}$ is such that $\text{bd}(0, \psi) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$ which, directly by definition of $\text{max}_{\text{bd}}(\gamma_{\text{md}(\varphi)})$, implies that $\text{max}_{\text{bd}}(\gamma_{\text{md}(\varphi)}) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$.

In order to build the sequence $\gamma_0, \gamma_1, \dots, \gamma_{\text{md}(\varphi)}$, let us first check that the base case of $\gamma_0 = \varphi$ satisfies all the properties (1)–(6). The properties (1), (3) and (4) are straightforward. The property (2) is satisfied thanks to the assumption described at the beginning of the proof. The properties (5) and (6) are satisfied as, by definition of bd , for every subformula ψ of φ , $\text{bd}(0, \psi) \leq |\varphi|$. Here, recall that the coefficients k appearing in graded modalities $\Diamond_{\geq k}$ are encoded in unary. We conclude that $\gamma_0 = \varphi$ satisfies (1)–(6). Let us now assume that, given $i \in [0, \text{md}(\varphi) - 1]$, there is a formula γ_i satisfying (1)–(6). We show how to manipulate γ_i in order to produce the next formula of the sequence, i.e. γ_{i+1} .

Let $\text{gm}(\text{md}(\varphi) - (i + 1), \gamma_i) = \{\Diamond_{\geq k_1}\chi_1, \dots, \Diamond_{\geq k_n}\chi_n\}$. By definition of lvl together with the property (2), we have $\text{lvl}(\text{md}(\varphi) - i, \gamma_i) = \{\chi_1, \dots, \chi_n\}$ and $\chi_1 \wedge \dots \wedge \chi_n$ is a GML formula in good shape. Let us consider the set of formulae $\text{lvl}(\text{md}(\varphi) - (i + 1), \gamma_i) = \{\psi_1, \dots, \psi_m\}$. By definition of lvl , the formulae ψ_1, \dots, ψ_m are subformulae of γ_i occurring under the scope of $\text{md}(\varphi) - (i + 1)$ nested graded modalities. By definition of gm , this implies that for all $j \in [1, m]$, $\text{top}_{\text{gm}}(\psi_j) \subseteq \text{gm}(\text{md}(\varphi) - (i + 1), \gamma_i) = \{\Diamond_{\geq k_1}\chi_1, \dots, \Diamond_{\geq k_n}\chi_n\}$. Since χ_1, \dots, χ_n are GML formulae, every ψ_j is a formula of the fragment \mathcal{F} , which allows us to apply Lemma 8.8.

Let $\hat{k} = \max(k_1, \dots, k_n)$ and $\Phi = \{\chi_1, \dots, \chi_n\}$. For $j \in [1, m]$, by Lemma 8.8, there is $\tilde{\psi}_j$ in GML and in good shape s.t. $\psi_j \equiv \tilde{\psi}_j$, $\text{top}_{\text{gm}}(\tilde{\psi}_j) \subseteq \{\Diamond_{\geq j}\chi \mid j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}], \chi \in 2^\Phi\}$ and,

- a. $\text{md}(\tilde{\psi}_j) \leq 1 + \max(\{\text{md}(\chi_k) \mid k \in [1, n]\})$,
- b. $\text{bd}(0, \tilde{\psi}_j) \leq (1 + (\text{cd}(\psi_j) + 1) \times \hat{k})^2 \times 2^{n-1}$,
- c. for every $d \geq 1$, $\text{bd}(d, \tilde{\psi}_j) \leq \text{bd}(m-1, \chi_1 \wedge \dots \wedge \chi_n)$.

Let γ_{i+1} be the formula obtained from γ_i by replacing the occurrences of ψ_1, \dots, ψ_m occurring under the scope of $\text{md}(\varphi) - (i+1)$ nested graded modalities with the formulae $\tilde{\psi}_1, \dots, \tilde{\psi}_m$, respectively (for all $j \in [1, m]$, ψ_j is replaced with $\tilde{\psi}_j$), so that $\text{lvl}(\text{md}(\varphi) - (i+1), \gamma_{i+1}) = \{\tilde{\psi}_1, \dots, \tilde{\psi}_m\}$. We show that γ_{i+1} satisfies the properties (1)–(6).

γ_{i+1} **satisfies (1).** Directly from the definition of γ_{i+1} together with the fact that $\gamma_i \equiv \varphi$ and, for every $j \in [1, m]$, $\psi_j \equiv \tilde{\psi}_j$.

γ_{i+1} **satisfies (2).** Consider $\text{lvl}(\text{md}(\varphi) - (i+1), \gamma_{i+1}) = \{\tilde{\psi}_1, \dots, \tilde{\psi}_m\}$. We show that the formula $\hat{\psi} \stackrel{\text{def}}{=} \tilde{\psi}_1 \wedge \dots \wedge \tilde{\psi}_m$ is in GML in good shape. Clearly, since every $\tilde{\psi}_j$ ($j \in [1, m]$) is in GML , so is $\hat{\psi}$. In order to prove that $\hat{\psi}$ is in good shape, we must check that for all $d \in [0, \text{md}(\hat{\psi}) - 1]$, $\text{lvl}(d + 1, \hat{\psi})$ is a set of disjoint formulae. By definition, for all $j \in [1, m]$, $\text{top}_{\text{gm}}(\tilde{\psi}_j) \subseteq \{\Diamond_{\geq j} \chi \mid j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}], \chi \in 2^\Phi\}$. Therefore,

$$\text{top}_{\text{gm}}(\hat{\psi}) \subseteq \{\Diamond_{\geq j} \chi \mid j \in [1, (\text{cd}(\varphi) + 1) \times \hat{k}], \chi \in 2^\Phi\},$$

which implies that $\text{lvl}(1, \hat{\psi}) \subseteq 2^\Phi$. Directly from Lemma 7.10, this means that $\text{lvl}(1, \hat{\psi})$ is a set of disjoint formulae. Now, recall that $\Phi = \{\chi_1, \dots, \chi_n\}$, and thus every formula in 2^Φ is a conjunction of possibly negated formulae from Φ , where each formula among χ_1, \dots, χ_n occurs exactly once. This implies that, for all $\chi \in 2^\Phi$, $\text{top}_{\text{gm}}(\chi) = \text{top}_{\text{gm}}(\chi_1 \wedge \dots \wedge \chi_n)$. Together with the fact that $\chi_1 \wedge \dots \wedge \chi_n$ is in good shape (since γ_i satisfies the property (2)), this implies that every formula in 2^Φ is in good shape. This allows us to conclude that for every $d \in [1, \text{md}(\hat{\psi}) - 1]$, $\text{lvl}(d + 1, \hat{\psi})$ is a set of disjoint formulae. Together with the fact that $\text{lvl}(1, \hat{\psi})$ is a set of disjoint formulae, this concludes the proof of the property (2).

γ_{i+1} **satisfies (3).** Let $j \in [1, m]$ such that $\text{md}(\psi_j) = \max(\{\text{md}(\psi_i) \mid i \in [1, m]\})$. We show that for every $k \in [1, m]$, $\text{md}(\tilde{\psi}_k) \leq \text{md}(\psi_j)$. Since γ_{i+1} is obtained from γ_i by substituting every ψ_1, \dots, ψ_m occurring under the scope of $\text{md}(\varphi) - (i+1)$ nested graded modalities with the formulae $\tilde{\psi}_1, \dots, \tilde{\psi}_m$, this implies that

$$\begin{aligned} \text{md}(\gamma_{i+1}) &= \text{md}(\varphi) - (i+1) + \max(\{\text{md}(\tilde{\psi}_i) \mid i \in [1, m]\}) \\ &\leq \text{md}(\varphi) - (i+1) + \text{md}(\psi_j) = \text{md}(\gamma_i), \end{aligned}$$

which ends the proof directly from $\text{md}(\gamma_i) \leq \text{md}(\varphi)$ (recall that γ_i satisfies (3)).

To show that for every $k \in [1, m]$ $\text{md}(\tilde{\psi}_k) \leq \text{md}(\psi_j)$, it is sufficient to notice that, by definition, $\bigcup_{\psi \in \text{lvl}(\text{md}(\varphi) - (i+1), \gamma_i)} \text{top}_{\text{gm}}(\psi) = \text{gm}(\text{md}(\varphi) - (i+1), \gamma_i) = \{\Diamond_{\geq k_1} \chi_1, \dots, \Diamond_{\geq k_n} \chi_n\}$. As $\text{lvl}(\text{md}(\varphi) - (i+1), \gamma_i) = \{\psi_1, \dots, \psi_m\}$, this means that $\text{top}_{\text{gm}}(\psi_j)$ contains the formula among $\Diamond_{\geq k_1} \chi_1, \dots, \Diamond_{\geq k_n} \chi_n$, with maximal md, which in turn implies that $\text{md}(\psi_j)$ is at least $1 + \max(\{\text{md}(\chi_i) \mid i \in [1, n]\})$. Directly from (a), this allows us to conclude that for every $k \in [1, m]$, $\text{md}(\tilde{\psi}_k) \leq \text{md}(\psi_j)$.

γ_{i+1} **satisfies (4).** Given $j \in [0, \text{md}(\varphi) - (i+2)]$, we show that $\text{cd}(\gamma_{i+1}) \leq \text{cd}(\gamma_i)$ and there are $k_1, \dots, k_p \in \mathbb{N}$, χ'_1, \dots, χ'_p and $\chi''_1, \dots, \chi''_p$ such that $\text{gm}(j, \gamma_i) = \{\Diamond_{k_1} \chi'_1, \dots, \Diamond_{k_p} \chi'_p\}$, $\text{gm}(j, \gamma_{i+1}) = \{\Diamond_{k_1} \chi''_1, \dots, \Diamond_{k_p} \chi''_p\}$ and for all $k \in [1, p]$, $\text{cd}(\chi''_k) \leq \text{cd}(\chi'_k)$. This allows us to conclude that γ_{i+1} satisfies the property (4) directly from the fact that γ_i satisfies the property (4). Let $\text{gm}(j, \gamma_i) = \{\Diamond_{k_1} \chi'_1, \dots, \Diamond_{k_p} \chi'_p\}$. Since γ_{i+1} is obtained from γ_i by substituting every ψ_1, \dots, ψ_m occurring under the scope of $\text{md}(\varphi) - (i+1)$ nested graded modalities with the formulae $\tilde{\psi}_1, \dots, \tilde{\psi}_m$, and $j < \text{md}(\varphi) - (i+1)$. We conclude that there are formulae $\chi''_1, \dots, \chi''_p$ such that $\text{gm}(j, \gamma_{i+1}) = \{\Diamond_{k_1} \chi''_1, \dots, \Diamond_{k_p} \chi''_p\}$. As $\tilde{\psi}_1, \dots, \tilde{\psi}_m$ are formulae of graded modal logic and thus have composition degree 0, $\text{cd}(\gamma_{i+1}) \leq \text{cd}(\gamma_i)$ and for all $k \in [1, p]$, $\text{cd}(\chi''_k) \leq \text{cd}(\chi'_k)$, ending the proof of the property (4). In particular, given $k \in [1, p]$, χ''_k is obtained from χ'_k by substituting every ψ_i ($i \in [1, m]$) occurring

under the scope of $\text{md}(\varphi) - (i+1) - (j+1)$ nested graded modalities with the formula $\tilde{\psi}_i$. As expected, this means that $\text{gm}(\text{md}(\varphi) - (i+2), \gamma_{i+1}) = \{\Diamond_{k_1} \tilde{\psi}_1, \dots, \Diamond_{k_n} \tilde{\psi}_n\}$.

γ_{i+1} **satisfies (5).** We show that for all $\chi \in \text{lvl}(\text{md}(\varphi) - (i+1), \gamma_{i+1})$, $\text{bd}(0, \chi) \leq (|\varphi| + 1)^4 \times 2^{|\varphi|}$. By definition of γ_{i+1} , $\text{lvl}(\text{md}(\varphi) - (i+1), \gamma_{i+1}) = \{\tilde{\psi}_1, \dots, \tilde{\psi}_m\}$. So, let $j \in [1, m]$. Directly from section 8.2.2, $\text{bd}(0, \tilde{\psi}_j) \leq (1 + (\text{cd}(\tilde{\psi}_j) + 1) \times \hat{k})^2 \times 2^{n-1}$. In order to conclude the proof of (5), it is sufficient to show that $\text{cd}(\tilde{\psi}_j) \leq |\varphi|$, $\hat{k} \leq |\varphi|$ and $n \leq |\varphi|$.

Let us start with $\text{cd}(\tilde{\psi}_j) \leq |\varphi|$. By definition, $\tilde{\psi}_j$ belongs to $\text{lvl}(\text{md}(\varphi) - (i+1), \gamma_i)$ and therefore there is a coefficient k such that $\Diamond_{\geq k} \tilde{\psi}_j$ belongs to $\text{gm}(\text{md}(\varphi) - (i+2), \gamma_i)$. Thanks to the fact that γ_i satisfies the property (4), we conclude that there is a subformula ψ of φ such that $\Diamond_{\geq k} \psi$ belongs to $\text{gm}(\text{md}(\varphi) - (i+2), \varphi)$ and $\text{cd}(\tilde{\psi}_j) \leq \text{cd}(\psi)$. Since the size of a formula is always at least its composition degree, we conclude that $\text{cd}(\psi) \leq |\psi| \leq |\varphi|$, which allows us to conclude that $\text{cd}(\tilde{\psi}_j) \leq |\varphi|$.

Now, let us show that $\hat{k} \leq |\varphi|$ and $n \leq |\varphi|$. We recall that $\text{gm}(\text{md}(\varphi) - (i+1), \gamma_i) = \{\Diamond_{\geq k_1} \chi_1, \dots, \Diamond_{\geq k_n} \chi_n\}$, and that \hat{k} is the maximum coefficient among k_1, \dots, k_n . Again from the fact that γ_i satisfies the property (4), there are formulae χ'_1, \dots, χ'_n such that $\text{gm}(\text{md}(\varphi) - (i+1), \varphi) = \{\Diamond_{\geq k_1} \chi'_1, \dots, \Diamond_{\geq k_n} \chi'_n\}$ and, for all $k \in [1, n]$, $\text{cd}(\chi_k) \leq \text{cd}(\chi'_k)$. As the coefficient of graded modalities are encoded in unary, this implies that $\hat{k} \leq |\varphi|$. Lastly, since $\text{gm}(\text{md}(\varphi) - (i+1), \varphi)$ contains subformulae of φ , we derive that $|\varphi|$ is at least $\text{card}(\text{gm}(\text{md}(\varphi) - (i+1), \varphi)) = n$, concluding the proof of the property (5).

γ_{i+1} **satisfies (6).** We show that $\text{bd}(0, \chi) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$ holds for all $d > \text{md}(\varphi) - (i+1)$ and $\chi \in \text{lvl}(d, \gamma_{i+1})$. Let $d > \text{md}(\varphi) - (i+1)$ and $\chi \in \text{lvl}(d, \gamma_{i+1})$. By definition, χ is a subformula of γ_{i+1} occurring under the scope of d nested graded modalities. Since $d > \text{md}(\varphi) - (i+1)$ and $\text{lvl}(\text{md}(\varphi) - (i+1), \gamma_{i+1}) = \{\tilde{\psi}_1, \dots, \tilde{\psi}_m\}$, we conclude that χ occurs as a subformula of $\tilde{\psi}_j$, for some $j \in [1, m]$, under the scope of $d - (\text{md}(\varphi) - (i+1))$ nested graded modalities. More precisely, $\chi \in \text{lvl}(d - (\text{md}(\varphi) - (i+1)), \tilde{\psi}_j)$. We recall that, by definition of bd , for every $d' \geq 1$, $\text{bd}(d', \tilde{\psi}_j) = \max(\{\text{bd}(0, \tilde{\chi}) \mid \tilde{\chi} \in \text{lvl}(d', \tilde{\psi}_j)\})$. Thus, $\text{bd}(0, \chi) \leq \text{bd}(d - (\text{md}(\varphi) - (i+1)), \tilde{\psi}_j)$. Since $d - (\text{md}(\varphi) - (i+1)) \geq 1$ and from (c), we derive $\text{bd}(0, \chi) \leq \text{bd}(d - (\text{md}(\varphi) - i), \chi_1 \wedge \dots \wedge \chi_n)$. We divide the proof depending on whether $d - (\text{md}(\varphi) - i) = 0$.

case: $d - (\text{md}(\varphi) - i) = 0$. By definition of bd ,

$$\text{bd}(0, \chi_1 \wedge \dots \wedge \chi_n) \leq \text{bd}(0, \chi_1) + \dots + \text{bd}(0, \chi_n).$$

From $\text{lvl}(\text{md}(\varphi) - i, \gamma_i) = \{\chi_1, \dots, \chi_n\}$ together with the fact that γ_i satisfies the property (5), we have $\text{bd}(0, \chi_j) \leq (|\varphi| + 1)^4 \times 2^{|\varphi|}$, for every $j \in [1, n]$. Since $n \leq |\varphi|$ (as we showed during the proof of property (5)), we derive

$$\text{bd}(0, \chi_1 \wedge \dots \wedge \chi_n) \leq |\varphi| \times (|\varphi| + 1)^4 \times 2^{|\varphi|}.$$

From $\text{bd}(0, \chi) \leq \text{bd}(d - (\text{md}(\varphi) - i), \chi_1 \wedge \dots \wedge \chi_n)$, we have $\text{bd}(0, \chi) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$.

case: $d - (\text{md}(\varphi) - i) \geq 1$. By definition of bd ,

$$\begin{aligned} \text{bd}(d - (\text{md}(\varphi) - i), \chi_1 \wedge \dots \wedge \chi_n) &\leq \max(\{\text{bd}(d - (\text{md}(\varphi) - i), \chi_j) \mid j \in [1, n]\}) \\ &\leq \max(\{\max_{\text{bd}}(\chi_j) \mid j \in [1, n]\}). \end{aligned}$$

Since $\text{lvl}(\text{md}(\varphi) - i, \gamma_i) = \{\chi_1, \dots, \chi_n\}$ and $d > (\text{md}(\varphi) - i)$, from the fact that γ_i satisfies the property (6) we conclude that $\max_{\text{bd}}(\chi_j) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$, for all $j \in [1, n]$. Thus, $\text{bd}(0, \chi) \leq \text{bd}(d - (\text{md}(\varphi) - i), \chi_1 \wedge \dots \wedge \chi_n) \leq (|\varphi| + 1)^5 \times 2^{|\varphi|}$. \square

Lemmata 8.7 and 8.9 directly entail that $\text{ML}(\mathbb{I})$ has an exponential-size small model property.

Lemma 8.10. There is a polynomial Q such that every satisfiable φ in $\text{ML}(\mathbb{I})$ is satisfied by a pointed forest of size bounded by $2^{Q(|\varphi|)}$.

Fundamentally, from this lemma we conclude that the satisfiability problem of $\text{ML}(\mathbb{I})$ can be solved in alternating exponential time, using only polynomially many alternations.

Theorem 8.11. The satisfiability problem of $\text{ML}(\mathbb{I})$ is in AEXP_{POL} .

Proof (sketch). We already described the algorithm in Section 8.1.2. Briefly, consider a formula φ be in $\text{ML}(\mathbb{I})$. The (standard) algorithm is divided in two steps.

1. Guess a pointed forest (\mathcal{K}, w) of size exponential in $|\varphi|$. Thanks to Lemma 8.10, whenever φ is satisfiable, in this step we can guess a model satisfying it. Since the model has exponential size, it can be guessed in NEXPTIME . By definition, $\text{NEXPTIME} \subseteq \text{AEXP}_{\text{POL}}$ [41].
2. Call the model checking algorithm for $\text{ML}(\mathbb{I})$ on input (\mathcal{K}, w) and φ . By Proposition 8.4, the model checking algorithm has runtime (alternating) exponential in $|\varphi|$ (because of the size of (\mathcal{K}, w)), and uses an amount of alternations that is polynomial in $|\varphi|$. \square

8.3 $\text{ML}(\mathbb{I})$ IS AEXP_{POL} -COMPLETE

Following Theorem 8.11, in order to show that the satisfiability problem of $\text{ML}(\mathbb{I})$ is complete for AEXP_{POL} , it suffices to establish its AEXP_{POL} -hardness. In this section, we provide this result thanks to a logspace reduction from the satisfiability problem for propositional logic interpreted on team semantics, shown AEXP_{POL} -complete in [85, Theorem 4.9]. Despite being quite straightforward, this reduction allows us to draw a direct connection between $\text{ML}(\mathbb{I})$ and propositional logic in team semantics, two logics which can otherwise be understood as instantiations of BBI.

8.3.1 Propositional logic in team semantics.

Propositional logic in team semantics ($\text{PL}(\sim)$, a.k.a. propositional team logic) is a compositional semantics for propositional logic introduced by H. Hodges [88, 89], and extensively studied in recent years due to its application to database theory [135, 85, 136, 83]. We follow the presentation in [85] and refer the reader to [136] for a complete description of propositional team logics. The formulae of $\text{PL}(\sim)$ are defined from the grammar below (where $p \in \text{AP}$),

$$\begin{array}{lll} \pi := \top & (\text{true}) & \varphi := \pi & (\text{atomic formulae}) \\ | & p & | \quad \varphi \wedge \varphi \quad | \quad \sim \varphi & (\text{Boolean connectives}) \\ | & \dot{\neg} p & | \quad \varphi \dot{\vee} \varphi & (\text{team disjunction}) \end{array}$$

As we will see in a moment, whereas \wedge and \sim behave respectively as classical conjunction and classical negation, the atomic formula $\dot{\neg} p$ has a negation flavour without being equivalent to $\sim p$, and $\dot{\vee}$ is a spatial connective whose semantics follows. The small dot at the top of $\dot{\neg}$ and $\dot{\vee}$ is not present in [85]: it is added herein to avoid any confusion with the connectives in $\text{ML}(\mathbb{I})$.

On team semantics, the truth of a propositional formula is evaluated with respect to a set of valuations, called teams. Let $P \subseteq_{\text{fin}} \text{AP}$ be a finite set of atomic propositions. A *team* (over P) is a set of Boolean valuations $\mathbf{v} : P \rightarrow \{\top, \perp\}$. Notice that, since P is finite, every team is finite. We write $\mathfrak{T}, \mathfrak{T}_1, \dots$ to denote teams. A formula φ of $\text{PL}(\sim)$ is interpreted on a team \mathfrak{T} over a set of atomic propositions that includes those occurring in φ . The satisfaction relation \models , is given in Figure 8.3. As we can see, the atomic formula p is interpreted so that every valuation in the

$\mathfrak{T} \models \top$	always,
$\mathfrak{T} \models p$	iff for every valuation $\mathfrak{v} \in \mathfrak{T}$, $\mathfrak{v}(p) = \top$,
$\mathfrak{T} \models \dot{\neg}p$	iff for every valuation $\mathfrak{v} \in \mathfrak{T}$, $\mathfrak{v}(p) = \perp$,
$\mathfrak{T} \models \sim\varphi$	iff $\mathfrak{T} \not\models \varphi$,
$\mathfrak{T} \models \varphi \wedge \psi$	iff $\mathfrak{T} \models \varphi$ and $\mathfrak{T} \models \psi$,
$\mathfrak{T} \models \varphi \dot{\vee} \psi$	iff there are teams $\mathfrak{T}_1, \mathfrak{T}_2$ such that $\mathfrak{T} = \mathfrak{T}_1 \cup \mathfrak{T}_2$, $\mathfrak{T}_1 \models \varphi$ and $\mathfrak{T}_2 \models \psi$.

Figure 8.3: Satisfaction relation for $\text{PL}(\sim)$.

team must satisfy p . Similarly, $\dot{\neg}p$ states that every valuation in the team does not satisfy p . Lastly, the *team disjunction* $\varphi \dot{\vee} \psi$ is satisfied by the team \mathfrak{T} whenever it is possible to find two teams \mathfrak{T}_1 and \mathfrak{T}_2 , the first satisfying φ and the second satisfying ψ , whose union is \mathfrak{T} . Clearly, the team disjunction is similar to the operators $*$ and $\|$ from separation logics and ambient logics. In fact, despite being independently developed, one can easily see that $\text{PL}(\sim)$ instantiate the framework of BBI (Section 2.3.3). Let us be a bit more precise. We write \mathbb{T} for the set of all teams, and given two teams \mathfrak{T}_1 and \mathfrak{T}_2 , over the same set of atomic propositions, let $\mathfrak{T}_1 \circ \mathfrak{T}_2$ be defined as $\{\mathfrak{T}_1 \cup \mathfrak{T}_2\}$ whenever \mathfrak{T}_1 and \mathfrak{T}_2 are defined over the same set of atomic propositions, and otherwise $\mathfrak{T}_1 \circ \mathfrak{T}_2 \stackrel{\text{def}}{=} \emptyset$. The triple $(\mathbb{T}, \circ, \emptyset)$ forms a non-deterministic monoid (Definition 2.21). Moreover, if we internalise the semantics of the atomic formulae p and $\dot{\neg}p$ with the following evaluation $\llbracket \cdot \rrbracket$,

$$\llbracket p \rrbracket = \{\mathfrak{T} \text{ team} \mid \text{for every } \mathfrak{v} \in \mathfrak{T}, \mathfrak{v}(p) = \top\}, \quad \llbracket \dot{\neg}p \rrbracket = \{\mathfrak{T} \text{ team} \mid \text{for every } \mathfrak{v} \in \mathfrak{T}, \mathfrak{v}(p) = \perp\},$$

we can easily connect the semantics of $\text{PL}(\sim)$ with the one of BBI. This is formalised with the following proposition (whose proof is left to the reader), in which $\varphi[\dot{\vee} \leftarrow *]$ stands for the BBI formula obtained from the $\text{PL}(\sim)$ formula φ by substituting every occurrence of the team disjunction $\dot{\vee}$ with the separating conjunction $*$ from BBI.

Proposition 8.12. Let φ in $\text{PL}(\sim)$ and \mathfrak{T} be a team. $\mathfrak{T} \models \varphi$ in $\text{PL}(\sim)$ if and only if $\mathfrak{T} \models \varphi[\dot{\vee} \leftarrow *]$ in BBI interpreted on the non-deterministic monoid $(\mathbb{T}, \circ, \emptyset)$ and the evaluation $\llbracket \cdot \rrbracket$.

8.3.2 From Propositional Team Logic to $\text{ML}(\|)$.

Let us now discuss the reduction from the satisfiability problem of $\text{PL}(\sim)$ to the satisfiability problem of $\text{ML}(\|)$. A direct encoding of a team \mathfrak{T} into a pointed forest (\mathcal{K}, w) consists in having a correspondence between the Boolean valuations in \mathfrak{T} and the Boolean valuations of the children of w . In this case, the formula p can be translated to $\Box p$, whereas the formula $\dot{\neg}p$ corresponds to $\Box \neg p$. This encoding would work fine if there were no mismatch between the semantics for $\|$, which requires to partition the children of w in two disjoint sets, and the one for $\dot{\vee}$, where disjointness is not required. To handle this mismatch, when checking the satisfaction of φ in $\text{PL}(\sim)$ with n occurrences of $\dot{\vee}$, we impose that if a Boolean valuation occurs among the children of w , then it occurs in at least $n + 1$ children. This property must be maintained after applying $\dot{\vee}$ several times, always with respect to the number of occurrences of $\dot{\vee}$ in the subformula of φ that is evaluated. Non-disjointness of the teams is encoded by carefully separating the children of w having identical valuations.

We now formalise the reduction. Given an $\text{PL}(\sim)$ formula φ , we write $w_{\vee}(\varphi)$ for its $\dot{\vee}$ -*weight*, that is the number of occurrences of the team disjunction $\dot{\vee}$ in φ . Assume that we wish to translate φ , and that the formula is written with atomic propositions in $P = \{p_1, \dots, p_m\}$. Given $n = w_{\vee}(\varphi)$, we introduce a set $Q = \{q_1, \dots, q_{n+1}\}$ of auxiliary propositions disjoint from P . The elements of Q are used to distinguish different copies of the same Boolean valuation of a team. Given P and Q , which we fix throughout the section, we introduce the encoding of a team \mathfrak{T} with valuations $v : P \rightarrow \{\top, \perp\}$, in a pointed forest (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. Given a valuation $v : P \rightarrow \{\top, \perp\}$, we write $(\mathcal{K}, w) \models v$ whenever for every $p \in P$, $(\mathcal{K}, w) \models p$ iff $v(p) = \top$.

Definition 8.13 (Team encoding). Let S be a non-empty subset of $[1, n + 1]$. We say that (\mathcal{K}, w) encodes \mathfrak{T} with respect to S , written $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$, whenever

1. every world in $R(w)$ satisfies exactly one atomic proposition among q_1, \dots, q_{n+1} ,
2. for every Boolean valuation $v : P \rightarrow \{\top, \perp\}$, if there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models v$, then for every $k \in S$ there is $w'' \in R(w)$ such that $(\mathcal{K}, w'') \models v$ and $(\mathcal{K}, w'') \models q_k$.
3. for each valuation $v : P \rightarrow \{\top, \perp\}$, $v \in \mathfrak{T}$ iff there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models v$.

When $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$, the set of indices S represent a subset of Q , so that every valuation in \mathfrak{T} is witnessed by at least $\text{card}(S)$ children of w . Let us now devise formulae in $\text{ML}(\Box)$ that correspond to the first two properties highlighted in Definition 8.13. According to the first property, we require each child of w to satisfy exactly one element of Q . This can be easily done with the ML formula below, whose size is quadratic in n :

$$\text{unique}(Q) \stackrel{\text{def}}{=} \Box \bigvee_{k \in [1, n+1]} (q_k \wedge \bigwedge_{j \in [1, n+1] \setminus \{k\}} \neg q_j).$$

For the property (2) of the encoding, we design a formula that is satisfied if and only if, for every child of w satisfying a Boolean valuation v over (elements in) P , there are at least $\text{card}(S)$ children satisfying v , each of them satisfying a distinct symbol q_k , where $k \in S$. The formula $\text{copies}(S)$ below does the job:

$$\text{copies}(S) \stackrel{\text{def}}{=} \bigwedge_{j \in [1, n+1], k \in S} \neg \left(\Box q_j \mid (\Diamond_{=1} q_j \wedge \neg(\top \mid (\Diamond_{=1} q_j \wedge \Diamond_{=1} q_k \wedge \bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box p_i)))) \right),$$

where we recall that $\Diamond_{=1} \varphi$ stands for $\Diamond \varphi \wedge \neg(\Diamond \varphi \mid \Diamond \varphi)$. Let us check the correctness of $\text{copies}(S)$.

Lemma 8.14. Let $\emptyset \neq S \subseteq [1, n + 1]$ and (\mathcal{K}, w) be a pointed forest satisfying $\text{unique}(Q)$. We have $(\mathcal{K}, w) \models \text{copies}(S)$ if and only if for every valuation $v : P \rightarrow \{\top, \perp\}$, if there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models v$, then for every $k \in S$ there is $w'' \in R(w) \cap \mathcal{V}(q_k)$ such that $(\mathcal{K}, w'') \models v$.

Proof. Below, we assume $(\mathcal{K}, w) \models \text{unique}(Q)$, and thus that every world in $R(w)$ satisfies exactly one atomic proposition in $Q = \{q_1, \dots, q_n\}$.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{copies}(S)$. Let $v : P \rightarrow \{\top, \perp\}$, $w' \in R(w)$ such that $(\mathcal{K}, w') \models v$, and $k \in S$. We show that there is $w'' \in R(w)$ such that $(\mathcal{K}, w'') \models q_k$ and $(\mathcal{K}, w'') \models v$. From $(\mathcal{K}, w) \models \text{unique}(Q)$, there is $j \in [1, n + 1]$ such that $(\mathcal{K}, w') \models q_j$. From $(\mathcal{K}, w) \models \text{copies}(S)$, we have

$$(\mathcal{K}, w) \not\models \Box q_j \mid (\Diamond_{=1} q_j \wedge \neg(\top \mid \Diamond_{=1} q_j \wedge \Diamond_{=1} q_k \wedge \bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box p_i))). \quad (\dagger)$$

Consider the two Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K}_1 +_w \mathcal{K}_2 = \mathcal{K}$ and $R_1(w) = \{\tilde{w} \in R(w) \mid \tilde{w} \neq w'\text{ and }(\mathcal{K}, w') \models q_j\}$. By definition, $R_1(w)$ contains all the worlds in $R(w)$ that satisfy q_j , with the exception of w' , which is therefore the only world satisfying q_j in $R_2(w)$. We have $(\mathcal{K}_1, w) \models \Box q_j$ and $(\mathcal{K}_2, w) \models \Diamond_{=1} q_j$. From (\dagger),

we conclude that $(\mathcal{K}_2, \mathbf{w}) \models \top \mid \Diamond_{=1} q_j \wedge \Diamond_{=1} q_k \wedge \bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box q_i)$. By definition, this implies that there is a Kripke-style finite forest $\mathcal{K}'_2 = (\mathcal{W}, R'_2, \mathcal{V})$ such that $R'_2(\mathbf{w}) \subseteq R(\mathbf{w})$ and $(\mathcal{K}'_2, \mathbf{w}) \models \Diamond_{=1} q_j \wedge \Diamond_{=1} q_k \wedge \bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box q_i)$. From the satisfaction of $\Diamond_{=1} q_j$ and $\Diamond_{=1} q_k$ we conclude that $R'_2(\mathbf{w})$ contains two worlds \mathbf{w}_1 and \mathbf{w}_2 such that $(\mathcal{K}'_2, \mathbf{w}_1) \models q_j$ and $(\mathcal{K}'_2, \mathbf{w}_2) \models q_k$. Since \mathbf{w}' is the only world in $R_2(\mathbf{w})$ satisfying q_j , and $R'_2(\mathbf{w}) \subseteq R_2(\mathbf{w})$, we have $\mathbf{w}_1 = \mathbf{w}'$. From the satisfaction of the formula $\bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box q_i)$, this allows us to derive $(\mathcal{K}'_2, \mathbf{w}_2) \models \mathbf{v}$. Since \mathcal{K} and \mathcal{K}'_2 share the same valuation \mathcal{V} and $R'_2(\mathbf{w}) \subseteq R_2(\mathbf{w}) \subseteq R(\mathbf{w})$, we conclude that $\mathbf{w}_2 \in R(\mathbf{w}) \cap \mathcal{V}(q_k)$ and $(\mathcal{K}, \mathbf{w}_2) \models \mathbf{v}$.

(\Leftarrow): Suppose that for every valuation $\mathbf{v} : P \rightarrow \{\top, \perp\}$, if there is $\mathbf{w}' \in R(\mathbf{w})$ such that $(\mathcal{K}, \mathbf{w}') \models \mathbf{v}$, then for every $k \in S$ there is $\mathbf{w}'' \in R(\mathbf{w}) \cap \mathcal{V}(q_k)$ such that $(\mathcal{K}, \mathbf{w}'') \models \mathbf{v}$. *Ad absurdum*, assume $(\mathcal{K}, \mathbf{w}) \not\models \text{copies}(S)$, and thus there are $j \in [1, n+1]$, $k \in S$ and Kripke-style forest $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $(\mathcal{K}_1, \mathbf{w}) \models \Box p_j$ and

1. $(\mathcal{K}_2, \mathbf{w}) \models \Diamond_{=1} q_j$,
2. $(\mathcal{K}_2, \mathbf{w}) \models \neg(\top \mid \Diamond_{=1} q_j \wedge \Diamond_{=1} q_k \wedge \bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box q_i))$.

From (1), there is $\mathbf{w}' \in R_2(\mathbf{w})$ such that $(\mathcal{K}_2, \mathbf{w}') \models q_j$. Notice that it must hold that $j \neq k$. Indeed, *ad absurdum*, suppose $j = k$ and consider the two Kripke-style finite forests \mathcal{K}'_1 and $\mathcal{K}'_2 = (\mathcal{W}, R'_2, \mathcal{V})$ such that $\mathcal{K}_2 = \mathcal{K}'_1 +_{\mathbf{w}} \mathcal{K}'_2$ and $R'_2(\mathbf{w}) = \{\mathbf{w}'\}$. We have $(\mathcal{K}'_1, \mathbf{w}) \models \top$ and $(\mathcal{K}'_2, \mathbf{w}) \models \Diamond_{=1} q_j \wedge \Diamond_{=1} q_k \wedge \bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box q_i)$, which contradicts (2). Now, let us consider the Boolean valuation $\mathbf{v} : P \rightarrow \{\top, \perp\}$ such that $(\mathcal{K}_2, \mathbf{w}') \models \mathbf{v}$. From $R_2(\mathbf{w}) \subseteq R(\mathbf{w})$ and the fact that \mathcal{K}_2 and \mathcal{K} share the same valuation \mathcal{V} , we conclude that $\mathbf{w}' \in R(\mathbf{w})$. From the hypotheses, this implies that there is a world $\mathbf{w}'' \in R(\mathbf{w})$ such that $(\mathcal{K}, \mathbf{w}'') \models q_k$ and $(\mathcal{K}, \mathbf{w}'') \models \mathbf{v}$. Since $j \neq k$ and $(\mathcal{K}_1, \mathbf{w}) \models \Box q_j$, we conclude that $\mathbf{w}'' \notin R_1(\mathbf{w})$ and therefore $\mathbf{w}'' \in R_2(\mathbf{w})$. We consider the Kripke-style finite forests \mathcal{K}'_1 and $\mathcal{K}'_2 = (\mathcal{W}, R'_2, \mathcal{V})$ such that $\mathcal{K}_2 = \mathcal{K}'_1 +_{\mathbf{w}} \mathcal{K}'_2$ and $R'_2(\mathbf{w}) = \{\mathbf{w}', \mathbf{w}''\}$. Since $j \neq k$ and $(\mathcal{K}, \mathbf{w}) \models \text{unique}(Q)$, we have $(\mathcal{K}'_2, \mathbf{w}) \models \Diamond_{=1} q_j$ (as $\mathbf{w}' \in \mathcal{V}(q_j)$) and $(\mathcal{K}'_2, \mathbf{w}) \models \Diamond_{=1} q_k$ (as $\mathbf{w}'' \in \mathcal{V}(q_k)$). Since \mathcal{K} and \mathcal{K}'_2 share the same valuation \mathcal{V} , for every $\tilde{\mathbf{w}} \in R'_2(\mathbf{w}) = \{\mathbf{w}', \mathbf{w}''\}$ we have $(\mathcal{K}'_2, \tilde{\mathbf{w}}) \models \mathbf{v}$. Thus, $(\mathcal{K}'_2, \mathbf{w}) \models \bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box q_i)$. However, together with $(\mathcal{K}'_1, \mathbf{w}) \models \top$, this allows us to deduce that $(\mathcal{K}_2, \mathbf{w}) \models \top \mid \Diamond_{=1} q_j \wedge \Diamond_{=1} q_k \wedge \bigwedge_{i \in [1, m]} (\Diamond p_i \Rightarrow \Box q_i)$, in contradiction with (2). We conclude that $(\mathcal{K}, \mathbf{w}) \models \text{copies}(S)$. \square

Before defining the translation map τ from $\text{PL}(\sim)$ to $\text{ML}(\|)$, we need to describe how worlds encoding the same Boolean valuation are split when dealing with the operator $\dot{\vee}$. Roughly speaking, given a formula $\varphi \dot{\vee} \psi$, we want to partition the set of indices $S \subseteq [1, n+1]$ into two sets S_1 and S_2 whose cardinalities imply the existence of enough copies of a Boolean valuation required to successfully evaluate φ and ψ . To do so, we introduce an auxiliary choice function \mathbf{g} that takes as arguments S and $n_1, n_2 \in \mathbb{N}$, with $\text{card}(S) \geq n_1 + n_2$, and returns a pair of sets (S_1, S_2) such that S_1 and S_2 partition S , $\text{card}(S_1) \geq n_1$ and $\text{card}(S_2) \geq n_2$. The map \mathbf{g} is instrumental to decide how to split S into two disjoint subsets respecting basic cardinality constraints. We are now ready to introduce the translation map τ from $\text{PL}(\sim)$ to $\text{ML}(\|)$, which is defined in Figure 8.4. The translation is straightforward, and whenever $\text{card}(S) \leq |\varphi|$, $\tau(\varphi, S)$ has size quadratic in $|\varphi|$. For instance, whereas $\mathfrak{T} \models p$ requires every valuation in \mathfrak{T} to satisfy p , $(\mathcal{K}, \mathbf{w}) \models \tau(p, S)$ requires p to be satisfied by all the children of \mathbf{w} that satisfy a propositional symbol corresponding to an index of S . To translate the team disjunction $\varphi \dot{\vee} \psi$, we simply rely on the choice function \mathbf{g} in order to split S into suitable sets of indices S_1 and S_2 that allows to correctly evaluate the translations of φ and ψ , depending on the number of team disjunctions they contain. We show that if $\mathfrak{T} \triangleright_S (\mathcal{K}, \mathbf{w})$, then $\mathfrak{T} \models \varphi$ and $(\mathcal{K}, \mathbf{w}) \models \tau(\varphi, S)$ agree.

$$\begin{aligned}
\tau(p, S) &\stackrel{\text{def}}{=} \square((\bigvee_{j \in S} q_j) \Rightarrow p), \\
\tau(\dot{\neg}p, S) &\stackrel{\text{def}}{=} \square((\bigvee_{j \in S} q_j) \Rightarrow \neg p), \\
\tau(\varphi_1 \wedge \varphi_2, S) &\stackrel{\text{def}}{=} \tau(\varphi_1, S) \wedge \tau(\varphi_2, S), \\
\tau(\sim\varphi, S) &\stackrel{\text{def}}{=} \neg\tau(\varphi, S), \\
\tau(\varphi_1 \dot{\vee} \varphi_2, S) &\stackrel{\text{def}}{=} (\tau(\varphi_1, S_1) \wedge \text{copies}(S_1)) \parallel (\tau(\varphi_2, S_2) \wedge \text{copies}(S_2)), \\
&\quad \text{where } (S_1, S_2) = \mathbf{g}(S, w_{\dot{\vee}}(\varphi_1) + 1, w_{\dot{\vee}}(\varphi_2) + 1).
\end{aligned}$$

Figure 8.4: Translation from $\text{PL}(\sim)$ to $\text{ML}(\mathbf{I})$.

Lemma 8.15. Let $\emptyset \neq S \subseteq [1, n + 1]$ and $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$. For every $\text{PL}(\sim)$ formula ψ built over P and such that $w_{\dot{\vee}}(\psi) \leq \text{card}(S) - 1$, we have $\mathfrak{T} \models \psi$ iff $(\mathcal{K}, w) \models \tau(\psi, S)$.

Proof. The proof is by structural induction on ψ .

base case: $\psi = p_i$, where $i \in [1, m]$. (\Rightarrow): Suppose $\mathfrak{T} \models p_i$. So, for every $\mathbf{v} \in \mathfrak{T}$, $\mathbf{v}(p_i) = \top$. *Ad absurdum*, suppose that $(\mathcal{K}, w) \not\models \tau(p_i, S)$, and so there is $k \in S$ and $w' \in R(w) \cap \mathcal{V}(q_k)$ such that $(\mathcal{K}, w') \not\models p_i$. Let \mathbf{v}' be the valuation over P satisfied by w' . In particular, we have $\mathbf{v}'(p_i) = \perp$. From the property (3) of the encoding (Definition 8.13), $\mathbf{v}' \in \mathfrak{T}$. However, from $\mathfrak{T} \models p_i$, this implies $\mathbf{v}(p_i) = \top$: a contradiction. Thus, $(\mathcal{K}, w) \models \tau(p_i, S)$.

(\Leftarrow): Similar to the other direction. Suppose $(\mathcal{K}, w) \models \tau(p_i, S)$, and so for all $w' \in R(w)$, if there is $j \in S$ such that $(\mathcal{K}, w') \models q_j$ then $(\mathcal{K}, w') \models p_i$. *Ad absurdum*, suppose that $\mathfrak{T} \not\models p_i$, and therefore there is $\mathbf{v} \in \mathfrak{T}$ such that $\mathbf{v}(p_i) = \perp$. From the property (3) of the encoding, there is $w'_1 \in R(w)$ such that $(\mathcal{K}, w'_1) \models \mathbf{v}$. In particular, $(\mathcal{K}, w'_1) \not\models p_i$. From the property (2) of the encoding, for every $j \in S$, there is $w'_2 \in R(w)$ such that $(\mathcal{K}, w'_2) \models \mathbf{v}$ and $(\mathcal{K}, w'_2) \models q_j$. However, as $S \neq \emptyset$, we derive that such a world w'_2 exists, and moreover $(\mathcal{K}, w'_2) \not\models p_i$. This contradicts the fact that $(\mathcal{K}, w) \models \tau(p_i, S)$. Therefore, $\mathfrak{T} \models p_i$.

base case: $\psi = \dot{\neg}p_i$, where $i \in [1, m]$. Similar to the case $\psi = p_i$.

The cases in the induction step for which the outermost connective of ψ is a Boolean connective are by an easy verification, and thus omitted.

induction step: $\psi = \psi_1 \dot{\vee} \psi_2$. Observe that $w_{\dot{\vee}}(\psi) = w_{\dot{\vee}}(\psi_1) + w_{\dot{\vee}}(\psi_2) + 1$ and recall that $w_{\dot{\vee}}(\psi) \leq \text{card}(S) - 1$. Consequently, $w_{\dot{\vee}}(\psi_1) + w_{\dot{\vee}}(\psi_2) + 2 \leq \text{card}(S)$, which implies that $\mathbf{g}(S, w_{\dot{\vee}}(\psi_1) + 1, w_{\dot{\vee}}(\psi_2) + 1)$ is defined. Let $(S_1, S_2) = \mathbf{g}(S, w_{\dot{\vee}}(\psi_1) + 1, w_{\dot{\vee}}(\psi_2) + 1)$. From the definition of the choice function \mathbf{g} , the sets S_1 and S_2 partition S , $\text{card}(S_1) \geq w_{\dot{\vee}}(\psi_1) + 1$ and $\text{card}(S_2) \geq w_{\dot{\vee}}(\psi_2) + 1$.

(\Rightarrow): Suppose $\mathfrak{T} \models \psi_1 \dot{\vee} \psi_2$. By definition, there are two teams \mathfrak{T}_1 and \mathfrak{T}_2 such that $\mathfrak{T} = \mathfrak{T}_1 \cup \mathfrak{T}_2$, $\mathfrak{T}_1 \models \varphi_1$ and $\mathfrak{T}_2 \models \varphi_2$. Let us define $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ so that the set $R(w)$ is divided in $R_1(w)$ and $R_2(w)$ as follows:

$$R_1(w) \stackrel{\text{def}}{=} \left\{ w' \in R(w) \mid \begin{array}{l} \text{there is } \mathbf{v} \in \mathfrak{T}_1 \text{ such that } (\mathcal{K}, w') \models \mathbf{v} \text{ and} \\ (\mathbf{v} \notin \mathfrak{T}_2 \text{ or there is } j \in S_1 \text{ s.t. } (\mathcal{K}, w') \models q_j) \end{array} \right\},$$

$$R_2(w) \stackrel{\text{def}}{=} R(w) \setminus R_1(w).$$

By definition of $+_w$, we have $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$. We show that $\mathfrak{T}_1 \triangleright_{S_1} (\mathcal{K}_1, w)$ and $\mathfrak{T}_2 \triangleright_{S_2} (\mathcal{K}_2, w)$. We consider the three properties of Definition 8.13 separately.

property (1). From $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$, every world in $R(w)$ satisfies exactly one atomic proposition among q_1, \dots, q_{n+1} . Since $R_1(w) \cup R_2(w) = R(w)$ and the Kripke-style finite forests \mathcal{K} , \mathcal{K}_1 and \mathcal{K}_2 share the same valuation \mathcal{V} , the same holds for every world in $R_1(w)$ and $R_2(w)$. Thus, (1) is satisfied.

property (3). We show that for all $i \in \{1, 2\}$ and valuation $v : P \rightarrow \{\top, \perp\}$, $v \in \mathfrak{T}_i$ iff there is $w' \in R_i(w)$ s.t. $(\mathcal{K}_i, w') \models v$. The proof is divided depending on the value of i .

case: $i = 1$. (\Rightarrow): Suppose $v \in \mathfrak{T}_1$. From $\mathfrak{T}_1 \subseteq \mathfrak{T}$, $v \in \mathfrak{T}$. From $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$ (property (3)), there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models v$. From $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$ (property (2)), together with $S_1 \subseteq S$, for every $j \in S_1$ there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models v$ and $(\mathcal{K}, w') \models q_j$. Since $S_1 \neq \emptyset$, there are $w' \in R(w)$ and $j \in S_1$ such that $(\mathcal{K}, w') \models v$ and $(\mathcal{K}, w') \models q_j$, which implies $w' \in R_1(w)$ directly by definition of $R_1(w)$. Since \mathcal{K} and \mathcal{K}_1 share the same valuation \mathcal{V} , we conclude that $(\mathcal{K}_1, w') \models \mathcal{V}$.

(\Leftarrow): Direct from the definition of $R_1(w)$.

case: $i = 2$. (\Rightarrow): Suppose $v \in \mathfrak{T}_2$. From $\mathfrak{T}_2 \subseteq \mathfrak{T}$, $v \in \mathfrak{T}$. Similarly to the previous case, from $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$ (properties (3) and (2)), together with $S_2 \subseteq S$, this implies that for every $j \in S_1$ there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models v$ and $(\mathcal{K}, w') \models q_j$. Since $S_2 \neq \emptyset$, this means that there are $w' \in R(w)$ and $j \in S_2$ such that $(\mathcal{K}, w') \models v$ and $(\mathcal{K}, w') \models q_j$. Since S_1 is disjoint from S_2 , together with the property (1) showed above, we conclude that for every $k \in S_1$, $(\mathcal{K}, w') \not\models q_k$. Together with $v \in \mathfrak{T}_2$ and by definition of $R_1(w)$, this implies $w' \notin R_1(w)$. From $w' \in R(w)$ and by definition of $R_2(w)$, we have $w' \in R_2(w)$. Since \mathcal{K} and \mathcal{K}_2 share the same valuation \mathcal{V} , we conclude that $(\mathcal{K}_2, w') \models v$.

(\Leftarrow): Suppose that there is $w' \in R_2(w)$ such that $(\mathcal{K}_2, w') \models v$. From $R_2(w) \subseteq R(w)$ we conclude $w' \in R(w)$ and, since \mathcal{K} and \mathcal{K}_2 share the same valuation \mathcal{V} , $(\mathcal{K}, w') \models v$. From $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$ (property (3)), $v \in \mathfrak{T}$. *Ad absurdum*, suppose $v \notin \mathfrak{T}_2$. From $\mathfrak{T}_1 \cup \mathfrak{T}_2 = \mathfrak{T}$, this implies $v \in \mathfrak{T}_1$. However, by definition of $R_1(w)$, this allows us to derive $w' \in R_1(w)$, in contradiction with the disjointness of $R_1(w)$ and $R_2(w)$. Thus, $v \in \mathfrak{T}_2$.

property (2). Given $i \in \{1, 2\}$, we show that for all Boolean valuations $v : P \rightarrow \{\top, \perp\}$, if there is $w' \in R_i(w)$ such that $(\mathcal{K}_i, w') \models v$, then for all $k \in S_i$ there is $w'' \in R_i(w)$ such that $(\mathcal{K}_i, w'') \models v$ and $(\mathcal{K}_i, w'') \models q_k$. The proof is divided depending on the value of i .

case: $i = 1$. Let $w' \in R_1(w)$ such that $(\mathcal{K}_1, w') \models v$ and let $k \in S_1$. By definition of $R_1(w)$, $v \in \mathfrak{T}_1$. From $R_1(w) \subseteq R(w)$, $w' \in R(w)$ and, since \mathcal{K} and \mathcal{K}_1 share the same valuation \mathcal{V} , $(\mathcal{K}, w') \models v$. From $S_1 \subseteq S$ together with $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$ (property (2)), there is $w'' \in R(w)$ such that $(\mathcal{K}, w'') \models v$ and $(\mathcal{K}, w'') \models q_k$. Together with $v \in \mathfrak{T}_1$, by definition of $R_1(w)$, we conclude that $w'' \in R_1(w)$ and, since \mathcal{K} and \mathcal{K}_1 share the same valuation \mathcal{V} , $(\mathcal{K}_1, w'') \models v$ and $(\mathcal{K}_1, w'') \models q_k$.

case: $i = 2$. Let $w' \in R_2(w)$ such that $(\mathcal{K}_2, w') \models v$ and let $k \in S_2$. From the satisfaction of the property (3) shown above, we conclude that $v \in \mathfrak{T}_2$. From $R_2(w) \subseteq R(w)$, $w' \in R(w)$ and, since \mathcal{K} and \mathcal{K}_2 share the same valuation \mathcal{V} , $(\mathcal{K}, w') \models v$. From $S_2 \subseteq S$ together with $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$ (property (2)), there is $w'' \in R(w)$ such that $(\mathcal{K}, w'') \models v$ and $(\mathcal{K}, w'') \models q_k$. Since S_1 is disjoint from S_2 , together with the property (1) showed above, we conclude that for every $j \in S_1$, $(\mathcal{K}, w'') \not\models q_j$. Together with $v \in \mathfrak{T}_2$ and by definition of $R_1(w)$, this implies $w'' \notin R_1(w)$. From $w'' \in R(w)$ and by definition of $R_2(w)$, we have $w'' \in R_2(w)$. Since \mathcal{K} and \mathcal{K}_2 share the same valuation \mathcal{V} , we conclude that $(\mathcal{K}_2, w'') \models v$ and $(\mathcal{K}_2, w'') \models q_k$.

We conclude that $\mathfrak{T}_1 \triangleright_{S_1} (\mathcal{K}_1, w)$ and $\mathfrak{T}_2 \triangleright_{S_2} (\mathcal{K}_2, w)$. Moreover, by definition of S_1 and S_2 , $w_V(\psi_1) \leq \text{card}(S_1) - 1$ and $w_V(\psi_2) \leq \text{card}(S_2) - 1$. We apply the induction hypothesis, and

derive that $(\mathcal{K}_1, w) \models \tau(\psi_1, S_1)$ and $(\mathcal{K}_2, w) \models \tau(\psi_2, S_2)$. Moreover, from $\mathfrak{T}_1 \triangleright_{S_1} (\mathcal{K}_1, w)$ and $\mathfrak{T}_2 \triangleright_{S_2} (\mathcal{K}_2, w)$, by Lemma 8.14, $(\mathcal{K}_1, w) \models \text{copies}(S_1)$ and $(\mathcal{K}_2, w) \models \text{copies}(S_2)$. Indeed, notice that the property (1) of the encoding implies $(\mathcal{K}_i, w) \models \text{unique}(Q)$ ($i \in \{1, 2\}$). From $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$, we conclude:

$$(\mathcal{K}, w) \models (\tau(\psi_1, S_1) \wedge \text{copies}(S_1)) \| (\tau(\psi_2, S_2) \wedge \text{copies}(S_2)).$$

(\Leftarrow): Suppose $(\mathcal{K}, w) \models (\tau(\psi_1, S_1) \wedge \text{copies}(S_1)) \| (\tau(\psi_2, S_2) \wedge \text{copies}(S_2))$. By definition, there are two Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$, $(\mathcal{K}_1, w) \models \tau(\psi_1, S_1) \wedge \text{copies}(S_1)$ and $(\mathcal{K}_2, w) \models \tau(\psi_2, S_2) \wedge \text{copies}(S_2)$. Let us define two teams \mathfrak{T}_1 and \mathfrak{T}_2 as follows:

$$\mathfrak{T}_1 \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathfrak{T} \mid (\mathcal{K}_1, w') \models \mathbf{v}, \text{ for some } w' \in R_1(w)\},$$

$$\mathfrak{T}_2 \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathfrak{T} \mid (\mathcal{K}_2, w') \models \mathbf{v}, \text{ for some } w' \in R_2(w)\}.$$

First of all, let us show that $\mathfrak{T}_1 \cup \mathfrak{T}_2 = \mathfrak{T}$. By definition, $\mathfrak{T}_1 \cup \mathfrak{T}_2 \subseteq \mathfrak{T}$, and thus we only need to show the other inclusion. Let $\mathbf{v} \in \mathfrak{T}$. From $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$ (property (3)), there is $w' \in R(w)$ such that $(\mathcal{K}, w) \models \mathbf{v}$. From $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$, there is $i \in \{1, 2\}$ such that $w' \in R_i(w)$. Since \mathcal{K} and \mathcal{K}_i share the same valuation \mathcal{V} , we have $(\mathcal{K}_i, w') \models \mathbf{v}$. By definition of \mathfrak{T}_i , $w' \in \mathfrak{T}_i$. So, $\mathfrak{T}_1 \cup \mathfrak{T}_2 = \mathfrak{T}$.

In order to conclude the proof, we show that for every $i \in \{1, 2\}$, $\mathfrak{T}_i \triangleright_{S_i} (\mathcal{K}_i, w)$. Indeed, from $w_{\dot{\vee}}(\psi_1) \leq \text{card}(S_1) - 1$ and $w_{\dot{\vee}}(\psi_2) \leq \text{card}(S_2) - 1$, this allows us to apply the induction hypothesis, deriving $\mathfrak{T}_1 \models \psi_1$ and $\mathfrak{T}_2 \models \psi_2$, which in turn implies $\mathfrak{T} \models \psi_1 \dot{\vee} \psi_2$. To show that $\mathfrak{T}_i \triangleright_{S_i} (\mathcal{K}_i, w)$, we consider the three properties of Definition 8.13 separately.

property (1). As in the left to right direction of the proof, this follows directly from $\mathfrak{T} \triangleright_S (\mathcal{K}, w)$, together with the fact that $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$.

property (2). Thanks to the property (1), we have $(\mathcal{K}_i, w) \models \text{unique}(Q)$. Thus, property (2) follows directly from $(\mathcal{K}_i, w) \models \text{copies}(S_i)$, as we apply Lemma 8.14.

property (3). Directly from the definition of \mathfrak{T}_i . □

Lemma 8.15 allows us to complete the reduction from the satisfiability problem of $\text{PL}(\sim)$ to the satisfiability problem of $\text{ML}(\|)$, as formalised in the lemma below.

Lemma 8.16. Let φ be in $\text{PL}(\sim)$ with n occurrences of $\dot{\vee}$ and built upon p_1, \dots, p_m . We have, φ is satisfiable iff so is $\text{unique}(\{q_1, \dots, q_{n+1}\}) \wedge \text{copies}([1, n+1]) \wedge \tau(\varphi, [1, n+1])$.

Proof. The proof is by easy verification, while relying on Lemma 8.15.

(\Rightarrow): Suppose that φ is satisfiable, and let $\mathfrak{T} = \{\mathbf{v}_1, \dots, \mathbf{v}_L\}$ be a team satisfying it. We consider the Kripke-style finite forest $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ such that

$$\mathcal{W} \stackrel{\text{def}}{=} \{0\} \cup ([1, L] \times [1, n+1]),$$

$$R \stackrel{\text{def}}{=} \{(0, (i, j)) \mid (i, j) \in [1, L] \times [1, n+1]\},$$

and \mathcal{V} is a valuation such that,

- for all $j \in [1, n+1]$, $\mathcal{V}(q_j) \stackrel{\text{def}}{=} [1, L] \times \{j\}$,
- for all $s \in [1, m]$, $\mathcal{V}(p_s) \stackrel{\text{def}}{=} \{(i, j) \in [1, L] \times [1, n+1] \mid \mathbf{v}_i(p_s) = \top\}$.

Let $w = 0$. We show that $\mathfrak{T} \triangleright_{[1, n+1]} (\mathcal{K}, w)$, which entails $(\mathcal{K}, w) \models \text{unique}(Q) \wedge \text{copies}([1, n+1])$ directly from the first and second properties of the encoding (Definition 8.13), and $(\mathcal{K}, w) \models \tau(\varphi, [1, n+1])$ by Lemma 8.15. To show $\mathfrak{T} \triangleright_{[1, n+1]} (\mathcal{K}, w)$ we consider the three properties of Definition 8.13 separately.

property (1). Let $w' \in R(w)$. By definition of \mathcal{K} , there is $(i, j) \in [1, L] \times [1, n + 1]$ such that $w' = (i, j)$. By definition of \mathcal{V} , $w' \in \mathcal{V}(q_j)$ and for every $k \in [1, n + 1] \setminus \{j\}$, $w' \notin \mathcal{V}(q_k)$.

property (2). Let $v : P \rightarrow \{\top, \perp\}$ and $w' \in R(w)$ such that $(\mathcal{K}, w') \models v$. Let $k \in [1, n + 1]$.

We show that there is $w'' \in R(w)$ such that $(\mathcal{K}, w'') \models v$ and $(\mathcal{K}, w'') \models q_k$. By definition of \mathcal{K} , there is $(i, j) \in [1, L] \times [1, n + 1]$ such that $w' = (i, j)$. Given $s \in [1, m]$, by definition of $\mathcal{V}(p_s)$ we have $(i, j) \in \mathcal{V}(p_s)$ if and only if $(i, k) \in \mathcal{V}(p_s)$. Let $w'' = (i, k)$. We have $(\mathcal{K}, w'') \models v$ and, by definition of \mathcal{V} , $w'' \in \mathcal{V}(q_k)$.

property (3). Directly from the definition of \mathcal{V} , given $w' = (i, j) \in [1, L] \times [1, n + 1]$, we have $(\mathcal{K}, w') \models v_i$. By definition of R , $w' \in R(w)$, concluding the proof.

(\Leftarrow): Suppose that $\text{unique}(Q) \wedge \text{copies}([1, n + 1]) \wedge \tau(\varphi, [1, n + 1])$ is satisfiable, and let (\mathcal{K}, w) be a pointed forest satisfying it, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. We define the team \mathfrak{T} below:

$$\mathfrak{T} \stackrel{\text{def}}{=} \{v : P \rightarrow \{\top, \perp\} \mid \text{there are } w' \in R(w) \text{ and } k \in [1, n + 1] \text{ s.t. } (\mathcal{K}, w') \models v \text{ and } (\mathcal{K}, w') \models q_k\}.$$

It is easy to check that $\mathfrak{T} \triangleright_{[1, n + 1]} (\mathcal{K}, w)$. The property (1) of Definition 8.13 follows directly from $(\mathcal{K}, w) \models \text{unique}(Q)$, whereas the property (2) follows from $(\mathcal{K}, w) \models \text{copies}([1, n + 1])$, by Lemma 8.14. Lastly, the property (3) holds directly from the definition of \mathfrak{T} . Thanks to $\mathfrak{T} \triangleright_{[1, n + 1]} (\mathcal{K}, w)$, from $(\mathcal{K}, w) \models \tau(\varphi, [1, n + 1])$ and by Lemma 8.15, we conclude that $\mathfrak{T} \models \varphi$. \square

Quite interestingly, the formulae in $\text{ML}(\mathbf{I})$ involved in Lemma 8.16 have modal depth at most one, which allows us to conclude that $\text{ML}(\mathbf{I})$ is AEXP_{POL}-hard already for formulae where the modalities \diamond are not imbricated. Together with the AEXP_{POL} upper bound obtained in Theorem 8.11, we conclude that the satisfiability problem of $\text{ML}(\mathbf{I})$ is AEXP_{POL}-complete.

Theorem 8.17. The satisfiability problem for $\text{ML}(\mathbf{I})$ is AEXP_{POL}-complete. It is already AEXP_{POL}-hard for the fragment of $\text{ML}(\mathbf{I})$ formulae with modal depth at most 1.

Theorem 8.17 concludes our study of $\text{ML}(\mathbf{I})$: compared to graded modal logic, we now know that $\text{ML}(\mathbf{I})$ has the same expressive power (Chapter 7), whereas the complexity of its satisfiability problem is higher: it is AEXP_{POL}-complete, instead of PSPACE-complete.

8.4 AN AEXP_{POL}-COMPLETE STATIC AMBIENT LOGIC

Throughout Chapters 7 and 8, we studied $\text{ML}(\mathbf{I})$ as a way of analysing the composition operator \mathbf{I} of ambient logic [39], without directly dealing with the calculus of Mobile Ambients [40]. To complete the picture, in this section we place $\text{ML}(\mathbf{I})$ in the context of a standard ambient logics. In particular, we consider a very expressive fragment of *Static Ambient Logic* (**SAL**) [103] whose satisfiability problem has been shown decidable and conjectured to be in PSPACE (see [34, Section 6]), and invalidate this conjecture by showing that it is in fact AEXP_{POL}-hard. More precisely, we show the AEXP_{POL}-completeness of the satisfiability problem for the intentional fragment of **SAL** [103], here denoted **SAL**(\mathbf{I}), by designing semantically faithful reductions between the satisfiability problem for $\text{ML}(\mathbf{I})$ and the one of **SAL**(\mathbf{I}), in both directions.

Let us start by introducing the syntax and semantics of the static ambient logic from [34]. As already stated in Chapter 7, the fundamental entity of ambient logic is the *ambient*. Each ambient has a name, taken from a countably infinite set of names. Without loss of generality, we assume that ambients are named after atomic propositions from AP . The formulae φ of the logic are defined from the grammar below (where $n \in \text{AP}$ is an *ambient name*):

INFORMATION TREES	STRUCTURAL EQUIVALENCE
$T := 0 \mid n[T] \mid T T$	– $T 0 \equiv T$
SEMANTICS	
$T \models 0 \quad \text{iff} \quad T \equiv 0,$	– $T_1 \equiv T_2 \Rightarrow T_2 \equiv T_1$
$T \models n[\varphi] \quad \text{iff} \quad \exists T' \text{ s.t. } T \equiv n[T'] \text{ and } T' \models \varphi,$	– $T_1 \equiv T_2, T_2 \equiv T_3 \Rightarrow T_1 \equiv T_3$
$T \models \varphi \psi \quad \text{iff} \quad T \equiv T_1 T_2, T_1 \models \varphi \text{ and } T_2 \models \psi$ for some information trees T_1 and T_2 .	– $T_1 \equiv T_2 \Rightarrow T_1 T \equiv T_2 T$ – $T_1 \equiv T_2 \Rightarrow n[T_1] \equiv n[T_2]$

Figure 8.5: Interpretation and semantics of $\text{SAL}(\|)$.

$$\begin{array}{lll}
 \pi := \top & \quad (\text{true}) & \varphi := \pi & \quad (\text{atomic formulae}) \\
 | & 0 & | & \varphi \wedge \varphi \mid \neg \varphi & \quad (\text{Boolean connectives}) \\
 & & | & n[\varphi] & \quad (\text{location modality}) \\
 & & | & \varphi \| \varphi & \quad (\text{composition operator}) \\
 & & | & \varphi \triangleright \varphi & \quad (\text{guarantee operator})
 \end{array}$$

As we will see in a moment, the semantics of 0 is essentially the one of $\square \perp$ in $\text{ML}(\|)$, whereas $n[\varphi]$ corresponds to the formula $\Diamond_{=1}\top \wedge \Diamond(n \wedge \varphi)$ stating that the current world w of a pointed forest (\mathcal{K}, w) has exactly one child w' , whose ambient name is n , and $(\mathcal{K}, w') \models \varphi$. The guarantee operator \triangleright is the right-adjoint of the composition operator $\|$, and thus satisfies the two following rules from BBI:

$$\begin{array}{c}
 \triangleright_1 \frac{\varphi \Rightarrow (\psi \triangleright \chi)}{\varphi \| \psi \Rightarrow \chi} \qquad \qquad \qquad \triangleright_2 \frac{\varphi \| \psi \Rightarrow \chi}{\varphi \Rightarrow (\psi \triangleright \chi)}
 \end{array}$$

In order to connect SAL with $\text{ML}(\|)$, we restrict ourselves to the fragment of SAL obtained by removing the guarantee operator. We write $\text{SAL}(\|)$ to denote this fragment. Historically, the semantics of SAL is interpreted on information trees: syntactical objects equipped with a structural equivalence relation \equiv , which forms a congruence with respect to $\|$. The grammar used to construct these structures, their structural equivalence as well as the satisfaction predicate \models for $\text{SAL}(\|)$ are provided in Figure 8.5 (the standard cases for \top and Boolean connectives are omitted). Notice how the semantics of the formulae 0 and $\varphi|\psi$ simply follows the constructors of the grammar for the information trees. The formula 0 asks the tree T to be congruent (w.r.t. the relation \equiv) to 0, whereas $\varphi|\psi$ asks the tree T to be congruent to $T_1|T_2$, for some T_1 and T_2 satisfying φ and ψ , respectively. Lastly, the formula $n[\varphi]$ asks the tree T to be congruent to $n[T']$, for some information tree T' satisfying φ . Here, notice that $n[\varphi]$ requires the root of the tree T to have exactly one child named n . As an example, the information tree $n[0]\|m[n[0]]$ is congruent to $0\|(n[0]\|n[m[0]])$, and satisfies both the formulae $0\|\neg 0$ and $n[\top]\|\neg 0$, but does not satisfy the formula $n[\top]$.

Clearly, the logics $\text{SAL}(\|)$ and $\text{ML}(\|)$ are strongly related, and in what follows we show that the satisfiability problem of these two logics are interreducible. Since both reductions are simple, we leave their formal proofs in Appendix F.

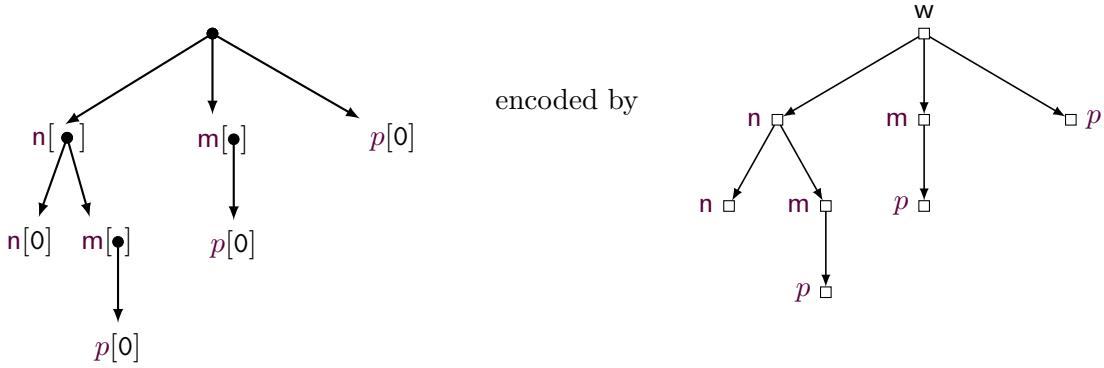


Figure 8.6: An information tree (on the left), and one of its possible encodings as a pointed forest (\mathcal{K}, w) (on the right), with $n \geq 2$.

8.4.1 From $\text{SAL}(\mathbf{|})$ to $\text{ML}(\mathbf{|})$.

The reduction from the satisfiability problem of $\text{SAL}(\mathbf{|})$ to the satisfiability problem of $\text{ML}(\mathbf{|})$ is straightforward, as $\text{SAL}(\mathbf{|})$ is essentially interpreted on pointed forests where each world satisfies a single propositional variable (its ambient name). Suppose we want to translate a formula φ in $\text{SAL}(\mathbf{|})$ into an equisatisfiable formula in $\text{ML}(\mathbf{|})$. We can encode an information tree T as a pointed forest (\mathcal{K}, w) by requiring the set of worlds reachable from w in at most $|\varphi|$ steps to form a tree that is isomorphic to T and preserves the ambient names. Formally, this encoding is defined as follows.

Definition 8.18 (Information Trees encoding). Let $n \in \mathbb{N}$ and let $P \subseteq_{\text{fin}} \text{AP}$. Let T be an information tree having ambient names from P , and let (\mathcal{K}, w) be a pointed forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. We say that (\mathcal{K}, w) is a (n, P) -encoding of T whenever

1. if $n = 0$, then $T \equiv 0$ if and only if $R(w) = \emptyset$,
2. if $n \geq 1$ and $T \equiv n_1[T_1] \mid \dots \mid n_k[T_k]$, then there are w_1, \dots, w_k such that $\{w_1, \dots, w_k\} = R(w)$ and for every $i \in [1, k]$, (\mathcal{K}, w_i) is a $(n-1, P)$ -encoding of T_i and, among the atomic propositions in P , w_i only satisfies n_i .

It is easy to verify that every information tree written with ambient names from P admits a (n, P) -encoding, for every $n \in \mathbb{N}$. Figure 8.6 depicts an information tree and one of its possible encodings as a pointed forest. Given a formula φ of $\text{SAL}(\mathbf{|})$, we introduce its translation $\tau(\varphi)$ in $\text{ML}(\mathbf{|})$. τ is inductively defined as follows:

$$\begin{aligned}\tau(\top) &\stackrel{\text{def}}{=} \top, & \tau(\varphi \wedge \psi) &\stackrel{\text{def}}{=} \tau(\varphi) \wedge \tau(\psi), & \tau(\neg\varphi) &\stackrel{\text{def}}{=} \neg\tau(\varphi), \\ \tau(0) &\stackrel{\text{def}}{=} \square \perp, & \tau(\varphi \mid \psi) &\stackrel{\text{def}}{=} \tau(\varphi) \mid \tau(\psi), & \tau(n[\varphi]) &\stackrel{\text{def}}{=} \Diamond_{=1} \top \wedge \Diamond(n \wedge \varphi).\end{aligned}$$

The following lemma, whose proof follows via an easy structural induction on φ (see Appendix F), shows the correctness of this translation.

Lemma 8.19. Let $n \in \mathbb{N}$, $P \subseteq_{\text{fin}} \text{AP}$. Let T be an information tree having ambient names from P and let (\mathcal{K}, w) be a pointed forest such that (\mathcal{K}, w) (n, P) -encodes T . For every formula φ in $\text{SAL}(\mathbf{|})$ such that $|\varphi| \leq n$ we have, $T \models \varphi$ if and only if $(\mathcal{K}, w) \models \tau(\varphi)$.

Thanks to Lemma 8.19, we are able to complete the reduction.

Lemma 8.20. Let φ be in $\text{SAL}(\mathbb{I})$ built over $P \subseteq_{\text{fin}} \text{AP}$ and $p \notin P$. φ is satisfiable if and only if $\tau(\varphi) \wedge \bigwedge_{i \in [1, |\varphi|]} \square^i \bigvee_{n \in P \cup \{p\}} (n \wedge \bigwedge_{m \in (P \cup \{p\}) \setminus \{n\}} \neg m)$ is satisfiable.

Notice that, in Lemma 8.20, we make use of an auxiliary ambient name p that does not appear in the formula φ . This is needed as there are formulae of $\text{SAL}(\mathbb{I})$ that are satisfied only by models having names not appearing in them, as for instance the formula $\neg 0$. However from [34, Lemma 8] we know that considering a single fresh name suffices. The subformula

$$\bigwedge_{i \in [1, |\varphi|]} \square^i \bigvee_{n \in P \cup \{p\}} (n \wedge \bigwedge_{m \in (P \cup \{p\}) \setminus \{n\}} \neg m)$$

characterises the classes of pointed forests (\mathcal{K}, w) that are $(|\varphi|, P \cup \{p\})$ -encodings of some information tree, by stating that all worlds reachable from w in at most $|\varphi|$ steps (w excluded) satisfy exactly one atomic proposition from $P \cup \{p\}$, as required by the property (2) of Definition 8.18.

From the AEXP_{POL} -completeness of the satisfiability problem of $\text{ML}(\mathbb{I})$, Lemma 8.20 allows us to conclude that the satisfiability problem of $\text{SAL}(\mathbb{I})$ is decidable in AEXP_{POL} .

8.4.2 From $\text{ML}(\mathbb{I})$ to $\text{SAL}(\mathbb{I})$.

To perform the reduction from the satisfiability problem of $\text{ML}(\mathbb{I})$ to the satisfiability problem of $\text{SAL}(\mathbb{I})$, the main difficulty is dealing with the fact that worlds in a Kripke-style finite forest can satisfy multiple atomic propositions, whereas each ambient of an information tree only satisfies exactly one atomic proposition: its ambient name. A simple yet effective solution to this problem relies on representing the atomic propositions satisfied by a world as ambients. For instance, a world w with no children and satisfying the atomic propositions p and q could be simply represented with the information tree $p[0] \parallel q[0]$. In order to represent the accessibility relation of the Kripke-style finite forest, we can reserve an ambient name rel , so that ambient sharing this name can be distinguished from the ones encoding atomic propositions. For instance, in this encoding, we can represent the fact that w is a child of a world satisfying p by encapsulating the information tree $p[0] \parallel q[0]$ of w under the context $\text{rel}[_] \parallel p[0]$, obtaining the information tree $\text{rel}[p[0] \parallel q[0]] \parallel p[0]$.

One last obstacle for the reduction is how to deal with the composition operator of $\text{ML}(\mathbb{I})$. Indeed, we would like to simply rely on the composition operator of $\text{SAL}(\mathbb{I})$, but this could a priori separate worlds encoding atomic propositions. For instance, again considering the world w satisfying p and q , $(\mathcal{K}, w) \models p \parallel p$ holds, whereas $p[0] \parallel q[0] \models p[0] \parallel p[0]$ does not. The solution is very simple: exactly as done when reducing the satisfiability problem of the propositional logic in team semantics $\text{PL}(\sim)$ to the satisfiability problem of $\text{ML}(\mathbb{I})$, we rely on enough copies of the worlds encoding a propositional symbol, so that we can correctly encode their valuation even after using the composition operator.

Let us start formalising the encoding described above. Let $P \subseteq_{\text{fin}} \text{AP}$ be a finite set of atomic propositions, and let rel and ap be two ambient names not in P . The ambient name rel encodes the accessibility relation of a Kripke-style finite forest, whereas ambients with name ap can be seen as containers for atomic propositions holding in a world. Given a subset $Q = \{p_1, \dots, p_k\}$ of P , we write $\text{ap}[Q]$ for the information tree $\text{ap}[p_1[0] \parallel \dots \parallel p_k[0]]$. Given an information tree T and $m \in \mathbb{N}$, we write $[T]^m$ for the information tree obtained by composing (in parallel) m copies of T , i.e. $[T]^0 \stackrel{\text{def}}{=} 0$ and $[T]^{m+1} \stackrel{\text{def}}{=} T \parallel [T]^m$. We define the encoding of a pointed forest as follows.

Definition 8.21 (Pointed forests as information trees). Let (\mathcal{K}, w) be a pointed forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, and let Q be the set of atomic propositions in P that are satisfied by w , and let

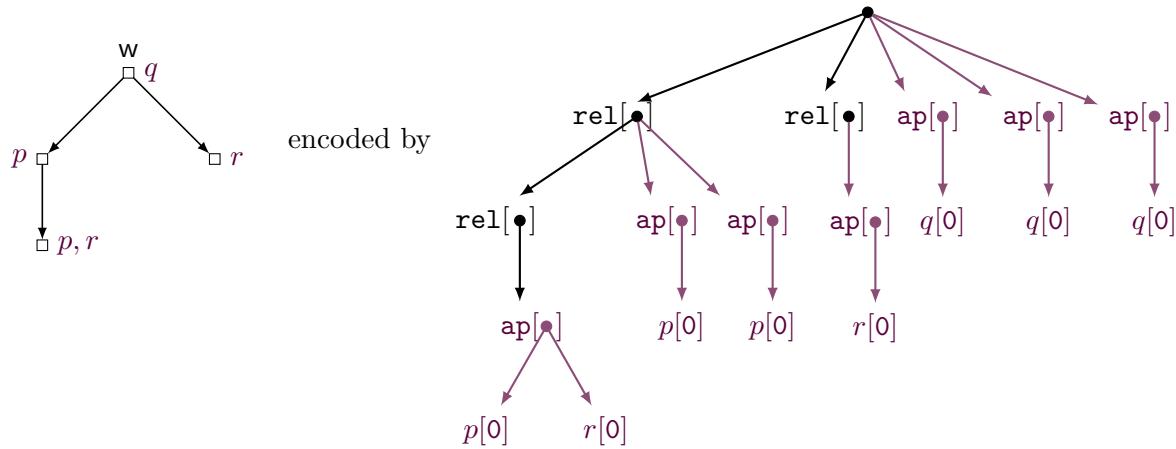


Figure 8.7: A pointed forest (\mathcal{K}, w) (on the left), and one of its possible encodings as an information tree (on the right), for $n \leq 3$. The portion of the information tree encoding atomic propositions is highlighted.

$R(w) = \{w_1, \dots, w_k\}$. Given a natural number $n \geq 1$, we say that an information tree T is a (n, P) -encoding of (\mathcal{K}, w) whenever there are $m \geq n$ and information trees T_1, \dots, T_k such that

1. $T \equiv [\text{ap}[Q]]^m \mid \text{rel}[T_1] \mid \dots \mid \text{rel}[T_k]$,
2. if $n > 1$, then for every $i \in [1, k]$, T_i is a $(n - 1, P)$ -encoding of (\mathcal{K}, w_i) .

Figure 8.7 shows a pointed forest and one of its possible encodings as an information tree.

Below, given an ambient name n , we write $\langle n \rangle \varphi$ for the $\text{SAL}(\|)$ formula $n[\varphi] \mid \top$. Notice that the formula $\langle n \rangle \varphi$, whose syntax is borrowed from Hennessy-Milner logic [86], is essentially a relativised version of the modality of possibility $\Diamond \varphi$, and states that the root of an information tree has a child named n satisfying φ . The dual $[n]\varphi$ of $\langle n \rangle \varphi$ is defined as expected: $[n]\varphi \stackrel{\text{def}}{=} \neg \langle n \rangle \neg \varphi$. Moreover, we define the graded extensions of these modalities, and write $\langle n \rangle_{\geq k} \varphi$ (where $k \in \mathbb{N}$) for the formula inductively defined as

$$\langle n \rangle_{\geq 0} \stackrel{\text{def}}{=} \top, \quad \langle n \rangle_{\geq k+1} \stackrel{\text{def}}{=} (\langle n \rangle \varphi) \mid \langle n \rangle_{\geq k} \varphi.$$

Given $i \in \mathbb{N}$, we write $\langle n \rangle^i$ for i imbrications of $\langle n \rangle$, that is:

$$\langle n \rangle^0 \varphi \stackrel{\text{def}}{=} \varphi, \quad \langle n \rangle^{i+1} \varphi \stackrel{\text{def}}{=} \langle n \rangle^i \langle n \rangle \varphi.$$

Lastly, $[n]^i \varphi$ stands for $\neg \langle n \rangle^i \neg \varphi$.

We introduce the translation $\tau(\varphi)$ in $\text{SAL}(\|)$ of a formula φ in $\text{ML}(\|)$. Exactly as in the translation from $\text{SAL}(\|)$ to $\text{ML}(\|)$, the translation τ is homomorphic for Boolean connectives and \top . For the atomic proposition, the modality \Diamond and the operator \mid it is instead defined as:

$$\tau(p) \stackrel{\text{def}}{=} \langle \text{ap} \rangle \langle p \rangle \top, \quad \tau(\Diamond \varphi) \stackrel{\text{def}}{=} \langle \text{rel} \rangle \tau(\varphi), \quad \tau(\varphi \mid \psi) \stackrel{\text{def}}{=} (\tau(\varphi) \wedge \langle \text{ap} \rangle_{\geq |\varphi|} \top) \mid (\tau(\psi) \wedge \langle \text{ap} \rangle_{\geq |\psi|} \top).$$

Notice that, in the translation of \mid , the model of $\text{SAL}(\|)$ has to be split in such a way that both subtrees contain enough ap ambients to correctly answer to the formula $\langle \text{ap} \rangle \langle p \rangle \top$. Because of this, one can easily show that the size of $\tau(\varphi)$ is quadratic in $|\varphi|$. The translation is shown correct in the lemma below, whose proof is given in Appendix F.

Lemma 8.22. Let $P \subseteq_{\text{fin}} \text{AP}$ and $n \geq 1$. Let T be a (n, P) -encoding of a pointed forest $(\mathcal{K}, \mathbf{w})$. For every formula φ in $\text{ML}(\mathbf{I})$, built over P and with $|\varphi| \leq n$, we have $(\mathcal{K}, \mathbf{w}) \models \varphi$ iff $T \models \tau(\varphi)$.

Thanks to Lemma 8.22, we establish the AEXP_{POL} -hardness of $\text{SAL}(\mathbf{I})$. To do so, one last step is to characterise the class of information trees that are encoding of one or more pointed forests. Below, given an ambient name \mathbf{n} and an information tree T congruent to $\mathbf{n}[T'] \parallel T''$, we say that T' is a \mathbf{n} -child of T . Given $i \in \mathbb{N}$, we say that T' is a (i, \mathbf{n}) -descendant of T whether there is a sequence of information trees T_0, \dots, T_i such that $T_0 \equiv T$, $T_i \equiv T'$, and for every $j \in [0, i-1]$, T_{j+1} is a \mathbf{n} -child. Let T be an information tree, $n \geq 1$ and $P = \{p_1, \dots, p_k\} \subseteq_{\text{fin}} \text{AP}$. Let T' be a (i, rel) -descendant of T , where $i \in [0, n-1]$. Following Definition 8.21, in order to characterise the fact that T is a (n, P) -encoding of a pointed forest, we require T' to satisfy the formulae:

Formula	Informal explanation
$\alpha_{i,n} \stackrel{\text{def}}{=} \langle \text{ap} \rangle_{\geq(n-i)}$	T' is congruent to $\text{ap}[T_1] \parallel \dots \parallel \text{ap}[T_m] \parallel T_{m+1}$, for some $m \geq n-i$ and trees T_1, \dots, T_{m+1} ,
$\beta \stackrel{\text{def}}{=} \neg(\top \parallel (\neg 0 \wedge \neg \langle \text{rel} \rangle \top \wedge \neg \langle \text{ap} \rangle \top))$	every child of T' is either a rel -child or a ap -child,
$\gamma_P \stackrel{\text{def}}{=} \bigwedge_{p \in P} (\langle \text{ap} \rangle \langle p \rangle \top \Rightarrow [\text{ap}] \langle p \rangle \top)$	rel -children of T' agree on the types of p -children they have, where $p \in P$,
$\delta_P \stackrel{\text{def}}{=} [\text{ap}]((p_1[0] \vee 0) \parallel \dots \parallel (p_k[0] \vee 0))$	there is $\{p_{i_1}, \dots, p_{i_l}\} \subseteq P$ such that every rel -child of T' is congruent to $p_{i_1}[0] \parallel \dots \parallel p_{i_l}[0]$.

We leave to the reader to check that T satisfies $\bigwedge_{i \in [0, n-1]} [\text{rel}]^i (\alpha_{i,n} \wedge \beta \wedge \gamma_P \wedge \delta_P)$ if and only if it is a (n, P) -encoding of some pointed forests (alternatively, see the proof of Lemma 8.23, which is given in Appendix F).

Lemma 8.23. Let φ be a formula in $\text{ML}(\mathbf{I})$ built over $P = \{p_1, \dots, p_k\} \subseteq_{\text{fin}} \text{AP}$. φ is satisfiable if and only if the formula $\tau(\varphi) \wedge \bigwedge_{i \in [0, |\varphi|-1]} [\text{rel}]^i (\alpha_{i,|\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P)$ is satisfiable in $\text{SAL}(\mathbf{I})$.

From Theorem 8.17, Lemmata 8.20 and 8.23 show that the satisfiability problem of $\text{SAL}(\mathbf{I})$ is complete for AEXP_{POL} . This implies that the static ambient logic SAL (i.e. the extension of $\text{SAL}(\mathbf{I})$ with the guarantee operator \triangleright) is AEXP_{POL} -hard, refuting hints from [34, Section 6].

Theorem 8.24. The satisfiability problem for $\text{SAL}(\mathbf{I})$ is AEXP_{POL} -complete. The satisfiability problem for SAL is AEXP_{POL} -hard.

The Complexity and Expressive Power of the Modal Logic $\text{ML}(*)$

Contents

9.1	$\text{ML}(*)$: when * replaces 	407
9.1.1	$\text{ML}(*)$: Syntax and Semantics.	407
9.2	$\text{ML}(*)$ is Strictly Less Expressive than $\text{ML}()$	411
9.2.1	$\text{ML}(*)$ is not more expressive than GML	411
9.2.2	Showing $\text{ML}(*) \prec \text{GML}$ via EF games for $\text{ML}(*)$	421
9.3	The complexity of $\text{ML}(*)$	432
9.3.1	The problem of tiling a (huge) grid.	432
9.3.2	Enforcing $t(j, n)$ children.	433
9.3.3	Nominals, forks and number comparisons.	437
9.3.4	Construction of $\text{type}(j)$: formulae for the base cases $i = j$ or $j = 1$	439
9.3.5	Construction of $\text{type}(j)$: formulae for the inductive step $1 \leq i < j$	443
9.3.6	Tiling a grid $[0, t(k, n) - 1] \times [0, t(k, n) - 1]$	452
9.4	Revisiting TOWER-hard Logics with $\text{ML}(*)$	459
9.4.1	From $\text{ML}(*)$ to the second-order modal logic QK^t	459
9.4.2	From $\text{ML}(*)$ to modal separation logic with converse.	461

In this chapter

We are interested in the following question:

How do the operators $|$ (from ambient logic) and $$ (from separation logic) compare in terms of expressivity and complexity?*

To answer this question, we propose to look at $\text{ML}(*)$: the logic obtained from $\text{ML}(|)$ by replacing the composition operator $|$ with the separating conjunction $*$.

Very surprisingly, whereas we find $\text{ML}(*)$ to be strictly less expressive than $\text{ML}(|)$, we show that replacing $|$ with $*$ makes the satisfiability problem jump from AEXP_{POL} to TOWER . From the TOWER -completeness of $\text{ML}(*)$ we are able to solve open problems on modal separation logics, as well as reproving the TOWER -hardness of the second-order modal logic QK on trees.

Here is a roadmap of the chapter.

Section 9.1. We introduce the logic $\text{ML}(*)$. Differently from the operator $|$, the separating conjunction allows to arbitrarily split the accessibility relation of a Kripke-style finite forest. We introduce the abbreviation $\blacklozenge \varphi \stackrel{\text{def}}{=} \square \perp * \varphi$, and show that the formula $\varphi * \psi$ is equivalent to $\blacklozenge(\varphi | \psi)$. This equivalence plays a fundamental role in bounding the expressiveness of $\text{ML}(*)$.

Section 9.2. We show that $\text{ML}(*)$ is strictly less expressive than the graded modal logic GML . First, we prove that $\text{ML}(*)$ is at most as expressive as GML . From Chapter 7, we know that GML and $\text{ML}(|)$ have the same expressive power. In virtue of the equivalence $\varphi * \psi \equiv \blacklozenge(\varphi | \psi)$, this implies that it is sufficient to show that, for every formula χ in GML , the formula $\blacklozenge \chi$ can be expressed in GML . We prove this result with a model theoretical argument that relies on the notion of g-bisimulation [51], i.e. a refinement of the classical back-and-forth conditions of a bisimulation (see e.g. [15]), tailored towards capturing graded modalities.

Afterwards, we show that the simple GML formula $\varphi = \Diamond_{=2} \Diamond_{=1} \top$, stating that the current world has exactly two children with exactly one child, cannot be expressed in $\text{ML}(*)$. As already done in previous chapters of the thesis, this is proven by introducing a notion of Ehrenfeucht-Fraïssé games for $\text{ML}(*)$. After showing that our games are sound and complete for $\text{ML}(*)$, we show the inexpressibility of φ by defining two pointed forests, one that satisfies φ and one that does not satisfy it, for which the duplicator has a winning strategy.

The following theorem summarises the expressivity results on $\text{ML}(*)$, $\text{ML}(|)$, GML and the standard modal logic ML . $\mathcal{L}_1 \prec \mathcal{L}_2$ means that the logic \mathcal{L}_1 is less expressive than the logic \mathcal{L}_2 , and $\mathcal{L}_1 \approx \mathcal{L}_2$ means that the two logics have the same expressive power,

Theorem 9.13. $\text{ML} \prec \text{ML}(*) \prec \text{GML} \approx \text{ML}(|)$.

Section 9.3. We move to the complexity of $\text{ML}(*)$ and show that its satisfiability problem is TOWER -hard. This result is shown with a uniform reduction, for all $k > 1$, from the k - NEXPTIME -complete problem of tiling the grid $[0, t(k, n) - 1] \times [0, t(k, n) - 1]$, where t is the (non-elementary) tetration function defined as $t(0, n) = n$ and $t(k + 1, n) = 2^{t(k, n)}$. In order to achieve the reduction, the main technical development consists in defining a formula, of size exponential in $k > 1$ and polynomial in $n \geq 1$, that is satisfied only by pointed forests (\mathcal{K}, w) where the current world w has exactly $t(k, n)$ children. This formula, denoted by $\text{type}(k)$, is defined inductively on k . Roughly speaking, we assume each child w' of w to satisfy $\text{type}(k - 1)$,

and use the $t(k - 1, n)$ children of w' to represent, in binary, a number $n(w') \in [0, t(k, n) - 1]$. Then, $\text{type}(k)$ requires w to be such that $\{n(w') \mid w' \text{ is a child of } w\} = [0, t(k, n) - 1]$.

Section 9.4. We rely on the TOWER-hardness of the satisfiability problem for $\text{ML}(*)$ to (re)prove the TOWER-hardness of two logics interpreted on tree-like structures.

First, we translate $\text{ML}(*)$ into the second-order modal logic QK . When interpreted on trees (QK^t), this logic is known to be TOWER-complete [8], which allows us to derive an upper bound on the satisfiability problem for $\text{ML}(*)$.

Theorem 9.43. The satisfiability problem of QK^t and $\text{ML}(*)$ are TOWER-complete.

Lastly, we consider the logic $\text{MSL}(*, \diamond^{-1})$, that is the fragment of modal separation logic (Section 2.3.2) featuring Boolean connectives, atomic propositions, the separating conjunction $*$ and the *converse modality of possibility* \diamond^{-1} . From [54], the satisfiability problem of this logic is known to be PSPACE-hard and decidable in TOWER. Thanks to $\text{ML}(*)$, we close this gap.

Theorem 9.46. The satisfiability problem of $\text{MSL}(*, \diamond^{-1})$ is TOWER-complete.

9.1 $\text{ML}(\ast)$: WHEN \ast REPLACES $|$

The main focus of Chapters 7 and 8 was the study of the modal logic $\text{ML}(|)$, which we found to be as expressive as the graded modal logic GML , and admitting a satisfiability problem that is elementary decidable, more precisely AEXP_{POL} -complete. The introduction of $\text{ML}(|)$ was motivated by similarities between the composition operator $|$ of ambient logic and the separating conjunction \ast of separation logic, which emerge by looking at the two proof systems for $\text{ML}(|)$ and $\text{SL}(\ast, \rightarrow)$ that have been designed in Chapter 7 and Chapter 6.

In this chapter, we undertake a thorough analysis of the differences between the separating conjunction and the composition operator in the framework of modal logic. We do so by replacing the composition operator of $\text{ML}(|)$ by the separating conjunction of separation logic, and studying the expressive power and complexity of the resulting logic, denoted by $\text{ML}(\ast)$. Despite the two logics being quite close, to the point that we are able to use results from $\text{ML}(|)$ in order to pursue the analysis of $\text{ML}(\ast)$, we show that $\text{ML}(\ast)$ is profoundly different from $\text{ML}(|)$. First of, while every formula of $\text{ML}(\ast)$ is expressible in $\text{ML}(|)$, we show that the converse does not hold: $\text{ML}(\ast)$ is strictly less expressive than $\text{ML}(|)$ (Section 9.2). However, in terms of computational complexity, the satisfiability problem for $\text{ML}(\ast)$ is by far harder than the one for $\text{ML}(|)$: instead of being AEXP_{POL} -complete (thus, solvable in EXPSPACE), the satisfiability problem of $\text{ML}(\ast)$ is TOWER -complete (Section 9.3). Our choice of framing the separating conjunction in the context of modal logic has further benefits, as we are able to close open problems in the realm of modal separation logics (Section 2.3.2) and reprove the TOWER -completeness of second-order modal logic (QK , see Section 8.1.1) interpreted on tree-like structures (Section 9.4).

9.1.1 $\text{ML}(\ast)$: Syntax and Semantics.

Below, we define the modal logic $\text{ML}(\ast)$, while we refer the reader to Chapter 7 for the definition of the logics $\text{ML}(|)$ and GML , which are both fundamental in the context of this chapter. As usual, we write AP for a countably infinite set of atomic propositions. The logic $\text{ML}(\ast)$ enriches modal logic ML with the separating conjunction of \ast . Its formulae are built from the grammar below (where $p \in \text{AP}$):

$$\begin{array}{lll} \pi := & \top & (\text{true}) \\ & | & p & (\text{propositional symbol}) \\ \varphi := & \pi & (\text{atomic formulae}) \\ & | & \varphi \Rightarrow \varphi & (\text{Boolean connectives}) \\ & | & \Diamond \varphi & (\text{modality of possibility}) \\ & | & \varphi \ast \varphi & (\text{separating conjunction}) \end{array}$$

Syntax-wise, $\text{ML}(\ast)$ is a fragment of MSL (see Section 2.3.2). However, we should here avoid to confuse the two logics, as the standard semantics of MSL is given in terms of Kripke-style finite functions (Definition 2.17), whereas in this chapter we interpret $\text{ML}(\ast)$ on Kripke-style finite forests, exactly as $\text{ML}(|)$. The fundamental difference between the two classes of structures is that the accessibility relation of a Kripke-style finite function is, as the word suggest, functional, leading to the validity of the formula $\Diamond \varphi \Rightarrow \Box \varphi$. This is clearly not true for $\text{ML}(\ast)$. This difference has staggering effects in terms of computational complexity: the satisfiability problem $\text{ML}(\ast)$ interpreted on Kripke-style finite function is known to be decidable in NP [54] (see also Figure 2.12) whereas we show that it is non-elementary on finite forests. Let us recall the definition of Kripke-style finite forest.

-
- $(\mathcal{K}, w) \models p$ iff $w \in \mathcal{V}(p)$,
 $(\mathcal{K}, w) \models \Diamond\varphi$ iff there is $w' \in \mathcal{W}$ such that $w' \in R(w)$ and $(\mathcal{K}, w') \models \varphi$,
 $(\mathcal{K}, w) \models \varphi * \psi$ iff there are \mathcal{K}_1 and \mathcal{K}_2 s.t. $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$, $(\mathcal{K}_1, w) \models \varphi$ and $(\mathcal{K}_2, w) \models \psi$.

Figure 9.1: Satisfaction relation for $\text{ML}(*)$, with respect to (\mathcal{K}, w) where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

Definition 7.1 (Kripke-style finite forest). A *(Kripke-style) finite forest* $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a triple where \mathcal{W} is a non-empty finite set of *worlds* (i.e. a *universe*), $\mathcal{V} : \text{AP} \rightarrow 2^{\mathcal{W}}$ is a *valuation* and $R \subseteq \mathcal{W} \times \mathcal{W}$ is a finite binary relation whose inverse R^{-1} is functional and acyclic.

As usual, we write $R(w) = \{w' \in \mathcal{W} \mid (w, w') \in R\}$ to denote the set of children of w . We introduce the notation $R|_w^{\leq n}$ which denotes the minimal subset of R encoding exactly the subtree rooted at w of height at most n . Formally,

$$R|_w^{\leq n} \stackrel{\text{def}}{=} \{(w', w'') \in R \mid w' \in R^i(w) \text{ for some } i \in [0, n - 1]\}.$$

Similarly, $R|_w$ stands for the set $\{(w', w'') \in R \mid w' \subseteq R^*(w)\}$, i.e. the maximal subset of R encoding exactly the subtree rooted at w . Alternatively, $R|_w = \bigcup_{n \in \mathbb{N}} R|_w^{\leq n}$.

The only difference between $\text{ML}(\mathbf{I})$ and $\text{ML}(*)$ rests on the semantics of their spatial connectives. Instead of relying on the composition $+_w$, $\text{ML}(*)$ uses the following notion of union.

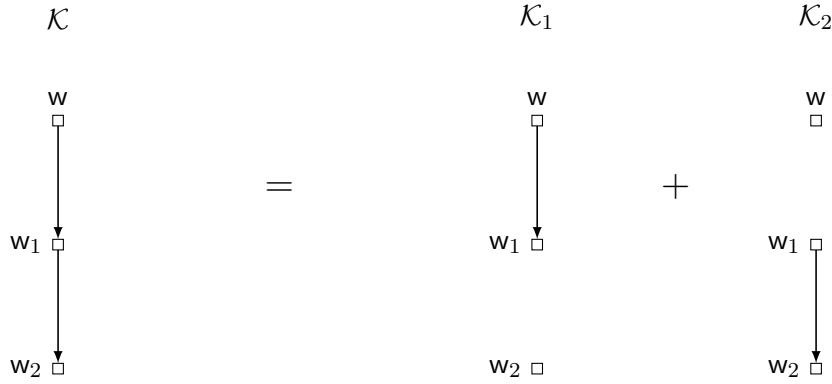
Definition 9.1 (+ : the Separation-like union). Two Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ are *disjoint* if $R_1 \cap R_2 = \emptyset$. When this holds, their *union* $\mathcal{K}_1 + \mathcal{K}_2$ is the Kripke-style finite function $(\mathcal{W}, R_1 \cup R_2, \mathcal{V})$. We write $\mathcal{K}_1 \subseteq \mathcal{K}_2$ if $R_1 \subseteq R_2$.

The notions of disjointness, union and subforest \subseteq are the ones of **MSL** (Definition 2.18), the only difference being the class of Kripke structures considered. The union $+$ is coarser than the union $+_w$, as shown by the equality below (where $\mathcal{K}_i = (\mathcal{W}, R_i, \mathcal{V})$, for $i \in [1, 3]$):

$$\mathcal{K}_1 +_w \mathcal{K}_2 = \mathcal{K}_3 \quad \text{iff} \quad \mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}_3 \text{ and for all } i \in \{1, 2\} \text{ and } w' \in R_i(w), R_i^+(w') = R_3^+(w').$$

In fact, whereas the union $+_w$ always preserves the trees rooted in children of w , like the composition from ambient logic, the union $+$ arbitrarily splits R , exactly as the union of heaps in separation logic.

Given a pointed forest (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a Kripke-style finite forest and $w \in \mathcal{W}$ the satisfaction relation \models for formulae in $\text{ML}(*)$ is defined in Figure 9.1, omitting the standard clauses for \top and the Boolean connectives \Rightarrow and \neg . As we can see, the semantics for the separating conjunction $*$ is the usual one of BBI, instantiated with respect to the monoidal operation $+$. This means that, in a lingua featuring both \mathbf{I} and $*$, the relationship between $+_w$ and $+$ provided above entails the validity of the formula $\varphi \mathbf{I} \psi \Rightarrow \varphi * \psi$. The converse is not true. For instance, the formula $\square \square \perp \mathbf{I} \square \square \perp$ is not equivalent to $\square \square \perp * \square \square \perp$. To see why, let us consider the pointed forest (\mathcal{K}, w) depicted in Figure 9.2. (\mathcal{K}, w) clearly satisfies $\square \square \perp * \square \square \perp$: it is sufficient to split \mathcal{K} into two finite forests \mathcal{K}_1 and \mathcal{K}_2 , the first containing the arrow (w, w_1) and the other containing the arrow (w_1, w_2) , as showed on the right side of Figure 9.2. In both finite forests, no world is accessible from w in two steps, leading to $(\mathcal{K}_1, w) \models \square \square \perp$ and $(\mathcal{K}_2, w) \models \square \square \perp$. Since $\mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2$, we conclude that $(\mathcal{K}, w) \models \square \square \perp * \square \square \perp$. However, (\mathcal{K}, w) does not satisfy $\square \square \perp \mathbf{I} \square \square \perp$. Indeed, in any split of \mathcal{K} carried out with the operator $+_w$, both

Figure 9.2: A pointed forest (\mathcal{K}, w) (on the left), and a possible decomposition via the union $+$.

the arrows (w, w_1) and (w_1, w_2) must belong to one of the two subforests, which cannot therefore satisfy the formula $\Box\Box\perp$.

The above example shows that, in general, $\varphi * \psi \not\models \varphi | \psi$. However, interesting equivalences between formulae of $\text{ML}(\ast)$ and $\text{ML}(|)$ can be easily established. Given formulae φ and ψ , let us write $\varphi \equiv \psi$ to denote that φ and ψ are *logically equivalent*, i.e. for every pointed forest (\mathcal{K}, w) , $(\mathcal{K}, w) \models \varphi$ if and only if $(\mathcal{K}, w) \models \psi$. For instance, we just showed that $(\Box\Box\perp | \Box\Box\perp) \not\equiv (\Box\Box\perp * \Box\Box\perp)$, and one can show that $\Diamond p | \Diamond q \equiv \Diamond p * \Diamond q$. These two (non)equivalences already shed some light on $\text{ML}(|)$ and $\text{ML}(\ast)$: the two logics are similar when it comes to their formulae of modal degree one. As usual, the *modal degree* of φ , denoted by $\text{md}(\varphi)$, is the maximal number of nested modalities \Diamond (or $\Diamond_{\geq k}$, in the case of GML) in φ .

Lemma 9.2. Let φ be a formula in $\text{ML}(|)$ with $\text{md}(\varphi) \leq 1$. Then, $\varphi \equiv \varphi[| \leftarrow \ast]$ where $\varphi[| \leftarrow \ast]$ is the formula in $\text{ML}(\ast)$ obtained from φ by replacing every occurrence of $|$ by \ast .

Notice that this lemma already implies that the satisfiability problem for $\text{ML}(\ast)$ is AEXP_{Pol}-hard, directly from Theorem 8.17. To facilitate the proof of Lemma 9.2, as well as subsequent lemmata, we rely on the following well-known result from ML , which can be easily reproved for $\text{ML}(\ast)$ and $\text{ML}(|)$ (by structural induction on φ).

Proposition 9.3. Let φ in ML (alternatively, GML, $\text{ML}(|)$ or $\text{ML}(\ast)$) and $n \geq \text{md}(\varphi)$. Let (\mathcal{K}, w) be a finite forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. $(\mathcal{K}, w) \models \varphi$ iff $((\mathcal{W}, R|_w^{\leq n}, \mathcal{V}), w) \models \varphi$.

Essentially, Proposition 9.3 states that restricting the accessibility relation R of pointed forest to those worlds that are reachable from w in at most $n \geq \text{md}(\varphi)$ steps do not change the satisfaction of φ . Then, in the proof of Proposition 9.3 we essentially restrict the accessibility relation to $R|_w^{\leq 1}$ and notice that, on this particular case, the definition of $+$ and $+_w$ coincide.

Proof of Lemma 9.2. As $\text{md}(\varphi) \leq 1$, by Proposition 9.3,

$$(\mathcal{K}, w) \models \varphi \text{ iff } ((\mathcal{W}, R|_w^{\leq 1}, \mathcal{V}), w) \models \varphi, \quad (\mathcal{K}, w) \models \varphi[| \leftarrow \ast] \text{ iff } ((\mathcal{W}, R|_w^{\leq 1}, \mathcal{V}), w) \models \varphi[| \leftarrow \ast].$$

Thus, without loss of generality, let us assume that $R = R|_w^{\leq 1}$, and thus that every child of w does not have any children. To show that $(\mathcal{K}, w) \models \varphi$ iff $(\mathcal{K}, w) \models \varphi[| \leftarrow \ast]$ the proof boils down to the proof of the equivalence

for every Kripke-style finite forest \mathcal{K}_1 and \mathcal{K}_2 , $\mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2$ if and only if $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$.

Given $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$, this follows directly from the characterisation of $+_{\mathbf{w}}$ in term of $+$ previously discussed, i.e.

$$\mathcal{K}_1 +_{\mathbf{w}} \mathcal{K}_2 = \mathcal{K} \text{ iff } \mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K} \text{ and for all } i \in \{1, 2\} \text{ and } \mathbf{w}' \in R_i(\mathbf{w}), R_i^+(\mathbf{w}') = R^+(\mathbf{w}').$$

Indeed, as $R = R|_{\mathbf{w}}^{\leq 1}$, for all $i \in \{1, 2\}$ and $\mathbf{w}' \in R_i(\mathbf{w})$, $R_i^+(\mathbf{w}')$ and $R^+(\mathbf{w}')$ are both empty, \square

Although the equivalence of Lemma 9.2 ceases to work for formulae of modal depth more than one, we can still somewhat rephrase the operator \ast in terms of $|$, albeit with some external help. Let us introduce the auxiliary operator \blacklozenge defined as $\blacklozenge \varphi \stackrel{\text{def}}{=} \varphi \ast \square \perp$. Formally,

$$(\mathcal{W}, R, \mathcal{V}), \mathbf{w} \models \blacklozenge \varphi \text{ iff there is } R' \subseteq R \text{ s.t. } R'(\mathbf{w}) = R(\mathbf{w}) \text{ and } (\mathcal{W}, R', \mathcal{V}), \mathbf{w} \models \varphi.$$

Despite being similar, the operator \blacklozenge should not be confused with the sabotage modality [4] we introduced in Chapter 4 in order to define ALT, which is only removing one arrow from the accessibility relation. In this sense, the operator \blacklozenge is closer to the operator \blacklozenge^* of ALT, as it allows us to remove multiple arrows at once. However, whereas \blacklozenge^* removes arrows arbitrarily, \blacklozenge always preserves the children of \mathbf{w} . For instance, with respect to the Kripke-style finite forests in Figure 9.2, \mathcal{K}_1 is a structure that should be considered when evaluating $\blacklozenge \varphi$ on $(\mathcal{K}, \mathbf{w})$, whereas \mathcal{K}_2 is not, as the set of children of \mathbf{w} is different between \mathcal{K}_2 and \mathcal{K} . We show that the operators \blacklozenge and $|$ are sufficient to capture the separating conjunction.

Lemma 9.4. Let φ and ψ be in either $\text{ML}(|)$, GML or $\text{ML}(\ast)$. We have $\varphi \ast \psi \equiv \blacklozenge(\varphi | \psi)$.

A way to understand this equivalence is to notice that, unlike $|$, when \ast splits a finite forest \mathcal{K} into \mathcal{K}_1 and \mathcal{K}_2 , it may disconnect in both submodels worlds that in \mathcal{K} are otherwise reachable from the current world. Applying \blacklozenge before $|$ allows us to imitate this behaviour. Indeed, even though $|$ preserves reachability in either \mathcal{K}_1 or \mathcal{K}_2 , \blacklozenge deletes part of \mathcal{K} , making some world inaccessible. The equivalence $\varphi \ast \psi \equiv \blacklozenge(\varphi | \psi)$, proved below, provides us with a useful insight which (in the next section) helps us analysing the expressive power of $\text{ML}(\ast)$.

Proof. Below, let $(\mathcal{K}, \mathbf{w})$ be a pointed forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

(\Rightarrow): Suppose $(\mathcal{K}, \mathbf{w}) \models \varphi \ast \psi$. There are $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$, $(\mathcal{K}_1, \mathbf{w}) \models \varphi$ and $(\mathcal{K}_2, \mathbf{w}) \models \psi$. By Lemma 9.3 together with the fact that $R|_{\mathbf{w}} = R|_{\mathbf{w}}^{\leq \text{card}(R) + \text{md}(\varphi)}$, we have $((\mathcal{W}, R_1|_{\mathbf{w}}, \mathcal{V}), \mathbf{w}) \models \varphi$ and $((\mathcal{W}, R_2|_{\mathbf{w}}, \mathcal{V}), \mathbf{w}) \models \psi$. Now, let us consider the model $\widehat{\mathcal{K}} = (\mathcal{W}, R_1|_{\mathbf{w}} \cup R_2|_{\mathbf{w}}, \mathcal{V})$. Fundamentally, for both $i \in \{1, 2\}$ and every $(\mathbf{w}_1, \mathbf{w}_2) \in R_i|_{\mathbf{w}}$, we have $\mathbf{w}_1 \in (R_i|_{\mathbf{w}}) \ast (\mathbf{w})$, directly from the definition of $R_i|_{\mathbf{w}}$. Thus, by definition of $+_{\mathbf{w}}$, it is easy to see that $(\mathcal{W}, R_1|_{\mathbf{w}}, \mathcal{V}) +_{\mathbf{w}} (\mathcal{W}, R_2|_{\mathbf{w}}, \mathcal{V}) = \widehat{\mathcal{K}}$. Hence $\widehat{\mathcal{K}}, \mathbf{w} \models \varphi | \psi$. Moreover, $R_1|_{\mathbf{w}} \cup R_2|_{\mathbf{w}} \subseteq R$ and $(R_1|_{\mathbf{w}} \cup R_2|_{\mathbf{w}})(\mathbf{w}) = R(\mathbf{w})$, which allows us to conclude that $\mathcal{K}, \mathbf{w} \models \blacklozenge(\varphi | \psi)$ directly from the semantics of \blacklozenge .

(\Leftarrow): Suppose $\mathcal{K}, \mathbf{w} \models \blacklozenge(\varphi | \psi)$. There is $\widehat{\mathcal{K}} = (\mathcal{W}, \widehat{R}, \mathcal{V})$ such that $\widehat{R} \subseteq R$ and $\widehat{R}(\mathbf{w}) = R(\mathbf{w})$, and there are $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K}_1 +_{\mathbf{w}} \mathcal{K}_2 = \widehat{\mathcal{K}}$, $(\mathcal{K}_1, \mathbf{w}) \models \varphi$ and $(\mathcal{K}_2, \mathbf{w}) \models \psi$. Consider now the set $\overline{R} = R \setminus \widehat{R}$. We define:

$$R'_1 \stackrel{\text{def}}{=} R_1 \cup \{(\mathbf{w}', \mathbf{w}'') \in \overline{R} \mid \mathbf{w}' \notin R_1^*(\mathbf{w})\}, \quad R'_2 \stackrel{\text{def}}{=} R_2 \cup (\overline{R} \setminus R'_1).$$

By definition, $R'_1|_{\mathbf{w}} = R_1|_{\mathbf{w}}$ and $R'_2|_{\mathbf{w}} = R_2|_{\mathbf{w}}$. Moreover, $R'_1 \cap R'_2 = \emptyset$ and $R'_1 \cup R'_2 = R$, and so $(\mathcal{W}, R'_1, \mathcal{V}) + (\mathcal{W}, R'_2, \mathcal{V}) = \mathcal{K}$. By Lemma 9.3, $(\mathcal{W}, R'_1, \mathcal{V}), \mathbf{w} \models \varphi$ and $(\mathcal{W}, R'_2, \mathcal{V}), \mathbf{w} \models \psi$. We conclude that $\mathcal{K}, \mathbf{w} \models \varphi \ast \psi$. \square

9.2 $\text{ML}(\ast)$ IS STRICTLY LESS EXPRESSIVE THAN $\text{ML}(\mid)$

In this section, we focus on the expressivity of $\text{ML}(\ast)$. For simplicity, given two logics \mathcal{L}_1 and \mathcal{L}_2 interpreted on the same class of structures, we write $\mathcal{L}_1 \preceq \mathcal{L}_2$ whenever the logic \mathcal{L}_1 is at most as expressive as \mathcal{L}_2 , i.e. every formula in \mathcal{L}_1 can be characterised in \mathcal{L}_2 . Moreover, we write $\mathcal{L}_1 \approx \mathcal{L}_2$ whenever the two logics have the same expressive power, i.e. $\mathcal{L}_1 \preceq \mathcal{L}_2$ and $\mathcal{L}_2 \preceq \mathcal{L}_1$, and $\mathcal{L}_1 \prec \mathcal{L}_2$ whenever \mathcal{L}_2 is strictly more expressive than \mathcal{L}_1 , i.e. $\mathcal{L}_1 \preceq \mathcal{L}_2$ and $\mathcal{L}_1 \not\approx \mathcal{L}_2$.

It is known that the standard modal logic ML is strictly less expressive than the graded modal logic GML [15, 51]. Together with the results in Chapter 7 (Corollary 7.18), this implies that $\text{ML} \prec \text{GML} \approx \text{ML}(\mid)$. We show that $\text{ML}(\ast)$ sits strictly between ML and GML , and it is therefore less expressive than $\text{ML}(\mid)$. To establish that $\text{ML}(\ast) \prec \text{GML}$, we first show $\text{ML}(\ast) \preceq \text{GML}$ and then we prove the strictness of the inclusion. The former result takes advantage of the notion of g-bisimulation, i.e. the underlying structural indistinguishability relation of GML , studied in [51]. For the latter result, we design an ad hoc notion of Ehrenfeucht-Fraïssé games for $\text{ML}(\ast)$, and define a simple formula in GML that cannot be expressed in $\text{ML}(\ast)$.

9.2.1 $\text{ML}(\ast)$ is not more expressive than GML .

To establish that $\text{ML}(\ast) \preceq \text{GML}$, let us first use the equivalence $\varphi * \psi \equiv \blacklozenge(\varphi \mid \psi)$ proved in Lemma 9.4, together with the fact that, by Corollary 7.18, given φ and ψ in GML , there is a formula χ in GML such that $\varphi \mid \psi \equiv \chi$. These two properties allows us to translate every formula of $\text{ML}(\ast)$ into an equivalent formula where the separating conjunction is only used in order to define the operator \blacklozenge . In order to prove that $\text{ML}(\ast) \preceq \text{GML}$, it is thus sufficient to show that GML is closed under the operator \blacklozenge , i.e. for every formula φ in GML there is a formula ψ in GML such that $\psi \equiv \blacklozenge \varphi$. To do so, we rely on the indistinguishability relation of GML , called g-bisimulation [51].

A g-bisimulation is a refinement of the classical back-and-forth conditions of a bisimulation (see e.g. [15]), tailored towards capturing graded modalities. It relates models with similar structural properties, but up to parameters $m, k \in \mathbb{N}$ and $P \subseteq_{\text{fin}} \text{AP}$, responsible for the modal degree, the graded rank and the set of atomic propositions used, respectively. Given a formula φ in GML , its *graded rank* $\text{gr}(\varphi)$ is the maximal coefficient k appearing in a subformula $\Diamond_{\geq k} \psi$ of φ . The following invariance result holds: g-bisimilar models (up to m, k and P) are modally equivalent in GML , i.e. no formula of GML with modal degree and graded rank bounded by m and k respectively, and written with symbols from P , can be satisfied by exactly one of the two models. Let us formalise the notion of g-bisimulation.

Definition 9.5 (g-bisimulation, [51]). Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$ be two finite forests. Let $m, k \in \mathbb{N}$ and $P \subseteq_{\text{fin}} \text{AP}$. A *g-bisimulation* up to (m, k, P) between \mathcal{K} and \mathcal{K}' is a sequence of $m + 1$ k -uple $\mathcal{Z}^0 = (\mathcal{Z}_1^0, \mathcal{Z}_2^0, \dots, \mathcal{Z}_k^0), \dots, \mathcal{Z}^m = (\mathcal{Z}_1^m, \mathcal{Z}_2^m, \dots, \mathcal{Z}_k^m)$ satisfying:

- (init) \mathcal{Z}_1^0 is not empty and for every $i \in [1, k]$ and $j \in [0, m]$, $\mathcal{Z}_i^j \subseteq 2^{\mathcal{W}} \times 2^{\mathcal{W}'}$,
- (refine) for every $i \in [1, k]$ and $j \in [1, m]$, $\mathcal{Z}_i^j \subseteq \mathcal{Z}_i^{j-1}$,
- (size) if $(X, Y) \in \mathcal{Z}_i^j$ then $\text{card}(X) = \text{card}(Y) = i$,
- (atoms) if $(\{w\}, \{w'\}) \in \mathcal{Z}_1^0$ then for every $p \in P$, $w \in \mathcal{V}(p)$ if and only if $w' \in \mathcal{V}'(p)$,
- (m-forth) if $(\{w\}, \{w'\}) \in \mathcal{Z}_1^{j+1}$ then for every $X \subseteq R(w)$ with $\text{card}(X) \in [1, k]$, there is $Y \subseteq R'(w')$ such that $(X, Y) \in \mathcal{Z}_{\text{card}(X)}^j$,

- (**m-back**) if $(\{w\}, \{w'\}) \in \mathcal{Z}_1^{j+1}$ then for every $Y \subseteq R'(w')$ with $\text{card}(Y) \in [1, k]$, there is $X \subseteq R(w)$ such that $(X, Y) \in \mathcal{Z}_{\text{card}(Y)}^j$,
- (**g-forth**) if $(X, Y) \in \mathcal{Z}_i^j$ then for all $w \in X$ there is $w' \in Y$ such that $(\{w\}, \{w'\}) \in \mathcal{Z}_1^j$,
- (**g-back**) if $(X, Y) \in \mathcal{Z}_i^j$ then for all $w' \in Y$ there is $w \in X$ such that $(\{w\}, \{w'\}) \in \mathcal{Z}_1^j$.

We write $(\mathcal{K}, w) \leftrightharpoons_{m,k}^P (\mathcal{K}', w')$ (and say that the two models are g-bisimilar) if and only if there is a g-bisimulation up to (m, k, P) between \mathcal{K} and \mathcal{K}' , say $\mathcal{Z}^0, \dots, \mathcal{Z}^m$, such that $(\{w\}, \{w'\}) \in \mathcal{Z}_1^m$.

To grasp the intuition behind g-bisimulation, let us look at the two conditions (**m-forth**) and (**g-forth**). In the standard bisimulation for ML , given two bisimilar structures (\mathcal{K}, w) and (\mathcal{K}', w') , for every child w_1 of w there is a child w'_1 of w' such that (\mathcal{K}, w'_1) and (\mathcal{K}', w'_1) are also bisimilar. Together with the other direction of the bisimulation, swapping (\mathcal{K}, w) and (\mathcal{K}', w') , this condition essentially characterises the modality \Diamond . The g-bisimulation is defined on the same principle, with the intent of characterising the graded modality $\Diamond_{\geq i}\varphi$. Together, (**m-forth**) and (**g-forth**) state that if (\mathcal{K}, w) and (\mathcal{K}', w') are g-bisimilar, then given children w_1, \dots, w_i of w , with $i \in [1, k]$, there are children w'_1, \dots, w'_i of w' such that, for every $l \in [1, i]$, (\mathcal{K}, w'_l) and (\mathcal{K}', w'_l) are also g-bisimilar. As we can see, for $i = 1$ we obtain exactly the definition of bisimulation, and indeed $\Diamond_{\geq 1}\varphi$ is equivalent to $\Diamond\varphi$.

Given $m, k \in \mathbb{N}$ and $P \subseteq_{\text{fin}} \text{AP}$, we write $\text{GML}[m, k, P]$ to denote the set of GML formulae ψ having $\text{md}(\psi) \leq m$, $\text{gr}(\psi) \leq k$ and propositional variables from P . $\text{GML}[m, k, P]$ is finite up to logical equivalence [51] (see Lemma 4.16 and its proof in Appendix B for a similar result).

Definition 9.6 (Indistinguishable forests). Given two pointed forests (\mathcal{K}, w) and (\mathcal{K}', w') , we write $(\mathcal{K}, w) \equiv_{m,k}^P (\mathcal{K}', w')$ whenever (\mathcal{K}, w) and (\mathcal{K}', w') are $\text{GML}[m, k, P]$ -indistinguishable, i.e. for every ψ in $\text{GML}[m, k, P]$, $(\mathcal{K}, w) \models \psi$ if and only if $(\mathcal{K}', w') \models \psi$.

We write $\mathcal{T}^P(m, k)$ to denote the quotient set induced by the equivalence relation $\equiv_{m,k}^P$. Let $m \leq m'$, $k \leq k'$ and $P \subseteq P'$. Since $\text{GML}[m, k, P] \subseteq \text{GML}[m', k', P']$, we have $\equiv_{m',k'}^{P'} \subseteq \equiv_{m,k}^P$. Moreover, as $\text{GML}[m, k, P]$ is finite up to logical equivalence, the set $\mathcal{T}^P(m, k)$ is finite, as stated in the proposition below, which summarises results in [51].

Proposition 9.7 ([51]). Let (\mathcal{K}, w) and (\mathcal{K}', w') be two pointed forests.

1. $(\mathcal{K}, w) \leftrightharpoons_{m,k}^P (\mathcal{K}', w')$ if and only if $(\mathcal{K}, w) \equiv_{m,k}^P (\mathcal{K}', w')$.
2. $\leftrightharpoons_{m,k}^P$ is a finite index equivalence relation. $\mathcal{T}^P(m, k)$ is finite.
3. $(\mathcal{K}, w) \leftrightharpoons_{m,k}^P ((\mathcal{W}, R|_w, \mathcal{V}), w)$, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

Whereas (1) is proved by induction on m , (2) holds from (1) and the finiteness of $\text{GML}[m, k, P]$ (up to logical equivalence). (3) holds from (2) together with Proposition 9.3, as $R|_w = R|_w^{\leq \text{card}(R)}$.

To establish that GML is closed under \blacklozenge , we show that there is a function $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ such that for all $m, k \in \mathbb{N}$ and $P \subseteq_{\text{fin}} \text{AP}$, if two pointed forests are in the same equivalence class of $\equiv_{m,f(m,k)}^P$, then they satisfy the same formulae of the form $\blacklozenge\varphi$, where φ is in $\text{GML}[m, k, P]$. By standard arguments (see e.g. Theorem 5.46) and from the fact that $\text{GML}[m, f(m, k), P]$ is finite up to logical equivalence, we then conclude that $\blacklozenge\varphi$ is equivalent to a formula in $\text{GML}[m, f(m, k), P]$. As we are not interested in the size of the equivalent formula, we can simply use the cardinality of the finite set $\mathcal{T}^P(m, k)$ in order to inductively define a suitable function:

$$f(0, k) \stackrel{\text{def}}{=} k, \quad f(m + 1, k) \stackrel{\text{def}}{=} k \times (\text{card}(\mathcal{T}^P(m, f(m, k))) + 1).$$

Essentially, the definition of $f(m+1, k)$ gives a bound to $\Diamond_{\geq i}$ modalities that is more than k times the number of equivalence classes in $\mathcal{T}^P(m, f(k, m))$. As $\text{GML}[m, k, P] \subseteq \text{GML}[m', k', P']$ holds for all $m \leq m'$, $k \leq k'$ and $P \subseteq P'$, we conclude that the map f is monotonic in both components. In fact, one can show that it is a tetration function. To prove that f satisfies the required properties, we show a technical \blacklozenge -simulation lemma that is similar to the simulation lemmata used in Chapter 5 to characterise a separation logic in terms of core formulae.

Lemma 9.8. Let $(\mathcal{K}, w) \equiv_{m, f(m, k)}^P (\mathcal{K}', w')$ where $m, k \in \mathbb{N}$, $P \subseteq_{\text{fin}} \text{AP}$, $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$. For every $R_1 \subseteq R$ there is $R'_1 \subseteq R'$ such that

1. $((\mathcal{W}, R_1, \mathcal{V}), w) \equiv_{m, k}^P ((\mathcal{W}', R'_1, \mathcal{V}'), w')$,
2. if $R_1(w) = R(w)$, then $R'_1(w') = R'(w')$.

The proof of Lemma 9.8 is by induction on m . The condition (2) serves in the proof of Lemma 9.9 (below), as it allows us to capture the semantics of \blacklozenge , by preserving the children of the world w' . In the proof, we rely on the properties of g-bisimulations [51] to define a binary relation \leftrightarrow between worlds of $R(w)$ and $R'(w')$. Every $w_1 \leftrightarrow w'_1$ is such that $(\mathcal{K}, w_1) \equiv_{m-1, f(m-1, k)}^P (\mathcal{K}', w'_1)$. The operator \blacklozenge does not necessarily preserve the children of w_1 and w'_1 , so that the induction hypothesis, naturally defined from the statement of Lemma 9.8, is applied on models where the premise of the condition (2) may not hold. We show that for all $R_1 \subseteq R$, it is possible to construct $R'_1 \subseteq R'$ such that, for all $w_1 \leftrightarrow w'_1$, $((\mathcal{W}, R_1, \mathcal{V}), w_1) \equiv_{m-1, k}^P ((\mathcal{W}', R'_1, \mathcal{V}'), w'_1)$. The result is then lifted to $((\mathcal{W}, R_1, \mathcal{V}), w) \equiv_{m, k}^P ((\mathcal{W}', R'_1, \mathcal{V}'), w')$, again thanks to the properties of the g-bisimulation.

For simplicity, let us write $\mathcal{T}^P(m, k)$ for the set $\mathcal{T}^P(m, f(m, k))$. Then, $\mathcal{T}^P(0, k) = \mathcal{T}^P(0, k)$, and for every $m \geq 1$, $\mathcal{T}^P(m, k) = \mathcal{T}^P(m, k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1))$. In the context of Lemma 9.8, which we now prove, we also notice that, by hypothesis, the pointed forests (\mathcal{K}, w) and (\mathcal{K}', w') belong to the same equivalence class in $\mathcal{T}^P(m, k)$, whereas the pointed forests in the condition (1) are “only” in the same equivalence class of $\mathcal{T}^P(m, k)$.

Proof. In the case $k = 0$, the lemma is trivially satisfied, as every formula in $\text{GML}[m, 0, P]$ is equivalent to a formula in the propositional calculus built over propositional variables in P . Indeed, the only graded modality appearing in formulae of $\text{GML}[m, 0, P]$ is $\Diamond_{\geq 0}$, and $\Diamond_{\geq 0}\psi$ is logically equivalent to \top .

Let us assume $k \geq 1$. We prove the result by induction on the modal depth m . In this proof, we switch quite freely between $\equiv_{m, k}^P$ and $\leftrightarrow_{m, k}^P$, which by Proposition 9.7 are the same relation, depending on what properties of pointed forests we want to focus on. The induction step is articulated in three main steps:

- (I) definition and proof of various properties of the two models (\mathcal{K}, w) and (\mathcal{K}', w') ,
- (II) definition of a strategy to define R'_1 from R' , that closely follows the relationship between R and R_1 with respect to the children of w ,
- (III) the proof that $((\mathcal{W}, R_1, \mathcal{V}), w)$ and $((\mathcal{W}', R'_1, \mathcal{V}'), w')$ satisfy the conditions (1) and (2) required by Lemma 9.8.

Let us begin with the base case.

base case: $m = 0$. In this case, let us work with $\equiv_{m, k}^P$. As $m = 0$, we have that (\mathcal{K}, w) and (\mathcal{K}', w') satisfy the same set of formulae of $\text{GML}[0, f(0, k), P]$. By definition, all these formulae have modal depth 0, and thus are Boolean combinations of atomic propositions. So, in general, for every $j \in \mathbb{N}$, two pointed forests are in the relation $\equiv_{0, j}^P$ if and only

if their current worlds satisfy the same atomic propositions from P . It is then sufficient to define R'_1 as R in order to conclude that $((\mathcal{W}, R_1, \mathcal{V}), w) \equiv_{0,k}^P ((\mathcal{W}', R'_1, \mathcal{V}'), w')$, and if $R_1(w) = R(w)$ then $R'_1(w') = R'(w')$.

Induction step: $m \geq 1$. By definition of $f(m, k)$ and Proposition 9.7, we have

$$(\mathcal{K}, w) \xrightarrow{P}_{m, k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)} (\mathcal{K}', w').$$

Let us explain the main idea of the proof. Let us pick a child w_1 of w in \mathcal{K} . Obviously, the pointed forest (\mathcal{K}, w_1) belongs to a specific equivalence class $T \in \mathcal{T}^P(m-1, k)$. The effect of reducing R to R_1 is that w_1 , together with the updated model, “jumps” to an equivalence class $T_1 \in \mathcal{T}^P(m-1, k)$, meaning that $((\mathcal{W}, R_1, \mathcal{V}), w_1)$ can a priori satisfy different formulae than (\mathcal{K}, w_1) , despite having the same current world. To prove the result, we need to show that there is a child $w'_1 \in R'(w')$ such that (\mathcal{K}', w'_1) is in the same equivalence class of (\mathcal{K}, w_1) , i.e. T , and that it is possible to modify R' into R'_1 in order to make w'_1 (and the updated model) “jump” to the equivalence class T_1 . The main difficulty is that we need to do this for all the children of w and w' , respecting the constraints of the g-bisimulation. The key step is to show that the graded rank $k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)$ is all we need in order to carry on the construction of R'_1 so that the resulting models are g-bisimilar up to (m, k, P) . Let us now formalise the proof, which requires some intermediate steps that are below [highlighted].

We start by considering a single equivalence class $T \in \mathcal{T}^P(m-1, k)$ (in fact, our proof is done modularly on these classes). We introduce the sets $R(w)|_T$ and $R'(w')|_T$, defined as

$$R(w)|_T \stackrel{\text{def}}{=} \{w_1 \in R(w) \mid (\mathcal{K}, w_1) \in T\}, \quad R'(w')|_T \stackrel{\text{def}}{=} \{w'_1 \in R'(w') \mid (\mathcal{K}', w'_1) \in T\}.$$

That is, $R(w)|_T$ contains the children w_1 of w such that (\mathcal{K}, w_1) belongs to T , and $R'(w')|_T$ is analogous but with respect to (\mathcal{K}', w') .

It is fairly simple to see that the following property holds:

$$(A) \quad \min(\text{card}(R(w)|_T), k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)) \\ = \min(\text{card}(R'(w')|_T), k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1))$$

Indeed, *ad absurdum* suppose that (A) does not hold, and thus $R(w)|_T$ and $\text{card}(R'(w')|_T)$ have different cardinalities, where at least one of them has cardinality less than the bound on the graded rank $k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)$. Let us assume $\text{card}(R(w)|_T) < \text{card}(R'(w')|_T)$ (the other case is symmetrical) and so

$$\text{card}(R(w)|_T) < \min(\text{card}(R'(w')|_T), k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)).$$

From $\mathcal{K}, w \xrightarrow{P}_{m, k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)} \mathcal{K}', w'$, there is a g-bisimulation between \mathcal{K} and \mathcal{K}' , say $\mathcal{Z}^0, \dots, \mathcal{Z}^j = (\mathcal{Z}_1^j, \dots, \mathcal{Z}_{k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)}^j), \dots, \mathcal{Z}^m$, such that $(\{w\}, \{w'\}) \in \mathcal{Z}_1^m$. Let Y be a subset of $R'(w')|_T$ of cardinality

$$\text{card}(Y) = \min(\text{card}(R'(w')|_T), k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)).$$

By (m-back), there is $X \subseteq R(w)$ such that $(X, Y) \in \mathcal{Z}_{\text{card}(Y)}^{m-1}$. By (size), $\text{card}(X) = \text{card}(Y)$, which implies that there is a world $w_2 \in X$ such that $(\mathcal{K}, w_2) \notin T$. From (g-forth), there is $w'_2 \in Y$ such that $(\{w_2\}, \{w'_2\}) \in \mathcal{Z}_1^{m-1}$. So, by definition of g-bisimulation,

$$(\mathcal{K}, w_2) \xrightarrow{P}_{m-1, k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)} (\mathcal{K}', w'_2).$$

As we know, for all $j < j'$ we have $\equiv_{m-1, j'}^P \subseteq \equiv_{m-1, j}^P$. Since, by Proposition 9.7, $\equiv_{m, k}^P$ equals to $\xrightarrow{P}_{m, k}$, the same property holds true for $\xrightarrow{P}_{m, k}$. Moreover,

$$\mathfrak{f}(m-1, k) \leq k \times (\text{card}(\mathcal{T}^P(m-2, k)) + 1) \leq k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1),$$

which allows us to derive $(\mathcal{K}, w_2) \xleftarrow{P}_{m-1, \mathfrak{f}(m-1, k)} (\mathcal{K}', w'_2)$. Notice that the set of equivalence classes induced by $\xleftarrow{P}_{m-1, \mathfrak{f}(m-1, k)}$ is $\mathcal{T}^P(m-1, k)$. However, this implies that (\mathcal{K}, w_2) and (\mathcal{K}', w'_2) belong to the same class of $\mathcal{T}^P(m-1, k)$, which is contradictory, as we have $(\mathcal{K}, w_2) \notin T$ and $(\mathcal{K}', w'_2) \in T$ (where $T \in \mathcal{T}^P(m-1, k)$). Therefore, (A) holds.

Given an equivalence class $T' \in \mathcal{T}^P(m-1, k)$, we define the set below

$$R_1(w)|_{T \blacktriangleright T'} \stackrel{\text{def}}{=} R(w)|_T \cap R_1(w)|_{T'}.$$

Following the proof idea presented above, a world $w_1 \in R_1(w)|_{T \blacktriangleright T'}$ is a child of w such that (\mathcal{K}, w_1) is in the class T and “jumps” to the class T' when updating the accessibility relation from R to R_1 . We are interested in the following key property:

(B) For every $w_1 \in R_1(w) _{T \blacktriangleright T'}$ and $w'_1 \in R'(w') _T$ there is $R'_{1, w_1} \subseteq R' _{w'_1}$ such that $((\mathcal{W}, R_1 _{w_1}, \mathcal{V}), w_1) \xleftarrow{P}_{m-1, k} ((\mathcal{W}', R'_{1, w'_1}, \mathcal{V}'), w'_1)$
--

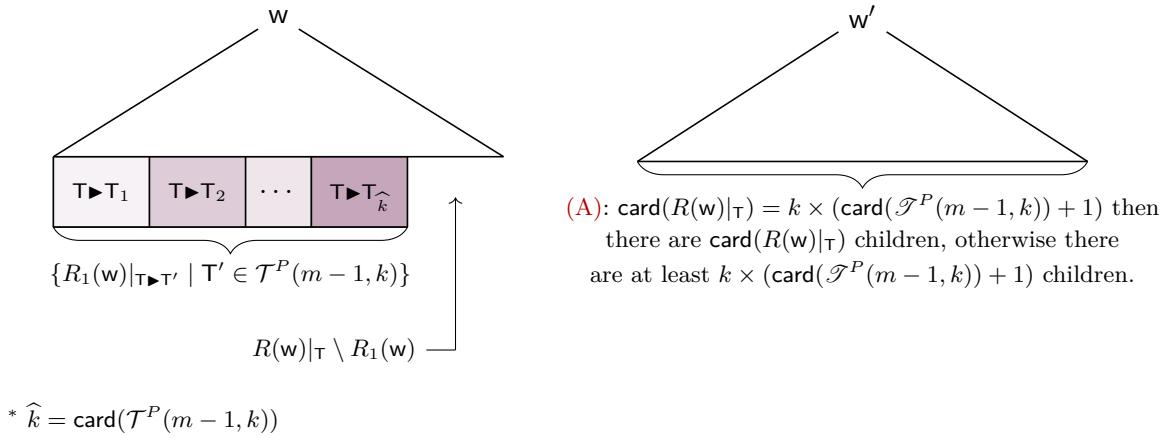
Let us prove (B). By definition, we have $w_1 \in R(w)|_T$ and $w'_1 \in R'(w')|_T$, and thus $(\mathcal{K}, w_1) \xleftarrow{P}_{m-1, \mathfrak{f}(m-1, k)} (\mathcal{K}', w'_1)$. From the equivalence of $\xleftarrow{P}_{m-1, \mathfrak{f}(m-1, k)}$ and $\equiv^P_{m-1, \mathfrak{f}(m-1, k)}$, together with Proposition 9.3, we conclude that $((\mathcal{W}, R|_{w_1}, \mathcal{V}), w_1)$ and $((\mathcal{W}', R'|_{w'_1}, \mathcal{V}'), w'_1)$ belong to T . Moreover, from $R_1 \subseteq R$ we conclude that $R_1|_{w_1} \subseteq R|_{w_1}$. This allows us to apply the induction hypothesis (notice that the modal degree is now $m-1$), and conclude that there is $R'_{1, w_1} \subseteq R'|_{w'_1}$ such that $((\mathcal{W}, R_1|_{w_1}, \mathcal{V}), w_1) \xleftarrow{P}_{m-1, k} ((\mathcal{W}', R'_{1, w'_1}, \mathcal{V}'), w'_1)$, concluding the proof of (B).

This intermediate result gives us an important information: every single “jump” (as informally expressed above) done while updating the accessibility relation of \mathcal{K} can be mimicked by updating \mathcal{K}' . An important missing piece is proving that all jumps can be simultaneously mimicked. To prove this, we start by considering the following partition of $R(w)|_T$:

$$R(w)|_{T \blacktriangleright R_1} \stackrel{\text{def}}{=} \{R_1(w)|_{T \blacktriangleright T'} \mid T' \in \mathcal{T}^P(m-1, k)\} \cup \{R(w)|_T \setminus R_1(w)\}.$$

Informally, $R(w)|_{T \blacktriangleright R_1}$ partitions the children of w in $R(w)|_T$ in different sets depending on what is the set $T' \in \mathcal{T}^P(m-1, k)$ they “jump” to. One additional set, i.e. $R(w)|_T \setminus R_1(w)$, contains all the children of w in $R(w)|_T$ that are lost when updating R to R_1 . To be completely formal, let us first prove that $R(w)|_{T \blacktriangleright R_1}$ is a partition of $R(w)|_T$. Indeed, $R(w)|_T$ can be written as $(R(w)|_T \cap R_1(w)) \cup (R(w)|_T \setminus R_1(w))$. Moreover, by definition of $\mathcal{T}^P(m-1, k)$ as the quotient set of $\xleftarrow{P}_{m-1, k}$, we have $R_1(w) = \bigcup_{T' \in \mathcal{T}^P(m-1, k)} R_1(w)|_{T'}$. Lastly, $R(w)|_T \cap \bigcup_{T' \in \mathcal{T}^P(m-1, k)} R_1(w)|_{T'}$ is equivalent to $\bigcup_{T' \in \mathcal{T}^P(m-1, k)} (R(w)|_T \cap R_1(w)|_{T'})$, which leads to the definition of the partition $R(w)|_{T \blacktriangleright R_1}$ from the definition of $R_1(w)|_{T \blacktriangleright T'}$ together with the remaining component $R(w)|_T \setminus R_1(w)$. Figure 9.3 presents schematically the results we have shown so far, only considering the children of w in $R(w)|_T$ (on the left) and the children of w' in $R'(w')|_T$ (on the right).

To work towards the definition of R'_1 (as required by the statement of the lemma), we now deal with the children in $R'(w')|_T$ and find suitable subsets of R'_1 in order to define a partition of $R'(w')|_T$ that is similar to $R(w)|_{T \blacktriangleright R_1}$, where by “similar” we mean that we will be able to construct a g-bisimulation using this partition. More precisely, we show that:

Figure 9.3: $R(w)|_T$ (on the left), and $R'(w')|_T$ (on the right).

(C) It is possible to construct a family of sets

$$R'(w')|_{T \rightsquigarrow T'} \quad (\text{for every } T' \in \mathcal{T}^P(m-1, k)), \quad \mathcal{G}_T,$$

satisfying the following properties:

1. For all $T' \in \mathcal{T}^P(m-1, k)$, $R'(w')|_{T \rightsquigarrow T'}$ is a set of pairs (R'_{1,w'_1}, w'_1) s.t.
 $w'_1 \in R'(w')|_T, \quad R'_{1,w'_1} \subseteq R'|_{w'_1} \quad \text{and} \quad ((\mathcal{W}', R'_{1,w'_1}, \mathcal{V}'), w'_1) \in T',$
2. each world of $R'(w')|_T$ appears in exactly one set among $R'(w')|_{T \rightsquigarrow T'}$ ($T' \in \mathcal{T}^P(m-1, k)$) and \mathcal{G}_T ,
3. $\min(\text{card}(R_1(w)|_{T \blacktriangleright T'}), k) = \min(\text{card}(R'(w')|_{T \rightsquigarrow T'}), k),$
 $(T' \in \mathcal{T}^P(m-1, k)),$
4. $\min(\text{card}(R(w)|_T \setminus R_1(w)), k) = \min(\text{card}(\mathcal{G}_T), k).$

The first property of (C) basically requires us to modify R' so that the children of $R'(w')|_T$ “jumps” to specific sets in $\mathcal{T}^P(m-1, k)$, in line with the developments that lead to the proof of (B). The set \mathcal{G}_T is dedicated to children of w' worlds that should be made inaccessible from w' , in the final relation R'_1 . The updates to R' cannot be arbitrary, and this is where the third and fourth properties come into play. These properties impose cardinality constraints on the sets we construct, in line with the graded rank k that is used in the equivalence relation $\leftrightharpoons_{m,k}^P$. For example, suppose that for a given set T' we have $\text{card}(R_1(w)|_{T \blacktriangleright T'}) < k$. Then, we need to select exactly $\text{card}(R_1(w)|_{T \blacktriangleright T'})$ children in $R'(w')|_T$ and modify R' so that all of them can be used to define the set $R'(w')|_{T \rightsquigarrow T'}$. If instead $\text{card}(R_1(w)|_{T \blacktriangleright T'}) \geq k$, it is possible to select an arbitrary amount of children from $R'(w')|_T$, as long as they are at least k . Again, after selecting these children we need to modify R' so that they define the set $R'(w')|_{T \rightsquigarrow T'}$. To comply with these two last properties we rely on (A).

The proof of (C) distinguishes two cases (which are very similar in substance):

case: $\text{card}(R(w)|_T) < k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)$. By (A), $R'(w')|_T$ and $R(w)|_T$ have the same cardinality. Consider a bijection $\mathbf{g}: R(w)|_T \rightarrow R'(w')|_T$. Then define \mathcal{G}_T as the set $\{\mathbf{g}(w_1) \mid w_1 \in R(w)|_T \setminus R_1(w)\}$, so that the fourth property in (C) is trivially satisfied. In order to define the sets of the form $R'(w')|_{T \rightsquigarrow T'}$, we start by an

initialisation to the empty set \emptyset and then we populate them. Iteratively, for every $T' \in \mathcal{T}^P(m-1, k)$ and every $w_1 \in R_1(w)|_{T \blacktriangleright T'}$, consider $g(w_1)$. By (B), there is $R'_{1,g(w_1)} \subseteq R'|_{g(w_1)}$ such that $((\mathcal{W}, R_1|_{w_1}, \mathcal{V}), w_1) \xrightarrow{P}_{m-1,k} ((\mathcal{W}', R'_{1,g(w_1)}, \mathcal{V}'), g(w_1))$. By Proposition 9.3, it follows that $((\mathcal{W}, R_1, \mathcal{V}), w_1) \xrightarrow{P}_{m-1,k} ((\mathcal{W}', R'_{1,g(w_1)}, \mathcal{V}'), g(w_1))$ and therefore $((\mathcal{W}', R'_{1,g(w_1)}, \mathcal{V}'), g(w_1)) \in T'$. Then, add to $R'(w')|_{T \rightsquigarrow T'}$ the pair $(R'_{1,g(w_1)}, g(w_1))$. Notice that this pair satisfies the constraints required in the first property of (C). After the iterations over all $T' \in \mathcal{T}^P(m-1, k)$ and $w_1 \in R_1(w)|_{T \blacktriangleright T'}$, the construction is completed. As we are guided by the bijection g , the second property of (C) is satisfied.

case: $\text{card}(R(w)|_T) \geq k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)$. By (A), it follows that $R'(w')|_T$ has at least $k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)$ elements. For this case, it is easy to show that there is a set in the partition $R(w)|_{\blacktriangleright_{R_1}}$ of $R(w)|_T$ that has cardinality at least k . Indeed, *ad absurdum*, suppose all the sets in $R(w)|_{\blacktriangleright_{R_1}}$ are of cardinality less than k . As $R(w)|_{\blacktriangleright_{R_1}}$ partitions $R(w)|_T$ and it contains $\text{card}(\mathcal{T}^P(m-1, k)) + 1$ sets (where the $+1$ refers to the set $R(w)|_T \setminus R_1(w)$) this would imply that

$$\text{card}(R(w)|_T) \leq (k-1) \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1).$$

However, this leads to a contradiction, as $\text{card}(\mathcal{T}^P(m-1, k)) \leq \text{card}(\mathcal{T}^P(m-1, k))$ and $\text{card}(R(w)|_T) \geq k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)$.

Hence, let Ω be a set in $R(w)|_{\blacktriangleright_{R_1}}$ that has at least k elements. For the construction, we initialise all the sets $R'(w')|_{T \rightsquigarrow T'}$ and \mathcal{G}_T to the empty set \emptyset and we show how to populate them. Moreover, we introduce an auxiliary set Δ which is initially equal to $R'(w')|_T$ and keeps track of which elements of this latter set have not been already used in the construction (and are hence available). Iteratively,

1. consider $T' \in \mathcal{T}^P(m-1, k)$ s.t. $R_1(w)|_{T \blacktriangleright T'} \neq \Omega$ and that was not already treated,
2. select $\beta = \min(\text{card}(R_1(w)|_{T \blacktriangleright T'}), k)$ worlds, say w'_1, \dots, w'_β , from the set Δ .
3. Let $w_1 \in R_1(w)|_{T \blacktriangleright T'}$. By (B) we have that for each $i \in [1, \beta]$ there is $R'_{1,w'_i} \subseteq R'|_{w'_i}$ such that

$$((\mathcal{W}, R_1|_{w_1}, \mathcal{V}), w_1) \xrightarrow{P}_{m-1,k} ((\mathcal{W}', R'_{1,w'_i}, \mathcal{V}'), w'_i).$$

By Proposition 9.3, we have $((\mathcal{W}, R_1, \mathcal{V}), w_1) \xrightarrow{P}_{m-1,k} ((\mathcal{W}', R'_{1,w'_i}, \mathcal{V}'), w'_i)$ and therefore $((\mathcal{W}', R'_{1,w'_i}, \mathcal{V}'), w'_i) \in T'$. Define the set $R'(w')|_{T \rightsquigarrow T'}$ as

$$\{(R'_{1,w'_i}, w'_i) \mid i \in [1, \beta]\}.$$

Notice that by construction this set satisfies the first and third properties of (C).

4. Remove w'_1, \dots, w'_β from Δ (they will not be used in the successive iterations).

After this iterative construction, only two sets need to be handled: $R(w)|_T \setminus R_1(w)$ and Ω . The proof split in two cases, depending on whether $R(w)|_T \setminus R_1(w) = \Omega$.

case: $R(w)|_T \setminus R_1(w) = \Omega$. We simply add to \mathcal{G}_T every worlds in Δ . Notice that, in this case, the previous iteration removed (in total) at most $k \times \text{card}(\mathcal{T}^P(m-1, k))$ locations from Δ . As Δ was initialized to be $R'(w')|_T$, which has cardinality at least $k \times (\text{card}(\mathcal{T}^P(m-1, k)) + 1)$, we conclude that \mathcal{G}_T has at least k elements, and thus satisfies the fourth property of (C).

case: $R(w)|_T \setminus R_1(w) \neq \Omega$. First, we treat $R(w)|_T \setminus R_1(w)$ with the same four-point strategy used for the other sets: we select $\beta = \min(\text{card}(R(w)|_T \setminus R_1(w)), k)$ worlds, say w'_1, \dots, w'_β , from the pool of available worlds Δ ; we define \mathcal{G}_T as $\{w'_1, \dots, w'_\beta\}$ and we remove these worlds from Δ . By construction, \mathcal{G}_T satisfies

the fourth property of (C). Notice that, after this step, Δ still contains at least k worlds, exactly as in the previous case of the proof (before dealing with Ω). As $R(w)|_T \setminus R_1(w) \neq \emptyset$, there is $T' \in \mathcal{T}^P(m-1, k)$ s.t. $R_1(w)|_{T \blacktriangleright T'} = \Omega$. We select all the worlds still in Δ , say w'_1, \dots, w'_α . Let $w_1 \in R_1(w)|_{T \blacktriangleright T'}$. We apply (B) to derive $R'_{1,w'_i} \subseteq R'|_{w'_i}$ ($i \in [1, \alpha]$) such that

$$((\mathcal{W}, R_1|_{w_1}, \mathcal{V}), w_1) \leftrightharpoons_{m-1, k}^P ((\mathcal{W}', R'_{1,w'_i}, \mathcal{V}'), w'_i).$$

By Proposition 9.3, we have $((\mathcal{W}, R_1, \mathcal{V}), w_1) \leftrightharpoons_{m-1, k}^P ((\mathcal{W}', R'_{1,w'_i}, \mathcal{V}'), w'_i)$ and therefore $((\mathcal{W}', R'_{1,w'_i}, \mathcal{V}'), w'_i) \in T'$. Define the set $R'(w')|_{T \rightsquigarrow T'}$ as

$$\{(R'_{1,w'_i}, w'_i) \mid i \in [1, \beta]\}.$$

Notice that by construction this set satisfies the first and third properties of (C).

This ends the construction of the sets $R'(w')|_{T \rightsquigarrow T'}$ ($T' \in \mathcal{T}^P(m-1, k)$) and \mathcal{G}_T . No element in $R'(w')|_T$ is considered twice during the definition of $R'(w')|_{T \rightsquigarrow T'}$ and \mathcal{G}_T , which allows us to conclude that the second property is satisfied, and therefore that (C) holds.

A last note about (C): given two pairs (R_{1,w'_1}, w'_1) and (R_{1,w'_2}, w'_2) defined in (C), if $w'_1 \neq w'_2$ then R_{1,w'_1} and R_{1,w'_2} are disjoint. This holds directly from the second property, together with the fact that $R'_{1,w'_1} \subseteq R'|_{w'_1}$ and $R'_{1,w'_2} \subseteq R'|_{w'_2}$. Keeping this in mind, we are now ready to construct R'_1 .

We consider every $T \in \mathcal{T}^P(m-1, k)$ and apply (C) to construct the sets $R'(w')|_{T \rightsquigarrow T'}$ ($T' \in \mathcal{T}^P(m-1, k)$) and \mathcal{G}_T . Afterwards, the relation R'_1 is defined as follows:

$$R'_1 \stackrel{\text{def}}{=} \bigcup_{\substack{T \in \mathcal{T}^P(m-1, k) \\ T' \in \mathcal{T}^P(m-1, k) \\ (R'_{1,w'_1}, w'_1) \in R'(w')|_{T \rightsquigarrow T'}}} \{(w', w'_1)\} \cup R'_{1,w'_1}.$$

Clearly, we have that $R'_1 \subseteq R_1$. Moreover, from the properties of (C), it holds that for every $w'_1 \in R'_1(w)$, $R'_1|_{w'_1} = R'_{1,w'_1}$. In order to conclude the proof, we need to show that

- (1) $((\mathcal{W}, R_1, \mathcal{V}), w) \leftrightharpoons_{m, k}^P ((\mathcal{W}', R'_1, \mathcal{V}'), w')$ (recall that $\leftrightharpoons_{m, k}^P$ is equal to $\equiv_{m, k}^P$),
- (2) if $R_1(w) = R(w)$ then $R'_1(w') = R'(w')$.

Let us first prove (2) by using the fourth property of (C). Suppose $R_1(w) = R(w)$ and hence $R(w) \setminus R_1(w) = \emptyset$. The set $R(w) \setminus R_1(w)$ can be written as $\bigcup_{T \in \mathcal{T}^P(m-1, k)} (R(w)|_T \setminus R_1(w))$. We conclude that $\text{card}(R(w)|_T \setminus R_1(w)) = 0$ for every $T \in \mathcal{T}^P(m-1, k)$. Similarly, $R'(w') \setminus R'_1(w')$ equals $\bigcup_{T \in \mathcal{T}^P(m-1, k)} (R'(w')|_T \setminus R'_1(w'))$. Notice that for every $T \in \mathcal{T}^P(m-1, k)$, a world $w'_1 \in R'(w')|_T \setminus R'_1(w')$ cannot be inside a pair of $R'(w')|_{T \rightsquigarrow T'}$ (for any $T' \in \mathcal{T}^P(m-1, k)$). Indeed, if this was the case, then $(w', w'_1) \in R'_1$ (see definition of R'_1) in contradiction with $w'_1 \in R'(w')|_T \setminus R'_1(w')$. Then $w'_1 \in \mathcal{G}_T$ and we derive $R'(w')|_T \setminus R'_1(w') = \mathcal{G}_T$ and $R'(w') \setminus R'_1(w') = \bigcup_{T \in \mathcal{T}^P(m-1, k)} \mathcal{G}_T$. By construction, every world $w'_1 \in R'(w)$ can appear in at most one set in $\{\mathcal{G}_T \mid T' \in \mathcal{T}^P(m-1, k)\}$ and hence $\text{card}(R'(w') \setminus R'_1(w')) = \sum_{T \in \mathcal{T}^P(m-1, k)} \text{card}(\mathcal{G}_T)$. We can now apply the fourth property of (C) which, together with $k \geq 1$ (assumed at the beginning of the proof) and $\text{card}(R(w)|_T \setminus R_1(w)) = 0$ leads to $\text{card}(R'(w') \setminus R'_1(w')) = 0$. As by definition $R'_1(w') \subseteq R'(w')$, this ends the proof of (2).

Lastly, let us prove (1). This is done by constructing a g-bisimulation $\mathcal{Z}^0, \dots, \mathcal{Z}^m$ up to (m, k, P) between $(\mathcal{W}, R_1, \mathcal{V})$ and $(\mathcal{W}', R'_1, \mathcal{V}')$ such that $(\{w\}, \{w'\}) \in \mathcal{Z}_1^m$. Here, we iteratively construct the g-bisimulation starting from the sets $\mathcal{Z}_1^j = \{(\{w\}, \{w'\})\}$ (for all $j \in [0, m]$). During the construction we make sure to always preserve the satisfaction of the conditions

(init), (refine), (size) and (atoms). Notice that these conditions hold for our initial sequence of relations. In particular, (atoms) holds directly from $(\mathcal{K}, \mathbf{w}) \equiv_{m, f(m, k)}^P (\mathcal{K}, \mathbf{w}')$. The construction can be split into four steps:

(m-forth-step) Let $X \subseteq R_1(\mathbf{w})$ be a set such that $\text{card}(X) \in [1, k]$. As required by the condition (m-forth), we want to pair this set with a suitable subset $Y \subseteq R'_1(\mathbf{w})$ of cardinality $\text{card}(X)$ so that it is possible to then satisfy the conditions (g-forth) and (g-back). Let us consider the partition of X defined as $\{X_{\mathsf{T}\blacktriangleright\mathsf{T}'} \mid \mathsf{T} \in \mathcal{T}^P(m-1, k) \text{ and } \mathsf{T}' \in \mathcal{T}^P(m-1, k)\}$ where $X_{\mathsf{T}\blacktriangleright\mathsf{T}'} = X \cap R_1(\mathbf{w})|_{\mathsf{T}\blacktriangleright\mathsf{T}'}$. We consider the set $R'(\mathbf{w}')|_{\mathsf{T}\rightsquigarrow\mathsf{T}'}$ and select $\text{card}(X_{\mathsf{T}\blacktriangleright\mathsf{T}'})$ worlds appearing in one of its pairs (which are of the form (R'_{1,w'_1}, w'_1)). Let $Y_{\mathsf{T}\rightsquigarrow\mathsf{T}'}$ be the set of these selected worlds. By (C) this set is guaranteed to exist and is such that every world w'_1 in it is also in $R'_1(\mathbf{w}')$. Let $Y = \bigcup_{\mathsf{T} \in \mathcal{T}^P(m-1, k), \mathsf{T}' \in \mathcal{T}^P(m-1, k)} Y_{\mathsf{T}\rightsquigarrow\mathsf{T}'}$. It is easy to see that $\text{card}(X) = \text{card}(Y)$. For every $j \in [0, m-1]$ we add (X, Y) to $\mathcal{Z}_{\text{card}(X)}^j$.

(m-back-step) Let $Y \subseteq R'_1(\mathbf{w})$ be a set such that $\text{card}(Y) \in [1, k]$. We focus on the condition (m-back), and work similarly to the previous step. Consider the partition of Y defined as $\{Y_{\mathsf{T}\rightsquigarrow\mathsf{T}'} \mid \mathsf{T} \in \mathcal{T}^P(m-1, k) \text{ and } \mathsf{T}' \in \mathcal{T}^P(m-1, k)\}$ where $Y_{\mathsf{T}\rightsquigarrow\mathsf{T}'}$ is defined as $Y \cap \{w'_1 \mid (R'_{1,w'_1}, w'_1) \in R'(\mathbf{w}')|_{\mathsf{T}\rightsquigarrow\mathsf{T}'} \text{ for some } R'_{1,w'_1}\}$. We select a subset $X_{\mathsf{T}\blacktriangleright\mathsf{T}'}$ of $R_1(\mathbf{w})|_{\mathsf{T}\blacktriangleright\mathsf{T}'}$ with cardinality $\text{card}(Y_{\mathsf{T}\rightsquigarrow\mathsf{T}'})$, whose existence is guaranteed by (C). Let $X = \bigcup_{\mathsf{T} \in \mathcal{T}^P(m-1, k), \mathsf{T}' \in \mathcal{T}^P(m-1, k)} X_{\mathsf{T}\blacktriangleright\mathsf{T}'}$. We have $\text{card}(Y) = \text{card}(X)$. For every $j \in [0, m-1]$, we add (X, Y) to $\mathcal{Z}_{\text{card}(Y)}^j$.

(g-forth-step) From the first two steps of the construction, the set \mathcal{Z}_i^j was updated with new pairs (X, Y) where every element in X is from $R_1(\mathbf{w})$ and every element of Y is from $R'_1(\mathbf{w})$. Consider then one of these pairs (X, Y) and a world $w_1 \in X$. Let $\mathsf{T} \in \mathcal{T}^P(m-1, k)$ and $\mathsf{T}' \in \mathcal{T}^P(m-1, k)$ be such that $w_1 \in R_1(\mathbf{w})|_{\mathsf{T}\blacktriangleright\mathsf{T}'}$. By construction (first and second steps above), there is $w'_1 \in Y$ such that for some $R'_{1,w'_1} \subseteq R'_1$ it holds that $(R'_{1,w'_1}, w'_1) \in R'(\mathbf{w}')|_{\mathsf{T}\rightsquigarrow\mathsf{T}'}$. Again from (C), $((\mathcal{W}, R_1, \mathcal{V}), w_1) \xrightarrow{P}_{m-1, k} ((\mathcal{W}', R'_1, \mathcal{V}'), w'_1)$. Since by definition $R'_{1,w'_1} = R'_1|_{w'_1}$, from Proposition 9.3 we obtain

$$((\mathcal{W}, R_1, \mathcal{V}), w_1) \xrightarrow{P}_{m-1, k} ((\mathcal{W}', R'_1, \mathcal{V}'), w'_1).$$

Let $\mathcal{Y}^0, \dots, \mathcal{Y}^{m-1}$ be the g-bisimulation up to $(m-1, k, P)$ of $(\mathcal{W}, R_1, \mathcal{V})$ and $(\mathcal{W}', R'_1, \mathcal{V}')$ such that $(\{w_1\}, \{w'_1\}) \in \mathcal{Y}_1^{m-1}$. For all $i \in [1, k]$ and all $j \in [0, m-1]$, add \mathcal{Y}_i^j to \mathcal{Z}_i^j .

(g-back-step) Symmetrically to the previous point of the construction, let us consider again a pair (X, Y) introduced by one of the two steps (m-forth-step) and (m-back-step). Let $w'_1 \in Y$. Then there is $\mathsf{T} \in \mathcal{T}^P(m-1, k)$ and $\mathsf{T}' \in \mathcal{T}^P(m-1, k)$ and $R'_{1,w'_1} \subseteq R'_1$ such that $(R'_{1,w'_1}, w'_1) \in R'(\mathbf{w}')|_{\mathsf{T}\rightsquigarrow\mathsf{T}'}$. By construction (steps (m-forth-step) and (m-back-step)), there is $w_1 \in X \cap R'(\mathbf{w})|_{\mathsf{T}\blacktriangleright\mathsf{T}'}$. As before, by (C) followed by Proposition 9.3, one can show that $(\mathcal{W}, R_1, \mathcal{V}), w_1 \xrightarrow{P}_{m-1, k} (\mathcal{W}', R'_1, \mathcal{V}'), w'_1$. Then, let $\mathcal{Y}^0, \dots, \mathcal{Y}^{m-1}$ be the g-bisimulation up to $(m-1, k, P)$ between $(\mathcal{W}, R_1, \mathcal{V})$ and $(\mathcal{W}', R'_1, \mathcal{V}')$ such that $\{w_1\} \mathcal{Y}_1^{m-1} \{w'_1\}$. For every $i \in [1, k]$ and every $j \in [0, m-1]$, update \mathcal{Z}_i^j to $\mathcal{Z}_i^j \cup \mathcal{Y}_i^j$.

It is simple to see that this construction leads to a sequence of relations $\mathcal{Z}^0, \dots, \mathcal{Z}^m$ that is a g-bisimulation up to (m, k, P) between $(\mathcal{W}, R_1, \mathcal{V})$ and $(\mathcal{W}', R'_1, \mathcal{V}')$ such that $\{\mathbf{w}\} \mathcal{Z}_1^m \{\mathbf{w}'\}$. The conditions (init), (refine), (size) and (atoms) hold at any point during the construction. For the other conditions, let (X, Y) be a pair in some \mathcal{Z}_i^j . If it was not introduced by the first two steps of the construction, then (X, Y) is a member of some set $\mathcal{Y}_i^j \subseteq \mathcal{Z}_i^j$ that is used in a g-bisimulation whose elements are all used to construct $\mathcal{Z}^0, \dots, \mathcal{Z}^m$ (third and fourth point of the

proof). Hence, w.r.t. (X, Y) no condition can be violated. If instead (X, Y) is added to the g-bisimulation during the first and second point of the construction, then by construction it is easy to check that it satisfies all the conditions. Therefore $((\mathcal{W}, R_1, \mathcal{V}), w) \xrightarrow{P}_{m,k} ((\mathcal{W}', R'_1, \mathcal{V}'), w')$, which ends the proof of the whole lemma. \square

Intuitively, Lemma 9.8 states that given two models satisfying the same formulae up to the parameters m and $f(m, k)$, we can extract submodels satisfying the same formulae up to m and k (reduced graded rank). This allows us to conclude that if φ is in GML, there is some GML formula equivalent to $\blacklozenge\varphi$ (Lemma 9.9). In other words, the operator \blacklozenge can be eliminated to obtain a GML formula. This, together with Lemma 9.4 and Corollary 7.18 entail $\text{ML}(\ast) \preceq \text{GML}$.

Lemma 9.9. For all φ in $\text{GML}[m, k, P]$ there is ψ in $\text{GML}[m, f(m, k), P]$ such that $\blacklozenge\varphi \equiv \psi$.

Proof. If $k = 0$, then the proof is by an easy verification as the formula φ from the statement is logically equivalent to a formula from the propositional calculus (each subformula $\Diamond_{\geq 0}\psi$ is logically equivalent to \top). Otherwise ($k \geq 1$), let $k^+ = k \times (\text{card}(\mathcal{T}^P(m - 1, k)) + 1)$. Since \equiv_{m, k^+}^P is finite index, there is a finite set $\{\chi_1, \dots, \chi_q\} \subseteq \text{GML}[m, k^+, P]$ such that

- $\chi_1 \vee \dots \vee \chi_q$ is valid, for every $i \in [1, q]$, χ_i is satisfiable,
- for all $i \neq j \in [1, Q]$, $\chi_i \wedge \chi_j$ is unsatisfiable,
- For all (\mathcal{K}, w) and (\mathcal{K}', w') , $(\mathcal{K}, w) \equiv_{m, k^+}^P (\mathcal{K}', w')$ iff there is $i \in [1, q]$ such that $(\mathcal{K}, w) \models \chi_i$ and $(\mathcal{K}', w') \models \chi_i$.

Let ψ be the formula $\bigvee \{\chi_i \mid \text{there is } (\mathcal{K}, w) \text{ s.t. } (\mathcal{K}, w) \models \chi_i \wedge \blacklozenge\varphi\}$. We show that ψ is logically equivalent to $\blacklozenge\varphi$.

First, let us suppose that $(\mathcal{K}, w) \models \blacklozenge\varphi$. As $\chi_1 \vee \dots \vee \chi_q$ is valid, there is $i \in [1, q]$ such that $(\mathcal{K}, w) \models \chi_i$. Therefore χ_i occurs in ψ and consequently, $(\mathcal{K}, w) \models \psi$.

Conversely, suppose that $(\mathcal{K}, w) \models \psi$ with $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. So, there is χ_i occurring in ψ such that $(\mathcal{K}, w) \models \chi_i$ and there exist a model $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$ and $w' \in \mathcal{W}'$ such that $(\mathcal{K}', w') \models \chi_i \wedge \blacklozenge\varphi$. From the equisatisfaction of χ_i , we have $(\mathcal{K}, w) \equiv_{m, k^+}^P (\mathcal{K}', w')$. From $(\mathcal{K}', w') \models \blacklozenge\varphi$, there is $R'_1 \subseteq R'$ such that $R'_1(w') = R'(w')$ and $((\mathcal{W}', R'_1, \mathcal{V}'), w') \models \varphi$. All the assumptions of Lemma 9.8 apply, and therefore there is $R_1 \subseteq R$ such that $R_1(w) = R(w)$ and $((\mathcal{W}, R_1, \mathcal{V}), w) \equiv_{m, k}^P ((\mathcal{W}', R'_1, \mathcal{V}'), w')$. As φ belongs to $\text{GML}[m, k, P]$, we derive $((\mathcal{W}, R_1, \mathcal{V}), w) \models \varphi$. From the semantics of \blacklozenge , we conclude that $(\mathcal{K}, w) \models \blacklozenge\varphi$. \square

Lemma 9.10. $\text{ML}(\ast) \preceq \text{GML}$.

Proof (sketch). We already saw similar proofs (see e.g. Theorem 6.22 and Theorem 7.16). Let φ be a formula in $\text{ML}(\ast)$. As $\Diamond\psi \equiv \Diamond_{\geq 1}\psi$, we can replace every occurrence of the modality \Diamond appearing in φ with the modality $\Diamond_{\geq 1}$. Moreover, by Lemma 9.4, we can replace every subformula of the form $\psi * \chi$ with the formula $\blacklozenge(\psi \parallel \chi)$. In this way, we obtain a formula φ' that is equivalent to φ and where all the modalities are of the form $\Diamond_{\geq 1}$, \parallel and \blacklozenge . If φ' has no occurrence of \parallel or \blacklozenge , we are done. Otherwise, let ψ be a subformula of φ' of the form $\blacklozenge(\varphi_1 \parallel \varphi_2)$ where φ_1 and φ_2 are in GML.

- Since $\text{ML}(\parallel) \approx \text{GML}$ (Corollary 7.18), there is a formula ψ' in GML such that $\psi' \equiv \varphi_1 \parallel \varphi_2$.
- By Lemma 9.9 there is a formula ψ'' in GML such that $\psi'' \equiv \blacklozenge\psi'$.

One can show that $\varphi' \equiv \varphi'[\psi \leftarrow \psi'']$, where $\varphi'[\psi \leftarrow \psi'']$ is obtained from φ' by replacing every occurrence of ψ by ψ'' . Note that the number of occurrences of \blacklozenge and \parallel in $\varphi'[\psi \leftarrow \psi'']$ is strictly less than the number of occurrences of \blacklozenge and \parallel in φ' . By repeating such a type of replacement,

EF-Game played on the state $((\mathcal{K}_1=(\mathcal{W}_1, R_1, \mathcal{V}_1), w_1), (\mathcal{K}_2=(\mathcal{W}_2, R_2, \mathcal{V}_2), w_2), (m, s, P))$

if there is $p \in P$ such that $((\mathcal{K}_1, w_1) \models p \text{ iff } (\mathcal{K}_2, w_2) \models p)$ does not hold

then the spoiler wins,

else the spoiler chooses $i \in \{1, 2\}$ and plays on (\mathcal{K}_i, w_i) .

The duplicator replies on (\mathcal{K}_j, w_j) where $j \in \{1, 2\} \setminus \{i\}$.

The spoiler **must** choose one of the following moves (otherwise the duplicator wins).

\diamond move: if $m \geq 1$ and $R_i(w_i) \neq \emptyset$, the spoiler **can** choose to play a \diamond move. If so,

1. The spoiler selects a world $w'_1 \in R_i(w_i)$.
2. The duplicator **must** select a node $w'_j \in R_j(w_j)$ (otherwise the spoiler wins).
3. The game continues on $((\mathcal{W}_1, R_1, \mathcal{V}_1), w'_1), ((\mathcal{W}_2, R_2, \mathcal{V}_2), w'_2), (m-1, s, P)$.

$*$ move: if $s \geq 1$, then the spoiler **can** choose to play a $*$ move.

1. The spoiler selects two Kripke-style finite forests \mathcal{K}_i^1 and \mathcal{K}_i^2 such that $\mathcal{K}_i = \mathcal{K}_i^1 + \mathcal{K}_i^2$,
2. The duplicator **must** reply with two finite forests \mathcal{K}_j^1 and \mathcal{K}_j^2 such that $\mathcal{K}_j = \mathcal{K}_j^1 + \mathcal{K}_j^2$,
3. The spoiler chooses $k \in [1, 2]$,
4. The game continues on $((\mathcal{K}_1^k, w_1), (\mathcal{K}_2^k, w_2), (m, s-1, P))$.

Figure 9.4: Ehrenfeucht-Fraïssé games for $\text{ML}(\ast)$

eventually we obtain a formula φ'' in GML such that $\varphi' \equiv \varphi''$. Indeed, all the occurrences of \blacklozenge and $\|$ only appear as instances of the pattern $\blacklozenge(\psi \| \chi)$. Hence, we get a formula in GML logically equivalent to φ . \square

9.2.2 Showing $\text{ML}(\ast) \prec \text{GML}$ via EF games for $\text{ML}(\ast)$

We now tackle the problem of showing that $\text{ML}(\ast)$ is strictly less expressive than GML, we rely on a notion of Ehrenfeucht-Fraïssé games (EF games, in short) [102] for $\text{ML}(\ast)$, exactly as we did in Chapter 4 in order to study the expressive power of ALT. We refer the reader to Section 4.2.2 for an introduction on these types of games. We write $\text{ML}(\ast)[m, s, P]$ for the set of formulae φ of $\text{ML}(\ast)$ having $\text{md}(\varphi) \leq m$, at most s nested $*$, and atomic propositions from $P \subseteq_{\text{fin}} \text{AP}$. Exactly as $\text{GML}[m, k, P]$, it is easy to see that $\text{ML}(\ast)[m, s, P]$ is finite up to logical equivalence.

As usual, the EF games for $\text{ML}(\ast)$ are played between the spoiler and the duplicator. A game state is a triple made of two pointed forests (\mathcal{K}, w) and (\mathcal{K}', w') and a rank (m, s, P) , where $m, s \in \mathbb{N}$ and $P \subseteq_{\text{fin}} \text{AP}$. The goal of the spoiler is to show that the two models are different. The goal of the duplicator is to counter the spoiler and to show that the two models are similar. Two models are different whenever there is $\varphi \in \text{ML}(\ast)[m, s, P]$ that is satisfied by only one of the two models. The EF games for $\text{ML}(\ast)$ are formally defined in Figure 9.4.

Using the standard definitions [102], the duplicator has a *winning strategy* for the game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$ if she can play in a way that guarantees her to win regardless how the spoiler plays. When this is the case, we write $(\mathcal{K}, w) \approx_{m,s}^P (\mathcal{K}', w')$. Similarly, the spoiler has a *winning strategy*, written $(\mathcal{K}, w) \not\approx_{m,s}^P (\mathcal{K}', w')$, if he can play in a way that guarantees him to win, regardless how the duplicator plays. Lemma 9.11 guarantees that the games are well-defined.

Lemma 9.11. $(\mathcal{K}, w) \not\approx_{m,s}^P (\mathcal{K}', w')$ iff there is φ in $\text{ML}(\ast)[m, s, P]$ s.t. $(\mathcal{K}, w) \models \varphi$ and $(\mathcal{K}', w') \not\models \varphi$.

Lemma 9.11 is proven similarly to Theorem 4.15 (i.e. the analogous result for the EF-games of ALT). For instance the left-to-right direction, i.e. the *completeness of the game*, is by induction

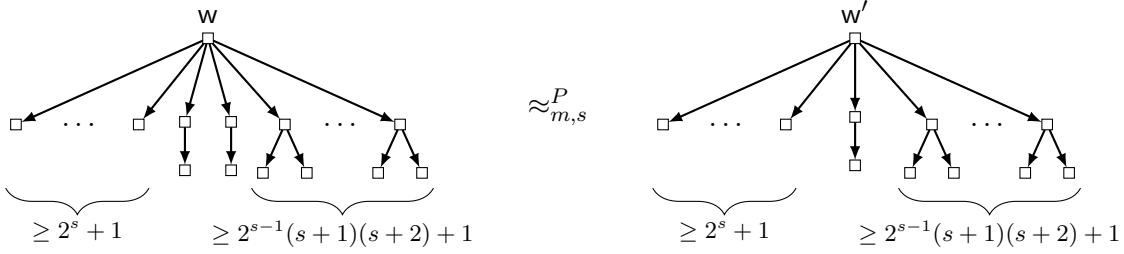


Figure 9.5: Pointed forests used to prove Lemma 9.12. Only the leftmost one satisfies $\Diamond_{=2}\Diamond_{=1}\top$.

on the rank (m, s, P) . The proof of Lemma 9.11 is given in Appendix G. Thanks to the EF games, we are able to find a GML formula φ that is not expressible in $\text{ML}(\ast)$. As we know, $\text{ML}(\mathbb{I})$ and $\text{ML}(\mathbb{I})$ have the same expressive power when it comes to formulae of modal depth at most one (Lemma 9.2). Since $\text{ML}(\mathbb{I}) \approx \text{GML}$ (Corollary 7.18), this implies that any formula discriminating the two logics must have modal degree at least 2. Happily, the formula $\varphi = \Diamond_{=2}\Diamond_{=1}\top$ does the job and cannot be expressed in $\text{ML}(\ast)$. For the proof, we show that for every rank (m, s, P) , there are two structures (\mathcal{K}, w) and (\mathcal{K}', w') such that $(\mathcal{K}, w) \approx_{m,s}^P (\mathcal{K}', w')$, $(\mathcal{K}, w) \models \varphi$ and $(\mathcal{K}', w') \not\models \varphi$. The inexpressibility of φ then stems from Lemma 9.11. The two structures are represented in Figure 9.5 ((\mathcal{K}, w) on the left).

In the following, we say that a world has *type i* if it has i children. As one can see in Figure 9.5, children of the current worlds w and w' are of three types: 0, 1 or 2. When the spoiler performs a spatial move in the game, a world of type i can take, in the submodels, a type between 0 and i . That is, the number of children of a world weakly monotonically decreases when taking submodels. This monotonicity, together with the finiteness of the game, lead to bounds on the number of children of each type, over which the duplicator is guaranteed to win. For instance, the bound for worlds of type 2 is given by the value $2^s(s+1)(s+2)$, where s is the number of spatial moves in the game. In the two presented pointed forests, one child of type 0 and one of type 2 are added with respect to these bounds, so that the duplicator can make up for the different numbers of children of type 1.

Lemma 9.12. $\text{ML}(\ast)$ cannot characterise the class of models satisfying $\Diamond_{=2}\Diamond_{=1}\top$ (in GML).

Proof. As already said, the non-expressivity of $\Diamond_{=2}\Diamond_{=1}\top$ is shown by proving that for every rank (m, s, P) there are two structures (\mathcal{K}, w) and (\mathcal{K}', w') such that

1. $(\mathcal{K}, w) \approx_{m,s}^P (\mathcal{K}', w')$,
2. $(\mathcal{K}, w) \models \Diamond_{=2}\Diamond_{=1}\top$ whereas $(\mathcal{K}', w') \not\models \Diamond_{=2}\Diamond_{=1}\top$.

We divide the proof into two parts, named below (A) and (B). We start with some preliminary definitions. Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ be a Kripke-style finite forest and $w \in \mathcal{W}$. We denote with $R(w)_{=n}$ the set of worlds in $R(w)$ having exactly n children, i.e. $\{w_1 \in R(w) \mid \text{card}(R(w_1)) = n\}$. During the proof, we only use pointed forests (\mathcal{K}, w) satisfying the following properties:

- (I) $\mathcal{V}(p) = \emptyset$ for every $p \in \text{AP}$;
- (II) $R(w)_{=0}$, $R(w)_{=1}$ and $R(w)_{=2}$ form a partition of $R(w)$;
- (III) $R^2(w) = \emptyset$, i.e. the set of worlds reachable from w in at least two steps is empty.

Figure 9.6 schematically represents the type of models satisfying the properties (I), (II) and (III) (notice that worlds do not satisfy atomic propositions). Let us consider two Kripke-style finite

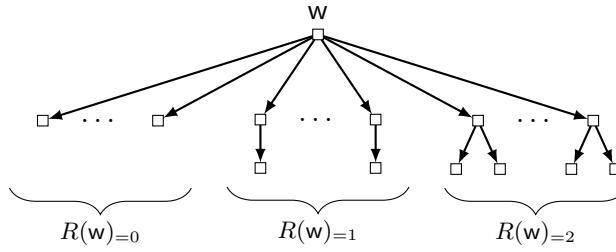


Figure 9.6: Schema of a pointed forest satisfying (I)–(III).

forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V}_2)$ such that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$. We pinpoint three important properties of the models we are considering.

S1: Every world in $R(w)=0$ is either in $R_1(w)=0$ or $R_2(w)=0$,

S2: Every world $w_1 \in R(w)=1$ is in $R_1(w)=0$, $R_2(w)=0$, $R_1(w)=1$ or in $R_2(w)=1$. Indeed, suppose $(w, w_1) \in R_i$ (for some $i \in \{1, 2\}$). If w_1 is in the domain of the same relation R_i then $w_1 \in R_i(w)=1$. Otherwise (w_1 is in the domain of R_{3-i}) then $w_1 \in R_i(w)=0$,

S3: Every world in $R(w)=2$ is in $R_1(w)=0$, $R_2(w)=0$, $R_1(w)=1$, $R_2(w)=1$, $R_1(w)=2$ or $R_2(w)=2$. The justification is similar to the one given above for $R(w)=1$.

We first prove the following property:

$$R_1(w)|_{\mathsf{T} \blacktriangleright \mathsf{T}'} \stackrel{\text{def}}{=} R(w)|_{\mathsf{T}} \cap R_1(w)|_{\mathsf{T}'}$$

Following the proof idea presented above, a world $w_1 \in R_1(w)|_{\mathsf{T} \blacktriangleright \mathsf{T}'}$ is a child of w such that (\mathcal{K}, w_1) is in the class T and “jumps” to the class T' when updating the accessibility relation from R to R_1 . We are interested in the following key property:

- (A) Given a rank (m, s, P) and two pointed forests $(\mathcal{K} = (\mathcal{W}, R, \mathcal{V}), w)$ and $(\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}'), w')$ satisfying (I), (II), (III) and

 - $\min(\text{card}(R(w)=0), 2^s) = \min(\text{card}(R'(w')=0), 2^s)$,
 - $\min(\text{card}(R(w)=1), 2^{s(s+1)}) = \min(\text{card}(R'(w')=1), 2^{s(s+1)})$,
 - $\min(\text{card}(R(w)=2), 2^{s-1}(s+1)(s+2)) = \min(\text{card}(R'(w')=2), 2^{s-1}(s+1)(s+2))$.

Then, $(\mathcal{K}, w) \approx_{m,s}^P (\mathcal{K}', w')$.

First, as worlds in our models do not satisfy any propositional symbol, the spoiler cannot win because of distinct propositional valuations. The proof is by cases on m and on the moves done by the spoiler, and by induction on s . First, suppose $m = 0$. Then it is easy to see that the duplicator has a winning strategy. Indeed, as $m = 0$, the spoiler cannot play the modal move and therefore cannot change the current worlds w and w' . Then, after s spatial moves the game will be in the state (\mathcal{K}_1, w) and (\mathcal{K}'_1, w') w.r.t. the rank $(0, 0, P)$. By (I), the duplicator wins.

Suppose now $m \geq 1$ and the spoiler decides to perform a modal move. Notice that, in particular, this case also takes care of the case where $s = 0$ and the spoiler is forced to play a modal move. Moreover, suppose that the spoiler chooses (\mathcal{K}, w) (the case where it picks (\mathcal{K}', w') is handled similarly by the duplicator). We have to distinguish the following situations.

- Suppose that the spoiler chooses a world $w_1 \in R(w)=0$. Then $\text{card}(R(w)=0) \geq 1$ and from $\min(\text{card}(R(w)=0), 2^s) = \min(\text{card}(R'(w')=0), 2^s)$, it follows that $\text{card}(R'(w')=0) \geq 1$.

It is then sufficient for the duplicator to choose $w_1 \in R'(w')_{=0}$ to guarantee him a victory, as the subtrees rooted in w_1 and w'_1 are isomorphic (by (I) and (III)).

- Suppose that the spoiler chooses a world $w_1 \in R(w)_{=1}$. Then $\text{card}(R(w)_{=1}) \geq 1$ and from $\min(\text{card}(R(w)_{=1}), 2^s(s+1)) = \min(\text{card}(R'(w')_{=1}), 2^s(s+1))$, we have $\text{card}(R'(w')_{=1}) \geq 1$. Then again, it is sufficient for the duplicator to choose $w_1 \in R'(w')_{=1}$ to guarantee him a victory, as the subtrees rooted in w_1 and w'_1 are isomorphic.
- Suppose that the spoiler chooses a world $w_1 \in R(w)_{=2}$. Then $\text{card}(R(w)_{=2}) \geq 1$ and from $\min(\text{card}(R(w)_{=2}), 2^{s-1}(s+1)(s+2)) = \min(\text{card}(R'(w')_{=2}), 2^{s-1}(s+1)(s+2))$, it follows that $\text{card}(R'(w')_{=2}) \geq 1$ (notice here that $2^{s-1}(s+1)(s+2) = 1$ for $s = 0$). Then again, it is sufficient for the duplicator to choose $w_1 \in R'(w')_{=2}$ to guarantee him a victory, as the subtrees rooted in w_1 and w'_1 are isomorphic.

As stated before, the case where the spoiler decides to perform a modal move also captures the base case of the induction on s . Then, it remains to show the case where $s \geq 1$ and the spoiler decides to do a spatial move. Again suppose that the spoiler chooses (\mathcal{K}, w) (the case where it picks (\mathcal{K}', w') is analogous). It then picks two structures $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$. Notice that these two structures are such that both (\mathcal{K}_1, w) and (\mathcal{K}_2, w) satisfy (I), (II) and (III), as these three properties are all preserved when considering subforests. The duplicator has now to pick two structures $\mathcal{K}'_1 = (\mathcal{W}', R'_1, \mathcal{V}')$ and $\mathcal{K}'_2 = (\mathcal{W}', R'_2, \mathcal{V}')$ such that $\mathcal{K}'_1 + \mathcal{K}'_2 = \mathcal{K}'$ and that guarantees him a victory. It does so by constructing R'_1 and R'_2 as follows (from the empty set):

split of $R'(w)_{=0}$. We introduce the sets

$$\begin{aligned} R_1(w)|_{0\blacktriangleright 0} &\stackrel{\text{def}}{=} R_1(w)_{=0} \cap R(w)_{=0} \\ R_2(w)|_{0\blacktriangleright 0} &\stackrel{\text{def}}{=} R_2(w)_{=0} \cap R(w)_{=0}. \end{aligned}$$

It is easy to see that these sets are pairwise disjoint. From (S1) it follows that

$$R(w)_{=0} = (R_1(w)_{=0} \cap R(w)_{=0}) \cup (R_2(w)_{=0} \cap R(w)_{=0}).$$

The duplicator starts by partitioning $R'(w)_{=0}$ into two sets Z_1 and Z_2 (“Z” is short for “zero”) according to the cardinalities of the two components of $R(w)_{=0}$ highlighted above, namely the two sets $R_1(w)_{=0} \cap R(w)_{=0}$ and $R_2(w)_{=0} \cap R(w)_{=0}$.

case: $\text{card}(R_1(w)|_{0\blacktriangleright 0}) < 2^{s-1}$ **and** $\text{card}(R_2(w)|_{0\blacktriangleright 0}) < 2^{s-1}$. We have $\text{card}(R(w)_{=0}) < 2^s$ and, by hypothesis, $\text{card}(R'(w')_{=0}) = \text{card}(R(w)_{=0})$. Then the split of $R'(w)_{=0}$ into Z_1 and Z_2 is made so that $\text{card}(Z_1) = \text{card}(R_1(w)|_{0\blacktriangleright 0})$ and $\text{card}(Z_2) = \text{card}(R_2(w)|_{0\blacktriangleright 0})$.

case: there is $i \in \{1, 2\}$ s.t. $\text{card}(R_i(w)|_{0\blacktriangleright 0}) < 2^{s-1}$ and $\text{card}(R_{3-i}(w)|_{0\blacktriangleright 0}) \geq 2^{s-1}$.

Note that $j = 3 - i$ is the index different from i . The split of $R'(w)_{=0}$ into Z_i and Z_j is made so that $\text{card}(Z_i) = \text{card}(R_i(w)|_{0\blacktriangleright 0})$. By hypothesis on the cardinality of $R'(w)_{=0}$, we have $\text{card}(Z_j) \geq 2^{s-1}$ (as otherwise $\min(\text{card}(R(w)_{=0}), 2^s) \neq \min(\text{card}(R'(w')_{=0}), 2^s)$, a contradiction).

case: $\text{card}(R_1(w)|_{0\blacktriangleright 0}) \geq 2^{s-1}$ **and** $\text{card}(R_2(w)|_{0\blacktriangleright 0}) \geq 2^{s-1}$. Then the split of $R'(w)_{=0}$ into Z_1 and Z_2 is made so that $\text{card}(Z_1) = 2^{s-1}$. By hypothesis on the cardinality of $R'(w)_{=0}$ it holds that $\text{card}(Z_j) \geq 2^{s-1}$.

For each $w'_1 \in Z_1$, the duplicator adds (w', w'_1) to R'_1 . For each $w'_2 \in Z_2$, it adds (w', w'_2) to R'_2 . Notice that by construction the two sets introduced are always such that

$$\mathbf{Z1: } \min(\text{card}(R_1(w)|_{0\blacktriangleright 0}), 2^{s-1}) = \min(\text{card}(Z_1), 2^{s-1}),$$

$$\mathbf{Z2: } \min(\text{card}(R_2(w)|_{0\blacktriangleright 0}), 2^{s-1}) = \min(\text{card}(Z_2), 2^{s-1}).$$

split of $R'(\mathbf{w})_{=1}$. We introduce the following sets:

$$\begin{aligned} R_1(\mathbf{w})|_{1\blacktriangleright 0} &\stackrel{\text{def}}{=} R_1(\mathbf{w})_{=0} \cap R(\mathbf{w})_{=1}, & R_2(\mathbf{w})|_{1\blacktriangleright 0} &\stackrel{\text{def}}{=} R_2(\mathbf{w})_{=0} \cap R(\mathbf{w})_{=1}, \\ R_1(\mathbf{w})|_{1\blacktriangleright 1} &\stackrel{\text{def}}{=} R_1(\mathbf{w})_{=1} \cap R(\mathbf{w})_{=1}, & R_2(\mathbf{w})|_{1\blacktriangleright 1} &\stackrel{\text{def}}{=} R_2(\mathbf{w})_{=1} \cap R(\mathbf{w})_{=1}. \end{aligned}$$

It is easy to see that these sets are pairwise disjoint. From (S2) it follows that

$$R(\mathbf{w})_{=1} = R_1(\mathbf{w})|_{1\blacktriangleright 0} \cup R_2(\mathbf{w})|_{1\blacktriangleright 0} \cup R_1(\mathbf{w})|_{1\blacktriangleright 1} \cup R_2(\mathbf{w})|_{1\blacktriangleright 1}.$$

The duplicator starts by partitioning $R'(\mathbf{w})_{=1}$ into four sets Z'_1 , Z'_2 , O_1 and O_2 according to the cardinalities of the four sets above (“Z” for “zero”, “O” for “one”). In order to shorten the presentation, instead of concretely make explicit all the cases as we did in the previous point of the construction, we treat them “schematically”. Let $\mathcal{S} = \{R_1(\mathbf{w})|_{1\blacktriangleright 0}, R_2(\mathbf{w})|_{1\blacktriangleright 0}, R_1(\mathbf{w})|_{1\blacktriangleright 1}, R_2(\mathbf{w})|_{1\blacktriangleright 1}\}$ and let \mathfrak{f} be the bijection

$$\mathfrak{f}(R_1(\mathbf{w})|_{1\blacktriangleright 0}) \stackrel{\text{def}}{=} Z'_1, \quad \mathfrak{f}(R_2(\mathbf{w})|_{1\blacktriangleright 0}) \stackrel{\text{def}}{=} Z'_2, \quad \mathfrak{f}(R_1(\mathbf{w})|_{1\blacktriangleright 1}) \stackrel{\text{def}}{=} O_1, \quad \mathfrak{f}(R_2(\mathbf{w})|_{1\blacktriangleright 1}) \stackrel{\text{def}}{=} O_2.$$

Moreover, we define (\mathcal{B} stands for “bound”)

$$\begin{aligned} \mathcal{B}(R_1(\mathbf{w})|_{1\blacktriangleright 0}) &\stackrel{\text{def}}{=} \mathcal{B}(R_2(\mathbf{w})|_{1\blacktriangleright 0}) \stackrel{\text{def}}{=} 2^{s-1}, \\ \mathcal{B}(R_1(\mathbf{w})|_{1\blacktriangleright 1}) &\stackrel{\text{def}}{=} \mathcal{B}(R_2(\mathbf{w})|_{1\blacktriangleright 1}) \stackrel{\text{def}}{=} 2^{s-1}s. \end{aligned}$$

So, these definitions (actually notations) are helpful at the metalevel. Besides, notice that, from $s \geq 1$, it holds that 2^{s-1} and $2^{s-1}s$ are both at least 1.

case: $\text{card}(S) < \mathcal{B}(S)$, for every set $S \in \mathcal{S}$. Then, since it holds that

$$\text{card}(R(\mathbf{w})_{=1}) = \text{card}(R_1(\mathbf{w})|_{1\blacktriangleright 0}) + \text{card}(R_2(\mathbf{w})|_{1\blacktriangleright 0}) + \text{card}(R_1(\mathbf{w})|_{1\blacktriangleright 1}) + \text{card}(R_2(\mathbf{w})|_{1\blacktriangleright 1})$$

we have $\text{card}(R(\mathbf{w})_{=1}) < 2^{s-1} + 2^{s-1} + 2^{s-1}s + 2^{s-1}s = 2^s(s+1)$ and therefore by hypothesis we conclude that $\text{card}(R(\mathbf{w})_{=1}) = \text{card}(R'(\mathbf{w}')_{=1})$. Then, the split of $R'(\mathbf{w}')_{=1}$ into Z'_1 , Z'_2 , O_1 and O_2 is made so that for every $S \in \mathcal{S}$, $\text{card}(\mathfrak{f}(S)) = \text{card}(S)$.

case: there is $\widehat{S} \in \mathcal{S}$ such that $\text{card}(\widehat{S}) \geq \mathcal{B}(\widehat{S})$. Then, the split of $R'(\mathbf{w}')_{=1}$ into Z'_1 , Z'_2 , O_1 and O_2 is made so that for every $S \in \mathcal{S} \setminus \{\widehat{S}\}$, $\text{card}(\mathfrak{f}(S)) = \min(\text{card}(S), \mathcal{B}(S))$. From the hypothesis

$$\min(\text{card}(R(\mathbf{w})_{=1}), 2^s(s+1)) = \min(\text{card}(R'(\mathbf{w}')_{=1}), 2^s(s+1)),$$

we conclude that this construction can be effectively made, and $\text{card}(\mathfrak{f}(\widehat{S})) \geq \mathcal{B}(\widehat{S})$.

For each $\mathbf{w}'_1 \in Z'_1$, the duplicator adds $(\mathbf{w}', \mathbf{w}'_1)$ to R'_1 and the only element of $R'|_{\mathbf{w}'_1}$ to R'_2 . For each $\mathbf{w}'_2 \in Z'_2$, it adds $(\mathbf{w}', \mathbf{w}'_2)$ to R'_2 and the only element of $R'|_{\mathbf{w}'_2}$ to R'_1 . For each $\mathbf{w}'_1 \in O_1$, it adds $(\mathbf{w}', \mathbf{w}'_1)$ and the only element of $R'|_{\mathbf{w}'_1}$ to R'_1 . Lastly, for each $\mathbf{w}'_2 \in O_2$, it adds $(\mathbf{w}', \mathbf{w}'_2)$ and the only element of $R'|_{\mathbf{w}'_2}$ to R'_2 . Notice that by construction the four sets introduced are always such that

$$\mathbf{Z11: } \min(\text{card}(R_1(\mathbf{w})|_{1\blacktriangleright 0}), 2^{s-1}) = \min(\text{card}(Z'_1), 2^{s-1}),$$

$$\mathbf{Z21: } \min(\text{card}(R_2(\mathbf{w})|_{1\blacktriangleright 0}), 2^{s-1}) = \min(\text{card}(Z'_2), 2^{s-1}),$$

$$\mathbf{O1: } \min(\text{card}(R_1(\mathbf{w})|_{1\blacktriangleright 1}), 2^{s-1}s) = \min(\text{card}(O_1), 2^{s-1}s),$$

$$\mathbf{O2: } \min(\text{card}(R_2(\mathbf{w})|_{1\blacktriangleright 1}), 2^{s-1}s) = \min(\text{card}(O_2), 2^{s-1}s),$$

or, more schematically, for every $S \in \mathcal{S}$, $\min(\text{card}(S), \mathcal{B}(S)) = \min(\text{card}(\mathfrak{f}(S)), \mathcal{B}(S))$.

split of $R'(\mathbf{w})_{=2}$. Similarly to the previous steps, we introduce the following sets:

$$\begin{aligned} R_1(\mathbf{w})|_{2\blacktriangleright 0} &\stackrel{\text{def}}{=} R_1(\mathbf{w})_{=0} \cap R(\mathbf{w})_{=2}, & R_2(\mathbf{w})|_{2\blacktriangleright 0} &\stackrel{\text{def}}{=} R_2(\mathbf{w})_{=0} \cap R(\mathbf{w})_{=2}, \\ R_1(\mathbf{w})|_{2\blacktriangleright 1} &\stackrel{\text{def}}{=} R_1(\mathbf{w})_{=1} \cap R(\mathbf{w})_{=2}, & R_2(\mathbf{w})|_{2\blacktriangleright 1} &\stackrel{\text{def}}{=} R_2(\mathbf{w})_{=1} \cap R(\mathbf{w})_{=2}, \\ R_1(\mathbf{w})|_{2\blacktriangleright 2} &\stackrel{\text{def}}{=} R_1(\mathbf{w})_{=2} \cap R(\mathbf{w})_{=2}, & R_2(\mathbf{w})|_{2\blacktriangleright 2} &\stackrel{\text{def}}{=} R_2(\mathbf{w})_{=2} \cap R(\mathbf{w})_{=2}. \end{aligned}$$

It is easy to see that these sets are pairwise disjoint. From (S3) it follows that

$$R(w)_{=2} = R_1(w)|_{2\blacktriangleright 0} \cup R_2(w)|_{2\blacktriangleright 0} \cup R_1(w)|_{2\blacktriangleright 1} \cup R_2(w)|_{2\blacktriangleright 1} \cup R_1(w)|_{2\blacktriangleright 2} \cup R_2(w)|_{2\blacktriangleright 2}.$$

The duplicator starts by partitioning $R'(w)_{=2}$ into six sets Z''_1 , Z''_2 , O'_1 , O'_2 , T_1 and T_2 according to the cardinalities of the six sets above (“T” for “two”). Again, to shorten the presentation we introduce the set

$$\mathcal{S} = \{R_1(w)|_{2\blacktriangleright 0}, R_2(w)|_{2\blacktriangleright 0}, R_1(w)|_{2\blacktriangleright 1}, R_2(w)|_{2\blacktriangleright 1}, R_1(w)|_{2\blacktriangleright 2}, R_2(w)|_{2\blacktriangleright 2}\},$$

and the bijection f such that

$$\begin{aligned} f(R_1(w)|_{2\blacktriangleright 0}) &\stackrel{\text{def}}{=} Z''_1, & f(R_2(w)|_{2\blacktriangleright 0}) &\stackrel{\text{def}}{=} Z''_2, & f(R_1(w)|_{2\blacktriangleright 1}) &\stackrel{\text{def}}{=} O'_1, \\ f(R_2(w)|_{2\blacktriangleright 1}) &\stackrel{\text{def}}{=} O'_2, & f(R_1(w)|_{2\blacktriangleright 2}) &\stackrel{\text{def}}{=} T_1, & f(R_2(w)|_{2\blacktriangleright 2}) &\stackrel{\text{def}}{=} T_2. \end{aligned}$$

Moreover, we define

$$\begin{aligned} \mathcal{B}(R_1(w)|_{2\blacktriangleright 0}) &\stackrel{\text{def}}{=} \mathcal{B}(R_2(w)|_{2\blacktriangleright 0}) \stackrel{\text{def}}{=} 2^{s-1}, \\ \mathcal{B}(R_1(w)|_{2\blacktriangleright 1}) &\stackrel{\text{def}}{=} \mathcal{B}(R_2(w)|_{2\blacktriangleright 1}) \stackrel{\text{def}}{=} 2^{s-1}s, \\ \mathcal{B}(R_1(w)|_{2\blacktriangleright 2}) &\stackrel{\text{def}}{=} \mathcal{B}(R_2(w)|_{2\blacktriangleright 2}) \stackrel{\text{def}}{=} 2^{s-2}s(s+1). \end{aligned}$$

Notice that, from $s \geq 1$, it holds that 2^{s-1} , $2^{s-1}s$ and $2^{s-2}s(s+1)$ are both at least 1.

case: for every $S \in \mathcal{S}$, $\text{card}(S) < \mathcal{B}(S)$. Then, since $\text{card}(R(w)_{=2})$ is

$$\begin{aligned} &\text{card}(R_1(w)|_{2\blacktriangleright 0}) + \text{card}(R_2(w)|_{2\blacktriangleright 0}) + \text{card}(R_1(w)|_{2\blacktriangleright 1}) \\ &+ \text{card}(R_2(w)|_{2\blacktriangleright 1}) + \text{card}(R_1(w)|_{2\blacktriangleright 2}) + \text{card}(R_2(w)|_{2\blacktriangleright 2}) \end{aligned}$$

it holds that

$$\text{card}(R(w)_{=2}) < 2 \times 2^{s-1} + 2 \times 2^{s-1}s + 2 \times 2^{s-2}s(s+1) = 2^{s-1}(s+1)(s+2),$$

and therefore by hypothesis we conclude that $\text{card}(R(w)_{=2}) = \text{card}(R'(w')_{=2})$. Then, the split of $R'(w')_{=2}$ into Z''_1 , Z''_2 , O'_1 , O'_2 , T_1 and T_2 is made so that for all $S \in \mathcal{S}$, $\text{card}(f(S)) = \text{card}(S)$.

case: there is $\hat{S} \in \mathcal{S}$ such that $\text{card}(\hat{S}) \geq \mathcal{B}(\hat{S})$. The split of $R'(w')_{=2}$ into Z''_1 , Z''_2 , O'_1 , O'_2 , T_1 and T_2 is made so that for all $S \in \mathcal{S} \setminus \hat{S}$, $\text{card}(f(S)) = \min(\text{card}(S), \mathcal{B}(S))$. From the hypothesis

$$\min(\text{card}(R(w)_{=2}), 2^{s-1}(s+1)(s+2)) = \min(\text{card}(R'(w')_{=2}), 2^{s-1}(s+1)(s+2)),$$

we conclude that this construction can be effectively made, and $\text{card}(f(\hat{S})) \geq \mathcal{B}(\hat{S})$.

Then, the duplicator updates R'_1 and R'_2 as follows:

- For each $w'_1 \in Z''_1$, the duplicator adds (w', w'_1) to R'_1 and the elements of $R'|_{w'_1}$ to R'_2 .
- For each $w'_2 \in Z''_2$, it adds (w', w'_2) to R'_2 and the elements of $R'|_{w'_2}$ to R'_1 .
- For each $w'_1 \in O'_1$, it adds (w', w'_1) and one of the two elements of $R'|_{w'_1}$ to R'_1 . The other element of $R'|_{w'_1}$ is assigned to R'_2 .
- For each $w'_2 \in O'_2$, it adds (w', w'_2) and one of the two elements of $R'|_{w'_2}$ to R'_2 . The other element of $R'|_{w'_2}$ is assigned to R'_1 .
- For each $w'_1 \in T_1$, it adds (w', w'_1) to R'_1 and the two elements of $R'|_{w'_1}$ to R'_1 .
- For each $w'_2 \in T_2$, it adds (w', w'_2) to R'_2 and the two elements of $R'|_{w'_2}$ to R'_2 .

Notice that by construction the six sets introduced are always such that

$$\mathbf{Z12: } \min(\text{card}(R_1(w)|_{2\blacktriangleright 0}), 2^{s-1}) = \min(\text{card}(Z''_1), 2^{s-1}),$$

$$\mathbf{Z22: } \min(\text{card}(R_2(w)|_{2\blacktriangleright 0}), 2^{s-1}) = \min(\text{card}(Z''_2), 2^{s-1}),$$

$$\mathbf{O11: } \min(\text{card}(R_1(w)|_{2\blacktriangleright 1}), 2^{s-1}s) = \min(\text{card}(O'_1), 2^{s-1}s),$$

O21: $\min(\text{card}(R_2(w)|_{2\blacktriangleright 1}), 2^{s-1}s) = \min(\text{card}(O'_2), 2^{s-1}s)$,

T1: $\min(\text{card}(R_1(w)|_{2\blacktriangleright 2}), 2^{s-2}s(s+1)) = \min(\text{card}(T_1), 2^{s-2}s(s+1))$,

T2: $\min(\text{card}(R_2(w)|_{2\blacktriangleright 2}), 2^{s-2}s(s+1)) = \min(\text{card}(T_2), 2^{s-2}s(s+1))$,

or, more schematically, for every $S \in \mathcal{S}$, $\min(\text{card}(S), \mathcal{B}(S)) = \min(\text{card}(\mathfrak{f}(S)), \mathcal{B}(S))$.

After these steps, since (\mathcal{K}', w') satisfies (II) and (III), every element $(w'_1, w'_2) \in R'$ such that $w'_1 \in R'^*(w)$ has been assigned to either R'_1 or R'_2 . Duplicator then concludes the construction of \mathcal{K}'_1 and \mathcal{K}'_2 by assigning the remaining elements of R' (i.e. the pairs $(w'_1, w'_2) \in R'$ such that $w'_1 \notin R'^*(w)$) to either R'_1 or R'_2 (for example, it can put all these elements in R'_1). The two models \mathcal{K}'_1 and \mathcal{K}'_2 are now defined and they trivially satisfy (I), (II) and (III) (as they are submodels of \mathcal{K}'). Moreover, by construction it is easy to verify that:

- $R'_1(w')=0 = Z_1 \cup Z'_1 \cup Z''_1$,
- $R'_1(w')=1 = O_1 \cup O'_1$,
- $R'_1(w')=2 = T_1$,
- for every $n > 2$, $R'_1(w')=n = \emptyset$,
- $R'_2(w')=0 = Z_2 \cup Z'_2 \cup Z''_2$,
- $R'_2(w')=1 = O_2 \cup O'_2$,
- $R'_2(w')=2 = T_2$,
- for every $n > 2$, $R'_2(w')=n = \emptyset$.

Indeed, we specifically built R'_1 and R'_2 so that these properties, which we later refer to with (\dagger) , hold. Now, we end the proof of (A) by showing that for all $i \in \{1, 2\}$,

(zero) $\min(\text{card}(R_i(w)=0), 2^{s-1}) = \min(\text{card}(R'_i(w')=0), 2^{s-1})$,

(one) $\min(\text{card}(R_i(w)=1), 2^{s-1}s) = \min(\text{card}(R'_i(w')=1), 2^{s-1}s)$,

(two) $\min(\text{card}(R_i(w)=2), 2^{s-2}s(s+1)) = \min(\text{card}(R'_i(w')=2), 2^{s-2}s(s+1))$.

Indeed, once these three properties are shown we can apply the induction hypothesis to conclude that $(\mathcal{K}_1, w) \approx_{m,s-1}^P (\mathcal{K}'_1, w')$ and $(\mathcal{K}_2, w) \approx_{m,s-1}^P (\mathcal{K}'_2, w')$ and therefore, the play described with the construction above leads to a winning strategy for the duplicator on the game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$, i.e. $(\mathcal{K}, w) \approx_{m,s}^P (\mathcal{K}', w')$. The proof of these three properties is quite easy (each case is similar to the others). Let $i \in \{1, 2\}$. By using the definitions given during the construction of R'_1 and R'_2 it holds that

- $R_i(w)=0 = R_i(w)|_{0\blacktriangleright 0} \cup R_i(w)|_{1\blacktriangleright 0} \cup R_i(w)|_{2\blacktriangleright 0}$, and by definition for all $j, k \in [0, 2]$ such that $j \neq k$ it holds that $R_i(w)|_{j\blacktriangleright 0} \cap R_i(w)|_{k\blacktriangleright 0} = \emptyset$,
- $R_i(w)=1 = R_i(w)|_{1\blacktriangleright 1} \cup R_i(w)|_{2\blacktriangleright 1}$, and by definition $R_i(w)|_{1\blacktriangleright 1} \cap R_i(w)|_{2\blacktriangleright 1} = \emptyset$,
- $R_i(w)|_{2\blacktriangleright 2} = R_i(w)|_{2\blacktriangleright 2}$.

In what follows, we refer to the three properties above by (\ddagger) .

proof of (zero). By (\ddagger) , $\text{card}(R_i(w)=0) = \text{card}(R_i(w)|_{0\blacktriangleright 0}) + \text{card}(R_i(w)|_{1\blacktriangleright 0}) + \text{card}(R_i(w)|_{2\blacktriangleright 0})$.

We divide the proof into two cases. For the first case, suppose $\text{card}(R_i(w)|_{0\blacktriangleright 0}) < 2^{s-1}$, $\text{card}(R_i(w)|_{1\blacktriangleright 0}) < 2^{s-1}$ and $\text{card}(R_i(w)|_{2\blacktriangleright 0}) < 2^{s-1}$. Then,

1. $\text{card}(Z_i) = \text{card}(R_i(w)|_{0\blacktriangleright 0})$ (by (Z1) or (Z2), depending on whether $i = 1$ or $i = 2$)
2. $\text{card}(Z'_i) = \text{card}(R_i(w)|_{1\blacktriangleright 0})$ (by (Z11)/(Z21))
3. $\text{card}(Z''_i) = \text{card}(R_i(w)|_{2\blacktriangleright 0})$ (by (Z12)/(Z22))
4. $\text{card}(R'_i(w')=0) = \text{card}(R_i(w)|_{0\blacktriangleright 0}) + \text{card}(R_i(w)|_{1\blacktriangleright 0}) + \text{card}(R_i(w)|_{2\blacktriangleright 0})$ (from (1), (2) and (3), by (\dagger))
5. $\text{card}(R'_i(w')=0) = \text{card}(R_i(w)=0)$ (from 4, by (\ddagger)).

Otherwise, suppose that there is a set among $R_i(w)|_{0\blacktriangleright 0}$, $R_i(w)|_{1\blacktriangleright 0}$ and $R_i(w)|_{2\blacktriangleright 0}$ whose cardinality is at least 2^{s-1} . Then from (Z1)/(Z2), (Z11)/(Z21) or (Z12)/(Z22) (depending on whether $i = 1$ or $i = 2$ and on which set has at least 2^{s-1} elements) there is a set among Z_i , Z'_i and Z''_i that has cardinality 2^{s-1} . Then, by (†) and (‡) we have that $R_i(w)=_0$ and $R'_i(w')=_0$ have both more than 2^{s-1} elements.

proof of (one). By (‡), $\text{card}(R_i(w)=_1) = \text{card}(R_i(w)|_{1\blacktriangleright 1}) + \text{card}(R_i(w)|_{2\blacktriangleright 1})$. We divide the proof into two cases. First, suppose $\text{card}(R_i(w)|_{1\blacktriangleright 1}) < 2^{s-1}s$ and $\text{card}(R_i(w)|_{2\blacktriangleright 1}) < 2^{s-1}s$. Then,

1. $\text{card}(O_i) = \text{card}(R_i(w)|_{1\blacktriangleright 1})$ (by (O1) or (O2), depending on whether $i = 1$ or $i = 2$)
2. $\text{card}(O'_i) = \text{card}(R_i(w)|_{2\blacktriangleright 1})$ (by (O11)/(O21))
3. $\text{card}(R'_i(w')=1) = \text{card}(R_i(w)|_{1\blacktriangleright 1}) + \text{card}(R_i(w)|_{2\blacktriangleright 1})$ (from (1) and (2), by (†))
4. $\text{card}(R'_i(w')=1) = \text{card}(R_i(w)=1)$ (from 3, by (‡)).

Otherwise, suppose that there is a set among $R_i(w)|_{1\blacktriangleright 1}$ and $R_i(w)|_{2\blacktriangleright 1}$ whose cardinality is at least $2^{s-1}s$. Then from (O1)/(O2) or (O11)/(O21) (depending on whether $i = 1$ or $i = 2$ and on which set has at least $2^{s-1}s$ elements) there is a set among O_i , O'_i that has cardinality $2^{s-1}s$. Then, by (†) and (‡) we have that $R_i(w)=1$ and $R'_i(w')=1$ have both more than $2^{s-1}s$ elements.

proof of (two). By (‡), it holds that $\text{card}(R_i(w)=2) = \text{card}(R_i(w)|_{2\blacktriangleright 2})$. Again we divide the proof into two cases. First, suppose $\text{card}(R_i(w)|_{2\blacktriangleright 2}) < 2^{s-2}s(s+1)$. Then,

1. $\text{card}(T_i) = \text{card}(R_i(w)|_{2\blacktriangleright 2})$ (by (T1) or (T2), depending on whether $i = 1$ or $i = 2$)
2. $\text{card}(R'_i(w')=2) = \text{card}(R_i(w)|_{2\blacktriangleright 2})$ (from (1), by (†))
3. $\text{card}(R'_i(w')=2) = \text{card}(R_i(w)=2)$ (from 2, by (‡)).

Otherwise, suppose that $\text{card}(R_i(w)|_{2\blacktriangleright 2})$, and hence $\text{card}(R_i(w)=2)$, is at least $2^{s-2}s(s+1)$. Then,

1. $\text{card}(T_i) \geq 2^{s-2}s(s+1)$ (by (T1)/(T2))
2. $\text{card}(R'_i(w')=2) \geq 2^{s-2}s(s+1)$ (from (1), by (†)).

This ends the proof of (A), in which we rely on to show the following crucial property.

(B) Given a rank (m, s, P) and two structures $(\mathcal{K} = (\mathcal{W}, R, \mathcal{V}), w)$ and $(\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}'), w')$ satisfying (I), (II), (III) and

- $\text{card}(R(w)=0) \geq 2^s + 1$ and $\text{card}(R'(w')=0) \geq 2^s + 1$,
- $\text{card}(R(w)=1) = 2$ and $\text{card}(R'(w')=1) = 1$,
- $\text{card}(R(w)=2) \geq 2^{s-1}(s+1)(s+2) + 1$ and $\text{card}(R'(w')=2) \geq 2^{s-1}(s+1)(s+2) + 1$.

Then, $(\mathcal{K}, w) \approx_{m,s}^P (\mathcal{K}', w')$.

Note that showing (B) concludes the proof, as $(\mathcal{K}, w) \models \Diamond_{=2}\Diamond_{=1}\top$ and $(\mathcal{K}', w') \not\models \Diamond_{=2}\Diamond_{=1}\top$. Indeed, *ad absurdum* suppose that there is a formula φ in $\text{ML}(\ast)$ such that $\varphi \equiv \Diamond_{=2}\Diamond_{=1}\top$. Let m be its modal degree, s be its maximal number of imbricated $*$ and P be the set of propositional variables occurring in φ . Let us consider two pointed forests (\mathcal{K}_1, w_1) and (\mathcal{K}_2, w_2) satisfying the conditions in (B) (with respect to the values of m , s and P of φ). From these conditions, $(\mathcal{K}_1, w_1) \models \Diamond_{=2}\Diamond_{=1}\top$ and $(\mathcal{K}_2, w_2) \not\models \Diamond_{=2}\Diamond_{=1}\top$. However, from $(\mathcal{K}, w) \approx_{m,s}^P (\mathcal{K}', w')$ and by Lemma 9.11, we conclude that (\mathcal{K}_1, w_1) and (\mathcal{K}_2, w_2) agree on the satisfaction of φ : a contradiction.

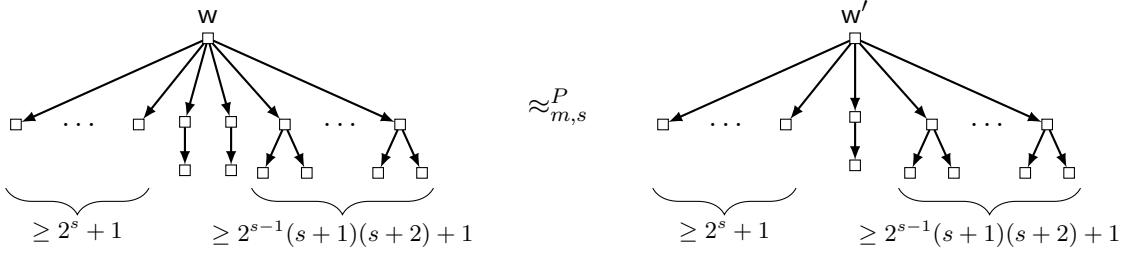


Figure 9.7: The pointed forests described in (B).

The two finite forests of (B) are schematically represented in Figure 9.7 (and previously in Figure 9.5), with (\mathcal{K}, w) on the left and (\mathcal{K}', w') on the right. The proof of (B) is shown by cases on m, s and on the moves done by the spoiler. As in the proof of (A), if $m = 0$ then the duplicator has a winning strategy as after s spatial moves the game will be in the state (\mathcal{K}_1, w) and (\mathcal{K}'_1, w') (notice that w and w' do not change, since $m = 0$) w.r.t. the rank $(0, 0, P)$. From (I), we conclude that the duplicator wins.

Now, suppose $m \geq 1$ and the spoiler decides to perform a modal move. Notice that, in particular, this case also takes care of the case $s = 0$, where the spoiler is forced to play a modal move. Moreover, suppose that the spoiler chooses (\mathcal{K}, w) (the case where it picks (\mathcal{K}', w') is analogous). Then, suppose that the spoiler chooses a world $w_1 \in R(w)_{=n}$ for some $n \in \{0, 1, 2\}$. It is then sufficient for the duplicator to choose $w \in R'(w')_{=n}$ (which is a non-empty set by hypothesis) to guarantee him a victory, as the subtrees rooted in w_1 and w'_1 are isomorphic.

It remains to show the strategy for the duplicator when the spoiler decides to perform a spatial move (and therefore $s \geq 1$). The proof distinguishes several cases depending on the structure chosen by the spoiler.

case: the spoiler picks (\mathcal{K}, w) . In this case, the spoiler chooses the pointed forest in which $\text{card}(R(w)_{=1}) = 2$, and the duplicator has to reply on the pointed forest (\mathcal{K}', w') , in which $\text{card}(R'(w')_{=1}) = 1$. The idea is to make up for this discrepancy by using an element of $R'(w')_{=2}$. Let us see how.

For a moment, consider the model obtained from \mathcal{K}' by removing from R' exactly one pair (w'_1, w'_2) where w'_1 is a world of $R'(w')_{=2}$. Formally, we are interested in a model $\widehat{\mathcal{K}'} = (\mathcal{W}', \widehat{R}', \mathcal{V}')$ such that $\widehat{R}' = R' \setminus \{(w'_1, w'_2)\}$ where $(w'_1, w'_2) \in R'$ and $w'_1 \in R'(w')_{=2}$. If the game was played on (\mathcal{K}, w) and $(\widehat{\mathcal{K}'}, w')$ w.r.t. (m, s, P) then it is clear than the duplicator would have a winning strategy. Indeed, both (\mathcal{K}, w) and $(\widehat{\mathcal{K}'}, w')$ satisfy (I), (II) and (III). Moreover,

- $\text{card}(R(w)_{=0})$ and $\text{card}(\widehat{R}'(w')_{=0})$ are both at least 2^s . Indeed, $\widehat{R}'(w')_{=0} = R'(w')_{=0}$,
- $\text{card}(R(w)_{=1}) = 2$ and $\text{card}(\widehat{R}'(w')_{=1}) = 2$. Indeed, $\widehat{R}'(w')_{=1} = R'(w')_{=1} \cup \{w'_1\}$,
- $\text{card}(R(w)_{=2})$ and $\text{card}(\widehat{R}'(w')_{=2})$ are both at least $2^{s-1}(s+1)(s+2)$. Indeed, by definition, $\widehat{R}'(w')_{=2} = R'(w')_{=2} \setminus \{w'_1\}$.

These properties allow us to apply (A) and conclude that $(\mathcal{K}, w) \approx_{m,s}^P (\widehat{\mathcal{K}'}, w')$. In particular, in this game, if the spoiler picks (\mathcal{K}, w) and chooses $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$, then the duplicator can apply the strategy described in (A) in order to construct two structures $\widehat{\mathcal{K}}'_1 = (\mathcal{W}', \widehat{R}'_1, \mathcal{V}')$ and $\widehat{\mathcal{K}}'_2 = (\mathcal{W}', \widehat{R}'_2, \mathcal{V}')$ such that $\widehat{\mathcal{K}}'_1 + \widehat{\mathcal{K}}'_2 = \widehat{\mathcal{K}'}$ and for every $i \in \{1, 2\}$:

- $\min(\text{card}(R_i(w)_{=0}), 2^{s-1}) = \min(\text{card}(\widehat{R}'_i(w')_{=0}), 2^{s-1})$,
- $\min(\text{card}(R_i(w)_{=1}), 2^{s-1}s) = \min(\text{card}(\widehat{R}'_i(w')_{=1}), 2^{s-1}s)$,
- $\min(\text{card}(R_i(w)_{=2}), 2^{s-2}s(s+1)) = \min(\text{card}(\widehat{R}'_i(w')_{=2}), 2^{s-2}s(s+1))$.

Notice that the three properties above, which we later refer to by $(\dagger\dagger)$ are exactly **(zero)**, **(one)** and **(two)** in the proof of **(A)**.

Let us see how to use these pieces of information to derive a strategy for the duplicator in the original game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$. As the spoiler chooses (\mathcal{K}, w) , it selects \mathcal{K}_1 and \mathcal{K}_2 such that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$. Consider the two structures $\widehat{\mathcal{K}}'_1 = (\mathcal{W}', \widehat{R}'_1, \mathcal{V}')$ and $\widehat{\mathcal{K}}'_2 = (\mathcal{W}', \widehat{R}'_2, \mathcal{V}')$ chosen by the duplicator following the strategy, discussed above, for the game $((\mathcal{K}, w), (\widehat{\mathcal{K}}', w'), (m, s, P))$ in the case when the spoiler chooses (\mathcal{K}, w) and again selects \mathcal{K}_1 and \mathcal{K}_2 . In particular these structures satisfy $(\dagger\dagger)$. Moreover, the two forests $\widehat{\mathcal{K}}'_1$ and $\widehat{\mathcal{K}}'_2$ are such that $\widehat{\mathcal{K}}'_1 + \widehat{\mathcal{K}}'_2 = \widehat{\mathcal{K}}$ and therefore $\widehat{R}'_1 \cup \widehat{R}'_2 = \widehat{R}' = R' \setminus \{(w'_1, w'_2)\}$ where $(w'_1, w'_2) \in R'$ and $w'_1 \in R'(w')_{=2}$. We distinguish two cases.

- If $w'_1 \in \widehat{R}'_1(w')$ then in the original game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$, the duplicator replies to \mathcal{K}_1 and \mathcal{K}_2 with the two forests $\mathcal{K}'_1 = (\mathcal{W}', R'_1, \mathcal{V}')$ and $\mathcal{K}'_2 = (\mathcal{W}', R'_2, \mathcal{V}')$ such that $R'_1 = \widehat{R}'_1$ and $R'_2 = \widehat{R}'_2 \cup \{(w'_1, w'_2)\}$.
- Otherwise $w'_1 \in \widehat{R}'_2(w')$ and in the game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$ the duplicator replies to \mathcal{K}_1 and \mathcal{K}_2 with the two forests $\mathcal{K}'_1 = (\mathcal{W}', R'_1, \mathcal{V}')$ and $\mathcal{K}'_2 = (\mathcal{W}', R'_2, \mathcal{V}')$ such that $R'_1 = \widehat{R}'_1 \cup \{(w'_1, w'_2)\}$ and $R'_2 = \widehat{R}'_2$.

In both cases, as the pair (w', w'_1) is in one relation between R'_1 and R'_2 whereas (w'_1, w'_2) is in the other relation, the world w'_1 effectively behaves like if it was a member of the set $R'(w')_{=1}$ instead of $R'(w')_{=2}$, exactly as in the case of \widehat{R}' . In particular, it is easy to see that for $i \in \{1, 2\}$ and $j \in [0, 3]$, $\text{card}(R'_i(w')_{=j}) = \text{card}(\widehat{R}'_i(w')_{=j})$. Hence, by $(\dagger\dagger)$,

- $\min(\text{card}(R_i(w)_{=0}), 2^{s-1}) = \min(\text{card}(R'_i(w')_{=0}), 2^{s-1})$,
- $\min(\text{card}(R_i(w)_{=1}), 2^{s-1}s) = \min(\text{card}(R'_i(w')_{=1}), 2^{s-1}s)$,
- $\min(\text{card}(R_i(w)_{=2}), 2^{s-2}s(s+1)) = \min(\text{card}(R'_i(w')_{=2}), 2^{s-2}s(s+1))$.

Moreover, \mathcal{K}_1 , \mathcal{K}_2 , \mathcal{K}'_1 and \mathcal{K}'_2 all satisfy **(I)**, **(II)** and **(III)** (as they are submodels of \mathcal{K} or \mathcal{K}'), we can apply **(A)** and derive $(\mathcal{K}_1, w) \approx_{m,s-1}^P (\mathcal{K}'_1, w')$ and $(\mathcal{K}_2, w) \approx_{m,s-1}^P (\mathcal{K}'_2, w')$. Therefore, the play we just described leads to a winning strategy for the duplicator on the game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$, under the hypothesis that the spoiler chooses (\mathcal{K}, w) .

case: the spoiler picks (\mathcal{K}', w') . In this case, the spoiler chooses the pointed forest in which $\text{card}(R'(w')_{=1}) = 1$, and the duplicator has to reply on the pointed forest (\mathcal{K}, w) in which $\text{card}(R(w)_{=1}) = 2$. The proof is very similar to the previous case, but instead of choosing an element of $R'(w')_{=2}$ to make up for the discrepancy between $\text{card}(R(w)_{=1})$ and $\text{card}(R'(w')_{=1})$, the duplicator manipulates the additional element in $R(w)_{=1}$ so that it becomes a member of $R_1(w)_{=0}$ or $R_2(w)_{=0}$. Let us formalise this strategy.

For a moment, consider the model obtained from \mathcal{K} by removing from R exactly one pair (w_1, w_2) where w_1 is a world of $R(w)_{=1}$. Formally, we are interested in a model $\widehat{\mathcal{K}} = (\mathcal{W}, \widehat{R}, \mathcal{V})$ such that $\widehat{R} = R \setminus \{(w_1, w_2)\}$ where $(w_1, w_2) \in R$ and $w_1 \in R(w)_{=1}$. If the game was played on $(\widehat{\mathcal{K}}, w)$ and (\mathcal{K}', w') w.r.t. (m, s, P) then it is clear than the duplicator would have a winning strategy. Indeed, both $(\widehat{\mathcal{K}}, w)$ and (\mathcal{K}', w') satisfy **(I)**, **(II)** and **(III)**. Moreover,

- $\text{card}(\widehat{R}(w)_{=0})$ and $\text{card}(R'(w')_{=0})$ are at least 2^s . Indeed, $\widehat{R}(w)_{=0} = R(w)_{=0} \cup \{w_1\}$,

- $\text{card}(\widehat{R}(w)_{=1}) = 1$ and $\text{card}(R'(w')_{=1}) = 1$. Indeed, $\widehat{R}(w)_{=1} = R(w)_{=1} \setminus \{w_1\}$,
- $\text{card}(\widehat{R}(w)_{=2})$ and $\text{card}(R'(w')_{=2})$ are both at least $2^{s-1}(s+1)(s+2)$. Indeed, by definition, $\widehat{R}(w)_{=2} = R(w)_{=2}$.

These properties allow us to apply (A) and conclude that $(\widehat{\mathcal{K}}, w) \approx_{m,s}^P (\mathcal{K}', w')$. In particular, in this game, if the spoiler picks (\mathcal{K}', w') and chooses $\mathcal{K}'_1 = (\mathcal{W}', R'_1, \mathcal{V}')$ and $\mathcal{K}'_2 = (\mathcal{W}', R'_2, \mathcal{V}')$ such that $\mathcal{K}'_1 + \mathcal{K}'_2 = \mathcal{K}'$, then the duplicator can apply the strategy described in (A). Two structures $\widehat{\mathcal{K}}_1 = (\mathcal{W}, \widehat{R}_1, \mathcal{V})$ and $\widehat{\mathcal{K}}_2 = (\mathcal{W}, \widehat{R}_2, \mathcal{V})$ are constructed such that $\widehat{\mathcal{K}}_1 + \widehat{\mathcal{K}}_2 = \widehat{\mathcal{K}}$ and for every $i \in \{1, 2\}$:

- $\min(\text{card}(\widehat{R}_i(w)_{=0}), 2^{s-1}) = \min(\text{card}(R'_i(w')_{=0}), 2^{s-1})$,
- $\min(\text{card}(\widehat{R}_i(w)_{=1}), 2^{s-1}s) = \min(\text{card}(R'_i(w')_{=1}), 2^{s-1}s)$,
- $\min(\text{card}(\widehat{R}_i(w)_{=2}), 2^{s-2}s(s+1)) = \min(\text{card}(R'_i(w')_{=2}), 2^{s-2}s(s+1))$.

Again, notice that these three properties, which we later refer to with (††), are exactly (zero), (one) and (two) in the proof of (A). Let us see how to use these pieces of information to derive a strategy for the duplicator for the game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$. As the spoiler chooses (\mathcal{K}', w') , it selects \mathcal{K}'_1 and \mathcal{K}'_2 such that $\mathcal{K}'_1 + \mathcal{K}'_2 = \mathcal{K}'$. Consider the two structures $\widehat{\mathcal{K}}_1 = (\mathcal{W}, \widehat{R}_1, \mathcal{V})$ and $\widehat{\mathcal{K}}_2 = (\mathcal{W}, \widehat{R}_2, \mathcal{V})$ chosen by the duplicator following the strategy, discussed above, for the game $((\widehat{\mathcal{K}}, w), (\mathcal{K}', w'), (m, s, P))$ in the case when the spoiler chooses (\mathcal{K}', w') and again select \mathcal{K}'_1 and \mathcal{K}'_2 . In particular these structures satisfy (††). Moreover, the two forests $\widehat{\mathcal{K}}_1$ and $\widehat{\mathcal{K}}_2$ are such that $\widehat{\mathcal{K}}_1 + \widehat{\mathcal{K}}_2 = \widehat{\mathcal{K}}$ and therefore $\widehat{R}_1 \cup \widehat{R}_2 = \widehat{R} = R \setminus \{(w_1, w_2)\}$ where $(w_1, w_2) \in R$ and $w_1 \in R(w)_{=1}$. We have two cases.

- If $w_1 \in \widehat{R}_1(w)$ then in the original game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$, the duplicator replies to \mathcal{K}'_1 and \mathcal{K}'_2 with the two structures $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $R_1 = \widehat{R}_1$ and $R_2 = \widehat{R}_2 \cup \{(w_1, w_2)\}$.
- Otherwise $w_1 \in \widehat{R}_2(w)$ and in the game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$ the duplicator replies to \mathcal{K}'_1 and \mathcal{K}'_2 with the two structures $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $R_1 = \widehat{R}_1 \cup \{(w_1, w_2)\}$ and $R_2 = \widehat{R}_2$.

In both cases, as the pair (w, w_1) is in one relation between R_1 and R_2 whereas (w_1, w_2) is in the other relation, the world w_1 effectively behaves as if it was a member of the set $R(w)_{=0}$ instead of $R(w)_{=1}$, exactly as in the case of \widehat{R}' . In particular, it is easy to see that for all $i \in \{1, 2\}$ and $j \in [1, 3]$, $\text{card}(R_i(w)_{=j}) = \text{card}(\widehat{R}_i(w)_{=j})$. Hence, by (††) we have

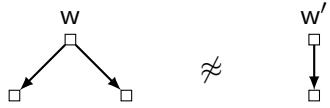
- $\min(\text{card}(R_i(w)_{=0}), 2^{s-1}) = \min(\text{card}(R'_i(w')_{=0}), 2^{s-1})$,
- $\min(\text{card}(R_i(w)_{=1}), 2^{s-1}s) = \min(\text{card}(R'_i(w')_{=1}), 2^{s-1}s)$,
- $\min(\text{card}(R_i(w)_{=2}), 2^{s-2}s(s+1)) = \min(\text{card}(R'_i(w')_{=2}), 2^{s-2}s(s+1))$.

Moreover, \mathcal{K}_1 , \mathcal{K}_2 , \mathcal{K}'_1 and \mathcal{K}'_2 all satisfy (I), (II) and (III) (as they are submodels of \mathcal{K} or \mathcal{K}'), we can apply (A) and derive $(\mathcal{K}_1, w) \approx_{m,s-1}^P (\mathcal{K}'_1, w')$ and $(\mathcal{K}_2, w) \approx_{m,s-1}^P (\mathcal{K}'_2, w')$. Therefore, the play we just described leads to a winning strategy for the duplicator on the game $((\mathcal{K}, w), (\mathcal{K}', w'), (m, s, P))$, under the hypothesis that the spoiler chooses (\mathcal{K}', w') .

As we constructed a strategy for the duplicator in both cases where the spoiler picks (\mathcal{K}, w) and (\mathcal{K}', w') , we have that $(\mathcal{K}, w) \approx_{m,s}^P (\mathcal{K}', w')$ and therefore (B) holds. This implies that the class of models satisfying $\Diamond_{=2}\Diamond_{=1}\top$ cannot be characterised by a formula in $\text{ML}(\ast)$. \square

Lemma 9.12 shows that GML is more expressive than $\text{ML}(\ast)$. This places the expressive power of $\text{ML}(\ast)$ strictly in between ML and GML. Indeed, it is quite easy to see that $\text{ML}(\ast)$ is more expressive than ML . Consider the formula $\Diamond\top * \Diamond\top$. By Lemma 9.2, this formula is

equivalent to $\Diamond \top \mid \Diamond \top$, which in turn is equivalent to $\Diamond_{\geq 2} \top$. However, this latter formula of GML is known to be not expressible in ML [15]. Formally, this can be proved by considering the two pointed forests below, which are bisimilar and hence indistinguishable in ML . However, only the pointed forest on the left satisfies $\Diamond_{\geq 2} \top$.



The following theorem summarises Corollary 7.18, Lemma 9.10, and Lemma 9.12.

Theorem 9.13. $\text{ML} \prec \text{ML}(*) \prec \text{GML} \approx \text{ML}(\mid)$.

9.3 THE COMPLEXITY OF $\text{ML}(*)$

In this section, we show the surprising fact that, despite $\text{ML}(*)$ being less expressive than $\text{ML}(\mid)$, the satisfiability problem for the former logic is by far higher than the one for the latter. In particular, we show that the satisfiability problem of $\text{ML}(*)$ is TOWER-complete. We recall that TOWER is the non-elementary class of all problems of time complexity bounded by a tower of exponentials whose height is an elementary function [128]. We already encountered several TOWER-complete logics in Chapter 4, where we analysed the interactions between reachability and submodel reasoning by introducing the logic ALT. Interestingly enough, the TOWER-hardness of the satisfiability problem of ALT was achieved, thanks to reachability and submodel reasoning, by encoding non-elementary long finite words as paths inside forests. The TOWER-hardness of $\text{ML}(*)$ is different in nature: from Proposition 9.3, we know that the length of paths required to satisfy a formula φ is bounded by the modal depth $\text{md}(\varphi)$, making impossible to describe paths of non-elementary lengths. This leaves only one dimension free from elementary bounds: the branching factor (i.e. the number of children) of the worlds in the pointed forest. This makes the complexities of ALT and $\text{ML}(*)$ in some sense orthogonal, as ALT cannot express huge branching factors. As we will see, bringing to fruition the idea of having a non-elementary number of children reveals to be technically challenging.

9.3.1 The problem of tiling a (huge) grid.

Given $k, n \geq 0$, we define the tetration function t as $t(0, n) \stackrel{\text{def}}{=} n$ and $t(k + 1, n) \stackrel{\text{def}}{=} 2^{t(k, n)}$. Intuitively, $t(k, n)$ defines a tower of exponentials of height k . We recall that k -NEXPTIME stands for the class of all problems decidable with a non-deterministic Turing machine having runtime in $\mathcal{O}(t(k, p(n)))$, for some polynomial $p(\cdot)$, on each input of length n . To show TOWER-hardness, we design a uniform elementary reduction allowing us to get k -NEXPTIME-hardness for all k greater than a certain (fixed) integer. In our case, we achieve an exponential-space reduction from the k -NEXPTIME variant of the (standard) tiling problem, for all $k \geq 2$ [119].

We introduce the problem of tiling a grid of size $t(k, n)^2$, and refer the reader to [119] for an introduction to tiling (a.k.a. domino) problems. The *tiling problem* Tile_k takes as input a triple $\mathcal{T} = (\mathsf{T}, \mathsf{H}, \mathsf{V})$ where T is a finite set of *tile types*, $\mathsf{H} \subseteq \mathsf{T} \times \mathsf{T}$ represents an *horizontal* matching relation, and $\mathsf{V} \subseteq \mathsf{T} \times \mathsf{T}$ represents a *vertical* matching relation, together with an initial tile type $c \in \mathsf{T}$. Informally, Tile_k asks to tile a square grid $[0, t(k, n) - 1] \times [0, t(k, n) - 1]$ by assigning a tile type to each position (i, j) of the grid, while respecting the horizontal and

TILE TYPES	MATCHING RELATION	A SOLUTION

Figure 9.8: An instance of (Wang) tiling problem, and a possible solution for a 3×3 grid.

vertical matching relations and assigning c to the position $(0, 0)$ of the grid. So, formally, a solution for the instance (\mathcal{T}, c) is a mapping $\tau : [0, t(k, n) - 1] \times [0, t(k, n) - 1] \rightarrow T$ such that

- (first) $\tau(0, 0) = c$,
- (hori) for every $i \in [0, t(k, n) - 2]$ and $j \in [0, t(k, n) - 1]$, $(\tau(i, j), \tau(i + 1, j)) \in H$,
- (vert) for all $i \in [0, t(k, n) - 1]$ and $j \in [0, t(k, n) - 2]$, $(\tau(i, j), \tau(i, j + 1)) \in V$.

The problem of checking whether an instance of Tile_k admits a solution is known to be k -NEXPTIME-complete (see [119]). Figure 9.8 shows an example of tiling problem due to H. Wang, where both horizontal and vertical matching relations can be represented by colours on the edges of tile types [142], as intuitively shown in the central diagram.

Our reduction from Tile_k to the satisfiability problem of $\text{ML}(\ast)$ recycles quite heavily ideas from [8] to reduce Tile_k to the satisfiability problem of second-order modal logic QK (see Section 8.1.1) interpreted on finite trees, here denoted by QK^t . However, to provide the adequate adaptation for $\text{ML}(\ast)$, we need to solve two major issues. First, QK^t admits second-order quantification, whereas in $\text{ML}(\ast)$, the second-order features are limited to the separating conjunction \ast . Second, the second-order quantification of QK^t essentially colours the nodes in Kripke-style structures without changing the frame (\mathcal{W}, R) . By contrast, the operator \ast modifies the accessibility relation, possibly making worlds that were reachable from the current world, unreachable in submodels. The TOWER-hardness proof for the satisfiability problem of $\text{ML}(\ast)$ becomes then much more challenging: we would like to characterise the position on the grid encoded by a world w by exploiting properties of its descendants (as done for QK^t), but at the same time, we need to be careful and only consider submodels where w keeps encoding the same position. In a sense, our encoding is robust: when the operator \ast is used to reason on submodels, we can enforce that no world changes the position of the grid that it encodes.

9.3.2 Enforcing $t(j, n)$ children.

Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ be a finite forest. We consider two disjoint sets of atomic propositions $P = \{p_1, \dots, p_n, \text{val}\}$ and $\text{Aux} = \{\mathbf{x}, \mathbf{y}, \mathbf{l}, \mathbf{s}, \mathbf{r}\}$ (whose respective role is later defined). Elements from Aux are understood as *auxiliary* propositions. We call **ax-node** a world satisfying the proposition $\mathbf{ax} \in \text{Aux}$, whereas an **Aux-node** world is a world satisfying some proposition in Aux . We call **t-node** a world that satisfies the formula $t \stackrel{\text{def}}{=} \bigwedge_{\mathbf{ax} \in \text{Aux}} \neg \mathbf{ax}$. Notice that every world of \mathcal{K}

is either a t -node or an Aux -node. We say that w' is a t -child of $w \in \mathcal{W}$ if $w' \in R(w)$ and w' is a t -node. We define the concepts of Aux -child and ax -child similarly.

The key development of our reduction is given by the definition of a formula, of exponential size in $j \geq 1$ and polynomial size in $n \geq 1$, that when satisfied by (\mathcal{K}, w) forces every t -node in $R^i(w)$, where $0 \leq i < j$, to have exactly $t(j-i, n)$ t -children, each of them encoding a different number in $[0, t(j-i, n) - 1]$. As we impose that w is a t -node, it must have $t(j, n)$ t -children. We assume n to be fixed throughout the section and denote this formula by $\text{type}(j)$. From the property we just described, $(\mathcal{K}, w) \models \text{type}(j)$ implies that for every $i \in [1, j-1]$ and every t -nodes $w' \in R^i(w)$, $\mathcal{K}, w' \models \text{type}(j-i)$.

First, let us informally describe how numbers are encoded in the model (\mathcal{K}, w) satisfying $\text{type}(j)$. Let $i \in [1, j]$. Given a t -node $w' \in R^i(w)$, $\mathbf{n}_{j-i}(w')$ denotes the number encoded by w' . We omit the subscript i when it is clear from the context. When $i = j$, we represent $\mathbf{n}_0(w')$ in binary, by using the truth values of the atomic propositions p_1, \dots, p_n . The proposition p_b is responsible for the b -th bit of the number, with the least significant bit being encoded by p_1 , and the most significant one being encoded by p_n . Formally, given $m \in [0, 2^n - 1]$ and its binary representation $b_n \dots b_1 \in \{0, 1\}^n$,

$$\mathbf{n}_0(w') = m \text{ if and only if for all } l \in [1, n], b_l = 1 \text{ iff } w' \in \mathcal{V}(p_l).$$

For example, for $n = 3$, we have $(\mathcal{K}, w') \models p_3 \wedge p_2 \wedge \neg p_1$ whenever $\mathbf{n}(w') = 6$. The formula $\text{type}(1)$ forces the parent w'' of w' (i.e. a t -node in $R^{j-1}(w)$) to have exactly 2^n t -children, each representing a distinct number in $[0, 2^n - 1]$, via the valuation of p_1, \dots, p_n . In general, for $i < j$ (and thus $j \geq 2$), the number $\mathbf{n}_{j-i}(w')$ is represented by the binary encoding of the truth values of val on the t -children of w' which, since $(\mathcal{K}, w') \models \text{type}(j-i)$, are $t(j-i, n)$ children implicitly ordered by the number they, in turn, encode. In the case of w'' for instance, the number $\mathbf{n}_1(w'')$ is encoded by relying the valuation of the atomic proposition val on its 2^n children, where its t -child w' such that $\mathbf{n}_0(w') = 6$ is responsible for the 6th least significant bit of $\mathbf{n}_1(w'')$. If w' satisfies val , then this bit is set to 1, otherwise it is set to 0. Therefore, the essential property of $\text{type}(j)$ is the following: the numbers encoded by the t -children of a t -node $w'' \in R^i(w)$, where $i < j$, represent positions in the binary representation of the number $\mathbf{n}_{j-i}(w'')$. Formally, given $m \in [0, t(j-i+1, n) - 1]$ and its binary representation $b_{t(j-i+1, n)-1} \dots b_0$,

$$\mathbf{n}_{j-i}(w') = m \text{ if and only if for all } l \in [0, t(j-i, n) - 1], b_l = 1 \text{ iff } w_l \in \mathcal{V}(\text{val}),$$

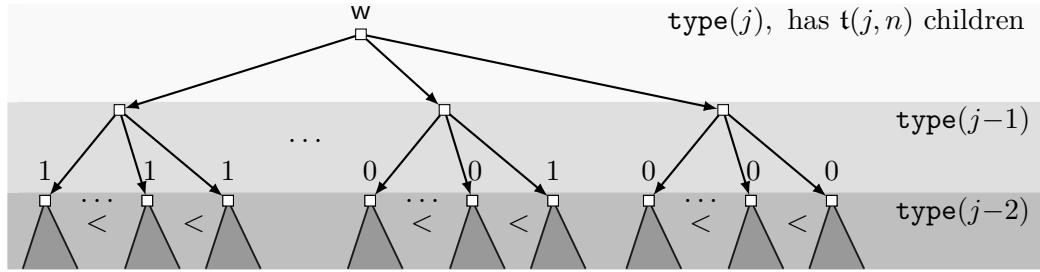
where w_l is the only t -child of w' such that $\mathbf{n}_{j-i-1}(w_l) = l$.

Thanks to this encoding, $\text{type}(j)$ then forces w to have exactly $t(j, n)$ children, all encoding different numbers in $[0, t(j, n) - 1]$. This is roughly depicted in Figure 9.9, where “1” and “0” stands for val being true and false, respectively.

In order to characterise these types of structures in $\text{ML}(\ast)$, we essentially simulate the second-order quantification of QK by using the Aux -nodes. Informally, we require a pointed forest (\mathcal{K}, w) satisfying $\text{type}(j)$ to be such that

- (i) every t -node $w' \in R(w)$ has exactly one x -child, and one (different) y -child. These nodes do not satisfy any other auxiliary proposition;
- (ii) for every $i \in [2, j]$, every t -node $w' \in R^i(w)$ has exactly five Aux -children, each satisfying a distinct auxiliary proposition from Aux .

We can simulate the second-order quantification on t -nodes with respect to the symbol $\text{ax} \in \text{Aux}$ by using the operator $*$ in order to remove edges leading to ax -nodes. Roughly speaking, this

Figure 9.9: Schema of a pointed forest $(\mathcal{K}, \mathbf{w})$ satisfying $\text{type}(j)$, for $j \geq 3$.

operation corresponds to $\exists \mathbf{ax} \varphi$. Then, we evaluate whether φ holds on the resulting pointed forest, where the satisfaction of \mathbf{ax} is interpreted on t -nodes, so that one of these nodes “satisfies” \mathbf{ax} if it has a child satisfying \mathbf{ax} . In a sense, \mathbf{Aux} -nodes are mere atomic propositions “attached” to t -nodes, ready to be removed in order to encode the second-order quantification. With this in mind, the modality \Diamond is almost exclusively used to navigate through t -nodes. To better show the complementary roles that t -nodes and \mathbf{Aux} -nodes have in the reduction, we relativise the modality \Diamond and, given a formula φ , we write $\langle t \rangle \varphi$ for the formula $\Diamond(t \wedge \varphi)$. Dually, we define $[t]$ as $\overset{\text{def}}{=} \Box(t \Rightarrow \varphi)$. $\langle t \rangle^i$ and $[t]^i$ are also defined, as expected.

Let us start to formalise this encoding. Let $j \geq 1$. First, we restrict ourselves to models where every t -node reachable in at most j steps does not have two \mathbf{Aux} -children satisfying the same proposition. Moreover, these \mathbf{Aux} -nodes have no children and only satisfy exactly one $\mathbf{ax} \in \mathbf{Aux}$. We express this condition with the formula $\text{init}(j)$ below:

$$\boxplus^j \bigwedge_{\mathbf{ax} \in \mathbf{Aux}} \left((t \Rightarrow \neg(\Diamond \mathbf{ax} * \Diamond \mathbf{ax})) \wedge \Box(\mathbf{ax} \Rightarrow \Box \perp \wedge \bigwedge_{\mathbf{bx} \in \mathbf{Aux} \setminus \{\mathbf{ax}\}} \neg \mathbf{bx}) \right),$$

where $\boxplus^0 \varphi \overset{\text{def}}{=} \varphi$ and $\boxplus^{m+1} \varphi \overset{\text{def}}{=} \varphi \wedge \Box \boxplus^m (\varphi)$. Essentially, $\boxplus^j \varphi$ is satisfied by a pointed forest $(\mathcal{K}, \mathbf{w})$ whenever every world reachable from \mathbf{w} in at most j steps satisfies φ . From this, one can easily show that if $(\mathcal{K}, \mathbf{w}) \models \text{init}(j)$ and $\mathcal{K}' \subseteq \mathcal{K}$, then $(\mathcal{K}', \mathbf{w}) \models \text{init}(j)$. The formal semantics of $\text{init}(j)$ is given in the following lemma.

Lemma 9.14. Let $(\mathcal{K}, \mathbf{w})$ be a pointed forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, and $j \geq 1$. $(\mathcal{K}, \mathbf{w}) \models \text{init}(j)$ if and only if for every $0 \leq i \leq j$, every $\mathbf{w}' \in R^i(\mathbf{w})$ and every $\mathbf{ax} \in \mathbf{Aux}$,

1. if $(\mathcal{K}, \mathbf{w}') \models t$ then for all $\mathbf{w}'_1, \mathbf{w}'_2 \in R(\mathbf{w}')$, if $(\mathcal{K}, \mathbf{w}'_1) \models \mathbf{ax}$ and $(\mathcal{K}, \mathbf{w}'_2) \models \mathbf{ax}$ then $\mathbf{w}'_1 = \mathbf{w}'_2$ (i.e. at most one child of \mathbf{w}' satisfies \mathbf{ax});
2. for every $\mathbf{w}'' \in R(\mathbf{w}')$, if $(\mathcal{K}, \mathbf{w}'') \models \mathbf{ax}$, then $R(\mathbf{w}'') = \emptyset$ (i.e. \mathbf{w}'' does not have children) and it cannot be that $(\mathcal{K}, \mathbf{w}'') \models \mathbf{bx}$ for some $\mathbf{bx} \in \mathbf{Aux}$ syntactically different from \mathbf{ax} (i.e. among the propositions in \mathbf{Aux} , \mathbf{w}'' only satisfies \mathbf{ax}).

Proof (sketch). The proof is straightforward. The phrase “for every $0 \leq i \leq j$, every $\mathbf{w}' \in R^i(\mathbf{w})$ and every $\mathbf{ax} \in \mathbf{Aux}$ ” is encoded by the prefix $\boxplus^j \bigwedge_{\mathbf{ax} \in \mathbf{Aux}}$ of $\text{init}(j)$. Then, (1) corresponds to the conjunct $t \Rightarrow \neg(\Diamond \mathbf{ax} * \Diamond \mathbf{ax})$ whereas (2) corresponds to $\Box(\mathbf{ax} \Rightarrow \Box \perp \wedge \bigwedge_{\mathbf{bx} \in \mathbf{Aux} \setminus \{\mathbf{ax}\}} \neg \mathbf{bx})$. \square

Let us define $\text{type}(0) \overset{\text{def}}{=} \top$. Among the pointed forests $((\mathcal{W}, R, \mathcal{V}), \mathbf{w})$ satisfying $\text{init}(j)$ ($j \geq 1$), the ones satisfying $\text{type}(j)$ are described by the following properties:

- (**sub_j**) every t -node in $R(\mathbf{w})$ satisfies $\text{type}(j-1)$,
- (**zero_j**) there is a t -node $\tilde{\mathbf{w}} \in R(\mathbf{w})$ such that $\mathbf{n}_{j-1}(\tilde{\mathbf{w}}) = 0$,

- (**uniq**_j) distinct t -nodes in $R(w)$ encode different numbers (w.r.t. $\mathbf{n}_{j-1}(.)$),
- (**compl**_j) for every t -node $w_1 \in R(w)$ where $\mathbf{n}_{j-1}(w_1) < t(j, n) - 1$, there is a t -node $w_2 \in R(w)$ such that $\mathbf{n}_{j-1}(w_2) = \mathbf{n}_{j-1}(w_1) + 1$,
- (**aux**) w is a t -node, every t -node in $R(w)$ has one x -child and one y -child, and every t -node in $R^2(w)$ has three children satisfying **l**, **r** and **s**, respectively.

As already explained, in a pointed forest (\mathcal{K}, w) satisfying $\text{type}(j)$, every t -node in $R^i(w)$, where $0 \leq i < j$, has exactly $t(j-i, n)$ t -children, each of them encoding a different number in $[0, t(j-i, n) - 1]$. In particular, (**sub**_j) inductively assures that this is true for the descendants of w , whereas the properties (**zero**_j), (**uniq**_j) and (**compl**_j) uses the fact that (**sub**_j) holds in order to force the correct number of children of w . In order to reflect the five properties above, let us define $\text{type}(j)$, where $j \geq 1$, as the following conjunction,

$$\text{type}(j) \stackrel{\text{def}}{=} \text{sub}(j) \wedge \text{zero}(j) \wedge \text{uniq}(j) \wedge \text{compl}(j) \wedge \text{aux},$$

where each conjunct expresses its homonymous property. The formulae for **sub**(j), **aux** and **zero**(j) can be defined very easily:

$$\begin{aligned} \text{sub}(j) &\stackrel{\text{def}}{=} [t]\text{type}(j-1), \\ \text{aux} &\stackrel{\text{def}}{=} t \wedge [t](\Diamond x * \Diamond y) \wedge [t]^2(\Diamond l * \Diamond s * \Diamond r), \\ \text{zero}(1) &\stackrel{\text{def}}{=} \langle t \rangle \wedge_{b \in [1, n]} \neg p_b, \\ \text{zero}(j+1) &\stackrel{\text{def}}{=} \langle t \rangle [t] \neg \text{val}. \end{aligned}$$

The challenge is therefore how to express **uniq**(j) and **compl**(j), the two conditions that, together with **zero**(j), guarantee that the numbers of children of w span all over $[0, t(j, n) - 1]$. Notice that **type**(j) is defined recursively on j , as highlighted by the definition of **sub**(j). As we will see, the same holds true for the formulae **uniq**(j) and **compl**(j), which forces us to divide the construction of **uniq**(j) and **compl**(j) into a base case, in which we are able to define **type**(1) (Section 9.3.4), and an inductive case in which we define **type**(j) for every $j > 1$ (Section 9.3.5).

The structural properties expressed by **type**(j) lead to strong constraints, which permits to control the effects of the separating conjunction when subforests are constructed. This is a key point in designing **type**(j) as it helps us to control which edges are lost when considering a subforest. Indeed, suppose $(\mathcal{K}, w) \models \text{init}(j) \wedge \text{type}(j)$, for $j \geq 1$. Consider two finite forest models \mathcal{K}_1 and \mathcal{K}_2 such that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$, $\mathcal{K}_1, w \models \Box \perp$ and $\mathcal{K}_2, w \models \text{sub}(j) \wedge \text{aux}$. We can conclude that, among the edges $(w', w'') \in R$ appearing in the subtree of depth j rooted at w , the structure \mathcal{K}_1 can only contain the ones such that $w' \in R(w)$ and w'' is a $\{l, s, r\}$ -node. In particular, every t -node $\tilde{w} \in R^i(w)$ ($i \in [0, j]$) must still be reachable from w in \mathcal{K}_2 , which implies that the number $\mathbf{n}(\tilde{w})$ encoded by \tilde{w} is the same in both structures. This robustness against the unpleasant side-effects of the operator $*$, i.e. the general inability to track which edges are lost when considering a subforest, is fundamental in order to reduce Tile_k . Let us formalise this fundamental property of **type**(j), assuming that the formula is correctly defined, following its specification.

Lemma 9.15. Let (\mathcal{K}, w) be a pointed forest, s.t. $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $(\mathcal{K}, w) \models \text{init}(j) \wedge \text{type}(j)$. Consider a world $w' \in R^i(w)$, where $i \in [0, j]$. Let $\mathcal{K}' \subseteq \mathcal{K}$ be such that $(\mathcal{K}', w') \models \text{type}(j-i)$. Then, the number $\mathbf{n}_{j-i}(w')$ encoded by w' is the same in both \mathcal{K} and \mathcal{K}' .

Proof. The proof is by induction on the difference $j - i$, with induction hypothesis stating that for every $k > i$, world $w'' \in R^k(w)$ and $\mathcal{K}'' \subseteq \mathcal{K}$ such that $(\mathcal{K}'', w'') \models \text{type}(j - k)$, w'' encodes the same number with respect to both \mathcal{K} and \mathcal{K}'' .

base case: $i = j$. In this case, $\text{type}(j - i) = \top$ and $\mathbf{n}_{j-i}(w')$ only depends on the satisfaction of the atomic propositions p_1, \dots, p_n , on the world w' . Since \mathcal{K}' and \mathcal{K} share the same valuation \mathcal{V} , w' encodes the same number with respect to both \mathcal{K} and \mathcal{K}' .

inductive step: $i < j$. Since $(\mathcal{K}', w') \models \text{type}(j - i)$, by (sub_j) every t -child w'' of w' is such that $(\mathcal{K}', w'') \models \text{type}(j - (i + 1))$. By induction hypothesis, w'' encodes the same number with respect to both \mathcal{K} and \mathcal{K}' . So, in order for w' to encode the same number in \mathcal{K} and \mathcal{K}' , we need to check that the set of t -children of w' is the same in both \mathcal{K} and \mathcal{K}' . Notice that is sufficient, as the number $\mathbf{n}_{j-i}(w')$ is encoded using the satisfaction of val in on these children, and \mathcal{K} and \mathcal{K}' share the same valuation function \mathcal{V} . From $(\mathcal{K}, w) \models \text{type}(j)$, by (sub_j), we know that $(\mathcal{K}, w') \models \text{type}(j - i)$. From (zero_j), (uniq_j) and (compl_j), we conclude that, in \mathcal{K} , the world w' has exactly $\mathbf{t}(j - i, n)$ t -children, each of them encoding a different number in $[0, \mathbf{t}(j - i, n)]$. From the hypothesis $(\mathcal{K}', w') \models \text{type}(j - i)$, the same holds true with respect to \mathcal{K}' . As $\mathcal{K}' \subseteq \mathcal{K}$, we conclude that the set of t -children of w' is the same in both \mathcal{K} and \mathcal{K}' . \square

9.3.3 Nominals, forks and number comparisons.

In order to define the formulae $\text{uniq}(j)$ and $\text{compl}(j)$ (completing the definition of $\text{type}(j)$), we introduce auxiliary formulae, characterising classes of models that emerge naturally when trying to capture the semantics of (uniq_j) and (compl_j). As usual, below we consider a pointed forest (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

A first ingredient is given by the concept of *local nominals*, borrowed from [8]. We say that $\mathbf{ax} \in \mathbf{Aux}$ is a (local) nominal for the depth $i \geq 1$ if there is exactly one t -node $w' \in R^i(w)$ having an \mathbf{ax} -child. In this case, w' is said to be the world that corresponds to the local nominal \mathbf{ax} . The following formula states that \mathbf{ax} is a local nominal for the depth i :

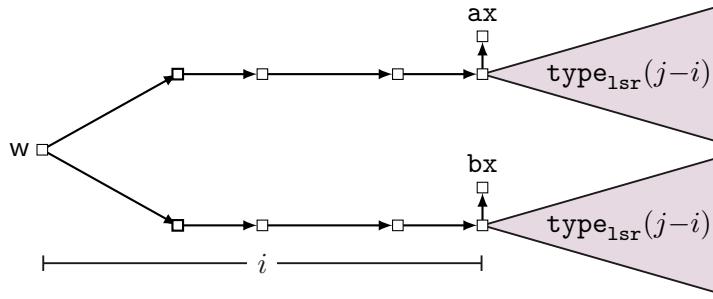
$$\text{nom}_i(\mathbf{ax}) \stackrel{\text{def}}{=} \langle t \rangle^i \Diamond \mathbf{ax} \wedge \bigwedge_{k \in [0, i-1]} [t]^k \neg (\langle t \rangle^{i-k} \Diamond \mathbf{ax} * \langle t \rangle^{i-k} \Diamond \mathbf{ax}).$$

We define the formula $\text{@}_{\mathbf{ax}}^i \varphi \stackrel{\text{def}}{=} \langle t \rangle^i (\Diamond \mathbf{ax} \wedge \varphi)$ which, under the hypothesis that \mathbf{ax} is a local nominal for the depth i , states that φ holds on the t -node that corresponds to \mathbf{ax} . Moreover, we define $\text{nom}_i(\mathbf{ax} \neq \mathbf{bx}) \stackrel{\text{def}}{=} \text{nom}_i(\mathbf{ax}) \wedge \text{nom}_i(\mathbf{bx}) \wedge \neg \text{@}_{\mathbf{ax}}^i \Diamond \mathbf{bx}$, which states that \mathbf{ax} and \mathbf{bx} are two nominals for the depth i with respect to two distinct t -nodes. The following lemma formalises the semantics of $\text{nom}_i(\mathbf{ax})$, $\text{@}_{\mathbf{ax}}^i \varphi$ and $\text{nom}_i(\mathbf{ax} \neq \mathbf{bx})$.

Lemma 9.16. Let $\mathbf{ax} \in \mathbf{Aux}$ and $0 < i \leq j \in \mathbb{N}$. Suppose $(\mathcal{K}, w) \models \text{init}(j)$.

1. $(\mathcal{K}, w) \models \text{nom}_i(\mathbf{ax})$ if and only if \mathbf{ax} is a nominal for the depth i .
2. Suppose $(\mathcal{K}, w) \models \text{nom}_i(\mathbf{ax})$. $(\mathcal{K}, w) \models \text{@}_{\mathbf{ax}}^i \varphi$ if and only if the world (say w') corresponding to the nominal \mathbf{ax} for the depth i is such that $(\mathcal{K}, w') \models \varphi$.
3. $(\mathcal{K}, w) \models \text{nom}_i(\mathbf{ax} \neq \mathbf{bx})$ iff \mathbf{ax} and \mathbf{bx} are nominals for the depth i , for two distinct worlds.

Below, we show the proof of Lemma 9.16(1), and refer the reader to Appendix G for the (easy) proofs of the other two statements.

Figure 9.10: Schema of a pointed forest (\mathcal{K}, w) satisfying $\text{fork}_j^i(\text{ax}, \text{bx})$.

Proof of Lemma 9.16(1). Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{nom}_i(\text{ax})$, and so there is a path of t -worlds w_0, w_1, \dots, w_i , such that $w = w_0$, for every $k \in [1, i]$ $(w_{k-1}, w_k) \in R$, and there is w' such that $(w_i, w') \in R$ and $(\mathcal{K}, w') \models \text{ax}$. The second conjunct of $\text{nom}_i(\text{ax})$ guarantees that there is only one such path, leading to w_i being a nominal for the depth i . Indeed, *ad absurdum*, assume that there is a second world $w'_i \in R^i(w)$, distinct from w_i , such that $(\mathcal{K}, w'_i) \models \Diamond \text{ax}$. Since $(\mathcal{K}, w) \models \text{init}(j)$, w'_i must be a t -node and there must be a path of t -worlds w'_0, w'_1, \dots, w'_i such that $w = w'_0$ and for every $k \in [1, i]$ $(w'_{k-1}, w'_k) \in R$. As we are assuming $w_i \neq w'_i$, there must be $k \in [0, i-1]$ such that for every $j \leq k$, $w_j = w'_j$, and for every $l \in [k+1, i]$, $w_l \neq w'_l$. By considering the pointed forest (\mathcal{K}, w_k) , we can easily show that $(\mathcal{K}, w_k) \models \langle t \rangle^{i-k} \Diamond \text{ax} * \langle t \rangle^{i-k} \Diamond \text{ax}$. This implies that $(\mathcal{K}, w) \models \langle t \rangle^k (\langle t \rangle^{i-k} \Diamond \text{ax} * \langle t \rangle^{i-k} \Diamond \text{ax})$, in contradiction with the second conjunct of $\text{nom}_i(\text{ax})$. We conclude that, $w'_i = w_i$.

(\Leftarrow): This direction is straightforward. Suppose that $(\mathcal{K}, w) \models \text{init}(j)$ and ax is a nominal for the depth i . By definition, there is a unique t -world w' in $R^i(w)$ having a child satisfying ax . Since $(\mathcal{K}, w) \models \text{init}(j)$, the path from w to w' must only witness t -nodes. Hence $(\mathcal{K}, w) \models \langle t \rangle^i \Diamond \text{ax}$. From the uniqueness of this path we conclude that the formula $(\mathcal{K}, w) \models \bigwedge_{k \in [0, i-1]} [t]^k \neg (\langle t \rangle^{i-k} \Diamond \text{ax} * \langle t \rangle^{i-k} \Diamond \text{ax})$ also holds. Thus, $(\mathcal{K}, w) \models \text{nom}_i(\text{ax})$. \square

As a second ingredient, we introduce the notion of *fork* that is a specific type of models naturally emerging when trying to compare the numbers $\mathbf{n}(w_1)$ and $\mathbf{n}(w_2)$ of two worlds $w_1, w_2 \in R^i(w)$ (e.g. when checking whether $\mathbf{n}(w_1) = \mathbf{n}(w_2)$ or $\mathbf{n}(w_2) = \mathbf{n}(w_1) + 1$ holds). Given $j \geq i \geq 1$ we introduce the formula $\text{fork}_j^i(\text{ax}, \text{bx})$, where $\text{ax} \neq \text{bx} \in \text{Aux}$, that is satisfied by (\mathcal{K}, w) if:

(fork₁) ax and bx are nominals for the depth i ,

(fork₂) w has exactly two t -children, say w_U and w_D ,

(fork₃) for every $k \in [1, i-1]$, both $R^k(w_U)$ and $R^k(w_D)$ contain exactly one t -child,

(fork₄) if $i < j$, then $(\mathcal{K}, w_{\text{ax}})$ and $(\mathcal{K}, w_{\text{bx}})$ satisfy

$$\text{type}_{1\text{sr}}(j-i) \stackrel{\text{def}}{=} \text{type}(j-i) \wedge [t](\Diamond \mathbf{l} \wedge \Diamond \mathbf{s} \wedge \Diamond \mathbf{r}).$$

Notice that the properties **(fork₁)** and **(fork₃)** entail that w_{ax} and w_{bx} are the only t -nodes in $R^{i-1}(w_U) \cup R^{i-1}(w_D)$, with one belonging in $R^{i-1}(w_U)$ and the other being in $R^{i-1}(w_D)$. Therefore, whenever (\mathcal{K}, w) satisfies the formula $\text{fork}_j^i(\text{ax}, \text{bx})$, we witness two paths of length i , both starting at w and leading to w_{ax} and w_{bx} , respectively. Worlds in this path may have additional Aux -children. Figure 9.10 schematises a pointed forest satisfying $\text{fork}_j^i(\text{ax}, \text{bx})$. Notice that, if $i < j$, then **(fork₄)** requires the two worlds corresponding to the nominal ax and bx to satisfy $\text{type}_{1\text{sr}}(j-i)$, making the definition of $\text{fork}_j^i(\text{ax}, \text{bx})$ recursive on i and j (recall

Hypothesis on $(\mathcal{K}, \mathbf{w})$	Formula	Semantics
$\text{init}(j)$	$\text{fork}_j^i(\mathbf{ax}, \mathbf{bx})$	$(\mathcal{K}, \mathbf{w})$ satisfies $(\text{fork}_1) - (\text{fork}_4)$
$\text{fork}_j^i(\mathbf{ax}, \mathbf{bx}) \wedge \text{init}(j)$	$[\mathbf{ax} < \mathbf{bx}]_j^i$	$\mathbf{n}_{j-i}(\mathbf{w}_{\mathbf{ax}}) < \mathbf{n}_{j-i}(\mathbf{w}_{\mathbf{bx}})$
$\text{fork}_j^1(\mathbf{ax}, \mathbf{bx}) \wedge \text{init}(j)$	$[\mathbf{bx} = \mathbf{ax}+1]_j$	$\mathbf{n}_{j-1}(\mathbf{w}_{\mathbf{ax}}) + 1 = \mathbf{n}_{j-1}(\mathbf{w}_{\mathbf{bx}})$
$\text{init}(j) \wedge \text{aux} \wedge \text{sub}(j)$	$\text{uniq}(j)$	$(\mathcal{K}, \mathbf{w})$ satisfies (uniq_j)
$\text{init}(j) \wedge \text{aux} \wedge \text{sub}(j)$	$\text{compl}(j)$	$(\mathcal{K}, \mathbf{w})$ satisfies (compl_j)
$\text{init}(j)$	$\text{type}(j)$	$(\mathcal{K}, \mathbf{w})$ satisfies $(\text{sub}_j) - (\text{aux})$

Figure 9.11: Summary of the formulae to be defined in the next sections.

that $\text{fork}_j^i(\mathbf{ax}, \mathbf{bx})$ is introduced in order to define $\text{type}(j)$. Because of this, we postpone its formal definition to the next two sections, where we treat the base cases for $i = j$ and the inductive case for $j > i$ separately.

The last two fundamental ingredients required to define $\text{uniq}(j)$ and $\text{compl}(j)$ are given by the formulae $[\mathbf{ax} < \mathbf{bx}]_j^i$ and $[\mathbf{bx} = \mathbf{ax}+1]_j$. Under the hypothesis that $(\mathcal{K}, \mathbf{w})$ satisfies $\text{fork}_j^i(\mathbf{ax}, \mathbf{bx})$ and $\text{init}(j)$, the formula $[\mathbf{ax} < \mathbf{bx}]_j^i$ is satisfied whenever the two (distinct) worlds $\mathbf{w}_{\mathbf{ax}}, \mathbf{w}_{\mathbf{bx}} \in R^i(\mathbf{w})$ corresponding to the nominals \mathbf{ax} and \mathbf{bx} are such that $\mathbf{n}_{j-i}(\mathbf{w}_{\mathbf{ax}}) < \mathbf{n}_{j-i}(\mathbf{w}_{\mathbf{bx}})$. Similarly, under the hypothesis that $(\mathcal{K}, \mathbf{w})$ satisfies $\text{fork}_j^1(\mathbf{ax}, \mathbf{bx})$, the formula $[\mathbf{bx} = \mathbf{ax}+1]_j$ is satisfied whenever $\mathbf{n}_{j-1}(\mathbf{w}_{\mathbf{bx}}) = \mathbf{n}_{j-1}(\mathbf{w}_{\mathbf{ax}}) + 1$ holds. Both formulae are recursively defined with respect to i and j , with base cases for $i = j$ and $j = 1$, respectively.

In the next section, we define the formulae $\text{fork}_j^j(\mathbf{ax}, \mathbf{bx})$ and $[\mathbf{ax} < \mathbf{bx}]_j^j$ (for arbitrary j), as well as $[\mathbf{bx} = \mathbf{ax}+1]_1$. From these formulae, we are then able to define $\text{uniq}(1)$ and $\text{compl}(1)$, which completes the characterisation of $\text{type}(1)$ and $\text{type}_{\text{lsr}}(1)$. Afterwards (Section 9.3.5), we consider the case $1 \leq i < j$ and $j \geq 2$, and define $\text{fork}_j^i(\mathbf{ax}, \mathbf{bx})$, $[\mathbf{ax} < \mathbf{bx}]_j^i$, $[\mathbf{bx} = \mathbf{ax}+1]_j$, as well as $\text{uniq}(j)$ and $\text{compl}(j)$, by only relying on formulae that are already defined (by inductive reasoning). Figure 9.11 summarises the semantics of all these formulae. In the table, the indices i and j are such that $1 \leq i \leq j$, whereas \mathbf{ax} and \mathbf{bx} are two distinct propositions from Aux . Lastly, $\mathbf{w}_{\mathbf{ax}}$ and $\mathbf{w}_{\mathbf{bx}}$ stands for worlds corresponding to nominals \mathbf{ax} and \mathbf{bx} , respectively.

9.3.4 Construction of $\text{type}(j)$: formulae for the base cases $i = j$ or $j = 1$.

In this section, we define the formulae $\text{fork}_j^j(\mathbf{ax}, \mathbf{bx})$, $[\mathbf{ax} < \mathbf{bx}]_j^j$ and $[\mathbf{bx} = \mathbf{ax}+1]_1$, in which we rely on in order to define $\text{uniq}(1)$ and $\text{compl}(1)$, completing the definition of $\text{type}(1)$. Since we are treating the base case for $i = j$ or $j = 1$, given a world \mathbf{w} , we are only interested in its number $\mathbf{n}_0(\mathbf{w})$, which is encoded by relying on the atomic propositions p_1, \dots, p_n . Thus, in this section we always write $\mathbf{n}(\mathbf{w})$ for $\mathbf{n}_0(\mathbf{w})$.

By looking at the properties required by $\text{fork}_j^i(\mathbf{ax}, \mathbf{bx})$, it is quite easy to see that the following formula defines a fork for $i = j$:

$$\text{fork}_j^j(\mathbf{ax}, \mathbf{bx}) \stackrel{\text{def}}{=} \Diamond_{=2} t \wedge [t] \boxplus^{j-2} (t \Rightarrow \Diamond_{=1} t) \wedge \text{nom}_j(\mathbf{ax} \neq \mathbf{bx}),$$

where we extend the definition of \boxplus^k so that, given $k < 0$, $\boxplus^k \varphi \stackrel{\text{def}}{=} \top$. Moreover, the formulae $\Diamond_{=1} t$ and $\Diamond_{=2} t$ stand for $\Diamond t \wedge \neg(\Diamond t * \Diamond t)$ and $\Diamond_{=1} t * \Diamond_{=1} t$, respectively. Since both formulae have modal depth 1, from Lemma 9.2 it is easy to see that $\Diamond_{=1} t$ and $\Diamond_{=2} t$ correspond (se-

 \parallel Figure 9.12: $\mathbf{n}(w_{ax}) < \mathbf{n}(w_{bx})$. \parallel Figure 9.13: $\mathbf{n}(w_{ax}) + 1 = \mathbf{n}(w_{bx})$.

mantically) to the syntactically equivalent formulae of GML. A simple check assures that the formula $\text{fork}_j^j(ax, bx)$ is correctly defined.

Lemma 9.17. Let $ax \neq bx \in \text{Aux}$, $j \geq 1$ and $(\mathcal{K}, w) \models \text{init}(j)$. $(\mathcal{K}, w) \models \text{fork}_j^j(ax, bx)$ if and only if (\mathcal{K}, w) satisfies (fork₁)–(fork₄).

Proof. As $i = j$, the property (fork₄) is trivially satisfied in both directions of the lemma.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{fork}_j^j(ax, bx)$. From $(\mathcal{K}, w) \models \text{nom}_j(ax \neq bx)$, ax and bx are two nominals for the depth j , as required by (fork₁), corresponding to two distinct worlds. Let w_{ax} (resp. w_{bx}) be the world that corresponds to the nominal ax (resp. bx). Moreover, let w_U be the only t -children of w such that $w_{ax} \in R^{j-1}(w_U)$, and w_D be the only t -children of w such that $w_{bx} \in R^{j-1}(w_D)$. From $(\mathcal{K}, w) \models [t] \boxplus^{j-2} (t \Rightarrow \Diamond_{=1} t)$ we deduce that for every $k \in [1, j-1]$, both $R^k(w_U)$ and $R^k(w_D)$ contain exactly one t -child, as required by (fork₃). As $w_{ax} \in R^{j-1}(w_U)$, $w_{bx} \in R^{j-1}(w_D)$, and $w_{ax} \neq w_{bx}$, this implies that $w_U \neq w_D$. From $(\mathcal{K}, w) \models \Diamond_{=2} t$, w has exactly two t -children, i.e. w_U and w_D , as required by (fork₂).

(\Leftarrow): This direction is straightforward. In short, from (fork₂) we conclude that $(\mathcal{K}, w) \models \Diamond_{=2} t$ whereas (fork₃) implies the satisfaction of $[t] \boxplus^{j-2} (t \Rightarrow \Diamond_{=1} t)$. Lastly, from (fork₁) together with Lemma 9.16, we have $(\mathcal{K}, w) \models \text{nom}_j(ax \neq bx)$. \square

Let us now move to the definition of $[ax < bx]_j^j$ and $[bx = ax+1]_1$. Given two worlds w_{ax} and w_{bx} corresponding to nominals ax and bx at depth j (resp. 1), we recall that $[ax < bx]_j^j$ (resp. $[bx = ax+1]_1$) states that $\mathbf{n}(w_{ax}) < \mathbf{n}(w_{bx})$ (resp. $\mathbf{n}(w_{ax}) + 1 = \mathbf{n}(w_{bx})$). In both the base case and inductive step, in order to define these formulae we rely on standard definitions of $<$ and $+1$ for bit vectors. In particular, given two m -bits (for some $m \in \mathbb{N}$) binary representations b_{ax} and b_{bx} of $\mathbf{n}(w_{ax})$ and $\mathbf{n}(w_{bx})$, respectively, we consider the characterisation of $\mathbf{n}(w_{ax}) < \mathbf{n}(w_{bx})$ and $\mathbf{n}(w_{ax}) + 1 = \mathbf{n}(w_{bx})$ described below:

$$\mathbf{n}(w_{ax}) < \mathbf{n}(w_{bx}) \quad \text{iff} \quad b_{ax} = b \cdot 0 \cdot b' \text{ and } b_{bx} = b \cdot 1 \cdot b'' \text{ for some } b, b', b'' \in \{0, 1\}^*,$$

$$\mathbf{n}(w_{ax}) + 1 = \mathbf{n}(w_{bx}) \quad \text{iff} \quad b_{ax} = b \cdot 0 \cdot 1^k \text{ and } b_{bx} = b \cdot 1 \cdot 0^k, \text{ for some } b \in \{0, 1\}^*, k \in [0, m].$$

where $b_1 \cdot b_2$ is the concatenation of bit vectors, b^0 is the zero size bit vector and $b^{k+1} \stackrel{\text{def}}{=} b \cdot b^k$. Here, notice that we are assuming b_{ax} and b_{bx} to be encoded so that the rightmost bit is the least significant. The visual representation of these standard characterisations of $<$ and $+1$ is given in Figure 9.12 and Figure 9.13, respectively. Let us rephrase them in the language of $\text{ML}(\ast)$.

As previously explained, in the base case, the number $\mathbf{n}(w')$ encoded by a t -node $w' \in R^j(w)$ is represented by the truth values of the atomic propositions p_1, \dots, p_n , where the truth of p_1 corresponds to the value of the least significant bit, and p_n corresponds to the value of the most significant one. Following the above description of $\mathbf{n}(w_{ax}) < \mathbf{n}(w_{bx})$, we define $[ax < bx]_j^j$ as:

$$[\text{ax} < \text{bx}]_j^j \stackrel{\text{def}}{=} \bigvee_{u \in [1, n]} (@_{\text{ax}}^j \neg p_u \wedge @_{\text{bx}}^j p_u \wedge \bigwedge_{v \in [u+1, n]} (@_{\text{ax}}^j p_v \Leftrightarrow @_{\text{bx}}^j p_v)).$$

Informally, $[\text{ax} < \text{bx}]_j^j$ ask for the existence of an index $u \in [0, n]$ such that the u -th most significant bit of $\mathbf{n}(w_{\text{ax}})$ is 0, the u -th most significant bit of $\mathbf{n}(w_{\text{bx}})$ is 1, and $\mathbf{n}(w_{\text{ax}})$ and $\mathbf{n}(w_{\text{bx}})$ agree on the $u - 1$ most significant bits. $[\text{bx} = \text{ax}+1]_1$ is defined in a similar way, again by following the arithmetical description given above:

$$[\text{bx} = \text{ax}+1]_1 \stackrel{\text{def}}{=} \bigvee_{u \in [1, n]} (@_{\text{ax}}^1 (\neg p_u \wedge \bigwedge_{v \in [1, u-1]} p_v) \wedge @_{\text{bx}}^1 (p_u \wedge \bigwedge_{v \in [1, u-1]} \neg p_v) \wedge \bigwedge_{v \in [u+1, n]} (@_{\text{ax}}^1 p_v \Leftrightarrow @_{\text{bx}}^1 p_v)).$$

As we can see, $[\text{bx} = \text{ax}+1]_1$ is essentially obtained from $[\text{ax} < \text{bx}]_j^j$ by also requiring every bit that is less significant than u to be 1 in $\mathbf{n}(w_{\text{ax}})$ and 0 in $\mathbf{n}(w_{\text{bx}})$.

Lemma 9.18. Let $\text{ax} \neq \text{bx} \in \text{Aux}$ and $j \geq 1$. Suppose $(\mathcal{K}, w) \models \text{init}(j) \wedge \text{fork}_j^j(\text{ax}, \text{bx})$. Let w_{ax} (resp. w_{bx}) be the world corresponding to the nominal ax (resp. bx) at depth j .

1. $(\mathcal{K}, w) \models [\text{ax} < \text{bx}]_j^j$ if and only if $\mathbf{n}(w_{\text{ax}}) < \mathbf{n}(w_{\text{bx}})$,
2. Let $j = 1$. $(\mathcal{K}, w) \models [\text{bx} = \text{ax}+1]_1$ if and only if $\mathbf{n}(w_{\text{ax}}) + 1 = \mathbf{n}(w_{\text{bx}})$.

The proof of Lemma 9.18 can be found in Appendix G.

Let us now move to the definitions of $\text{uniq}(1)$ and $\text{compl}(1)$. First, let us recall that a model satisfying $\text{type}(1)$ satisfies the formula aux and hence every t -node in $R(w)$ has two auxiliary children, one x -node and one y -node. In order to define $\text{uniq}(1)$, we use these Aux -children and rely on \ast to state that it is not possible to find a subforest of \mathcal{K} such that w has only two distinct children w_x and w_y corresponding to the nominals x and y , respectively, and such that $\mathbf{n}(w_x) = \mathbf{n}(w_y)$. In a sense, the operator \ast is here used to simulates a first-order quantification on x and y . Let $[x = y]_1^1 \stackrel{\text{def}}{=} \neg([x < y]_1^1 \vee [y < x]_1^1)$. Here is the formula $\text{uniq}(1)$:

$$\text{uniq}(1) \stackrel{\text{def}}{=} \neg(\top \ast (\text{fork}_1^1(x, y) \wedge [x = y]_1^1)).$$

Lemma 9.19. Suppose $(\mathcal{K}, w) \models \text{init}(1) \wedge \text{aux}$. $(\mathcal{K}, w) \models \text{uniq}(1)$ iff (\mathcal{K}, w) satisfies (uniq_1) .

Proof. Recall that (uniq_1) requires all t -children of w to encode a different number in $[0, 2^n - 1]$.
 (\Rightarrow) : Let us show the converse. Suppose there are two distinct t -nodes w_x and w_y encoding the same number. We show that $\text{uniq}(1)$ is not satisfied. Since $(\mathcal{K}, w) \models \text{init}(1) \wedge \text{aux}$, every world in $R(w)$ has exactly one x -child and exactly one (different) y -child. Consider the subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K} where $R'(w) = \{w_x, w_y\}$, $R'(w_x) = \{w_1\}$ and $R'(w_y) = \{w_2\}$, where w_1 is the only x -child of w_x , and w_2 is the only y -child of w_y . By Lemma 9.17, $(\mathcal{K}', w) \models \text{fork}_1^1(x, y)$. By Lemma 9.15, $\mathbf{n}(w_x) = \mathbf{n}(w_y)$ holds also with respect to \mathcal{K}' (notice that this corresponds to the base case, where numbers are encoded using the satisfaction of the atomic propositions p_1, \dots, p_n). By Lemma 9.18(1), this implies $(\mathcal{K}', w) \models [x = y]_1^1$. By definition, $(\mathcal{K}, w) \not\models \text{uniq}(1)$.

(\Leftarrow) : Again, we show the converse. Suppose $(\mathcal{K}, w) \not\models \text{uniq}(1)$, and thus there is a subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K} such that $(\mathcal{K}', w) \models \text{fork}_1^1(x, y) \wedge [x = y]_1^1$. Since the satisfaction of $\text{init}(1)$ is monotonic w.r.t. subforests, we have $(\mathcal{K}', w) \models \text{init}(1)$. We apply Lemma 9.17 and Lemma 9.18(1) in order to conclude that there are two distinct worlds w_x and w_y in $R'(w)$ such that $\mathbf{n}(w_x) = \mathbf{n}(w_y)$. By Lemma 9.15, w_x and w_y encode the same number also with respect to the pointed forest (\mathcal{K}, w) . Thus, (\mathcal{K}, w) does not satisfy (uniq_1) . \square

Let us now consider $\text{compl}(1)$. As done for $\text{uniq}(1)$, we rely on the auxiliary propositions x and y , and use the operator $*$ in order to simulate a first-order quantification on these atomic propositions. We state that it is not possible to find a subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ such that

1. w keeps all its children, i.e. $R(w) = R'(w)$, which in turn keep their y -children,
2. every t -children of w loses its x -child, with the exception of one world $w_x \in R(w)$ such that $\mathbf{n}(w_x) < 2^n - 1$,
3. it is not possible to find a subforest $\mathcal{K}'' = (\mathcal{W}, R'', \mathcal{V})$ of \mathcal{K}' such that $R''(w)$ contains only two t -children: w_x and w_y , where w_y corresponds to the nominal y and $\mathbf{n}(w_y) = \mathbf{n}(w_x) + 1$.

The definition of $\text{compl}(1)$ is given below, highlighting which part of the formula is responsible for the three conditions above:

$$\text{compl}(1) \stackrel{\text{def}}{=} \neg \left(\underbrace{\square \perp * ([t] \diamond y \wedge @_x^1 \neg 1_1)}_{(1)} \wedge \underbrace{\neg (\top * (\text{fork}_1^1(x, y) \wedge [y = x+1]_1))}_{(2)} \right),$$

where the formula 1_1 is defined as $\bigwedge_{i \in [1, n]} p_i$, reflecting the encoding of $2^n - 1$. Here, the subscript “1” in the formula 1_1 refers to the fact that we are treating the base case of $\text{compl}(j)$, with $j = 1$.

Lemma 9.20. Suppose $(\mathcal{K}, w) \models \text{init}(1) \wedge \text{aux}$. $(\mathcal{K}, w) \models \text{compl}(1)$ iff (\mathcal{K}, w) satisfies (compl_1) .

Proof. Recall that (compl_1) states that for every t -node $w_1 \in R(w)$, if $\mathbf{n}(w_1) < 2^n - 1$ then there is a t -node $w_2 \in R(w)$ such that $\mathbf{n}(w_2) = \mathbf{n}(w_1) + 1$.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{compl}(1)$. From the definition of $\text{compl}(1)$, this implies that for every subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K} , if $R'(w) = R(w)$ and $(\mathcal{K}', w) \models [t] \diamond y \wedge @_x^1 \neg 1_1$, then

$$(\mathcal{K}', w) \models \top * (\text{fork}_1^1(x, y) \wedge [y = x+1]_1).$$

Let us pick a t -node $w_x \in R'(w) = R(w)$ such that $\mathbf{n}(w_x) < 2^n - 1$. We show that there must be a world $w_y \in R'(w)$ such that $\mathbf{n}(w_y) = \mathbf{n}(w_x) + 1$.

Recall that, by $(\mathcal{K}, w) \models \text{init}(1) \wedge \text{aux}$, every t -child of w has exactly one x -child and one y -child. Consider the subforest $\mathcal{K}'' = (\mathcal{W}, R'', \mathcal{V})$ of \mathcal{K} such that $R'(w_x)$ contains both the x -child and y -child of w_x (w.r.t. \mathcal{K}), and otherwise for every $\bar{w} \in R(w)$ distinct from w_x , $R'(\bar{w}) = \{w_1\}$, where w_1 is the y -child of \bar{w} (w.r.t. \mathcal{K}). Therefore, in \mathcal{K}' , the world w_x corresponds to the nominal x for the depth 1, and every $w' \in R'(w)$ has a y -child. That is, $(\mathcal{K}', w) \models [t] \diamond y$. Moreover, $(\mathcal{K}, w) \models @_x^1 \neg 1_1$, directly from $\mathbf{n}(w_x) < 2^n - 1$ and Lemma 9.15 (here, recall that we are treating the base case, and thus the encoding of numbers uses the satisfaction of the atomic proposition p_1, \dots, p_n). From $(\mathcal{K}, w) \models \text{compl}(1)$, we conclude that $(\mathcal{K}', w) \models \top * (\text{fork}_1^1(x, y) \wedge [y = x+1]_1)$, which implies that there is a subforest $\mathcal{K}'' = (\mathcal{W}, R'', \mathcal{V})$ of \mathcal{K}' such that $(\mathcal{K}'', w) \models \text{fork}_1^1(x, y) \wedge [y = x+1]_1$. By Lemma 9.17 and Lemma 9.18(2), there is $w_y \in R''(w)$ such that $\mathbf{n}(w_y) = \mathbf{n}(w_x) + 1$. By Lemma 9.15, $\mathbf{n}(w_y) = \mathbf{n}(w_x) + 1$ holds also in \mathcal{K} . Thus, (\mathcal{K}, w) satisfies (compl_1) .

(\Leftarrow): Suppose that (\mathcal{K}, w) satisfies (compl_1) . *Ad absurdum* assume that $(\mathcal{K}, w) \not\models \text{compl}(1)$. Then, there is a subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K} such that $R'(w) = R(w)$ and

$$(\mathcal{K}', w) \models [t] \diamond y \wedge @_x^1 \neg 1_1 \wedge \neg (\top * (\text{fork}_1^1(x, y) \wedge [y = x+1]_1)).$$

Notice that this formula does not enforce x to be a nominal for the depth 1. However, from $(\mathcal{K}', w) \models @_x^1 \neg 1_1$, we deduce that there is at least one t -node w_x such that $(\mathcal{K}', w_x) \models \diamond x \wedge \neg 1_1$. Then, $\mathbf{n}(w_x) < 2^n - 1$ in \mathcal{K}' and, by Lemma 9.15, the same holds in \mathcal{K} . By hypothesis, there is a t -node w_y such that $\mathbf{n}(w_y) = \mathbf{n}(w_x) + 1$ (w.r.t. \mathcal{K}). Let us consider now the subforest $\mathcal{K}'' =$

$(\mathcal{W}, R'', \mathcal{V})$ of \mathcal{K}' where $R''(w) = \{w_x, w_y\}$, $R''(w_x) = \{w_1\}$ and $R''(w_y) = \{w_2\}$, where w_1 (resp. w_2) is the only x -child (resp. y -child) of w_x (resp. w_y). The existence of w_1 and w_2 is guaranteed directly from $(\mathcal{K}', w_x) \models \Diamond x \wedge \neg 1_1$ and $(\mathcal{K}', w) \models [t]\Diamond y$. By Lemma 9.17, $(\mathcal{K}'', w) \models \text{fork}_1^1(x, y)$. From $\mathbf{n}(w_y) = \mathbf{n}(w_x) + 1$ (in \mathcal{K} and by Lemma 9.15 and Lemma 9.18(2)), $(\mathcal{K}'', w) \models [y = x+1]_1$. However, this allows us to conclude that (\mathcal{K}', w) satisfies $\top * (\text{fork}_1^1(x, y) \wedge [y = x+1]_1)$: a contradiction. Thus, $(\mathcal{K}, w) \models \text{compl}(1)$. \square

As $\text{uniq}(1)$ and $\text{compl}(1)$ are now defined, so is $\text{type}(1)$. Its correctness, which follows directly from Lemma 9.19 and Lemma 9.20 together with the (straightforward) correctness of $\text{sub}(j)$, aux and $\text{zero}(1)$, is stated in the next lemma.

Lemma 9.21. Let $(\mathcal{K}, w) \models \text{init}(1)$. $(\mathcal{K}, w) \models \text{type}(1)$ if and only if (\mathcal{K}, w) satisfies (sub_1) , (zero_1) , (uniq_1) , (compl_1) and (aux) .

We conclude the section treating the base case of the construction of $\text{type}(j)$ by showing a model for $\text{init}(1) \wedge \text{type}(1)$, which will be useful when considering the tiling problem.

Lemma 9.22. $\text{init}(1) \wedge \text{type}(1)$ is satisfiable.

Proof (sketch). Consider the pointed forest (\mathcal{K}, w) such that $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and

1. $\mathcal{W} = \{w, w_0, w_0^x, w_0^y, w_1, w_1^x, w_1^y, \dots, w_{2^n-1}, w_{2^n-1}^x, w_{2^n-1}^y\}$,
2. $R = \{(w, w_i), (w_i^x, w_i^y) \mid i \in [0, 2^n - 1]\}$,
3. $\mathcal{V}(x) = \{w_0^x, \dots, w_{2^n-1}^x\}$, $\mathcal{V}(y) = \{w_0^y, \dots, w_{2^n-1}^y\}$ and for every $i \in [0, 2^n - 1]$ and $j \in [1, n]$, $w_i \in \mathcal{V}(p_j)$ if and only if the j -th bit in the binary encoding of i is 1.

It is easy to check that (\mathcal{K}, w) satisfies $\text{init}(1)$ as well as (sub_1) , (zero_1) , (uniq_1) , (compl_1) and (aux) . Thus, by Lemma 9.21, $(\mathcal{K}, w) \models \text{init}(1) \wedge \text{type}(1)$. \square

9.3.5 Construction of $\text{type}(j)$: formulae for the inductive step $1 \leq i < j$.

We now move to the inductive definitions of $\text{fork}_j^i(ax, bx)$, $[ax < bx]_j^i$, $[bx = ax+1]_j$, $\text{uniq}(j)$, $\text{compl}(j)$ and $\text{type}(j)$, where $1 \leq i < j$. So, during this section, let us fix the indices i and j such that $1 \leq i < j$ and, rely on an implicit induction hypothesis to prove that the formulae introduced are well-defined. More precisely, given a lexeme φ among $\text{fork}_j^i(ax, bx)$ and $[ax < bx]_j^i$, we write $d(\varphi)$ for the pair $(j, j-i) \in \mathbb{N}^2$ (recall that $i < j$), whereas for φ among $[bx = ax+1]_j$, $\text{uniq}(j)$, $\text{compl}(j)$ and $\text{type}(j)$, we write $d(\varphi)$ for (j, j) . We order the formulae using the well-founded lexicographic order $<_{lex}$ on \mathbb{N}^2 :

$$(j_1, i_1) <_{lex} (j_2, i_2) \text{ whenever } j_1 < j_2 \text{ or } (j_1 = j_2 \text{ and } i_1 < i_2).$$

Then, whenever defining one of these formulae φ , we assume by induction hypothesis that the formulae ψ (among the ones above) with $d(\psi) <_{lex} d(\varphi)$ are correctly defined. For instance, the formulae $\text{fork}_j^{i+1}(ax, bx)$ and $\text{type}(j-i)$ can be used in order to define $[ax < bx]_j^i$, and the formula $\text{fork}_j^1(ax, bx)$ can be used in the definition of $\text{uniq}(j)$.

As done for the base case, we start with the definition of $\text{fork}_j^i(ax, bx)$, which is quite simple:

$$\text{fork}_j^i(ax, bx) \stackrel{\text{def}}{=} \text{fork}_i^i(ax, bx) \wedge [t]^i \text{type}_{1sr}(j-i).$$

Clearly, the formula is well-defined, as $\text{fork}_i^i(ax, bx)$ was defined during the base case, whereas $\text{type}_{1sr}(j-i) = \text{type}(j-i) \wedge [t](\Diamond 1 \wedge \Diamond s \wedge \Diamond r)$ is defined by inductive hypothesis, since $j-i < j$.

The correctness of this formula is immediate, from its characterisation given during Section 9.3.3. Together with Lemma 9.17, we obtain the following result.

Lemma 9.23. Let $\text{ax} \neq \text{bx} \in \text{Aux}$, $1 \leq i \leq j$ and $(\mathcal{K}, w) \models \text{init}(j)$. $(\mathcal{K}, w) \models \text{fork}_j^i(\text{ax}, \text{bx})$ if and only if (\mathcal{K}, w) satisfies (fork₁)–(fork₄).

Let us now consider the formula $[\text{ax} < \text{bx}]_j^i$. Assuming that (\mathcal{K}, w) satisfies $\text{fork}_j^i(\text{ax}, \text{bx})$, we wish to express that $\mathbf{n}_{j-i}(w_{\text{ax}}) < \mathbf{n}_{j-i}(w_{\text{bx}})$ holds for the two distinct worlds w_{ax} and w_{bx} of $R^i(w)$ corresponding to the nominals ax and bx , respectively, for the depth i . As explained during Section 9.3.2, since $i < j$, these numbers $\mathbf{n}(w_{\text{ax}})$ are encoded (in binary) using the truth value of the atomic proposition val , on the t -children of w_{ax} and w_{bx} . To rely on the same arithmetical properties of binary numbers used to define $[\text{ax} < \text{bx}]_j^i$ (in the base case), we need to find two triples of set of worlds $P_{\text{ax}} = (L_{\text{ax}}, \{s_{\text{ax}}\}, R_{\text{ax}})$ and $P_{\text{bx}} = (L_{\text{bx}}, \{s_{\text{bx}}\}, R_{\text{bx}})$ such that:

(LSR) Given $b \in \{\text{ax}, \text{bx}\}$, $\{L_b, \{s_b\}, R_b\}$ is a partition of the set of t -children of w_b . Moreover, $\mathbf{n}_{j-(i+1)}(w_r) < \mathbf{n}_{j-(i+1)}(s_b)$ holds for every $w_r \in R_b$, and $\mathbf{n}_{j-(i+1)}(w_l) > \mathbf{n}_{j-(i+1)}(s_b)$ holds for every $w_l \in L_b$,

(LESS) P_{ax} and P_{bx} have the arithmetical properties of $<$:

(S) $\mathbf{n}_{j-(i+1)}(s_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(s_{\text{bx}})$, $(\mathcal{K}, s_{\text{ax}}) \models \neg \text{val}$ and $(\mathcal{K}, s_{\text{bx}}) \models \text{val}$,

(L) $(\mathcal{K}, l_{\text{ax}}) \models \text{val}$ iff $(\mathcal{K}, l_{\text{bx}}) \models \text{val}$, for every $l_{\text{ax}} \in L_{\text{ax}}$ and every $l_{\text{bx}} \in L_{\text{bx}}$ such that $\mathbf{n}_{j-(i+1)}(l_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(l_{\text{bx}})$.

It is important to notice that these conditions essentially revolve around the numbers encoded by t -children, which will be compared using the already defined (by inductive reasoning) formulae $[\text{ax} < \text{bx}]_j^{i+1}$. Since the semantics of $[\text{ax} < \text{bx}]_j^i$ is given under the hypothesis that $(\mathcal{K}, w) \models \text{fork}_j^i(\text{ax}, \text{bx})$, by (fork₄) we can assume that every child of w_{ax} and w_{bx} has all the possible Aux -children. Then, we rely on the auxiliary propositions **l**, **s** and **r** in order to describe the partitions P_{ax} and P_{bx} , and mimic the reasoning done in (LSR) and (LESS).

First of all, with the help of **l**, **s** and **r**, we define a formula $\text{lsr}(k)$ highlighting the partitions $\{L_b, \{s_b\}, R_b\}$ described in (LSR). In particular, given (\mathcal{K}, w) where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, and $k \geq 1$, $\text{lsr}(k)$ shall be satisfied whenever:

(lsr₁) (\mathcal{K}, w) satisfies $\text{type}(k)$,

(lsr₂) All t -children of w have exactly one $\{\text{l}, \text{s}, \text{r}\}$ -child. Only one, say w_s , has a **s**-child,

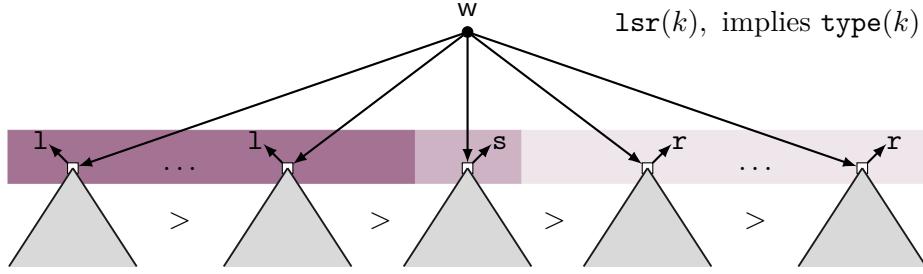
(lsr₃) $\mathbf{n}_{k-1}(w') > \mathbf{n}_{k-1}(w_s)$ (resp. $\mathbf{n}_{k-1}(w') < \mathbf{n}_{k-1}(w_s)$), for every t -child w' of w having a **l**-child (resp. **r**-child).

Notice that, as $\text{lsr}(k)$ is based on the satisfaction of $\text{type}(k)$, it is well-defined (by inductive reasoning) as soon as $k < j$. Figure 9.14 schematises a pointed forest satisfying $\text{lsr}(k)$.

The definition of $\text{lsr}(k)$ follows closely its specification:

$$\begin{aligned} \text{lsr}(k) \stackrel{\text{def}}{=} & \overbrace{\text{type}(k) \wedge [t] \diamondsuit_{=1} (\text{l} \vee \text{s} \vee \text{r}) \wedge \text{nom}_1(\text{s})}^{\text{(lsr}_1\text{)}} \\ & \wedge \underbrace{\neg(\top * (\text{fork}_k^1(\text{s}, \text{l}) \wedge \neg[\text{s} < \text{l}]_k^1)) \wedge \neg(\top * (\text{fork}_k^1(\text{s}, \text{r}) \wedge \neg[\text{r} < \text{s}]_k^1))}_{\text{(lsr}_3\text{)}}. \end{aligned}$$

Lemma 9.24. Let $k < j$ and $(\mathcal{K}, w) \models \text{init}(k)$. $(\mathcal{K}, w) \models \text{lsr}(k)$ iff (\mathcal{K}, w) satisfies (lsr₁)–(lsr₃).

Figure 9.14: Shape of a pointed forest satisfying $\text{lsr}(k)$.

Proof. The properties (lsr_1) and (lsr_2) are trivially characterised by the formulae $\text{type}(k)$ and $[t]\Diamond_{=1}(1 \vee s \vee r) \wedge \text{nom}_1(s)$, respectively. In what follows, we focus on the property (lsr_3) .

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{lsr}(k)$. From $\text{nom}_1(s)$ there is exactly one world $w_s \in R(w)$ having a s -child. *Ad absurdum*, assume that (lsr_3) is not satisfied, and thus one of the following holds:

1. there is a t -child w' of w that has a 1-child and $\mathbf{n}_{k-1}(w') < \mathbf{n}_{k-1}(w_s)$, or
2. there is a t -child w' of w that has a r -child and $\mathbf{n}_{k-1}(w') > \mathbf{n}_{k-1}(w_s)$.

We consider the first case, and derive a contradiction with $(\mathcal{K}, w) \not\models \top * (\text{fork}_k^1(s, 1) \wedge \neg[s < 1]_k^1)$. The second case is analogous and in contradiction with $(\mathcal{K}, w) \not\models \top * (\text{fork}_k^1(s, r) \wedge \neg[r < s]_j^k)$. Thus, we are able to show that (\mathcal{K}, w) satisfies (lsr_3) . Let w' be the t -child of w having a 1-child and being such that $\mathbf{n}_{k-1}(w') < \mathbf{n}_{k-1}(w_s)$. Let us consider the Kripke-style finite forest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ obtained from \mathcal{K}' by removing all the arrows from w to worlds different from w' and w_s , i.e. $R' = R \setminus \{(w, w'') \in R \mid w'' \notin \{w_s, w'\}\}$. So, $\mathcal{K}' \subseteq \mathcal{K}$. Let us show that $(\mathcal{K}, w) \models \text{fork}_k^1(s, 1)$. In \mathcal{K}' , the worlds w_s and w' only have one Aux-child: the one of w_s satisfies s whereas the one of w' satisfies 1 . As $R'(w) = \{w_s, w'\}$, this allows us to conclude that the properties (fork_1) , (fork_2) and (fork_3) of $\text{fork}_k^1(s, 1)$ are satisfied. It remains to show (fork_4) , i.e. that (\mathcal{K}', w_s) and (\mathcal{K}', w') satisfy $\text{type}_{\text{lsr}}(k-1) = \text{type}(k-1) \wedge [t](\Diamond 1 \wedge \Diamond s \wedge \Diamond r)$, which holds directly from $(\mathcal{K}, w) \models \text{type}(k)$, thanks to the properties (sub_j) and (aux) . Now, from $\mathcal{K}' \subseteq \mathcal{K}$ together with $(\mathcal{K}, w_s) \models \text{type}(k-1)$ and $(\mathcal{K}, w') \models \text{type}(k-1)$, by Lemma 9.15 we conclude that w_s and w' keep encoding their respective number even after \mathcal{K} is modified in \mathcal{K}' . Therefore, in \mathcal{K}' , we have $\mathbf{n}_{k-1}(w') < \mathbf{n}_{k-1}(w_s)$. However, by relying on Lemma 9.15, this allows us to conclude that $(\mathcal{K}', w) \models \text{fork}_k^1(s, 1) \wedge \neg[s < 1]_k^1$ (notice that $k < j$, and thus $[s < 1]_k^1$ is well-defined), which contradicts $(\mathcal{K}, w) \not\models \top * (\text{fork}_k^1(s, 1) \wedge \neg[s < 1]_k^1)$, as $\mathcal{K}' \subseteq \mathcal{K}$. We conclude that (\mathcal{K}, w) satisfies (lsr_3) .

(\Leftarrow): This direction is analogous. Suppose that (\mathcal{K}, w) satisfies (lsr_1) , (lsr_2) and (lsr_3) . As (lsr_1) entails $(\mathcal{K}, w) \models \text{type}(k)$ and (lsr_2) entails $(\mathcal{K}, w) \models [t]\Diamond_{=1}(1 \vee s \vee r) \wedge \text{nom}_1(s)$, we focus on proving that

$$(\mathcal{K}, w) \models \neg(\top * (\text{fork}_k^1(s, 1) \wedge \neg[s < 1]_k^1)) \wedge \neg(\top * (\text{fork}_k^1(s, r) \wedge \neg[r < s]_k^1)).$$

We show (by contradiction) that $(\mathcal{K}, w) \models \neg(\top * (\text{fork}_k^1(s, 1) \wedge \neg[s < 1]_k^1))$. *Ad absurdum*, suppose $(\mathcal{K}, w) \models \top * (\text{fork}_k^1(s, 1) \wedge \neg[s < 1]_k^1)$. Thus, there is a subforest $\mathcal{K}' \subseteq \mathcal{K}$ such that $(\mathcal{K}, w) \models \text{fork}_k^1(s, 1) \wedge \neg[s < 1]_k^1$. From $(\mathcal{K}', w) \models \text{fork}_k^1(s, 1)$, in \mathcal{K}' there are two t -children w_s and w' of w that correspond to the nominals (for the depth 1) s and 1 , respectively. Moreover, (\mathcal{K}', w_s) and (\mathcal{K}', w') satisfy $\text{type}(k-1)$, and, by $(\mathcal{K}', w) \models [s < r]_k^1$, $\mathbf{n}_{k-1}(w') < \mathbf{n}_{k-1}(w_s)$ (w.r.t. \mathcal{K}'). However, from Lemma 9.15, this implies that $\mathbf{n}_{k-1}(w') < \mathbf{n}_{k-1}(w_s)$ holds also with respect to \mathcal{K} , in contradiction with (lsr_3) . Therefore, $(\mathcal{K}, w) \models \neg(\top * (\text{fork}_k^1(s, 1) \wedge \neg[s < 1]_k^1))$.

Analogously, one can show that $(\mathcal{K}, w) \models \neg(\top * (\text{fork}_k^1(s, r) \wedge \neg[r < s]_k^1))$, which concludes the proof of $(\mathcal{K}, w) \models \text{lsr}(k)$. \square

Thanks to the formula $\text{lsr}(k)$, in order to define the formula $[\text{ax} < \text{bx}]_j^i$ it is now sufficient to encode the arithmetical properties described by (LESS). In particular, below we define the two formulae $S_j^i(\text{ax}, \text{bx})$ and $L_j^i(\text{ax}, \text{bx})$ that essentially characterise the properties (S) and (L) of (LESS). In their definition, $[\text{ax} = \text{bx}]_j^{i+1}$ stands for $\neg([\text{ax} < \text{bx}]_j^{i+1} \vee [\text{bx} < \text{ax}]_j^{i+1})$, which is well-defined (by inductive reasoning) as $j - (i + 1) < j - i$.

$$\begin{aligned} S_j^i(\text{ax}, \text{bx}) &\stackrel{\text{def}}{=} \top * (\text{fork}_j^{i+1}(x, y) \wedge @_{\text{ax}}^i(t)(\Diamond s \wedge \Diamond x) \wedge @_{\text{bx}}^i(t)(\Diamond s \wedge \Diamond y) \\ &\quad \wedge [x = y]_j^{i+1} \wedge @_x^{i+1}\neg\text{val} \wedge @_y^{i+1}\text{val}), \end{aligned}$$

$$\begin{aligned} L_j^i(\text{ax}, \text{bx}) &\stackrel{\text{def}}{=} \neg(\top * (\text{fork}_j^{i+1}(x, y) \wedge @_{\text{ax}}^i(t)(\Diamond 1 \wedge \Diamond x) \wedge @_{\text{bx}}^i(t)(\Diamond 1 \wedge \Diamond y) \\ &\quad \wedge [x = y]_j^{i+1} \wedge \neg(@_x^{i+1}\text{val} \Leftrightarrow @_y^{i+1}\text{val}))). \end{aligned}$$

By inductive reasoning, $S_j^i(\text{ax}, \text{bx})$ and $L_j^i(\text{ax}, \text{bx})$ are well-defined. Thanks to these two formulae $[\text{ax} < \text{bx}]_j^i$ is simply defined as:

$$[\text{ax} < \text{bx}]_j^i \stackrel{\text{def}}{=} \top * (\text{nom}_i(\text{ax} \neq \text{bx}) \wedge [t]^i\text{lsr}(j - i) \wedge S_j^i(\text{ax}, \text{bx}) \wedge L_j^i(\text{ax}, \text{bx})).$$

Informally, given a pointed forest $(\mathcal{K}, w) \models \text{fork}_j^i(\text{ax}, \text{bx})$, the formula $[\text{ax} < \text{bx}]_j^i$ asks to consider the subforest $\mathcal{K}' \subseteq \mathcal{K}$ where the worlds w_{ax} and w_{bx} corresponding to the nominal ax and bx , respectively, at depth i (which exist thanks to the satisfaction of $\text{fork}_j^i(\text{ax}, \text{bx})$), satisfy $\text{lsr}(j - i)$, and $S_j^i(\text{ax}, \text{bx})$ and $L_j^i(\text{ax}, \text{bx})$ are also satisfied. Let us briefly explain the formula $S_j^i(\text{ax}, \text{bx})$, and how it describes the property (S) of (LESS). A similar analysis can be performed for $L_j^i(\text{ax}, \text{bx})$ and (L). Since $[t]^i\text{lsr}(j - i)$ is satisfied, the t -children of w_{ax} and w_{bx} are partitioned into $P_{\text{ax}} = (L_{\text{ax}}, \{s_{\text{ax}}\}, R_{\text{ax}})$ and $P_{\text{bx}} = (L_{\text{bx}}, \{s_{\text{bx}}\}, R_{\text{bx}})$, respectively, following which type of $\{1, s, r\}$ -child they have. Moreover, as $\text{lsr}(j - i)$ implies $\text{type}(j - i)$, from (aux) all these children have one x -child and one y -child. Then, $S_j^i(\text{ax}, \text{bx})$ requires to create a fork such that one path of length $i + 1$ ends in s_{ax} , whereas the other ends in s_{bx} , and where s_{ax} (resp. s_{bx}) corresponds to the nominal x (resp. y). Following the description of (S), $S_j^i(\text{ax}, \text{bx})$ states that $\mathbf{n}(s_{\text{ax}})$ should be equal to $\mathbf{n}(s_{\text{bx}})$, by relying on the formula $[x = y]_j^{i+1}$, and that $(\mathcal{K}, s_{\text{ax}}) \models \neg\text{val}$ and $(\mathcal{K}, s_{\text{bx}}) \models \text{val}$, by relying on the formulae $@_x^{i+1}\neg\text{val}$ and $@_y^{i+1}\text{val}$.

Lemma 9.25. Let (\mathcal{K}, w) be a pointed forest satisfying $\text{init}(j)$. Let $\text{ax} \neq \text{bx} \in \text{Aux}$. Suppose $(\mathcal{K}, w) \models \text{fork}_j^i(\text{ax}, \text{bx})$, and let w_{ax} and w_{bx} be the worlds corresponding to the nominals ax and bx , respectively, for the depth i . $(\mathcal{K}, w) \models [\text{ax} < \text{bx}]_j^i$ if and only if $\mathbf{n}_{j-i}(w_{\text{ax}}) < \mathbf{n}_{j-i}(w_{\text{bx}})$.

In order to prove Lemma 9.25, we first establish the correctness of $S_j^i(\text{ax}, \text{bx})$ and $L_j^i(\text{ax}, \text{bx})$. Following the definition $[\text{ax} < \text{bx}]_j^i$, we work under the assumption that (\mathcal{K}, w) satisfies $\text{init}(j)$, $\text{nom}_i(\text{ax} \neq \text{bx})$ and $[t]^i\text{lsr}(j - i)$. As already stated, this implies that there are two worlds w_{ax} and w_{bx} that correspond to the nominals (for the depth i) ax and bx , respectively, and that satisfy $\text{lsr}(j - i)$. By Lemma 9.24, this means that the t -children of w_{ax} and w_{bx} are partitioned into $P_{\text{ax}} = (L_{\text{ax}}, \{s_{\text{ax}}\}, R_{\text{ax}})$ and $P_{\text{bx}} = (L_{\text{bx}}, \{s_{\text{bx}}\}, R_{\text{bx}})$, respectively, following their only $\{1, s, r\}$ -child, so that (LSR) holds. In particular, the worlds in L_{ax} and L_{bx} have an 1-child, s_{ax} and s_{bx} have an s -child, and the worlds in R_{ax} and R_{bx} have an r -child. For conciseness, in the following lemma regarding the correctness of $S_j^i(\text{ax}, \text{bx})$ and $L_j^i(\text{ax}, \text{bx})$, we refer directly to these objects.

Lemma 9.26. Let (\mathcal{K}, w) be a finite forest satisfying $\text{init}(j) \wedge \text{nom}_i(\text{ax} \neq \text{bx}) \wedge [t]^i\text{lsr}(j - i)$.

1. $(\mathcal{K}, w) \models S_j^i(ax, bx)$ iff $\mathbf{n}_{j-(i+1)}(s_{ax}) = \mathbf{n}_{j-(i+1)}(s_{bx})$, $(\mathcal{K}, s_{ax}) \models \neg \text{val}$ and $(\mathcal{K}, s_{bx}) \models \text{val}$.
2. $(\mathcal{K}, w) \models L_j^i(ax, bx)$ iff for all $l_{ax} \in L_{ax}$ and all $l_{bx} \in L_{bx}$, if $\mathbf{n}_{j-(i+1)}(l_{ax}) = \mathbf{n}_{j-(i+1)}(l_{bx})$, then $(\mathcal{K}, l_{ax}) \models \text{val}$ iff $(\mathcal{K}, l_{bx}) \models \text{val}$.

Proof of Lemma 9.26(1). (\Rightarrow): Suppose $(\mathcal{K}, w) \models S_j^i(ax, bx)$, and therefore there is a subforest $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V})$ of \mathcal{K} that satisfies

$$\mathbf{fork}_j^{i+1}(x, y) \wedge @_x^i \langle t \rangle (\Diamond s \wedge \Diamond x) \wedge @_x^i \langle t \rangle (\Diamond s \wedge \Diamond y) \wedge [x=y]_j^{i+1} \wedge @_x^{i+1} \neg \text{val} \wedge @_y^{i+1} \text{val}.$$

From $(\mathcal{K}', w) \models @_x^i \langle t \rangle (\Diamond s \wedge \Diamond x)$ we conclude that there is a world $w' \in R'^i(w)$ that has an ax -child as well as a t -child that satisfies $\Diamond s \wedge \Diamond x$. From $(\mathcal{K}, w) \models \text{nom}_i(ax \neq bx)$, there is only one world, i.e. w_{ax} , in $R^i(w)$ that has an ax -child. By $\mathcal{K}' \subseteq \mathcal{K}$, we conclude that $w' = w_{ax}$. From $(\mathcal{K}, w) \models [t]^i \text{lsr}(j-i)$, the only children of w_{ax} that has an s -child is s_{ax} (as in the statement of the lemma). Similarly, by $(\mathcal{K}', w) \models @_x^i \langle t \rangle (\Diamond s \wedge \Diamond y)$, we conclude that the t -child s_{bx} of $w_{bx} \in R^i(w)$ belongs to $R'^{i+1}(w)$ and it is such that $(\mathcal{K}', s_{bx}) \models \Diamond y$. Remember: s_{bx} is the only child of w_{bx} that has an s -child. From $(\mathcal{K}, w) \models \mathbf{fork}_j^{i+1}(x, y)$, s_{ax} and s_{bx} are the only t -nodes in $R'^{i+1}(w)$, and moreover (\mathcal{K}', s_{ax}) and (\mathcal{K}', s_{bx}) satisfy $\text{type}_{\text{lsr}}(j-(i+1))$. From $(\mathcal{K}', w) \models [x=y]_j^{i+1}$ (by induction hypothesis, the formula $[x=y]_j^{i+1}$ is well-defined), we conclude that $\mathbf{n}_{j-(i+1)}(s_{ax}) = \mathbf{n}_{j-(i+1)}(s_{bx})$ holds with respect to \mathcal{K}' . As (\mathcal{K}', s_{ax}) and (\mathcal{K}', s_{bx}) satisfy $\text{type}_{\text{lsr}}(j-(i+1))$, by Lemma 9.15, $\mathbf{n}_{j-(i+1)}(s_{ax}) = \mathbf{n}_{j-(i+1)}(s_{bx})$ holds with respect to \mathcal{K} . Lastly, from $(\mathcal{K}', w) \models @_x^{i+1} \neg \text{val} \wedge @_y^{i+1} \text{val}$ and $\mathcal{K}' \subseteq \mathcal{K}$, we conclude that $(\mathcal{K}, s_{ax}) \models \neg \text{val}$ and $(\mathcal{K}, s_{bx}) \models \text{val}$.

(\Leftarrow): Suppose that $\mathbf{n}_{j-(i+1)}(s_{ax}) = \mathbf{n}_{j-(i+1)}(s_{bx})$, $(\mathcal{K}, s_{ax}) \models \neg \text{val}$ and $(\mathcal{K}, s_{bx}) \models \text{val}$, where we recall that s_{ax} is the only s -child of w_{ax} whereas s_{bx} is the only s -child of w_{bx} , the worlds w_{ax} and w_{bx} being the worlds corresponding to the nominals (for the depth i) ax and bx , respectively. From $(\mathcal{K}, w) \models [t]^i \text{lsr}(j-i)$, both (\mathcal{K}, w_{ax}) and (\mathcal{K}, w_{bx}) satisfy $\text{type}(j-i)$, and so by (aux) both s_{ax} and s_{bx} have one x -child and one y -child. Let us consider the Kripke-style finite forest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ with accessibility relation R' defined as

$$R' \setminus \left(\begin{array}{l} \left\{ (w', w'') \in R \mid \text{there is } k \in [0, i-1] \text{ s.t. } w' \in R^k(w), \text{ and } \{w_{ax}, w_{bx}\} \cap R^*(w'') = \emptyset \right\} \\ \cup \{(w_{ax}, w') \in R \mid w' \neq s_{ax}\} \cup \{(w_{bx}, w') \in R \mid w' \neq s_{bx}\} \cup \{(s_{ax}, w_y), (s_{bx}, w_x)\} \end{array} \right).$$

where w_y is the only y -child of s_{ax} and w_x is the only x -child of s_{bx} . Informally, \mathcal{K}' is obtained by removing all the t -nodes that are reachable from w in at most $i-1$ steps and that do not reach neither w_{ax} nor w_{bx} (first line of the definition of R'), together with all the t -children of w_{ax} and w_{bx} , with the exception of s_{ax} and s_{bx} . Moreover, the only y -child of s_{ax} and the only x -child of s_{bx} are also removed. Let us show that $(\mathcal{K}', w) \models \mathbf{fork}_j^{i+1}(x, y)$. From the first line of the definition of R' , $R'(w)$ only has two t -children: one ancestor w_U of w_{ax} and one ancestor w_D of w_{bx} . So, (fork₂) is satisfied. Moreover, again from the first line of the definition, for every $k \in [1, i-1]$, both $R^k(w_U)$ and $R^k(w_D)$ contain exactly one t -child. This is also true for $k = i$, as s_{ax} is the only t -node in $R'(w_{ax})$, and s_{bx} is the only t -node in $R'(w_{bx})$. Therefore, (fork₃) is satisfied. Notice that this implies that s_{ax} and s_{bx} are the only t -nodes in $R^{i+1}(w)$. Therefore, as (in \mathcal{K}') s_{ax} has one x -child and no y -children, whereas s_{bx} has one y -child and no x -children, we conclude that x and y are nominals for the depth $i+1$, as required by (fork₁). Lastly, in \mathcal{K}' , the subtrees rooted in s_{ax} and s_{bx} are obtained by only removing $\{x, y\}$ -children of these two worlds. By Lemma 9.23, to conclude that $(\mathcal{K}', w) \models \mathbf{fork}_j^{i+1}(x, y)$ it is now sufficient to show that (fork₄) holds, i.e. that both (\mathcal{K}', s_{ax}) and (\mathcal{K}', s_{bx}) satisfy $\text{type}_{\text{lsr}}(j-(i+1))$. Let us discuss why $(\mathcal{K}', s_{ax}) \models \text{type}_{\text{lsr}}(j-(i+1))$, the case of (\mathcal{K}', s_{bx}) being analogous. From $(\mathcal{K}, w) \models [t]^i \text{lsr}(j-i)$ we have that $(\mathcal{K}, w_{ax}) \models \text{lsr}(j-i)$.

By definition, this implies that $(\mathcal{K}, w_{ax}) \models \text{type}(j - i)$, which in turn implies (by (sub_j)) that $(\mathcal{K}, s_{ax}) \models \text{type}(j - (i + 1))$ and that every t -children of s_{ax} has one $\{1, s, r\}$ -children for every proposition among $1, s$ and r (by (aux)). All these children are kept in \mathcal{K}' , which allows us to conclude that $(\mathcal{K}, s_{ax}) \models \text{type}_{1sr}(j - (i + 1))$ holds. This ends the proof of $(\mathcal{K}', w) \models \text{fork}_j^{i+1}(x, y)$. As already stated, in \mathcal{K}' the world s_{ax} has one x -child and no y -children, whereas s_{bx} has one y -child and no x -children. We derive $(\mathcal{K}', w) \models @_ax^i(t)(\Diamond s \wedge \Diamond x) \wedge @_bx^i(t)(\Diamond s \wedge \Diamond y)$. From the hypothesis $(\mathcal{K}, s_{ax}) \models \neg \text{val}$ and $(\mathcal{K}, s_{bx}) \models \text{val}$, we have $(\mathcal{K}', w) \models @_x^{i+1} \neg \text{val} \wedge @_y^{i+1} \text{val}$. Lastly, since both (\mathcal{K}', s_{ax}) and (\mathcal{K}', s_{bx}) satisfy $\text{type}_{1sr}(j - (i + 1))$, by Lemma 9.15, we conclude that s_{ax} and s_{bx} encode in \mathcal{K}' the same numbers that they encode in \mathcal{K} . Therefore, the hypothesis $\mathbf{n}_{j-(i+1)}(s_{ax}) = \mathbf{n}_{j-(i+1)}(s_{bx})$ lifts from \mathcal{K} to \mathcal{K}' , and allows us to derive $(\mathcal{K}', w) \models [x = y]_j^{i+1}$. From $\mathcal{K}' \subseteq \mathcal{K}$, we derive $(\mathcal{K}, w) \models S_j^i(ax, bx)$. \square

Proof of Lemma 9.26(2). (\Rightarrow): Suppose $(\mathcal{K}, w) \models L_j^i(ax, bx)$, and therefore for every $\mathcal{K}' \subseteq \mathcal{K}$, if $(\mathcal{K}', w) \models \text{fork}_j^{i+1}(x, y) \wedge @_ax^i(t)(\Diamond 1 \wedge \Diamond x) \wedge @_bx^i(t)(\Diamond 1 \wedge \Diamond y) \wedge [x = y]_j^{i+1}$, then (\mathcal{K}', w) satisfies $@_x^{i+1} \text{val} \Leftrightarrow @_y^{i+1} \text{val}$. Let $l_{ax} \in L_{ax}$ and $l_{bx} \in L_{bx}$ such that $\mathbf{n}_{j-(i+1)}(l_{ax}) = \mathbf{n}_{j-(i+1)}(l_{bx})$. We recall that, according to $1sr(j - i)$, l_{ax} is a t -child of w_{ax} that has exactly one $\{1, s, r\}$ -child, which satisfies 1. Similarly, l_{bx} is a t -child of w_{bx} that has exactly one $\{1, s, r\}$ -child, which satisfies 1. We show that $(\mathcal{K}, l_{ax}) \models \text{val}$ iff $(\mathcal{K}, l_{bx}) \models \text{val}$.

Essentially, we start by considering the same Kripke-style finite forest \mathcal{K}' used in the right to left direction of Lemma 9.26(1), the only difference being that we focus on l_{ax} and l_{bx} instead of s_{ax} and s_{bx} . In particular, from $(\mathcal{K}, w) \models [t]^i 1sr(j - i)$, both (\mathcal{K}, w_{ax}) and (\mathcal{K}, w_{bx}) satisfy $\text{type}(j - i)$, and so by (aux) both l_{ax} and l_{bx} have one x -child and one y -child. We define the Kripke-style finite forest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ where R' is defined as

$$R' \setminus \left(\left\{ (w', w'') \in R \mid \text{there is } k \in [0, i - 1] \text{ s.t. } w' \in R^k(w), \text{ and } \{w_{ax}, w_{bx}\} \cap R^*(w'') = \emptyset \right\} \cup \{(w_{ax}, w') \in R \mid w' \neq l_{ax}\} \cup \{(w_{bx}, w') \in R \mid w' \neq l_{bx}\} \cup \{(l_{ax}, w_y), (l_{bx}, w_x)\} \right).$$

where w_y is the only y -child of l_{ax} and w_x is the only x -child of l_{bx} . Notice the similarities between (\mathcal{K}', w) and the homonymous pointed forest considered in the proof of the right to left direction of Lemma 9.26(1). With no surprises, following that proof, one can show that $(\mathcal{K}', w) \models \text{fork}_j^{i+1}(x, y) \wedge @_ax^i(t)(\Diamond 1 \wedge \Diamond x) \wedge @_bx^i(t)(\Diamond 1 \wedge \Diamond y) \wedge [x = y]_j^{i+1}$, where l_{ax} corresponds to the nominal x for the depth $i + 1$, whereas l_{bx} corresponds to the nominal y for the depth $i + 1$. Form $(\mathcal{K}, w) \models L_j^i(ax, bx)$, we derive that $(\mathcal{K}', w) \models @_x^{i+1} \text{val} \Leftrightarrow @_y^{i+1} \text{val}$, which in turn implies $(\mathcal{K}, l_{ax}) \models \text{val}$ iff $(\mathcal{K}, l_{bx}) \models \text{val}$.

(\Leftarrow): Let us assume now that for all $l_{ax} \in L_{ax}$ and all $l_{bx} \in L_{bx}$, if $\mathbf{n}_{j-(i+1)}(l_{ax}) = \mathbf{n}_{j-(i+1)}(l_{bx})$, then $(\mathcal{K}, l_{ax}) \models \text{val}$ iff $(\mathcal{K}, l_{bx}) \models \text{val}$. Ad absurdum, assume that $(\mathcal{K}, w) \not\models L_j^i(ax, bx)$, and therefore there is a subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K} that satisfies

$$\text{fork}_j^{i+1}(x, y) \wedge @_ax^i(t)(\Diamond 1 \wedge \Diamond x) \wedge @_bx^i(t)(\Diamond 1 \wedge \Diamond y) \wedge [x = y]_j^{i+1} \wedge \neg(@_x^{i+1} \text{val} \Leftrightarrow @_y^{i+1} \text{val}).$$

Similarly to the left to right direction of Lemma 9.26(1), from $(\mathcal{K}', w) \models @_ax^i(t)(\Diamond 1 \wedge \Diamond x)$ we conclude that there is a world $w' \in R'^i(w)$ that has a ax -child as well as a t -child l_{ax} that satisfies $\Diamond 1 \wedge \Diamond x$. Form $(\mathcal{K}, w) \models \text{nom}_i(ax \neq bx)$, there is only one world, i.e. w_{ax} , in $R^i(w)$ that has an ax -child. By $\mathcal{K}' \subseteq \mathcal{K}$, we conclude that $w' = w_{ax}$. From $(\mathcal{K}, w) \models [t]^i 1sr(j - i)$, and since $(\mathcal{K}, l_{ax}) \models \Diamond 1$, we conclude that l_{ax} belongs to L_{ax} . Similarly, from $(\mathcal{K}', w) \models @_bx^i(t)(\Diamond 1 \wedge \Diamond y)$, we conclude that one t -child l_{bx} of $w_{bx} \in R^i(w)$ having one 1 -child belongs to $R'^{i+1}(w)$ and it is such that $(\mathcal{K}', w) \models \Diamond y$. From $(\mathcal{K}, w) \models \text{fork}_j^{i+1}(x, y)$, l_{ax} and l_{bx} are the only t -nodes in $R'^{i+1}(w)$, and moreover (\mathcal{K}', l_{ax}) and (\mathcal{K}', l_{bx}) satisfy $\text{type}_{1sr}(j - (i + 1))$. From $(\mathcal{K}', w) \models [x = y]_j^{i+1}$, where the formula $[x = y]_j^{i+1}$ is well-defined (by induction hypothesis), this allows us to conclude

that, in \mathcal{K}' , $\mathbf{n}_{j-(i+1)}(l_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(l_{\text{bx}})$. As $(\mathcal{K}', l_{\text{ax}})$ and $(\mathcal{K}', l_{\text{bx}})$ satisfy $\text{type}_{\text{lsr}}(j - (i + 1))$, by Lemma 9.15, $\mathbf{n}_{j-(i+1)}(l_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(l_{\text{bx}})$ holds also with respect to \mathcal{K} . Therefore, by hypothesis, we derive that $(\mathcal{K}, l_{\text{ax}}) \models \text{val}$ iff $(\mathcal{K}, l_{\text{bx}}) \models \text{val}$. However, as $\mathcal{K}' \subseteq \mathcal{K}$, this implies that $(\mathcal{K}', l_{\text{ax}}) \models \text{val}$ iff $(\mathcal{K}', l_{\text{bx}}) \models \text{val}$, which contradicts $(\mathcal{K}, w) \models \neg(@_x^{i+1}\text{val} \Leftrightarrow @_y^{i+1}\text{val})$. We conclude that $(\mathcal{K}, w) \models L_j^i(\text{ax}, \text{bx})$. \square

We are now ready to prove Lemma 9.25.

Proof of Lemma 9.25. Let (\mathcal{K}, w) be a pointed forest satisfying $\text{init}(j) \wedge \text{fork}_j^i(\text{ax}, \text{bx})$, and let w_{ax} and w_{bx} be the worlds corresponding to the nominals ax and bx , for the depth i . The existence of w_{ax} and w_{bx} is guaranteed by Lemma 9.23.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models [\text{ax} < \text{bx}]_j^i$, and therefore there is a subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K}' that satisfies $\text{nom}_i(\text{ax} \neq \text{bx}) \wedge [t]^i \text{lsr}(j - i) \wedge S_j^i(\text{ax}, \text{bx}) \wedge L_j^i(\text{ax}, \text{bx})$. From $(\mathcal{K}', w) \models \text{nom}_i(\text{ax} \neq \text{bx})$, we derive that the worlds w_{ax} and w_{bx} belong to $R^i(w)$, and by $(\mathcal{K}', w) \models [t]^i \text{lsr}(j - i)$ we have that $(\mathcal{K}', w_{\text{ax}})$ and $(\mathcal{K}', w_{\text{bx}})$ satisfy $\text{lsr}(j - i)$. By Lemma 9.24, this means that the t -children of w_{ax} and w_{bx} are partitioned into $P_{\text{ax}} = (L_{\text{ax}}, \{s_{\text{ax}}\}, R_{\text{ax}})$ and $P_{\text{bx}} = (L_{\text{bx}}, \{s_{\text{bx}}\}, R_{\text{bx}})$, respectively, following their only $\{1, s, r\}$ -child, so that (LSR) holds. In particular, the worlds in L_{ax} and L_{bx} have an 1-child, s_{ax} and s_{bx} have an s -child, and the worlds in R_{ax} and R_{bx} have an r -child. Moreover (see (lsr₃)), for every $l_{\text{ax}} \in L_{\text{ax}}$, $\mathbf{n}_{j-(i+1)}(l_{\text{ax}}) > \mathbf{n}_{j-(i+1)}(s_{\text{ax}})$, and similarly for every $l_{\text{bx}} \in L_{\text{bx}}$, $\mathbf{n}_{j-(i+1)}(l_{\text{bx}}) > \mathbf{n}_{j-(i+1)}(s_{\text{bx}})$.

Directly from Lemma 9.26, we conclude that the arithmetical constraint of $<$ described in (LESS) are satisfied by (\mathcal{K}, w) , leading to $\mathbf{n}_{j-i}(w_{\text{ax}}) < \mathbf{n}_{j-i}(w_{\text{bx}})$. Indeed, $(\mathcal{K}', w) \models S_j^i(\text{ax}, \text{bx})$, allows us to apply Lemma 9.26(1) and conclude that, in \mathcal{K}' , $\mathbf{n}_{j-(i+1)}(s_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(s_{\text{bx}})$, $(\mathcal{K}', s_{\text{ax}}) \models \neg\text{val}$ and $(\mathcal{K}', s_{\text{bx}}) \models \text{val}$. Similarly, from $(\mathcal{K}', w) \models L_j^i(\text{ax}, \text{bx})$, by Lemma 9.26(2), we have that (in \mathcal{K}') for every $l_{\text{ax}} \in L_{\text{ax}}$ and $l_{\text{bx}} \in L_{\text{bx}}$, if $\mathbf{n}_{j-(i+1)}(l_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(l_{\text{bx}})$, then $(\mathcal{K}, l_{\text{ax}}) \models \text{val}$ iff $(\mathcal{K}, l_{\text{bx}}) \models \text{val}$. Now, by Lemma 9.15 and since $(\mathcal{K}', w_{\text{ax}})$ and $(\mathcal{K}', w_{\text{bx}})$ satisfy $\text{lsr}(j - i)$ (and thus $\text{type}(j - i)$, by (lsr₁)), we conclude that all these worlds satisfy the same numbers in both \mathcal{K}' and \mathcal{K} . More precisely, with respect to \mathcal{K} , we have $\mathbf{n}_{j-(i+1)}(s_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(s_{\text{bx}})$, $(\mathcal{K}, s_{\text{ax}}) \models \neg\text{val}$ and $(\mathcal{K}, s_{\text{bx}}) \models \text{val}$, as required by (S). Moreover, for every $l_{\text{ax}} \in L_{\text{ax}}$ and $l_{\text{bx}} \in L_{\text{bx}}$, if $\mathbf{n}_{j-(i+1)}(l_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(l_{\text{bx}})$, then $(\mathcal{K}, l_{\text{ax}}) \models \text{val}$ iff $(\mathcal{K}, l_{\text{bx}}) \models \text{val}$, as required by (L). That is, $\mathbf{n}_{j-i}(w_{\text{ax}}) < \mathbf{n}_{j-i}(w_{\text{bx}})$, with respect to \mathcal{K} .

(\Leftarrow): Suppose $\mathbf{n}_{j-i}(w_{\text{ax}}) < \mathbf{n}_{j-i}(w_{\text{bx}})$. Following the encoding of numbers, this implies that the t -children of w_{ax} and w_{bx} can be split into $P_{\text{ax}} = (L_{\text{ax}}, \{s_{\text{ax}}\}, R_{\text{ax}})$ and $P_{\text{bx}} = (L_{\text{bx}}, \{s_{\text{bx}}\}, R_{\text{bx}})$, respectively, so that the properties (LSR) and (LESS) are satisfied. Recall that, by hypothesis, $(\mathcal{K}, w) \models \text{fork}_j^i(\text{ax}, \text{bx})$, which implies that all the t -children of w_{ax} and w_{bx} have one $\{1, s, r\}$ -child for each proposition among 1, s and r . We use P_{ax} and P_{bx} to derive a subforest $\mathcal{K}' \subseteq \mathcal{K}$ where both $(\mathcal{K}', w_{\text{ax}})$ and $(\mathcal{K}', w_{\text{bx}})$ satisfy $\text{lsr}(j - i)$, and then rely on Lemma 9.26 to show that $(\mathcal{K}, w) \models [\text{ax} < \text{bx}]_j^i$.

The definition of $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ is quite simple: it is the Kripke-style finite forest obtained from \mathcal{K} by removing from R

- all (w_l, w') where w_l belongs to L_{ax} or L_{bx} , and w' is a $\{s, r\}$ -node,
- all (w_s, w') where w_s is either s_{ax} or s_{bx} , and w' is a $\{1, r\}$ -node,
- all (w_r, w') where w_r belongs to R_{ax} or R_{bx} , and w' is a $\{1, s\}$ -node.

Clearly, in \mathcal{K}' every world of $L_{\text{ax}} \cup L_{\text{bx}}$ (resp. $\{s_{\text{ax}}, s_{\text{bx}}\}; R_{\text{ax}} \cup R_{\text{bx}}$) has exactly one $\{1, s, r\}$ -child, which is in particular an 1-child (resp. s -child; r -child). Thus, $(\mathcal{K}', w_{\text{ax}})$ and $(\mathcal{K}', w_{\text{bx}})$ satisfy (lsr₂). Moreover, since only $\{1, s, r\}$ -nodes in $R^2(w_{\text{ax}}) \cup R^2(w_{\text{bx}})$ are removed in order to define R' , from the fact that $(\mathcal{K}, w_{\text{ax}})$ and $(\mathcal{K}, w_{\text{bx}})$ satisfy $\text{type}(j - i)$, we deduce that $(\mathcal{K}', w_{\text{ax}})$

and $(\mathcal{K}', w_{\text{bx}})$ still satisfy $\text{type}(j - i)$, as required by (lsr₁). Lastly, by definition of P_{ax} and P_{bx} , $(\mathcal{K}', w_{\text{ax}})$ and $(\mathcal{K}', w_{\text{bx}})$ satisfy (lsr₃). From Lemma 9.24, we conclude that $(\mathcal{K}', w_{\text{ax}})$ and $(\mathcal{K}', w_{\text{bx}})$ satisfy $\text{lsr}(j - i)$. Again from the fact that $(\mathcal{K}, w) \models \text{fork}_j^i(\text{ax}, \text{bx})$ we know that w_{ax} and w_{bx} are two distinct worlds corresponding to the nominals ax and bx , respectively. By definition of R' , the same holds true in \mathcal{K}' , and therefore $(\mathcal{K}, w) \models \text{nom}_i(\text{ax} \neq \text{bx})$. Moreover, we have $(\mathcal{K}, w) \models [t]^i \text{lsr}(j - i)$ and, from $(\mathcal{K}, w) \models \text{init}(j)$ and $\mathcal{K}' \subseteq \mathcal{K}$, we have $(\mathcal{K}', w) \models \text{init}(j)$. Therefore, (\mathcal{K}', w) satisfies the hypothesis needed to apply Lemma 9.26.

From the fact that P_{ax} and P_{bx} satisfy (LESS), with respect to \mathcal{K} we have:

- (S) $\mathbf{n}_{j-(i+1)}(s_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(s_{\text{bx}})$, $(\mathcal{K}, s_{\text{ax}}) \models \neg \text{val}$ and $(\mathcal{K}, s_{\text{bx}}) \models \text{val}$,
- (L) $(\mathcal{K}, l_{\text{ax}}) \models \text{val}$ iff $(\mathcal{K}, l_{\text{bx}}) \models \text{val}$, for every $l_{\text{ax}} \in L_{\text{ax}}$ and every $l_{\text{bx}} \in L_{\text{bx}}$ such that $\mathbf{n}_{j-(i+1)}(l_{\text{ax}}) = \mathbf{n}_{j-(i+1)}(l_{\text{bx}})$.

Since $(\mathcal{K}', \text{ax})$ and $(\mathcal{K}', \text{bx})$ satisfy $\text{type}(j - i)$, by Lemma 9.15 we deduce that all the worlds in (S) and (L) keep encoding the same numbers also with respect to \mathcal{K}' . From Lemma 9.26(1) we conclude that $(\mathcal{K}', w) \models S_j^i(\text{ax}, \text{bx})$, and from Lemma 9.26(2) we derive $(\mathcal{K}', w) \models L_j^i(\text{ax}, \text{bx})$. We conclude: $(\mathcal{K}', w) \models \text{nom}_i(\text{ax} \neq \text{bx}) \wedge [t]^i \text{lsr}(j - i) \wedge S_j^i(\text{ax}, \text{bx}) \wedge L_j^i(\text{ax}, \text{bx})$ and, from $\mathcal{K}' \subseteq \mathcal{K}$, $(\mathcal{K}, w) \models [\text{ax} < \text{bx}]_j^i$. \square

Let us now move to the formula $[\text{bx} = \text{ax} + 1]_j$. Similarly to $[\text{ax} < \text{bx}]_j^i$, we defined this formula under the hypothesis that (\mathcal{K}, w) satisfies $\text{fork}_j^1(\text{ax}, \text{bx})$. Therefore, w has exactly two t -children w_{ax} and w_{bx} , which correspond to the nominals ax and bx , respectively, for the depth 1. Moreover, $(\mathcal{K}, w_{\text{ax}})$ and $(\mathcal{K}, w_{\text{bx}})$ satisfy $\text{type}_{\text{lsr}}(j - 1)$. Recall that $[\text{bx} = \text{ax} + 1]_j$ should hold whenever $\mathbf{n}_{j-1}(w_{\text{bx}}) = \mathbf{n}_{j-1}(w_{\text{ax}}) + 1$.

As done in the base case for the definition of $[\text{bx} = \text{ax} + 1]_1$, we take advantage of arithmetical properties on binary numbers, and we search for two triples $P_{\text{ax}} = (L_{\text{ax}}, S_{\text{ax}}, R_{\text{ax}})$ and $P_{\text{bx}} = (L_{\text{bx}}, S_{\text{bx}}, R_{\text{bx}})$ satisfying the condition (LSR), and

(PLUS) P_{ax} and P_{bx} have the arithmetical properties of +1:

- (S&L) P_{ax} and P_{bx} satisfy (LESS),
- (R) for every $r_{\text{ax}} \in R_{\text{ax}}$ $(\mathcal{K}, r_{\text{ax}}) \models \text{val}$, and for every $r_{\text{bx}} \in R_{\text{bx}}$, $(\mathcal{K}, r_{\text{bx}}) \models \neg \text{val}$.

With this characterisation in mind, the definition of $[\text{bx} = \text{ax} + 1]_j$ is essentially the one of $[\text{ax} < \text{bx}]_j^i$, with the addition of the subformula $R(\text{ax}, \text{bx}) \stackrel{\text{def}}{=} @_{\text{ax}}^1[t](\Diamond r \Rightarrow \text{val}) \wedge @_{\text{bx}}^1[t](\Diamond r \Rightarrow \neg \text{val})$ added to capture the property (R). We have:

$$[\text{bx} = \text{ax} + 1]_j \stackrel{\text{def}}{=} \top * (\text{nom}_1(\text{ax} \neq \text{bx}) \wedge [t]^i \text{lsr}(j - 1) \wedge S_j^1(\text{ax}, \text{bx}) \wedge L_j^1(\text{ax}, \text{bx}) \wedge R(\text{ax}, \text{bx})),$$

Lemma 9.27. Let (\mathcal{K}, w) be a pointed forest satisfying $\text{init}(j)$. Let $\text{ax} \neq \text{bx} \in \text{Aux}$. Suppose $(\mathcal{K}, w) \models \text{fork}_j^1(\text{ax}, \text{bx})$, and let w_{ax} and w_{bx} be the worlds corresponding to the nominals ax and bx , respectively, for the depth 1. $(\mathcal{K}, w) \models [\text{bx} = \text{ax} + 1]_j$ if and only if $\mathbf{n}_{j-1}(w_{\text{ax}}) + 1 = \mathbf{n}_{j-1}(w_{\text{bx}})$.

Proof (sketch). The proof carries out very similarly to the proof of Lemma 9.25. Indeed, the formula $[\text{bx} = \text{ax} + 1]_j$ is essentially obtained from $[\text{ax} < \text{bx}]_j^1$ by juxtaposing the formula $R(\text{ax}, \text{bx})$ to the formulae $L_j^1(\text{ax}, \text{bx})$ and $S_j^1(\text{ax}, \text{bx})$.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models [\text{bx} = \text{ax} + 1]_j$, and therefore there is a subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K}' that satisfies $\text{nom}_i(\text{ax} \neq \text{bx}) \wedge [t]^i \text{lsr}(j - i) \wedge S_j^i(\text{ax}, \text{bx}) \wedge L_j^i(\text{ax}, \text{bx}) \wedge R(\text{ax}, \text{bx})$. From the first four conjuncts of this formula, by applying the proof of Lemma 9.25 we conclude that there are two partitions $P_{\text{ax}} = (L_{\text{ax}}, S_{\text{ax}}, R_{\text{ax}})$ and $P_{\text{bx}} = (L_{\text{bx}}, S_{\text{bx}}, R_{\text{bx}})$ of the t -worlds of w_{ax} and w_{bx} ,

respectively, that satisfy the conditions **(LSR)** and **(LESS)**. In order to conclude that **(PLUS)** is satisfied, it is sufficient to show that **(R)** holds, which follows from the satisfaction of $R(ax, bx)$. (\Leftarrow): Suppose $n_{j-i}(w_{ax}) + 1 = n_{j-i}(w_{bx})$. Following the encoding of numbers, this implies that the t -children of w_{ax} and w_{bx} can be split into $P_{ax} = (L_{ax}, \{s_{ax}\}, R_{ax})$ and $P_{bx} = (L_{bx}, \{s_{bx}\}, R_{bx})$, respectively, so that the properties **(LSR)** and **(PLUS)** are satisfied. We consider the Kripke-style finite forest $\mathcal{K}' \subseteq \mathcal{K}$ defined in the proof of the right to left direction of Lemma 9.25, for which we know that $(\mathcal{K}', w) \models \text{nom}_i(ax \neq bx) \wedge [t]^i \text{lsr}(j-i) \wedge S_j^i(ax, bx) \wedge L_j^i(ax, bx)$. Afterwards, $(\mathcal{K}', w) \models R(ax, bx)$ follows from the satisfaction of **(R)**. \square

Finally, we rely on the formulae $\text{fork}_j^i(ax, bx)$, $[ax < bx]_j^i$ and $[bx = ax+1]_j$ defined throughout this section in order to define the formulae $\text{uniq}(j)$ and $\text{compl}(j)$. The formula $\text{uniq}(j)$ is a natural extension of $\text{uniq}(1)$, where the formulae $\text{fork}_1^1(x, y)$ and $[x = y]_1^1$ are simply replaced by $\text{fork}_j^1(x, y)$ and $[x = y]_j^1 \stackrel{\text{def}}{=} \neg([ax < bx]_j^1 \vee [bx < ax]_j^1)$:

$$\text{uniq}(j) \stackrel{\text{def}}{=} \neg(\top * (\text{fork}_j^1(x, y) \wedge [x = y]_j^1)).$$

We establish the following result, which is proved as Lemma 9.19, simply by relying on the correctness of the formulae $\text{fork}_j^i(x, y)$ (Lemma 9.23) and $[x = y]_j^i$ (Lemma 9.25).

Lemma 9.28. Let (\mathcal{K}, w) be a pointed forest satisfying $\text{init}(j) \wedge \text{aux} \wedge \text{sub}(j)$. $(\mathcal{K}, w) \models \text{uniq}(j)$ if and only if (\mathcal{K}, w) satisfies **(uniq_j)**.

The formula $\text{compl}(j)$ is also quite similar to the formula $\text{compl}(1)$, the main difference being that the conjunct $[t] \Diamond y$ of $\text{compl}(1)$ is replaced by $[t](\text{type}_{\text{lsr}}(j-1) \wedge \Diamond y)$ in $\text{compl}(j)$, as needed to correctly evaluate $\text{fork}_j^1(x, y)$ (which replaces $\text{fork}_1^1(x, y)$). Formally, $\text{compl}(j)$ is defined as

$$\neg\left(\square \perp * \left([t](\text{type}_{\text{lsr}}(j-1) \wedge \Diamond y) \wedge \text{nom}_1(x) \wedge @_x^1 \neg 1_j \wedge \neg(\top * (\text{fork}_j^1(x, y) \wedge [y = x+1]_j))\right)\right),$$

where $1_j \stackrel{\text{def}}{=} [t]\text{val}$ reflects the encoding of the number $t(j, n) - 1$ for $j > 1$.

Lemma 9.29. Let (\mathcal{K}, w) be a pointed forest satisfying $\text{init}(j) \wedge \text{aux} \wedge \text{sub}(j)$. $(\mathcal{K}, w) \models \text{compl}(j)$ if and only if (\mathcal{K}, w) satisfies **(compl_j)**.

The proof of Lemma 9.29, which is analogous to the one of Lemma 9.20, is given in Appendix G. The definition of $\text{compl}(j)$ completes the definition of the formula $\text{type}(j)$. It is quite easy to check that the size of this formula is exponential in $j > 1$ and polynomial in $n \geq 1$. As its size is elementary, we can use this formula as a starting point to reduce Tile_k .

Below, we state the correctness of $\text{type}(j)$, which follows directly from Lemma 9.28 and Lemma 9.29, together with the (straightforward) correctness of $\text{sub}(j)$, aux and $\text{zero}(j)$.

Lemma 9.30. Let (\mathcal{K}, w) be a pointed forest satisfying $\text{init}(j)$. We have $(\mathcal{K}, w) \models \text{type}(j)$ iff (\mathcal{K}, w) satisfies **(sub_j)**, **(zero_j)**, **(uniq_j)**, **(compl_j)** and **(aux)**.

A quick check of $\text{init}(j)$ and the conditions **(sub_j)**, **(zero_j)**, **(uniq_j)**, **(compl_j)** and **(aux)** should be enough to convince the reader that they are simultaneously satisfiable, making $\text{init}(j) \wedge \text{type}(j)$ satisfiable by Lemma 9.30. As done in Lemma 9.22, we show below a model satisfying $\text{init}(j) \wedge \text{type}(j)$.

Lemma 9.31. Let $j \geq 2$. $\text{init}(j) \wedge \text{type}(j)$ is satisfiable.

Proof. By induction on j , we suppose that $\text{init}(j - 1) \wedge \text{type}(j - 1)$ is satisfiable (we already treated the base case for $j = 1$ in Lemma 9.22). So, let us consider $t(j, n)$ distinct worlds $w_0, \dots, w_{t(j,n)-1}$ and, by induction hypothesis, construct $t(j, n)$ Kripke-style finite forests $\mathcal{K}_i = (\mathcal{W}_i, R_i, \mathcal{V}_i)$ ($i \in [0, t(j, n) - 1]$), so that $w_i \in \mathcal{W}_i$ and $(\mathcal{K}_i, w_i) \models \text{init}(j - 1) \wedge \text{type}(j - 1)$. Without loss of generality, we assume that the universes of these finite forests to be mutually disjoint, i.e. for every distinct $i, j \in [0, t(j, n) - 1]$, $\mathcal{W}_i \cap \mathcal{W}_j = \emptyset$. From Proposition 9.3, we can also assume that the world w_i ($i \in [0, t(j, n) - 1]$) of \mathcal{W}_i is not a children of some other world in R_i . Lastly, we assume that every world w_i has exactly one x -child and one (distinct) y -child, and all its t -children have exactly one 1 -child, one s -child and one r -child (all distinct). Indeed, the presence of these children is not imposed nor excluded by (aux).

Since $(\mathcal{K}_i, w_i) \models \text{type}(j - 1)$, w_i has $t(j - 1, n)$ children, all encoding a distinct number in $[0, t(j - 1, n) - 1]$. Let us modify the valuation \mathcal{V}_i so that w_i encodes the number $i \in [0, t(j, n) - 1]$, which by definition can be written with $t(j - 1, n)$ bits. Let $b = b_{t(j-1,n)-1} \dots b_0$ be the binary representation of i (where b_0 is the least significant bit). We update \mathcal{V}_i (in a minimal way) so that every t -child \bar{w} of w_i belongs to $\mathcal{V}(\text{val})$ if and only if the bit $b_{n_{j-2}(\bar{w})}$ is 1. One can easily check that, after this update, $(\mathcal{K}_i, w_i) \models \text{init}(j - 1) \wedge \text{type}(j - 1)$ still holds, and moreover we have $n_{j-1}(w_i) = i$.

The pointed forest (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is defined below, can be shown to satisfy $\text{init}(j)$ (sub $_j$), (zero $_j$), (uniq $_j$), (compl $_j$) and (aux), leading to $(\mathcal{K}, w) \models \text{init}(j) \wedge \text{type}(j)$ directly from Lemma 9.30.

1. $\mathcal{W} \stackrel{\text{def}}{=} \{w\} \cup \mathcal{W}_i,$
2. $R \stackrel{\text{def}}{=} \{(w, w_0), \dots, (w, w_{t(j,n)-1})\} \cup \bigcup_{i \in [0, t(j,n)-1]} R_i,$
3. for every $p \in \text{AP}$, $\mathcal{V}(p) = \bigcup_{i \in [0, t(j,n)-1]} \mathcal{V}_i(p).$

Essentially, to define $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ we merge all the \mathcal{K}_i ($i \in [0, t(j, n) - 1]$), and add one edge from w to each w_i (which we assumed to not have a parent in R_i). As the subtree rooted in w_i is the same in both \mathcal{K} and \mathcal{K}_i , we conclude that for every $\bar{w} \in R(w)$, (\mathcal{K}, \bar{w}) satisfies both $\text{type}(j - 1)$ (as required by (sub $_j$)) and $\text{init}(j - 1)$ (as required by $\text{init}(j)$). The properties (zero $_j$), (uniq $_j$) and (compl $_j$) are satisfied from the fact that, given $i \in [0, t(j - n) - 1]$, w_i is a t -child of w and $n_{j-1}(w_i) = i$. Lastly, (aux) follows from the assumptions on (\mathcal{K}_i, w_i) . \square

9.3.6 Tiling a grid $[0, t(k, n) - 1] \times [0, t(k, n) - 1]$.

Strong of our formula $\text{type}(j)$, let us now develop a uniform reduction for Tile_k , for every $k \geq 2$. Some adaptations are required in order to encode a grid, but the hardest part was the design of $\text{type}(j)$. Let $k \geq 2$ and (\mathcal{T}, c) be an instance of Tile_k , where $\mathcal{T} = (\mathsf{T}, \mathsf{H}, \mathsf{V})$. In this section we construct a formula $\text{tiling}_{\mathcal{T}, c}(k)$ such that the following lemma holds.

Lemma 9.32. (\mathcal{T}, c) as a solution for Tile_k if and only if $\text{tiling}_{\mathcal{T}, c}(k)$ is satisfiable.

Recall that a solution for (\mathcal{T}, c) with respect to Tile_k is a map $\tau : [0, t(k, n) - 1]^2 \rightarrow \mathsf{T}$ satisfying the three conditions (first), (hori) and (vert) below:

- (first) $\tau(0, 0) = c$,
- (hori) for every $i \in [0, t(k, n) - 2]$ and $j \in [0, t(k, n) - 1]$, $(\tau(i, j), \tau(i + 1, j)) \in \mathsf{H}$,
- (vert) for all $i \in [0, t(k, n) - 1]$ and $j \in [0, t(k, n) - 2]$, $(\tau(i, j), \tau(i, j + 1)) \in \mathsf{V}$.

Let us first describe how to represent a grid $[0, t(k, n) - 1]^2$ in the pointed forest $(\mathcal{K}, \mathbf{w})$. We use the same ideas needed in order to define $\text{type}(j)$, but with some minor modifications. As previously stated, if $(\mathcal{K}, \mathbf{w}) \models \text{type}(k)$ then given a t -node $w' \in R(w)$, the number $\mathbf{n}_{k-1}(w') \in [0, t(k, n) - 1]$ is encoded using the t -children of w' , where the numbers encoded by these children represent positions in the binary encoding of $\mathbf{n}_{k-1}(w')$. Instead of being a single number, a position in the grid $[0, t(k, n) - 1]^2$ is a pair of numbers $(h, v) \in [0, t(k, n) - 1]^2$. Hence, in a model $(\mathcal{K}, \mathbf{w})$ satisfying $\text{tiling}_{\mathcal{T}, c}(k)$ we require that $w' \in R(w)$ encodes two numbers $\mathbf{n}_H(w')$ and $\mathbf{n}_V(w')$, and say that w' encodes the position (h, v) if and only if $\mathbf{n}_H(w') = h$ and $\mathbf{n}_V(w') = v$. Since both h and v are from $[0, t(k, n) - 1]$, the same amount of t -children as in $\text{type}(k)$ can be used in order to encode both $\mathbf{n}_H(w')$ and $\mathbf{n}_V(w')$. Thus, we require w' to satisfy the formula $\text{type}(k-1)$, forcing it to have the correct amount of t -children and, similarly to what is done previously for $\text{type}(j)$ ($j \geq 2$), we encode the numbers $\mathbf{n}_H(w')$ and $\mathbf{n}_V(w')$ by using the truth value, on the t -children of w' , of two new atomic propositions val_H and val_V , respectively. Then, we rely on formulae that are similar to $\text{zero}(k)$, $\text{uniq}(k)$ and $\text{compl}(k)$ in order to state that w witnesses exactly one child for each position in the grid. For example, the condition (uniq_j) becomes

for all distinct t -nodes $w_1, w_2 \in R(w)$, either $\mathbf{n}_H(w_1) \neq \mathbf{n}_H(w_2)$ or $\mathbf{n}_V(w_1) \neq \mathbf{n}_V(w_2)$.

Once the grid is encoded, the tiling conditions can be enforced rather easily, thanks to the kit of formulae defined for $\text{type}(j)$, that allows us to have access to $\mathbf{n}_H(w')$ and $\mathbf{n}_V(w')$. We see the tile types T as a set of atomic propositions, disjoint from the set $\{p_1, \dots, p_n, \text{val}, \text{val}_H, \text{val}_V\} \cup \text{Aux}$ used in order to define the grid, and for instance express vertical constraint (vert) , with a formula in $\text{ML}(\ast)$ stating that

for all t -nodes $w_1, w_2 \in R(w)$, if $\mathbf{n}_V(w_2) = \mathbf{n}_V(w_1) + 1$ and $\mathbf{n}_H(w_2) = \mathbf{n}_H(w_1)$ then there is $(c_1, c_2) \in V$ such that $w_1 \in \mathcal{V}(c_1)$ and $w_2 \in \mathcal{V}(c_2)$.

The section is divided in two parts: we first show how to modify $\text{type}(k)$ in order to encode the grid $[0, t(k, n) - 1]^2$, and then introduce the formulae characterising the tiling conditions.

Encoding the grid. We introduce the formula $\text{grid}_{\mathcal{T}}(k)$ that characterises the set of models encoding the $[0, t(k, n) - 1]^2$ grid. A pointed forest $(\mathcal{K}, \mathbf{w})$, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, satisfying $\text{grid}_{\mathcal{T}}(k)$ is characterised as follows:

(init/sub/aux) $(\mathcal{K}, \mathbf{w})$ satisfies $\text{init}(k)$, $\text{sub}(k)$ and aux ,

(zero $_{\mathcal{T}, k}$) there is a t -node $\tilde{w} \in R(w)$ such that $\mathbf{n}_H(\tilde{w}) = 0$ and $\mathbf{n}_V(\tilde{w}) = 0$,

(uniq $_{\mathcal{T}, k}$) for all two distinct t -nodes $w_1, w_2 \in R(w)$, $\mathbf{n}_H(w_1) \neq \mathbf{n}_H(w_2)$ or $\mathbf{n}_V(w_1) \neq \mathbf{n}_V(w_2)$,

(compl[H] $_{\mathcal{T}, k}$) for all t -node $w_1 \in R(w)$, if $\mathbf{n}_H(w_1) < t(k, n) - 1$ then there is a t -node $w_2 \in R(w)$ such that $\mathbf{n}_H(w_2) = \mathbf{n}_H(w_1) + 1$ and $\mathbf{n}_V(w_2) = \mathbf{n}_V(w_1)$,

(compl[V] $_{\mathcal{T}, k}$) for every t -node $w_1 \in R(w)$, if $\mathbf{n}_V(w_1) < t(k, n) - 1$ there is a t -node $w_2 \in R(w)$ such that $\mathbf{n}_H(w_2) = \mathbf{n}_H(w_1)$ and $\mathbf{n}_V(w_2) = \mathbf{n}_V(w_1) + 1$.

It is easy to see that, with these conditions, $(\mathcal{K}, \mathbf{w})$ correctly encodes the grid $[0, t(k, n) - 1]^2$. Exactly as in the definition of $\text{type}(j)$, we define $\text{grid}_{\mathcal{T}}(k)$ as a conjunction of seven formulae, following the characterisation above:

$$\text{grid}_{\mathcal{T}}(k) \stackrel{\text{def}}{=} \text{zero}_{\mathcal{T}}(k) \wedge \text{uniq}_{\mathcal{T}}(k) \wedge \text{compl[H]}_{\mathcal{T}}(k) \wedge \text{compl[V]}_{\mathcal{T}}(k) \wedge \text{init}(k) \wedge \text{sub}(k) \wedge \text{aux}.$$

Each conjunct of $\text{grid}_{\mathcal{T}}(k)$ expresses the homonymous property. Exactly as for the formula $\text{zero}(j)$ of $\text{type}(j)$, defining the formula $\text{zero}_{\mathcal{T}}(k)$ is very easy:

$$\mathbf{zero}_{\mathcal{T}}(k) \stackrel{\text{def}}{=} \langle t \rangle ([t](\neg \mathbf{val}_H \wedge \neg \mathbf{val}_V)).$$

Indeed, given a world $w' \in R(w)$, $\mathbf{n}_H(w') = 0$ holds whenever all its t -children falsify \mathbf{val}_H , whereas $\mathbf{n}_V(w') = 0$ holds whenever all its t -children falsify \mathbf{val}_V .

In order to define the three conjuncts $\mathbf{uniq}_{\mathcal{T}}(k)$, $\mathbf{compl}[H]_{\mathcal{T}}(k)$ and $\mathbf{compl}[V]_{\mathcal{T}}(k)$, and thus completing the definition of $\mathbf{grid}_{\mathcal{T}}(k)$, we start by defining the formulae $L[D]_k(ax, bx)$, $S[D]_k(ax, bx)$ and $R[D](ax, bx)$, where $D \in \{H, V\}$. The semantics of these three formulae is similar to $L_k^1(ax, bx)$, $S_k^1(ax, bx)$ and $R(ax, bx)$, respectively, with the difference that, for a given t -node in $R^2(w)$, we are interested in the satisfaction of \mathbf{val}_D instead of \mathbf{val} . For instance, we recall that the formula $S_k^1(ax, bx)$ is defined as

$$\begin{aligned} S_k^1(ax, bx) &= \top * (\mathbf{fork}_k^2(x, y) \wedge @_{ax}^1 \langle t \rangle (\Diamond s \wedge \Diamond x) \wedge @_{bx}^1 \langle t \rangle (\Diamond s \wedge \Diamond y) \\ &\quad \wedge [x=y]_k^2 \wedge @_x^2 \neg \mathbf{val} \wedge @_y^2 \mathbf{val}). \end{aligned}$$

The formula $S[D]_k(ax, bx)$ is defined simply by replacing the two occurrences of \mathbf{val} above by \mathbf{val}_D . Notice that other occurrences of \mathbf{val} appearing in the definitions of the subformulae $\mathbf{fork}_k^2(x, y)$ and $[x=y]_k^2$ are kept, as the grid is only enforced at the level of the children of w . We obtain:

$$\begin{aligned} S[D]_k(ax, bx) &\stackrel{\text{def}}{=} \top * (\mathbf{fork}_k^2(x, y) \wedge @_{ax}^1 \langle t \rangle (\Diamond s \wedge \Diamond x) \wedge @_{bx}^1 \langle t \rangle (\Diamond s \wedge \Diamond y) \\ &\quad \wedge [x=y]_k^2 \wedge @_x^2 \neg \mathbf{val}_D \wedge @_y^2 \mathbf{val}_D). \end{aligned}$$

Similarly, $L[D]_k(ax, bx)$ is defined by replacing the last conjunct of $L_k^1(ax, bx)$, that is the subformula $\neg(@_x^2 \mathbf{val} \Leftrightarrow @_y^2 \mathbf{val})$, by $\neg(@_x^2 \mathbf{val}_D \Leftrightarrow @_y^2 \mathbf{val}_D)$. The formula $R[D](ax, bx)$ is defined from $R(ax, bx)$ by replacing every occurrence of \mathbf{val} by \mathbf{val}_D . Explicitly,

$$\begin{aligned} L[D]_k(ax, bx) &\stackrel{\text{def}}{=} \neg(\top * (\mathbf{fork}_j^{i+1}(x, y) \wedge @_{ax}^i \langle t \rangle (\Diamond 1 \wedge \Diamond x) \wedge @_{bx}^i \langle t \rangle (\Diamond 1 \wedge \Diamond y) \\ &\quad \wedge [x=y]_j^{i+1} \wedge \neg(@_x^{i+1} \mathbf{val}_D \Leftrightarrow @_y^{i+1} \mathbf{val}_D))), \end{aligned}$$

$$R[D](ax, bx) \stackrel{\text{def}}{=} @_ax^1[t](\Diamond r \Rightarrow \mathbf{val}_D) \wedge @_bx^1[t](\Diamond r \Rightarrow \neg \mathbf{val}_D).$$

The formulae $L[D]_k(ax, bx)$, $S[D]_k(ax, bx)$ and $R[D](ax, bx)$ allow us to rephrase $[ax < bx]_k^1$ and $[bx = ax+1]_k$ so that the numbers encoded with \mathbf{val}_D are considered instead: it is sufficient to replace $L_k^1(ax, bx)$, $S_k^1(ax, bx)$ and $R(ax, bx)$ by $L[D]_k(ax, bx)$, $S[D]_k(ax, bx)$ and $R[D](ax, bx)$. This leads to the following formulae $[ax \stackrel{D}{<} bx]_k$ and $[bx \stackrel{D}{=} ax+1]_k$:

$$\begin{aligned} [ax \stackrel{D}{<} bx]_k &\stackrel{\text{def}}{=} \top * (\mathbf{nom}_i(ax \neq bx) \wedge [t]\mathbf{lsl}(k-1) \wedge S[D]_k(ax, bx) \wedge L[D]_k(ax, bx)), \\ [bx \stackrel{D}{=} ax+1]_k &\stackrel{\text{def}}{=} \top * (\mathbf{nom}_1(ax \neq bx) \wedge [t]\mathbf{lsl}(k-1) \wedge L[D]_k(ax, bx) \wedge S[D]_k(ax, bx) \wedge R[D](ax, bx)). \end{aligned}$$

The following result establishes the semantics of these two formulae.

Lemma 9.33. Let $ax \neq bx \in \text{Aux}$. Suppose $(\mathcal{K}, w) \models \mathbf{init}(k) \wedge \mathbf{fork}_k^1(ax, bx)$, and let w_{ax} and w_{bx} be the two t -children of w that correspond to the nominals ax and bx , respectively.

1. $(\mathcal{K}, w) \models [ax \stackrel{D}{<} bx]_k$ if and only if $\mathbf{n}_D(w_{ax}) < \mathbf{n}_D(w_{bx})$,
2. $(\mathcal{K}, w) \models [bx \stackrel{D}{=} ax+1]_k$ if and only if $\mathbf{n}_D(w_{bx}) = \mathbf{n}_D(w_{ax}) + 1$.

The proof of (1) unfolds as the proof of Lemma 9.25, whereas the proof of (2) unfolds as the one of Lemma 9.27. We leave the technical developments of these proofs to the reader. We write $[ax \stackrel{D}{=} bx]_k$ for $\neg([ax \stackrel{D}{<} bx]_k \vee [bx \stackrel{D}{<} ax]_k)$, the formula satisfied whenever $\mathbf{n}_D(w_{ax}) = \mathbf{n}_D(w_{bx})$.

We are now ready to define the formulae $\text{uniq}_{\mathcal{T}}(k)$, $\text{compl}[\mathsf{H}]_{\mathcal{T}}(k)$ and $\text{compl}[\mathsf{V}]_{\mathcal{T}}(k)$, achieving the conditions $(\text{uniq}_{\mathcal{T},k})$, $(\text{compl}[\mathsf{H}]_{\mathcal{T},k})$ and $(\text{compl}[\mathsf{V}]_{\mathcal{T},k})$, respectively. All these formulae follow closely the definitions of $\text{uniq}(k)$ and $\text{compl}(k)$ introduced in the previous section, hence we refer to these latter formulae for an informal description on how they work. Indeed, in order to define $\text{uniq}_{\mathcal{T}}(k)$ it is sufficient to replace, in $\text{uniq}(k)$, the subformula $[x=y]_k^1$ by $[x \stackrel{\mathsf{H}}{=} y]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k$, thus obtaining the formula below:

$$\text{uniq}_{\mathcal{T}}(k) \stackrel{\text{def}}{=} \neg(\top * (\text{fork}_k^1(x, y) \wedge [x \stackrel{\mathsf{H}}{=} y]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k)).$$

Lemma 9.34. Suppose $(\mathcal{K}, w) \models \text{init}(k) \wedge \text{aux}$. $(\mathcal{K}, w) \models \text{uniq}(k)$ iff (\mathcal{K}, w) satisfies $(\text{uniq}_{\mathcal{T},k})$.

Proof (sketch). This lemma is proven as Lemma 9.19 and Lemma 9.28, by relying on Lemma 9.33 in order to show that, given two distinct worlds w_x and w_y corresponding to nominals (for the depth 1) x and y , respectively, $[x \stackrel{\mathsf{H}}{=} y]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k$ holds if and only if $\mathbf{n}_{\mathsf{H}}(w_x) = \mathbf{n}_{\mathsf{H}}(w_y)$ and $\mathbf{n}_{\mathsf{V}}(w_x) = \mathbf{n}_{\mathsf{V}}(w_y)$. \square

Similarly, the definitions of $\text{compl}[\mathsf{H}]_{\mathcal{T}}(k)$ and $\text{compl}[\mathsf{V}]_{\mathcal{T}}(k)$ are obtained by replacing, in $\text{compl}(k)$, the formula $[y = x+1]_j$ by $[y \stackrel{\mathsf{H}}{=} x+1]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k$ and $[x \stackrel{\mathsf{H}}{=} y]_k \wedge [y \stackrel{\mathsf{V}}{=} x+1]_k$, respectively. This is not surprising, as for instance the formula $[y \stackrel{\mathsf{H}}{=} x+1]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k$ is satisfied whenever $\mathbf{n}_{\mathsf{H}}(w_x) + 1 = \mathbf{n}_{\mathsf{H}}(w_y)$ and $\mathbf{n}_{\mathsf{V}}(w_x) = \mathbf{n}_{\mathsf{V}}(w_y)$, as required in $(\text{compl}[\mathsf{H}]_{\mathcal{T},k})$. In order to define $\text{compl}[\mathsf{H}]_{\mathcal{T}}(k)$, we also replace the formula 1_k , which in $\text{type}(k)$ reflects the encoding of the number $t(j, n) - 1$, by $1_k^{\mathsf{H}} \stackrel{\text{def}}{=} [t]\text{val}_{\mathsf{H}}$, which again reflects the encoding of $t(j, n) - 1$, but with respect to the horizontal dimension of the grid. Likewise, to define correctly $\text{compl}[\mathsf{V}]_{\mathcal{T}}(k)$ we replace 1_k by $1_k^{\mathsf{V}} \stackrel{\text{def}}{=} [t]\text{val}_{\mathsf{V}}$. We obtain the following formulae:

$$\begin{aligned} \text{compl}[\mathsf{H}]_{\mathcal{T}}(k) &\stackrel{\text{def}}{=} \neg \left(\square \perp * \left([t](\text{type}_{\text{lsr}}(k-1) \wedge \Diamond y) \wedge \text{nom}_1(x) \wedge @_x^1 \neg 1_k^{\mathsf{H}} \right. \right. \\ &\quad \left. \left. \wedge \neg(\top * (\text{fork}_k^1(x, y) \wedge [y \stackrel{\mathsf{H}}{=} x+1]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k)) \right) \right), \\ \text{compl}[\mathsf{V}]_{\mathcal{T}}(k) &\stackrel{\text{def}}{=} \neg \left(\square \perp * \left([t](\text{type}_{\text{lsr}}(k-1) \wedge \Diamond y) \wedge \text{nom}_1(x) \wedge @_x^1 \neg 1_k^{\mathsf{V}} \right. \right. \\ &\quad \left. \left. \wedge \neg(\top * (\text{fork}_k^1(x, y) \wedge [x \stackrel{\mathsf{H}}{=} y]_k \wedge [y \stackrel{\mathsf{V}}{=} x+1]_k)) \right) \right). \end{aligned}$$

Lemma 9.35. Suppose $(\mathcal{K}, w) \models \text{init}(k) \wedge \text{aux}$. We have,

1. $(\mathcal{K}, w) \models \text{compl}[\mathsf{H}]_{\mathcal{T}}(k)$ if and only if (\mathcal{K}, w) satisfies $(\text{compl}[\mathsf{H}]_{\mathcal{T},k})$,
2. $(\mathcal{K}, w) \models \text{compl}[\mathsf{V}]_{\mathcal{T}}(k)$ if and only if (\mathcal{K}, w) satisfies $(\text{compl}[\mathsf{V}]_{\mathcal{T},k})$.

Proof (sketch). Both (1) and (2) are proved as Lemma 9.20 and Lemma 9.29, with the sole difference that we rely on Lemma 9.33 in order to show that, given two distinct worlds w_x and w_y corresponding to nominals (for the depth 1) x and y , respectively, the formula $[y \stackrel{\mathsf{H}}{=} x+1]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k$ holds if and only if $\mathbf{n}_{\mathsf{H}}(w_x) = \mathbf{n}_{\mathsf{H}}(w_y) + 1$ and $\mathbf{n}_{\mathsf{V}}(w_x) = \mathbf{n}_{\mathsf{V}}(w_y)$ (which leads to (1)), whereas the formula $[y \stackrel{\mathsf{V}}{=} x+1]_k \wedge [x \stackrel{\mathsf{H}}{=} y]_k$ holds if and only if $\mathbf{n}_{\mathsf{H}}(w_x) = \mathbf{n}_{\mathsf{H}}(w_y)$ and $\mathbf{n}_{\mathsf{V}}(w_x) = \mathbf{n}_{\mathsf{V}}(w_y) + 1$ (which leads to (2)). \square

This concludes the definition of $\text{grid}_{\mathcal{T}}(k)$. Its correctness, whose proof follows directly from Lemmata 9.34 and 9.35, together with the straightforward correctness of $\text{zero}_{\mathcal{T}}(k)$, is stated in the following lemma.

Lemma 9.36. $(\mathcal{K}, w) \models \text{grid}_{\mathcal{T}}(k)$ if and only if (\mathcal{K}, w) satisfies (init/sub/aux) , $(\text{zero}_{\mathcal{T},k})$, $(\text{uniq}_{\mathcal{T},k})$, $(\text{compl}[\mathsf{H}]_{\mathcal{T},k})$ and $(\text{compl}[\mathsf{V}]_{\mathcal{T},k})$.

Corollary 9.37. $\text{grid}_{\mathcal{T}}(k)$ is satisfiable.

Proof (sketch). The satisfiability of $\text{grid}_{\mathcal{T}}(k)$ can be established by Lemma 9.36 as $(\text{zero}_{\mathcal{T},k})$, $(\text{uniq}_{\mathcal{T},k})$, $(\text{compl[H]}_{\mathcal{T},k})$, $(\text{compl[V]}_{\mathcal{T},k})$ and (init/sub/aux) can be simultaneously satisfied. A pointed forest $(\mathcal{K}, \mathbf{w})$ satisfying these constraints can be defined similarly to what is done in Lemma 9.31 for the formula $\text{init}(j) \wedge \text{type}(j)$, the main difference being that $t(k, n)^2$ t -children of \mathbf{w} need to be considered, instead of $t(k, n)$, in order to encode every pair in $[0, t(k, n) - 1]^2$. \square

Forcing the tiling conditions. We now encode the tiling conditions. Again, let $k \geq 2$ and, without loss of generality, suppose the tile types T of \mathcal{T} to be a finite set of atomic propositions from AP , disjoint from the set $\{p_1, \dots, p_n, \text{val}, \text{val}_H, \text{val}_V\} \cup \text{Aux}$ used in order to define the formula $\text{grid}_{\mathcal{T}}(k)$. Given a pointed forest $(\mathcal{K}, \mathbf{w})$ satisfying $\text{grid}_{\mathcal{T}}(k)$, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, the existence of a solution for the instance $(\mathcal{T}, \mathbf{c})$ of the tiling problem Tile_k can be expressed with the following conditions:

- $(\text{one}_{\mathcal{T}})$ every t -node in $R(\mathbf{w})$ satisfies exactly one tile type $\mathsf{T} \subseteq_{\text{fin}} \text{AP}$,
- $(\text{first}_{\mathcal{T}, \mathbf{c}})$ for $\tilde{\mathbf{w}} \in R(\mathbf{w})$, if $\mathbf{n}_H(\tilde{\mathbf{w}}) = \mathbf{n}_V(\tilde{\mathbf{w}}) = 0$ then $\tilde{\mathbf{w}} \in \mathcal{V}(\mathbf{c})$,
- $(\text{hor}_{\mathcal{T}})$ for all $\mathbf{w}_1, \mathbf{w}_2 \in R(\mathbf{w})$, if $\mathbf{n}_H(\mathbf{w}_2) = \mathbf{n}_H(\mathbf{w}_1) + 1$ and $\mathbf{n}_V(\mathbf{w}_2) = \mathbf{n}_V(\mathbf{w}_1)$ then there is $(\mathbf{c}_1, \mathbf{c}_2) \in \mathsf{H}$ such that $\mathbf{w}_1 \in \mathcal{V}(\mathbf{c}_1)$ and $\mathbf{w}_2 \in \mathcal{V}(\mathbf{c}_2)$,
- $(\text{vert}_{\mathcal{T}})$ for all $\mathbf{w}_1, \mathbf{w}_2 \in R(\mathbf{w})$, if $\mathbf{n}_V(\mathbf{w}_2) = \mathbf{n}_V(\mathbf{w}_1) + 1$ and $\mathbf{n}_H(\mathbf{w}_2) = \mathbf{n}_H(\mathbf{w}_1)$ then there is $(\mathbf{c}_1, \mathbf{c}_2) \in \mathsf{V}$ such that $\mathbf{w}_1 \in \mathcal{V}(\mathbf{c}_1)$ and $\mathbf{w}_2 \in \mathcal{V}(\mathbf{c}_2)$.

The connection between these four conditions and a solution $\tau : [0, t(k, n) - 1]^2 \rightarrow \mathsf{T}$ of Tile_k should be clear: $(\text{one}_{\mathcal{T}})$ simply models the functionality of τ , whereas $(\text{first}_{\mathcal{T}, \mathbf{c}})$, $(\text{hor}_{\mathcal{T}})$ and $(\text{vert}_{\mathcal{T}})$ correspond to the tiling conditions (first) , (hori) and (vert) . So, the formula $\text{tiling}_{\mathcal{T}, \mathbf{c}}(k)$ characterising the admissible instances of Tile_k (Lemma 9.32) can be defined as follows:

$$\text{tiling}_{\mathcal{T}, \mathbf{c}}(k) \stackrel{\text{def}}{=} \text{grid}_{\mathcal{T}}(k) \wedge \text{one}_{\mathcal{T}} \wedge \text{first}_{\mathcal{T}, \mathbf{c}}(k) \wedge \text{hor}_{\mathcal{T}}(k) \wedge \text{vert}_{\mathcal{T}}(k),$$

where the last four conjuncts express the homonymous property above. Given the tool kit of formulae introduced up to now, these four formulae are easy to define. For $\text{one}_{\mathcal{T}}$ and $\text{first}_{\mathcal{T}, \mathbf{c}}(k)$, we simply have:

$$\begin{aligned} \text{one}_{\mathcal{T}} &\stackrel{\text{def}}{=} [t] \bigvee_{\mathbf{c}_1 \in \mathsf{T}} (\mathbf{c}_1 \wedge \bigwedge_{\mathbf{c}_2 \in \mathsf{T}} \neg \mathbf{c}_2), \\ \text{first}_{\mathcal{T}, \mathbf{c}}(k) &\stackrel{\text{def}}{=} [t]([t](\neg \text{val}_H \wedge \neg \text{val}_V) \Rightarrow \mathbf{c}). \end{aligned}$$

Lemma 9.38. Suppose $(\mathcal{K}, \mathbf{w}) \models \text{grid}_{\mathcal{T}}(k)$. We have,

1. $(\mathcal{K}, \mathbf{w}) \models \text{one}_{\mathcal{T}}$ if and only if $(\mathcal{K}, \mathbf{w})$ satisfies $(\text{one}_{\mathcal{T}})$,
2. $(\mathcal{K}, \mathbf{w}) \models \text{first}_{\mathcal{T}, \mathbf{c}}(k)$ if and only if $(\mathcal{K}, \mathbf{w})$ satisfies $(\text{first}_{\mathcal{T}, \mathbf{c}})$.

Proof. Both statements follow easily from the definition of the two formulae. For the formula $\text{first}_{\mathcal{T}, \mathbf{c}}(k)$, we recall that $(\mathcal{K}, \mathbf{w}) \models \text{grid}_{\mathcal{T}}(k)$ implies the existence of a t -node $\mathbf{w}_0 \in R(\mathbf{w})$ encoding the position $(0, 0)$ of the grid. Among the t -children of \mathbf{w} , \mathbf{w}_0 is the only one satisfying $[t](\neg \text{val}_H \wedge \neg \text{val}_V)$. \square

The formula $\text{hor}_{\mathcal{T}}(k)$ is defined by stating that there cannot be two t -nodes $\mathbf{w}_1, \mathbf{w}_2 \in R(\mathbf{w})$ such that \mathbf{w}_2 encodes the position $(\mathbf{n}_H(\mathbf{w}_1) + 1, \mathbf{n}_V(\mathbf{w}_1))$, but $\mathbf{w}_1 \in \mathcal{V}(\mathbf{c}_1)$, $\mathbf{w}_2 \in \mathcal{V}(\mathbf{c}_2)$ does not hold for any $(\mathbf{c}_1, \mathbf{c}_2) \in \mathsf{H}$. In the lingua of $\text{ML}(\ast)$:

$$\text{hor}_{\mathcal{T}}(k) \stackrel{\text{def}}{=} \neg(\top * (\text{fork}_k^1(x, y) \wedge [y \stackrel{H}{=} x+1]_k \wedge [x \stackrel{V}{=} y]_k \wedge \neg \bigvee_{(\mathbf{c}_1, \mathbf{c}_2) \in \mathsf{H}} (@_x^1 \mathbf{c}_1 \wedge @_y^1 \mathbf{c}_2))).$$

Lastly, $\text{vert}_{\mathcal{T}}(k)$ is defined from $\text{hor}_{\mathcal{T}}(k)$ by swapping H and V :

$$\text{vert}_{\mathcal{T}}(k) \stackrel{\text{def}}{=} \neg(\top * (\text{fork}_k^1(x, y) \wedge [y \stackrel{\mathsf{V}}{=} x+1]_k \wedge [x \stackrel{\mathsf{H}}{=} y]_k \wedge \neg V_{(c_1, c_2) \in \mathsf{H}}(@_x^1 c_1 \wedge @_y^1 c_2))).$$

Lemma 9.39. Suppose $(\mathcal{K}, w) \models \text{grid}_{\mathcal{T}}(k)$. We have,

1. $(\mathcal{K}, w) \models \text{hor}_{\mathcal{T}}(k)$ if and only if (\mathcal{K}, w) satisfies $(\text{hor}_{\mathcal{T}})$,
2. $(\mathcal{K}, w) \models \text{vert}_{\mathcal{T}}(k)$ if and only if (\mathcal{K}, w) satisfies $(\text{vert}_{\mathcal{T}})$.

Proof. We develop the proof of (1), the one for (2) being analogous. Recall that the condition $(\text{hor}_{\mathcal{T}})$ states that for all $w_1, w_2 \in R(w)$, if $\mathbf{n}_{\mathsf{H}}(w_2) = \mathbf{n}_{\mathsf{H}}(w_1) + 1$ and $\mathbf{n}_{\mathsf{V}}(w_2) = \mathbf{n}_{\mathsf{V}}(w_1)$ then there is $(c_1, c_2) \in \mathsf{H}$ such that $w_1 \in \mathcal{V}(c_1)$ and $w_2 \in \mathcal{V}(c_2)$. Since $(\mathcal{K}, w) \models \text{grid}_{\mathcal{T}}(k)$, every t -child w' of w encodes a pair of numbers $(\mathbf{n}_{\mathsf{H}}(w), \mathbf{n}_{\mathsf{V}}(w)) \in [0, t(k, n) - 1]^2$. Moreover, from $(\text{init}/\text{sub}/\text{aux})$, these children satisfy $\text{type}(k-1)$ and have exactly one x -child and exactly one distinct y -children.

(\Rightarrow) : Suppose $(\mathcal{K}, w) \models \text{hor}_{\mathcal{T}}(k)$. Then, by definition, for every subforest $\mathcal{K}' \subseteq \mathcal{K}$, if (\mathcal{K}', w) satisfies $\text{fork}_k^1(x, y) \wedge [y \stackrel{\mathsf{H}}{=} x+1]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k$ then $(\mathcal{K}', w) \models V_{(c_1, c_2) \in \mathsf{H}}(@_x^1 c_1 \wedge @_y^1 c_2)$. Consider two t -nodes $w_x, w_y \in R(w)$ such that $\mathbf{n}_{\mathsf{H}}(w_y) = \mathbf{n}_{\mathsf{H}}(w_x) + 1$ and $\mathbf{n}_{\mathsf{V}}(w_y) = \mathbf{n}_{\mathsf{V}}(w_x)$. Let $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ be the subforest of \mathcal{K} where R' is defined from R by removing the following pairs of worlds:

- every $(w, w') \in R$ where w' is different from w_x and w_y ,
- $(w_x, w'') \in R$ where w'' is the only y -child of w_x ,
- $(w_y, w''') \in R$ where w''' is the only x -child of w_y .

Since (\mathcal{K}, w_x) and (\mathcal{K}, w_y) satisfy $\text{type}(k-1)$, and w_x and w_y only lost Aux -children when defining R' from R , it is easy to see that (\mathcal{K}', w_x) and (\mathcal{K}', w_y) satisfy $\text{type}(k-1)$. Moreover, since (\mathcal{K}, w) satisfies aux , all the t -children of w_x and w_y have, in both \mathcal{K} and \mathcal{K}' , exactly one $\{1, s, r\}$ -child for each symbol among $1, s$ and r . We conclude that the pointed forest (\mathcal{K}', w) satisfies $\text{fork}_k^1(x, y)$. By Lemma 9.15 (which can be easily adapted in order to consider the pairs of numbers described with val_{H} and val_{V} , instead of a number described with val), we conclude that $\mathbf{n}_{\mathsf{H}}(w_y) = \mathbf{n}_{\mathsf{H}}(w_x) + 1$ and $\mathbf{n}_{\mathsf{V}}(w_y) = \mathbf{n}_{\mathsf{V}}(w_x)$ holds also with respect to \mathcal{K}' . By Lemma 9.33 we derive $(\mathcal{K}', w) \models [y \stackrel{\mathsf{H}}{=} x+1]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k$. From $(\mathcal{K}, w) \models \text{hor}_{\mathcal{T}}(k)$, we conclude that there is a pair $(c_1, c_2) \in \mathsf{H}$ such that $(\mathcal{K}', w) \models @_x^1 c_1 \wedge @_y^1 c_2$. Since w_x (resp. w_y) corresponds to the nominal x (resp. y) for the depth 1, we have $w_x \in \mathcal{V}(c_1)$ and $w_y \in \mathcal{V}(c_2)$. This allows us to conclude that (\mathcal{K}, w) satisfies $(\text{hor}_{\mathcal{T}})$.

(\Leftarrow) : This direction is rather straightforward and, analogously to the left-to-right direction, relies on Lemmata 9.15 and 9.33. Briefly, suppose that (\mathcal{K}, w) satisfies $(\text{hor}_{\mathcal{T}})$ and, *ad absurdum*, assume that $(\mathcal{K}, w) \not\models \text{hor}_{\mathcal{T}}(k)$. Therefore, there is a subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K} such that

$$(\mathcal{K}, w) \models \text{fork}_k^1(x, y) \wedge [y \stackrel{\mathsf{H}}{=} x+1]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k \wedge \neg V_{(c_1, c_2) \in \mathsf{H}}(@_x^1 c_1 \wedge @_y^1 c_2).$$

From $(\mathcal{K}', w) \models \text{fork}_k^1(x, y)$ we conclude that there are two worlds w_x and w_y corresponding to two nominals (depth 1) x and y , respectively. Moreover, by Lemma 9.15, these worlds encode the same two numbers w.r.t. \mathcal{K} and \mathcal{K}' . From $(\mathcal{K}', w) \models [y \stackrel{\mathsf{H}}{=} x+1]_k \wedge [x \stackrel{\mathsf{V}}{=} y]_k$ and the fact that (\mathcal{K}, w) satisfies $(\text{hor}_{\mathcal{T}})$, together with Lemma 9.33, we conclude that there is a pair $(c_1, c_2) \in \mathsf{H}$ such that $w_x \in \mathcal{V}(c_1)$ and $w_y \in \mathcal{V}(c_2)$. However, this contradicts $\mathcal{K}', w \models \neg V_{(c_1, c_2) \in \mathsf{H}}(@_x^1 c_1 \wedge @_y^1 c_2)$. Thus, $\mathcal{K}, w \models \text{hor}_{\mathcal{T}}(k)$. \square

The formulae $\text{one}_{\mathcal{T}}, \text{first}_{\mathcal{T}, c}(k) \text{ hor}_{\mathcal{T}}(k)$, $\text{vert}_{\mathcal{T}}(k)$ conclude the definition of $\text{tiling}_{\mathcal{T}, c}(k)$. We are now ready to establish Lemma 9.32 (recalled below), which leads directly to the TOWER-hardness of the satisfiability problem of $\text{ML}(\ast)$.

Lemma 9.32. (\mathcal{T}, c) as a solution for Tile_k if and only if $\text{tiling}_{\mathcal{T},c}(k)$ is satisfiable.

Proof. (\Rightarrow): Suppose that (\mathcal{T}, c) has a solution $\tau : [0, t(k, n) - 1]^2 \rightarrow T$. Let (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, be a pointed forest satisfying $\text{grid}_{\mathcal{T}}(k)$ (such a pointed forest exists by Corollary 9.37). We slightly modify the valuation \mathcal{V} for the propositional symbols in T , so that the resulting pointed forest satisfies $(\text{one}_{\mathcal{T}})$, $(\text{first}_{\mathcal{T},c})$, $(\text{hor}_{\mathcal{T}})$ and $(\text{vert}_{\mathcal{T}})$. Notice that this does not break the satisfaction of $\text{grid}_{\mathcal{T}}(k)$, as this formula does not contain propositional symbols from T . Since $(\mathcal{K}, w) \models \text{grid}_{\mathcal{T}}(k)$, by Lemma 9.36 every t -node $w' \in R(w)$ encodes a pair of numbers $(n_H(w'), n_V(w')) \in [0, t(k, n) - 1]^2$. Let us consider the valuation \mathcal{V}' obtained from \mathcal{V} as follows:

1. for all $p \in AP \setminus T$, $\mathcal{V}'(p) = \mathcal{V}(p)$,
2. for every $c \in T$ and $w' \in R(w)$, $w' \in \mathcal{V}(c)$ if and only if $\tau(n_H(w'), n_V(w')) = c$.

Let $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}')$. As already stated, from the first part of the definition of \mathcal{V}' we derive $(\mathcal{K}', w) \models \text{grid}_{\mathcal{T}}(k)$. The second part of the definition allows us to conclude that (\mathcal{K}', w) satisfies $(\text{one}_{\mathcal{T}})$, $(\text{first}_{\mathcal{T},c})$, $(\text{hor}_{\mathcal{T}})$ and $(\text{vert}_{\mathcal{T}})$. Indeed, $(\text{one}_{\mathcal{T}})$ holds from the functionality of τ , $(\text{first}_{\mathcal{T},c})$ holds as τ satisfies (first) , and lastly $(\text{hor}_{\mathcal{T}})$ and $(\text{vert}_{\mathcal{T}})$ hold as τ satisfies (hori) and (vert) . Therefore, $(\mathcal{K}', w) \models \text{tiling}_{\mathcal{T},c}(k)$.

(\Leftarrow): Suppose $(\mathcal{K}, w) \models \text{tiling}_{\mathcal{T},c}(k)$, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. Let us consider the relation $\tau \subseteq [0, t(k, n) - 1] \times [0, t(k, n) - 1] \times T$ defined as follows:

$$(i, j, c') \in \tau \text{ if and only if there is } w' \in R(w) \text{ s.t. } n_H(w') = i, n_V(w') = j \text{ and } w' \in \mathcal{V}(c').$$

We have:

- I. from $(\mathcal{K}, w) \models \text{grid}_{\mathcal{T}}(k) \wedge \text{one}_{\mathcal{T}}$ and by Lemmata 9.36 and 9.38, (\mathcal{K}, w) satisfies $(\text{uniq}_{\mathcal{T},k})$ and $(\text{one}_{\mathcal{T}})$, which implies that τ is a (possibly weakly) functional relation, in its first two components, i.e. for every $(i, j) \in [0, t(k, n) - 1]^2$ there is at most one $c' \in T$ such that $(i, j, c') \in \tau$. Moreover, again from $(\mathcal{K}, w) \models \text{grid}_{\mathcal{T}}(k)$ and by Lemma 9.36, (\mathcal{K}, w) satisfies $(\text{zero}_{\mathcal{T},k})$, $(\text{compl[H]}_{\mathcal{T},k})$ and $(\text{compl[V]}_{\mathcal{T},k})$. This implies that τ is total (hence not weakly functional), i.e. for every $(i, j) \in [0, t(k, n) - 1]^2$ there is $c' \in T$ such that $(i, j, c') \in \tau$. Therefore, τ is a map, and below we write $\tau(i, j)$ for the only element in T such that $(i, j, \tau(i, j)) \in \tau$.
- II. From $(\mathcal{K}, w) \models \text{first}_{\mathcal{T},c}(k)$ and Lemma 9.38, (\mathcal{K}, w) satisfies $(\text{first}_{\mathcal{T},c})$, which implies $\tau(0, 0) = c$, Therefore, τ satisfies (first) .
- III. From $(\mathcal{K}, w) \models \text{hor}_{\mathcal{T}}(k) \wedge \text{vert}_{\mathcal{T}}(k)$ and Lemma 9.39, (\mathcal{K}, w) satisfies $(\text{hor}_{\mathcal{T}})$ and $(\text{vert}_{\mathcal{T}})$, which implies that for all $i \in [0, t(k, n) - 1]$ and $j \in [0, t(k, n) - 2]$, $(\tau(j, i), \tau(j + 1, i)) \in H$ and $(\tau(i, j), \tau(i, j + 1)) \in V$. Therefore, τ satisfies (hori) and (vert) .

We conclude that τ is a solution for Tile_k . \square

Theorem 9.40. The satisfiability problem for $\text{ML}(\ast)$ is TOWER-hard.

Proof. As already stated, the size of the formula $\text{type}(k)$ is exponential in $k > 1$ and polynomial in $n \geq 1$, which leads to the size of $\text{tiling}_{\mathcal{T},c}(k)$ being exponential in k and polynomial in n and $\text{card}(T)$. Since $|\text{tiling}_{\mathcal{T},c}(k)|$ is exponential on the input size, whereas the tiling problem Tile_k climbs the exponential hierarchy as k gets bigger (i.e. it is k -NEXPTIME-complete) the uniform reduction from Tile_k to the satisfiability problem of $\text{ML}(\ast)$ provided by Lemma 9.32 entails that the latter problem is TOWER-hard. \square

9.4 REVISITING TOWER-HARD LOGICS WITH $\text{ML}(\ast)$

In this section, we make good use of Theorem 9.40 to (re)prove the TOWER-hardness of two logics interpreted on tree-like structures. In particular, we reprove that the satisfiability problem of the second-order modal logic interpreted on trees QK^t is TOWER-hard, and discover that the same holds for the the modal separation logic featuring only the converse modality \Diamond^{-1} and the separating conjunction \ast . The first reduction, from the satisfiability problem of $\text{ML}(\ast)$ to the one of QK^t , shows that the former problem is TOWER-complete. In order to keep the presentation light, the proofs (all simple) of the results in this section are left in Appendix G.

9.4.1 From $\text{ML}(\ast)$ to the second-order modal logic QK^t .

We already introduced the modal logic QK in Section 8.1.1, and showed that it captures $\text{ML}(\mathbb{I})$. We write QK^t for the logic QK interpreted on trees, which has been proved to admit a TOWER-complete satisfiability problem in [8]. We reprove this result by simply internalising the semantics of $\text{ML}(\ast)$ in QK , and then reduce the satisfiability problem of the former logic to the one of QK^t . Briefly, we recall that the syntax of the formulae in QK is as follows (where $p \in \text{AP}$):

$$\varphi := \top \mid p \mid \varphi \Rightarrow \varphi \mid \neg \varphi \mid \Diamond \varphi \mid \exists p \varphi.$$

The logic QK^t is interpreted on Kripke tree (Definition 4.36), which makes it a syntactical fragment of the quantified computation tree logic QCTL^t introduced in Section 4.4.2. This connection is quite interesting, as in Section 4.4.2 we have shown instead the TOWER-hardness of $\text{QCTL}^t(\text{EU}^0)$ and $\text{QCTL}^t(\text{EF}^1)$, where $\text{QCTL}^t(\text{EU}^0)$ stands for the fragments of QCTL^t featuring only the until modality $\mathbf{E}(\varphi \mathbf{U} \psi)$, which cannot be imbricated, whereas $\text{QCTL}^t(\text{EF}^1)$ is the fragment only allowing the exists-finally modality \mathbf{EF} , which can be imbricated only once. None of these two logics are directly related to QK^t , which is instead the fragment of QCTL^t only featuring the modality $\mathbf{EX} \varphi \equiv \Diamond \varphi$. Indeed, whereas the TOWER-hardness of $\text{QCTL}^t(\text{EU}^0)$ and $\text{QCTL}^t(\text{EF}^1)$ holds for a fixed number of imbrications of the temporal operators, the TOWER-hardness of QK^t requires an arbitrary number of imbrications, as the logic is otherwise elementary when the modal depth of the formula is fixed a priori [99, 8].

We recall the definition of Kripke tree already introduced in Definition 4.36, followed by the semantics of the second-order propositional quantification. The reader refer to Section 8.1.1 for the semantics of QK .

Definition 4.36 (Kripke tree). A Kripke structure $(\mathcal{W}, R, \mathcal{V})$ is a (*finitely-branching*) *Kripke tree* if

1. R^{-1} is functional and acyclic,
2. for every world $w \in \mathcal{W}$, $R(w)$ is finite,
3. it has a *root*, i.e. $R^*(r) = \mathcal{W}$ for some $r \in \mathcal{W}$.

Given a Kripke tree $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and a world $w \in \mathcal{W}$, the semantics of $\exists p \varphi$ is as follows:

$$(\mathcal{K}, w) \models \exists p \varphi \text{ iff there is } \mathcal{W}' \subseteq \mathcal{W} \text{ such that } (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W}']) \models \varphi.$$

where $\mathcal{V}[p \leftarrow \mathcal{W}']$ stands for the valuation obtained from \mathcal{V} by updating the set of worlds satisfying p , from $\mathcal{V}(p)$ to \mathcal{W}' .

As we have done in order to characterise $\text{ML}(\mathbb{I})$ in QK (Section 8.1.1), we rely on the second order quantification in order to simulate the separating conjunction of $\text{ML}(\ast)$, thus internalising the logic in QK (or QK^t). Assume we want to check whether a pointed forest (\mathcal{K}, w) satisfies a

formula φ in $\text{ML}(\ast)$, built over the set of propositions $P \subseteq_{\text{fin}} \text{AP}$. Exactly as in Section 8.1.1, to represent the submodels obtained from \mathcal{K} through the union $+$, we consider the satisfaction of three atomic propositions $Q = \{q_1, q_2, q_3\}$ that do not belong to P . These atomic propositions are used to represent subsets of \mathcal{W} that must be considered when evaluating a formula of $\text{ML}(\ast)$: one of these symbols is used to represent the current set of worlds in the tree, also called *active* universe, whereas the other two auxiliary atomic propositions of Q are used to capture the semantics of \ast . The three symbols are reused when considering multiple application of the operator \ast , since at any time only one symbol encodes the active universe. To formalise this idea, given $\{q_i, q_j, q_k\} = Q$ and $n \in \mathbb{N}$, we define the formula:

$$[q_i = q_j + q_k]^m \stackrel{\text{def}}{=} \boxplus^m((q_i \Leftrightarrow q_j \vee q_k) \wedge \neg(q_j \wedge q_k)).$$

where we recall that $\boxplus^0\varphi = \varphi$ and $\boxplus^{m+1}\varphi = \varphi \wedge \square \boxplus^m(\varphi)$. Essentially, given a *pointed tree* (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a Kripke tree, it satisfies $[q_i = q_j + q_k]^m$ whenever the set of worlds reachable from w in at most m steps that satisfy q_i is partitioned into the set of worlds reachable from w in at most m steps that satisfy q_j and the set of those that satisfy q_k . The formula $[q_i = q_j + q_k]^m$ operates on the worlds reachable from w in at most m steps exactly as the separating conjunction of $\text{ML}(\ast)$. Given a formula φ in $\text{ML}(\ast)$ with modal depth $\text{md}(\varphi) \leq m$, we know that looking at these worlds is enough to check whether it is satisfiable. In fact, we recall that, by Proposition 9.3, a pointed forest $((\mathcal{W}', R', \mathcal{V}'), w)$ satisfies φ if and only if so does $((\mathcal{W}', R'|_{\overline{w}}^{\leq \text{md}(\varphi)}, \mathcal{V}'), w)$. Moreover, notice that $((\mathcal{W}', R'|_{\overline{w}}^{\leq \text{md}(\varphi)}, \mathcal{V}'), w)$ is by definition a Kripke tree, which simplify even further the connections between $\text{ML}(\ast)$ and QK^t .

Before introducing the translation from $\text{ML}(\ast)$ to QK^t , let us take a moment to compare $[q_i = q_j + q_k]^m$ with the analogous formula $[q_i = q_j \mid q_k]$ used in Section 8.1.1 in order to capture $\text{ML}(\mid)$ in QK . The definition of this formula is recalled below:

$$[q_i = q_j \mid q_k] = \square((q_i \Leftrightarrow q_j \vee q_k) \wedge \neg(q_j \wedge q_k)).$$

As we can see, the main difference between $[q_i = q_j + q_k]^m$ and $[q_i = q_j \mid q_k]$ rest on the fact that the former formula inspects the whole structure at distance at most m from the current worlds w , whereas the latter only looks at the children of w . Indeed, we recall that the union $+_w$ of $\text{ML}(\mid)$ can be uniquely defined from a partition in two sets of the children of w , whereas the $+$ of $\text{ML}(\ast)$ requires to partition all the descendants of w , which is part of the reason why we are able to carry out the TOWER-hardness proof of $\text{ML}(\ast)$, while giving an EXPSPACE upper bound on the satisfiability of $\text{ML}(\mid)$.

Given and index $i \in [1, 3]$ denoting which of the symbols in Q is currently used to represent the active children of the current world, the translation $\tau_i(\varphi)$ in QK is defined as shown in Figure 9.15. The translation is straightforward for atomic formulae and Boolean connectives. In the translation of $\varphi_1 * \varphi_2$, we rely on the propositional quantification together with $[q_i = q_j + q_k]^{\text{md}(\varphi_1 * \varphi_2)}$ order to partition the universe. Lastly, for the translation of $\diamond\psi$, we essentially relativise the modality of possibility to only consider children that belong to the active universe. To show the correctness of the translation, we first introduce a suitable encoding between Kripke trees which preserves the satisfaction of formulae in $\text{ML}(\ast)$ with respect to their translation in QK .

Definition 9.41 (Trees as trees). Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$ be two Kripke trees. Let $w \in \mathcal{W}$, $m \in \mathbb{N}$ and $i \in [1, 3]$. We write $\mathcal{K} \triangleright_{m,i}^w \mathcal{K}'$ whenever:

1. \mathcal{W} is finite, $\mathcal{W} \subseteq \mathcal{W}'$ and $R \subseteq R'$,

$$\begin{aligned}
\tau_i(\top) &\stackrel{\text{def}}{=} \top, \\
\tau_i(p) &\stackrel{\text{def}}{=} p, \\
\tau_i(\varphi_1 \Rightarrow \varphi_2) &\stackrel{\text{def}}{=} \tau_i(\varphi_1) \Rightarrow \tau_i(\varphi_2), \\
\tau_i(\neg\varphi) &\stackrel{\text{def}}{=} \neg\tau_i(\varphi), \\
\tau_i(\Diamond\psi) &\stackrel{\text{def}}{=} \Diamond(q_i \wedge \tau_i(\psi)), \\
\tau_i(\varphi_1 * \varphi_2) &\stackrel{\text{def}}{=} \exists q_j \exists q_k ([q_i = q_j * q_k]^{\text{md}(\varphi_1 * \varphi_2)} \wedge \tau_j(\varphi_1) \wedge \tau_k(\varphi_2)), \\
&\quad \text{where } j, k \in [1, 3], j < k \text{ and } j \neq i \neq k.
\end{aligned}$$

Figure 9.15: Translation from $\text{ML}(\ast)$ to QK .

2. for all $p \in \text{AP} \setminus \{q_1, q_2, q_3\}$, the satisfaction of p is preserved, i.e. $\mathcal{V}(p) = \mathcal{V}'(p) \cap \mathcal{W}$,
3. the worlds reachable in at most m steps in R are, among the ones reachable in at most m steps in R' , exactly those in $\mathcal{V}'(q_i)$, i.e. $\bigcup_{j \in [1, m]} R'^j(\mathbf{w}) = \{\mathbf{w}' \in R'^m(\mathbf{w}) \cap \mathcal{V}'(q_i) \mid j \in [1, m]\}$.

Definition 9.41 formalises the fact that whereas the separating conjunction split the structure, the second-order quantification of QK simply colours it. In particular, the condition (3) essentially states that, when $\mathcal{K} \triangleright_{m,i}^{\mathbf{w}} \mathcal{K}'$ holds, the accessibility relation of \mathcal{K}' relativised to the propositional symbol q_j should correspond to the accessibility relation of \mathcal{K} .

Lemma 9.42. Let φ be a formula in $\text{ML}(\ast)$ written without using atomic propositions from $\{q_1, q_2, q_3\}$, and let $i \in [1, 3]$ and $m \geq \text{md}(\varphi)$. Let \mathcal{K} and \mathcal{K}' be two Kripke trees and \mathbf{w} be a world, so that $\mathcal{K} \triangleright_{m,i}^{\mathbf{w}} \mathcal{K}'$. $(\mathcal{K}, \mathbf{w}) \models \varphi$ in $\text{ML}(\ast)$ if and only if $(\mathcal{K}', \mathbf{w}) \models \tau_i(\varphi)$ in QK^t .

Notice that Lemma 9.42 (which is proved in Appendix G, by structural induction on φ) is given with respect to the class of Kripke trees having a finite universe \mathcal{W} (because of $\mathcal{K} \triangleright_{m,i}^{\mathbf{w}} \mathcal{K}'$), which is a subclass of both Kripke-style finite forests and (general) Kripke tree. As already stated, by Proposition 9.3, we know that considering structures from this class is enough to check whether a formula of $\text{ML}(\ast)$ is satisfiable. This also implies that, applying Lemma 9.42, a formula φ in $\text{ML}(\ast)$ is satisfiable if and only if $\tau_1(\varphi)$ in QK^t is satisfiable. From Theorem 9.40, this reproves the TOWER-hardness of QK^t . Reciprocally, the fact that the satisfiability problem of QK^t is decidable in TOWER [8] carries over to the satisfiability problem of $\text{ML}(\ast)$.

Theorem 9.43. The satisfiability problem of QK^t and $\text{ML}(\ast)$ are TOWER-complete.

9.4.2 From $\text{ML}(\ast)$ to modal separation logic with converse.

In this small section, we show a simple reduction from the satisfiability problem of $\text{ML}(\ast)$ to the satisfiability problem of $\text{MSL}(\ast, \Diamond^{-1})$, that is the fragment of modal separation logic (Section 2.3.2) featuring Boolean connectives, atomic propositions, the separating conjunction $*$ and the *converse modality of possibility* \Diamond^{-1} . This allows us to conclude that $\text{MSL}(\ast, \Diamond^{-1})$ admits a TOWER-complete satisfiability problem, closing the complexity gap from [54], where the problem is shown to be PSPACE-hard and decidable in TOWER. Besides, this section completes our investigation on TOWER-hard fragments of MSL started in Section 4.4.3, where we have

-
- $(\mathcal{K}, w) \models p$ iff $w \in \mathcal{V}(p)$,
 $(\mathcal{K}, w) \models \Diamond^{-1}\varphi$ iff there is $w' \in \mathcal{W}$ such that $(w', w) \in R$ and $(\mathcal{K}, w') \models \varphi$,
 $(\mathcal{K}, w) \models \varphi * \psi$ iff $(\mathcal{K}_1, w) \models \varphi$ and $(\mathcal{K}_2, w) \models \psi$ for some $\mathcal{K}_1, \mathcal{K}_2$ s.t. $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$.

Figure 9.16: Satisfaction relation for $\text{MSL}(*, \Diamond^{-1})$.

shown that the modal separation logic without atomic propositions and featuring Boolean connectives, the universal modality $\langle U \rangle$, the separating conjunction and the modality \Diamond admits a TOWER-complete satisfiability problem.

The formulae of $\text{MSL}(*, \Diamond^{-1})$ are defined from the grammar below (where $p \in \text{AP}$):

$$\varphi := \top \mid p \mid \varphi \wedge \varphi \mid \neg \varphi \mid \Diamond^{-1}\varphi \mid \varphi * \varphi .$$

As discussed in Section 2.3.2, modal separation logic is interpreted on a class of Kripke structures that is inspired from the memory states used in separation logic, called Kripke-style finite functions and recalled below.

Definition 2.17 (Kripke-style finite function). A *(Kripke-style) finite function* $(\mathcal{W}, R, \mathcal{V})$ is a triple where \mathcal{W} is a countably infinite set of *worlds*, $R \subseteq \mathcal{W} \times \mathcal{W}$ is a finite weakly functional¹ accessibility relation, and $\mathcal{V} : \text{AP} \rightarrow 2^{\mathcal{W}}$ is a *labelling function* assigning to every propositional symbol p the set of worlds satisfying it.

Essentially, in a Kripke-style finite function $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, R is a heap. Differently from the accessibility relation of a Kripke-style finite forests, R is functional, and can a priori contain cycles. The semantics of $\text{MSL}(*, \Diamond^{-1})$ is recalled in Figure 9.16 (omitting standard cases for \top and Boolean connectives). Notice that the union $+$ characterising the semantics of $\varphi * \psi$ is defined exactly as in $\text{ML}(*)$. The connection between $\text{ML}(*)$ and $\text{MSL}(*, \Diamond^{-1})$ lies on the fact that the modality \Diamond^{-1} traverses R backwards, so that its inverse R^{-1} is instead considered, which is almost as a Kripke-style finite forest, if it was not for the possibility of encounter a cycle. However, notice that given a Kripke-style finite forest $(\mathcal{W}', R', \mathcal{V}')$, the structure $(\mathcal{W}, R^{-1}, \mathcal{V}')$ is an acyclic Kripke-style finite function. Therefore, to translate a formula φ of $\text{ML}(*)$ into a formula of $\text{MSL}(*, \Diamond^{-1})$ we simply replace the modality \Diamond with its converse \Diamond^{-1} , obtaining the formula $\varphi[\Diamond \leftarrow \Diamond^{-1}]$. With a straightforward induction hypothesis (see Appendix G), one can show that this translation is correct.

Lemma 9.44. Let (\mathcal{K}, w) be a Kripke-style finite forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, and let φ be a formula in $\text{ML}(*)$. $(\mathcal{K}, w) \models \varphi$ in $\text{ML}(*)$ iff $((\mathcal{W}, R^{-1}, \mathcal{V}), w) \models \varphi[\Diamond \leftarrow \Diamond^{-1}]$ in $\text{MSL}(*, \Diamond^{-1})$.

In order to conclude the reduction from $\text{ML}(*)$ to $\text{MSL}(*, \Diamond^{-1})$, it is now sufficient to restrict ourselves to a subclass of Kripke-style finite functions that is, in some sense, acyclic. From Proposition 9.3, we know that only the set of worlds reachable from the current one in at most $\text{md}(\varphi)$ steps influences the satisfiability of a formula φ in $\text{ML}(*)$. Therefore, a suitable subclass is given by those Kripke-style finite functions where the current world is not reached by any world in more than $\text{md}(\varphi) + 1$ steps. In formula: $\Box^{-(\text{md}(\varphi)+1)} \perp$, where $\Box^{-1}\psi \stackrel{\text{def}}{=} \neg \Diamond^{-1}\neg \psi$ and for every $n \geq 1$, $\Box^{-(n+1)}\psi \stackrel{\text{def}}{=} \Box^{-1}\Box^{-n}\psi$.

¹ R is finite and for every $w, w', w'' \in \mathcal{W}$, if $(w, w') \in R$ and $(w, w'') \in R$ then $w' = w''$.

Lemma 9.45. φ in $\text{ML}(\ast)$ is satisfiable iff so is $\varphi[\Diamond \leftarrow \Diamond^{-1}] \wedge \Box^{-(\text{md}(\varphi)+1)} \perp$ in $\text{MSL}(\ast, \Diamond^{-1})$.

Lemma 9.45 closes the complexity gap left open in [54].

Theorem 9.46. The satisfiability problem of $\text{MSL}(\ast, \Diamond^{-1})$ is TOWER-complete.

Conclusion

In Chapters 8 and 9, we relied on the two modal logics $\text{ML}(\|)$ and $\text{ML}(*)$ to carry out an in-depth comparison between the composition operator $\|$ from ambient logic and the separating conjunction $*$ from separation logic. We have not only characterised the expressive power and the complexity for both logics, but also identified remarkable differences and export our results to other logics, such as static ambient logic and modal separation logics.

$\text{ML}(\|)$, which was shown in Chapter 6 to be as expressive as graded modal logic, enjoys an AEXP_{POL}-complete satisfiability problem. The AEXP_{POL} upper bound was obtained by showing an exponential-size model property, whereas hardness was proved by reducing the satisfiability problem for the propositional team logic $\text{PL}(\sim)$ [85] (another instantiation of BBI). Besides the obvious similarities between $\text{ML}(\|)$ and $\text{ML}(*)$, these results are counter-intuitive: whereas the logic $\text{ML}(*)$ is shown to be strictly less expressive than graded modal logic (and consequently, than $\text{ML}(\|)$), its satisfiability problem is TOWER-complete. This surprising result rests on the fact that, in Section 9.3, we are able to define a very concise formula φ encoding a grid whose size grows non-elementarily with respect to the modal depth of φ .

Why the separating conjunction makes Modal Logics robustly hard?

Both $\text{ML}(\|)$ and $\text{ML}(*)$ are fragments of the second-order modal logic QK, whose satisfiability for the class of Kripke trees has been recently shown TOWER-hard in [8]. In Section 9.4 we reproved this result thanks to a simple reduction from the satisfiability problem of $\text{ML}(*)$. The main formula that allows us translate the formula $\varphi * \psi$ into a formula of QK is $[q_3 = q_1 + q_2]^{\text{md}(\varphi * \psi)}$, which essentially simulates the union $+$ used by $\text{ML}(*)$ to reason on submodels. More precisely, given a Kripke tree $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and a (current) world $w \in \mathcal{W}$, this formula partitions the worlds satisfying q_3 , and reachable from w in at most $\text{md}(\varphi * \psi)$ steps, in two sets encoded using the atomic propositions q_1 and q_2 . By contrast, to reduce $\text{ML}(\|)$ to QK (Section 8.1.1), we simulated the union $+_w$, used in $\text{ML}(\|)$ to reason on submodel, with the formula $[q_3 = q_1 \| q_2]$, which only partition the children of w in two sets. This intuitively means that the union $+_w$ is completely determined by how the children of w are partitioned.

To further investigate the essence of the gap between the complexities of $\text{ML}(\|)$ and $\text{ML}(*)$, a natural direction is to look at a hierarchy of unions $+_w^k$, where k is a positive natural number, which can be characterised in QK with the formula $[q_3 = q_1 + q_2]^k$. Notice that k is now fixed, whereas in the case of $\text{ML}(*)$ it depends on the modal depth of the formula we wish to translate. Moreover, for $k = 1$ we obtain exactly the formula $[q_3 = q_1 \| q_2]$. Explicitly, the union $+_w^k$ shall be defined on Kripke-style finite forests $\mathcal{K}_i = (\mathcal{W}_i, R_i, \mathcal{V}_i)$ ($i \in [1, 3]$) as follows:

$$(\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3) \in +_w^k \text{ iff } \mathcal{W}_1 = \mathcal{W}_2 = \mathcal{W}_3 = \mathcal{W}, \mathcal{V}_1 = \mathcal{V}_2 = \mathcal{V}_3, R_1 \cap R_2 = \emptyset, R_1 \cup R_2 = R_3, \\ \text{and for all } i \in \{1, 2\} \text{ and } w' \in R_i^k(w), R_i^+(w') = R_3^+(w').$$

Informally, whenever $(\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3) \in +_w^k$ holds, every world that in either \mathcal{K}_1 or \mathcal{K}_2 is reachable from w in k steps, must have the same subtree that it has in \mathcal{K}_3 . As expected, $+_w^1$ corresponds to the union $+_w$ of $\text{ML}(\mathsf{I})$. Let us write $\text{ML}(*_k)$ for the logic obtained from $\text{ML}(*)$ by swapping the union $+$ with the union $+_w^k$. Interestingly, in $\text{ML}(*_k)$ we can still perform the reduction of Tile_{k-2} of Section 9.3, as the formula we obtain has modal depth k . However, the translation does not work when considering Tile_j with $j > k$, as formulae like $[\mathbf{ax} < \mathbf{bx}]_j^{j-1}$ require to break the subtree of worlds that are reachable from the current one in more than k steps.

In view of the complexity of $\text{ML}(\mathsf{I})$, we conjecture the satisfiability problem of $\text{ML}(*_k)$ to be k -AEXP_{POL}-complete, where k -AEXP_{POL} is the class of decision problems solvable by an alternating Turing machine running in k -EXPTIME and alternating between existential and universal states a polynomial amount of times. This result, if proved, shows that rather than a huge gap going from AEXP_{POL} to TOWER, the logics $\text{ML}(\mathsf{I})$ and $\text{ML}(*)$ belongs to a hierarchy of modal logics characterised by their composition operator $+_w^k$.

ML(I) with guarantee operator and iteration.

Even though the goal of Chapters 8 and 9 is to compare $\text{ML}(\mathsf{I})$ and $\text{ML}(*)$, both logics are certainly interesting by themselves. In Section 8.4 we have shown that $\text{ML}(\mathsf{I})$ is essentially equivalent (modulo technical changes in the model) to the static ambient logic $\text{SAL}(\mathsf{I})$. From this, we were able to deduce that the static ambient logic SAL from [34] admits an AEXP_{POL}-hard satisfiability problem. Up to our knowledge, the best upper bound for this logic is 2EXPTIME, which was proved in [47] by relying on interesting connections between static ambient logic and Presburger arithmetic. Actually, the logic considered in [47] not only features the guarantee operator \triangleright of SAL (i.e. the right adjoint of the operator I), but also the Kleene closure which corresponds to iterated applications of I . This means that the logic satisfies the axioms of commutative Kleene algebras, which (on an abstract level) are known to correspond to the space of semi-linear sets [30] or, equivalently, the set of Presburger definable formulae [77]. These last two results have recently gained a significant importance, as Kleene algebras and Presburger arithmetic offer the basis of several state-of-the-art formalisms for automatic verification (see [97, 130, 131] for Kleene algebras; the existential fragment of Presburger arithmetic is implemented in both the SMT solvers CVC4 [5] and Z3 [50]). As ambient logic is geared towards distributed systems, the logic in [47] offers a different view on these topics, which we hope to further investigate. To start, one can hope that the refined analysis of the composition operator I given in Section 8.2 can be extended for the logic in [47], leading to a more efficient decision procedure.

If $\text{ML}(*)$ can tile a huge grid, then other logics can do it too.

Our interest in $\text{ML}(*)$ is mainly theoretical: following the research agenda started with modal separation logics [52], the logic emphasises similarities between separation logics and modal logics. For instance, Section 9.2 uniquely relies on standard tools from modal logic and finite model theory, such as g-bisimulations and Ehrenfeucht-Fraïssé games, in order to study the expressive power of $\text{ML}(*)$. After taking advantage of these tools, we hope that $\text{ML}(*)$ will in return shed new light on the complexity of other modal logics. In this regard, we find reduction from the tiling problem Tile_k to the satisfiability problem form $\text{ML}(*)$ fascinating. Not only it reproves the TOWER-hardness of QK^t from [8], but also shows a set of features that leads to non-elementary logics. The central one is probably the notion of local nominals, thanks to which we are able to encode and compare the positions of the grid to be tiled. Other logics

can easily express the notion of local nominals, such as sabotage modal logic [4] and first-order modal logic [21]. We believe that the proof of TOWER-hardness of $\text{ML}(\ast)$ can be adapted to show the TOWER-hardness of both these logics, under their interpretation on Kripke tree. This is not completely satisfactory, as we would like to rely directly on TOWER-hardness of $\text{ML}(\ast)$, without passing through Tile_k . However, we struggle to see easy reductions between $\text{ML}(\ast)$ and sabotage or first-order modal logic (in both directions), which leaves us with the question of whether there is a simpler TOWER-hard logic that is easily captured by all these logics.

References

- [1] T. Antonopoulos and A. Dawar, “Separating graph logic from MSO,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 5504. Springer, 2009, pp. 63–77.
- [2] C. Areces, P. Blackburn, and M. Marx, “Hybrid logics: Characterization, interpolation and complexity,” *The Journal of Symbolic Logic*, vol. 66, pp. 977–1010, 2001.
- [3] G. Aucher, P. Balbiani, L. Fariñas del Cerro, and A. Herzig, “Global and local graph modifiers,” *Electronic Notes in Theoretical Computer Science*, vol. 231, pp. 293–307, 2009.
- [4] G. Aucher, J. van Benthem, and D. Grossi, “Sabotage modal logic: Some model and proof theoretic aspects,” in *Logic, Rationality, and Interaction*, ser. LNCS, vol. 9394. Springer, 2015, pp. 1–13.
- [5] C. W. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli, “CVC4,” in *Computer-Aided Verification*, ser. LNCS, vol. 6806. Springer, 2011, pp. 171–177.
- [6] G. Barthe, J. Hsu, and K. Liao, “A probabilistic separation logic,” *ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, vol. 4, pp. 55:1–55:30, 2020.
- [7] K. Batz, B. L. Kaminski, J. Katoen, C. Matheja, and T. Noll, “Quantitative separation logic: a logic for reasoning about probabilistic pointer programs,” *ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, vol. 3, pp. 34:1–34:29, 2019.
- [8] B. Bednarczyk and S. Demri, “Why propositional quantification makes modal logics on trees robustly hard?” in *Logic in Computer Science*. IEEE, 2019, pp. 1–13.
- [9] B. Bednarczyk, S. Demri, R. Fervari, and A. Mansutti, “Modal logics with composition on finite forests: Expressivity and complexity,” in *Logic in Computer Science*. ACM, 2020, pp. 167–180.
- [10] J. Berdine, C. Calcagno, and P. O’Hearn, “A decidable fragment of separation logic,” in *Foundations of Software Technology and Theoretical Computer Science*, ser. LNCS, vol. 3328. Springer, 2004, pp. 97–109.
- [11] J. Berdine, C. Calcagno, and P. W. O’Hearn, “Symbolic execution with separation logic,” in *Asian Symposium on Programming Languages and Systems*, ser. LNCS, vol. 3780. Springer, 2005, pp. 52–68.

- [12] J. Berdine, B. Cook, and S. Ishtiaq, “Slayer: Memory safety for systems-level code,” in *Computer-Aided Verification*, ser. LNCS, vol. 6806. Springer, 2011, pp. 178–183.
- [13] J. Berdine, C. Calcagno, and P. W. O’Hearn, “Smallfoot: Modular automatic assertion checking with separation logic,” in *FMCO*, ser. Lecture Notes in Computer Science, vol. 4111. Springer, 2005, pp. 115–137.
- [14] A. Bijlsma, “Calculating with pointers,” *Science of Computer Programming*, vol. 12, no. 3, pp. 191–205, 1989.
- [15] P. Blackburn, M. de Rijke, and Y. Venema, *Modal Logic*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University, 2001.
- [16] B. Blanchet and D. Pointcheval, “Automated security proofs with sequences of games,” in *International Cryptology Conference*, ser. LNCS, vol. 4117. Springer, 2006, pp. 537–554.
- [17] E. Börger, E. Grädel, and Y. Gurevich, *The Classical Decision Problem*, ser. Perspectives in Mathematical Logic. Springer, 1997.
- [18] L. Borkowski and J. Ślupecki, “The logical works of j. Lukasiewicz,” *Studia Logica*, vol. 8, pp. 7–56, 1958.
- [19] M. Bozga, R. Iosif, and S. Perarnau, “Quantitative separation logic and programs with lists,” *Journal of Automated Reasoning*, vol. 45, no. 2, pp. 131–156, 2010.
- [20] L. Bozzelli, H. van Ditmarsch, and S. Pinchinat, “The complexity of one-agent refinement modal logic,” *Theoretical Computer Science*, vol. 603, pp. 58–83, 2015.
- [21] T. Braüner and S. Ghilardi, “First-order modal logic,” in *Handbook of Modal Logic*, ser. Studies in logic and practical reasoning. North-Holland, 2007, vol. 3, pp. 549–620.
- [22] R. Brochenin, S. Demri, and E. Lozes, “On the almighty wand,” *Information and Computation*, vol. 211, pp. 106–137, 2012.
- [23] S. Brookes and P. W. O’Hearn, “Concurrent separation logic,” *ACM SIGLOG News*, vol. 3, no. 3, pp. 47–65, 2016.
- [24] S. Brookes, “A semantics for concurrent separation logic,” *Theoretical Computer Science*, vol. 375, no. 1-3, pp. 227–270, 2007.
- [25] J. Brotherston, “Bunched logics displayed,” *Studia Logica*, vol. 100, no. 6, pp. 1223–1254, 2012.
- [26] J. Brotherston and M. Kanovich, “On the complexity of pointer arithmetic in separation logic,” in *Asian Symposium on Programming Languages and Systems*, ser. LNCS, vol. 11275. Springer, 2018, pp. 329–349.
- [27] J. Brotherston and J. Villard, “Parametric completeness for separation theories,” in *ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 2014, pp. 453–464.

- [28] J. Brotherton, C. Fuhs, J. A. N. Pérez, and N. Gorogiannis, “A decision procedure for satisfiability in separation logic with inductive predicates,” in *CSL-LICS*. ACM, 2014, pp. 25:1–25:10.
- [29] J. Brotherton, N. Gorogiannis, and M. Kanovich, “Biabduction (and related problems) in array separation logic,” in *Conference on Automated Deduction*, ser. LNCS, vol. 10395. Springer, 2017, pp. 472–490.
- [30] P. Brunet, “A note on commutative Kleene algebra,” *CoRR*, vol. abs/1910.14381, 2019.
- [31] R. M. Burstall, “Some techniques for proving correctness of programs which alter data-structures,” *Machine Intelligence*, vol. 7, pp. 23–50, 1972.
- [32] L. Caires and L. Cardelli, “A spatial logic for concurrency (part I),” *Information and Computation*, vol. 186, no. 2, pp. 194–235, 2003.
- [33] C. Calcagno, H. Yang, and P. W. O’Hearn, “Computability and complexity results for a spatial assertion language for data structures,” in *Foundations of Software Technology and Theoretical Computer Science*, ser. LNCS, vol. 2245. Springer, 2001, pp. 108–119.
- [34] C. Calcagno, L. Cardelli, and A. D. Gordon, “Deciding validity in a spatial logic for trees,” in *International Workshop on Types in Languages Design and Implementation*. ACM, 2003, pp. 62–73.
- [35] C. Calcagno, P. Gardner, and M. Hague, “From separation logic to first-order logic,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 3441. Springer, 2005, pp. 395–409.
- [36] C. Calcagno, T. Dinsdale-Young, and P. Gardner, “Adjunct elimination in context logic for trees,” *Information and Computation*, vol. 208, pp. 474–499, 2010.
- [37] C. Calcagno, D. Distefano, P. W. O’Hearn, and H. Yang, “Compositional shape analysis by means of bi-abduction,” *Journal of the Association for Computing Machinery*, vol. 58, pp. 26:1–26:66, 2011.
- [38] C. Calcagno, D. Distefano, J. Dubreil, D. Gabi, P. Hooimeijer, M. Luca, P. W. O’Hearn, I. Papakonstantinou, J. Purbrick, and D. Rodriguez, “Moving fast with software verification,” in *Nasa Formal Methods*, ser. LNCS, vol. 9058. Springer, 2015, pp. 3–11.
- [39] L. Cardelli and A. D. Gordon, “Anytime, anywhere: Modal logics for mobile ambients,” in *ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 2000, pp. 365–377.
- [40] ——, “Mobile ambients,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 1378. Springer, 1998, pp. 140–155.
- [41] A. K. Chandra, D. Kozen, and L. J. Stockmeyer, “Alternation,” *Journal of the Association for Computing Machinery*, vol. 28, no. 1, pp. 114–133, 1981.
- [42] E. M. Clarke, “The birth of model checking,” in *25 Years of Model Checking: History, Achievements, Perspectives*. Springer, 2008, pp. 1–26.

- [43] E. M. Clarke and E. A. Emerson, “Design and synthesis of synchronization skeletons using branching time temporal logic,” in *Logics of Programs*, ser. LNCS, vol. 131. Springer, 1982, pp. 52–71.
- [44] B. Cook, C. Haase, J. Ouaknine, M. J. Parkinson, and J. Worrell, “Tractable reasoning in a fragment of separation logic,” in *Concurrency Theory*, ser. LNCS, vol. 6901. Springer, 2011, pp. 235–249.
- [45] J. Courtault and D. Galmiche, “A modal separation logic for resource dynamics,” *Journal of Logic and Computation*, vol. 28, no. 4, pp. 733–778, 2018.
- [46] J. Courtault, D. Galmiche, and D. J. Pym, “A logic of separating modalities,” *Theoretical Computer Science*, vol. 637, pp. 30–58, 2016.
- [47] S. Dal Zilio, D. Lugiez, and C. Meyssonier, “A logic you can count on,” in *ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 2004, pp. 135–146.
- [48] A. Dawar, P. Gardner, and G. Ghelli, “Adjunct elimination through games in static ambient logic,” in *Foundations of Software Technology and Theoretical Computer Science*, ser. LNCS, vol. 3328. Springer, 2004, pp. 211–223.
- [49] F. De Caro, “Graded modalities. II,” *Studia Logica*, vol. 47, pp. 1–10, 1988.
- [50] L. M. de Moura and N. Bjørner, “Z3: an efficient SMT solver,” in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS, vol. 4963. Springer, 2008, pp. 337–340.
- [51] M. de Rijke, “A note on graded modal logic,” *Studia Logica*, vol. 64, no. 2, pp. 271–283, 2000.
- [52] S. Demri and M. Deters, “Two-variable separation logic and its inner circle,” *Transactions on Computational Logic*, vol. 16, pp. 15:1–15:36, 2015.
- [53] ——, “Expressive completeness of separation logic with two variables and no separating conjunction,” *Transactions on Computational Logic*, pp. 1–44, 2016.
- [54] S. Demri and R. Fervari, “On the complexity of modal separation logics,” in *Advances in Modal Logic*. College Publications, 2018, pp. 179–198.
- [55] S. Demri, D. Galmiche, D. Larchey-Wendling, and D. Méry, “Separation logic with one quantified variable,” *Theory of Computing Systems*, vol. 61, pp. 371–461, 2017.
- [56] S. Demri, E. Lozes, and A. Mansutti, “The effects of adding reachability predicates in propositional separation logic,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 10803. Springer, 2018, pp. 476–493.
- [57] S. Demri, R. Fervari, and A. Mansutti, “Axiomatising logics with separating conjunction and modalities,” in *Logics in Artificial Intelligence*, ser. LNCS, vol. 11468. Springer, 2019, pp. 692–708.

- [58] S. Demri, E. Lozes, and A. Mansutti, “Internal calculi for separation logics,” in *Computer Science Logic*, ser. LIPIcs, vol. 152. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020, pp. 19:1–19:18.
- [59] S. Docherty and D. Pym, “Modular tableaux calculi for separation theories,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 10803. Springer, 2018, pp. 441–458.
- [60] S. Docherty, “Bunched logics: a uniform approach,” Ph.D. dissertation, University College London, 2019.
- [61] A. Doumane, “Constructive completeness for the linear-time μ -calculus,” in *Logic in Computer Science*. IEEE, 2017, pp. 1–12.
- [62] M. Echenim, R. Iosif, and N. Peltier, “Prenex separation logic with one selector field,” in *Automated Reasoning with Analytic Tableaux and Related Methods*, ser. LNCS, vol. 11714. Springer, 2019, pp. 409–427.
- [63] ——, “The Bernays-Schönfinkel-Ramsey class of separation logic on arbitrary domains,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 11425. Springer, 2019, pp. 242–259.
- [64] ——, “Entailment checking in separation logic with inductive definitions is 2-EXPTIME hard,” in *LPAR*, ser. EPiC Series in Computing, vol. 73. EasyChair, 2020, pp. 191–211.
- [65] C. Enea, O. Lengál, M. Sighireanu, and T. Vojnar, “Compositional entailment checking for a fragment of separation logic,” *Formal Methods in System Design*, vol. 51, pp. 575–607, 2017.
- [66] M. Fattorosi-Barnaba and F. De Caro, “Graded modalities. I,” *Studia Logica*, vol. 44, no. 2, pp. 197–221, 1985.
- [67] J. Ferrante and C. Rackoff, “A decision procedure for the first order theory of real addition with order,” *SIAM Journal of Computing*, vol. 4, no. 1, pp. 69–76, 1975.
- [68] R. Fervari and F. R. Velázquez-Quesada, “Introspection as an action in relational models,” vol. 108, pp. 1–23, 2019.
- [69] K. Fine, “Propositional quantifiers in modal logic,” *Theoria*, vol. 36, pp. 336–346, 1970.
- [70] ——, “In so many possible worlds,” *Notre Dame Journal of Formal Logic*, vol. 13, no. 4, pp. 516–520, 10 1972.
- [71] R. W. Floyd, “Assigning meanings to programs,” in *Proceedings of the American Mathematical Society Symposia on Applied Mathematics*, 1967.
- [72] D. Gabbay, *Labelled Deductive Systems*. Oxford University, 1996.
- [73] H. Gaifman, “On local and non-local properties,” *Studies in Logic and the Foundations of Mathematics*, vol. 107, pp. 105–135, 1982.
- [74] D. Galmiche and D. Méry, “Tableaux and resource graphs for separation logic,” *Journal of Logic and Computation*, vol. 20, no. 1, pp. 189–231, 2010.

- [75] D. Galmiche and D. Larchey-Wendling, “Expressivity properties of boolean BI through relational models,” in *Foundations of Software Technology and Theoretical Computer Science*, ser. LNCS, vol. 4337. Springer, 2006, pp. 357–368.
- [76] S. Genaim and D. Zanardini, “Inference of field-sensitive reachability and cyclicity,” *Transactions on Computational Logic*, vol. 15, pp. 1–41, 2014.
- [77] S. Ginsburg and E. H. Spanier, “Semigroups, Presburger formulas, and languages,” *Pacific Journal of Mathematics*, vol. 16, no. 2, pp. 285–296, 1966.
- [78] J. Girard, “Linear logic,” *Theoretical Computer Science*, vol. 50, pp. 1–102, 1987.
- [79] L. F. Goble, “Grades of modality,” *Logique et Analyse*, vol. 13, no. 51, pp. 323–334, 1970.
- [80] V. Goranko and S. Passy, “Using the universal modality: Gains and questions,” *Journal of Logic and Computation*, vol. 2, pp. 5–30, 1992.
- [81] V. Goranko and G. van Drimmelen, “Complete axiomatization and decidability of alternating-time temporal logic,” *Theoretical Computer Science*, vol. 353, no. 1-3, pp. 93–117, 2006.
- [82] V. Goranko, A. Montanari, and G. Sciavicco, “A road map of interval temporal logics and duration calculi,” *Journal of Applied Non-Classical Logics*, vol. 14, pp. 9–54, 2004.
- [83] E. Grädel and M. Otto, “Guarded teams: The horizontally guarded case,” in *Computer Science Logic*, ser. LIPIcs, vol. 152. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020, pp. 22:1–22:17.
- [84] E. Grädel, P. G. Kolaitis, and M. Y. Vardi, “On the decision problem for two-variable first-order logic,” *Bull. Symb. Log.*, vol. 3, no. 1, pp. 53–69, 1997.
- [85] M. Hannula, J. Kontinen, J. Virtema, and H. Vollmer, “Complexity of propositional logics in team semantic,” *Transactions on Computational Logic*, vol. 19, no. 1, pp. 2:1–2:14, 2018.
- [86] M. Hennessy and R. Milner, “On observing nondeterminism and concurrency,” in *Automata, Languages and Programming*, ser. LNCS, vol. 85. Springer, 1980, pp. 299–309.
- [87] C. A. R. Hoare, “An axiomatic basis for computer programming,” *Communications of the Association for Computing Machinery*, vol. 12, no. 10, pp. 576–580, 1969.
- [88] W. Hodges, “Compositional semantics for a language of imperfect information,” *Log. J. IGPL*, vol. 5, no. 4, pp. 539–563, 1997.
- [89] ———, *Some strange quantifiers*, ser. LNCS. Springer, 1997, pp. 51–65.
- [90] Z. Hóu, R. Goré, and A. Tiu, “Automated theorem proving for assertions in separation logic with all connectives,” in *Conference on Automated Deduction*, ser. LNCS, vol. 9195. Springer, 2015, pp. 501–516.
- [91] Z. Hóu, R. Clouston, R. Goré, and A. Tiu, “Modular labelled sequent calculi for abstract separation logics,” *Transactions on Computational Logic*, vol. 19, no. 2, pp. 13:1–13:35, 2018.

- [92] R. Iosif, A. Rogalewicz, and T. Vojnar, “Deciding entailments in inductive separation logic with tree automata,” in *Automated Technology for Verification and Analysis*, ser. LNCS, vol. 8837. Springer, 2014, pp. 201–218.
- [93] C. Jansen, J. Katelaan, C. Matheja, T. Noll, and F. Zuleger, “Unified reasoning about robustness properties of symbolic-heap separation logic,” in *European Symposium on Programming*, ser. LNCS, vol. 10201. Springer, 2017, pp. 611–638.
- [94] R. Jung, R. Krebbers, J. Jourdan, A. Bizjak, L. Birkedal, and D. Dreyer, “Iris from the ground up: A modular foundation for higher-order concurrent separation logic,” *J. Funct. Program.*, vol. 28, p. e20, 2018.
- [95] R. Kaivola, “Axiomatising linear time mu-calculus,” in *Concurrency Theory*, ser. LNCS, vol. 962. Springer, 1995, pp. 423–437.
- [96] T. Kowaltowski, “Data structures and correctness of programs,” *Journal of the Association for Computing Machinery*, vol. 26, no. 2, p. 283–301, 1979.
- [97] D. Kozen, “On Hoare logic and Kleene algebra with tests,” *Transactions on Computational Logic*, vol. 1, no. 1, pp. 60–76, 2000.
- [98] S. A. Kripke, “Semantical considerations on modal logic,” *Acta Philosophica Fennica*, vol. 16, pp. 83–94, 1963.
- [99] F. Laroussinie and N. Markey, “Quantified CTL: expressiveness and complexity,” *Logical Methods in Computer Science*, vol. 10, 2014.
- [100] K. Larsen, R. Mardare, and B. Xue, “Probabilistic mu-calculus: Decidability and complete axiomatization,” in *Foundations of Software Technology and Theoretical Computer Science*, ser. LIPIcs, vol. 65. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 25:1–25:18.
- [101] Q. L. Le, M. Tatsuta, J. Sun, and W. Chin, “A decidable fragment in separation logic with inductive predicates and arithmetic,” in *Computer-Aided Verification*, ser. LNCS, vol. 10427. Springer, 2017, pp. 495–517.
- [102] L. Libkin, *Elements of Finite Model Theory*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.
- [103] E. Lozes, “Adjuncts elimination in the static ambient logic,” *Electronic Notes in Theoretical Computer Science*, vol. 96, pp. 51–72, 2004.
- [104] ———, “Elimination of spatial connectives in static spatial logics,” *Theoretical Computer Science*, vol. 330, pp. 475–499, 2005.
- [105] M. Lück, “Axiomatizations of team logics,” *Annals of Pure and Applied Logic*, vol. 169, no. 9, pp. 928–969, 2018.
- [106] D. C. Luckham and N. Suzuki, “Verification of array, record, and pointer operations in pascal,” *Transactions on Programming Languages and Systems*, vol. 1, no. 2, pp. 226–244, 1979.

- [107] A. Mansutti, “Extending propositional separation logic for robustness properties,” in *Foundations of Software Technology and Theoretical Computer Science*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 42:1–42:23.
- [108] ———, “An auxiliary logic on trees: on the TOWER-hardness of logics featuring reachability and submodel reasoning,” in *Foundations of Software Science and Computational Structures*, ser. LNCS, vol. 12077. Springer, 2020, pp. 462–481.
- [109] D. A. Martin, “Borel determinacy,” *Annals of Mathematics*, vol. 102, pp. 363–371, 1975.
- [110] A. R. Meyer, “Weak monadic second order theory of successor is not elementary-recursive,” in *Logic Colloquium*, ser. Lecture Notes in Mathematics. Springer, 1975, pp. 132–154.
- [111] A. R. Meyer and L. J. Stockmeyer, “Word problems requiring exponential time: Preliminary report,” in *Symposium on Theory of Computing*. ACM, 1973, pp. 1–9.
- [112] B. C. Moszkowski, “Reasoning about digital circuits,” Ph.D. dissertation, 1983.
- [113] P. W. O’Hearn and D. J. Pym, “The logic of bunched implications,” *Bulletin of Symbolic Logic*, vol. 5, pp. 215–244, 1999.
- [114] P. W. O’Hearn, J. C. Reynolds, and H. Yang, “Local reasoning about programs that alter data structures,” in *Computer Science Logic*, ser. LNCS, vol. 2142. Springer, 2001, pp. 1–19.
- [115] P. W. O’Hearn, “Separation logic,” *Communications of the Association for Computing Machinery*, vol. 62, pp. 86–95, 2019.
- [116] P. W. O’Hearn, D. Pym, and J. M. Spring, “Why separation logic works,” *Philosophy and Technology*, vol. 32, pp. 483–516, 2019.
- [117] J. Pagel and F. Zuleger, “Strong-separation logic,” *CoRR*, vol. abs/2001.06235, 2020.
- [118] J. Pagel, C. Matheja, and F. Zuleger, “Complete entailment checking for separation logic with inductive definitions,” *CoRR*, vol. abs/2002.01202, 2020.
- [119] C. H. Papadimitriou, *Computational complexity*. Addison-Wesley, 1994.
- [120] R. Piskać, T. Wies, and D. Zufferey, “Automating separation logic using SMT,” in *Computer-Aided Verification*, ser. LNCS, vol. 8044. Springer, 2013, pp. 773–789.
- [121] D. Pym, *The Semantics and Proof Theory of the Logic of Bunched Implications*, ser. Applied Logic. Springer, 2002, vol. 26.
- [122] M. O. Rabin, “Decidability of second-order theories and automata on infinite trees,” *Transactions of the American Mathematical Society*, vol. 41, pp. 1–35, 1969.
- [123] A. Reynolds, R. Iosif, C. Serban, and T. King, “A decision procedure for separation logic in SMT,” in *ATVA ’16*, ser. LNCS, vol. 9938, 2016, pp. 244–261.
- [124] J. C. Reynolds, “Separation logic: A logic for shared mutable data structures,” in *Logic in Computer Science*. IEEE, 2002, pp. 55–74.

- [125] M. Reynolds, “An axiomatization of full computation tree logic,” *The Journal of Symbolic Logic*, vol. 66, no. 3, pp. 1011–1057, 2001.
- [126] J. F. Santos, P. Maksimovic, S. Ayoun, and P. Gardner, “Gillian, part I: a multi-language platform for symbolic execution,” in *Programming Language Design and Implementation*. ACM, 2020, pp. 927–942.
- [127] W. Savitch, “Relationships between nondeterministic and deterministic tape complexities,” *Journal of Computer and System Sciences*, vol. 4, no. 2, pp. 177–192, 1970.
- [128] S. Schmitz, “Complexity hierarchies beyond elementary,” *Transactions on Computation Theory*, vol. 8, pp. 3:1–3:36, 2016.
- [129] L. Schröder and Y. Venema, “Completeness of flat coalgebraic fixpoint logics,” *Transactions on Computational Logic*, vol. 19, no. 1, pp. 4:1–4:34, 2018.
- [130] A. Silva, “Models of concurrent Kleene algebra,” in *LPAR*, ser. EPiC Series in Computing, vol. 73. EasyChair, 2020, p. 516.
- [131] S. Smolka, N. Foster, J. Hsu, T. Kappé, D. Kozen, and A. Silva, “Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear time,” *ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, vol. 4, pp. 61:1–61:28, 2020.
- [132] L. Stockmeyer, “The complexity of decision problems in automata theory and logic,” Ph.D. dissertation, Department of Electrical Engineering, MIT, 1974.
- [133] S. Tobies, “PSPACE reasoning for graded modal logics,” *Journal of Logic and Computation*, vol. 11, no. 1, pp. 85–106, 2001.
- [134] A. Turing, “Checking a large routine,” in *The Early British Computer Conferences*. MIT Press, 1989, p. 70–72.
- [135] J. A. Väänänen, *Dependence Logic - A New Approach to Independence Friendly Logic*, ser. London Mathematical Society student texts. Cambridge University, 2007, vol. 70.
- [136] J. Väänänen and F. Yang, “Propositional team logics,” *Annals of Pure and Applied Logic*, vol. 168, no. 7, pp. 1406–1441, 2017.
- [137] V. Vafeiadis and M. J. Parkinson, “A marriage of rely/guarantee and separation logic,” in *Concurrency Theory*, ser. LNCS, vol. 4703. Springer, 2007, pp. 256–271.
- [138] J. van Benthem, *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2011.
- [139] H. van Ditmarsch, W. van der Hoek, and B. Kooi, *Dynamic Epistemic Logic*, ser. Synthese Library Series. Springer, 2008, vol. 337.
- [140] M. Y. Vardi, “Efficiency vs. resilience: what COVID-19 teaches computing,” vol. 63, no. 5, p. 9, 2020.
- [141] I. Walukiewicz, “Completeness of Kozen’s axiomatisation of the propositional μ -calculus,” *Information and Computation*, vol. 157, no. 1–2, pp. 142–182, 2000.

- [142] H. Wang, “Proving theorems by pattern recognition — II,” *Bell System Technical Journal*, vol. 40, no. 1, pp. 1–41, 1961.
- [143] Y. Wang and Q. Cao, “On axiomatizations of public announcement logic,” *Synthese*, vol. 190, no. Supplement-1, pp. 103–134, 2013.
- [144] H. Yang, O. Lee, J. Berdine, C. Calcagno, B. Cook, D. Distefano, and P. W. O’Hearn, “Scalable shape analysis for systems code,” in *CAV*, ser. Lecture Notes in Computer Science, vol. 5123. Springer, 2008, pp. 385–398.
- [145] E. Zermelo, “über eine anwendung der mengenlehre auf die theorie des schachspiels,” in *International Congress of Mathematicians*, vol. 2. Cambridge University, 1913, pp. 501–504.

Appendices

A

Appendix of Chapter 3

Contents

Proof of Lemma 3.13.	483
Proof of Lemma 3.15.	483
Proof of Lemma 3.16.	484
Proof of Theorem 3.20.	486

Proof of Lemma 3.13.

Let $x, y \in \text{VAR}$. We recall the definition of the formula $\text{alloc}_y^{-1}(x)$:

$$\text{alloc}_y^{-1}(x) \stackrel{\text{def}}{=} x \hookrightarrow x \vee y \hookrightarrow x \vee (\top * (y \hookrightarrow_{} \wedge \text{size} = 1 \rightsquigarrow y \hookrightarrow^2 x)).$$

Lemma 3.13. Let (s, h) be a memory state such that $s(x) \neq s(y)$. We have,

$$(s, h) \models \text{alloc}_y^{-1}(x) \text{ if and only if } s(x) \in \text{ran}(h).$$

Proof. (\Rightarrow): Suppose that $(s, h) \models \text{alloc}_y^{-1}(x)$. If (s, h) satisfies either $x \hookrightarrow x$ or $y \hookrightarrow x$, then obviously $s(x) \in \text{ran}(h)$. Otherwise, (s, h) must satisfy the third conjunct of $\text{alloc}_y^{-1}(x)$:

$$(s, h) \models \top * (y \hookrightarrow_{} \wedge \text{size} = 1 \rightsquigarrow y \hookrightarrow^2 x).$$

In this case, there is a heap $h' \subseteq h$ such that $(s, h') \models y \hookrightarrow_{} \wedge \text{size} = 1 \rightsquigarrow y \hookrightarrow^2 x$. From the semantics of the subtraction operator, there is a heap h'' disjoint from h' and such that

$$1. \quad s(y) \in \text{dom}(h''), \quad 2. \quad \text{card}(h'') = 1, \quad 3. \quad (s, h' + h'') \models y \hookrightarrow^2 x.$$

From (1) and $h'' \perp h'$, we conclude that $s(y) \notin \text{dom}(h')$. Now, (3) implies that there is a location ℓ such that $\{s(y) \mapsto \ell \mapsto s(x)\} \subseteq h' + h''$, and moreover ℓ must be distinct from both $s(x)$ and $s(y)$ (which are also assumed to be distinct). From (1) and (2), it must be that $h'' = \{s(y) \mapsto \ell\}$ and therefore $\{\ell \mapsto s(x)\} \subseteq h'$. From $h' \subseteq h$ we then conclude that $s(x) \in \text{ran}(h)$.

(\Leftarrow): Suppose there is a location $\ell \in \text{dom}(h)$ such that $h(\ell) = s(x)$ (i.e. $s(x) \in \text{ran}(h)$). First, suppose that $\ell = s(x)$ or $\ell = s(y)$. In this case, we directly derive that $(s, h) \models x \hookrightarrow x \vee y \hookrightarrow x$, which in turn shows that $(s, h) \models \text{alloc}_y^{-1}(x)$. Otherwise, consider the case where $\ell \neq s(x)$ and $\ell \neq s(y)$. Let $h' \subseteq h$ be the heap $\{\ell \mapsto s(x)\}$. As $\ell \neq s(y)$, the location $s(y)$ is not a memory cell of h' . Let us consider the heap $h'' = \{s(y) \mapsto \ell\}$, so that $(s, h'') \models y \hookrightarrow_{} \wedge \text{size} = 1$. The heaps h' and h'' are disjoint, and from their definition we have $h' + h'' = \{s(y) \mapsto \ell \mapsto s(x)\}$. As $s(x)$, ℓ and $s(y)$ are all distinct locations, $(s, h' + h'') \models y \hookrightarrow^2 x$ holds, which in turn allows us to conclude that $(s, h') \models y \hookrightarrow_{} \wedge \text{size} = 1 \rightsquigarrow y \hookrightarrow^2 x$. As $h' \subseteq h$, this implies that (s, h) satisfies $\top * (y \hookrightarrow_{} \wedge \text{size} = 1 \rightsquigarrow y \hookrightarrow^2 x)$, which is sufficient to show that $(s, h) \models \text{alloc}_y^{-1}(x)$. \square

Proof of Lemma 3.15.

Let $x, y \in \text{VAR}$. We recall the definition of $\text{next}(x = y)$:

$$\text{next}(x = y) \stackrel{\text{def}}{=} x \hookrightarrow_{} \wedge \left(x \neq y \Rightarrow \left[x \hookrightarrow_{} \wedge y \hookrightarrow_{} \wedge \neg(\top \rightsquigarrow x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x) \right]_2 \right).$$

Lemma 3.15. Let (s, h) be a memory state such that $\{s(x), s(y)\} \cap \text{ran}(h) = \emptyset$.

$$(s, h) \models \text{next}(x = y) \text{ if and only if } h(s(x)) = h(s(y)).$$

Proof. (\Rightarrow): Suppose $(s, h) \models \text{next}(x = y)$. From the left conjunct of $\text{next}(x = y)$, we have $s(x) \in \text{dom}(h)$. If $s(x) = s(y)$ holds, this allows us to conclude that $h(s(x)) = h(s(y))$. Otherwise, suppose that $s(x) \neq s(y)$ and therefore, by definition of $\text{next}(x = y)$ we have:

$$(s, h) \models [x \hookrightarrow_{} \wedge y \hookrightarrow_{} \wedge \neg(\top \rightsquigarrow x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x)]_2.$$

This means that there is a heap h' such that $\text{card}(h') = 2$, both $s(x)$ and $s(y)$ are memory cells, and $(s, h') \not\models \top \rightsquigarrow x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x$. In particular, there are two locations $\ell, \ell' \in \text{LOC}$ such

that $h' = \{s(x) \mapsto \ell, s(y) \mapsto \ell'\}$. *Ad absurdum*, suppose $\ell \neq \ell'$. Since $\{s(x), s(y)\} \cap \text{ran}(h) = \emptyset$, both ℓ and ℓ' can be neither $s(x)$ nor $s(y)$. Let us consider the heap $h'' = \{\ell' \mapsto s(x), \ell \mapsto s(y)\}$. This heap is disjoint from h' , and moreover $h' + h'' = \{s(x) \mapsto \ell \mapsto s(y)\} + \{s(y) \mapsto \ell' \mapsto s(x)\}$. Thus, $(s, h' + h'') \models x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x$. However, this contradicts $(s, h') \not\models \top \multimap x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x$. We conclude that $\ell = \ell'$, which implies $h(s(x)) = h(s(y))$ by $h' \subseteq h$.

(\Leftarrow): Suppose that there is a location $\ell \in \text{LOC}$ such that $h(s(x)) = \ell$ and $h(s(y)) = \ell$. Of course this means that the left conjunct of $\text{next}(x = y)$, i.e. $x \hookrightarrow _$, is satisfied. Let us consider the right conjunct of the formula. If $s(x) = s(y)$, this conjunct is trivially satisfied as the antecedent of the implication is false. Suppose then $s(x) \neq s(y)$, and let us prove that

$$(s, h) \models [x \hookrightarrow _ \wedge y \hookrightarrow _ \wedge \neg(\top \multimap x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x)]_2.$$

We consider the subheap $h' \subseteq h$ defined as $h' \stackrel{\text{def}}{=} \{s(x) \mapsto \ell, s(y) \mapsto \ell\}$. Clearly, (s, h') satisfies both $x \hookrightarrow _$ and $y \hookrightarrow _$, and from $s(x) \neq s(y)$ we have $\text{card}(h') = 2$. To conclude the proof, let us show that $(s, h') \not\models \top \multimap x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x$. *Ad absurdum*, let us suppose there is a heap h'' disjoint from h' and such that $(s, h'' + h') \models x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x$. Then, there are two locations ℓ_1 and ℓ_2 such that

$$\{s(x) \mapsto \ell_1, \ell_1 \mapsto s(y), s(y) \mapsto \ell_2, \ell_2 \mapsto s(x)\} \subseteq h' + h'',$$

where $\ell_1 \neq s(x)$ and $\ell_2 \neq s(y)$. By definition of h' we have $h'(s(x)) = \ell = h'(s(y))$, which implies that $\ell_1 = \ell_2 = \ell$. However, this contradicts $s(x) \neq s(y)$, as it implies its negation $s(x) = s(y)$ directly by $\{\ell_1 \mapsto s(y), \ell_2 \mapsto s(x)\} \subseteq h' + h''$. Thus, h'' does not exist, which implies that $(s, h') \not\models \top \multimap x \hookrightarrow^2 y \wedge y \hookrightarrow^2 x$, concluding the proof. \square

Proof of Lemma 3.16.

Let $x, y, z \in \text{VAR}$. We recall the definition of $\text{next}_z(x \hookrightarrow y)$:

$$\text{next}_z(x \hookrightarrow y) \stackrel{\text{def}}{=} (\text{next}(x = y) \wedge [x \hookrightarrow _ \wedge \neg(\top \multimap x \hookrightarrow^3 z)]_2) \vee [\text{size} = 1 \multimap x \hookrightarrow^3 z \wedge y \hookrightarrow^2 z]_3.$$

Lemma 3.16. Let (s, h) be such that $s(x) \neq s(z) \neq s(y)$ and $\{s(x), s(y), s(z)\} \cap \text{ran}(h) = \emptyset$.

$$(s, h) \models \text{next}_z(x \hookrightarrow y) \text{ if and only if } h(h(s(x))) = h(s(y)).$$

Proof. Let (s, h) be a memory state where $s(x) \neq s(z) \neq s(y)$ and $\{s(x), s(y), s(z)\} \cap \text{ran}(h) = \emptyset$. (\Rightarrow): Suppose $(s, h) \models \text{next}_z(x \hookrightarrow y)$. We divide the proof into two cases, depending on whether (s, h) satisfies the left or right disjunct.

left disjunct. Suppose $(s, h) \models \text{next}(x = y) \wedge [x \hookrightarrow _ \wedge \neg(\top \multimap x \hookrightarrow^3 z)]_2$, which means that there is a location ℓ such that $h(s(x)) = \ell = h(s(y))$ (by Lemma 3.15) and there is a subheap $h' \subseteq h$ with the following properties:

1. $\text{card}(h') = 2$,
2. $s(x) \in \text{dom}(h')$,
3. $(s, h') \not\models \top \multimap x \hookrightarrow^3 z$.

We prove that $h'(\ell) = \ell$ holds, which implies $h(h(s(x))) = h(s(y))$ from the definition of ℓ together with $h' \subseteq h$. *Ad absurdum*, let us suppose the opposite, i.e. either $\ell \notin \text{dom}(h')$ or $(\ell \in \text{dom}(h'))$ and $h'(\ell) \neq \ell$ holds. In the first case ($\ell \notin \text{dom}(h')$), we consider the heap $h'' = \{\ell \mapsto \ell' \mapsto s(z)\}$ where ℓ' is a location not appearing in $\text{dom}(h') \cup \text{ran}(h') \cup \{s(z)\}$. In particular, $\ell' \neq s(x)$ (by (2)) and h'' is disjoint from h' . Thus, $\{s(x) \mapsto \ell \mapsto \ell' \mapsto s(z)\} \subseteq h' + h''$. However, as $s(x) \neq s(z)$, this implies that the memory state $(s, h' + h'')$ satisfies $x \hookrightarrow^3 z$, which leads to a contradiction with (3).

So, $\ell \in \text{dom}(h')$. Now, let us suppose that $h'(\ell) = \ell''$ holds for some location $\ell'' \neq \ell$. From (1), $h' = \{s(x) \mapsto \ell \mapsto \ell''\}$. Moreover, as $\{s(x), s(z)\} \cap \text{ran}(h) = \emptyset$ and $h' \subseteq h$, the locations ℓ' and ℓ are different from both $s(x)$ and $s(z)$. This implies that $\ell'' \notin \text{dom}(h')$. Let us consider the heap $h''' = \{\ell'' \mapsto s(z)\}$. Clearly, h''' is disjoint from h' , and we have that $\{s(x) \mapsto \ell \mapsto \ell'' \mapsto s(z)\} \subseteq h' + h'''$. Again, as $s(x) \neq s(z)$, this implies that the memory state $(s, h' + h''')$ satisfies $x \hookrightarrow^3 z$, which leads to a contradiction with (3). Thus, $h'(\ell) = \ell$, concluding this case of the proof.

right disjunct. Suppose that (s, h) satisfies the right disjunct of $\text{next}_z(x \hookrightarrow y)$, i.e.

$$(s, h) \models [\text{size} = 1 \multimap x \hookrightarrow^3 z \wedge y \hookrightarrow^2 z]_3.$$

By definition, there is a subheap $h' \subseteq h$ satisfying the following properties:

$$1. \quad \text{card}(h') = 3, \quad 2. \quad (s, h') \models \text{size} = 1 \multimap x \hookrightarrow^3 z \wedge y \hookrightarrow^2 z.$$

From (2), there is a heap h'' of cardinality 1 that is disjoint from h' and it is such that $(s, h' + h'')$ satisfies $x \hookrightarrow^3 z \wedge y \hookrightarrow^2 z$. Thus, there are locations ℓ_1^x, ℓ_2^x and ℓ_1^y s.t.

$$\begin{aligned} \{s(x) \mapsto \ell_1^x \mapsto \ell_2^x \mapsto s(z)\} &\subseteq h' + h'' \\ \{s(y) \mapsto \ell_1^y \mapsto s(z)\} &\subseteq h' + h'' \end{aligned}$$

where $s(x), \ell_1^x, \ell_2^x$ and $s(z)$ are four distinct locations, and similarly $s(y), \ell_1^y$ and $s(z)$ are also three distinct locations. As h' and h'' have cardinality three (by (1)) and one, respectively, their union satisfies $\text{card}(h' + h'') = 4$. This means that the two subheaps $\{s(x) \mapsto \ell_1^x \mapsto \ell_2^x \mapsto s(z)\}$ and $\{s(y) \mapsto \ell_1^y \mapsto s(z)\}$ are not disjoint, which in turn implies $\ell_2^x = \ell_1^y$. Since $s(z) \notin \text{ran}(h')$, we conclude $h'' = \{\ell_1^y \mapsto s(z)\}$, which leaves us with $h' = \{s(x) \mapsto \ell_1^x, \ell_2^x \mapsto \ell_1^y, s(y) \mapsto \ell_1^y\}$. Thus, by $h' \subseteq h$, $h(h(s(x))) = h(s(y))$.

We not consider the right-to-left direction of the lemma.

(\Leftarrow): Suppose $h(h(s(x))) = h(s(y))$. We divide the proof into two cases, depending on whether or not $h(s(x)) = h(s(y))$ holds.

case: $h(s(x)) = h(s(y))$. We show that (s, h) satisfies the left disjunct of $\text{next}_z(x \hookrightarrow y)$. Let ℓ be the location $h(s(x))$. As $\{s(x), s(y)\} \cap \text{ran}(h) = \emptyset$ holds by hypothesis, by Lemma 3.15 we derive that $(s, h) \models \text{next}(x = y)$ and $\ell \neq s(x)$. The two equalities $h(s(x)) = h(s(y))$ and $h(h(s(x))) = h(s(y))$ implies that $h(\ell) = \ell$. In order to conclude the proof, we show that $(s, h) \models [x \hookrightarrow _ \wedge \neg(\top \multimap x \hookrightarrow^3 z)]_2$. We consider the subheap $h' \subseteq h$ defined as $h' = \{s(x) \mapsto \ell \mapsto \ell\}$. Clearly, $\text{card}(h') = 2$ and $(s, h') \models x \hookrightarrow _$, so that we only need to show that $(s, h') \not\models \top \multimap x \hookrightarrow^3 z$. *Ad absurdum*, suppose there is a heap h'' disjoint from h' and such that $(s, h' + h'') \models x \hookrightarrow^3 z$. By definition, of \hookrightarrow^3 , there are two distinct locations $\ell_1 \neq \ell_2$ such that $\{s(x) \mapsto \ell_1 \mapsto \ell_2 \mapsto s(z)\} \subseteq h' + h''$. However, this is contradictory, as from $h' = \{s(x) \mapsto \ell \mapsto \ell\}$ we are able to conclude $\ell_1 = \ell = \ell_2$. Therefore, h'' does not exist, which implies $(s, h') \not\models \top \multimap x \hookrightarrow^3 z$.

case: $h(s(x)) \neq h(s(y))$. We show that (s, h) satisfies the right disjunct of $\text{next}_z(x \hookrightarrow y)$. From the hypothesis $h(h(s(x))) = h(s(y))$, there is a subheap $h' \subseteq h$ s.t. for some $\ell, \ell' \in \text{LOC}$, we have $h' = \{s(x) \mapsto \ell, \ell \mapsto \ell', s(y) \mapsto \ell'\}$. From the hypothesis $\{s(x), s(y), s(z)\} \cap \text{ran}(h) = \emptyset$ we conclude that both ℓ and ℓ' are distinct from $s(x), s(y)$ and $s(z)$. By $h(s(x)) \neq h(s(y))$ and $s(x) \neq s(z) \neq s(y)$, we then derive that $s(x), s(y), s(z), \ell$ and ℓ' are five distinct locations. In particular, this means that $\ell' \notin \text{dom}(h')$ and $\text{card}(h') = 3$. To conclude the proof, it is sufficient to show that $(s, h') \models \text{size} = 1 \multimap x \hookrightarrow^3 z \wedge y \hookrightarrow^2 z$. Let us consider the heap $h'' = \{\ell' \mapsto s(z)\}$. Clearly, $\text{card}(h'') = 1$ (thus, $(s, h'') \models \text{size} = 1$) and h'' is disjoint from h' . As $\{s(x) \mapsto \ell \mapsto \ell' \mapsto s(z)\} \subseteq h' + h''$ and $s(x), s(z), \ell$ and ℓ' are distinct

locations, we conclude that $(s, h' + h'') \models x \hookrightarrow^3 z$. Similarly, $(s, h' + h'') \models y \hookrightarrow^2 z$ holds as $\{s(y) \mapsto \ell' \mapsto s(z)\} \subseteq h' + h''$ and $s(y)$, ℓ' and $s(z)$ are distinct locations. \square

Proof of Theorem 3.20.

The theorem follows from Lemma 3.22 and Lemma 3.23, which we recall and prove in this section. First, we establish the correctness of the formulae emp , $\text{uniq}(p)$, $\text{nom}(p)$ and is_a_spy_x , defined as follows:

$$\begin{aligned}\text{emp} &= \neg\Diamond\top \wedge ((\Diamond\neg\Diamond\top) \multimap \neg\Diamond\Diamond\top), \\ \text{uniq}(p) &= \top * (p \wedge \text{emp} \wedge (\text{size}=1 \wedge \Diamond p) \multimap \Diamond\Diamond\top), \\ \text{nom}(p) &= \top * (\text{emp} \wedge ((\Diamond\text{uniq}(p)) \multimap \top)), \\ \text{is_a_spy}_x &= \text{uniq(spy)} \wedge (\bigwedge_{x \in X} \neg x) \wedge \neg\Diamond\top \wedge \neg((\text{size}=1 \wedge \Diamond\neg\text{spy}) \multimap \Diamond\Diamond\text{spy}).\end{aligned}$$

Recall that we assume $\text{VAR} = \text{AP}$.

Lemma A.1. Let $(\mathcal{K} = (\mathcal{W}, R, \mathcal{V}), w)$ be a pointed finite function, and let $X \subseteq_{\text{fin}} \text{VAR}$.

- (I) $(\mathcal{K}, w) \models \text{emp}$ if and only if $R = \emptyset$,
- (II) $(\mathcal{K}, w) \models \text{uniq}(p)$ if and only if $\mathcal{V}(p) = \{w\}$,
- (III) $(\mathcal{K}, w) \models \text{nom}(p)$ if and only if $\text{card}(\mathcal{V}(p)) = 1$,
- (IV) $(\mathcal{K}, w) \models \text{is_a_spy}_X$ if and only if $\mathcal{V}(\text{spy}) = \{w\}$, $w \notin \bigcup_{x \in X} \mathcal{V}(x)$ and $w \notin \pi_1(R) \cup \pi_2(R)$.

Proof of (I). (\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{emp}$, and so $(\mathcal{K}, w) \not\models \Diamond\top$ and $(\mathcal{K}, w) \models (\Diamond\neg\Diamond\top) \multimap \neg\Diamond\Diamond\top$. From the former statement, we conclude that $R(w) = \emptyset$. *Ad absurdum*, suppose $R \neq \emptyset$ and let $(w', w'') \in R$. Since $R(w) = \emptyset$, $w' \neq w$. We consider the Kripke-style finite function $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ where $R' = \{(w, w')\}$. Clearly, \mathcal{K} and \mathcal{K}' are disjoint and, since $w' \neq w$, $(\mathcal{K}', w) \models \Diamond\neg\Diamond\top$. From $(\mathcal{K}, w) \models (\Diamond\neg\Diamond\top) \multimap \neg\Diamond\Diamond\top$ we conclude that $(\mathcal{K} + \mathcal{K}', w) \models \neg\Diamond\Diamond\top$. However, this is contradictory, as $\{(w, w'), (w', w'')\} \subseteq R \cup R'$.

(\Leftarrow): Suppose $R = \emptyset$. Clearly, this implies $R(w) = \emptyset$ and so $(\mathcal{K}, w) \models \neg\Diamond\top$. To prove that $(\mathcal{K}, w) \models (\Diamond\neg\Diamond\top) \multimap \neg\Diamond\Diamond\top$, let $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ be a finite function that is disjoint from \mathcal{K} and such that $(\mathcal{K}', w) \models \Diamond\neg\Diamond\top$. We show that $(\mathcal{K} + \mathcal{K}', w) \not\models \Diamond\Diamond\top$. Since $(\mathcal{K}', w) \models \Diamond\neg\Diamond\top$, we conclude that $R'(w) = \{w'\}$ for some world w' , but $R(w') = \emptyset$. In particular, this implies $(\mathcal{K}', w) \not\models \Diamond\Diamond\top$. As $R = \emptyset$, $\mathcal{K} + \mathcal{K}' = \mathcal{K}'$. So, $(\mathcal{K} + \mathcal{K}', w) \not\models \Diamond\Diamond\top$. \square

Proof of (II). (\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{uniq}(p)$. There is a finite function $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ such that $\mathcal{K}' \subseteq \mathcal{K}$ and (\mathcal{K}', w) satisfies $p \wedge \text{emp} \wedge (\text{size}=1 \wedge \Diamond p) \multimap \Diamond\Diamond\top$. From the first and second conjuncts of this formula, we conclude that $w \in \mathcal{V}(p)$ and $R' = \emptyset$. *Ad absurdum*, let us suppose that there is a world w' distinct from w and such that $w' \in \mathcal{V}(p)$. We consider the finite function $\mathcal{K}'' = (\mathcal{W}, \{(w, w')\}, \mathcal{V})$. Clearly, \mathcal{K}'' is disjoint from \mathcal{K}' and (\mathcal{K}'', w) satisfies $\text{size}=1 \wedge \Diamond p$. From $w \neq w'$ we conclude that $(\mathcal{K}' + \mathcal{K}'', w) \not\models \Diamond\Diamond\top$. However, this means that $(\mathcal{K}', w) \models (\text{size}=1 \wedge \Diamond p) \multimap \neg\Diamond\Diamond\top$, in contradiction with the satisfaction of the formula $(\text{size}=1 \wedge \Diamond p) \multimap \Diamond\Diamond\top$. Thus, $w' = w$ and so $\mathcal{V}(p) = \{w\}$.

(\Leftarrow): Suppose $\mathcal{V}(p) = \{w\}$. We consider the Kripke-style finite function $\mathcal{K}' = (\mathcal{W}, \emptyset, \mathcal{V}) \subseteq \mathcal{K}$. Clearly, $(\mathcal{K}', w) \models p \wedge \text{emp}$. Let us show that $(\mathcal{K}', w) \models (\text{size}=1 \wedge \Diamond p) \multimap \Diamond\Diamond\top$. We consider a finite function $\mathcal{K}'' = (\mathcal{W}, R'', \mathcal{V})$ disjoint from \mathcal{K}' and such that $(\mathcal{K}'', w) \models \text{size}=1 \wedge \Diamond p$. Together with $\mathcal{V}(p) = \{w\}$, this implies that $R'' = \{(w, w)\}$. Thus, R'' witnesses a selfloop on w , which

entails that $(\mathcal{K}'', w) \models \diamond\diamond p$. Lastly, $\mathcal{K}' + \mathcal{K}'' = \mathcal{K}'$ as $\mathcal{K}' = (\mathcal{W}, \emptyset, \mathcal{V})$, and so $(\mathcal{K}' + \mathcal{K}'', w) \models \diamond\diamond p$, as required by the separating implication. \square

Proof of (III). (\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{nom}(p)$. There is a finite function $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ such that $\mathcal{K}' \subseteq \mathcal{K}$ and (\mathcal{K}', w) satisfies $\text{emp} \wedge ((\diamond\text{uniq}(p)) \rightsquigarrow \top)$. From the first conjunct of this formula, $R' = \emptyset$. From the second conjunct, there is a finite function $\mathcal{K}'' = (\mathcal{W}, R'', \mathcal{V})$ disjoint from \mathcal{K}' and such that $(\mathcal{K}'', w) \models \diamond\text{uniq}(p)$. From Lemma A.1(II) and the semantics of the modality \diamond , there is a world $w' \in R''(w)$ such that $\mathcal{V}(p) = \{w'\}$. Thus, $\text{card}(\mathcal{V}(p)) = 1$.

(\Leftarrow): Suppose $\text{card}(\mathcal{V}(p)) = 1$ and so let us assume w' to be the only world in $\mathcal{V}(p)$. Let us consider the finite function $\mathcal{K}' = (\mathcal{W}, \emptyset, \mathcal{V})$ and $\mathcal{K}'' = (\mathcal{W}, \{(w, w')\}, \mathcal{V})$. Clearly, \mathcal{K}'' is disjoint from \mathcal{K}' , $(\mathcal{K}', w) \models \text{emp}$, and $(\mathcal{K}'', w) \models \diamond\text{uniq}(p)$. So, $(\mathcal{K}', w) \models \text{emp} \wedge (\diamond\text{uniq}(p) \rightsquigarrow \top)$. Lastly, $\mathcal{K}' \subseteq \mathcal{K}$ leads to $(\mathcal{K}, w) \models \text{nom}(p)$. \square

Proof of (IV). The double implication

$$(\mathcal{K}, w) \models \text{uniq(spy)} \wedge \left(\bigwedge_{x \in X} \neg x \right) \wedge \neg \diamond \top \text{ iff } \mathcal{V}(\text{spy}) = \{w\}, w \notin \bigcup_{x \in X} \mathcal{V}(x), \text{ and } w \notin \pi_1(R),$$

is straightforward. Thus, in order to prove the result it is sufficient to assume that $\mathcal{V}(\text{spy}) = \{w\}$ and $w \notin \pi_1(R)$ hold, and prove the following double implication:

$$(\mathcal{K}, w) \models \neg((\text{size} = 1 \wedge \diamond\neg\text{spy}) \rightsquigarrow \diamond\diamond\text{spy}) \text{ if and only if } w \notin \pi_2(R).$$

(\Rightarrow): We show the contrapositive. Suppose $w \in \pi_2(R)$, and so there is a world $w' \in \mathcal{W}$ such that $w \in R(w')$. Let us consider the Kripke-style finite function $\mathcal{K}' = (\mathcal{W}, \{(w, w')\}, \mathcal{V})$. From $w \notin \pi_1(R)$, we conclude that $w \neq w'$ and that \mathcal{K}' is disjoint from \mathcal{K} . By $\mathcal{V}(\text{spy}) = \{w\}$ it holds that $(\mathcal{K}', w) \models \text{size} = 1 \wedge \diamond\neg\text{spy}$, and moreover $(\mathcal{K} + \mathcal{K}', w) \models \diamond\diamond\text{spy}$. By semantics of the subtraction, $(\mathcal{K}, w) \models (\text{size} = 1 \wedge \diamond\neg\text{spy}) \rightsquigarrow \diamond\diamond\text{spy}$.

(\Leftarrow): Again, let us take the contrapositive. Suppose $(\mathcal{K}, w) \models (\text{size} = 1 \wedge \diamond\neg\text{spy}) \rightsquigarrow \diamond\diamond\text{spy}$, and so there is a finite function $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ that is disjoint from \mathcal{K} and such that

- A. $(\mathcal{K}', w) \models \text{size} = 1 \wedge \diamond\neg\text{spy}$, which implies that $R' = \{(w, w')\}$ for some $w' \neq w$,
- B. $(\mathcal{K} + \mathcal{K}', w) \models \diamond\diamond\text{spy}$. So, $R \cup R'$ witnesses a path of length two going from w to itself.

Notice that the two statements above holds as $\mathcal{V}(\text{spy}) = \{w\}$. As $R \cup R'$ is a functional relation, (A) and (B) imply that $\{(w, w'), (w', w)\} \in R \cup R'$, where $w' \neq w$. From $R' = \{(w, w')\}$, we conclude that $(w', w) \in R$. Thus, $w \in \pi_2(R)$. \square

Lemma 3.22. Let (s, h) be a memory state and let φ be a formula in $\text{SL}(*, \neg*, \rightarrow^2, \rightarrow^3)$, with variables from $X \subseteq_{\text{fin}} \text{VAR} \setminus \{\text{spy}\}$. Let (\mathcal{K}, w) be a pointed finite function that is an X -encoding of (s, h) . We have, $(s, h) \models \varphi$ if and only if $(\mathcal{K}, w) \models \tau_X(\varphi)$.

Proof. The proof is by structural induction on φ . Below, we recall the properties of (\mathcal{K}, w) , where $\mathcal{K} = (\text{LOC}, R, \mathcal{V})$, as an encoding of (s, h) whenever $R = h$ and moreover

1. every $x \in X$ is a *nominal* that corresponds to $s(x)$, i.e. $\mathcal{V}(x) = \{s(x)\}$,
2. the current world w is a *spy*, i.e. it is a nominal for a fixed propositional symbol $\text{spy} \notin X$, it does not satisfy any propositional symbol in X , and it does not belong to a pair in R .

Below, we call the properties (1) and (2) *hypothesis* whenever they refer to (\mathcal{K}, w) and (s, h) . Instead, we call them *properties* where referring to their analogue on other two structures, for which we want to prove their satisfaction. The base cases for $\varphi = \top$ and $\varphi = \text{emp}$ are obvious.

base case: $\varphi = \text{x} = \text{y}$.

$$\begin{aligned}
& (s, h) \models \text{x} = \text{y}, \\
\Leftrightarrow & s(\text{x}) = s(\text{y}), && (\text{by definition of } \models) \\
\Leftrightarrow & \mathcal{V}(\text{x}) = \{s(\text{x})\} = \{s(\text{y})\} = \mathcal{V}(\text{y}), && (\text{by hypothesis (1)}) \\
\Leftrightarrow & (\text{w}, s(\text{x})) \notin R \text{ and } (\text{LOC}, R \cup \{(\text{w}, s(\text{x}))\}, \mathcal{V}) \models \diamond(\text{x} \wedge \text{y}), && (\text{by hypothesis (2)}) \\
\Leftrightarrow & (\mathcal{K}, \text{w}) \models \text{size} = 1 \dashv \diamond(\text{x} \wedge \text{y}). && (\text{by definition of } \dashv)
\end{aligned}$$

base case: $\varphi = \text{x} \hookrightarrow \text{y}$.

$$\begin{aligned}
& (s, h) \models \text{x} \hookrightarrow \text{y}, \\
\Leftrightarrow & h(s(\text{x})) = s(\text{y}), && (\text{by definition of } \models) \\
\Leftrightarrow & \mathcal{V}(\text{x}) = \{s(\text{x})\}, \mathcal{V}(\text{y}) = \{s(\text{y})\}, (s(\text{x}), s(\text{y})) \in R, && (\text{by } R = h \text{ and hyp. (1)}) \\
\Leftrightarrow & (\text{w}, s(\text{x})) \notin R \text{ and } (\text{LOC}, R \cup \{(\text{w}, s(\text{x}))\}, \mathcal{V}) \models \diamond(\text{x} \wedge \diamond \text{y}), && (\text{by hypothesis (2)}) \\
\Leftrightarrow & (\mathcal{K}, \text{w}) \models \text{size} = 1 \dashv \diamond(\text{x} \wedge \diamond \text{y}). && (\text{by definition of } \dashv)
\end{aligned}$$

The cases for $\text{x} \hookrightarrow^2 \text{y}$ and $\text{x} \hookrightarrow^3 \text{y}$ are very similar. Let us assume that $((s, h) \models \text{x} \hookrightarrow^2 \text{y})$ iff $(\mathcal{K}, \text{w}) \models \tau_{\text{X}}(\text{x} \hookrightarrow^2 \text{y})$ holds, and show the case for $\text{x} \hookrightarrow^3 \text{y}$.

base case: $\varphi = \text{x} \hookrightarrow^3 \text{y}$.

$$\begin{aligned}
& (s, h) \models \text{x} \hookrightarrow^3 \text{y}, \\
\Leftrightarrow & h^3(s(\text{x})) = s(\text{y}), \text{ and for every } \delta \in [0, 2] \text{ } h^\delta(s(\text{x})) \neq s(\text{y}), \\
& (\text{by definition of } \models) \\
\Leftrightarrow & \mathcal{V}(\text{x}) = \{s(\text{x})\}, \mathcal{V}(\text{y}) = \{s(\text{y})\}, (s(\text{x}), s(\text{y})) \in R^3, \text{ and } (s(\text{x}), s(\text{y})) \notin R^\delta \text{ for } \delta \in [0, 2] \\
& (\text{by } R = h \text{ and hypothesis (1)}) \\
\Leftrightarrow & (\text{w}, s(\text{x})) \notin R, (\text{LOC}, R \cup \{(\text{w}, s(\text{x}))\}, \mathcal{V}) \models \diamond(\text{x} \wedge \diamond \diamond \diamond \text{y}), \text{ and } (\mathcal{K}, \text{w}) \text{ does not satisfy} \\
& \text{neither } \tau_{\text{X}}(\text{x} = \text{y}), \tau_{\text{X}}(\text{x} \hookrightarrow \text{y}) \text{ nor } \tau_{\text{X}}(\text{x} \hookrightarrow^2 \text{y}), \\
& (\text{by hypothesis (2) and the other base cases}) \\
\Leftrightarrow & (\mathcal{K}, \text{w}) \models \neg \tau_{\text{X}}(\text{x} = \text{y}) \wedge \neg \tau_{\text{X}}(\text{x} \hookrightarrow \text{y}) \wedge \neg \tau_{\text{X}}(\text{x} \hookrightarrow^2 \text{y}) \wedge (\text{size} = 1 \dashv \diamond(\text{x} \wedge \diamond \diamond \diamond \text{y})). \\
& (\text{by semantics of } \dashv \text{ and } \wedge)
\end{aligned}$$

For the induction steps, the cases for Boolean connectives are obvious. The case for the separating conjunction is also quite straightforward:

induction step: $\varphi = \varphi_1 * \varphi_2$.

$$\begin{aligned}
& (s, h) \models \varphi_1 * \varphi_2, \\
\Leftrightarrow & \text{there are two disjoint heaps } h_1 \text{ and } h_2 \text{ s.t. } h = h_1 + h_2, (s, h_1) \models \varphi_1 \text{ and } (s, h_2) \models \varphi_2, \\
& (\text{by definition of } \models) \\
\Leftrightarrow & \text{there are two disjoint heaps } h_1 \text{ and } h_2 \text{ such that } \mathcal{K}_1 \stackrel{\text{def}}{=} (\text{LOC}, h_1, \mathcal{V}), \mathcal{K}_2 \stackrel{\text{def}}{=} (\text{LOC}, h_2, \mathcal{V}), \\
& \mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2, (\mathcal{K}_1, \text{w}) \models \tau_{\text{X}}(\varphi_1) \text{ and } (\mathcal{K}_2, \text{w}) \models \tau_{\text{X}}(\varphi_2) \\
& (\text{by induction hypothesis, as } (\mathcal{K}_j, \text{w}) \text{ is a X-encoding of } (s, h_j), \text{ for } j \in \{1, 2\}) \\
\Leftrightarrow & (\mathcal{K}, \text{w}) \models \tau_{\text{X}}(\varphi_1 * \varphi_2). \\
& (\text{by definition of } \models)
\end{aligned}$$

In the double implications above, proving that $(\mathcal{K}_j, \text{w})$ is a X-encoding of (s, h_j) , for $j \in \{1, 2\}$ is quite straightforward. Indeed, the property (1) for $(\mathcal{K}_j, \text{w})$ holds directly from the hypothesis (1). The property (2) for $(\mathcal{K}_j, \text{w})$, follows instead from the fact that the satisfaction of `is_a_spy_X` is monotone with respect to submodels, i.e. if $(\mathcal{K}, \text{w}) \models \text{is_a_spy}_X$ then for every $\mathcal{K}' \subseteq \mathcal{K}$ we have $(\mathcal{K}', \text{w}) \models \text{is_a_spy}_X$.

We conclude the proof by considering the case for the separating implication.

induction step: $\varphi = \varphi_1 \multimap \varphi_2$.

(\Rightarrow): Suppose $(s, h) \models \varphi_1 \multimap \varphi_2$, and so for every h' disjoint from h , if $(s, h) \models \varphi_1$ then $(s, h_1 + h_2) \models \varphi_2$. We prove that $(\mathcal{K}, w) \models (\text{is_a_spy}_x \wedge \tau_x(\varphi_1)) \multimap \tau_x(\varphi_2)$. Equivalently,

for every Kripke-style finite function $\mathcal{K}' = (\text{LOC}, R', \mathcal{V})$ disjoint from \mathcal{K} ,
if $(\mathcal{K}', w) \models \text{is_a_spy}_x \wedge \tau_x(\varphi_1)$ then $(\mathcal{K} + \mathcal{K}', w) \models \tau_x(\varphi_2)$.

Let us consider the Kripke-style finite function $\mathcal{K}' = (\text{LOC}, R', \mathcal{V})$ disjoint from \mathcal{K} and such that $(\mathcal{K}', w) \models \text{is_a_spy}_x \wedge \tau_x(\varphi_1)$. Let $h' = R'$. From Lemma A.1(IV), and (1), (\mathcal{K}', w) is a x -encoding of (s, h') . By induction hypothesis, $(s, h') \models \varphi_1$. From $h = R$ together with the fact that \mathcal{K}' is disjoint from \mathcal{K} , we conclude that $h \perp h'$ and thus $(s, h + h') \models \varphi_2$. Let $\mathcal{K}'' = (\text{LOC}, R \cup R', \mathcal{V})$, so that $\mathcal{K}'' = \mathcal{K} + \mathcal{K}'$. We show that $(\mathcal{K} + \mathcal{K}', w) \models \tau_x(\varphi_2)$. First, since both (\mathcal{K}, w) and (\mathcal{K}', w) satisfy is_a_spy_x , it is quite straightforward to see that $(\mathcal{K} + \mathcal{K}', w) \models \text{is_a_spy}_x$. Indeed, the first two conjuncts of is_a_spy_x only depends on the labelling function \mathcal{V} , whereas from $w \notin \pi_1(R) \cup \pi_2(R)$ and $w \notin \pi_1(R') \cup \pi_2(R')$ we conclude that $w \notin \pi_1(R \cup R') \cup \pi_2(R \cup R')$. Thus, $(\mathcal{K} + \mathcal{K}', w) \models \text{is_a_spy}_x$ follows from Lemma A.1(IV). We show that $(\mathcal{K} + \mathcal{K}', w)$ is a x -encoding of $(s, h + h')$. From $h = R$, $h' = R'$ and $R \cap R' = \emptyset$, we conclude that $R \cup R' = h + h'$. The property (1) for $(\mathcal{K} + \mathcal{K}', w)$ holds directly from the hypothesis (1). The property (2) for $(\mathcal{K} + \mathcal{K}', w)$ follows from $(\mathcal{K} + \mathcal{K}', w) \models \text{is_a_spy}_x$. Thus, by induction hypothesis, $(\mathcal{K} + \mathcal{K}', w) \models \tau_x(\varphi_2)$.

(\Leftarrow): Suppose $(\mathcal{K}, w) \models (\text{is_a_spy}_x \wedge \tau_x(\varphi_1)) \multimap \tau_x(\varphi_2)$, and so for every Kripke-style finite function $\mathcal{K}' = (\text{LOC}, R', \mathcal{V})$ disjoint from \mathcal{K} and such that $(\mathcal{K}', w) \models \text{is_a_spy}_x \wedge \tau_x(\varphi_1)$, we have $(\mathcal{K} + \mathcal{K}', w) \models \tau_x(\varphi_2)$. Let us prove that $(s, h) \models \varphi_1 \multimap \varphi_2$. Let h' be a heap disjoint from h and such that $(s, h') \models \varphi_1$. Without loss of generality, we can assume that $w \notin \text{dom}(h') \cup \text{ran}(h')$. Indeed, if this is not the case, it is sufficient to consider a location $w' \notin \text{dom}(h+h') \cup \text{ran}(h+h')$ and the heap h'_1 obtained from h replacing every occurrence of w with w' . Since $w \notin \text{dom}(h) \cup \text{ran}(h)$, we have that $w \notin \text{dom}(h + h'_1) \cup \text{ran}(h + h'_1)$. Moreover, it is quite easy to see that $(s, h') \simeq_x (s, h'_1)$ and $(s, h + h') \simeq_x (s, h + h'_1)$, where \simeq_x is the x -heap-isomorphism introduced in Definition 2.9. By Proposition 2.10, (s, h') and (s, h'_1) (resp. $(s, h + h')$ and $(s, h + h'_1)$) satisfy the same formulae having variables in x . So, assume that $w \notin \text{dom}(h') \cup \text{ran}(h')$. We show that $(s, h + h') \models \varphi_2$. Consider the finite function $\mathcal{K}' = (\text{LOC}, R', \mathcal{V})$ where $R' = h$. From $w \notin \text{dom}(h') \cup \text{ran}(h')$ and by hypothesis (1) and (2), we conclude that (\mathcal{K}', w) is a x -encoding of (s, h') . Again by (2), $(\mathcal{K}', w) \models \text{is_a_spy}_x$ and by induction hypothesis $(\mathcal{K}', w) \models \tau_x(\varphi_1)$. Since h' is disjoint from h , we conclude that $\mathcal{K} + \mathcal{K}'$ is defined and so $(\mathcal{K} + \mathcal{K}', w) \models \tau_x(\varphi_2)$. From the fact that (\mathcal{K}, w) is a x -encoding of (s, h) , and (\mathcal{K}', w) is a x -encoding of (s, h') , we show that $(\mathcal{K} + \mathcal{K}', w)$ is a x -encoding of $(s, h_1 + h_2)$. Indeed, by definition of R and R' we have $R \cup R' = h + h'$. The property (1) for $(\mathcal{K} + \mathcal{K}', w)$ holds directly from the hypothesis (1). The property (2) for $(\mathcal{K} + \mathcal{K}', w)$ holds from hypothesis (2) together with $w \notin \pi_1(R') \cup \pi_2(R')$. Thus, by induction hypothesis $(s, h + h') \models \varphi_2$. \square

Lemma 3.23. Let φ be a formula in $\text{SL}(*, \multimap, \hookrightarrow^2, \hookrightarrow^3)$, with variables from $x \subseteq_{\text{fin}} \text{VAR} \setminus \{\text{spy}\}$.

- (I) φ and $\text{is_a_spy}_x \wedge \bigwedge_{x \in x} \text{nom}(x) \wedge \tau_x(\varphi)$ are equisatisfiable.
- (II) φ and $\text{is_a_spy}_x \wedge \bigwedge_{x \in x} \text{nom}(x) \Rightarrow \tau_x(\varphi)$ are equivalent.

We show the proof of the statement (I). The proof of the statement (II) is analogous.

Proof of (I). (\Rightarrow): Suppose φ satisfiable and let (s, h) be a memory state satisfying it. Consider a Kripke-style finite function $\mathcal{K} = (\text{LOC}, h, \mathcal{V})$ such that

- for every $x \in X$, $\mathcal{V}(x) \stackrel{\text{def}}{=} \{s(x)\}$,
- $\mathcal{V}(\text{spy}) = \{w\}$ for some world w not in $\pi_1(R) \cup \pi_2(R)$ nor in $\bigcup_{x \in X} \mathcal{V}(x)$.

By Definition 3.21, (\mathcal{K}, w) is an X -encoding of (s, h) . By Lemma A.1, (\mathcal{K}, w) satisfies both the formulae is_a_spy_X and $\bigwedge_{x \in X} \text{nom}(x)$. By Lemma 3.22, $(\mathcal{K}, w) \models \tau_X(\varphi)$.

(\Leftarrow): Suppose $\text{is_a_spy}_X \wedge \bigwedge_{x \in X} \text{nom}(x) \wedge \tau_X(\varphi)$ satisfiable and let (\mathcal{K}, w) be a Kripke-style finite function satisfying it, where $\mathcal{K} = (\text{LOC}, R, \mathcal{V})$. By $(\mathcal{K}, w) \models \text{is_a_spy}_X \wedge \bigwedge_{x \in X} \text{nom}(x)$ together with Lemma A.1, we have:

1. for every $x \in X$ there is a world w_x such that $\mathcal{V}(x) = \{w_x\}$,
2. $\mathcal{V}(\text{spy}) = \{w\}$, $w \notin \pi_1(R) \cup \pi_2(R)$, and $w \notin \bigcup_{x \in X} \mathcal{V}(x)$.

Let (s, h) be a memory state s.t. $h \stackrel{\text{def}}{=} R$ and for every $x \in X$ $s(x) = w_x$ (w_x is s.t. $\mathcal{V}(x) = \{w_x\}$). By Definition 3.21, (\mathcal{K}, w) is a X -encoding of (s, h) . By Lemma 3.22, $(s, h) \models \varphi$. \square

B

Appendix of Chapter 4

Contents

Proof of Lemma 4.9.	493
Proof of Lemma 4.16.	494
Proof of Lemma 4.25.	495
Proof of Lemma 4.26.	497
Proof of Lemma 4.31.	500
Proof of Lemma 4.32.	502
Proof of Lemma 4.39.	502
Proof of Lemma 4.40.	505
Proof of Theorem 4.41.	506
Proof of Lemma 4.43.	508
Proof of Lemma 4.44.	509

Proof of Lemma 4.9.

Lemma 4.9. Let (\mathcal{F}, t, n) be a pointed forest. Then,

- (I) $(\mathcal{F}, t, n) \models \#\text{desc} \geq \beta$ iff n has at least β descendants and it is a descendant of t .
- (II) $(\mathcal{F}, t, n) \models \#\text{child} \geq \beta$ iff n has at least β children and it is a descendant of t .

Proof of (I). Let us recall the definition of $\#\text{desc} \geq \beta$.

$$\#\text{desc} \geq \beta \stackrel{\text{def}}{=} \blacklozenge^* ([U] \neg \text{Miss} \wedge \text{Hit} \wedge \blacklozenge(\neg \text{inDom} \wedge \text{size}(\text{Miss}) \geq \beta)).$$

$\mathcal{F}[\text{Miss}]$ is empty removing n lead to at least β miss nodes

(\Rightarrow) : Suppose $(\mathcal{F}, t, n) \models \#\text{desc} \geq \beta$. From the semantics of \blacklozenge^* , there is a forest $\mathcal{F}' \subseteq \mathcal{F}$ s.t.

- A. $(\mathcal{F}', t, n) \models [U] \neg \text{Miss}$. So, $\mathcal{F}'[\text{Miss}]_t = \emptyset$, and every $n' \in \text{dom}(\mathcal{F}')$ is an \mathcal{F}' -descendant of t ,
- B. $(\mathcal{F}', t, n) \models \text{Hit}$. So, n is an \mathcal{F}' -descendant of t (thus, n is also an \mathcal{F} -descendant of t),
- C. $(\mathcal{F}', t, n) \models \blacklozenge(\neg \text{inDom} \wedge \text{size}(\text{Miss}) \geq \beta)$.

From (C), there is a finite forest \mathcal{F}'' such that $\mathcal{F}'' \subseteq \mathcal{F}'$ and

- D. $\text{card}(\text{dom}(\mathcal{F}'')) = \text{card}(\text{dom}(\mathcal{F}')) - 1$,
- E. $(\mathcal{F}'', t, n) \models \neg \text{inDom}$,
- F. $(\mathcal{F}'', t, n) \models \text{size}(\text{Miss}) \geq \beta$.

By (B), we obtain that $n \in \text{dom}(\mathcal{F}')$, whereas (E) implies $n \notin \text{dom}(\mathcal{F}'')$. Therefore, by (D) it holds that $\text{dom}(\mathcal{F}'') = \text{dom}(\mathcal{F}') \setminus \{n\}$. We now consider the set $\mathcal{F}''[\text{Miss}]_t$, which by (F) has at least β elements. We prove that every $n' \in \mathcal{F}''[\text{Miss}]_t$ is an \mathcal{F}' -descendant of n . *Ad absurdum*, suppose that there is $n' \in \mathcal{F}''[\text{Miss}]_t$ that is not an \mathcal{F}' -descendant of n . By definition of $\mathcal{F}''[\text{Miss}]_t$,

- G. n' is not an \mathcal{F}'' -descendant of t ,
- H. $n' \in \text{dom}(\mathcal{F}'')$ and therefore by $\mathcal{F}'' \subseteq \mathcal{F}'$ it holds that $n' \in \text{dom}(\mathcal{F}')$.

From (A) and (H), n' is an \mathcal{F}' -descendant of t . Thus, from (G), there must be $\delta \geq 1$ such that the node $\mathcal{F}'^\delta(n')$ is an \mathcal{F}' -descendant of t , but it is not in the domain of \mathcal{F}'' . However, as we already established that $\text{dom}(\mathcal{F}'') = \text{dom}(\mathcal{F}') \setminus \{n\}$, this implies that $\mathcal{F}'^\delta(n') = n$ and therefore n' is an \mathcal{F}' -descendant of n : a contradiction. Therefore, every element in $\mathcal{F}''[\text{Miss}]_t$ is an \mathcal{F}' -descendant of n . Together with (B) and $\mathcal{F}' \subseteq \mathcal{F}$, we conclude that n has at least β \mathcal{F} -descendants and it is an \mathcal{F} -descendant of t .

(\Leftarrow) : Suppose (\mathcal{F}, t, n) to be a finite forest such that n has at least β \mathcal{F} -descendants and it is a \mathcal{F} -descendant of t . We then consider the two subforests \mathcal{F}'' and \mathcal{F}' of \mathcal{F} characterised as

$$\begin{aligned} \text{dom}(\mathcal{F}'') = & \{n' \in \mathcal{N} \mid n' \text{ is an } \mathcal{F}\text{-descendant of } n\} \cup \\ & \{n' \in \mathcal{N} \mid n' \text{ is an } \mathcal{F}\text{-ancestor of } n \text{ and an } \mathcal{F}\text{-descendant of } t\}. \\ \mathcal{F}' = & \mathcal{F}'' \cup \{(n, \mathcal{F}(n))\} \end{aligned}$$

Notice that $n \notin \text{dom}(\mathcal{F}'')$. From their characterisation, it is easy to see that

- A. $\mathcal{F}'' \subseteq \mathcal{F}' \subseteq \mathcal{F}$,
- B. $\{n' \in \mathcal{N} \mid n' \text{ is an } \mathcal{F}\text{-descendant of } n\} \subseteq \mathcal{F}''[\text{Miss}]_t$ and $\text{card}(\mathcal{F}''[\text{Miss}]_t) \geq \beta$.

This property holds because n is not an \mathcal{F}'' -descendant of t , and all \mathcal{F} -descendants of n are also \mathcal{F}'' -descendants of n . Thus, every \mathcal{F} -descendant of n is in $\mathcal{F}''[\text{Miss}]_t$. By hypothesis, n has at least β descendants,

C. $n \in \text{dom}(\mathcal{F}')$ and $\mathcal{F}'[\text{Miss}]_t = \emptyset$ (as n is an \mathcal{F}' -descendant of t).

This property holds because \mathcal{F}' contains all the \mathcal{F} -descendants of n plus a path going from n to t , and nothing else. So, every element in $\text{dom}(\mathcal{F}')$ is a hit node for \mathcal{F}' .

Then, we conclude that:

D. from (B), $(\mathcal{F}'', t, n) \models \neg \text{inDom} \wedge \text{size}(\text{Miss}) \geq \beta$,

E. from (C). $(\mathcal{F}', t, n) \models [U] \neg \text{Miss} \wedge \text{Hit}$,

F. from $\mathcal{F}' = \mathcal{F}'' \cup \{(n, \mathcal{F}(n))\}$ and (D), $(\mathcal{F}', t, n) \models \blacklozenge(\neg \text{inDom} \wedge \text{size}(\text{Miss}) \geq \beta)$.

Lastly, by (A), (E) and (F), $(\mathcal{F}, t, n) \models \blacklozenge^* ([U] \neg \text{Miss} \wedge \text{Hit} \wedge \blacklozenge(\neg \text{inDom} \wedge \text{size}(\text{Miss}) \geq \beta))$. \square

Proof of (II). Let us recall the definition of $\#\text{child} \geq \beta$.

$$\#\text{child} \geq 0 \stackrel{\text{def}}{=} \text{Hit},$$

$$\#\text{child} \geq \beta + 1 \stackrel{\text{def}}{=} \#\text{desc} \geq \beta + 1 \wedge \underbrace{\blacksquare^\beta(\text{Hit} \Rightarrow \#\text{desc} \geq 1)}_{\text{whenever } \beta \text{ nodes of } \text{dom}(\mathcal{F}) \text{ are removed, if } n \text{ still reaches } t \text{ then it has at least one descendant}}$$

whenever β nodes of $\text{dom}(\mathcal{F})$ are removed, if n still reaches t then it has at least one descendant

The lemma is trivial for $\#\text{child} \geq 0$, so we consider $\#\text{child} \geq \beta + 1$ where $\beta \in \mathbb{N}$.

(\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \#\text{child} \geq \beta + 1$. From the first conjunct of $\#\text{child} \geq \beta + 1$:

A. n has at least $\beta + 1$ \mathcal{F} -descendants;

B. n is an \mathcal{F} -descendant of t .

Ad absurdum, suppose that n has $k < \beta + 1$ \mathcal{F} -children $\{n_1, \dots, n_k\}$. Let us consider a subset S of β descendants of n , such that $\{n_1, \dots, n_k\} \subseteq S$. From (A), S exists. Let \mathcal{F}' be the finite forest such that $\mathcal{F}' \subseteq \mathcal{F}$ and $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus S$. Since \mathcal{F}' is constructed from \mathcal{F} by only removing descendants of n , and in particular removing all its children, we have:

C. By (B), $(\mathcal{F}', t, n) \models \text{Hit}$,

D. $(\mathcal{F}', t, n) \models \#\text{desc} = 0$,

E. $\text{card}(\text{dom}(\mathcal{F}')) = \text{card}(\text{dom}(\mathcal{F})) - \beta$. Indeed, $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus S$ and $\text{card}(S) = \beta$.

However, from the semantics of \blacksquare^β and (C)–(E), $(\mathcal{F}, t, n) \models \blacksquare^\beta(\text{Hit} \wedge \#\text{desc} = 0)$, in contradiction with the second conjunct of $\#\text{child} \geq \beta + 1$. Thus, n has at least $\beta + 1$ \mathcal{F} -children and, from (B), it is an \mathcal{F} -descendant of t .

(\Leftarrow): Suppose (\mathcal{F}, t, n) to be a forest such that n has at least $\beta + 1$ children and it is a descendant of t . Trivially, as every \mathcal{F} -child is an \mathcal{F} -descendant, we obtain $(\mathcal{F}, t, n) \models \#\text{desc} \geq \beta + 1$. We now show that (\mathcal{F}, t, n) also satisfies $\blacksquare^\beta(\text{Hit} \Rightarrow \#\text{desc} \geq 1)$. Let $\mathcal{F}' \subseteq \mathcal{F}$ be the forest such that

A. n is an \mathcal{F}' -descendant of t ;

B. there is a set $S \subseteq \text{dom}(\mathcal{F})$ such that $\text{card}(S) = \beta$ and $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus S$.

To prove that $(\mathcal{F}, t, n) \models \blacksquare^\beta(\text{Hit} \Rightarrow \#\text{desc} \geq 1)$ it is sufficient to establish $(\mathcal{F}', t, n) \models \#\text{desc} \geq 1$. This is quite straightforward. From (A), we conclude that $(\mathcal{F}', t, n) \models \text{Hit}$. From (B), since S has cardinality β and n has $\beta + 1$ \mathcal{F} -children, n has at least one \mathcal{F}' -child (thus, an \mathcal{F}' -descendant). \square

Proof of Lemma 4.16.

Lemma 4.16. For each rank $\text{rk} \in \mathbb{N}^3$, ALT_{rk} is finite up to logical equivalence.

Proof. The proof is standard and relies on the analogous result from propositional logic [102]:

- (\star) given a set of formulae S that is finite up to logical equivalence, there are only finitely many Boolean combinations of formulae from S , up to logical equivalence.

The proof of the lemma is by induction on $\text{rk} \in \mathbb{N}^3$ with respect to the order $<_{\text{rk}}$.

base case: $\text{rk} = (0, 0, 0)$. Every formula of rank rk is a Boolean combination of formulae from $\{\text{Hit}, \text{Miss}\}$. By (\star) the set of formulae of rank $(0, 0, 0)$ is finite up to logical equivalence. For the inductive case, we partition the set of formulae of rank $\text{rk} = (m, s, k)$ in two disjoint sets and show that both of them are finite up to logical equivalence:

induction step: formulae dominated by $\langle U \rangle$, \diamond or \diamond^* . We consider the set of formulae dominated by the $\langle U \rangle$ operator, i.e. the set of every formula φ that is syntactically equivalent to $\langle U \rangle \psi$ for some $\psi \in \text{ALT}_{(m-1, s, k)}$. By induction hypothesis, there are only finitely many such ψ up to logical equivalence. As $\psi \equiv \chi$ implies $\langle U \rangle \psi \equiv \langle U \rangle \chi$, the set of formulae dominated by $\langle U \rangle$ is finite up to logical equivalence. The same reasoning holds for formulae dominated by \diamond or \diamond^* .

induction step: formulae that are not dominated by $\langle U \rangle$, \diamond or \diamond^* . We consider the set of formulae belonging to $\text{ALT}_{m, s, k}$ and that are not dominated by $\langle U \rangle$, \diamond or \diamond^* operators. Each formula φ of this set is therefore a Boolean combination of formulae $\varphi_1, \dots, \varphi_n$, syntactically different from φ , that have rank at most (m, s, k) and that are equal to **Hit** or **Miss**, or are dominated by $\langle U \rangle$, \diamond or \diamond^* operators. From the previous case as well as the base case, the set of such formulae $\varphi_1, \dots, \varphi_n$ is finite up to logical equivalence. Then, by (\star) we conclude that the set of formulae of ALT_{rk} that are not dominated by $\langle U \rangle$, \diamond or \diamond^* operators is also finite up to logical equivalence. \square

Proof of Lemma 4.25.

Lemma 4.25. Let $w \in \Sigma_\bullet^+$ and let (\mathcal{F}, t, n) be a pointed forest encoding the word $f(w) \in [1, 2n]^+$.

- (I) $(\mathcal{F}, t, n) \models \text{mark}_\Sigma$ iff n encodes a marked symbol of Σ_\bullet .
- (II) $(\mathcal{F}, t, n) \models \text{marks}_\Sigma \geq \beta$ iff \mathcal{F} contains at least β nodes encoding marked symbols of Σ_\bullet .
- (III) $(\mathcal{F}, t, n) \models \#\text{markAnc}_\Sigma \geq \beta$ iff n has at least β ancestors encoding marked symbols of Σ_\bullet .

We recall that a main node n' in \mathcal{F} encodes a marked symbol (resp. non marked symbol) whenever it has exactly $2a$ (resp. $2a + 1$) children that are character nodes, for some $a \in [1, n]$. The proofs of (I) and (III) are done by simply unrolling the definitions. Of these two statements, we just show the left-to-right direction. The right-to-left direction is quite straightforward. The proof of (II) is by induction on β .

Proof of (I). We recall the definition of mark_Σ :

$$\text{mark}_\Sigma \stackrel{\text{def}}{=} \bigvee_{a \in \Sigma} ((\#\text{child} = 2a \wedge \text{1st}_{[1, 2n]}) \vee (\#\text{child} = 2a + 1 \wedge \neg \text{1st}_{[1, 2n]})).$$

(\Rightarrow) : Suppose $(\mathcal{F}, t, n) \models \text{mark}_\Sigma$, so there must be a symbol $a \in \Sigma$ such that

$$(\mathcal{F}, t, n) \models (\#\text{child} = 2a \wedge \text{1st}_{[1, 2n]}) \vee (\#\text{child} = 2a + 1 \wedge \neg \text{1st}_{[1, 2n]}).$$

First, suppose $(\mathcal{F}, t, n) \models \#\text{child} = 2a \wedge \text{1st}_{[1, 2n]}$. From Lemma 4.9(II), n has exactly $2a$ children and it is a descendant of t . Moreover, from $\text{1st}_{[1, 2n]}$, n is the first node in the main path of \mathcal{F} . Then, by definition of encoding of a word (Definition 4.4), all the children of n are character nodes, and so n encodes a marked symbol. Otherwise, consider the case where the second disjunct holds, i.e. $(\mathcal{F}, t, n) \models \#\text{child} = 2a + 1 \wedge \neg \text{1st}_{[1, 2n]}$. From Lemma 4.9(II), n has exactly $2a + 1$ \mathcal{F} -children and it is a descendant of t . Again, from Definition 4.4, n is a node in the main path of \mathcal{F} . From $\neg \text{1st}_{[1, 2n]}$, n cannot be the first node in the main path of \mathcal{F} . So, one of its children is a main node, whereas all its other children are character nodes (by Definition 4.4). Thus, n has exactly $2a$ character nodes, meaning that it encodes a marked symbol. \square

Proof of (II). We recall the definition of $\text{marks}_\Sigma \geq \beta$, where $\beta \in \mathbb{N}$.

$$\begin{aligned}\text{marks}_\Sigma \geq 0 &\stackrel{\text{def}}{=} \top, \\ \text{marks}_\Sigma \geq \beta+1 &\stackrel{\text{def}}{=} \langle U \rangle (\text{mark}_\Sigma \wedge \Diamond(\neg \text{inDom} \wedge \text{marks}_\Sigma \geq \beta)).\end{aligned}$$

It should be noted that we cannot apply Lemma 4.7 to prove this statement, as its hypothesis (2) is not satisfied. The proof is by induction on β . The base case for $\beta = 0$ is trivial. Let us look at the *induction step*. Assume $\beta = k + 1$ for some $k \in \mathbb{N}$.

(\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \text{marks}_\Sigma \geq k+1$, and therefore there is a node $n' \in \mathcal{N}$ such that

- A. $(\mathcal{F}, t, n') \models \text{mark}_\Sigma$. From Lemma 4.25(I), n' encodes a marked symbol in Σ_\bullet ,
- B. $(\mathcal{F}, t, n') \models \Diamond(\neg \text{inDom} \wedge \text{marks}_\Sigma \geq k)$.

From (B), there is a forest \mathcal{F}' such that

- C. $\mathcal{F}' \subseteq \mathcal{F}$ and $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$;
- D. $(\mathcal{F}', t, n') \models \neg \text{inDom} \wedge \text{marks}_\Sigma \geq k$.

From (D):

- E. by semantics of inDom , the node n' does not belong to $\text{dom}(\mathcal{F}')$;
- F. by induction hypothesis, \mathcal{F}' contains at least k nodes encoding marked symbols of Σ_\bullet .

(A) implies that n' is a \mathcal{F} -descendant of t that is a main node. Together with (C) and (E), this implies that $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus \{n'\}$. Because of this (by (F)) the k nodes encoding marked symbols w.r.t. \mathcal{F}' must be elements of the main path of \mathcal{F} that are ancestors of n' . Indeed, n' and all its \mathcal{F} -descendants are not \mathcal{F}' -descendants of t . Recall that a symbol is encoded by a main node by using the number of its children that are character nodes, and that main nodes are not character nodes. As $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus \{n'\}$ and n' is a main node, every node encoding a marked symbol in \mathcal{F}' also encodes a marked symbol in \mathcal{F} . Thus, by (F) and (A), \mathcal{F} contains at least $k + 1$ nodes encoding marked symbols in Σ_\bullet .

(\Leftarrow): Let $\{(n_1, n_2) \dots (n_{m-1}, n_m)\}$ be the main path of \mathcal{F} , so that, for every $i \in [1, m]$, n_i correspond to the main node encoding the i -th character of $f(w)$, where we recall that $f : \Sigma_\bullet \rightarrow [1, 2n]$ is the bijection defined as $f(a) \stackrel{\text{def}}{=} 2a$. Suppose that \mathcal{F} contains at least $k+1$ nodes encoding marked symbols of Σ_\bullet , or equivalently that there are $k+1$ positions $\{i_1, \dots, i_{k+1}\}$ such that, for every $j \in [1, k+1]$, n_{i_j} encodes a marked symbol. Consider n_{i_1} , the node encoding the first marked symbol of $f(w)$. Furthermore, let us consider the subforest $\mathcal{F}' = \mathcal{F} \setminus \{(n_{i_j}, n_{i_j+1})\}$. We have:

- A. as n_{i_1} encodes a marked symbol, from Lemma 4.25(I), $(\mathcal{F}, t, n_{i_1}) \models \text{mark}_\Sigma$,
- B. $n_{i_1} \notin \text{dom}(\mathcal{F}')$. So, $(\mathcal{F}', n_{i_1}, t) \models \neg \text{inDom}$,
- C. $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$.
- D. For every node n'' of the k nodes in $\{n_{i_2}, \dots, n_{i_{k+1}}\}$, the number of \mathcal{F}' -children of n'' that are character nodes is the same as the number of \mathcal{F} -children of n'' that are character nodes. Moreover, n'' is a \mathcal{F}' -descendant of n'' . This property holds as n_{i_1} is a descendant of every node in $\{n_{i_2}, \dots, n_{i_{k+1}}\}$.

From (D), \mathcal{F}' contains at least k nodes encoding marked symbols of Σ_\bullet . By induction hypothesis, $(\mathcal{F}', t, n_{i_1}) \models \text{marks}_\Sigma \geq k$. With (B) and (C), $(\mathcal{F}, t, n_{i_1}) \models \Diamond(\neg \text{inDom} \wedge \text{marks}_\Sigma \geq k)$. Lastly, by (A) and the definition of the modality $\langle U \rangle$, we conclude: $(\mathcal{F}, t, n) \models \text{marks}_\Sigma \geq k+1$. \square

Proof of (III). We recall the definition of $\#\text{markAnc}_\Sigma \geq \beta$:

$$\#\text{markAnc}_\Sigma \geq \beta \stackrel{\text{def}}{=} \text{symb} \wedge \Diamond(\neg \text{inDom} \wedge \text{marks}_\Sigma \geq \beta).$$

Let $n_1 n_2 \dots n_m$ be the nodes in the main path of \mathcal{F} , so that, for every $j \in [1, m]$, n_j corresponds to the node encoding the j -th character of $f(w)$.

(\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \#markAnc_{\Sigma}$. From the first conjunct of $\#markAnc_{\Sigma}$, n is a main node. Equivalently, there is $j \in [1, m]$ such that $n_j = n$. From the second conjunct of $\#markAnc_{\Sigma}$, we conclude that there is a finite forest $\mathcal{F}' \subseteq \mathcal{F}$ such that

- A. $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus \{n\}$. Indeed, n is a \mathcal{F} -descendant of t (as it is a main node), but $(\mathcal{F}', t, n) \not\models \text{inDom}$ implies $n \notin \text{dom}(\mathcal{F}')$. The equivalence then follows as the semantics of the sabotage modality \blacklozenge requires $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$.
- B. $(\mathcal{F}', t, n) \models \text{marks}_{\Sigma} \geq \beta$. Thus, by Lemma 4.25(II) \mathcal{F}' contains at least β main nodes encoding marked symbols of Σ_{\bullet} .

From (A), since \mathcal{F} encodes a finite word $f(w)$ and n is the j -th element of its main path, we conclude that the pointed forest (\mathcal{F}', t, n) encodes the suffix of the word $f(w)$ corresponding to the nodes $n_{j+1} \dots n_m$ (which characterises the main path of \mathcal{F}'). These nodes are all ancestors of n and, from (B), at least β of them encode marked symbols (w.r.t. both \mathcal{F}' and \mathcal{F} , as the former structure encodes a suffix of the word encoded by \mathcal{F}). So, n has at least β ancestors encoding marked symbols of Σ_{\bullet} . \square

Proof of Lemma 4.26.

Lemma 4.26. Let $w \in \Sigma_{\bullet}^+$ with a marked word with $\beta \geq 1$ marked symbols. Let (\mathcal{F}, t, n) be an encoding of $f(w)$. For every φ in PITL, $w \models_{\bullet} \varphi$ if and only if $(\mathcal{F}, t, n) \models \tau_{\beta}(\varphi)$.

Proof. By induction on the structure of φ , with the standard induction hypothesis stating that the lemma holds for every strict subformula of φ . Let $w = a_1 \dots a_k \in \Sigma_{\bullet}^+$ with β marked symbols. According to Definition 4.4, let $M = (n_1, \dots, n_k)$ be the tuple of main nodes of (\mathcal{F}, t, n) . The base case for \top is obvious, whereas for the two atomic formulae a and 1 is by easy verification.

base case: $\varphi = a$. Then,

$$\begin{aligned} w \models_{\bullet} a &\text{ iff } a_1 = a \text{ or } a_1 = \bar{a}, & (\text{by definition of } \models_{\bullet}) \\ &\text{ iff } n_1 \text{ encodes } 2a \text{ or } 2a - 1, & (\text{by definition of } \mathcal{F} \text{ and } f) \\ &\text{ iff } (\mathcal{F}, t, n_1) \models \text{1st}_{[2a-1, 2a]}, & (\text{by definition of } \text{1st}_{[2a-1, 2a]}) \\ &\text{ iff } (\mathcal{F}, t, n) \models \langle U \rangle \text{1st}_{[2a-1, 2a]}, & (\text{by semantics of } \langle U \rangle \text{ and def. of } n_1) \\ &\text{ iff } (\mathcal{F}, t, n) \models \tau_{\beta}(a). & (\text{by definition of } \tau_{\beta}) \end{aligned}$$

In the second to last step, we need to rely on the definition of encoding in order to perform the backward direction, i.e. derive $(\mathcal{F}, t, n_1) \models \text{1st}_{[2a-1, 2a]}$ from $(\mathcal{F}, t, n) \models \langle U \rangle \text{1st}_{[2a-1, 2a]}$. Indeed, recall that the first node in the main path, i.e. n_1 , is the only one satisfying $\text{1st}_{[i, j]}$ (for some $1 \leq i < j \leq 2n$). As above, we write “def. of n_1 ” when this property is used.

base case: $\varphi = 1$. Then,

$$\begin{aligned} w \models_{\bullet} 1 &\text{ iff } \bar{a} \in \bar{\Sigma} \text{ such that } a_1 = \bar{a}, & (\text{by definition of } \models_{\bullet}) \\ &\text{ iff } n_1 \text{ encodes } 2a - 1 \text{ for some } a \in \Sigma, & (\text{by definition of } \mathcal{F} \text{ and } f) \\ &\text{ iff } (\mathcal{F}, t, n_1) \models \text{1st}_{[1, 2n]} \wedge \text{mark}_{\Sigma}, & (\text{by definition of } \text{1st}_{[1, 2n]} \wedge \text{mark}_{\Sigma}) \\ &\text{ iff } (\mathcal{F}, t, n) \models \langle U \rangle (\text{1st}_{[1, 2n]} \wedge \text{mark}_{\Sigma}), & (\text{by semantics of } \langle U \rangle \text{ and def. of } n_1) \\ &\text{ iff } (\mathcal{F}, t, n) \models \tau_{\beta}(1). & (\text{by definition of } \tau_{\beta}) \end{aligned}$$

As in Lemma 4.24, the cases for Boolean connectives are obvious, so let us focus on $\varphi = \varphi_1 \mid \varphi_2$. Let $w' \in \Sigma^*$, $\bar{a} \in \bar{\Sigma}$ and $w'' \in \Sigma_\bullet^*$ be such that $\Delta(w) = (w', \bar{a}, w'')$. As \bar{a} is the leftmost marked symbol in w , notice that w'' contains $\beta - 1$ marked symbols.

induction step: $\varphi = \varphi_1 \mid \varphi_2$. We recall that $w \models_{\bullet} \varphi_1 \mid \varphi_2$ if and only if there is $b \in \Sigma$ such that

- (a) $w' = \epsilon$, $b = a$ and $\bar{a} w'' \models_{\bullet} \varphi_1 \wedge \varphi_2$
- or (b) $w' = bw_2$ and $\bar{b} w_2 \bar{a} w'' \models_{\bullet} \varphi_1$ and $b w_2 \bar{a} w'' \models_{\bullet} \varphi_2$, for some $w_2 \in \Sigma^*$
- or (c) $w' \neq \epsilon$ and $b = a$ and $w' \bar{a} w'' \models_{\bullet} \varphi_1$ and $\bar{a} w'' \models_{\bullet} \varphi_2$
- or (d) $w' = w_1 bw_2$ and $w_1 \bar{b} w_2 \bar{a} w'' \models_{\bullet} \varphi_1$ and $b w_2 \bar{a} w'' \models_{\bullet} \varphi_2$,
for some $w_1 \in \Sigma^+$ and $w_2 \in \Sigma^*$.

We split the proof into four double implications, making a correspondence between the four cases (a)–(d) in the semantics of $w \models_{\bullet} \varphi_1 \mid \varphi_2$. and the four disjuncts in the definition of $\tau_{\beta}(\varphi_1 \mid \varphi_2)$. More precisely, we prove:

- A. there is $b \in \Sigma$ such that $w' = \epsilon$, $b = a$ and $\bar{a} w'' \models_{\bullet} \varphi_1 \wedge \varphi_2$, if and only if
 $(\mathcal{F}, t, n) \models \langle U \rangle (\text{symb} \wedge \text{1st}_{[1,2n]} \wedge \text{mark}_{\Sigma} \wedge \tau_{\beta}(\varphi_1) \wedge \tau_{\beta}(\varphi_2))$,
- B. there are $b \in \Sigma$ and $w_2 \in \Sigma^*$ s.t. $w' = bw_2$, $\bar{b} w_2 \bar{a} w'' \models \varphi_1$ and $b w_2 \bar{a} w'' \models \varphi_2$, iff
 $(\mathcal{F}, t, n) \models \langle U \rangle (\text{symb} \wedge \text{1st}_{[1,2n]} \wedge \neg \text{mark}_{\Sigma} \wedge \blacklozenge(\text{mark}_{\Sigma} \wedge \tau_{\beta+1}(\varphi_1)) \wedge \tau_{\beta}(\varphi_2))$,
- C. there is $b \in \Sigma$ such that $w' \neq \epsilon$, $b = a$, $w' \bar{a} w'' \models \varphi_1$ and $\bar{a} w'' \models \varphi_2$, if and only if
 $(\mathcal{F}, t, n) \models \langle U \rangle (\text{symb} \wedge \neg \text{1st}_{[1,2n]} \wedge \text{mark}_{\Sigma} \wedge \#\text{markAnc}_{\Sigma} \geq \beta - 1$
 $\wedge \tau_{\beta}(\varphi_1) \wedge \blacklozenge(\text{1st}_{[1,2n]} \wedge \tau_{\beta}(\varphi_2)))$,
- D. there are $b \in \Sigma$, $w_1 \in \Sigma^+$ and $w_2 \in \Sigma^*$ such that $w' = w_1 bw_2$, $w_1 \bar{b} w_2 \bar{a} w'' \models \varphi_1$ and $b w_2 \bar{a} w'' \models \varphi_2$, if and only if
 $(\mathcal{F}, t, n) \models \langle U \rangle (\text{symb} \wedge \neg \text{1st}_{[1,2n]} \wedge \neg \text{mark}_{\Sigma} \wedge \#\text{markAnc}_{\Sigma} \geq \beta$
 $\wedge \blacklozenge(\text{mark}_{\Sigma} \wedge \tau_{\beta+1}(\varphi_1)) \wedge \blacklozenge(\text{1st}_{[1,2n]} \wedge \tau_{\beta}(\varphi_2)))$.

Proving these four correspondences suffices to prove the lemma. Indeed, the disjunction of the four ALT formulae in these four cases is equivalent to $\tau_{\beta}(\varphi_1 \mid \varphi_2)$ since the somewhere modality distributes over disjunction, i.e. $\langle U \rangle (\varphi \vee \psi) \Leftrightarrow \langle U \rangle \varphi \vee \langle U \rangle \psi$, and conjunction distributes over disjunction. In the proofs below, we often use the fact that if (\mathcal{F}, t, n) encodes $f(w)$, then for every node n' , (\mathcal{F}, t, n') encodes $f(w)$. This follows directly from the definition of encoding, which does not depend on the current world n .

proof of (A): Let us consider the first double implication.

- there is $b \in \Sigma$ such that $w' = \epsilon$, $b = a$ and $\bar{a} w'' \models_{\bullet} \varphi_1 \wedge \varphi_2$,
- $\Leftrightarrow w$ is headed by the marked symbol \bar{a} and $w \models_{\bullet} \varphi_1 \wedge \varphi_2$,
- \Leftrightarrow
 - 1. n_1 encodes a marked symbols, i.e. $(\mathcal{F}, t, n_1) \models \text{symb} \wedge \text{1st}_{[1,2n]} \wedge \text{mark}_{\Sigma}$,
(by definition of \mathcal{F} and Lemma 4.25. Note: n_1 always satisfies $\text{symb} \wedge \text{1st}_{[1,2n]}$)
 - 2. $(\mathcal{F}, t, n_1) \models \tau_{\beta}(\varphi_1) \wedge \tau_{\beta}(\varphi_2)$,
(by induction hypothesis)
- $\Leftrightarrow (\mathcal{F}, t, n_1) \models \text{symb} \wedge \text{1st}_{[1,2n]} \wedge \text{mark}_{\Sigma} \wedge \tau_{\beta}(\varphi_1) \wedge \tau_{\beta}(\varphi_2)$
(by definition of \models)
- $\Leftrightarrow (\mathcal{F}, t, n) \models \langle U \rangle (\text{symb} \wedge \text{1st}_{[1,2n]} \wedge \text{mark}_{\Sigma} \wedge \tau_{\beta}(\varphi_1) \wedge \tau_{\beta}(\varphi_2))$.
(by semantics of $\langle U \rangle$ and def. of n_1)

proof of (B): For the second double implication,

- there are $b \in \Sigma$ and $w_2 \in \Sigma^*$ s.t. $w' = bw_2$, $\bar{b}w_2\bar{a}w'' \models_{\bullet} \varphi_1$ and $bw_2\bar{a}w'' \models_{\bullet} \varphi_2$,
- $\Leftrightarrow w$ is headed by a (non marked) symbol in Σ , $w \models_{\bullet} \varphi_2$ and the word obtained from w by marking the first symbol satisfies φ_2 ,
- \Leftrightarrow
 1. n_1 does not encode a marked symbol, i.e. $(\mathcal{F}, t, n_1) \models \text{symb} \wedge \text{1st}_{[1,2n]} \wedge \neg \text{mark}_{\Sigma}$,
(by definition of \mathcal{F} and Lemma 4.25. Note: n_1 satisfies $\text{symb} \wedge \text{1st}_{[1,2n]}$)
 2. $(\mathcal{F}, t, n_1) \models \tau_{\beta}(\varphi_2)$,
(by induction hypothesis form $w \models \varphi_2$)
 3. There is $\mathcal{F}' \subseteq \mathcal{F}$ s.t. $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$, $(\mathcal{F}', t, n_1) \models \text{mark}_{\Sigma} \wedge \tau_{\beta+1}(\varphi_1)$.
For this step, consider the finite forest $\mathcal{F}' \subseteq \mathcal{F}$ obtained from \mathcal{F} by removing one children of n_1 . As n_1 encodes the non marked symbol b w.r.t. \mathcal{F} , by definition it encodes the marked symbol \bar{b} w.r.t. \mathcal{F}' . Every other node has the same number of \mathcal{F}' -children as in \mathcal{F} . In other words, \mathcal{F}' encodes the word $\bar{b}w_2\bar{a}w''$, with $\beta + 1$ marked symbols, obtained from w by marking the first (non marked) symbol. By definition of \mathcal{F}' , $(\mathcal{F}', t, n_1) \models \text{mark}_{\Sigma}$. By induction hypothesis, $(\mathcal{F}', t, n_1) \models \tau_{\beta+1}(\varphi_1)$.
- $\Leftrightarrow (\mathcal{F}, t, n_1) \models \text{symb} \wedge \text{1st}_{[1,2n]} \wedge \neg \text{mark}_{\Sigma} \wedge \blacklozenge(\text{mark}_{\Sigma} \wedge \tau_{\beta+1}(\varphi_1)) \wedge \tau_{\beta}(\varphi_2)$,
(by definition of $\mathcal{F}' \subseteq \mathcal{F}$ and \blacklozenge)
- $\Leftrightarrow (\mathcal{F}, t, n) \models \langle U \rangle (\text{symb} \wedge \text{1st}_{[1,2n]} \wedge \neg \text{mark}_{\Sigma} \wedge \blacklozenge(\text{mark}_{\Sigma} \wedge \tau_{\beta+1}(\varphi_1)) \wedge \tau_{\beta}(\varphi_2))$.
(by semantics of $\langle U \rangle$ and def. of n_1)

proof of (C): For the third double implication,

- there is $b \in \Sigma$ such that $w' \neq \epsilon$, $b = a$, $w'\bar{a}w'' \models \varphi_1$ and $\bar{a}w'' \models \varphi_2$,
- \Leftrightarrow there is an index $j \in [2, k]$ (recall $w = a_1 \dots a_k$ and $\Delta(w) = (w', \bar{a}, w'')$) s.t.
 1. the main node n_j encodes a marked symbol, and among the main nodes in the set $\{n_{j+1}, \dots, n_k\}$ (ancestors of n_j), exactly $\beta - 1$ encode marked symbols.
Equivalently, $(\mathcal{F}, t, n_j) \models \text{symb} \wedge \neg \text{1st}_{[1,2n]} \wedge \text{mark}_{\Sigma} \wedge \#\text{markAnc}_{\Sigma} \geq \beta - 1$,
(by definition of \mathcal{F} , Lemma 4.25 and as $j > 1$)
 2. $(\mathcal{F}, t, n_j) \models \tau_{\beta}(\varphi_1)$,
(by induction hypothesis from $w \models \varphi_1$)
 3. There is $\mathcal{F}' \subseteq \mathcal{F}$ s.t. $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$, $(\mathcal{F}', t, n_j) \models \text{1st}_{[1,2n]} \wedge \tau_{\beta}(\varphi_2)$.
For this step, consider the finite forest $\mathcal{F}' \subseteq \mathcal{F}$ obtained from \mathcal{F} by removing n_{j-1} , i.e. the only main node such that $\mathcal{F}(n_{j-1}) = n_j$ (which exists as $j > 1$). It is quite straightforward to see that \mathcal{F}' encodes the word $a_j a_{j+1} \dots a_k = \bar{a}w''$. By definition, $(\mathcal{F}', t, n_j) \models \text{1st}_{[1,2n]}$. By induction hypothesis, $(\mathcal{F}', t, n_j) \models \tau_{\beta}(\varphi_2)$,
- \Leftrightarrow there is a main node n_j in the main path of \mathcal{F} such that
$$(\mathcal{F}, t, n_j) \models \text{symb} \wedge \neg \text{1st}_{[1,2n]} \wedge \text{mark}_{\Sigma} \wedge \#\text{markAnc}_{\Sigma} \geq \beta - 1$$

$$\wedge \tau_{\beta}(\varphi_1) \wedge \blacklozenge(\text{1st}_{[1,2n]} \wedge \tau_{\beta}(\varphi_2))$$

(by definition of $\mathcal{F}' \subseteq \mathcal{F}$ and \blacklozenge)
- $\Leftrightarrow (\mathcal{F}, t, n) \models \langle U \rangle (\text{symb} \wedge \neg \text{1st}_{[1,2n]} \wedge \text{mark}_{\Sigma} \wedge \#\text{markAnc}_{\Sigma} \geq \beta - 1$

$$\wedge \tau_{\beta}(\varphi_1) \wedge \blacklozenge(\text{1st}_{[1,2n]} \wedge \tau_{\beta}(\varphi_2)))$$
.
(by semantics of $\langle U \rangle$)

proof of (D): Lastly, for the fourth double implication,

there are $b \in \Sigma$, $w_1 \in \Sigma^+$ and $w_2 \in \Sigma^*$ s.t. $w' = w_1 b w_2$, $w_1 \bar{b} w_2 \bar{a} w'' \models \varphi_1$ and $b w_2 \bar{a} w'' \models \varphi_2$,

\Leftrightarrow there is $j \in [2, k]$ (recall $w = a_1 \dots a_k$ and $\Delta(w) = (w', \bar{a}, w'')$) such that

1. the main node n_j encodes a non marked symbol and has exactly β ancestors in $\{n_{j+1}, \dots, n_k\}$ that encode marked symbols (i.e. it encodes a symbol of w that strictly precedes all the β marked symbols in w). Equivalently, (\mathcal{F}, t, n_j) satisfies $\text{symb} \wedge \neg \text{1st}_{[1,2n]} \wedge \neg \text{mark}_\Sigma \wedge \#\text{markAnc}_\Sigma \geq \beta$,

(by definition of \mathcal{F} , Lemma 4.25 and as $j > 1$)

2. There is $\mathcal{F}' \subseteq \mathcal{F}$ s.t. $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$, $(\mathcal{F}', t, n_j) \models \text{mark}_\Sigma \wedge \tau_{\beta+1}(\varphi_1)$.

For this step, consider the finite forest $\mathcal{F}' \subseteq \mathcal{F}$ obtained from \mathcal{F} by removing one character node of n_j . As n_j encodes the non marked symbol b w.r.t. \mathcal{F} , by definition it encodes the marked symbol \bar{b} w.r.t. \mathcal{F}' . Every other node has the same number of \mathcal{F}' -children as in \mathcal{F} . In other words, \mathcal{F}' encodes the word $w_1 \bar{b} w_2 \bar{a} w''$, with $\beta + 1$ marked symbols, obtained from w by marking the j -th symbol (which belongs to w'). By definition of \mathcal{F}' , $(\mathcal{F}', t, n_j) \models \text{mark}_\Sigma$. By induction hypothesis, $(\mathcal{F}', t, n_j) \models \tau_{\beta+1}(\varphi_1)$.

3. There is $\mathcal{F}'' \subseteq \mathcal{F}$ s.t. $\text{card}(\mathcal{F}'') = \text{card}(\mathcal{F}) - 1$, $(\mathcal{F}'', t, n_j) \models \text{1st}_{[1,2n]} \wedge \tau_\beta(\varphi_2)$.

For this step, consider the subforest $\mathcal{F}'' \subseteq \mathcal{F}$ obtained from \mathcal{F} by removing n_{j-1} , i.e. the only main node such that $\mathcal{F}(n_{j-1}) = n_j$ (which exists as $j > 1$). So, \mathcal{F}'' encodes the word $a_j a_{j+1} \dots a_k = b w_2 \bar{a} w''$. By definition, $(\mathcal{F}'', t, n_j) \models \text{1st}_{[1,2n]}$. By induction hypothesis, $(\mathcal{F}'', t, n_j) \models \tau_\beta(\varphi_2)$,

\Leftrightarrow there is a main node n_j in the main path of \mathcal{F} such that

$$(\mathcal{F}, t, n_j) \models \text{symb} \wedge \neg \text{1st}_{[1,2n]} \wedge \neg \text{mark}_\Sigma \wedge \#\text{markAnc}_\Sigma \geq \beta \\ \wedge \blacklozenge(\text{mark}_\Sigma \wedge \tau_{\beta+1}(\varphi_1)) \wedge \blacklozenge(\text{1st}_{[1,2n]} \wedge \tau_\beta(\varphi_2)).$$

(by definition of $\mathcal{F}' \subseteq \mathcal{F}$, $\mathcal{F}'' \subseteq \mathcal{F}$ and \blacklozenge)

$\Leftrightarrow (\mathcal{F}, t, n) \models \langle U \rangle (\text{symb} \wedge \neg \text{1st}_{[1,2n]} \wedge \neg \text{mark}_\Sigma \wedge \#\text{markAnc}_\Sigma \geq \beta$

$$\wedge \blacklozenge(\text{mark}_\Sigma \wedge \tau_{\beta+1}(\varphi_1)) \wedge \blacklozenge(\text{1st}_{[1,2n]} \wedge \tau_\beta(\varphi_2))).$$

(by semantics of $\langle U \rangle$)

□

Proof of Lemma 4.31.

Lemma 4.31. Let (s, h) be an (x, y) -encoding of a pointed forest (\mathcal{F}, t, n) . Let φ be a formula in ALT. We have, $(\mathcal{F}, t, n) \models \varphi$ if and only if $(s, h) \models \tau_{x,y}(\varphi)$.

Proof. The proof is by structural induction on φ . The base case for $\varphi = \top$ is obvious.

base case: $\varphi = \text{Hit}$.

$$(\mathcal{F}, t, n) \models \text{Hit},$$

\Leftrightarrow there is $\delta \geq 1$ such that $\mathcal{F}^\delta(n) = t$,

(by definition of \models)

\Leftrightarrow there is $\delta \geq 2$ such that $h^\delta(s(x)) = t = s(y)$ and for $\delta' < \delta$, $h^{\delta'}(s(x)) \neq s(y)$,

($n \neq t$ and by definition of encoding, i.e. $h(s(x)) = n$, $s(y) = t$, $s(x) \neq s(y)$)

\Leftrightarrow there is $\delta \geq 0$ $h^\delta(s(x)) = s(y)$ and $h(s(x)) \neq s(y)$,

(left-to-right direction: weakening. right-to-left direction: definition of encoding)

$$\Leftrightarrow (s, h) \models x \hookrightarrow^* y \wedge \neg x \hookrightarrow y.$$

(definition of \models)

Before treating the base case for $\varphi = \text{Miss}$, let us prove the following intermediate result:

NDom. Let (s, h) be a memory state s.t. $s(x) \in \text{dom}(h)$ and for all $\delta \geq 0$, $h^\delta(s(x)) \neq s(y)$.
 $h(s(x)) \notin \text{dom}(h)$ iff $(s, h) \models \text{size} = 1 \dashv(1) (\top * (\text{ls}(x, y) \wedge \text{size} = 2))$.

Proof of (NDom). (\Rightarrow): Suppose $h(s(x)) \notin \text{dom}(h)$. Let $h' = \{(h(s(x)), s(y))\}$. So, $h' \perp h$ and $(s, h') \models \text{size} = 1$. From the two hypotheses $s(x) \in \text{dom}(h)$ and for all $\delta \geq 0$, $h^\delta(s(x)) \neq s(y)$, we conclude that there is a location ℓ s.t. $h'' \stackrel{\text{def}}{=} \{s(x) \mapsto \ell \mapsto s(y)\} \subseteq h + h'$, where $s(x) \neq s(y)$ and $\ell \neq s(y)$. Therefore, $(s, h'') \models \text{ls}(x, y) \wedge \text{size} = 2$, which in turn implies that $(s, h + h')$ satisfies $\top * (\text{ls}(x, y) \wedge \text{size} = 2)$. By definition of h' , $(s, h) \models \text{size} = 1 \dashv(1) (\top * (\text{ls}(x, y) \wedge \text{size} = 2))$.
(\Leftarrow): Suppose $(s, h) \models \text{size} = 1 \dashv(1) (\top * (\text{ls}(x, y) \wedge \text{size} = 2))$, and so there is a heap h' such that $\text{card}(h') = 1$ and a heap $h'' \subseteq h + h'$ such that $(s, h'') \models \text{ls}(x, y) \wedge \text{size} = 2$. So, $h'' = \{s(x) \mapsto \ell \mapsto s(y)\}$ for some location $\ell \neq s(y)$. From the hypothesis $s(x) \in \text{dom}(h)$ and for all $\delta \geq 0$, $h^\delta(s(x)) \neq s(y)$, we conclude that $\{\ell \mapsto s(y)\} \subseteq h'$, and thus $h(s(x)) \notin \text{dom}(h)$.

(End of Proof of (NDom))

base case: $\varphi = \text{Miss}$. (\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \text{Miss}$, and so $n \in \text{dom}(\mathcal{F})$ and n is not a \mathcal{F} -descendant of t . By definition of encoding, $h(s(x)) = n \neq s(x)$ and for all $\delta \geq 0$, we have $h^\delta(s(x)) \neq s(y) = t$. Therefore, $(s, h) \not\models x \hookrightarrow y$ and $h(s(x)) \in \text{dom}(\mathcal{F})$. From (NDom), this implies $(s, h) \not\models \text{size} = 1 \dashv(1) (\top * (\text{ls}(x, y) \wedge \text{size} = 2))$, whereas from the previous base case we conclude that $(s, h) \not\models \tau_{x,y}(\text{Hit})$.

(\Leftarrow): Suppose $(s, h) \models \neg \tau_{x,y}(\text{Hit}) \wedge \neg x \hookrightarrow y \wedge (\text{size} = 1 \dashv(1) \neg (\top * (\text{ls}(x, y) \wedge \text{size} = 2)))$. Recalling that $\tau_{x,y}(\text{Hit}) = x \hookrightarrow^* y \wedge \neg x \hookrightarrow y$, the satisfaction of $\neg \tau_{x,y}(\text{Hit}) \wedge \neg x \hookrightarrow y$ implies that for every $\delta \geq 0$, $h^\delta(s(x)) = s(y)$. Since by definition of encoding $s(x) \in \text{dom}(h)$, by (NDom) we conclude that $h(s(x)) \in \text{dom}(h)$. So, $n = h(s(x)) \in \text{dom}(\mathcal{F})$. From the previous base case, $(\mathcal{F}, t, n) \not\models \text{Hit}$. Thus, $(\mathcal{F}, t, n) \models \text{Miss}$.

The induction steps for Boolean connectives are obvious. The cases for $\varphi = \blacklozenge \psi$ and $\varphi = \blacklozenge^* \psi$ are very similar. So, we just explicit the case for $\varphi = \blacklozenge \psi$.

induction step: $\varphi = \blacklozenge \psi$. (\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \blacklozenge \psi$, and so there is a subforest $\mathcal{F}' \subseteq \mathcal{F}$ such that $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$ and $(\mathcal{F}', t, n) \models \psi$. Consider the heap $h' \stackrel{\text{def}}{=} \mathcal{F}' + \{s(x) \mapsto n\}$. Since $n \neq s(x)$ and $s(y) = t \notin \text{dom}(h)$ (from the fact that (s, h) is a (x, y) -encoding of (\mathcal{F}, t, n)), we conclude that (s, h') is a (x, y) -encoding of (\mathcal{F}', t, n) . By induction hypothesis $(s, h) \models \tau_{x,y}(\psi)$. By definition of h' , $(s, h') \models x \hookrightarrow _, h' \subseteq h$ and $\text{card}(h') = \text{card}(h) - 1$. Thus, $(s, h) \models \tau_{x,y}(\blacklozenge \psi)$.

(\Leftarrow): Suppose $(s, h) \models \blacklozenge_{\text{SL}}(x \hookrightarrow _ \wedge \tau_{x,y}(\psi))$, and so there is a subheap $h' \subseteq h$ such that $\text{card}(h') = \text{card}(h) - 1$ and $(s, h') \models x \hookrightarrow _ \wedge \tau_{x,y}(\psi)$. Thus, together with the fact that $h = \mathcal{F} + \{s(x) \mapsto n\}$, we conclude that there is a subforest $\mathcal{F}' \subseteq \mathcal{F}$ such that $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$ and $h' = \mathcal{F}' + \{s(x) \mapsto n\}$. So, (s, h') is a (x, y) -encoding of (\mathcal{F}', t, n) . By induction hypothesis, $(\mathcal{F}', t, n) \models \psi$. From the semantics of the modality \blacklozenge , $(\mathcal{F}, t, n) \models \blacklozenge \psi$.

induction step: $\varphi = \langle U \rangle \psi$. (\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \langle U \rangle \psi$, and so there is $n' \in \mathcal{N}$ such that $(\mathcal{F}, t, n') \models \psi$. Let us consider the heap $h' \stackrel{\text{def}}{=} \mathcal{F} + \{s(x) \mapsto n'\}$. Since $n \neq s(x)$ and $s(y) = t \notin \text{dom}(h)$ (from the fact that (s, h) is a (x, y) -encoding of (\mathcal{F}, t, n)), we conclude that (s, h') is a (x, y) -encoding of (\mathcal{F}, t, n') . Clearly, $(s, h') \models x \hookrightarrow _ \wedge \neg x \hookrightarrow x$, and by induction hypothesis $(s, h') \models \tau_{x,y}(\psi)$. By definition of h' and the semantics of $\dashv(1)$,

$(s, \mathcal{F}) \models \text{size} = 1 \rightarrow (x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \tau_{x,y}(\psi))$. Lastly, from $h = \mathcal{F} + \{(s(x), n)\}$, we conclude that $(s, h) \models \tau_{x,y}(\psi)$.

(\Leftarrow): Suppose $(s, h) \models \tau_{x,y}(\psi)$. So, there are two disjoint heaps h_1 and h_2 s.t. $h = h_1 + h_2$, $(s, h_1) \models \text{size} = 1 \wedge x \hookrightarrow _$ and $(s, h_2) \models \text{size} = 1 \rightarrow (x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \tau_{x,y}(\psi))$. By definition of encoding $h = \mathcal{F} + \{(s(x), n)\}$, which allows us to conclude that $h_1 = \{(s(x), n)\}$ and $h_2 = \mathcal{F}$. So, there is a heap h' disjoint from \mathcal{F} and such that $(s, h') \models \text{size} = 1$ and $(s, \mathcal{F} + h') \models x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \tau_{x,y}(\psi)$. As $(s, \mathcal{F} + h') \models x \hookrightarrow _ \wedge \neg x \hookrightarrow x$ and $s(x) \notin \text{dom}(\mathcal{F})$, there must be a node $n' \neq s(x)$ such that $h' = \{s(x) \mapsto n'\}$. By definition of encoding, $(s, \mathcal{F} + h')$ is an (x, y) -encoding of (\mathcal{F}, t, n') . By induction hypothesis, $(\mathcal{F}, t, n') \models \psi$, which implies $(\mathcal{F}, t, n) \models \langle U \rangle \psi$ by semantics of the modality $\langle U \rangle$. \square

Proof of Lemma 4.32.

Lemma 4.32. φ in ALT is satisfiable iff so is $x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \neg y \hookrightarrow _ \wedge \tau_{x,y}(\varphi)$ in $SL(*, \rightarrow [1], 1s)$.

Proof. (\Rightarrow): Suppose φ satisfiable, and let (\mathcal{F}, t, n) be a pointed forest satisfying it. Thanks to Lemma 4.18(I), without loss of generality we can assume that $t \notin \text{dom}(\mathcal{F})$. Let us consider a memory state (s, h) such that $h = \mathcal{F} + \{s(x) \mapsto n\}$, $n \neq s(x)$ and $s(y) = t \notin \text{dom}(h)$. By Definition 4.30, this memory state is a (x, y) -encoding of (\mathcal{F}, t, n) . The three (dis)equalities characterising (s, h) directly imply that $(s, h) \models x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \neg y \hookrightarrow _ \wedge \tau_{x,y}(\varphi)$. By Lemma 4.31, $(s, h) \models \tau_{x,y}(\varphi)$. (\Leftarrow): Suppose $x \hookrightarrow _ \wedge \neg x \hookrightarrow x \wedge \neg y \hookrightarrow _ \wedge \tau_{x,y}(\varphi)$ satisfiable, and let (s, h) be a memory state satisfying it. From the satisfaction of $x \hookrightarrow x$, there is a location ℓ such that $h(s(x)) = \ell$. We consider the pointed forest (\mathcal{F}, t, n) where $\mathcal{F} = h \setminus \{s(x) \mapsto h(s(x))\}$, $n = \ell$, and $t = s(y)$. From $(s, h) \not\models x \hookrightarrow x$ we have $s(x) \neq n$. From $(s, h) \not\models y \hookrightarrow _$ we have $t \notin \text{dom}(h)$. Thus, by Definition 4.30 (s, h) is a (x, y) -encoding of (\mathcal{F}, t, n) . By Lemma 4.31, $(\mathcal{F}, t, n) \models \varphi$. \square

Proof of Lemma 4.39.

Lemma 4.39. Let (\mathcal{F}, t, n) be a pointed forest such that $t \notin \text{dom}(\mathcal{F})$, and let (\mathcal{K}, w) be a (S, u) -encoding of (\mathcal{F}, t, n) . Given a formula φ in ALT, $(\mathcal{F}, t, n) \models \varphi$ if and only if $(\mathcal{K}, w) \models \tau_u(\varphi)$.

Proof. Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. Let $f : \mathcal{N} \rightarrow R^*(w)$ be an injection witnessing that (\mathcal{K}, w) is a (S, u) -encoding of (\mathcal{F}, n, t) ($u \in \{t, E\}$). We recall that this means that:

- 1_f. $f(t) \stackrel{\text{def}}{=} w$ is the only world in $\text{ran}(f) \cap \mathcal{V}(t)$, and $f(n)$ is the only world in $\text{ran}(f) \cap \mathcal{V}(n)$,
- 2_f. for every $n' \in \text{dom}(\mathcal{F})$ it holds that $(f(\mathcal{F}(n')), f(n')) \in R$,
- 3_f. for every infinite path $(w_0, w_1 \dots) \in \Pi_R(w)$ there is $i \in \mathbb{N}$ such that
 - a. $w_i \in \mathcal{V}(\text{end})$ and for every $j \in [0, i - 1]$ we have $w_j \notin \mathcal{V}(\text{end})$,
 - b. for every $j \in \mathbb{N}$, $(w_j \in \mathcal{V}(u)$ and $j < i$) if and only if there is $n' \in \text{dom}(\mathcal{F})$ $f(n') = w_j$.

Below, we call the three properties (1_f)–(3_f) *hypotheses* whenever they refer to (\mathcal{K}, w) and (\mathcal{F}, n, t) . Instead, we call them *properties* when referring to their analogue on other two structures, for which we want to prove their satisfaction. The proof is by structural induction on φ .

base case: $\varphi = \text{Hit}$.

$$\begin{aligned}
 & (\mathcal{F}, n, t) \models \text{Hit} \\
 \Leftrightarrow & \text{there is } k \geq 1 \text{ and there are } k + 1 \text{ different nodes } n_0, n_1, \dots, n_k \text{ such that } n_0 = n, \\
 & n_k = t \text{ and for every } i \in [0, k - 1], \mathcal{F}(n_i) = n_{i+1}, \\
 & (\text{by definition of } \models)
 \end{aligned}$$

- \Leftrightarrow there are $k \geq 1$ and $k + 1$ different worlds $w_k = f(n_k), w_{k-1} = f(n_{k-1}), \dots, w_0 = f(n_0)$ such that
1. from hypothesis (1_f) , $w_0 = f(n)$, $\{w_0\} = V(n)$, and $w_k = f(t)$, $\{w_k\} = V(t)$,
 2. from hypothesis (2_f) , for every $j \in [0, k - 1]$, $(w_{j+1}, w_j) \in R$,
 3. from hypothesis (3_f) , $w_k \notin V(u)$, for every $j \in [0, k - 1]$, $w_j \in V(u)$. Moreover, for every $j \in [0, k]$, $w_j \notin V(end)$.
- \Leftrightarrow there is a path $(w_k, w_{k-1}, \dots, w_0)$ in \mathcal{K} such that
1. $\{w_0\} = V(n)$, $w_0 \in V(u)$,
 2. $w = w_k \notin V(u)$,
 3. $j \in [1, k]$, $(w_j \in V(u) \text{ or } w_j \in V(t))$ and $w_j \notin V(end)$,
(manipulation from last step, and the hypothesis of f)
- $\Leftarrow (\mathcal{K}, w) \models E((u \vee t) \wedge \neg end \text{ M } u \wedge n)$
(by definition of \models).

base case: $\varphi = \text{Miss}$.

- $$(\mathcal{F}, n, t) \models \text{Miss}$$
- $\Leftrightarrow n \in \text{dom}(t)$ and $(\mathcal{F}, n, t) \not\models \text{Miss}$ (by definition of \models)
- \Leftrightarrow there is a path $(w_k, w_{k-1}, \dots, w_0)$ in \mathcal{K} (by hypothesis (2_f)) such that
1. from hypothesis (1_f) and (3_f) $w_k = w$, $w_0 \in V(n)$, $w_0 \notin V(end)$ and $w_0 \in V(u)$,
 2. from hypothesis (3_f) , for every $j \in [1, k]$ $w_j \notin V(end)$,
 3. $(\mathcal{K}, w) \not\models \tau_u(\text{Hit})$, (from the previous base case)
- $\Leftarrow (\mathcal{K}, w) \models E(\neg end \text{ M } u \wedge n) \wedge \neg \tau_u(\text{Hit}).$
(by definition of \models)

We omit the obvious cases for \top and Boolean connectives.

- induction step:** $\varphi = \langle U \rangle \psi$. (\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \langle U \rangle \psi$, and so there is $n' \in N$ such that $(\mathcal{F}, t, n') \models \psi$. Let us consider the Kripke tree $\mathcal{K}' = (\mathcal{W}, R, V[n \leftarrow \{f(n')\}])$ obtained from \mathcal{K} by updating the evaluation of n from $\{f(n)\}$ to $\{f(n')\}$ (see hypothesis (1_f) for $V(n) = \{f(n)\}$). It is easy to verify that (\mathcal{K}', w) is a (S, u) -encoding of (\mathcal{F}, t, n') . Indeed, it is quite obvious that the same injection f considered for the two structures (\mathcal{F}, t, n) and (\mathcal{K}, w) satisfy the properties (1_f) – (3_f) also with respect to the two structures (\mathcal{K}', w) and (\mathcal{F}, t, n') . By definition of \mathcal{K}' , clearly $(\mathcal{K}', w) \models \text{uniq}(n)$. By induction hypothesis, $(\mathcal{K}', w) \models \tau_u(\psi)$. Lastly, from the semantics of $\exists n$, we conclude that $(\mathcal{K}, w) \models \exists n (\text{uniq}(n) \wedge \tau_u(\psi))$.
- (\Leftarrow): The other direction follows with similar arguments (backwards). Briefly, suppose $(\mathcal{K}, w) \models \exists n (\text{uniq}(n) \wedge \tau_u(\psi))$. Then there is $\mathcal{K}' = (\mathcal{W}, R, V[n \leftarrow \mathcal{W}'])$, for some $\mathcal{W}' \subseteq \mathcal{W}$ such that $(\mathcal{K}', w) \models \text{uniq}(n) \wedge \tau_u(\varphi)$. From $\text{uniq}(n)$ we conclude that there is a world $w' \in R^*(w)$ such that $V(n) = \{w'\}$. Let n' be the node such that $f(n') = w'$. It is quite easy to see that (\mathcal{K}', w) is an (S, u) -encoding of (\mathcal{F}, t, n') . Again, f is a witness of this encoding. By induction hypothesis $(\mathcal{F}, t, n') \models \psi$. Thus, $(\mathcal{F}, t, n) \models \langle U \rangle \psi$.

- induction step:** $\varphi = \blacklozenge \psi$. (\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \blacklozenge \psi$, and so there is a subforest $\mathcal{F}' \subseteq \mathcal{F}$ such that $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$ and $(\mathcal{F}', t, n) \models \psi$. Let \hat{n} be the (only) node such that $\text{dom}(\mathcal{F}) = \text{dom}(\mathcal{F}') \cup \{\hat{n}\}$. Notice that $\hat{n} \in \text{dom}(\mathcal{F})$ and hence, by hypothesis (2_f) and (3_f) :

- A. $f(\hat{n}) \in V(u)$;
- B. There is a path (w_0, \dots, w_k) in (\mathcal{K}, w) going from $w_0 = w$ to $w_k = f(\hat{n})$. Moreover, for every $j \in [0, k]$ we have $w_j \notin V(end)$.

Let us consider the Kripke tree $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'])$ where $\mathcal{W}' \stackrel{\text{def}}{=} \mathcal{V}(u) \setminus \{f(\hat{n})\}$. Notice that then $\mathcal{W}' \subseteq \mathcal{V}(u)$ and from (A) we have $\text{card}(\mathcal{W}') = \text{card}(\mathcal{V}(u)) - 1$. Thus, $(\mathcal{K}', w) \models AG(\bar{u} \Rightarrow u) \wedge \text{uniq}(u \wedge \neg \bar{u})$. Furthermore, from (B) we conclude that (\mathcal{K}', w) also satisfies $E(\neg end M u \wedge \neg \bar{u})$. It remains to show that $(\mathcal{K}', w) \models \tau_{\bar{u}}(\psi)$, which follows by induction hypothesis, as we show that (\mathcal{K}', w) is a (S, \bar{u}) -encoding of (\mathcal{F}', t, n) (notice that now the encoding uses \bar{u} instead of u). More precisely, it is sufficient to check that f satisfies the properties (1_f)–(3_f) for the two structures (\mathcal{K}', w) and (\mathcal{F}', t, n) . We show the three properties separately.

Proof of property (1_f). We show that

- $f(t) \stackrel{\text{def}}{=} w$ is the only element in $\text{ran}(f)$ such that $w \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](t)$, and
- $f(n)$ is the only element in $\text{ran}(f)$ such that $f(n) \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](n)$.

Clearly, both statements hold directly from hypothesis (1_f). Indeed, $\mathcal{V}[\bar{u} \leftarrow \mathcal{W}']$ only updates the evaluation of \bar{u} , so $\mathcal{V}(n) = \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](n)$ and $\mathcal{V}(t) = \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](t)$.

Proof of property (2_f). We show that $(f(\mathcal{F}(n')), f(n')) \in R$ holds for every $n' \in \text{dom}(\mathcal{F})$. This holds directly from $\mathcal{F}' \subseteq \mathcal{F}$ and hypothesis (2_f).

Proof of property (3_f). We show that for every infinite path $(w_0, w_1 \dots) \in \Pi_R(w)$ there is $i \in \mathbb{N}$ such that

1. $w_i \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\text{end})$ and for every $j \in [0, i-1]$ we have $w_j \notin \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\text{end})$,
2. for all $j \in \mathbb{N}$, $(w_j \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u}) \text{ and } j < i)$ iff there is $n' \in \text{dom}(\mathcal{F}')$ $f(n') = w_j$.

The proof of (1) holds directly from hypothesis (3_f)(a), as $\mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\text{end}) = \mathcal{V}(\text{end})$.

To prove (2), let $(w_0, w_1 \dots) \in \Pi_R(w)$ and $i \in \mathbb{N}$ so that (1) is satisfied. Let $j \in \mathbb{N}$. First, consider $w_j \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u})$ and $j < i$. As $\mathcal{W}' = \mathcal{V}(u) \setminus \{f(\hat{n})\}$, we conclude that $w_j \in \mathcal{V}(u)$ and $w_j \neq f(\hat{n})$. From hypothesis (3_f)(b), there is $n' \in \text{dom}(\mathcal{F})$ such that $f(n') = w_j$. As $w_j \neq f(\hat{n})$, $n' \neq \hat{n}$ and so $n' \in \text{dom}(\mathcal{F}')$. Conversely, suppose that there is $n' \in \text{dom}(\mathcal{F}')$ such that $f(n') = w_j$. In particular, $n' \neq \hat{n}$. From (3_f)(b), $w_j \in \mathcal{V}(u)$ and $j < i$. As $\mathcal{W}' = \mathcal{V}(u) \setminus \{f(\hat{n})\}$, we conclude $w_j \in \mathcal{W}'$. So, $w_j \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u})$.

This concludes the proof that (\mathcal{K}', w) is an (S, \bar{u}) -encoding of (\mathcal{F}', t, n) , which allows us to conclude that $(\mathcal{K}, w) \models \tau_u(\Diamond \psi)$ from the semantics of the modality \Diamond .

(\Leftarrow): For the converse direction, let us assume that

$$(\mathcal{K}, w) \models \exists \bar{u} (AG(\bar{u} \Rightarrow u) \wedge \text{uniq}(u \wedge \neg \bar{u}) \wedge E(\neg end M u \wedge \neg \bar{u}) \wedge \tau_{\bar{u}}(\psi)).$$

There is $\mathcal{W}' \subseteq \mathcal{W}$ and $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'])$ such that

$$(\mathcal{K}', w) \models AG(\bar{u} \Rightarrow u) \wedge \text{uniq}(u \wedge \neg \bar{u}) \wedge E(\neg end M u \wedge \neg \bar{u}) \wedge \tau_{\bar{u}}(\psi).$$

By $(\mathcal{K}', w) \models AG(\bar{u} \Rightarrow u) \wedge \text{uniq}(u \wedge \neg \bar{u})$, there is a world $\hat{w} \in R^*(w) \cap \mathcal{V}(u)$ such that

$$(\dagger) \quad R^*(w) \cap \mathcal{W}' = R^*(w) \cap \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u}) = (R^*(w) \cap \mathcal{V}(u)) \setminus \{\hat{w}\}.$$

Moreover, from $(\mathcal{K}', w) \models E(\neg end M u \wedge \neg \bar{u})$, we conclude that the only path $(w_0, w_1 \dots, w_k)$ going from $w_0 = w$ to $w_k = \hat{w}$ is such that for all $i \in [0, k]$, $w_i \notin \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\text{end}) = \mathcal{V}(\text{end})$. From hypothesis (3_f) and $\hat{w} \in \mathcal{V}(u)$, we conclude that there is a node $\hat{n} \in \text{dom}(\mathcal{F})$ such that $f(\hat{n}) = \hat{w}$. Let us then consider the finite forest $\mathcal{F}' \subseteq \mathcal{F}$ such that $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus \{\hat{n}\}$. From (†) and the hypothesis (1_f)–(3_f) we can show that (\mathcal{K}', w) is an (S, \bar{u}) -encoding of (\mathcal{F}', t, n) . More precisely, f is a witness of the encoding between these two structures. Details are omitted, as the proof is analogous to the left-to-right direction. By induction hypothesis, $(\mathcal{F}', t, n) \models \psi$. As $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$ and $\mathcal{F}' \subseteq \mathcal{F}$, $(\mathcal{F}, t, n) \models \Diamond \psi$.

induction step: $\varphi = \blacklozenge^* \psi$. This case is very similar to the case for $\varphi = \blacklozenge \psi$.

(\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \blacklozenge^* \psi$, and so there is $\mathcal{F}' \subseteq \mathcal{F}$ such that $(\mathcal{F}', t, n) \models \psi$. Let us consider the set $\mathcal{W}' = \{w' \in \mathcal{W} \mid \text{there is } n' \in \text{dom}(\mathcal{F}') \text{ s.t. } f(n') = w'\}$. Informally, this is the set of worlds that corresponds to nodes in $\text{dom}(\mathcal{F}')$. Consider the Kripke tree $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'])$. By hypothesis (3_f), for every $n' \in \text{dom}(\mathcal{F})$ we have $f(n') \in \mathcal{V}(u)$, which in turn implies $\mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u}) = \mathcal{W}' \subseteq \mathcal{V}(u) = \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](u)$ (as $\mathcal{F}' \subseteq \mathcal{F}$). This implies that (\mathcal{K}', w) satisfies $\text{AG}(\bar{u} \Rightarrow u)$. It remains then to show that $(\mathcal{K}', w) \models \tau_{\bar{u}}(\psi)$, which follows by induction hypothesis, as we show that (\mathcal{K}', w) is a (S, \bar{u}) -encoding of (\mathcal{F}', t, n) . As in the induction step dealing with the modality \blacklozenge , it is sufficient to check that f satisfies properties (1_f)–(3_f) for the two structures (\mathcal{K}', w) and (\mathcal{F}', t, n) . The proof of the properties (1_f) and (2_f) is analogous to the one given for the induction step $\varphi = \blacklozenge \psi$. Therefore, let us focus on the proof of property (3_f):

Proof of property (3_f). We show that for every infinite path $(w_0, w_1 \dots) \in \Pi_R(w)$ there is $i \in \mathbb{N}$ such that

1. $w_i \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\text{end})$ and for every $j \in [0, i - 1]$ we have $w_j \notin \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\text{end})$,
2. for all $j \in \mathbb{N}$, $(w_j \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u}) \text{ and } j < i)$ iff there is $n' \in \text{dom}(\mathcal{F}')$ $f(n') = w_j$.

The proof of (1) holds directly from hypothesis (3_f)(a), as $\mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\text{end}) = \mathcal{V}(\text{end})$. To prove (2), let $(w_0, w_1 \dots) \in \Pi_R(w)$ and $i \in \mathbb{N}$ so that (1) is satisfied. Let $j \in \mathbb{N}$. The left-to-right direction of (2) is obvious. Indeed, given $\mathcal{W} \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u}) = \mathcal{W}'$, by definition of \mathcal{W}' , there is a node $n' \in \text{dom}(\mathcal{F}')$ such that $f(n') = w_j$. For the left-to-right direction, suppose that there is a node $n' \in \text{dom}(\mathcal{F}')$ such that $f(n') = w_j$. Clearly, from the definition of \mathcal{W}' , $w_j \in \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u}) = \mathcal{W}'$. Moreover, $j < i$ directly from the fact that $n' \in \text{dom}(\mathcal{F})$ (by $\mathcal{F}' \subseteq \mathcal{F}$) and from the hypothesis (3_f)(b).

This concludes the proof that (\mathcal{K}', w) is an (S, \bar{u}) -encoding of (\mathcal{F}', t, n) , which allows us to conclude that $(\mathcal{K}, w) \models \tau_u(\blacklozenge^* \psi)$ from the semantics of the modality \blacklozenge^* .

(\Leftarrow): Suppose $(\mathcal{K}, w) \models \exists \bar{u} (\text{AG}(\bar{u} \Rightarrow u) \wedge \tau_{\bar{u}}(\psi))$, and so there is a $\mathcal{W}' \subseteq \mathcal{W}$ and a Kripke tree $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'])$ such that $(\mathcal{K}', w) \models \text{AG}(\bar{u} \Rightarrow u) \wedge \tau_{\bar{u}}(\psi)$. We define $\mathcal{F}' \subseteq \mathcal{F}$ such that $\text{dom}(\mathcal{F}') = \{n \in \text{dom}(\mathcal{F}) \mid f(n) \in \mathcal{W}'\}$. From $(\mathcal{K}', w) \models \text{AG}(\bar{u} \Rightarrow u)$ we derive that $\mathcal{W}' = \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](\bar{u}) \subseteq \mathcal{V}[\bar{u} \leftarrow \mathcal{W}'](u) = \mathcal{V}(u)$. By hypothesis (1_f)–(3_f) we can show that (\mathcal{K}', w) is a (S, \bar{u}) -encoding of (\mathcal{F}', t, n) . Details are omitted, as the proof is analogous to the left-to-right direction. By induction hypothesis, $(\mathcal{F}', t, n) \models \psi$ holds. From $\mathcal{F}' \subseteq \mathcal{F}$, we conclude: $(\mathcal{F}, t, n) \models \blacklozenge^* \psi$. \square

Proof of Lemma 4.40.

We recall that $\text{enc} \stackrel{\text{def}}{=} \neg t \wedge t \wedge \text{uniq}(t) \wedge \text{uniq}(n) \wedge \text{AF}(\text{end})$.

Lemma 4.40. A formula φ in ALT is satisfiable iff so is $\text{enc} \wedge \tau_t(\varphi)$ in QCTL^t.

Proof. (\Rightarrow): Let (\mathcal{F}, t, n) be a pointed forest satisfying φ . From Lemma 4.18(I), we can assume without loss of generality that $t \notin \text{dom}(\mathcal{F})$. This auxiliary property allows us to construct an (S, t) -encoding of (\mathcal{F}, t, n) . Let (\mathcal{K}, w) be such an encoding. By Lemma 4.39, $(\mathcal{K}, w) \models \tau_t(\varphi)$. From the property (1) of the encoding (Definition 4.38), $(\mathcal{K}, w) \models t \wedge \text{uniq}(t) \wedge \text{uniq}(n)$. From the property (3)(a) of the encoding, $(\mathcal{K}, w) \models \text{AF}(\text{end})$. Lastly, property (3)(b) of the encoding, $(\mathcal{K}, w) \models \neg t$, as t is not in the domain of \mathcal{F} .

(\Leftarrow): Let $(\mathcal{K}, \mathbf{w})$, with $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, be a pointed Kripke tree satisfying $\text{enc} \wedge \tau_t(\varphi)$. We show that $(\mathcal{K}, \mathbf{w})$ is an (S, t) -encoding of some pointed forest. From $(\mathcal{K}, \mathbf{w}) \models \text{AF}(\text{end})$, in every infinite path $(\mathbf{w}_0, \mathbf{w}_1, \dots) \in \Pi_R(\mathbf{w})$ there is $i \in \mathbb{N}$ such that $\mathbf{w}_i \in \mathcal{V}(\text{end})$ and for every $j \in [0, i - 1]$ we have $\mathbf{w}_j \notin \mathcal{V}(\text{end})$. (notice that this corresponds to property (a) of the encoding). Let us consider the set $U \stackrel{\text{def}}{=} \{\mathbf{w}_k \mid k \in \mathbb{N}, (\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_k, \dots) \in \Pi_R(\mathbf{w}), \text{ for every } j \in [0, k], \mathbf{w}_j \notin \mathcal{V}(\text{end})\}$. Informally, U is the set of those worlds that can be reached from \mathbf{w} without passing through or ending in a world satisfying end . Thanks to $\text{AF}(\text{end})$ and the fact that Kripke trees are finitely-branching, U is finite. W.l.o.g. let us assume $\mathcal{N} = \mathcal{W}$ (both sets are countably infinite). Let $(\mathcal{F}, \mathbf{t}, \mathbf{n})$ be the pointed forest characterised as follows:

- A. $\mathbf{t} = \mathbf{w}$ and $\{\mathbf{n}\} = \mathcal{V}(\mathbf{n})$,
- B. $\mathcal{F}(\mathbf{w}) = \mathbf{w}'$ if and only if $\mathbf{w} \in U$, $(\mathcal{K}, \mathbf{w}) \models \mathbf{t}$ and $\{\mathbf{w}'\} = R^{-1}(\mathbf{w})$.

Notice that this pointed forest is well-defined and unique. Indeed, from $(\mathcal{K}, \mathbf{w}) \models \text{uniq}(\mathbf{n})$, $\text{card}(\mathcal{V}(\mathbf{n})) = 1$, and since R encodes the children relation of a tree, R^{-1} is functional. Moreover, from $(\mathcal{K}, \mathbf{w}) \models \neg \mathbf{t}$, we conclude that $\mathbf{t} \notin \text{dom}(\mathcal{F})$. In order to conclude the proof, it is sufficient to show that $(\mathcal{K}, \mathbf{w})$ is an (S, t) -encoding of $(\mathcal{F}, \mathbf{t}, \mathbf{n})$, as we can then derive $(\mathcal{F}, \mathbf{t}, \mathbf{n}) \models \varphi$ by Lemma 4.39. As a witness of the encoding, let us take any injection $\mathbf{f} : \mathcal{N} \rightarrow R^*(\mathbf{w})$ such that for every $\mathbf{n} \in \text{dom}(\mathcal{F})$, $\mathbf{f}(\mathbf{n}) = \mathbf{n}$. We show the following properties:

- 1_f. $\mathbf{f}(\mathbf{t}) \stackrel{\text{def}}{=} \mathbf{w}$ is the only world in $\text{ran}(\mathbf{f}) \cap \mathcal{V}(\mathbf{t})$, and $\mathbf{f}(\mathbf{n})$ is the only world in $\text{ran}(\mathbf{f}) \cap \mathcal{V}(\mathbf{n})$,
- 2_f. for every $\mathbf{n}' \in \text{dom}(\mathcal{F})$, it holds that $(\mathbf{f}(\mathcal{F}(\mathbf{n}')), \mathbf{f}(\mathbf{n}')) \in R$,
- 3_f. for every infinite path $(\mathbf{w}_0, \mathbf{w}_1, \dots) \in \Pi_R(\mathbf{w})$ there is $i \in \mathbb{N}$ such that
 - a. $\mathbf{w}_i \in \mathcal{V}(\text{end})$ and for every $j \in [0, i - 1]$ we have $\mathbf{w}_j \notin \mathcal{V}(\text{end})$,
 - b. for every $j \in \mathbb{N}$, $(\mathbf{w}_j \in \mathcal{V}(\mathbf{t}) \text{ and } j < i)$ if and only if there is $\mathbf{n}' \in \text{dom}(\mathcal{F})$ $\mathbf{f}(\mathbf{n}') = \mathbf{w}_j$.

The property (1_f) holds directly from (A) and $(\mathcal{K}, \mathbf{w}) \models \text{uniq}(\mathbf{t})$. The property (2_f) holds directly from (B). We have already shown property (3_f)(a) when considering the formula $\text{AF}(\text{end})$. Lastly, (3_f)(b) is also quite direct. Let us consider some infinite path $(\mathbf{w}_0, \mathbf{w}_1, \dots) \in \Pi_R(\mathbf{w})$ and some $i \in \mathbb{N}$ such that $\mathbf{w}_i \in \mathcal{V}(\text{end})$ and for every $j \in [0, i - 1]$ we have $\mathbf{w}_j \notin \mathcal{V}(\text{end})$. For the left-to-right direction of (3_f)(b), we consider a world $\mathbf{w}_j \in \mathcal{V}(\mathbf{t})$ where $j < i$. By definition of U , $\mathbf{w}_j \in U$. As $\mathbf{w}_j \in \mathcal{V}(\mathbf{t})$, $\mathbf{w}_j \neq \mathbf{w}$ and so $j > 1$, and moreover $(\mathcal{K}, \mathbf{w}_j) \models \mathbf{t}$. By (B), we have $\mathbf{f}(\mathbf{w}_j) = \mathbf{w}_j \in \text{dom}(\mathcal{F})$. Conversely, suppose that $\mathbf{f}(\mathbf{w}_j) = \mathbf{w}_j \in \text{dom}(\mathcal{F})$. From (B), $\mathbf{w}_j \in U$ and $(\mathcal{K}, \mathbf{w}_j) \models \mathbf{t}$, which in turn implies that $\mathbf{w}_j \in \mathcal{V}(\mathbf{t})$ and $j < i$, completing the proof. \square

Proof of Theorem 4.41.

Theorem 4.41. The satisfiability problems of $\text{QCTL}^t(\text{EU}^0)$ and $\text{QCTL}^t(\text{EF}^1)$ are TOWER-c.

In order to show this result it is sufficient to prove the two equivalences $\text{E}(\varphi \mathbf{U} \psi) \equiv \chi_{\text{EU}}(\varphi, \psi)$ and $\text{AG}(\varphi \Rightarrow \text{AG} \psi) \equiv \chi_{\text{AG}} \text{AG}(\varphi, \psi)$. The theorem then follows from Lemma 4.40 together with the equivalence $\text{EG} \varphi \equiv \chi_{\text{EG}}(\varphi)$ showed at the end of Section 4.4.2.

Proof of $\text{E}(\varphi \mathbf{U} \psi) \equiv \chi_{\text{EU}}(\varphi, \psi)$. We recall the definition of $\chi_{\text{EU}}(\varphi, \psi)$:

$$\chi_{\text{EU}}(\varphi, \psi) \stackrel{\text{def}}{=} \exists p (\text{AG}(\neg \varphi \wedge \neg \psi \Rightarrow p) \wedge \text{AG}(p \Rightarrow \text{AG} p) \wedge \text{EF}(\psi \wedge \neg p)).$$

where p does not appear in φ or ψ .

(\Rightarrow): Suppose $(\mathcal{K}, \mathbf{w}) \models \text{E}(\varphi \mathbf{U} \psi)$, and so there are $(\mathbf{w}_0, \mathbf{w}_1, \dots) \in \Pi_R(\mathbf{w})$ and $j \in \mathbb{N}$ such that $(\mathcal{K}, \mathbf{w}_j) \models \psi$ and for all $i \in [0, j - 1]$, $(\mathcal{K}, \mathbf{w}_i) \models \varphi$. We focus on the finite prefix $(\mathbf{w}_0, \dots, \mathbf{w}_j)$ of the path above, and consider the Kripke tree $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W} \setminus \{\mathbf{w}_0, \dots, \mathbf{w}_j\}])$ obtained

form \mathcal{K} by changing the evaluation of p from $\mathcal{V}(p)$ to the set of every world that is not in the finite prefix (w_1, \dots, w_j) . Let us show that (\mathcal{K}', w) satisfies the following three subformulae of $\chi_{\text{EU}}(\varphi, \psi)$: $\text{AG}(\neg\varphi \wedge \neg\psi \Rightarrow p)$, $\text{AG}(p \Rightarrow \text{AG } p)$, and $\text{EF}(\psi \wedge \neg p)$. Notice that every world in $\{w_0, \dots, w_{j-1}\}$ satisfies φ , w_j satisfies ψ , and every other world satisfies p . This implies that $(\mathcal{K}', w) \models \text{AG}(\neg\varphi \wedge \neg\psi \Rightarrow p)$. Moreover, given a world $w' \in R^*(w)$ that satisfies p , it holds that $\{w_0, \dots, w_j\} \cap R^*(w') = \emptyset$, which in turn means that every world in $R^*(w')$ satisfies p . So, $(\mathcal{K}', w) \models \text{AG}(p \Rightarrow \text{AG } p)$. By definition $(\mathcal{K}', w_j) \models \psi \wedge \neg p$ holds, which implies $(\mathcal{K}', w) \models \text{EF}(\psi \wedge \neg p)$. By applying the definition of $\exists p$, we conclude: $(\mathcal{K}, w) \models \chi_{\text{EU}}(\varphi, \psi)$.

(\Leftarrow): Suppose that $(\mathcal{K}, w) \models \chi_{\text{EU}}(\varphi, \psi)$ and therefore there is a model $(\mathcal{K}', \mathcal{W})$, where $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W}'])$ for some $\mathcal{W}' \subseteq \mathcal{W}$ such that:

- A. $(\mathcal{K}', w) \models \text{AG}(\neg\varphi \wedge \neg\psi \Rightarrow p)$. So, every $w' \in R^*(w)$ satisfies φ, ψ or p ,
- B. $(\mathcal{K}', w) \models \text{AG}(p \Rightarrow \text{AG } p)$. So, for every $w' \in R^*(w)$ if $(\mathcal{K}', w') \models p$ then $R^*(w') \subseteq \mathcal{V}(p)$,
- C. $(\mathcal{K}', w) \models \text{EF}(\psi \wedge \neg p)$. So, there is a path $(w_0, w_1, \dots) \in \Pi_R(w)$ such that, for some $j \geq 0$, $(\mathcal{K}', w_j) \models \psi$ and $w_j \notin \mathcal{V}(p)$.

Let us consider the finite prefix (w_0, w_1, \dots, w_j) of the path in (C), so that $(\mathcal{K}', w_j) \models \psi$ and $w_j \notin \mathcal{V}(p)$. In order to conclude the proof, it is sufficient to show that every world in this finite path satisfies φ or ψ . *Ad absurdum*, suppose that there is $i \in [0, j]$ such that w_i does not satisfy neither φ nor ψ . Clearly, $w_j \in R^*(w_i)$ and, by (A), $w_i \in \mathcal{V}(p)$. However, this is contradictory, as by (B) it implies $w_j \in \mathcal{V}(p)$. So, every world in the path (w_0, w_1, \dots, w_j) satisfies either φ or ψ . This implies that $(\mathcal{K}', w) \models \text{E}(\varphi \cup \psi)$. As p does not occur in φ nor it does occur in ψ , by semantics of $\exists p$ we conclude that $(\mathcal{K}, w) \models \text{E}(\varphi \cup \psi)$. \square

Proof of $\text{AG}(\varphi \Rightarrow \text{AG } \psi) \equiv \chi_{\text{AGAG}}(\varphi, \psi)$. We recall the definition of $\chi_{\text{AGAG}}(\varphi, \psi)$:

$$\chi_{\text{AGAG}}(\varphi, \psi) \stackrel{\text{def}}{=} \forall p \forall q (\text{uniq}(p) \wedge \text{uniq}(q) \wedge \text{EF}(p \wedge \varphi) \wedge \text{EF}(q \wedge \neg\psi) \Rightarrow \text{E}(\neg p \mathbin{\text{M}} q)),$$

where p and q do not appear in φ or ψ .

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{AG}(\varphi \Rightarrow \text{AG } \psi)$. Let $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[p \leftarrow \mathcal{W}'][q \leftarrow \mathcal{W}''])$ be an arbitrary Kripke tree obtained from \mathcal{K} by changing the evaluation of p and q in such a way that $(\mathcal{K}', w) \models \text{uniq}(p) \wedge \text{uniq}(q) \wedge \text{EF}(p \wedge \varphi) \wedge \text{EF}(q \wedge \neg\psi)$. In order to prove the result, we show that $(\mathcal{K}', w) \models \text{E}(\neg p \mathbin{\text{M}} q)$. By definition of $\text{uniq}()$, there are two worlds $w', w'' \in R^*(w)$ such that $\mathcal{W}' \cap R^*(w) = \{w'\}$ and $\mathcal{W}'' \cap R^*(w) = \{w''\}$. From the satisfaction of $\text{EF}(p \wedge \varphi)$ we have $(\mathcal{K}', w') \models \varphi$, whereas by $\text{EF}(q \wedge \neg\psi)$ we get $(\mathcal{K}', w'') \not\models \psi$. From $(\mathcal{K}', w') \models \varphi$ together with the hypothesis $(\mathcal{K}, w) \models \text{AG}(\varphi \Rightarrow \text{AG } \psi)$, and since p and q do not appear in φ or ψ , we conclude that for every $\hat{w} \in R^*(w')$, $(\mathcal{K}', \hat{w}) \models \psi$ holds. Thus, $w'' \notin R^*(w')$. Let us consider the path in R going from w to w'' , say $(w_0 = w, \dots, w_k = w'')$. As w' is not in this path, for every $j \in [0, k]$, $w_j \notin \mathcal{V}[p \leftarrow \mathcal{W}'][q \leftarrow \mathcal{W}'](p)$. Moreover, $w_k = w'' \in \mathcal{V}[p \leftarrow \mathcal{W}'][q \leftarrow \mathcal{W}'](q)$. We conclude that $(\mathcal{K}', w) \models \text{E}(\neg p \mathbin{\text{M}} q)$.

(\Leftarrow): We prove the contrapositive: $(\mathcal{K}, w) \not\models \text{AG}(\varphi \Rightarrow \text{AG } \psi)$ implies $(\mathcal{K}, w) \not\models \chi_{\text{AGAG}}(\varphi, \psi)$. Since $\neg \text{AG}(\varphi \Rightarrow \text{AG } \psi) \equiv \text{EF}(\varphi \wedge \text{EF} \neg\psi)$, $(\mathcal{K}, w) \not\models \text{AG}(\varphi \Rightarrow \text{AG } \psi)$ implies that there is a world $w' \in R^*(w)$ such that $(\mathcal{K}, w') \models \varphi$ and for some $w'' \in R^*(w')$ it holds that $(\mathcal{K}, w'') \models \neg\psi$. Then, let us consider the Kripke tree $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[p \leftarrow \{w'\}][q \leftarrow \{w''\}])$ obtained from \mathcal{K} by changing the evaluation of p and q from $\mathcal{V}(p)$ and $\mathcal{V}(q)$ to $\{w'\}$ and $\{w''\}$, respectively. A simple check reveals that $(\mathcal{K}', w) \models \text{uniq}(p) \wedge \text{uniq}(q) \wedge \text{EF}(p \wedge \varphi) \wedge \text{EF}(q \wedge \neg\psi)$. Since $w' \in R^*(w)$ and $w'' \in R^*(w')$, there must exist a path $(w_0, w_1, \dots) \in \Pi_R(w)$ such that, for some $i \geq 0$ and $j \geq i$, we have $w' = w_i$ and $w'' = w_j$. From $(\mathcal{K}', w) \models \text{uniq}(p) \wedge \text{uniq}(q)$, the prefix (w_0, \dots, w_j) is the only path starting in w that ends with a world satisfying q . We conclude that $(\mathcal{K}', w) \not\models \text{E}(\neg p \mathbin{\text{M}} q)$, and thus $(\mathcal{K}, w) \not\models \chi_{\text{AGAG}}(\varphi, \psi)$. \square

Proof of Lemma 4.43.

Lemma 4.43. Let (\mathcal{F}, t, n) be a pointed model s.t. $n \neq t$ and $t \notin \text{dom}(\mathcal{F})$. Let (\mathcal{K}, n) be an encoding of (\mathcal{F}, t, n) . Given a formula φ in ALT, $(\mathcal{F}, t, n) \models \varphi$ iff $(\mathcal{K}, n) \models \tau(\varphi)$.

Proof. Recall that we assume $\mathcal{W} = \mathcal{N}$, and so in what follows we write directly \mathcal{N} and $n, n', n'' \dots$ instead of \mathcal{W} and $w, w', w'' \dots$. Then, as in the statement, let $\mathcal{K} = (\mathcal{N}, R, \mathcal{V})$ be a Kripke-style finite function so that (\mathcal{K}, n) is the encoding of a finite forest (\mathcal{F}, n, t) where $n \neq t \notin \text{dom}(\mathcal{F})$. In particular, this means that $R = \mathcal{F} \cup \{(t, t)\}$ (see Definition 4.42).

Similarly to Lemma 4.39, the proof is by structural induction on φ .

base case: $\varphi = \text{Hit}$.

$$\begin{aligned}
 & (\mathcal{F}, t, n) \models \text{Hit}, \\
 \Leftrightarrow & \text{there is } \delta \geq 1 \text{ such that } \mathcal{F}^\delta(n) = t \text{ (by definition of } \models), \\
 \Leftrightarrow & t \in R^+(n) \text{ (as } R = \mathcal{F} \cup \{(t, t)\}), \\
 \Leftrightarrow & \text{there is a subset } R_1 \subseteq R \text{ such that} \\
 & \quad 1. \text{ for every } n' \in \mathcal{N}, \text{ if } R_1(n') \neq \emptyset \text{ then } t \in R_1^+(n'), \\
 & \quad 2. R_1(n) \neq \emptyset \text{ and } (t, t) \in R_1, \\
 & \quad (\text{again from the definition of } R. R_1 \text{ simply removes worlds that do not reach } t) \\
 \Leftrightarrow & \text{there is } R_1 \subseteq R \text{ such that } R_1(n) \neq \emptyset \text{ and for every } n' \in \mathcal{N}, \text{ if } R_1(n') \neq \emptyset \text{ then there is} \\
 & \quad n'' \in R_1(n') \text{ such that } R_1(n'') \neq \emptyset, \\
 & \quad (\text{as } R \text{ is finite and its only cycle is the self-loop on } t) \\
 \Leftrightarrow & \text{there is } R_1 \subseteq R \text{ such that } ((\mathcal{N}, R_1, \mathcal{V}), n) \models \Diamond \top \wedge [U](\Diamond \top \Rightarrow \Diamond \Diamond \top) \text{ (by def. of } \models), \\
 \Leftrightarrow & (\mathcal{K}, n) \models \blacklozenge^*(\Diamond \top \wedge [U](\Diamond \top \Rightarrow \Diamond \Diamond \top)) \text{ (by definition of } \blacklozenge^* \text{ and } \models).
 \end{aligned}$$

base case: $\varphi = \text{Miss}$.

$$\begin{aligned}
 & (\mathcal{F}, t, n) \models \text{Miss}, \\
 \Leftrightarrow & n \in \text{dom}(\mathcal{F}) \text{ and } (\mathcal{F}, t, n) \not\models \text{Hit} \text{ (by definition of } \models), \\
 \Leftrightarrow & R(n) \neq \emptyset \text{ and } (\mathcal{K}, n) \not\models \tau(\text{Hit}) \text{ (by def. of the encoding and the previous base case),} \\
 \Leftrightarrow & (\mathcal{K}, n) \models \Diamond \top \wedge \neg \tau(\text{Hit}) \text{ (by definition of } \models).
 \end{aligned}$$

We omit the obvious cases for \top and Boolean connectives.

induction step: $\varphi = \langle U \rangle \psi$. By relying on Lemma 4.18(II), $t \notin \text{dom}(\mathcal{F})$ and $R = \mathcal{F} \cup \{(t, t)\}$, we have the following set of equivalences:

$$\begin{aligned}
 & (\mathcal{F}, t, n) \models \langle U \rangle \psi, \\
 \Leftrightarrow & \text{there is } n' \in \mathcal{N} \text{ such that } (\mathcal{F}, t, n') \models \psi \text{ (by def. of } \models), \\
 \Leftrightarrow & \text{there is } n' \in \mathcal{N} \text{ such that } (\mathcal{F}, t, n') \models \psi \text{ and } n' \neq t, \\
 & \quad (\text{by Lemma 4.18(II) and } t \notin \text{dom}(\mathcal{F})) \\
 \Leftrightarrow & \text{there is } n' \in \mathcal{N} \text{ such that } (\mathcal{K}, n') \models \tau(\psi) \text{ and } n' \neq t, \\
 & \quad (\text{by induction hypothesis, as obviously } (\mathcal{K}, n') \text{ encodes } (\mathcal{F}, t, n')) \\
 \Leftrightarrow & \text{there is } n' \in \mathcal{N} \text{ such that } (\mathcal{K}, n') \models \tau(\psi) \text{ and } (n', n') \notin R, \\
 & \quad (\text{by } R = \mathcal{F} \cup \{(t, t)\}, \text{ we have } (n', n') \in R \text{ if and only if } n' = t) \\
 \Leftrightarrow & \text{there is } n' \in \mathcal{N} \text{ such that } (\mathcal{K}, n') \models \neg \text{selfloop} \wedge \tau(\psi) \text{ (by def. of } \models), \\
 \Leftrightarrow & (\mathcal{K}, n) \models \langle U \rangle (\neg \text{selfloop} \wedge \tau(\psi)) \text{ (from the semantics of } \langle U \rangle).
 \end{aligned}$$

induction case: $\varphi = \blacklozenge\psi$. (\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \blacklozenge\psi$, and so there is a subforest $\mathcal{F}' \subseteq \mathcal{F}$ such that $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$ and $(\mathcal{F}', t, n) \models \psi$. Let n' be the node removed from $\text{dom}(\mathcal{F})$ in order to obtain \mathcal{F}' , i.e. $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus \{n'\}$. We consider two Kripke-style finite functions $\mathcal{K}_1 = (\mathcal{N}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{N}, R_2, \mathcal{V})$ such that

$$\text{A. } R_1 \stackrel{\text{def}}{=} \{(n', \mathcal{F}(n'))\}, \quad \text{B. } R_2 = \mathcal{F}' \cup \{(t, t)\}.$$

From (A), we conclude that $(\mathcal{K}_1, n) \models \text{size} = 1$. From (B), and by definition of encoding it holds that (\mathcal{K}_2, n) is an encoding of (\mathcal{F}', t, n) . As $(t, t) \in R_2$, $(\mathcal{K}_2, n) \models \langle U \rangle \text{selfloop}$. By induction hypothesis, $(\mathcal{K}_2, n) \models \tau(\psi)$. From $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus \{n'\}$, $R_1 \cap R_2 = \emptyset$. From $R = \mathcal{F} \cup \{(t, t)\}$, (A) and (B), we conclude that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$. Thus, (\mathcal{K}, n) satisfies $\text{size} = 1 * (\tau(\psi) \wedge \langle U \rangle \text{selfloop})$, i.e. $\tau(\blacklozenge\psi)$.

(\Leftarrow): Suppose that $(\mathcal{K}, n) \models \text{size} = 1 * (\tau(\psi) \wedge \langle U \rangle \text{selfloop})$, and so there are two finite functions $\mathcal{K}_1 = (\mathcal{N}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{N}, R_2, \mathcal{V})$ s.t. $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$, $\text{card}(R_1) = 1$, $(\mathcal{K}_2, n) \models \tau(\psi)$ and $(\mathcal{K}_2, n) \models \langle U \rangle \text{selfloop}$. As $R = \mathcal{F} \cup \{(t, t)\}$ and \mathcal{F} is acyclic, we have $(t, t) \in R_2$ and $R_1 \subseteq \mathcal{F}$. Consider the forest $\mathcal{F}' \stackrel{\text{def}}{=} \mathcal{F} \setminus R_1$. We have $R_2 = \mathcal{F}' \cup \{(t, t)\}$, and so (\mathcal{K}_2, n) encodes (\mathcal{F}', t, n) . $(\mathcal{F}', t, n) \models \psi$ follows by induction hypothesis. By definition of $\mathcal{F}' \subseteq \mathcal{F}$ we have $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$, and so $(\mathcal{F}, t, n) \models \blacklozenge\psi$.

induction case: $\varphi = \blacklozenge^* \psi$. This case is very similar to the previous one, the only difference being that \mathcal{F}' is not constrained to be such that $\text{card}(\mathcal{F}') = \text{card}(\mathcal{F}) - 1$.

(\Rightarrow): Suppose $(\mathcal{F}, t, n) \models \blacklozenge^* \psi$, and so there is $\mathcal{F}' \subseteq \mathcal{F}$ such that $(\mathcal{F}', t, n) \models \psi$. Let S be the set of nodes removed from $\text{dom}(\mathcal{F})$ in order to obtain \mathcal{F}' , i.e. $\text{dom}(\mathcal{F}') = \text{dom}(\mathcal{F}) \setminus S$. We consider two Kripke structures $\mathcal{K}_1 = (\mathcal{N}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{N}, R_2, \mathcal{V})$ such that

$$\text{A. } R_1 = \{(n', \mathcal{F}(n')) \mid n' \in S\}, \quad \text{B. } R_2 = \mathcal{F}' \cup \{(t, t)\}.$$

Clearly, $R_1 \cap R_2 = \emptyset$ and $R_1 \cup R_2 = R$ (since $R = \mathcal{F} \cup \{(t, t)\}$). So, $\mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2$. From $(t, t) \in R_2$, we have $(\mathcal{K}_2, n) \models \langle U \rangle \text{selfloop}$. From (B) we conclude that (\mathcal{K}_2, n) is an encoding of (\mathcal{F}', t, n) , and thus $(\mathcal{K}_2, n) \models \tau(\psi)$ by induction hypothesis. Therefore, (\mathcal{K}, n) satisfies $\top * (\tau(\psi) \wedge \langle U \rangle \text{selfloop})$, i.e. $\tau(\blacklozenge^* \psi)$.

(\Leftarrow): Suppose that $(\mathcal{K}, n) \models \top * (\tau(\psi) \wedge \langle U \rangle \text{selfloop})$, and so there are two finite functions $\mathcal{K}_1 = (\mathcal{N}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{N}, R_2, \mathcal{V})$ s.t. $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$, $(\mathcal{K}_2, n) \models \tau(\psi)$ and $(\mathcal{K}_2, n) \models \langle U \rangle \text{selfloop}$. As $R = \mathcal{F} \cup \{(t, t)\}$ and \mathcal{F} is acyclic, we have $(t, t) \in R_2$, which in turn implies that $R_1 \subseteq \mathcal{F}$. Then, let us consider the finite forest $\mathcal{F}' \stackrel{\text{def}}{=} \mathcal{F} \setminus R_2$. Clearly, $R_1 = \mathcal{F}' \cup \{(t, t)\}$, and so (\mathcal{K}_2, n) encodes (\mathcal{F}', t, n) . By induction hypothesis, $(\mathcal{F}', t, n) \models \psi$ and from $\mathcal{F}' \subseteq \mathcal{F}$ we conclude that $(\mathcal{F}, t, n) \models \blacklozenge\psi$. \square

Proof of Lemma 4.44.

As a preliminary result, we show that the formula `hascycles`, recalled below, is correct.

$$\text{hascycles} \stackrel{\text{def}}{=} \blacklozenge_{\text{ML}}^* (\langle U \rangle \Diamond \top \wedge [U](\Diamond \top \Rightarrow \Diamond \Diamond \top)).$$

Lemma B.1. Let (\mathcal{K}, w) be a pointed finite function, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $w \in \mathcal{W}$. $(\mathcal{K}, w) \models \text{hascycles}$ if and only if there is a world $w' \in \mathcal{W}$ and $\delta \geq 1$ such that $(w', w') \in R^\delta$.

Proof. (\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{hascycles}$ and therefore there is a finite function $\mathcal{K}' \subseteq \mathcal{K}$, where $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$, such that (\mathcal{K}', w) satisfies $\langle U \rangle \Diamond \top \wedge [U](\Diamond \top \Rightarrow \Diamond \Diamond \top)$. *Ad absurdum*, suppose that for every $w' \in \mathcal{W}$ and $\delta \geq 1$, $(w', w') \notin R'^\delta$. In other words, R' is acyclic. From $(\mathcal{K}', w) \models \langle U \rangle \Diamond \top$, $R' \neq \emptyset$. As R' is finite, this means that there is a world w' that is a root

of a non-empty tree, i.e. $R'(w) = \emptyset$ and there is a world w'' such that $w'' \in R(w)$. When considering $(w'', w') \in R'$ such that $R'(w) = \emptyset$, we find that $(\mathcal{K}', w'') \models \Diamond \top \wedge \neg \Diamond \Diamond \top$. However, this contradicts the fact that $(\mathcal{K}', w) \models [U](\Diamond \top \Rightarrow \Diamond \Diamond \top)$. Thus, there is a world $w' \in \mathcal{W}$ and $\delta \geq 1$ such that $(w', w') \in R^\delta$.

(\Leftarrow): Suppose there is a world $w' \in \mathcal{W}$ and $\delta \geq 1$ such that $(w', w') \in R^\delta$. Let us consider the finite function $\mathcal{K}' \subseteq \mathcal{K}$, where $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$, such that $R' \stackrel{\text{def}}{=} \{(w_1, w_2) \in R \mid w' \in R^*(w_2)\}$. Informally, R' contains all the pairs $(w_1, w_2) \in R$ such that there is a path going from w_2 to w' . Since $(w', w') \in R^\delta$ and $R' \subseteq R$, this implies that $(w', w') \in R'^\delta$, and so $R'(w') \neq \emptyset$. Thus, $(\mathcal{K}', w) \models \langle U \rangle \Diamond \top$. Let us show that $(\mathcal{K}', w) \models \langle U \rangle (\Diamond \top \Rightarrow \Diamond \Diamond \top)$. Suppose $w_1 \in \mathcal{W}$ such that $(\mathcal{K}', w_1) \models \Diamond \top$. So, there is w_2 such that $(w_1, w_2) \in R'$. If $w_2 \neq w'$, then from $w' \in R^*(w_2)$ there must be w_3 such that $(w_2, w_3) \in R$ and $w' \in R^*(w_3)$. By definition of R' , $(w_2, w_3) \in R'$ and thus $R^2(w_1) \neq \emptyset$. Otherwise $w_2 = w'$, and by $R(w') \neq \emptyset$ we conclude again $R^2(w_1) \neq \emptyset$. So, $(\mathcal{K}', w_1) \models \Diamond \Diamond \top$. By $\mathcal{K}' \subseteq \mathcal{K}$, we conclude: $(\mathcal{K}, w) \models \blacklozenge_{\text{ML}}^* (\langle U \rangle \Diamond \top \wedge [U](\Diamond \top \Rightarrow \Diamond \Diamond \top))$. \square

From the correctness of `hascycles` it is fairly easy to see that `existselfloop` is also correct.

$$\exists \text{selfloop} \stackrel{\text{def}}{=} \langle U \rangle (\text{selfloop} \wedge \neg \blacklozenge_{\text{ML}}(\Box \perp \wedge \text{hascycles})).$$

Lemma B.2. Let (\mathcal{K}, w) be a pointed finite function, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $w \in \mathcal{W}$. $(\mathcal{K}, w) \models \exists \text{selfloop}$ iff there is a world $w' \in \mathcal{W}$ s.t. $(w', w') \in R$ and $R \setminus \{(w', w')\}$ is acyclic.

Proof. (\Rightarrow): $(\mathcal{K}, w) \models \exists \text{selfloop}$, and therefore $(\mathcal{K}, w') \models \text{selfloop} \wedge \neg \blacklozenge_{\text{ML}}(\Box \perp \wedge \text{hascycles})$ holds for some world w' . So, $(w', w') \in R$. *Ad absurdum*, suppose that $R \setminus \{(w', w')\}$ is cyclic, and consider the Kripke-style finite function $\mathcal{K}' = (\mathcal{W}, R \setminus \{(w', w')\}, \mathcal{V})$. By Lemma B.1, we have $(\mathcal{K}', w') \models \Box \perp \wedge \text{hascycles}$. However, as $\mathcal{K}' \subseteq \mathcal{K}$ and $\text{card}(\mathcal{K}') = \text{card}(\mathcal{K}) - 1$, this contradicts the fact that $(\mathcal{K}, w') \models \neg \blacklozenge_{\text{ML}}(\Box \perp \wedge \text{hascycles})$. So, $R \setminus \{(w', w')\}$ is acyclic.

(\Leftarrow): This direction follows similar arguments as the left-to-right direction (backwards). \square

We are now ready to prove Lemma 4.44.

Lemma 4.44. Every formula φ in ALT is equisatisfiable with $\tau(\varphi) \wedge \exists \text{selfloop} \wedge \neg \text{selfloop}$.

Proof. (\Rightarrow): Suppose φ to be satisfiable, and let (\mathcal{F}, t, n) be a pointed forest satisfying it. By Lemma 4.18(I) and (II), we can assume w.l.o.g. that $t \notin \text{dom}(\mathcal{F})$ and $n \neq t$. Let (\mathcal{K}, n) being the pointed Kripke-style finite function where $\mathcal{K} = (\mathcal{N}, R, \mathcal{V})$ and $R = \mathcal{F} + \{(t, t)\}$. By Definition 4.42, (\mathcal{K}, n) is an encoding of (\mathcal{F}, t, n) . By Lemma B.2, $(\mathcal{K}, n) \models \exists \text{selfloop}$. By $n \neq t$ and the definition of R , $(\mathcal{K}, n) \models \neg \text{selfloop}$. By Lemma 4.43, $(\mathcal{K}, n) \models \tau(\varphi)$.

(\Leftarrow): Suppose that $\tau(\varphi) \wedge \exists \text{selfloop} \wedge \neg \text{selfloop}$ is satisfiable, and let us consider (\mathcal{K}, n) , where $\mathcal{K} = (\mathcal{N}, R, \mathcal{V})$, be a pointed Kripke-style finite function satisfying it. By Lemma B.2, there is a world t such that $(t, t) \in R$ and $R \setminus \{(t, t)\}$ is acyclic (and so it is a finite forest). From $(\mathcal{K}, n) \models \neg \text{selfloop}$ we have $n \neq t$. Consider the pointed forest (\mathcal{F}, n, t) where $\mathcal{F} = R \setminus \{(t, t)\}$. By Definition 4.42, (\mathcal{K}, n) is an encoding of (\mathcal{F}, t, n) . By Lemma 4.43, $(\mathcal{F}, t, n) \models \varphi$. \square

C

Appendix of Chapter 5

Contents

Proofs of Lemma 5.19(II) and Lemma 5.19(III).	513
Vade mecum on the upper bounds for $\text{Core}[s](x, \alpha)$	519
Proofs of Lemma 5.40(I), Lemma 5.40(IV) and Lemma 5.40(V).	522
Proof of Lemma 5.41.	534

Proofs of Lemma 5.19(II) and Lemma 5.19(III).

Lemma 5.19(II). Consider two memory states (s, h) , (s, h') such that $(s, h) \approx_{\mathbf{X}, \alpha}^{\mathcal{W}} (s, h')$ and $h \setminus \{(\ell, \ell') \in h \mid \ell \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}}\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}}\}$. Then, $(s, h) \leftrightarrow_{\mathbf{X}, \alpha}^{\mathcal{W}} (s, h')$.

Proof. The proof follows very closely the one given for Lemma 5.19(I). Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Notice that this requires α to be at least two, otherwise the lemma trivially holds. We partition the set $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}}$ into two sets S_1 and S_2 , using the following case analysis:

```

if card( $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_1)$ ) <  $\alpha_1$  then
    let  $S_1$  be a set of card( $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_1)$ ) locations in  $\mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}}$ 
        such that  $s(u) \in S_1$  if and only if  $s(u) \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_1)$ .
     $S_2 \leftarrow \mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}} \setminus S_1$ .
else if card( $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_2)$ ) <  $\alpha_2$  then
    let  $S_2$  be a set of card( $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_2)$ ) locations in  $\mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}}$ 
        such that  $s(u) \in S_2$  if and only if  $s(u) \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_2)$ .
     $S_1 \leftarrow \mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}} \setminus S_2$ .
else (i.e. card( $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_1)$ )  $\geq \alpha_1$  and card( $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_2)$ )  $\geq \alpha_2$ )
    let  $S_1$  be a set of  $\alpha_1$  locations in  $\mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}}$ 
        such that  $s(u) \in S_1$  if and only if  $s(u) \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_1)$ .
     $S_2 \leftarrow \mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}} \setminus S_1$ .

```

Notice that S_1 and S_2 are always well-defined, since both (s, h) and (s, h') satisfy the same formulae among $u \in \mathbf{self}_{\mathbf{X}}^{\mathcal{W}}$ and $\mathbf{self}_{\mathbf{X}}^{\mathcal{W}} \geq \beta$, for every $\beta \in [1, \alpha]$. Indeed, thanks to the formula $u \in \mathbf{self}_{\mathbf{X}}^{\mathcal{W}}$, if $s(u) \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_j)$ (where $j \in \{1, 2\}$) then $s(u) \in \mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}}$ and so $s(u)$ can be selected when building S_j . From the formulae of the form $\mathbf{self}_{\mathbf{X}}^{\mathcal{W}} \geq \beta$, if $\text{card}(\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_j)) < \alpha_j$ then, as $\alpha_j < \alpha$ we conclude that $\mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}}$ contains at least $\text{card}(\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_j))$ locations, allowing us to correctly define S_j in the first two cases above. If instead $\text{card}(\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_1)) \geq \alpha_1$ and $\text{card}(\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_2)) \geq \alpha_2$, then we conclude that both $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}}$ and $\mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}}$ contains at least $\alpha > \alpha_1$ locations. Again, this allows us to correctly define S_1 in the last case above. S_1 and S_2 enjoy the following properties, given with respect to $j \in \{1, 2\}$.

1. $s(u) \in S_j$ if and only if $s(u) \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_j)$,
2. $\min(\alpha_j, \text{card}(S_j)) = \min(\alpha_j, \text{card}(\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}} \cap \text{dom}(h_j)))$.

Proof of (1). From the equisatisfaction of $u \in \mathbf{rem}_{\mathbf{X}}^{\mathcal{W}}$, $s(u) \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}}$ iff $s(u) \in \mathbf{Self}[\mathcal{W}]_{s,h'}^{\mathbf{X}}$. Then, the property follows from the definition of S_1 and S_2 .

Proof of (2). Proof analogous to (2) in Lemma 5.19(I).

We rely on S_1 and S_2 in order to define the heaps h'_1 and h'_2 such that $(s, h_1) \approx_{\mathbf{X}, \alpha_1}^{\mathcal{W}} (s, h'_1)$ and $(s, h_2) \approx_{\mathbf{X}, \alpha_2}^{\mathcal{W}} (s, h'_2)$ (as required by the hop relation $\leftrightarrow_{\mathbf{X}, \alpha}^{\mathcal{W}}$). As done in Lemma 5.19(I), let us define $\widehat{h}_1 \stackrel{\text{def}}{=} h_1 \setminus \{(\ell, \ell') \in h_1 \mid \ell \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}}\}$ and $\widehat{h}_2 \stackrel{\text{def}}{=} h_2 \setminus \{(\ell, \ell') \in h_2 \mid \ell \in \mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}}\}$, i.e. the two heaps obtained from h_1 and h_2 by removing the locations in $\mathbf{Self}[\mathcal{W}]_{s,h}^{\mathbf{X}}$ from their domain. From $h = h_1 + h_2$ we conclude that:

$$h = \widehat{h_1} + \widehat{h_2} + \{(\ell, \ell') \in h \mid \ell \in \mathbf{Self}[w]_{s,h}^X\}.$$

Thus, from the hypothesis $h \setminus \{(\ell, \ell') \in h \mid \ell \in \mathbf{Self}[w]_{s,h}^X\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \mathbf{Self}[w]_{s,h'}^X\}$ we derive $\widehat{h_1} + \widehat{h_2} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \mathbf{Self}[w]_{s,h'}^X\}$. We define the heaps h'_1 and h'_2 as:

$$h'_1 \stackrel{\text{def}}{=} \widehat{h_1} + \{(\ell, \ell') \in h' \mid \ell \in S_1\}, \quad h'_2 \stackrel{\text{def}}{=} \widehat{h_2} + \{(\ell, \ell') \in h' \mid \ell \in S_2\}.$$

As $\{(\ell, \ell') \in h' \mid \ell \in S_1\} + \{(\ell, \ell') \in h' \mid \ell \in S_2\} = \{(\ell, \ell') \in h' \mid \ell \in \mathbf{Self}[w]_{s,h'}^X\}$ by definition of S_1 and S_2 , the two heaps h'_1 and h'_2 are well-defined, they are disjoint, and $h' = h'_1 + h'_2$. Moreover, $S_1 = \mathbf{Self}[w]_{s,h'}^X \cap \text{dom}(h'_1)$ and $S_2 = \mathbf{Self}[w]_{s,h'}^X \cap \text{dom}(h'_2)$. The heaps h'_1 and h'_2 enjoy the following four properties. Let $j \in \{1, 2\}$.

- A. (a) for every $t \in T[w]^X$, $\llbracket t \rrbracket_{s,h_j}^X$ is defined iff so is $\llbracket t \rrbracket_{s,h'_j}^X$. If defined, $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h'_j}^X$.
- (b) $\llbracket t \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$ if and only if $\llbracket t \rrbracket_{s,h'_j}^X \in \text{dom}(h'_j)$.
- (c) Given $t' \in X \cup \{t\}$, we have $h_j(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t' \rrbracket_{s,h_j}^X$ if and only if $h'_j(\llbracket t \rrbracket_{s,h'_j}^X) = \llbracket t' \rrbracket_{s,h'_j}^X$.
- (d) $\ell \in \mathbf{Lab}[w]_{s,h}^X \setminus \mathbf{Lab}[w]_{s,h_j}^X$ if and only if $\ell \in \mathbf{Lab}[w]_{s,h'}^X \setminus \mathbf{Lab}[w]_{s,h'_j}^X$.

These four statements are the same as the ones in (A), proof of Lemma 5.19(I). As in that proof we have only shown the left-to-right direction, here we do the opposite and show the right-to-left direction. Again, the left-to-right direction is analogous.

Proof of (a). Obvious for $t \in X$, so suppose $t = n(x) \in NV[w]^X$.

(\Leftarrow): Suppose $\llbracket n(x) \rrbracket_{s,h'_j}^X$ defined, and so $\llbracket n(x) \rrbracket_{s,h'_j}^X \stackrel{\text{by def}}{=} h'_j(s(x))$. As $s(x) \in \mathbf{Lab}[w]_{s,h'}^X$, $s(x) \notin \mathbf{Self}[w]_{s,h'}^X$, and therefore $s(x) \in \text{dom}(\widehat{h_j})$ and $\widehat{h_j}(s(x)) = h'_j(s(x))$. By definition of $\widehat{h_j}$, $\widehat{h_j}(s(x)) = h_j(s(x))$. Thus, $\llbracket n(x) \rrbracket_{s,h_j}^X = \llbracket n(x) \rrbracket_{s,h'_j}^X$.

Proof of (b). (\Leftarrow): Suppose $\llbracket t \rrbracket_{s,h'_j}^X \in \text{dom}(h'_j)$. By Lemma 5.14(I), $\llbracket t \rrbracket_{s,h'_j}^X = \llbracket t \rrbracket_{s,h'}^X$ and thus $\llbracket t \rrbracket_{s,h'_j}^X \notin \mathbf{Self}[w]_{s,h'}^X$. Therefore, $\llbracket t \rrbracket_{s,h'_j}^X \in \text{dom}(\widehat{h_j})$. From (a), $\llbracket t \rrbracket_{s,h'_j}^X = \llbracket t \rrbracket_{s,h_j}^X$, which in turn implies $\llbracket t \rrbracket_{s,h_j}^X \in \text{dom}(\widehat{h_j}) \subseteq \text{dom}(h_j)$.

Proof of (c). (\Leftarrow): $h'_j(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t' \rrbracket_{s,h'_j}^X$ and therefore by (a) we have $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h'_j}^X$ and $\llbracket t' \rrbracket_{s,h_j}^X = \llbracket t' \rrbracket_{s,h'_j}^X$. As done in the proof of (b), we conclude that $\llbracket t \rrbracket_{s,h'_j}^X \in \text{dom}(\widehat{h_j})$. By $\widehat{h_j} \subseteq h'_j$, $\widehat{h_j}(\llbracket t \rrbracket_{s,h'_j}^X) = \llbracket t' \rrbracket_{s,h'_j}^X$. By $\widehat{h_j} \subseteq h_j$, $h_j(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t' \rrbracket_{s,h_j}^X$.

Proof of (d). (\Leftarrow): Assume $\ell \in \mathbf{Lab}[w]_{s,h'}^X \setminus \mathbf{Lab}[w]_{s,h'_j}^X$. By definition, it cannot be that ℓ is assigned to a program variable in X , as otherwise $\ell \in \mathbf{Lab}[w]_{s,h'_j}^X$. So, there is a next-point variable $n(x)$ such that $\llbracket n(x) \rrbracket_{s,h'}^X = \ell$. From $s(x) \in \mathbf{Lab}[w]_{s,h'}^X$, we derive that $s(x) \notin \mathbf{Self}[w]_{s,h'}^X$ and therefore $s(x) \in \text{dom}(h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \mathbf{Self}[w]_{s,h'}^X\})$. From $h \setminus \{(\ell, \ell') \in h \mid \ell \in \mathbf{Self}[w]_{s,h}^X\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \mathbf{Self}[w]_{s,h'}^X\}$, we derive that $h(s(x)) = \ell$ and so $\ell \in \mathbf{Lab}[w]_{s,h}^X$. *Ad absurdum*, suppose $\ell \in \mathbf{Lab}[w]_{s,h'_j}^X$. From (a) we have $\ell \in \mathbf{Lab}[w]_{s,h'_j}^X$, contradicting $\ell \in \mathbf{Lab}[w]_{s,h'}^X \setminus \mathbf{Lab}[w]_{s,h'_j}^X$. Thus, $\ell \notin \mathbf{Lab}[w]_{s,h'_j}^X$ and therefore $\ell \in \mathbf{Lab}[w]_{s,h}^X \setminus \mathbf{Lab}[w]_{s,h'_j}^X$.

- B. For every $x \in X$, $\mathbf{Pred}[w]_{s,h'_j}^X(x) = \mathbf{Pred}[w]_{s,h_j}^X(x)$.

We show the right-to-left direction of this statement. The left-to-right direction is analogous (see also (B) in the proof of Lemma 5.19(I)).

Proof. (\Leftarrow): Suppose $\ell \in \mathbf{Pred}[w]_{s,h_j}^X(x)$. By definition, $\ell \notin \mathbf{Lab}[w]_{s,h_j}^X$ and $h_j(\ell) = s(x)$. From (a), $\ell \notin \mathbf{Lab}[w]_{s,h'_j}^X$. From $h_j \subseteq h$, $h(\ell) = s(x)$ and therefore it cannot be that ℓ

belongs to $\mathbf{Self}[\mathcal{W}]_{s,h}^X$. Indeed if $h(\ell) = \ell$ then $\ell = s(x)$ and we derive a contradiction with $\ell \notin \mathbf{Lab}[\mathcal{W}]_{s,h_j}^X$. By definition of \widehat{h}_j , $\ell \in \text{dom}(\widehat{h}_j)$. From $\widehat{h}_j \subseteq h_j$, $\widehat{h}_j(\ell) = s(x)$. From $\widehat{h}_j \subseteq h'_j$, $h'_j(\ell) = s(x)$. As $\ell \notin \mathbf{Lab}[\mathcal{W}]_{s,h'_j}^X$, this implies $\ell \in \mathbf{Pred}[\mathcal{W}]_{s,h'_j}^X(x)$,

$$\text{C. } \mathbf{Rem}[\mathcal{W}]_{s,h'_j}^X = \mathbf{Rem}[\mathcal{W}]_{s,h_j}^X.$$

We show right-to-left direction. Thanks to \widehat{h}_j , the left-to-right direction is analogous.

Proof. (\Leftarrow): Suppose $\ell \in \mathbf{Rem}[\mathcal{W}]_{s,h_j}^X$. By definition we have $\ell \notin \mathbf{Lab}[\mathcal{W}]_{s,h_j}^X$, $\ell \in \text{dom}(h_j)$, $\ell \notin \mathbf{Self}[\mathcal{W}]_{s,h_j}^X$ and $\ell \notin \mathbf{Pred}[\mathcal{W}]_{s,h_j}^X(x)$, for any $x \in X$. In particular, this means that $h_j(\ell) \neq \ell$ and for every $x \in X$, $h_j(\ell) \neq s(x)$. From $\ell \notin \mathbf{Lab}[\mathcal{W}]_{s,h_j}^X$ and (a), we derive $\ell \notin \mathbf{Lab}[\mathcal{W}]_{s,h'_j}^X$. From $\ell \in \text{dom}(h'_j)$ and $\ell \notin \mathbf{Self}[\mathcal{W}]_{s,h'_j}^X$ we derive $\ell \in \text{dom}(\widehat{h}_j)$. From $\widehat{h}_j \subseteq h_j$ we derive $\widehat{h}_j(\ell) \neq \ell$ and for every $x \in X$, $\widehat{h}_j(\ell) \neq s(x)$. By $\widehat{h}_j \subseteq h'_j$, this means that $\ell \in \text{dom}(h'_j)$, $h'_j(\ell) \neq \ell$ and for every $x \in X$, $h'_j(\ell) \neq s(x)$. So, $\ell \notin \mathbf{Self}[\mathcal{W}]_{s,h'_j}^X$ and $\ell \notin \mathbf{Pred}[\mathcal{W}]_{s,h'_j}^X(x)$, for any $x \in X$. As moreover $\ell \in \text{dom}(h'_j)$ and $\ell \notin \mathbf{Lab}[\mathcal{W}]_{s,h'_j}^X$, we conclude that $\ell \in \mathbf{Rem}[\mathcal{W}]_{s,h'_j}^X$.

$$\text{D. } \min(\alpha_j, \text{card}(\mathbf{Self}[\mathcal{W}]_{s,h'_j}^X)) = \min(\alpha_j, \text{card}(\mathbf{Self}[\mathcal{W}]_{s,h_j}^X)).$$

Proof. From Lemma 5.14(III) we have:

$$\mathbf{Self}[\mathcal{W}]_{s,h'_j}^X = (\mathbf{Self}[\mathcal{W}]_{s,h'}^X \cap \text{dom}(h'_j)) \cup \{\ell \in \mathbf{Lab}[\mathcal{W}]_{s,h'}^X \setminus \mathbf{Lab}[\mathcal{W}]_{s,h'_j}^X \mid h'_j(\ell) = \ell\},$$

$$\mathbf{Self}[\mathcal{W}]_{s,h_j}^X = (\mathbf{Self}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j)) \cup \{\ell \in \mathbf{Lab}[\mathcal{W}]_{s,h}^X \setminus \mathbf{Lab}[\mathcal{W}]_{s,h_j}^X \mid h_j(\ell) = \ell\}.$$

By definition of h'_j , $\mathbf{Self}[\mathcal{W}]_{s,h'}^X \cap \text{dom}(h'_j) = S_j$. By (2),

$$\min(\alpha_j, \text{card}(\mathbf{Self}[\mathcal{W}]_{s,h'}^X \cap \text{dom}(h'_j))) = \min(\alpha_j, \text{card}(\mathbf{Self}[\mathcal{W}]_{s,h}^X \cap \text{dom}(h_j))).$$

Thus, in order to prove (D) we only need to show that the two sets

$$\{\ell \in \mathbf{Lab}[\mathcal{W}]_{s,h'}^X \setminus \mathbf{Lab}[\mathcal{W}]_{s,h'_j}^X \mid h'_j(\ell) = \ell\} \text{ and } \{\ell \in \mathbf{Lab}[\mathcal{W}]_{s,h}^X \setminus \mathbf{Lab}[\mathcal{W}]_{s,h_j}^X \mid h_j(\ell) = \ell\}$$

are equal. This amounts to showing that, given a location $\ell \in \text{LOC}$,

$$\text{e. } h'_j(\ell) = \ell \in \mathbf{Lab}[\mathcal{W}]_{s,h'}^X \setminus \mathbf{Lab}[\mathcal{W}]_{s,h'_j}^X \text{ iff } h_j(\ell) = \ell \in \mathbf{Lab}[\mathcal{W}]_{s,h}^X \setminus \mathbf{Lab}[\mathcal{W}]_{s,h_j}^X.$$

We show the right-to-left direction of this statement. The left-to-right direction is analogous (see also (D)(e) in the proof of Lemma 5.19(I)).

Proof of (e). (\Leftarrow): Suppose $\ell \in \mathbf{Lab}[\mathcal{W}]_{s,h}^X \setminus \mathbf{Lab}[\mathcal{W}]_{s,h_j}^X$ such that $h_j(\ell) = \ell$. From (A)(d), we derive $\ell \in \mathbf{Lab}[\mathcal{W}]_{s,h'}^X \setminus \mathbf{Lab}[\mathcal{W}]_{s,h'_j}^X$. As $\ell \in \mathbf{Lab}[\mathcal{W}]_{s,h}^X$, $\ell \notin \mathbf{Self}[\mathcal{W}]_{s,h}^X$ and so $\ell \in \text{dom}(\widehat{h}_j)$.

From $\widehat{h}_j \subseteq h_j$ we derive $\widehat{h}_j(\ell) = \ell$. From $\widehat{h}_j \subseteq h'_j$ we derive $h'_j(\ell) = \ell$.

Thanks to the properties (A)–(D), proving $(s, h_j) \approx_{X, \alpha_j}^{\mathcal{W}} (s, h'_j)$, for $j \in \{1, 2\}$, is straightforward. Consider a core formula φ in $\mathbf{Core}[\mathcal{W}](X, \alpha_j)$. Then, $(s, h_j) \models \varphi$ iff $(s, h'_j) \models \varphi$, as shown below:

case: $\varphi = t_1 = t_2$. Follows directly from (A)(a).

case: $\varphi = t \hookrightarrow __$. Follows directly from (A)(b).

case: $\varphi = t \hookrightarrow x$ or $\varphi = t \hookrightarrow t$. Follows directly from (A)(c).

case: $\varphi = \text{pred}_X^{\mathcal{W}}(x) \geq \beta$. Follows directly from (B).

case: $\varphi = \text{self}_X^{\mathcal{W}} \geq \beta$. Follows directly from (D).

case: $\varphi = \text{rem}_X^{\mathcal{W}} \geq \beta$. Follows directly from (C).

case: $\varphi = u = t$. Follows directly from (A)(a), since (s, h_j) and (s, h_j) share the same store.

case: $\varphi = u \in \text{pred}_X^W(x)$. Follows directly from (B).

case: $\varphi = u \in \text{self}_X^W$. Since $S_j = \text{Self}[w]_{s,h'}^X \cap \text{dom}(h'_j)$, it follows from (1) and (D)(e).

case: $\varphi = u \in \text{rem}_X^W$. Follows directly from (C).

Therefore, $(s, h) \leftrightarrow_{X,\alpha}^W (s, h')$. □

Lemma 5.19(III). Consider two memory states $(s, h), (s, h')$ such that $(s, h) \approx_{X,\alpha}^W (s, h')$ and $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[w]_{s,h}^X(x)\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[w]_{s,h'}^X(x)\}$, for some $x \in X$. Then, $(s, h) \leftrightarrow_{X,\alpha}^W (s, h')$.

Proof. The proof follows very closely the ones given for Lemma 5.19(I) and (II)). Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Notice that this requires α to be at least two, otherwise the lemma trivially holds. We partition the set $\text{Pred}[w]_{s,h'}^X(x)$ into two sets S_1 and S_2 , using the following case analysis:

```

if  $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_1)) < \alpha_1$  then
  let  $S_1$  be a set of  $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_1))$  locations in  $\text{Pred}[w]_{s,h'}^X(x)$ 
    such that  $s(u) \in S_1$  if and only if  $s(u) \in \text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_1)$ .
   $S_2 \leftarrow \text{Pred}[w]_{s,h'}^X(x) \setminus S_1$ .
else if  $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_2)) < \alpha_2$  then
  let  $S_2$  be a set of  $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_2))$  locations in  $\text{Pred}[w]_{s,h'}^X(x)$ 
    such that  $s(u) \in S_2$  if and only if  $s(u) \in \text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_2)$ .
   $S_1 \leftarrow \text{Pred}[w]_{s,h'}^X(x) \setminus S_2$ .
else (i.e.  $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_1)) \geq \alpha_1$  and  $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_2)) \geq \alpha_2$ )
  let  $S_1$  be a set of  $\alpha_1$  locations in  $\text{Pred}[w]_{s,h'}^X(x)$ 
    such that  $s(u) \in S_1$  if and only if  $s(u) \in \text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_1)$ .
   $S_2 \leftarrow \text{Pred}[w]_{s,h'}^X(x) \setminus S_1$ .

```

Notice that S_1 and S_2 are always well-defined, since both (s, h) and (s, h') satisfy the same formulae among $u \in \text{pred}_X^W(x)$ and $\text{pred}_X^W(x) \geq \beta$, for every $\beta \in [1, \alpha]$. Indeed, thanks to the formula $u \in \text{pred}_X^W(x)$, if $s(u) \in \text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_j)$ (where $j \in \{1, 2\}$) then $s(u) \in \text{Pred}[w]_{s,h'}^X(x)$ and so $s(u)$ can be selected when building S_j . From the formulae of the form $\text{self}_X^W \geq \beta$, if $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_j)) < \alpha_j$ then, as $\alpha_j < \alpha$ we conclude that $\text{Pred}[w]_{s,h'}^X(x)$ contains at least $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_j))$ locations, allowing us to correctly define S_j in the first two cases. If instead $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_1)) \geq \alpha_1$ and $\text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_2)) \geq \alpha_2$, then we conclude that both $\text{Pred}[w]_{s,h}^X(x)$ and $\text{Pred}[w]_{s,h'}^X(x)$ contains at least $\alpha > \alpha_1$ locations. Again, this allows us to correctly define S_1 in the last of the cases above. S_1 and S_2 enjoy the following properties, given with respect to $j \in \{1, 2\}$.

1. $s(u) \in S_j$ if and only if $s(u) \in \text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_j)$,
2. $\min(\alpha_j, \text{card}(S_j)) = \min(\alpha_j, \text{card}(\text{Pred}[w]_{s,h}^X(x) \cap \text{dom}(h_j)))$.

Proof of (1). From the equisatisfaction of $u \in \text{pred}_X^W(x)$, we have $s(u) \in \text{Pred}[w]_{s,h}^X(x)$ if and only if $s(u) \in \text{Pred}[w]_{s,h'}^X(x)$. Then, the property follows by definition of S_1 and S_2 .

Proof of (2). Proof analogous to (2) in Lemma 5.19(I).

We rely on S_1 and S_2 in order to define the heaps h'_1 and h'_2 such that $(s, h_1) \approx_{X,\alpha_1}^W (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^W (s, h'_2)$ (as required by the hop relation $\leftrightarrow_{X,\alpha}^W$). As done in Lemma 5.19(I), let us define $\widehat{h}_1 \stackrel{\text{def}}{=} h_1 \setminus \{(\ell, \ell') \in h_1 \mid \ell \in \text{Pred}[W]_{s,h}^X(x)\}$ and $\widehat{h}_2 \stackrel{\text{def}}{=} h_2 \setminus \{(\ell, \ell') \in h_2 \mid \ell \in \text{Pred}[W]_{s,h}^X(x)\}$, i.e. the two heaps obtained from h_1 and h_2 by removing the locations in $\text{Pred}[W]_{s,h}^X(x)$ from their domain. From $h = h_1 + h_2$ we conclude that:

$$h = \widehat{h}_1 + \widehat{h}_2 + \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[W]_{s,h}^X(x)\}.$$

From the hypothesis $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[W]_{s,h}^X(x)\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[W]_{s,h'}^X(x)\}$ we derive $\widehat{h}_1 + \widehat{h}_2 = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[W]_{s,h'}^X(x)\}$. We define the heaps h'_1 and h'_2 as:

$$h'_1 \stackrel{\text{def}}{=} \widehat{h}_1 + \{(\ell, \ell') \in h' \mid \ell \in S_1\}, \quad h'_2 \stackrel{\text{def}}{=} \widehat{h}_2 + \{(\ell, \ell') \in h' \mid \ell \in S_2\}.$$

As $\{(\ell, \ell') \in h' \mid \ell \in S_1\} + \{(\ell, \ell') \in h' \mid \ell \in S_2\} = \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[W]_{s,h'}^X(x)\}$ by definition of S_1 and S_2 , the two heaps h'_1 and h'_2 are well-defined, they are disjoint, and $h' = h'_1 + h'_2$. Moreover, $S_1 = \text{Pred}[W]_{s,h'}^X(x) \cap \text{dom}(h'_1)$ and $S_2 = \text{Pred}[W]_{s,h'}^X(x) \cap \text{dom}(h'_2)$. The heaps h'_1 and h'_2 enjoy the following five properties. Let $j \in \{1, 2\}$.

- A. (a) for every $t \in T[W]^X$, $\llbracket t \rrbracket_{s,h_j}^X$ is defined iff so is $\llbracket t \rrbracket_{s,h'_j}^X$. If defined, $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h'_j}^X$.
- (b) $\llbracket t \rrbracket_{s,h_j}^X \in \text{dom}(h_j)$ if and only if $\llbracket t \rrbracket_{s,h'_j}^X \in \text{dom}(h'_j)$.
- (c) Given $t' \in X \cup \{t\}$, we have $h_j(\llbracket t \rrbracket_{s,h_j}^X) = \llbracket t' \rrbracket_{s,h_j}^X$ if and only if $h'_j(\llbracket t \rrbracket_{s,h'_j}^X) = \llbracket t' \rrbracket_{s,h'_j}^X$.
- (d) $\ell \in \text{Lab}[W]_{s,h}^X \setminus \text{Lab}[W]_{s,h_j}^X$ if and only if $\ell \in \text{Lab}[W]_{s,h'}^X \setminus \text{Lab}[W]_{s,h'_j}^X$.

These four statements are the same as in the proofs of Lemma 5.19(I) (step (A)) and Lemma 5.19(II) (step (A)). Thus, we refer to these two proofs for similar results, and show here only the proof of (a) (which is needed to prove the other three statements).

Proof of (a). Obvious for $t \in X$, so suppose $t = n(x) \in NV[W]^X$.

(\Rightarrow): Suppose $\llbracket n(x) \rrbracket_{s,h_j}^X$ defined, and so $\llbracket n(x) \rrbracket_{s,h_j}^X \stackrel{\text{by def}}{=} h_j(s(x))$. As $s(x) \in \text{Lab}[W]_{s,h}^X$ we derive $s(x) \notin \text{Pred}[W]_{s,h'}^X(x)$ and thus $s(x) \in \text{dom}(\widehat{h}_j)$. From $\widehat{h}_j \subseteq h_j$, we conclude that $\widehat{h}_j(s(x)) = h_j(s(x))$. From $\widehat{h}_j \subseteq h'_j$ we derive that $\widehat{h}_j(s(x)) = h'_j(s(x))$. Therefore, $\llbracket n(x) \rrbracket_{s,h_j}^X = \llbracket n(x) \rrbracket_{s,h'_j}^X$.

(\Leftarrow): Suppose $\llbracket n(x) \rrbracket_{s,h'_j}^X$ defined, and so $\llbracket n(x) \rrbracket_{s,h'_j}^X \stackrel{\text{by def}}{=} h'_j(s(x))$. As $s(x) \in \text{Lab}[W]_{s,h'}^X$ we derive $s(x) \notin \text{Pred}[W]_{s,h'}^X(x)$ and thus $s(x) \in \text{dom}(\widehat{h}_j)$. From $\widehat{h}_j \subseteq h'_j$, we conclude that $\widehat{h}_j(s(x)) = h'_j(s(x))$. From $\widehat{h}_j \subseteq h_j$ we derive that $\widehat{h}_j(s(x)) = h_j(s(x))$. Therefore, $\llbracket n(x) \rrbracket_{s,h_j}^X = \llbracket n(x) \rrbracket_{s,h'_j}^X$.

- B. $\text{Self}[W]_{s,h'_j}^X = \text{Self}[W]_{s,h_j}^X$.

The proof of this statement is analogous to the proof in the step (C) of Lemma 5.19(I).

- C. $\text{Rem}[W]_{s,h'_j}^X = \text{Rem}[W]_{s,h_j}^X$.

The proof of this statement is analogous to the proof in the step (D) of Lemma 5.19(II).

- D. For every $y \in X$, if $s(y) \neq s(x)$ then $\text{Pred}[W]_{s,h'_j}^X(y) = \text{Pred}[W]_{s,h_j}^X(y)$.

We show the right-to-left direction of this statement. The left-to-right direction is analogous (see also proof of Lemma 5.19(I), step (B), and Lemma 5.19(II), step (B)).

Proof of (D). (\Leftarrow): Suppose $\ell \in \text{Pred}[W]_{s,h_j}^X(y)$. By definition, $\ell \notin \text{Lab}[W]_{s,h_j}^X$ and $h_j(\ell) = s(y)$. From (a), $\ell \notin \text{Lab}[W]_{s,h'_j}^X$. From $h_j \subseteq h$, $h(\ell) = s(y)$ and since $s(y) \neq s(x)$, it

cannot be that $\ell \in \text{Pred}[\mathcal{W}]_{s,h}^X(x)$. By definition of $\widehat{h_j}$, $\ell \in \text{dom}(\widehat{h_j})$. From $\widehat{h_j} \subseteq h_j$, $\widehat{h_j}(\ell) = s(y)$. From $\widehat{h_j} \subseteq h'_j$, $h'_j(\ell) = s(y)$. As $\ell \notin \text{Lab}[\mathcal{W}]_{s,h'_j}^X$, this implies $\ell \in \text{Pred}[\mathcal{W}]_{s,h'_j}^X(y)$.

E. For every $y \in X$, if $s(y) = s(x)$ then

$$\min(\alpha_j, \text{card}(\text{Pred}[\mathcal{W}]_{s,h'_j}^X(y))) = \min(\alpha_j, \text{card}(\text{Pred}[\mathcal{W}]_{s,h_j}^X(y))).$$

Proof of (E). We show that $\min(\alpha_j, \text{card}(\text{Pred}[\mathcal{W}]_{s,h'_j}^X(x))) = \min(\alpha_j, \text{card}(\text{Pred}[\mathcal{W}]_{s,h_j}^X(x)))$.

The proof of (E) then follows as $s(y) = s(x)$ implies $\text{Pred}[\mathcal{W}]_{s,h_j}^X(x) = \text{Pred}[\mathcal{W}]_{s,h_j}^X(y)$ and $\text{Pred}[\mathcal{W}]_{s,h'_j}^X(x) = \text{Pred}[\mathcal{W}]_{s,h'_j}^X(y)$. From Lemma 5.14(II) we have:

$$\begin{aligned} \text{Pred}[\mathcal{W}]_{s,h'_j}^X(x) &= (\text{Pred}[\mathcal{W}]_{s,h'}^X(x) \cap \text{dom}(h'_j)) \\ &\cup \{\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X \mid h'_j(\ell) = s(x)\}, \end{aligned}$$

$$\begin{aligned} \text{Pred}[\mathcal{W}]_{s,h_j}^X(x) &= (\text{Pred}[\mathcal{W}]_{s,h}^X(x) \cap \text{dom}(h_j)) \\ &\cup \{\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X \mid h_j(\ell) = s(x)\}. \end{aligned}$$

By definition of h'_j , $\text{Pred}[\mathcal{W}]_{s,h'}^X(x) \cap \text{dom}(h'_j) = S_j$. By (2),

$$\min(\alpha_j, \text{card}(\text{Pred}[\mathcal{W}]_{s,h'}^X(x) \cap \text{dom}(h'_j))) = \min(\alpha_j, \text{card}(\text{Pred}[\mathcal{W}]_{s,h}^X(x) \cap \text{dom}(h_j))).$$

Thus, in order to prove (E) we only need to show that the two sets

$$\{\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X \mid h'_j(\ell) = s(x)\}$$

$$\text{and } \{\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X \mid h_j(\ell) = s(x)\}$$

are equivalent. This amounts to showing that, given a location $\ell \in \text{LOC}$,

- e. $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X$ and $h'_j(\ell) = s(x)$ if and only if
 $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$ and $h_j(\ell) = s(x)$.

We show the left-to-right direction of (e). The other direction is analogous (see also proof of Lemma 5.19(I), step (D)(e), and proof of Lemma 5.19(II), step (D)(e)).

Proof of (e). (\Rightarrow): Suppose $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X \setminus \text{Lab}[\mathcal{W}]_{s,h'_j}^X$ s.t. $h'_j(\ell) = s(x)$. From (A)(d), we have $\ell \in \text{Lab}[\mathcal{W}]_{s,h}^X \setminus \text{Lab}[\mathcal{W}]_{s,h_j}^X$. As $\ell \in \text{Lab}[\mathcal{W}]_{s,h'}^X$, $\ell \notin \text{Pred}[\mathcal{W}]_{s,h'}^X(x)$ and so $\ell \in \text{dom}(\widehat{h_j})$. By $\widehat{h_j} \subseteq h'_j$, we derive $\widehat{h_j}(\ell) = s(x)$. By $\widehat{h_j} \subseteq h_j$, we derive $h_j(\ell) = s(x)$.

Thanks to the properties (A)–(E), proving $(s, h_j) \approx_{X,\alpha_j}^{\mathcal{W}} (s, h'_j)$, for $j \in \{1, 2\}$, is straightforward. Consider a core formula φ in $\text{Core}[\mathcal{W}](X, \alpha_j)$. Then, $(s, h_j) \models \varphi$ iff $(s, h'_j) \models \varphi$, as shown below:

case: $\varphi = t_1 = t_2$. Follows directly from (A)(a).

case: $\varphi = t \hookrightarrow \dots$. Follows directly from (A)(b).

case: $\varphi = t \hookrightarrow y$ or $\varphi = t \hookrightarrow t$. Follows directly from (A)(c).

case: $\varphi = \text{pred}_X^{\mathcal{W}}(y) \geq \beta$. Follows from (D) and (E) (depending on whether $s(y) = s(x)$ holds).

case: $\varphi = \text{self}_X^{\mathcal{W}} \geq \beta$. Follows directly from (B).

case: $\varphi = \text{rem}_X^{\mathcal{W}} \geq \beta$. Follows directly from (C).

case: $\varphi = u = t$. Follows directly from (A)(a), since (s, h_j) and (s, h'_j) share the same store.

case: $\varphi = u \in \text{pred}_X^{\mathcal{W}}(y)$. If $s(y) = s(x)$, it follows from (1) (as $S_j = \text{Pred}[\mathcal{W}]_{s,h'}^X(x) \cap \text{dom}(h'_j)$) and (E)(e). Otherwise, it follows directly from (D).

case: $\varphi = u \in \text{self}_X^{\mathcal{W}}$. Follows directly from (B).

case: $\varphi = u \in \text{rem}_X^{\mathcal{W}}$. Follows directly from (C).

Therefore, $(s, h) \leftrightarrow_{X,\alpha}^{\mathcal{W}} (s, h')$. □

Vade mecum on the upper bounds for $\text{Core}[s](X, \alpha)$.

In this section we provide the inequalities and recurrence systems that have been used to compute the upper bounds for the core formulae $\text{Core}[s](X, \alpha)$ introduced in Section 5.5.2. Note that the recurrence systems considered are generally more constrained than the inequalities that we want to satisfy. This is not a problem, as one can check that the solutions of the recurrence system do indeed satisfy also the original inequalities. Also, we avoid the formal proofs regarding the correctness of these inequalities, in favor of an informal explanation. After all, the upper bounds are shown to be sufficient for the satisfaction of the $*$ -simulation property of $\text{Core}[s](X, \alpha)$, which is all we need. We refer the reader to Example 5.33 for the upper bound on β^\circlearrowleft for the core formulae of the form $\text{loop}_X^S(\beta_1) \geq \beta^\circlearrowleft$ and $\text{loop}_{X,\alpha}^S \geq \beta^\circlearrowleft$. Below, let (s, h) be a memory state, $h_1 + h_2 = h$ and $\alpha, \alpha_1, \alpha_2 \geq 1$ such that $\alpha = \alpha_1 + \alpha_2$ (as in Lemma 5.6).

Upper bound for β for the core formulae of the form $\text{rem}_{X,\alpha}^S \geq \beta$.

Let us write $\mathcal{R}(\alpha)$ for this upper bound, as done in Section 5.5.2.

Explanation: One can show that $\text{Rem}[s]_{s,h}^{X,\alpha} \subseteq \text{Rem}[s]_{s,h_1}^{X,\alpha_1} \cup \text{Rem}[s]_{s,h_2}^{X,\alpha_2}$, which leads to the inequality.

Inequality: $\mathcal{R}(\alpha) \geq \max_{\substack{\alpha_1, \alpha_2 \geq 1 \\ \alpha = \alpha_1 + \alpha_2}} (\mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2))$.

This inequality implies $\mathcal{R}(\alpha) \geq \mathcal{R}(\alpha - 1) + \mathcal{R}(1)$, which is used in the recurrence system.

Recurrence system: $\{\mathcal{R}(1) = 1, \mathcal{R}(\alpha + 1) = \mathcal{R}(\alpha) + \mathcal{R}(1)\}$.

Solution: $\mathcal{R}(\alpha) = \alpha$.

Upper bound for β for the core formulae of the form $\text{pred}_X^S(x) \geq \beta$.

Let us write $\mathcal{P}(\alpha)$ for this upper bound.

Explanation: One can show that, for all $x \in X$, $\text{Pred}[s]_{s,h}^X(x) \subseteq \text{Pred}[s]_{s,h_1}^X(x) \cup \text{Pred}[s]_{s,h_2}^X(x)$.

Inequality: $\mathcal{P}(\alpha) \geq \max_{\substack{\alpha_1, \alpha_2 \geq 1 \\ \alpha = \alpha_1 + \alpha_2}} (\mathcal{P}(\alpha_1) + \mathcal{P}(\alpha_2))$.

This inequality implies $\mathcal{P}(\alpha) \geq \mathcal{P}(\alpha - 1) + \mathcal{P}(1)$, which is used in the recurrence system.

Recurrence system: $\{\mathcal{P}(1) = 1, \mathcal{P}(\alpha + 1) = \mathcal{P}(\alpha) + \mathcal{P}(1)\}$.

Solution: $\mathcal{P}(\alpha) = \alpha$.

Upper bound for $\overrightarrow{\beta}$ in the core formulae of the form $u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$.

Let us write $\mathcal{S}_{\text{right}}(\alpha)$ for this upper bound.

Explanation: $(s, h) \models u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$ implies that h is such that it witnesses a path going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $\llbracket t_2 \rrbracket_{s,h}^X$. Moreover,

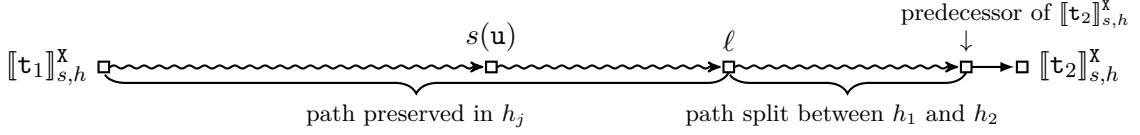
- every intermediate location of this path does not belong to $\text{Lab}[s]_{s,h}^X$,
- the path can be split into two paths, one of length at least $\overleftarrow{\beta}$ and going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $s(u)$, and one of length at least $\overrightarrow{\beta}$ going from $s(u)$ to $\llbracket t_2 \rrbracket_{s,h}^X$.

Let $j \in \{1, 2\}$ be one of the indices for the subheaps h_1 and h_2 of h , and $k \in \{1, 2\} \setminus \{j\}$ be the other index. When the heap h is split into h_1 and h_2 it can be that:

- $\llbracket t_1 \rrbracket_{s,h}^X$ is a labelled location of (s, h_j) ,

- the heap h_j witnesses a path going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $s(u)$.

For simplicity, let us assume that both these properties hold. In this case, the path in h going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $\llbracket t_2 \rrbracket_{s,h}^X$ is essentially split as shown in the following diagram:



As we can see, in this case the heap h_j witnesses a path that goes from $\llbracket t_1 \rrbracket_{s,h}^X$ to $s(u)$ and that can perhaps extend beyond $s(u)$, say until a location ℓ (highlighted in the diagram). It could be that $\ell = \llbracket t_2 \rrbracket_{s,h}^X$. In this case, the path in h that goes from $\llbracket t_1 \rrbracket_{s,h}^X$ to $\llbracket t_2 \rrbracket_{s,h}^X$ is entirely assigned to h_j . Otherwise, $\ell \in \text{dom}(h_k)$ and so the path in h that goes from ℓ to $\llbracket t_2 \rrbracket_{s,h}^X$ is split between h_1 and h_2 . All the locations in this path belong to either $\text{Rem}[s]_{s,h_1}^{X,\alpha_1}$ or $\text{Rem}[s]_{s,h_2}^{X,\alpha_2}$, with (possibly) the exception of the predecessor of $\llbracket t_2 \rrbracket_{s,h}^X$. Indeed, if $\llbracket t_2 \rrbracket_{s,h}^X$ is assigned to a variable in X , then this location belongs to either $\text{Pred}[s]_{s,h_1}^X(x)$ or $\text{Pred}[s]_{s,h_2}^X(x)$. Therefore, we conclude that:

- the locations in the path of h_j going from $s(u)$ to ℓ counts for the satisfaction of core formulae of the form $u \in \text{sees}_X(t', t'') \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$, again with respect to $\overrightarrow{\beta}$. Here, t' is such that $\llbracket t' \rrbracket_{s,h_j}^X = \llbracket t_1 \rrbracket_{s,h}^X$, whereas $\llbracket t'' \rrbracket_{s,h_j}^X = \ell$. Indeed, since we assumed that $\llbracket t_1 \rrbracket_{s,h}^X$ belongs to $\text{Lab}[s]_{s,h_j}^X$, and $\ell \in \text{dom}(h_k)$, it is quite easy to show that then ℓ correspond to a term $e(x)$, for some $x \in X$,
- the locations split in the path of h that goes from ℓ to $\llbracket t_2 \rrbracket_{s,h}^X$ counts for the satisfaction of the core formulae of the form $\text{rem}_{X,\alpha_1}^S \geq \beta_1$ and $\text{rem}_{X,\alpha_2}^S \geq \beta_2$, with (possibly) the exception of the predecessor of $\llbracket t_2 \rrbracket_{s,h}^X$, which could instead count for the satisfaction of the core formulae of the form $\text{pred}_X^S(x) \geq \beta_1$, for some $x \in X$.

Therefore, $S_{\text{right}}(\alpha)$ must be at least $S_{\text{right}}(\max(\alpha_1, \alpha_2))$ (from (A)) plus $\mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2) + 1$ (from (B)), where the last addend 1 is introduced to deal with the case where the predecessor of $\llbracket t_2 \rrbracket_{s,h}^X$ belongs to $\text{Pred}[s]_{s,h_1}^X(x)$ or $\text{Pred}[s]_{s,h_2}^X(x)$. As $\mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2) = \alpha_1 + \alpha_2 = \alpha$, we have $S_{\text{right}}(\alpha) \geq S_{\text{right}}(\max(\alpha_1, \alpha_2)) + \alpha + 1$. For the base case, we notice that $S_{\text{right}}(1)$ should be at least 2. Indeed, we remind the reader that every atomic formula of s must be expressible as a Boolean combination of formulae in $\text{Core}[s](X, 1)$. If $S_{\text{right}}(1) < 2$, then we are not able to provide a Boolean combination of formulae in $\text{Core}[s](X, 1)$ that is equivalent to $u \hookrightarrow x$. Otherwise, this formula can be shown equivalent to

$$\begin{aligned} u \in \text{pred}_X^S(x) \vee \bigwedge_{t \in T[S]^X} ((u = t \wedge \text{sees}_X(t, x) \geq 1 \wedge \neg \text{sees}_X(t, x) \geq 2) \\ \vee (u \in \text{sees}_X(t, x) \geq (1, 1) \wedge \neg u \in \text{sees}_X(t, x) \geq (1, 2))). \end{aligned}$$

In order for the subformula $u \in \text{sees}_X(t, x) \geq (1, 2)$ to be in $\text{Core}[s](X, 1)$, $S_{\text{right}}(1)$ must be at least 2.

Inequalities: $S_{\text{right}}(1) \geq 2$,

$$S_{\text{right}}(\alpha) \geq \max_{\substack{\alpha_1, \alpha_2 \geq 1 \\ \alpha = \alpha_1 + \alpha_2}} (S_{\text{right}}(\max(\alpha_1, \alpha_2)) + \alpha + 1).$$

In the second inequality, $S_{\text{right}}(\max(\alpha_1, \alpha_2))$ is maximal for $\alpha_1 = \alpha - 1$ or $\alpha_2 = \alpha - 1$.

Recurrence system: $\{S_{\text{right}}(1) = 2, S_{\text{right}}(\alpha + 1) = S_{\text{right}}(\alpha) + (\alpha + 1) + 1\}$.

Solution: $S_{\text{right}}(\alpha) = \frac{1}{2}\alpha(\alpha + 3)$.

Upper bound for β in the core formulae of the form $\text{sees}_X(t_1, t_2) \geq \beta$ and for $\overleftarrow{\beta}$ in the core formulae of the form $u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$.

We write $S(\alpha)$ for the upper bound on β and $S_{\text{left}}(\alpha)$ for the upper bound on $\overleftarrow{\beta}$.

Explanation: First of all, we notice that the inequality $S(\alpha) \geq S_{\text{left}}(\alpha) + S_{\text{right}}(\alpha)$ must hold.

Indeed, if a memory state satisfies $u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$, then it also satisfies $\text{sees}_X(t_1, t_2) \geq \overleftarrow{\beta} + \overrightarrow{\beta}$. Therefore, in order to satisfy the \exists -simulation property, the latter formula must belong to $\text{Core}[S](X, \alpha)$.

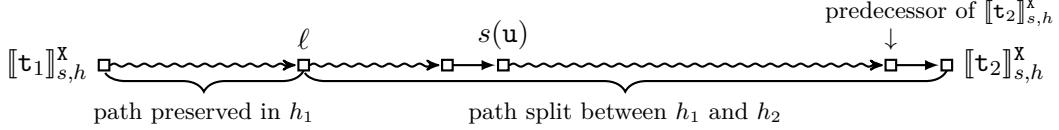
Let us look at $S_{\text{left}}(\alpha)$. If $(s, h) \models u \in \text{sees}_X(t_1, t_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta})$ then h witnesses a path going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $\llbracket t_2 \rrbracket_{s,h}^X$ such that

- every intermediate location of this path does not belong to $\text{Lab}[S]_{s,h}^X$,
- the path can be split into two paths, one of length at least $\overleftarrow{\beta}$ and going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $s(u)$, and one of length at least $\overrightarrow{\beta}$ going from $s(u)$ to $\llbracket t_2 \rrbracket_{s,h}^X$.

Let $j \in \{1, 2\}$ be one of the indices for the subheaps h_1 and h_2 of h , and $k \in \{1, 2\} \setminus \{j\}$ be the other index. When the heap h is split into h_1 and h_2 it can be that:

- $\llbracket t_1 \rrbracket_{s,h}^X$ is a labelled location of (s, h_j) and is such that $\llbracket t_1 \rrbracket_{s,h_1}^X \in \text{dom}(h_j)$,
- in h_j , $s(u)$ is no longer reachable from $\llbracket t_1 \rrbracket_{s,h}^X$.

When this is the case, the path in h going from $\llbracket t_1 \rrbracket_{s,h}^X$ to $\llbracket t_2 \rrbracket_{s,h}^X$ is essentially split as shown in the following diagram:



As we can see, in this case the heap h_j witness a path that goes from $\llbracket t_1 \rrbracket_{s,h}^X$ to a location ℓ and does not feature the location $s(u)$. Moreover, the path in h going from ℓ to $\llbracket t_2 \rrbracket_{s,h}^X$ is split between the subheaps h_1 and h_2 . The locations in this path belong to either $\text{Rem}[S]_{s,h_1}^{X,\alpha_1}$ or $\text{Rem}[S]_{s,h_2}^{X,\alpha_2}$, with the exception of the predecessor of $\llbracket t_2 \rrbracket_{s,h}^X$, which could instead belong to $\text{Pred}[S]_{s,h_1}^X(x)$ or $\text{Pred}[S]_{s,h_2}^X(x)$. In this case,

- A. the locations in the path of h_j going from $s(u)$ to ℓ counts for the satisfaction of core formulae of the form $\text{sees}_X(t', t'') \geq \beta$. Here, t' is such that $\llbracket t' \rrbracket_{s,h_j}^X = \llbracket t_1 \rrbracket_{s,h}^X$, whereas $\llbracket t'' \rrbracket_{s,h_j}^X = \ell$.
- B. the locations split in the path of h that goes from ℓ to $\llbracket t_2 \rrbracket_{s,h}^X$ counts for the satisfaction of the core formulae of the form $\text{rem}_{X,\alpha_1}^S \geq \beta_1$ and $\text{rem}_{X,\alpha_2}^S \geq \beta_2$, with the exception of the predecessor of $\llbracket t_2 \rrbracket_{s,h}^X$, which could count instead for the satisfaction of the core formulae of the form $\text{pred}_X^S(x) \geq \beta_1$, for some $x \in X$. Moreover, as $s(u)$ belongs to this path and it is different from $\llbracket t_2 \rrbracket_{s,h}^X$, exactly one among $(s, h_1) \models u \in \text{rem}_{X,\alpha}^S$, $(s, h_2) \models u \in \text{rem}_{X,\alpha}^S$, $(s, h_1) \models u \in \text{pred}_X^S(x)$, $(s, h_2) \models u \in \text{pred}_X^S(x)$ holds.

So, $S_{\text{left}}(\alpha)$ should be at least $S(\max(\alpha_1, \alpha_2))$ (from (A)). Moreover, $S_{\text{left}}(\alpha) + S_{\text{right}}(\alpha)$ should be at least $S(\max(\alpha_1, \alpha_2))$ (again from (A)) plus $R(\alpha_1) + R(\alpha_2) + 2$ (from (B)), where the last addend 2 is introduced to deal with the fact that the predecessor of $\llbracket t_2 \rrbracket_{s,h}^X$ could belong to $\text{Pred}[S]_{s,h_1}^X(x)$ or $\text{Pred}[S]_{s,h_2}^X(x)$, and to deal with the fact that $s(u)$ can be in

$\text{Rem}[S]_{s,h_1}^{X,\alpha_1} \text{Rem}[S]_{s,h_2}^{X,\alpha_2}$, $\text{Pred}[S]_{s,h_1}^X(x)$ or $\text{Pred}[S]_{s,h_2}^X(x)$. Since $\mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2) = \alpha_1 + \alpha_2 = \alpha$, the expression $\mathcal{R}(\alpha_1) + \mathcal{R}(\alpha_2) + 2$ is equivalent to $\alpha + 2$. We have the following inequalities,

$$\begin{aligned}\mathcal{S}_{\text{left}}(\alpha) &\geq \mathcal{S}(\max(\alpha_1, \alpha_2)), \\ \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha) &\geq \mathcal{S}(\max(\alpha_1, \alpha_2)) + \alpha + 2.\end{aligned}$$

Since we already found $\mathcal{S}_{\text{right}}(\alpha) = \frac{1}{2}\alpha(\alpha+3)$, we know that $\mathcal{S}_{\text{right}}(\alpha) \geq \alpha + 1$ and therefore the two inequalities above are satisfied whenever $\mathcal{S}_{\text{left}}(\alpha) \geq \mathcal{S}(\max(\alpha_1, \alpha_2)) + 1$. Similarly to $\mathcal{S}_{\text{right}}(1)$, we also require $\mathcal{S}_{\text{left}}(1) \geq 2$ to hold. Indeed, if $\mathcal{S}_{\text{left}}(1) < 2$, then we are not able to provide a Boolean combination of core formulae in $\text{Core}[S](X, 1)$ that is equivalent to $x \hookrightarrow u$. Otherwise, this formula can be shown equivalent to

$$\begin{aligned}\bigwedge_{t \in T[S]^X} (u = t \wedge \text{sees}_X(x, t) \geq 1 \wedge \neg \text{sees}_X(x, t) \geq 2) \\ \vee (u \in \text{sees}_X(x, t) \geq (1, 1) \wedge \neg u \in \text{sees}_X(x, t) \geq (2, 1)).\end{aligned}$$

To belong to $\text{Core}[S](X, 1)$, the subformula $u \in \text{sees}_X(x, t) \geq (2, 1)$ requires $\mathcal{S}_{\text{left}}(1) \geq 2$.

Inequalities: $\mathcal{S}(\alpha) \geq \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha)$,

$$\begin{aligned}\mathcal{S}_{\text{left}}(1) &\geq 2, \\ \mathcal{S}_{\text{left}}(\alpha) &\geq \max_{\substack{\alpha_1, \alpha_2 \geq 1 \\ \alpha = \alpha_1 + \alpha_2}} (\mathcal{S}(\max(\alpha_1 + \alpha_2)) + 1).\end{aligned}$$

In the inequalities above, $\mathcal{S}(\max(\alpha_1, \alpha_2))$ is maximal for $\alpha_1 = \alpha - 1$ or $\alpha_2 = \alpha - 1$.

Recurrence system: $\{\mathcal{S}(\alpha) = \mathcal{S}_{\text{left}}(\alpha) + \mathcal{S}_{\text{right}}(\alpha), \mathcal{S}_{\text{left}}(1) = 2, \mathcal{S}_{\text{left}}(\alpha + 1) = \mathcal{S}(\alpha) + 1\}$.

Solution: $\mathcal{S}_{\text{left}}(\alpha) = \frac{1}{6}\alpha(\alpha + 1)(\alpha + 2) + 1$ and $\mathcal{S}(\alpha) = \frac{1}{6}(\alpha + 1)(\alpha + 2)(\alpha + 3)$.

Proofs of Lemma 5.40(I), Lemma 5.40(IV) and Lemma 5.40(V).

We first prove an intermediate technical result.

Lemma C.1. Let (s, h) be a memory state and consider a set of locations $L \subseteq \text{dom}(h)$ such that, for every $t \in T[S]^X$, $L \cap \text{Path}[S]_{s,h}^X(t) = \emptyset$. Let $\hat{h} = h \setminus \{(\ell, \ell') \in h \mid \ell \in L\}$. We have:

- O. Let $x \in X$ and $\delta \geq 0$. $h^\delta(s(x))$ and $\hat{h}^\delta(s(x))$ are equidefined. When defined, they are equal.
- A. For all $t \in T[S]^X$, $\llbracket t \rrbracket_{s,h}^X$ and $\llbracket t \rrbracket_{s,\hat{h}}^X$ are equidefined. When defined, they are equal.
- B. For every $t \in T[S]^X$,
 - (a) $\text{sby}_{s,h}^X(t)$ and $\text{sby}_{s,\hat{h}}^X(t)$ are equidefined. When defined, they are equal,
 - (b) $\text{Path}[S]_{s,h}^X(t) = \text{Path}[S]_{s,\hat{h}}^X(t)$,
 - (c) let $\delta \in [1, \text{card}(\text{Path}[S]_{s,h}^X(t))]$. $h^\delta(\llbracket t \rrbracket_{s,h}^X) = s(u)$ iff $\hat{h}^\delta(\llbracket t \rrbracket_{s,\hat{h}}^X) = s(u)$.

Essentially, this lemma generalises an intermediate step in the proof of Lemma 5.40(III).

Proof of Lemma C.1(O). The proof is by induction on δ .

base case: $\delta = 0$. Straightforward.

induction step: $\delta \geq 1$. Suppose $h^\delta(s(x)) = \ell$. Let $\ell' = h^{\delta-1}(s(x))$, and so $h(\ell') = \ell$. By induction hypothesis, $\ell' = \hat{h}^{\delta-1}(s(x))$. We prove that $\ell' \in \text{dom}(\hat{h})$, which implies $\hat{h}(\ell') = \ell$ directly from $\hat{h} \subseteq h$, concluding the proof. Let $\tilde{\ell}$ be the last labelled location in the minimal path of h going from $s(x)$ to ℓ' . Formally, $\tilde{\ell} \in \text{Lab}[S]_{s,h}^X$ and there is $\delta' \in [0, \delta]$ such that $h^{\delta'}(\tilde{\ell}) = \ell'$ and for every $\delta'' \in [0, \delta' - 1]$, $h^{\delta''-1}(\tilde{\ell}) \notin \text{Lab}[S]_{s,h}^X$. Since $s(x)$ is a labelled location, the location $\tilde{\ell}$ exists. We divide the proof depending on whether $\tilde{\ell} = \ell'$.

case: $\tilde{\ell} = \ell'$. In this case, ℓ' is a labelled location and, since it belongs to $\text{dom}(h)$, we have $\ell' \in \text{Path}[s]_{s,h}^X(\ell')$. Therefore, by definition of L , $\ell' \notin L$, which in turn implies $\ell' \in \text{dom}(\hat{h})$, by definition of \hat{h} .

case: $\tilde{\ell} \neq \ell'$. By definition $\tilde{\ell}$ is a labelled location and belongs to $\text{dom}(h)$. Moreover, in the minimal path of h going from $\tilde{\ell}$ to ℓ , no location except for $\tilde{\ell}$ belongs to $\text{Lab}[s]_{s,h}^X$. By definition of $\text{Path}[s]_{s,h}^X(\ell)$, we conclude that ℓ' belong to this set. This means that $\ell' \notin L$, which in turn implies $\ell' \in \text{dom}(\hat{h})$, from $\ell \in \text{dom}(h)$ and by definition of \hat{h} . \square

Proof of Lemma C.1(A). Follows directly from (O). The statement is trivial when t corresponds to a program variable. We show the cases for end-point variables and meet-point variables.

case: $t = e(x)$. Suppose $\llbracket e(x) \rrbracket_{s,h}^X = \ell$. By definition of end-point variable, there is $\delta \geq 1$ such that $h^\delta(s(x)) = \ell$ and, if $\ell \in \text{dom}(h)$ then ℓ belongs to a cycle in h whereas $h_j^{\delta-1}(s(x))$ does not. From Lemma C.1(O), $\hat{h}^\delta(s(x)) = \ell$, $\hat{h}^{\delta-1}(s(x)) = \hat{h}^{\delta-1}(s(x))$, and $\ell \in \text{dom}(\hat{h})$ if and only if $\ell \in \text{dom}(\hat{h})$. Moreover, again by Lemma C.1(O), if ℓ (resp. $\hat{h}^{\delta-1}(s(x))$) belongs to a cycle in h then it belongs to a cycle in \hat{h} , and vice versa. So, $\llbracket e(x) \rrbracket_{s,\hat{h}}^X = \llbracket e(x) \rrbracket_{s,h}^X$. The other direction, i.e. if $\llbracket e(x) \rrbracket_{s,\hat{h}}^X$ is defined then $\llbracket e(x) \rrbracket_{s,h}^X = \llbracket e(x) \rrbracket_{s,\hat{h}}^X$, is symmetrical.

case: $t = m(x, y)$. Suppose $\llbracket m(x, y) \rrbracket_{s,h}^X = \ell$. By definition of meet-point variable, there are $\delta_1, \delta_2 \geq 1$ such that $h^{\delta_1}(s(x)) = h^{\delta_2}(s(y)) = \ell$ and for all $\delta'_1 \in [0, \delta_1]$ and $\delta'_2 \in [0, \delta_2]$, if $\delta'_1 + \delta'_2 < \delta_1 + \delta_2$ then $h^{\delta'_1}(s(x)) \neq h^{\delta'_2}(s(y))$. Moreover, ℓ does not belong to a cycle. From Lemma C.1(O), $\hat{h}^{\delta_1}(s(x)) = \hat{h}^{\delta_2}(s(y)) = \ell$, and for all $\delta'_1 \in [0, \delta_1]$ and $\delta'_2 \in [0, \delta_2]$, if $\delta'_1 + \delta'_2 < \delta_1 + \delta_2$ then $\hat{h}^{\delta'_1}(s(x)) \neq \hat{h}^{\delta'_2}(s(y))$. Moreover, since ℓ does not belong to a cycle in h and $\hat{h} \subseteq h$, we conclude that ℓ does not belong to a cycle in \hat{h} . By definition of meet-point variable, $\ell = \llbracket m(x, y) \rrbracket_{s,\hat{h}}^X$.

For the other direction, suppose $\llbracket m(x, y) \rrbracket_{s,\hat{h}}^X = \ell$. By definition of meet-point variable and the fact that $\hat{h} \subseteq h$, there are $\delta_1, \delta_2 \geq 1$ such that $h^{\delta_1}(s(x)) = h^{\delta_2}(s(y)) = \ell$ and for all $\delta'_1 \in [0, \delta_1]$ and $\delta'_2 \in [0, \delta_2]$, if $\delta'_1 + \delta'_2 < \delta_1 + \delta_2$ then $h^{\delta'_1}(s(x)) \neq h^{\delta'_2}(s(y))$. In order to conclude that $\ell = \llbracket m(x, y) \rrbracket_{s,h}^X$, it is sufficient to proof that ℓ does not belong to a cycle in h . *Ad absurdum*, suppose that ℓ belongs to a cycle in h . Then, there is $\delta' \geq 1$ such that $h^{\delta_1}(s(x)) = \ell = h^{\delta_1+\delta'}(s(x)) = \ell$. From Lemma C.1(O), we conclude that $\hat{h}^{\delta_1}(s(x)) = \ell = \hat{h}^{\delta_1+\delta'}(s(x)) = \ell$. However, this implies that ℓ belongs to a cycle in \hat{h} : a contradiction. Thus, ℓ does not belong to a cycle in h , and so $\ell = \llbracket m(x, y) \rrbracket_{s,h}^X$. \square

Proof of Lemma C.1(B). As in the case of (A), the statements (a)–(c) follow directly form (O) and the definition of $\llbracket \cdot \rrbracket^X$. In particular, given $t \in T[s]^X$ and $\delta \geq 0$, by (O), we conclude that, when defined, $h^\delta(\llbracket t \rrbracket_{s,h_j}^X) = \hat{h}^\delta(\llbracket t \rrbracket_{s,h'_j}^X)$. This implies (c). Moreover, by (A), $\text{Lab}[s]_{s,h}^X = \text{Lab}[s]_{s,\hat{h}}^X$, which allows us to conclude that (a) and (b) hold. \square

We now conclude the proof of Lemma 5.40, by dealing with the cases (I), (IV) and (V). Notice that in the statement of Lemma 5.40, the two memory states (s, h) and (s, h') are supposed to be such that for every $t \in T[s]^X$, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h'}^X$. This hypothesis is not needed for the cases (I), (IV) and (V), and thus it is dropped in their formal statements provided below.

Lemma 5.40(I). Let (s, h) and (s, h') be two memory states such that $(s, h) \approx_{X,\alpha}^S (s, h')$ and there is $x \in X$ such that $\{(\ell, \ell') \in h \mid \ell \in \text{Pred}[s]_{s,h}^X(x)\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[s]_{s,h'}^X(x)\}$. Then, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h')$.

Proof. Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Notice that this requires α to be at least two, otherwise the lemma trivially holds. We recall that the integer β in formulae of $\text{Core}[S](X, \alpha)$ of the form $\text{pred}_X^S(x) \geq \beta$ ranges over $[1, \alpha]$. If $\text{card}(\text{Pred}[S]_{s,h}^X(x)) < \alpha$ then the lemma holds directly from Lemma 5.39. Indeed, in this case, from the equisatisfaction of the core formulae of the form $\text{pred}_X^S(x) \geq \beta$ we conclude that $\text{card}(\text{Pred}[S]_{s,h}^X(x)) = \text{card}(\text{Pred}[S]_{s,h'}^X(x))$. By definition of h and h' , we conclude that $((s, h), (s, h')) \in (\bigcap_{\alpha' \geq 1} \approx_{X, \alpha'}^S)$, which allows us to apply Lemma 5.39. Therefore, in the following we assume $\text{card}(\text{Pred}[S]_{s,h}^X(x)) \geq \alpha$, which implies $\text{card}(\text{Pred}[S]_{s,h'}^X(x)) \geq \alpha$ again from the equisatisfaction of the core formulae $\text{pred}_X^S(x) \geq \beta$.

We define the following two sets T_1, T_2 :

$$T_1 = \{\ell \in \text{Pred}[S]_{s,h}^X(x) \mid \ell \in \text{dom}(h_1)\}, \quad T_2 = \{\ell \in \text{Pred}[S]_{s,h}^X(x) \mid \ell \in \text{dom}(h_2)\}.$$

Given $j \in [1, 2]$, T_j contains every location of $\text{Pred}[S]_{s,h}^X(x)$ that is a memory cell of h_j . Moreover, T_1, T_2 are mutually disjoint sets, and their union is $\text{Pred}[S]_{s,h}^X(x)$. As a first step of the proof, we aim at defining three similar sets T'_1, T'_2 with respect to $\text{Pred}[S]_{s,h'}^X(x)$. These sets should also satisfy cardinality constraints depending on α_j , as well as constraints involving the location $s(u)$. More precisely:

1. T'_1, T'_2 are mutually disjoint. Their union is $\text{Pred}[S]_{s,h'}^X(x)$,
2. for all $j \in \{1, 2\}$, $\min(\text{card}(T_j), \alpha_j) = \min(\text{card}(T'_j), \alpha_j)$,
3. for all $j \in \{1, 2\}$, $s(u) \in T_j$ if and only if $s(u) \in T'_j$.

The definition of T'_1 and T'_2 follows the strategy below.

```

if  $\text{card}(T_1) < \alpha_1$  then
  let  $T'_1$  be a set of  $\text{card}(T_1)$  locations in  $\text{Pred}[S]_{s,h'}^X(x)$  such that  $s(u) \in T_1$  iff  $s(u) \in T'_1$ .
   $T'_2 \leftarrow \text{Pred}[S]_{s,h'}^X(x) \setminus T'_1$ .
else if  $\text{card}(T_2) < \alpha_2$  then
  let  $T'_2$  be a set of  $\text{card}(T_2)$  locations in  $\text{Pred}[S]_{s,h'}^X(x)$  such that  $s(u) \in T_2$  iff  $s(u) \in T'_2$ .
   $T'_1 \leftarrow \text{Pred}[S]_{s,h'}^X(x) \setminus T'_2$ .
else (i.e.  $\text{card}(T_1) \geq \alpha_1$  and  $\text{card}(T_2) \geq \alpha_2$ )
  let  $T'_1$  be a set of  $\alpha_1$  locations in  $\text{Pred}[S]_{s,h'}^X(x)$  such that  $s(u) \in T_1$  iff  $s(u) \in T'_1$ .
   $T'_2 \leftarrow \text{Pred}[S]_{s,h'}^X(x) \setminus T'_1$ .

```

Since we are assuming that both $\text{Pred}[S]_{s,h}^X(x)$ and $\text{Pred}[S]_{s,h'}^X(x)$ contain at least α elements (where $\alpha = \alpha_1 + \alpha_2$), and moreover we have $s(u) \in \text{Pred}[S]_{s,h}^X(x)$ iff $s(u) \in \text{Pred}[S]_{s,h'}^X(x)$ (from the equisatisfaction of the core formula $u \in \text{pred}_X^S(x)$), we conclude that the sets T'_1 and T'_2 are well-defined. Moreover, their definition is such that $T_1 \cap T_2 = \emptyset$ and $T_1 \cup T_2 = \text{Pred}[S]_{s,h'}^X(x)$. The property (1) is satisfied. Let us show that the same holds true for the properties (2) and (3). The proof is divided in three cases, according to the strategy.

case: $\text{card}(T_1) < \alpha_1$. From the first case of the strategy, we have $\text{card}(T_1) = \text{card}(T'_1)$. By definition of T_1 and T_2 we have $\text{card}(\text{Pred}[S]_{s,h}^X(x)) = \text{card}(T_1) + \text{card}(T_2)$. Similarly, from (1), $\text{card}(\text{Pred}[S]_{s,h'}^X(x)) = \text{card}(T'_1) + \text{card}(T'_2)$. As $\text{card}(\text{Pred}[S]_{s,h}^X(x))$ and $\text{card}(\text{Pred}[S]_{s,h'}^X(x))$ contain at least α locations, where $\alpha = \alpha_1 + \alpha_2 \geq \text{card}(T_1) + \alpha_2$, this allows us to conclude that T_2 and T'_2 contain at least α_2 locations. Thus, (2) is satisfied. Again from the definition of T'_1 , we have $s(u) \in T_1$ if and only if $s(u) \in T'_1$. Therefore, in order to conclude that (3) it is sufficient to show that $s(u) \in T_2$ if and only if $s(u) \in T'_2$. This is quite ob-

vious: for the left-to-right direction, if $s(u) \in T_2$ then $s(u) \notin T_1$ and $s(u) \in \text{Pred}[\mathcal{S}]_{s,h}^X(x)$. By definition of T'_1 we derive $s(u) \notin T'_1$, whereas from the equisatisfaction of the core formula $u \in \text{pred}_X^{\mathcal{S}}(x)$ we have $s(u) \in \text{Pred}[\mathcal{S}]_{s,h'}^X(x)$. As $T'_1 \cup T'_2 = \text{Pred}[\mathcal{S}]_{s,h'}^X(x)$, we conclude that $s(u) \in T'_2$. The other direction is proved symmetrically.

case: $\text{card}(T_2) < \alpha_2$. Analogous to the previous case, by swapping the indices 1 and 2 and considering the second case of the strategy.

case: $\text{card}(T_1) \geq \alpha_1$ and $\text{card}(T_2) \geq \alpha_2$. From the third case of the strategy, $\text{card}(T'_1) = \alpha_1$. Similarly to the first case, this allows us to conclude that T'_2 has at least α_2 locations. Thus, (2) is satisfied. The proof of (3) is equal to the one given in the first case.

We rely on T'_1 and T'_2 in order to define the heaps h'_1 and h'_2 such that $(s, h_1) \approx_{X,\alpha_1}^{\mathcal{S}} (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^{\mathcal{S}} (s, h'_2)$ (as required by the hop relation $\leftrightarrow_{X,\alpha}^W$). First, let us define the two heaps $\widehat{h_1} \stackrel{\text{def}}{=} h_1 \setminus \{(\ell, \ell') \in h_1 \mid \ell \in T_1\}$ and $\widehat{h_2} \stackrel{\text{def}}{=} h_2 \setminus \{(\ell, \ell') \in h_2 \mid \ell \in T_2\}$. By definition of T_1 and T_2 , $\text{dom}(\widehat{h_1}) \cap \text{Pred}[\mathcal{S}]_{s,h}^X(x)$ and $\text{dom}(\widehat{h_2}) \cap \text{Pred}[\mathcal{S}]_{s,h}^X(x)$ are both empty. Moreover, by $h = h_1 + h_2$,

$$h = \widehat{h_1} + \widehat{h_2} + \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[\mathcal{S}]_{s,h}^X(x)\}.$$

From the hypothesis $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[\mathcal{S}]_{s,h}^X(x)\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[\mathcal{S}]_{s,h'}^X(x)\}$ we derive $\widehat{h_1} + \widehat{h_2} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[\mathcal{S}]_{s,h'}^X(x)\}$. We define the heaps h'_1 and h'_2 as:

$$h'_1 \stackrel{\text{def}}{=} \widehat{h_1} + \{(\ell, \ell') \in h' \mid \ell \in T'_1\}, \quad h'_2 \stackrel{\text{def}}{=} \widehat{h_2} + \{(\ell, \ell') \in h' \mid \ell \in T'_2\}.$$

From (1), the heaps $\{(\ell, \ell') \in h' \mid \ell \in T'_1\}$ and $\{(\ell, \ell') \in h' \mid \ell \in T'_2\}$ are disjoint, and their union is $\{(\ell, \ell') \in h' \mid \ell \in \text{Pred}[\mathcal{S}]_{s,h'}^X(x)\}$. Thus, h'_1 and h'_2 are well-defined, they are disjoint, and $h' = h'_1 + h'_2$. We now discuss seven properties of h'_1 and h'_2 which directly lead to $(s, h_1) \approx_{X,\alpha_1}^{\mathcal{S}} (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^{\mathcal{S}} (s, h'_2)$, as done for Lemma 5.40(III). Let $j \in \{1, 2\}$.

- A. For all $t \in T[\mathcal{S}]^X$, $\llbracket t \rrbracket_{s,h_j}^X$ and $\llbracket t \rrbracket_{s,h'_j}^X$ are equidefined. When defined, they are equal.

Proof of (A). We notice that the sets $\text{Pred}[\mathcal{S}]_{s,h}^X(x)$ (resp. $\text{Pred}[\mathcal{S}]_{s,h'}^X(x)$) enjoy the property of the set L of Lemma C.1, with respect to the memory states $\widehat{h_j}$ and h_j (resp. h'_j). Therefore if $\llbracket t \rrbracket_{s,h_j}^X$ is defined, by Lemma C.1(A), $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,\widehat{h_j}}^X$. Again by Lemma C.1(A), as $\llbracket t \rrbracket_{s,\widehat{h_j}}^X$ is defined, $\llbracket t \rrbracket_{s,h'_j}^X = \llbracket t \rrbracket_{s,\widehat{h_j}}^X$. Thus, $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h'_j}^X$. The other direction, i.e. if $\llbracket t \rrbracket_{s,h'_j}^X$ is defined then $\llbracket t \rrbracket_{s,h_j}^X = \llbracket t \rrbracket_{s,h'_j}^X$, is analogous.

- B. For every $t \in T[\mathcal{S}]^X$,

- (a) $\text{sby}_{s,h_j}^X(t)$ and $\text{sby}_{s,h'_j}^X(t)$ are equidefined. When defined, they are equal,
- (b) $\text{Path}[\mathcal{S}]_{s,h_j}^X(t) = \text{Path}[\mathcal{S}]_{s,h'_j}^X(t)$,
- (c) let $\delta \in [1, \text{card}(\text{Path}[\mathcal{S}]_{s,h_j}^X(t))]$. $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) = s(u)$ iff $h'_j^\delta(\llbracket t \rrbracket_{s,h'_j}^X) = s(u)$.

Proof of (B). Similarly to (A), it follows directly from Lemma C.1(B).

- C. For every $y \in X$,

- (d) $\min(\text{Pred}[\mathcal{S}]_{s,h_j}^X(y), \alpha_j) = \min(\text{Pred}[\mathcal{S}]_{s,h'_j}^X(y), \alpha_j)$,
- (e) $s(u) \in \text{Pred}[\mathcal{S}]_{s,h_j}^X(y)$ if and only if $s(u) \in \text{Pred}[\mathcal{S}]_{s,h'_j}^X(y)$.

We start by showing the following equality and inclusions:

- (f) $\text{Pred}[\mathcal{S}]_{s,h_j}^X(y) \setminus T_j = \text{Pred}[\mathcal{S}]_{s,h'_j}^X(y) \setminus T'_j$,
- (g) $T_j \subseteq \text{Pred}[\mathcal{S}]_{s,h_j}^X(x)$ and $T'_j \subseteq \text{Pred}[\mathcal{S}]_{s,h'_j}^X(x)$.

Proof of (f). (\subseteq): Suppose $\ell \in \text{Pred}[\mathcal{S}]_{s,h_j}^X(y) \setminus T_j$. By definition of $\text{Pred}[\mathcal{S}]_{s,h_j}^X(y)$:

- h. $h_j(\ell) = s(y)$,
- i. for every $z \in X$ and every $\delta \geq 0$, $h_j^\delta(s(z)) \neq \ell$.

Since ℓ is in the domain of h_j but does not belong to T_j , by definition of $\widehat{h_j}$, we have $\ell \in \text{dom}(\widehat{h_j})$. From (h) and $\widehat{h_j} \subseteq h_j$, $\widehat{h_j}(\ell) = s(y)$. By definition of h'_j , we conclude that $h'_j(\ell) = s(y)$ and $\ell \notin T'_j$. From (i), (b), it cannot be that there is $z \in X$ and $\delta \geq 0$ such that $h'^\delta(s(z)) = \ell$. Indeed, *ad absurdum*, suppose there is $z \geq 0$ and $\delta \geq 0$ such that $h'^\delta(s(z)) = \ell$. This implies that there is a term $t \in T[\mathcal{S}]^X$ such that $\ell \in \text{Path}[\mathcal{S}]_{s,h'_j}^X(t)$. By (b), $\ell \in \text{Path}[\mathcal{S}]_{s,h_j}^X(t)$. By definition of $\llbracket \cdot \rrbracket_{s,h_j}^X$, h_j witnesses a path going from a location assigned to a program variable in X , say v , to $\llbracket t \rrbracket_{s,h_j}^X$. However, by definition of $\text{Path}[\mathcal{S}]_{s,h_j}^X(t)$, this implies that h_j witnesses a path going from $s(v)$ to ℓ , which contradicts (i). We conclude that, $\ell \in \text{Pred}[\mathcal{S}]_{s,h'_j}^X(y)$.

(\supseteq): Symmetrical to the other direction.

Proof of (g). We prove the inclusion $T_j \subseteq \text{Pred}[\mathcal{S}]_{s,h_j}^X(x)$. Suppose $\ell \in T_j$. By definition, $\ell \in \text{Pred}[\mathcal{S}]_{s,h}^X(x)$, which means that $h(\ell) = s(x)$ and for every $z \in X$ and every $\delta \geq 0$, $h^\delta(s(z)) \neq \ell$. By definition of h_j , $h_j(\ell) = s(x)$. As $h_j \subseteq h$, for every $z \in X$ and every $\delta \geq 0$, $h_j^\delta(s(z)) \neq \ell$. We conclude that $\ell \in \text{Pred}[\mathcal{S}]_{s,h_j}^X(x)$.

The proof of the inclusion $T'_j \subseteq \text{Pred}[\mathcal{S}]_{s,h'_j}^X(x)$ is analogous.

Proof of (C). If $s(x) \neq s(y)$ then both (d) and (e) hold directly from (f). Otherwise, let us suppose that $s(x) = s(y)$. We prove that (d) holds. The property (2) of the constructions states that $\min(\text{card}(T_j), \alpha_j) = \min(\text{card}(T'_j), \alpha_j)$. By using the fact that for all $a, b, c, d \in \mathbb{N}$, $\min(a, d) = \min(b, d)$ implies $\min(a + c, d) = \min(b + c, d)$,

$$\begin{aligned} & \min(\text{card}(T_j) + \text{card}(\text{Pred}[\mathcal{S}]_{s,h_j}^X(x) \setminus T_j), \alpha_j) \\ &= \min(\text{card}(T'_j) + \text{card}(\text{Pred}[\mathcal{S}]_{s,h_j}^X(x) \setminus T_j), \alpha_j). \end{aligned}$$

From (f), $\text{card}(\text{Pred}[\mathcal{S}]_{s,h_j}^X(x) \setminus T_j) = \text{card}(\text{Pred}[\mathcal{S}]_{s,h'_j}^X(x) \setminus T'_j)$, and so

$$\begin{aligned} & \min(\text{card}(T_j) + \text{card}(\text{Pred}[\mathcal{S}]_{s,h_j}^X(x) \setminus T_j), \alpha_j) \\ &= \min(\text{card}(T'_j) + \text{card}(\text{Pred}[\mathcal{S}]_{s,h'_j}^X(x) \setminus T'_j), \alpha_j). \end{aligned} \tag{\dagger}$$

By (g), we have $\text{Pred}[\mathcal{S}]_{s,h_j}^X(x) = (\text{Pred}[\mathcal{S}]_{s,h_j}^X(x) \setminus T_j) \cup T_j$ and so

$$\text{card}(\text{Pred}[\mathcal{S}]_{s,h_j}^X(x)) = \text{card}(\text{Pred}[\mathcal{S}]_{s,h_j}^X(x) \setminus T_j) + \text{card}(T_j).$$

Similarly (again from (g)),

$$\text{card}(\text{Pred}[\mathcal{S}]_{s,h'_j}^X(x)) = \text{card}(\text{Pred}[\mathcal{S}]_{s,h'_j}^X(x) \setminus T'_j) + \text{card}(T'_j).$$

By (d) we have (d), i.e. $\min(\text{card}(\text{Pred}[\mathcal{S}]_{s,h_j}^X(x)), \alpha_j) = \min(\text{card}(\text{Pred}[\mathcal{S}]_{s,h'_j}^X(x)), \alpha_j)$.

Let us prove (e). For the left-to-right direction, suppose $s(u) \in \text{Pred}[\mathcal{S}]_{s,h_j}^X(x)$. By (g), we have either $s(u) \in \text{Pred}[\mathcal{S}]_{s,h_j}^X(x) \setminus T_j$ or $s(u) \in T_j$. In the former case, directly from (f), we conclude that $s(u) \in \text{Pred}[\mathcal{S}]_{s,h'_j}^X(x)$. In the latter case, from the property (3) of the construction, we have $s(u) \in T'_j$, which implies $s(u) \in \text{Pred}[\mathcal{S}]_{s,h'_j}^X(x)$ by (g). The right-to-left direction is proved symmetrically.

- D. For every $\beta \in [1, \alpha_j]$, $\text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta) = \text{Cycl}[\mathcal{S}]_{s,h'_j}^X(\beta)$.

Proof of (D). (\subseteq): Consider a set $L \in \text{Cycl}[\mathcal{S}]_{s,h_j}^X(\beta)$. By definition, L describes an unlabelled cycle in h_j , of length β . In particular, this means that $L \cap \text{Pred}[\mathcal{S}]_{s,h}^X(x) = \emptyset$,

as otherwise $s(\mathbf{x})$ would also belong to \mathbf{L} , in contradiction with the fact that \mathbf{L} does not contain labelled locations. Therefore, $\mathbf{L} \cap T_j = \emptyset$, which implies $\mathbf{L} \subseteq \text{dom}(\widehat{h_j})$ by definition of $\widehat{h_j}$. As $\widehat{h_j} \subseteq h_j$, we conclude that \mathbf{L} describes a cycle in $\widehat{h_j}$. As $\widehat{h_j} \subseteq h'_j$, we conclude that \mathbf{L} describes a cycle in h'_j . From (A) this cycle does not contain labelled locations. Thus, $\mathbf{L} \in \text{Cycl}[s]_{s,h'_j}^{\mathbf{x},\alpha_j}(\beta)$.

(\supseteq): Symmetrical to the other direction.

$$\text{E. } \uparrow\text{Cycl}[s]_{s,h_j}^{\mathbf{x},\alpha_j} = \uparrow\text{Cycl}[s]_{s,h'_j}^{\mathbf{x},\alpha_j}.$$

The proof of this case is analogous to the one of (D).

$$\text{F. } \text{Rem}[s]_{s,h_j}^{\mathbf{x},\alpha_j} = \text{Rem}[s]_{s,h'_j}^{\mathbf{x},\alpha_j}.$$

Proof of (F). (\subseteq): Consider a location $\ell \in \text{Rem}[s]_{s,h_j}^{\mathbf{x},\alpha_j}$. By definition of $\text{Rem}[s]_{s,h_j}^{\mathbf{x},\alpha_j}$, we have $\ell \in \text{dom}(h_j)$ and $h_j(\ell) \neq s(\mathbf{x})$. Therefore, $\ell \notin T_j$, which implies $\ell \in \text{dom}(\widehat{h_j})$ and $\widehat{h_j}(\ell) \neq s(\mathbf{x})$ by definition of $\widehat{h_j}$. As $\widehat{h_j} \subseteq h'_j$, $\ell \in \text{dom}(h'_j)$ and $h'_j(\ell) \neq s(\mathbf{x})$. From (A), $\ell \notin \text{Lab}[s]_{s,h'_j}^{\mathbf{x}}$. From (B), for all $\ell' \in \text{Lab}[s]_{s,h'_j}^{\mathbf{x}}$, $\ell' \notin \text{Path}[s]_{s,h'_j}^{\mathbf{x}}(\ell')$.

From (D), for every $\beta \in [1, \alpha_j]$, $\ell \notin \text{Cycl}[s]_{s,h'_j}^{\mathbf{x},\alpha_j}(\beta)$. From (E), $\ell \notin \uparrow\text{Cycl}[s]_{s,h'_j}^{\mathbf{x},\alpha_j}$.

By definition of $\text{Rem}[s]_{s,h'_j}^{\mathbf{x},\alpha_j}$, we conclude that $\ell \in \text{Rem}[s]_{s,h'_j}^{\mathbf{x},\alpha_j}$.

(\supseteq): Symmetrical to the other direction.

The properties (A)–(F) lead directly to $(s, h_j) \approx_{\mathbf{x},\alpha_j}^S (s', h'_j)$, with the same case analysis provided at the end of the proof of Lemma 5.39. Therefore, $(s, h) \leftrightarrow_{\mathbf{x},\alpha}^S (s, h')$. \square

Lemma 5.40(IV). Let (s, h) and (s, h') be two memory states such that $(s, h) \approx_{\mathbf{x},\alpha}^S (s, h')$. If $h \setminus \{(\ell, \ell') \in h \mid \ell \in [\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}]^\flat\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in [\uparrow\text{Cycl}[s]_{s,h'}^{\mathbf{x},\alpha}]^\flat\}$, then $(s, h) \leftrightarrow_{\mathbf{x},\alpha}^S (s, h')$.

Proof. The proof of this lemma largely follows the steps given for the proof of Lemma 5.40(III). Here, we mainly expand on the points that are different from that proof. Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. This requires α to be at least two, otherwise the lemma trivially holds. We define $\mathcal{L}(\alpha) = \frac{1}{2}\alpha(\alpha+3)-1$ and $\mathcal{R}(\alpha) = \alpha$, i.e. the upper bounds given to β' in formulae of the form $\text{loop}_{\mathbf{x}}^S(\beta) \geq \beta'$ and $\text{rem}_{\mathbf{x},\alpha}^S \geq \beta'$, respectively. As in Lemma 5.40(III), the following inequality has a fundamental role in the proof:

$$\mathcal{L}(\alpha) \geq \mathcal{L}(\alpha_1) + \mathcal{L}(\alpha_2) + \mathcal{R}(\max(\alpha_1, \alpha_2)) + 1. \quad (\star)$$

Differently from Lemma 5.40(III), we cannot rely on Lemma 5.39 in order to deal with the case where $\text{card}(\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}) < \mathcal{L}(\alpha)$. Indeed, even though this implies that $\uparrow\text{Cycl}[s]_{s,h'}^{\mathbf{x},\alpha}$ and $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}$ have the same cardinality directly from the equisatisfaction of the core formulae of the form $\uparrow\text{loop}_{\mathbf{x},\alpha}^S \geq \beta$, we cannot deduce that $((s, h), (s, h')) \in (\bigcap_{\alpha' \geq 1} \approx_{\mathbf{x},\alpha'}^S)$. For instance, it could be that $\uparrow\text{Cycl}[s]_{s,h'}^{\mathbf{x},\alpha}$ contains a set of cardinality $\alpha+1$ whereas $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}$ does not, leading to $(s, h) \not\approx_{\mathbf{x},\alpha+1}^S (s, h')$. Nonetheless, as in Lemma 5.40(III), we define the three sets T_1 , T_2 and S as follows:

- $T_1 = \{L \in \uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha} \mid L \subseteq \text{dom}(h_1)\}$,
- $T_2 = \{L \in \uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha} \mid L \subseteq \text{dom}(h_2)\}$,
- $S = \{(L_1, L_2) \mid L_1 \cup L_2 \in \uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}, \emptyset \neq L_1 \subseteq \text{dom}(h_1), \emptyset \neq L_2 \subseteq \text{dom}(h_2)\}$.

Given $j \in [1, 2]$, T_j contains every set of $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{x},\alpha}$ whose locations are memory cells of h_j . The sets in T_1 and T_2 describe loops of length greater than α in h_1 and h_2 , respectively. The

set S contains pairs (L_1, L_2) of non-empty sets of locations that partition a set of $\Uparrow \text{Cycl}[S]_{s,h}^{X,\alpha}$ and are such that L_1 contains memory cells of h_1 whereas L_2 contains memory cells of h_2 . S represents the loops in $\Uparrow \text{Cycl}[S]_{s,h}^{X,\alpha}$ that are split between h_1 and h_2 . The sets T_1 , T_2 and $\{L_1 \cup L_2 \mid (L_1, L_2) \in S\}$ are mutually disjoint sets, and their union is $\Uparrow \text{Cycl}[S]_{s,h}^{X,\alpha}$. We aim at defining three similar sets T'_1 , T'_2 and S' with respect to $\Uparrow \text{Cycl}[S]_{s,h'}^{X,\alpha}$. These sets should satisfy the following properties:

1. $S' \subseteq \{(L'_1, L'_2) \mid L'_1 \text{ and } L'_2 \text{ are non-empty and disjoint, and } L'_1 \cup L'_2 \in \Uparrow \text{Cycl}[S]_{s,h'}^{X,\alpha}\},$
2. T'_1 , T'_2 and $\{L'_1 \cup L'_2 \mid (L'_1, L'_2) \in S'\}$ are mutually disjoint. Their union is $\Uparrow \text{Cycl}[S]_{s,h'}^{X,\alpha}$.
3. For all $j \in \{1, 2\}$, $\min(\text{card}(T_j), \mathcal{L}(\alpha_j)) = \min(\text{card}(T'_j), \mathcal{L}(\alpha_j)),$
4. for all $j \in \{1, 2\}$, $\min(\text{card}([\pi_j(S)]^b), \mathcal{R}(\alpha_j)) = \min(\text{card}([\pi_j(S')]^b), \mathcal{R}(\alpha_j)),$
5. for all $j \in \{1, 2\}$, $s(u) \in [T_j]^b$ if and only if $s(u) \in [T'_j]^b,$
6. for all $j \in \{1, 2\}$, $s(u) \in [\pi_j(S)]^b$ if and only if $s(u) \in [\pi_j(S')]^b.$

Notice that these are exactly the properties required in Lemma 5.40(III) for the homonymous sets. The definition of T'_1 , T'_2 and S' follows the strategy below.

```

1: if  $\text{card}(S) \geq \max(\alpha_1, \alpha_2)$  then
2:   let  $T'_1 \subseteq \text{Cycl}[S]_{s,h'}^X(\beta)$  such that •  $\text{card}(T'_1) = \min(\text{card}(T_1), \mathcal{L}(\alpha_1)),$ 
      •  $s(u) \in [T'_1]^b$  iff  $s(u) \in [T_1]^b.$ 
3:   let  $T'_2 \subseteq \text{Cycl}[S]_{s,h'}^X(\beta) \setminus T'_1$  such that •  $\text{card}(T'_2) = \min(\text{card}(T_2), \mathcal{L}(\alpha_2)),$ 
      •  $s(u) \in [T'_2]^b$  iff  $s(u) \in [T_2]^b.$ 
4:   if  $s(u) \in [\pi_1(S)]^b$  then
5:      $S' \leftarrow \{(L' \setminus \{\ell'\}, \{\ell'\}) \mid L' \in \text{Cycl}[S]_{s,h'}^X(\beta) \setminus (T'_1 \cup T'_2) \text{ and } \ell' = \min(L' \setminus \{s(u)\})\}.$ 
6:   else  $S' \leftarrow \{(\{\ell'\}, L' \setminus \{\ell'\}) \mid L' \in \text{Cycl}[S]_{s,h'}^X(\beta) \setminus (T'_1 \cup T'_2) \text{ and } \ell' = \min(L' \setminus \{s(u)\})\}.$ 
7: else
8:   if  $\text{card}(T_1) \geq \mathcal{L}(\alpha_1)$  or  $\text{card}(T_2) \geq \mathcal{L}(\alpha_2)$  then
9:     let  $i \in \{1, 2\}$  such that  $\text{card}(T_i) \geq \mathcal{L}(\alpha_i).$ 
10:    else  $i \leftarrow 2.$ 
11:    let  $T'_{3-i} \subseteq \text{Cycl}[S]_{s,h'}^X(\beta)$  such that •  $\text{card}(T'_{3-i}) = \min(\text{card}(T_{3-i}), \mathcal{L}(\alpha_{3-i})),$ 
        •  $s(u) \in [T'_{3-i}]^b$  iff  $s(u) \in [T_{3-i}]^b.$ 
12:    let  $Q$  be a set of  $\text{card}(S)$  sets in  $\text{Cycl}[S]_{s,h'}^X(\beta) \setminus T'_{3-i}$  such that  $s(u) \in [Q]^b$  iff  $s(u) \in [S]^b.$ 
13:     $T'_i \leftarrow \text{Cycl}[S]_{s,h'}^X(\beta) \setminus (T'_{3-i} \cup Q).$ 
14:    let  $f : Q \rightarrow S$  be a bijection s.t. • for all  $L' \in Q$ , if  $s(u) \in L'$  then  $s(u) \in [f(L')]^b,$ 
        • for all  $P \in S$ , if  $s(u) \in [P]^b$  then  $s(u) \in f^{-1}(P).$ 
15:     $S' \leftarrow \begin{cases} (L'_1, L'_2) & \text{there is } L' \in Q \text{ such that:} \\ & \bullet L'_1 \cup L'_2 = L' \text{ and } \max(L'_1 \setminus \{s(u)\}) < \min(L'_2 \setminus \{s(u)\}), \\ & \bullet \text{for all } j \in \{1, 2\}, \min(\text{card}(L'_j), \alpha_j) = \min(\text{card}(\pi_j(f(L'))), \alpha_j), \\ & \bullet \text{for all } j \in \{1, 2\}, s(u) \in L'_j \text{ iff } s(u) \in \pi_j(f(L')). \end{cases}.$ 

```

Let us show that, following this strategy, T'_1 , T'_2 and S' are well-defined and satisfy the properties (1)–(6). The proof is divided in two cases, depending on whether $\text{card}(S) \geq \max(\alpha_1, \alpha_2)$.

case: $\text{card}(S) \geq \max(\alpha_1, \alpha_2)$. In this case, the strategy (lines 2–6) is the one in Lemma 5.40(III).

Therefore, the properties (1)–(6) follow as shown in Lemma 5.40(II).

case: $\text{card}(S) < \max(\alpha_1, \alpha_2)$. In this case, the definitions of T'_1 and T'_2 , as well as the definition of the auxiliary set Q , are quite similar to the ones given in lines 8–13 of the strategy in Lemma 5.40(III). The only difference lies on the fact that, in Lemma 5.40(III), we assume the sets $\text{Cycl}[s]_{s,h}^X(\beta)$ and $\text{Cycl}[s]_{s,h'}^X(\beta)$ to contain at least $\mathcal{L}(\alpha)$ elements. As this assumption is dropped in this proof, the strategy provided here starts (line 8–10) by checking whether “ $\text{card}(T'_1) \geq \mathcal{L}(\alpha_1)$ or $\text{card}(T'_2) \geq \mathcal{L}(\alpha_2)$ ” holds. If this is the case, then the definition of T'_1 , T'_2 and Q is exactly as the one provided in Lemma 5.40(III), which allows us to conclude that the properties (3) and (5) are satisfied. Moreover,

- (‡₁) T'_1 , T'_2 and Q partition $\uparrow\text{Cycl}[s]_{s,h'}^{X,\alpha}$,
- (‡₂) $\text{card}(Q) = \text{card}(S)$, and $s(u) \in [Q]^\flat$ iff $s(u) \in [S]^\flat$.

Otherwise, since $\text{card}(s) < \max(\alpha_1, \alpha_2)$, by (★) we conclude that

$$\text{card}(\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}) = \text{card}(S) + \text{card}(T_1) + \text{card}(T_2) < \max(\alpha_1, \alpha_2) + \mathcal{L}(\alpha_1) + \mathcal{L}(\alpha_2) < \mathcal{L}(\alpha).$$

From the equisatisfaction of the core formulae $\uparrow\text{loop}_{X,\alpha}^S \geq \beta$, this implies that $\uparrow\text{Cycl}[s]_{s,h'}^{X,\alpha}$ and $\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}$ have the same cardinality. Then, we can again derive that (3), (5), (‡₁) and (‡₂) are satisfied, using the same arguments as in Lemma 5.40(III).

The main difference between the strategy provided here and the one in Lemma 5.40(III) is given by the definition of S' (lines 14–15). First of all (line 14), we define a bijection f from Q and S so that it relates the two sets, one in Q and one in S , containing $s(u)$, if any. Thanks to (‡₂), f is well-defined. The idea behind the definition of S' is to split every set in $L' \in Q$ into two non-empty disjoint sets L'_1 and L'_2 , according to the pair of sets $f(L')$. In particular, we require the two following properties to hold:

- (‡₃) for all $j \in \{1, 2\}$, $\min(\text{card}(L'_j), \alpha_j) = \min(\text{card}(\pi_j(f(L'))), \alpha_j)$,
- (‡₄) for all $j \in \{1, 2\}$, $s(u) \in L'_j$ iff $s(u) \in \pi_j(f(L'))$.

These two constraints are satisfiable thanks to the properties of f and the fact that both L' and $[f(L')]^\flat$ contain more than α locations. In particular, we follow the construction:

- if $\text{card}(L_1) < \alpha_1$, it is sufficient to define L'_1 as a set of $\text{card}(L_1)$ locations in L' , so that $s(u)$ is included in L'_1 whenever $s(u) \in \pi_1(f(L'))$. This constraint on $s(u)$ can be satisfied as, by definition of f , $s(u) \in L'_1$ if and only if $s(u) \in [f(L')]^\flat$. Thanks to $\text{card}(L') > \alpha = \alpha_1 + \alpha_2$, this means that $L'_2 = L' \setminus L'_1$ contains more than α_2 locations. Similarly, from $\text{card}([f(L')]^\flat) > \alpha$, $\text{card}(\pi_2(f(L'))) > \alpha_2$. Therefore, the property (‡₃) is satisfied. Lastly, if $s(u) \in \pi_2(f(L'))$, then by disjointness $s(u) \notin \pi_1(f(L'))$ and from the properties of f and the definition of L'_1 we derive that $s(u) \in L'_2$. As the converse also holds, (‡₄) is satisfied.
- if $\text{card}(L_2) < \alpha_2$ then the construction carries out symmetrically to the previous case.
- if both $\text{card}(L_1) \geq \alpha_1$ and $\text{card}(L_2) \geq \alpha_2$, then it is sufficient to define L'_1 as a set of α_1 locations in L' , so that $s(u)$ is included in L'_1 whenever $s(u) \in \pi_1(f(L'))$. The satisfaction of both (‡₃) and (‡₄) follows as in the first case.

S' is defined as a set of pairs (L'_1, L'_2) constructed as above from a set $L' \in Q$. Notably, L'_1 and L'_2 must be uniquely defined, so that for every $L' \in Q$ there is exactly one pair $P \in S'$ such that $[P]^\flat = L'$. For this problem, the technical solution provided in line 15 is to require $\max(L'_1 \setminus \{s(u)\}) < \min(L'_2 \setminus \{s(u)\})$ to hold. Informally, this means that, apart for the location $s(u)$ which is deterministically assigned to either L'_1 or L'_2 , in the construction above we define L'_1 and L'_2 so that every location in L'_1 precedes the ones in L'_2 .

This does not break the satisfaction of (‡3) and (‡4), and allows us to uniquely define S' . Moreover, this also means that L'_1 and L'_2 are disjoint. Together with the fact that every pair in S is made of non-empty sets, by (‡3) we conclude that (1) holds. The property (‡3) also implies (4), since $[\pi_j(S)]^b = \bigcup_{L \in \pi_j(S)} L$, $[\pi_j(S')]^b = \bigcup_{L' \in \pi_j(S')} L'$, and every two sets in $\pi_j(S)$ (resp. $\pi_j(S')$) are disjoint. Similarly, the property (‡3) implies (6). Lastly, by definition of S' we have $\{[P]^b \mid P \in S'\} = Q$. As Q , T_1 and T_2 partition $\uparrow\text{Cycl}[S]_{s,h'}^{X,\alpha}$, this implies the satisfaction of (2). We conclude that the properties (1)–(6) are satisfied.

We rely on the sets T'_1 , T'_2 and S' to define the heaps h'_1 and h'_2 such that $(s, h_1) \approx_{X,\alpha_1}^S (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^S (s, h'_2)$ (as required by the hop relation $\leftrightarrow_{X,\alpha}^S$). First, let us define the two heaps $\widehat{h_1} \stackrel{\text{def}}{=} h_1 \setminus \{(\ell, \ell') \in h_1 \mid \ell \in [T_1 \cup \pi_1(S)]^b\}$ and $\widehat{h_2} \stackrel{\text{def}}{=} h_2 \setminus \{(\ell, \ell') \in h_2 \mid \ell \in [T_2 \cup \pi_2(S)]^b\}$. By definition of T_1 , T_2 and S , $\text{dom}(\widehat{h_1}) \cap \uparrow\text{Cycl}[S]_{s,h}^{X,\alpha}$ and $\text{dom}(\widehat{h_2}) \cap \uparrow\text{Cycl}[S]_{s,h}^{X,\alpha}$ are both empty. Moreover, by $h = h_1 + h_2$, we have that

$$h = \widehat{h_1} + \widehat{h_2} + \{(\ell, \ell') \in h \mid \ell \in [\uparrow\text{Cycl}[S]_{s,h}^{X,\alpha}]^b\}.$$

From the hypothesis $h \setminus \{(\ell, \ell') \in h \mid \ell \in [\uparrow\text{Cycl}[S]_{s,h}^{X,\alpha}]^b\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in [\uparrow\text{Cycl}[S]_{s,h'}^{X,\alpha}]^b\}$ we derive $\widehat{h_1} + \widehat{h_2} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in [\uparrow\text{Cycl}[S]_{s,h'}^{X,\alpha}]^b\}$. We define the heaps h'_1 and h'_2 as:

$$h'_1 \stackrel{\text{def}}{=} \widehat{h_1} + \{(\ell, \ell') \in h' \mid \ell \in [T'_1 \cup \pi_1(S')]^b\}, \quad h'_2 \stackrel{\text{def}}{=} \widehat{h_2} + \{(\ell, \ell') \in h' \mid \ell \in [T'_2 \cup \pi_2(S')]^b\}.$$

From (2), the heaps $\{(\ell, \ell') \in h' \mid \ell \in [T'_1 \cup \pi_1(S')]^b\}$ and $\{(\ell, \ell') \in h' \mid \ell \in [T'_2 \cup \pi_2(S')]^b\}$ are disjoint, and their union is $\{(\ell, \ell') \in h' \mid \ell \in [\uparrow\text{Cycl}[S]_{s,h'}^{X,\alpha}]^b\}$. Thus, h'_1 and h'_2 are well-defined, they are disjoint, and $h' = h'_1 + h'_2$. As in Lemma 5.40(III), we now discuss seven properties of h'_1 and h'_2 . Let $j \in \{1, 2\}$.

- A. For all $t \in T[S]^X$, $\llbracket t \rrbracket_{s,h_j}^X$ and $\llbracket t \rrbracket_{s,h'_j}^X$ are equidefined. When defined, they are equal.

Proof of (A). We notice that the set $\uparrow\text{Cycl}[S]_{s,h}^{X,\alpha}$ (resp. $\uparrow\text{Cycl}[S]_{s,h'}^{X,\alpha}$) enjoys the property of the set L of Lemma C.1, with respect to the memory states $\widehat{h_j}$ and h_j (resp. h'_j). Thus, the result follows from Lemma C.1(A), as shown in the proof of Lemma 5.40(I).

- B. For every $t \in T[S]^X$,

- (a) $\text{sby}_{s,h_j}^X(t)$ and $\text{sby}_{s,h'_j}^X(t)$ are equidefined. When defined, they are equal,
- (b) $\text{Path}[S]_{s,h_j}^X(t) = \text{Path}[S]_{s,h'_j}^X(t)$,
- (c) let $\delta \in [1, \text{card}(\text{Path}[S]_{s,h_j}^X(t))]$. $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) = s(u)$ iff $h'_j^\delta(\llbracket t \rrbracket_{s,h'_j}^X) = s(u)$.

Proof of (B). Similarly to (A), it follows directly from Lemma C.1(B).

- C. For every $x \in X$, $\text{Pred}[S]_{s,h_j}^X(x) = \text{Pred}[S]_{s,h'_j}^X(x)$.

Proof of (C). The proof of this statement is as in Lemma 5.40(III)(case (C)).

- D. For every $\beta \in [1, \alpha_j]$, $\text{Cycl}[S]_{s,h_j}^X(\beta) = \text{Cycl}[S]_{s,h'_j}^X(\beta)$

Proof of (D). The proof of this statement is as in Lemma 5.40(III)(case (D)). In particular, notice that it corresponds to the statement (f), since both $\text{Cycl}[S]_{s,h_j}^X(\beta) \cap T_j$ and $\text{Cycl}[S]_{s,h'_j}^X(\beta) \cap T'_j$ are empty.

- E. (d) $\min(\text{card}(\uparrow\text{Cycl}[S]_{s,h_j}^{X,\alpha_j}), \mathcal{L}(\alpha_j)) = \min(\text{card}(\uparrow\text{Cycl}[S]_{s,h'_j}^{X,\alpha_j}), \mathcal{L}(\alpha_j))$,
- (e) $s(u) \in [\uparrow\text{Cycl}[S]_{s,h_j}^{X,\alpha_j}]^b$ if and only if $s(u) \in [\uparrow\text{Cycl}[S]_{s,h'_j}^{X,\alpha_j}]^b$.

The proofs of these two statements rely on the following equality and inclusions:

- (f) $\uparrow\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{x},\alpha_j} \setminus T_j = \uparrow\text{Cycl}[\mathcal{S}]_{s,h'_j}^{\mathbf{x},\alpha_j} \setminus T'_j,$
- (g) $T_j \subseteq \uparrow\text{Cycl}[\mathcal{S}]_{s,h_j}^{\mathbf{x},\alpha_j}$ and $T'_j \subseteq \uparrow\text{Cycl}[\mathcal{S}]_{s,h'_j}^{\mathbf{x},\alpha_j}.$

The proof of (f) (resp. (g)) is analogous to the one given to the homonymous statement of Lemma 5.40(III). Then, the proof of (E) follows analogously to Lemma 5.40(III)(case (D)).

- F. (j) $\min(\text{card}(\text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{x},\alpha_j}), \mathcal{R}(\alpha_j)) = \min(\text{card}(\text{Rem}[\mathcal{S}]_{s,h'_j}^{\mathbf{x},\alpha_j}), \mathcal{R}(\alpha_j)),$
- (k) $s(u) \in \text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{x},\alpha_j}$ if and only if $s(u) \in \text{Rem}[\mathcal{S}]_{s,h'_j}^{\mathbf{x},\alpha_j}.$

The proofs of these two statements rely on the following equality and inclusions:

- (l) $\text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{x},\alpha_j} \setminus [\pi_j(S)]^\flat = \text{Rem}[\mathcal{S}]_{s,h'_j}^{\mathbf{x},\alpha_j} \setminus [\pi_j(S')]^\flat,$
- (m) $[\pi_j(S)]^\flat \subseteq \text{Rem}[\mathcal{S}]_{s,h_j}^{\mathbf{x},\alpha_j}$ and $[\pi_j(S')]^\flat \subseteq \text{Rem}[\mathcal{S}]_{s,h'_j}^{\mathbf{x},\alpha_j}.$

The statement (l) (resp. (m)) is proved analogously to the statement (p) (resp. (q)) of Lemma 5.40(III). Then, the proof of (F) follows similarly to Lemma 5.40(III)(case (F)).

The properties (A)–(F) lead directly to $(s, h_j) \approx_{\mathbf{x},\alpha_j}^{\mathcal{S}} (s', h'_j)$, with the same case analysis provided at the end of the proof of Lemma 5.39. Therefore, $(s, h) \leftrightarrow_{\mathbf{x},\alpha}^{\mathcal{S}} (s, h')$. \square

Lemma 5.40(V). Let (s, h) and (s, h') be two memory states such that $(s, h) \approx_{\mathbf{x},\alpha}^{\mathcal{S}} (s, h')$. and $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[\mathcal{S}]_{s,h}^{\mathbf{x},\alpha}\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{S}]_{s,h'}^{\mathbf{x},\alpha}\}$. Then, $(s, h) \leftrightarrow_{\mathbf{x},\alpha}^{\mathcal{S}} (s, h')$.

Proof. Consider two heaps h_1 and h_2 , $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$ such that $h = h_1 + h_2$ and $\alpha = \alpha_1 + \alpha_2$. Notice that this requires α to be at least two, otherwise the lemma trivially holds. We recall that the integer β in formulae of $\text{Core}[\mathcal{S}](\mathbf{x}, \alpha)$ of the form $\text{rem}_{\mathbf{x},\alpha}^{\mathcal{S}} \geq \beta$ ranges over $[1, \alpha]$. If $\text{card}(\text{Rem}[\mathcal{S}]_{s,h}^{\mathbf{x},\alpha}) < \alpha$ then the lemma holds directly from Lemma 5.39. Indeed, in this case, from the equisatisfaction of the core formulae of the form $\text{rem}_{\mathbf{x},\alpha}^{\mathcal{S}} \geq \beta$ we conclude that $\text{card}(\text{Rem}[\mathcal{S}]_{s,h}^{\mathbf{x},\alpha}) = \text{card}(\text{Rem}[\mathcal{S}]_{s,h'}^{\mathbf{x},\alpha})$. By definition of h and h' , we conclude that $((s, h), (s, h')) \in (\cap_{\alpha' \geq 1} \approx_{\mathbf{x},\alpha'}^{\mathcal{S}})$, which allows us to apply Lemma 5.39. Therefore, in the following we assume $\text{card}(\text{Rem}[\mathcal{S}]_{s,h}^{\mathbf{x},\alpha}) \geq \alpha$, which implies $\text{card}(\text{Rem}[\mathcal{S}]_{s,h'}^{\mathbf{x},\alpha}) \geq \alpha$ again from the equisatisfaction of the core formulae $\text{rem}_{\mathbf{x},\alpha}^{\mathcal{S}} \geq \beta$.

We define the following two sets T_1 , T_2 :

$$T_1 = \{\ell \in \text{Rem}[\mathcal{S}]_{s,h}^{\mathbf{x},\alpha} \mid \ell \in \text{dom}(h_1)\}, \quad T_2 = \{\ell \in \text{Rem}[\mathcal{S}]_{s,h}^{\mathbf{x},\alpha} \mid \ell \in \text{dom}(h_2)\}.$$

Given $j \in [1, 2]$, T_j contains every location of $\text{Rem}[\mathcal{S}]_{s,h}^{\mathbf{x},\alpha}$ that is a memory cell of h_j . Moreover, T_1 , T_2 are mutually disjoint sets, and their union is $\text{Rem}[\mathcal{S}]_{s,h}^{\mathbf{x},\alpha}$. As a first step of the proof, we aim at defining three similar sets T'_1 , T'_2 with respect to $\text{Rem}[\mathcal{S}]_{s,h'}^{\mathbf{x},\alpha}$. These sets should also satisfy cardinality constraints depending on α_j , as well as constraints involving the location $s(u)$. More precisely:

1. T'_1 , T'_2 are mutually disjoint. Their union is $\text{Rem}[\mathcal{S}]_{s,h'}^{\mathbf{x},\alpha}$,
2. for all $j \in \{1, 2\}$, $\min(\text{card}(T_j), \alpha_j) = \min(\text{card}(T'_j), \alpha_j)$,
3. for all $j \in \{1, 2\}$, $s(u) \in T_j$ if and only if $s(u) \in T'_j$.

The definition of T'_1 and T'_2 follows the strategy below.

if $\text{card}(T_1) < \alpha_1$ **then**

let T'_1 be a set of $\text{card}(T_1)$ locations in $\text{Rem}[\mathcal{S}]_{s,h'}^{\mathbf{x},\alpha}$ such that $s(u) \in T_1$ iff $s(u) \in T'_1$.

$T'_2 \leftarrow \text{Rem}[\mathcal{S}]_{s,h'}^{\mathbf{x},\alpha} \setminus T'_1$.

else if $\text{card}(T_2) < \alpha_2$ **then**

let T'_2 be a set of $\text{card}(T_2)$ locations in $\text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}$ such that $s(u) \in T_2$ iff $s(u) \in T'_2$.
 $T'_1 \leftarrow \text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha} \setminus T'_2$.

else (*i.e.* $\text{card}(T_1) \geq \alpha_1$ and $\text{card}(T_2) \geq \alpha_2$)

let T'_1 be a set of α_1 locations in $\text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}$ such that $s(u) \in T_1$ iff $s(u) \in T'_1$.
 $T'_2 \leftarrow \text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha} \setminus T'_1$.

Since we are assuming that both $\text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}$ and $\text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}$ contain at least $\alpha = \alpha_1 + \alpha_2$ elements, and we have $s(u) \in \text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}$ iff $s(u) \in \text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}$ (from the equisatisfaction of the core formula $\text{urem}_{X,\alpha}^S$), we conclude that the sets T'_1 and T'_2 are well-defined. Moreover, their definition is such that $T_1 \cap T_2 = \emptyset$ and $T_1 \cup T_2 = \text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}$. So, the property (1) is satisfied. Let us show that the same holds true for the properties (2) and (3). The proof is divided in three cases, according to the strategy.

case: $\text{card}(T_1) < \alpha_1$. From the first case of the strategy, we have $\text{card}(T_1) = \text{card}(T'_1)$. By definition of T_1 and T_2 we have $\text{card}(\text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}) = \text{card}(T_1) + \text{card}(T_2)$. Similarly, from (1), $\text{card}(\text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}) = \text{card}(T'_1) + \text{card}(T'_2)$. Since $\text{card}(\text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha})$ and $\text{card}(\text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha})$ contain at least α locations, where $\alpha = \alpha_1 + \alpha_2 \geq \text{card}(T_1) + \alpha_2$, this allows us to conclude that T_2 and T'_2 contain at least α_2 locations. Thus, (2) is satisfied. Again from the definition of T'_1 , we have $s(u) \in T_1$ if and only if $s(u) \in T'_1$. Therefore, in order to conclude that (3) it is sufficient to show that $s(u) \in T_2$ if and only if $s(u) \in T'_2$. This is quite obvious: for the left-to-right direction, if $s(u) \in T_2$ then $s(u) \notin T_1$ and $s(u) \in \text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}$. By definition of T'_1 we derive $s(u) \notin T'_1$, whereas from the equisatisfaction of the core formula $u \in \text{rem}_{X,\alpha}^S$ we have $s(u) \in \text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}$. As $T'_1 \cup T'_2 = \text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}$, we conclude that $s(u) \in T'_2$. The other direction is proved symmetrically.

case: $\text{card}(T_2) < \alpha_2$. Analogous to the previous case, by swapping the indices 1 and 2 and considering the second case of the strategy.

case: $\text{card}(T_1) \geq \alpha_1$ and $\text{card}(T_2) \geq \alpha_2$. From the third case of the strategy, $\text{card}(T'_1) = \alpha_1$. Similarly to the first case, this allows us to conclude that T'_2 has at least α_2 locations. Thus, (2) is satisfied. The proof of (3) is equal to the one given in the first case.

We rely on T'_1 and T'_2 in order to define the heaps h'_1 and h'_2 such that $(s, h_1) \approx_{X,\alpha_1}^S (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^S (s, h'_2)$ (as required by the hop relation $\leftrightarrow_{X,\alpha}^W$). First, let us define the two heaps $\widehat{h}_1 \stackrel{\text{def}}{=} h_1 \setminus \{(\ell, \ell') \in h_1 \mid \ell \in T_1\}$ and $\widehat{h}_2 \stackrel{\text{def}}{=} h_2 \setminus \{(\ell, \ell') \in h_2 \mid \ell \in T_2\}$. By definition of T_1 and T_2 , $\text{dom}(\widehat{h}_1) \cap \text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}$ and $\text{dom}(\widehat{h}_2) \cap \text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}$ are both empty. Moreover, by $h = h_1 + h_2$,

$$h = \widehat{h}_1 + \widehat{h}_2 + \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}\}.$$

From the hypothesis $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}\} = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}\}$ we derive $\widehat{h}_1 + \widehat{h}_2 = h' \setminus \{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}\}$. We define the heaps h'_1 and h'_2 as:

$$h'_1 \stackrel{\text{def}}{=} \widehat{h}_1 + \{(\ell, \ell') \in h' \mid \ell \in T'_1\}, \quad h'_2 \stackrel{\text{def}}{=} \widehat{h}_2 + \{(\ell, \ell') \in h' \mid \ell \in T'_2\}.$$

From (1), the heaps $\{(\ell, \ell') \in h' \mid \ell \in T'_1\}$ and $\{(\ell, \ell') \in h' \mid \ell \in T'_2\}$ are disjoint, and their union is $\{(\ell, \ell') \in h' \mid \ell \in \text{Rem}[\mathcal{S}]_{s,h'}^{X,\alpha}\}$. Thus, h'_1 and h'_2 are well-defined, they are disjoint, and $h' = h'_1 + h'_2$. We now discuss seven properties of h'_1 and h'_2 which directly lead to $(s, h_1) \approx_{X,\alpha_1}^S (s, h'_1)$ and $(s, h_2) \approx_{X,\alpha_2}^S (s, h'_2)$, as done for Lemma 5.40(III). Let $j \in \{1, 2\}$.

- A. For all $t \in T[s]^X$, $\llbracket t \rrbracket_{s,h_j}^X$ and $\llbracket t \rrbracket_{s,h'_j}^X$ are equidefined. When defined, they are equal.

Proof of (A). We notice that the set $\text{Rem}[s]_{s,h}^{X,\alpha}$ (resp. $\text{Rem}[s]_{s,h'}^{X,\alpha}$) enjoys the property of the set L of Lemma C.1, with respect to the memory states \widehat{h}_j and h_j (resp. h'_j). Thus, the result follows from Lemma C.1(A), as shown in the proof of Lemma 5.40(I).

- B. For every $t \in T[s]^X$,

- (a) $\text{sby}_{s,h_j}^X(t)$ and $\text{sby}_{s,h'_j}^X(t)$ are equidefined. When defined, they are equal,
- (b) $\text{Path}[s]_{s,h_j}^X(t) = \text{Path}[s]_{s,h'_j}^X(t)$,
- (c) let $\delta \in [1, \text{card}(\text{Path}[s]_{s,h_j}^X(t))]$. $h_j^\delta(\llbracket t \rrbracket_{s,h_j}^X) = s(u)$ iff $h_j'^\delta(\llbracket t \rrbracket_{s,h'_j}^X) = s(u)$.

Proof of (B). Similarly to (A), it follows directly from Lemma C.1(B).

- C. For every $x \in X$, $\text{Pred}[s]_{s,h_j}^X(x) = \text{Pred}[s]_{s,h'_j}^X(x)$.

Proof of (C). (\subseteq): Let $\ell \in \text{Pred}[s]_{s,h_j}^X(x)$, and so $h_j(\ell) = s(x)$ and, in h_j , ℓ is not reached by any location corresponding to program variables in X . By Lemma C.1(O), h'_j does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. As $h_j \subseteq h$ we have $h(\ell) = s(x)$. By definition $\ell \notin \text{Rem}[s]_{s,h}^{X,\alpha}$, which implies $\ell \in \text{dom}(\widehat{h}_j)$ directly from the definition of \widehat{h}_j and the fact that $\ell \in \text{dom}(h_j)$. By $\widehat{h}_j \subseteq h_j$, $\widehat{h}_j(\ell) = s(x)$. Since $\widehat{h}_j \subseteq h'_j$, $h'_j(\ell) = s(x)$. Thus, $\ell \in \text{Pred}[s]_{s,h'_j}^X(x)$.

(\supseteq): Symmetrical to the other direction.

- D. For every $\beta \in [1, \alpha_j]$, $\text{Cycl}[s]_{s,h_j}^X(\beta) = \text{Cycl}[s]_{s,h'_j}^X(\beta)$.

Proof of (D). (\Rightarrow): Consider a set $L \in \text{Cycl}[s]_{s,h_j}^X(\beta)$. By definition, L describes an unlabelled cycle in h_j , of length β . By definition, $L \cap \text{Rem}[s]_{s,h}^{X,\alpha} = \emptyset$. Therefore, $L \cap T_j = \emptyset$, which implies $L \subseteq \text{dom}(\widehat{h}_j)$ by definition of \widehat{h}_j . As $\widehat{h}_j \subseteq h_j$, we conclude that L describes a cycle in \widehat{h}_j . As $\widehat{h}_j \subseteq h'_j$, we conclude that L describes a cycle in h'_j . From (A) this cycle does not contain labelled locations. Thus, $L \in \text{Cycl}[s]_{s,h'_j}^X(\beta)$.

(\Leftarrow): Symmetrical to the other direction.

- E. $\uparrow\text{Cycl}[s]_{s,h_j}^{X,\alpha_j} = \uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}$.

The proof of this case is analogous to the one of (D).

- F. (d) $\min(\text{Rem}[s]_{s,h_j}^{X,\alpha_j}, \alpha_j) = \min(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j}, \alpha_j)$,
- (e) $s(u) \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ if and only if $s(u) \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

We start by showing the following equivalence and inclusions:

- (f) $\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j = \text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus T'_j$,
- (g) $T_j \subseteq \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ and $T'_j \subseteq \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

Proof of (f). (\subseteq): Suppose $\ell \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j$. By definition of $\text{Rem}[s]_{s,h_j}^{X,\alpha_j}$ we have

h. $\ell \in \text{dom}(h_j)$

- i. ℓ does not belong to $\text{Lab}[s]_{s,h_j}^X$, $\text{Pred}[s]_{s,h_j}^X(x)$, $\text{Path}[s]_{s,h_j}^X(\ell)$, $\text{Cycl}[s]_{s,h_j}^X(\beta)$ and $\uparrow\text{Cycl}[s]_{s,h_j}^{X,\alpha_j}$, for every $x \in X$, $\ell \in \text{Lab}[s]_{s,h_j}^X$ and $\beta \in [1, \alpha_j]$.

From (h) and $\ell \notin T_j$, we derive $\ell \in \text{dom}(\widehat{h_j})$. As $\widehat{h_j} \subseteq h'_j$, $\ell \in \text{dom}(h'_j)$. From (i), together with (A)–(E), ℓ does not belong to $\text{Lab}[s]_{s,h'_j}^X$, $\text{Pred}[s]_{s,h'_j}^X(x)$, $\text{Path}[s]_{s,h'_j}^X(\ell)$, $\text{Cycl}[s]_{s,h'_j}^X(\beta)$ and $\uparrow\text{Cycl}[s]_{s,h'_j}^{X,\alpha_j}$, for every $x \in X$, $\ell \in \text{Lab}[s]_{s,h_j}^X$ and $\beta \in [1, \alpha_j]$. Hence, $\ell \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$.

(\supseteq): Symmetrical to the other direction.

Proof of (g). We prove the inclusion $T_j \subseteq \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$. Suppose $\ell \in T_j$. By definition, $\ell \in \text{Rem}[s]_{s,h}^{X,\alpha}$, which means that:

- * for every $x \in X$, $h(\ell) \neq s(x)$,
- * for every $x \in X$ and $\delta \geq 0$, $h^\delta(s(x)) \neq \ell$,
- * for every $\delta \geq 1$, $h^\delta(\ell) \neq \ell$.

As $h_j \subseteq h$, these three properties carry over to h_j (they are monotonous under sub-heaps). Lastly, $\ell \in T_j$ implies $\ell \in \text{dom}(h_j)$, which allows us to derive $\ell \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j}$.

The proof of the inclusion $T'_j \subseteq \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$ is analogous.

Proof of (F). First, we prove that (d) holds. The property (2) of the constructions states that $\min(\text{card}(T_j), \alpha_j) = \min(\text{card}(T'_j), \alpha_j)$. By using the fact that for all $a, b, c, d \in \mathbb{N}$, $\min(a, d) = \min(b, d)$ implies $\min(a + c, d) = \min(b + c, d)$,

$$\begin{aligned} & \min(\text{card}(T_j) + \text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j), \alpha_j) \\ &= \min(\text{card}(T'_j) + \text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus T_j), \alpha_j). \end{aligned}$$

From (f), $\text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j) = \text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus T'_j)$, and so

$$\begin{aligned} & \min(\text{card}(T_j) + \text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j), \alpha_j) \\ &= \min(\text{card}(T'_j) + \text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus T'_j), \alpha_j). \end{aligned} \tag{\dagger}$$

By (g), we have $\text{Rem}[s]_{s,h_j}^{X,\alpha_j} = (\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j) \cup T_j$ and so

$$\text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j}) = \text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j) + \text{card}(T_j).$$

Similarly (again from (g)),

$$\text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j}) = \text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j} \setminus T'_j) + \text{card}(T'_j).$$

By (†) we have (d), i.e. $\min(\text{card}(\text{Rem}[s]_{s,h_j}^{X,\alpha_j}), \alpha_j) = \min(\text{card}(\text{Rem}[s]_{s,h'_j}^{X,\alpha_j}), \alpha_j)$.

Let us prove (e). For the left-to-right direction, suppose $s(u) \in \text{Rem}[s]_{s,h_j}^{X,x}$. By (g), we have either $s(u) \in \text{Rem}[s]_{s,h_j}^{X,\alpha_j} \setminus T_j$ or $s(u) \in T_j$. In the former case, directly from (f), we conclude that $s(u) \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$. In the latter case, from the property (3) of the construction, we have $s(u) \in T'_j$, which in turn implies $s(u) \in \text{Rem}[s]_{s,h'_j}^{X,\alpha_j}$ by (g). The right-to-left direction is proved symmetrically.

The properties (A)–(F) lead directly to $(s, h_j) \approx_{X,\alpha_j}^S (s', h'_j)$, with the same case analysis provided at the end of the proof of Lemma 5.39. Therefore, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h')$. \square

Proof of Lemma 5.41.

Lemma 5.41 (s : *-simulation). $\approx_{X,\alpha}^S \subseteq \leftrightarrow_{X,\alpha}^S$.

Proof. This proof follows rather closely the proof of the $*$ -simulation property given for the core formulae of the weak fragment in Lemma 5.20. Let us consider (s, h) and (s', h') such that $(s, h) \approx_{\mathbf{X}, \alpha}^S (s', h')$. We rely on Lemma 5.39 and Lemma 5.40 to build a chain of hops as the one below, leading to the result by transitivity of $\leftrightarrow_{\mathbf{X}, \alpha}^S$,

$$(s, h) = (s_1, h_1) \leftrightarrow_{\mathbf{X}, \alpha}^S (s_2, h_2) \leftrightarrow_{\mathbf{X}, \alpha}^S \dots \leftrightarrow_{\mathbf{X}, \alpha}^S (s_{k-1}, h_{k-1}) \leftrightarrow_{\mathbf{X}, \alpha}^S (s_k, h_k) = (s', h').$$

The proof is by induction on the cardinality of the set $[(s, h) \#_{\mathbf{X}, \alpha} (s', h')]$ defined as follows:

$$\left\{ (S, T) \in \left\{ \begin{array}{l} (\text{Pred}[s]_{s,h}^{\mathbf{X}}(\mathbf{x}), \text{Pred}[s']_{s',h'}^{\mathbf{X}}(\mathbf{x})), \\ (\text{Path}[s]_{s,h}^{\mathbf{X}}(\mathbf{t}), \text{Path}[s']_{s',h'}^{\mathbf{X}}(\mathbf{t})), \\ (\text{Rem}[s]_{s,h}^{\mathbf{X}, \alpha}, \text{Rem}[s']_{s',h'}^{\mathbf{X}, \alpha}), \\ (\text{Cycl}[s]_{s,h}^{\mathbf{X}}(\beta), \text{Cycl}[s']_{s',h'}^{\mathbf{X}}(\beta)) \end{array} \middle| \begin{array}{l} \mathbf{x} \in \mathbf{X} \\ \mathbf{t} \in \mathbf{T}[s]^{\mathbf{X}} \\ \beta \in [1, \alpha] \end{array} \right\} \middle| \text{card}(S) \neq \text{card}(T) \right\}$$

Intuitively, this set contains pairs of predecessors sets, paths sets, bounded loops or remainder sets that have different cardinalities in the two memory states. Notice that the sets of unbounded loops, i.e. $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{X}, \alpha}$ and $\uparrow\text{Cycl}[s']_{s',h'}^{\mathbf{X}, \alpha}$, are excluded, as we treat them separately inside the base case of the induction. We build the chain of hops so that for every intermediate memory state in it is obtained from the previous one by modifying the heap in a way that strictly reduces the number of these pairs, always with respect to the last memory state of the chain, i.e. (s', h') .

base case: $[(s, h) \#_{\mathbf{X}, \alpha} (s', h')] = 0$. In this case, (s, h) and (s', h') agree on the cardinality of every set of the partition, with (possibly) the exception of $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{X}, \alpha}$ and $\uparrow\text{Cycl}[s']_{s',h'}^{\mathbf{X}, \alpha}$. The idea is build a chain of hops featuring (s, h) and (s', h') , such that every two memory states in the chain differ on the sets of unlabelled and unbounded cycles, e.g. $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{X}, \alpha}$ and $\uparrow\text{Cycl}[s']_{s',h'}^{\mathbf{X}, \alpha}$. This is done by updating (s, h) and (s', h') in order to reach two memory states in the relation $(\cap_{\alpha' \geq 1} \approx_{\mathbf{X}, \alpha'}^S)$, which allows us to apply Lemma 5.39. We divide the process of building the chain of hops in three steps.

step 1: Let us focus on (s, h) . Let $n = \text{card}(\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{X}, \alpha})$, and let us assume that $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{X}, \alpha} = \{L_1, \dots, L_n\}$. L_1, \dots, L_n are mutually disjoint sets describing unlabelled cycles of (s, h) , of length at least $\alpha + 1$. Without loss of generality, we assume that if $s(u)$ belongs to a set in $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{X}, \alpha}$, then this set is L_1 . Let $m = \min(n, \mathcal{L}(\alpha))$, where $\mathcal{L}(\alpha) = \frac{1}{2}\alpha(\alpha + 3) - 1$ is the upper bound given to β for the core formulae of $\text{Core}[s](\mathbf{X}, \alpha)$ of the form $\uparrow\text{loop}_{\mathbf{X}, \alpha}^S \geq \beta$. We introduce m sets of locations L'_1, \dots, L'_m such that

1. for all $j \in [1, m]$, $L'_j = \{\ell_0^j, \dots, \ell_\alpha^j\} \subseteq L_j$. So, $\text{card}(L'_j) = \alpha + 1$,
2. $s(u) \in L'_1$ if and only if $s(u) \in L_1$.

Informally, L'_j is obtained from L_j by removing some locations in order to obtain a set of cardinality $\alpha + 1$. If the location $s(u)$ belongs to L_1 , then it is kept in L'_1 . As L_1, \dots, L_n are mutually disjoint, so are L'_1, \dots, L'_m . Given $j \in [1, m]$, we consider a heap $\tilde{h}_j \stackrel{\text{def}}{=} \{\ell_i^j \mapsto \ell_{(i+1) \bmod (\alpha+1)}^j \mid i \in [0, \alpha]\}$. Essentially, \tilde{h}_j witnesses a cycle of length $\alpha + 1$, and nothing else. That is, $L'_j = \text{dom}(\tilde{h}_j)$ is a minimal set describing this cycle. The heaps $h, \tilde{h}_1, \dots, \tilde{h}_m$ are mutually disjoint. We define \hat{h} as the heap $h \setminus \{(\ell, \ell') \in h \mid \ell \in [\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{X}, \alpha}]^b\}$. \hat{h} is disjoint from every heap \tilde{h}_j ($j \in [1, m]$), which allows us to define the heap

$$h_1 = \hat{h} + \tilde{h}_1 + \dots + \tilde{h}_m.$$

One can see h_1 as a heap obtained from h by shrinking all the cycles in $\uparrow\text{Cycl}[s]_{s,h}^{\mathbf{X}, \alpha}$ to cardinality $\alpha + 1$, while keeping $s(u)$ in one of these cycles, whenever $s(u) \in$

$[\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}]^b$. Moreover, if $\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}$ contains more than $\mathcal{L}(\alpha)$ cycles, then the cycles which are in excess are removed. The following properties are trivially satisfied:

3. $\text{dom}(\widehat{h}) \subseteq \text{dom}(h_1) \subseteq \text{dom}(h)$,
4. $\text{dom}(\widehat{h}) = \text{dom}(h) \setminus [\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}]^b$,
5. For every $\ell \in \text{dom}(\widehat{h})$, $h_1(\ell) = \widehat{h}(\ell) = h_1(\ell)$.

We prove that $(s, h) \approx_{X,\alpha}^S (s, h_1)$. This is done by showing the properties (A)–(F) already discussed in Lemma 5.39 and similar lemmata.

- A. For all $t \in T[s]^X$, $\llbracket t \rrbracket_{s,h}^X$ and $\llbracket t \rrbracket_{s,h_1}^X$ are equidefined. When defined, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h_1}^X$.

Proof of (A). We notice that the set $\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}$ (resp. $\bigcup_{j \in [1,m]} L'_j$) enjoys the property of the set L of Lemma C.1, with respect to the memory states \widehat{h} and h (resp. h_1). Therefore, (A) follows directly from Lemma C.1(A), as shown in the proof of Lemma 5.40(I).

- B. For every $t \in T[s]^X$,

- (a) $\text{sby}_{s,h}^X(t)$ and $\text{sby}_{s,h_1}^X(t)$ are equidefined. If defined, they are equal,
- (b) $\text{Path}[s]_{s,h}^X(t) = \text{Path}[s]_{s,h_1}^X(t)$,
- (c) let $\delta \in [1, \text{card}(\text{Path}[s]_{s,h}^X(t))]$. $h^\delta(\llbracket t \rrbracket_{s,h}^X) = s(u)$ iff $h_1^\delta(\llbracket t \rrbracket_{s,h_1}^X) = s(u)$.

Proof of (B). Similarly to (A), it follows directly from Lemma C.1(B).

- C. For every $x \in X$, $\text{Pred}[s]_{s,h}^X(x) = \text{Pred}[s]_{s,h_1}^X(x)$.

Proof of (C). (\Rightarrow): Suppose $\ell \in \text{Pred}[s]_{s,h}^X(x)$. By definition, $h(\ell) = s(x)$ and h does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. By Lemma C.1(O), h_1 does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. Moreover, $\ell \in \text{Pred}[s]_{s,h}^X(x)$ means that $\ell \notin \uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}$, which in turn implies $\ell \in \text{dom}(\widehat{h})$. From (5), $h_1(\ell) = s(x)$. Thus, $\ell \in \text{Pred}[s]_{s,h_1}^X(x)$.

(\Leftarrow): Symmetrical to the other direction.

- D. For every $\beta \in [1, \alpha]$, $\text{Cycl}[s]_{s,h}^X(\beta) = \text{Cycl}[s]_{s,h_1}^X(\beta)$.

Proof of (D). (\Rightarrow): Let $L \in \text{Cycl}[s]_{s,h}^X(\beta)$. So, L is a minimal set describing an unlabelled cycle in h , of length β . Since $L \notin \uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}$ and $L \subseteq \text{dom}(h)$, by (4) we have $L \subseteq \text{dom}(\widehat{h})$. By (5), L is a minimal set describing a cycle in h_1 . From (A), $L \cap \text{Lab}[s]_{s,h_1}^X = \emptyset$. So $L \in \text{Cycl}[s]_{s,h_1}^X(\beta)$.

(\Leftarrow): Symmetrical to the other direction.

- E. (d) $\min(\text{card}(\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}), \mathcal{L}(\alpha)) = \min(\text{card}(\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}), \mathcal{L}(\alpha))$.

(e) $s(u) \in [\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}]^b$ if and only if $s(u) \in [\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}]^b$.

(f) For every $L \in \uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}$, $\text{card}(L) = \alpha + 1$.

Proof of (E). The three statements follow directly from (A) and the definition of $\widehat{h}_1, \dots, \widehat{h}_m$. Precisely, $\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha} = \{\widehat{L}'_1, \dots, \widehat{L}'_m\}$.

- F. $\text{Rem}[s]_{s,h}^{X,\alpha} = \text{Rem}[s]_{s,h_1}^{X,\alpha}$.

Proof of (F). (\Rightarrow): Suppose $\ell \in \text{Rem}[s]_{s,h}^{X,\alpha}$. Thus, $\ell \in \text{dom}(h)$ and moreover

$$\begin{aligned} \ell &\notin \text{Lab}[s]_{s,h}^X, & \ell &\notin \text{Path}[s]_{s,h}^X(t), & \ell &\notin \text{Pred}[s]_{s,h}^X(x), \\ \ell &\notin [\text{Cycl}[s]_{s,h}^X(\beta)]^b, & \ell &\notin [\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}]^b, \end{aligned}$$

where $t \in T[s]^X$, $x \in X$ and $\beta \in [1, \alpha]$. From (A)–(D), we have

$$\ell \notin \text{Lab}[s]_{s,h_1}^X, \quad \ell \notin \text{Path}[s]_{s,h_1}^X(t), \quad \ell \notin \text{Pred}[s]_{s,h_1}^X(x), \quad \ell \notin [\text{Cycl}[s]_{s,h_1}^X(\beta)]^b,$$

where $t \in T[s]^X$, $x \in X$ and $\beta \in [1, \alpha]$. Moreover, $\ell \in \text{dom}(\hat{h})$, which in turn implies $\ell \in \text{dom}(h_1)$ and $\ell \notin [\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}]^\flat$. We conclude that $\ell \in \text{Rem}[s]_{s,h_1}^{X,\alpha}$.
 (\Leftarrow) : Symmetrical to the other direction.

We have $(s, h) \approx_{X,\alpha}^S (s, h_1)$, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h_1)$ and $[(s, h) \#_{X,\alpha} (s, h_1)] = \emptyset$.

Proof of $(s, h) \approx_{X,\alpha}^S (s, h_1)$. By (A)–(F) (see for instance the proof of Lemma 5.39).

Proof of $(s, h) \leftrightarrow_{X,\alpha}^S (s, h_1)$. We have $h \setminus [\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}]^\flat = \hat{h} = h_1 \setminus [\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}]^\flat$.

By (A) and $(s, h) \approx_{X,\alpha}^S (s, h_1)$, from Lemma 5.40(V), $(s, h) \leftrightarrow_{X,\alpha}^S (s, h_1)$. We notice that, since $\approx_{X,\alpha}^S$ is symmetrical, we also deduce $(s, h_1) \leftrightarrow_{X,\alpha}^S (s, h)$.

Proof of $[(s, h) \#_{X,\alpha} (s, h_1)] = \emptyset$. Directly from (A)–(F).

step 2: Symmetrically, we consider memory state (s', h') and, with the same strategy of the previous step, produce a heap h'_1 such that

- $\uparrow\text{Cycl}[s]_{s',h'_1}^{X,\alpha}$ contains $\min(\text{card}(\uparrow\text{Cycl}[s]_{s',h'}^{X,\alpha}), \mathcal{L}(\alpha))$ sets, all of cardinality $\alpha + 1$,
- $(s', h'_1) \approx_{X,\alpha}^S (s', h')$,
- $(s', h'_1) \leftrightarrow_{X,\alpha}^S (s', h')$,
- $[(s', h') \#_{X,\alpha} (s', h'_1)] = \emptyset$.

step 3: From $[(s, h) \#_{X,\alpha} (s, h_1)] = [(s, h) \#_{X,\alpha} (s', h')] = [(s', h') \#_{X,\alpha} (s', h'_1)] = \emptyset$, we deduce that $[(s, h_1) \#_{X,\alpha} (s, h'_1)] = \emptyset$. Since $\approx_{X,\alpha}^S$ is an equivalence relation, from $(s', h'_1) \approx_{X,\alpha}^S (s', h')$ and $(s, h_1) \approx_{X,\alpha}^S (s, h)$, we derive $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$. Thus, as (s, h_1) and (s', h'_1) satisfy the same core formulae of the form $\uparrow\text{loop}_{X,\alpha}^S \geq \beta$, we conclude that

$$\min(\text{card}(\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}), \mathcal{L}(\alpha)) = \min(\text{card}(\uparrow\text{Cycl}[s]_{s',h'_1}^{X,\alpha}), \mathcal{L}(\alpha)).$$

More precisely, from the fact that $\text{card}(\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}) = \min(\text{card}(\uparrow\text{Cycl}[s]_{s,h}^{X,\alpha}), \mathcal{L}(\alpha))$ and $\text{card}(\uparrow\text{Cycl}[s]_{s',h'_1}^{X,\alpha}) = \min(\text{card}(\uparrow\text{Cycl}[s]_{s',h'}^{X,\alpha}), \mathcal{L}(\alpha))$, this allows us to derive

$$\text{card}(\uparrow\text{Cycl}[s]_{s,h_1}^{X,\alpha}) = \text{card}(\uparrow\text{Cycl}[s]_{s',h'_1}^{X,\alpha}).$$

Together with $[(s, h_1) \#_{X,\alpha} (s, h'_1)] = \emptyset$ and $(s, h_1) \approx_{X,\alpha}^S (s', h'_1)$, this implies that (s, h_1) and (s', h'_1) are in $(\bigcap_{\alpha \geq 1} \approx_{X,\alpha}^S)$. We apply Lemma 5.39: $(s, h_1) \leftrightarrow_{X,\alpha}^S (s', h'_1)$. From $(s, h) \leftrightarrow_{X,\alpha}^S (s, h_1)$ and $(s', h'_1) \leftrightarrow_{X,\alpha}^S (s', h')$, and thanks to the transitivity of $\leftrightarrow_{X,\alpha}^S$, we conclude: $(s, h) \leftrightarrow_{X,\alpha}^S (s', h')$.

induction step: $[(s, h) \#_{X,\alpha} (s', h')] > 0$. Let $(S, T) \in [(s, h) \#_{X,\alpha} (s', h')]$. We split the proof in three cases, all of them quite similar, dealing with the different types of sets S and T .

case $(S, T) = (\text{Rem}[s]_{s,h}^{X,\alpha}, \text{Rem}[s]_{s',h'}^{X,\alpha})$. Let us assume that $\text{card}(S) > \text{card}(T)$. This assumption is without loss of generality: in the case where $\text{card}(S) < \text{card}(T)$, it is sufficient to swap (s, h) and (s', h') in the proof, and apply the construction we now show to produce a chain of hops going from (s', h') to (s, h) , i.e.

$$(s', h') = (s_1, h_1) \leftrightarrow_{X,\alpha}^S (s_2, h_2) \leftrightarrow_{X,\alpha}^S \dots \leftrightarrow_{X,\alpha}^S (s_{k-1}, h_{k-1}) \leftrightarrow_{X,\alpha}^S (s_k, h_k) = (s, h).$$

So, assuming $\text{card}(S) > \text{card}(T)$, consider the heap h'' obtained from h by removing from its domain $\text{card}(S) - \text{card}(T)$ locations in $\text{Rem}[s]_{s,h}^{X,\alpha}$ and different from $s(u)$. Formally, $h'' \subseteq h$ and there is a set $Q \subseteq \text{Rem}[s]_{s,h}^{X,\alpha}$ such that $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, $\text{dom}(h'') = \text{dom}(h) \setminus Q$ and $s(u) \notin Q$. Notice that a heap h'' satisfying these conditions exists. In particular, as $\text{card}(S) \neq \text{card}(T)$ and $(s, h) \approx_{X,\alpha}^S (s', h')$, both memory states must satisfy $\text{rem}_{X,\alpha}^S \geq \alpha$, where α is assumed strictly positive. So, both S and T have at least $\alpha \geq 1$ elements, which allows us to keep $s(u)$ in the domain

of h'' , in the case it belongs to $\text{dom}(h)$. We aim at showing that $(s, h) \approx_{X,\alpha}^S (s, h'')$. Similar to the base case concerning unbounded loops, this is done by relying on the six properties (A)–(F) below:

- A. for all $t \in T[S]^X$, $\llbracket t \rrbracket_{s,h}^X$ and $\llbracket t \rrbracket_{s,h''}^X$ are equidefined. When defined, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h''}^X$.

Proof of (A). The set $\text{Rem}[S]_{s,h}^{X,\alpha}$ (resp. Q) enjoys the property required by the set L of Lemma C.1, with respect to the memory states \hat{h} and h (resp. h''). Thus, the result follows from Lemma C.1(A).

- B. For every $t \in T[S]^X$,

- (a) $\text{sby}_{s,h}^X(t)$ and $\text{sby}_{s,h''}^X(t)$ are equidefined. If defined, they are equal,
- (b) $\text{Path}[S]_{s,h}^X(t) = \text{Path}[S]_{s,h''}^X(t)$,
- (c) let $\delta \in [1, \text{card}(\text{Path}[S]_{s,h}^X(t))]$. $h^\delta(\llbracket t \rrbracket_{s,h}^X) = s(u)$ iff $h''^\delta(\llbracket t \rrbracket_{s,h''}^X) = s(u)$.

Proof of (B). Similarly to (A), it follows directly from Lemma C.1(B).

- C. For every $x \in X$, $\text{Pred}[S]_{s,h}^X(x) = \text{Pred}[S]_{s,h''}^X(x)$.

Proof of (C). Analogous to the proof of (C) in the base case, since we can rely on Lemma C.1(O).

- D. For every $\beta \in [1, \alpha]$, $\text{Cycl}[S]_{s,h}^X(\beta) = \text{Cycl}[S]_{s,h''}^X(\beta)$.

Proof of (D). (\Rightarrow): Let $L \in \text{Cycl}[S]_{s,h}^X(\beta)$. So, L is a minimal set describing an unlabelled cycle in h , of length β . Since $L \cap \text{Rem}[S]_{s,h}^{X,\alpha} = \emptyset$ and $L \subseteq \text{dom}(h)$, by definition of h'' we have $L \subseteq \text{dom}(h'')$, and L is a minimal set describing a cycle in h'' . From (A), $L \cap \text{Lab}[S]_{s,h''}^X = \emptyset$. So $L \in \text{Cycl}[S]_{s,h''}^X(\beta)$.

(\Leftarrow): Symmetrical to the other direction.

- E. $\uparrow\text{Cycl}[S]_{s,h}^{X,\alpha} = \uparrow\text{Cycl}[S]_{s,h''}^{X,\alpha}$.

Proof of (E). Analogous to the proof of (D).

- F. (d) $\text{Rem}[S]_{s,h''}^{X,\alpha} \subseteq \text{Rem}[S]_{s,h}^{X,\alpha}$,

- (e) $s(u) \in \text{Rem}[S]_{s,h}^{X,\alpha}$ if and only if $s(u) \in \text{Rem}[S]_{s,h''}^{X,\alpha}$.

Proof of (d). Let $\ell \in \text{Rem}[S]_{s,h''}^{X,\alpha}$. So, $\ell \in \text{dom}(h'')$ and moreover

$$\begin{aligned} \ell &\notin \text{Lab}[S]_{s,h''}^X, & \ell &\notin \text{Path}[S]_{s,h''}^X(t), & \ell &\notin \text{Pred}[S]_{s,h''}^X(x), \\ \ell &\notin [\text{Cycl}[S]_{s,h''}^X(\beta)]^\flat, & \ell &\notin [\uparrow\text{Cycl}[S]_{s,h''}^{X,\alpha}]^\flat, \end{aligned}$$

where $t \in T[S]^X$, $x \in X$ and $\beta \in [1, \alpha]$. From (A)–(F) we conclude that

$$\begin{aligned} \ell &\notin \text{Lab}[S]_{s,h}^X, & \ell &\notin \text{Path}[S]_{s,h}^X(t), & \ell &\notin \text{Pred}[S]_{s,h}^X(x), \\ \ell &\notin [\text{Cycl}[S]_{s,h}^X(\beta)]^\flat, & \ell &\notin [\uparrow\text{Cycl}[S]_{s,h}^{X,\alpha}]^\flat, \end{aligned}$$

where $t \in T[S]^X$, $x \in X$, $\beta \in [1, \alpha]$. By $h'' \subseteq h$, $\ell \in \text{dom}(h)$. So, $\ell \in \text{Rem}[S]_{s,h}^{X,\alpha}$.

Proof of (e). From (d) we conclude that $Q \cup \text{Rem}[S]_{s,h''}^{X,\alpha} = \text{Rem}[S]_{s,h}^{X,\alpha}$. Then, the result follows from the fact that $s(u) \notin Q$ (by definition).

Notice that $Q \cup \text{Rem}[S]_{s,h''}^{X,\alpha} = \text{Rem}[S]_{s,h}^{X,\alpha}$, together with $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, implies $\text{card}(\text{Rem}[S]_{s,h''}^{X,\alpha}) = \text{card}(T)$. We show that $(s, h) \approx_{X,\alpha}^S (s, h'')$, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$, and $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$.

Proof of $(s, h) \approx_{X,\alpha}^S (s, h'')$. By (A)–(F) (see for instance the proof of Lemma 5.39).

Proof of $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$. Directly from the properties (A)–(F), we conclude that

$h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Rem}[S]_{s,h}^{X,\alpha}\} = h'' \setminus \{(\ell, \ell') \in h'' \mid \ell \in \text{Rem}[S]_{s,h''}^{X,\alpha}\}$. Thus, by (A) and $(s, h) \approx_{X,\alpha}^S (s, h'')$, we apply Lemma 5.40(V) and derive $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$.

Proof of $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$. As $\approx_{X,\alpha}^S$ is an equivalence relation, $(s, h) \approx_{X,\alpha}^S (s, h'')$ allows us to derive that $(s, h'') \approx_{X,\alpha}^S (s', h')$. The properties (A)–(F), together with $\text{card}(\text{Rem}[W]_{s,h''}^X) = \text{card}(T)$, we derive $[(s, h'') \#_{X,\alpha}(s', h')] < [(s, h) \#_{X,\alpha}(s', h')]$. By induction hypothesis, $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$.

From $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$, $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$ and by transitivity of the hop relation $\leftrightarrow_{X,\alpha}^S$ we conclude: $(s, h) \leftrightarrow_{X,\alpha}^S (s', h')$.

case $(S, T) = (\text{Cycl}[s]_{s,h}^X(\beta), \text{Cycl}[s]_{s',h'}^X(\beta))$, **for some** $\beta \in [1, \alpha]$. As in the previous case, without loss of generality we can assume $\text{card}(S) > \text{card}(T)$. We consider the heap h'' obtained from h by discharging the locations of $\text{card}(S) - \text{card}(T)$ sets in $\text{Cycl}[s]_{s,h}^X(\beta)$, non containing $s(u)$. Formally, $h'' \subseteq h$ and there is a set $Q \subseteq \text{Cycl}[s]_{s,h}^X(\beta)$ such that $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, $\text{dom}(h'') = \text{dom}(h) \setminus [Q]^\beta$ and $s(u) \notin [Q]^\beta$. Notice that a heap h'' satisfying these conditions exists. In particular, as $\text{card}(S) \neq \text{card}(T)$ and $(s, h) \approx_{X,\alpha}^S (s', h')$, both memory states must satisfy $\text{loop}_X^S(\beta) \geq \alpha$, where α is assumed strictly positive. So, both S and T have at least $\alpha \geq 1$ elements, which allows us to keep $s(u)$ in the domain of h'' , in the case it belongs to $\text{dom}(h)$. (s, h) and (s, h'') enjoy the following properties (A)–(F):

- A. for all $t \in T[s]^X$, $\llbracket t \rrbracket_{s,h}^X$ and $\llbracket t \rrbracket_{s,h''}^X$ are equidefined. When defined, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h''}^X$.

Proof of (A). The set $[\text{Cycl}[s]_{s,h}^X(\beta)]^\beta$ (resp. $[Q]^\beta$) enjoys the property of the set L of Lemma C.1, with respect to the memory states \hat{h} and h (resp. h''). So, (A) follows from Lemma C.1(A).

- B. For every $t \in T[s]^X$,

- (a) $\text{sby}_{s,h}^X(t)$ and $\text{sby}_{s,h''}^X(t)$ are equidefined. If defined, they are equal,
- (b) $\text{Path}[s]_{s,h}^X(t) = \text{Path}[s]_{s,h''}^X(t)$,
- (c) let $\delta \in [1, \text{card}(\text{Path}[s]_{s,h}^X(t))]$. $h^\delta(\llbracket t \rrbracket_{s,h}^X) = s(u)$ iff $h''^\delta(\llbracket t \rrbracket_{s,h''}^X) = s(u)$.

Proof of (B). Similarly to (A), it follows directly from Lemma C.1(B).

- C. For every $x \in X$, $\text{Pred}[s]_{s,h}^X(x) = \text{Pred}[s]_{s,h''}^X(x)$.

Proof of (C). Analogous to the proof of (C) in the base case, since we can rely on Lemma C.1(O).

- D. (d) $\text{Cycl}[s]_{s,h''}^X(\beta) \subseteq \text{Cycl}[s]_{s,h}^X(\beta)$,

- (e) for all $\beta' \in [1, \alpha] \setminus \{\beta\}$, $\text{Cycl}[s]_{s,h}^X(\beta') = \text{Cycl}[s]_{s,h''}^X(\beta')$.

Proof of (d). Let $L \in \text{Cycl}[s]_{s,h''}^X(\beta)$. So, L is a minimal set describing an unlabelled cycle in h'' , of length β . By $h'' \subseteq h$, L describes an cycle in h , of length β . From (A), $L \cap \text{Lab}[s]_{s,h}^X = \emptyset$. Thus, $L \in \text{Cycl}[s]_{s,h}^X(\beta)$.

Proof of (e). (\Rightarrow): Let $L \in \text{Cycl}[s]_{s,h}^X(\beta')$. So, L is a minimal set describing an unlabelled cycle in h'' , of length β' . As $\beta \neq \beta'$, $L \notin \text{Cycl}[s]_{s,h}^X(\beta)$, and so $L \subseteq \text{dom}(h'')$. By $h'' \subseteq h$, L describes an cycle in h , of length β' . From (A), $L \cap \text{Lab}[s]_{s,h}^X = \emptyset$. Thus, $L \in \text{Cycl}[s]_{s,h}^X(\beta')$.

(\Leftarrow): Symmetrical to the other direction.

- E. $\uparrow \text{Cycl}[s]_{s,h}^{X,\alpha} = \uparrow \text{Cycl}[s]_{s,h''}^{X,\alpha}$.

Proof of (E). Analogous to the proof of (e).

- F. $\text{Rem}[s]_{s,h}^{X,\alpha} = \text{Rem}[s]_{s,h''}^{X,\alpha}$.

Proof of (F). (\Rightarrow): Let $\ell \in \text{Rem}[s]_{s,h}^{X,\alpha}$. So, $\ell \in \text{dom}(h)$ and moreover

$$\begin{aligned} \ell \notin \text{Lab}[s]_{s,h}^X, \quad & \ell \notin \text{Path}[s]_{s,h}^X(t), \quad \ell \notin \text{Pred}[s]_{s,h}^X(x), \\ \ell \notin [\text{Cycl}[s]_{s,h}^X(\beta')]^\beta, \quad & \ell \notin [\uparrow \text{Cycl}[s]_{s,h}^{X,\alpha}]^\beta, \end{aligned}$$

where $\mathbf{t} \in T[\mathcal{S}]^X$, $x \in X$ and $\beta' \in [1, \alpha]$. From (A)–(D) we conclude that

$$\begin{aligned}\ell &\notin \text{Lab}[\mathcal{S}]_{s,h}^X, & \ell &\notin \text{Path}[\mathcal{S}]_{s,h''}^X(\mathbf{t}), & \ell &\notin \text{Pred}[\mathcal{S}]_{s,h''}^X(x), \\ \ell &\notin [\text{Cycl}[\mathcal{S}]_{s,h''}^X(\beta')]^\flat, & \ell &\notin [\uparrow\text{Cycl}[\mathcal{S}]_{s,h''}^{X,\alpha}]^\flat,\end{aligned}$$

where $\mathbf{t} \in T[\mathcal{S}]^X$, $x \in X$, $\beta' \in [1, \alpha]$. Moreover, $\ell \notin [Q]^\flat \subseteq [\text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)]^\flat$ and so $\ell \in \text{dom}(h'')$. So, $\ell \in \text{Rem}[\mathcal{S}]_{s,h''}^{X,\alpha}$.

(\Leftarrow): Analogous to the other direction. Recall that $\ell \in \text{dom}(h'') \cap Q = \emptyset$.

Notice that (A)–(F) allow us to conclude that $Q \cup \text{Cycl}[\mathcal{S}]_{s,h''}^X(\beta) = \text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)$. By $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, this implies that $\text{card}(\text{Cycl}[\mathcal{S}]_{s,h''}^X(\beta)) = \text{card}(T)$. Moreover, directly from the fact that $s(u) \notin Q$,

(f) $s(u) \in \text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)$ if and only if $s(u) \in \text{Cycl}[\mathcal{S}]_{s,h''}^X(\beta)$.

We show $(s, h) \approx_{X,\alpha}^S (s, h'')$, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$, and $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$. By transitivity of the hop relation $\leftrightarrow_{X,\alpha}^S$, the last two memberships imply $(s, h) \leftrightarrow_{X,\alpha}^S (s', h')$.

Proof of $(s, h) \approx_{X,\alpha}^S (s, h'')$. By (A)–(F) and (f) (e.g. see the proof of Lemma 5.39).

Proof of $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$. Directly from the properties (A)–(F), we conclude that $h \setminus \{(\ell, \ell') \in h \mid \ell \in [\text{Cycl}[\mathcal{S}]_{s,h}^X(\beta)]^\flat\} = h'' \setminus \{(\ell, \ell') \in h'' \mid \ell \in [\text{Cycl}[\mathcal{S}]_{s,h''}^X(\beta)]^\flat\}$. Thus, by (A) and $(s, h) \approx_{X,\alpha}^S (s, h'')$, we can apply Lemma 5.40(III) and derive that $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$.

Proof of $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$. As $\approx_{X,\alpha}^S$ is an equivalence relation, $(s, h) \approx_{X,\alpha}^S (s, h'')$ allows us to derive that $(s, h'') \approx_{X,\alpha}^S (s', h')$. From the properties (A)–(F), and $\text{card}(\text{Cycl}[\mathcal{S}]_{s,h''}^X(\beta)) = \text{card}(T)$, we derive $[(s, h'') \#_{X,\alpha} (s', h')] < [(s, h) \#_{X,\alpha} (s', h')]$. By induction hypothesis, $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$.

case $(S, T) = (\text{Pred}[\mathcal{S}]_{s,h}^X(x), \text{Pred}[\mathcal{S}]_{s',h'}^X(x))$, **for some** $x \in X$. As previously done, without loss of generality we assume $\text{card}(S) > \text{card}(T)$. Consider the heap h'' obtained from h by removing from its domain $\text{card}(S) - \text{card}(T)$ locations in $\text{Pred}[\mathcal{S}]_{s,h}^X(x)$ and different from $s(u)$. Formally, $h'' \subseteq h$ and there is a set $Q \subseteq \text{Pred}[\mathcal{S}]_{s,h}^X(x)$ such that $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, $\text{dom}(h'') = \text{dom}(h) \setminus Q$ and $s(u) \notin Q$. Notice that a heap h'' satisfying these conditions exists. In particular, as $\text{card}(S) \neq \text{card}(T)$ and $(s, h) \approx_{X,\alpha}^S (s', h')$, both memory states must satisfy $\text{pred}_X^S(x) \geq \alpha$, where α is assumed strictly positive. So, both S and T have at least $\alpha \geq 1$ elements, which allows us to keep $s(u)$ in the domain of h'' , in the case it belongs to $\text{dom}(h)$. (s, h) and (s, h'') enjoy the following properties:

A. for all $\mathbf{t} \in T[\mathcal{S}]^X$, $\llbracket \mathbf{t} \rrbracket_{s,h}^X$ and $\llbracket \mathbf{t} \rrbracket_{s,h''}^X$ are equidefined. When defined, $\llbracket \mathbf{t} \rrbracket_{s,h}^X = \llbracket \mathbf{t} \rrbracket_{s,h''}^X$.

Proof of (A). The set $\text{Pred}[\mathcal{S}]_{s,h}^X(x)$ (resp. Q) enjoys the property of the set L of Lemma C.1, with respect to the memory states \widehat{h} and h (resp. h''). Thus, (A) follows from Lemma C.1(A).

B. For every $\mathbf{t} \in T[\mathcal{S}]^X$,

- (a) $\text{sby}_{s,h}^X(\mathbf{t})$ and $\text{sby}_{s,h''}^X(\mathbf{t})$ are equidefined. If defined, they are equal,
- (b) $\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t}) = \text{Path}[\mathcal{S}]_{s,h''}^X(\mathbf{t})$,
- (c) let $\delta \in [1, \text{card}(\text{Path}[\mathcal{S}]_{s,h}^X(\mathbf{t}))]$. $h^\delta(\llbracket \mathbf{t} \rrbracket_{s,h}^X) = s(u)$ iff $h''^\delta(\llbracket \mathbf{t} \rrbracket_{s,h''}^X) = s(u)$.

Proof of (B). Similarly to (A), it follows directly from Lemma C.1(B).

C. (d) $\text{Pred}[\mathcal{S}]_{s,h''}^X(x) \subseteq \text{Pred}[\mathcal{S}]_{s,h}^X(x)$,

- (e) for all $y \in X$, if $s(y) \neq s(x)$ then $\text{Pred}[\mathcal{S}]_{s,h}^X(y) = \text{Pred}[\mathcal{S}]_{s,h''}^X(y)$.

Proof of (d). Let $\ell \in \text{Pred}[S]_{s,h''}^X(x)$. $h''(\ell) = s(x)$, and h'' does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. From $h'' \subseteq h$, $h(\ell) = s(x)$. By Lemma C.1(O), h does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. Thus, $\ell \in \text{Pred}[S]_{s,h}^X(x)$.

Proof of (e). Analogous to the proof of (C) in the base case, since we can rely on Lemma C.1(O).

- D. For every $\beta \in [1, \alpha]$, $\text{Cycl}[S]_{s,h}^X(\beta) = \text{Cycl}[S]_{s,h''}^X(\beta)$.

Proof of (D). (\Rightarrow): Let $L \in \text{Cycl}[S]_{s,h}^X(\beta)$. So, L is a minimal set describing an unlabelled cycle in h , of length β . Since $L \notin \text{Pred}[S]_{s,h}^X(x)$ and $L \subseteq \text{dom}(h)$, by definition of h'' we have $L \subseteq \text{dom}(h'')$, and L is a minimal set describing a cycle in h'' . From (A), $L \cap \text{Lab}[S]_{s,h''}^X = \emptyset$. So $L \in \text{Cycl}[S]_{s,h''}^X(\beta)$.

(\Leftarrow): Symmetrical to the other direction.

- E. $\uparrow \text{Cycl}[S]_{s,h}^{X,\alpha} = \uparrow \text{Cycl}[S]_{s,h''}^{X,\alpha}$.

Proof of (E). Analogous to the proof of (D).

- F. $\text{Rem}[S]_{s,h''}^{X,\alpha} = \text{Rem}[S]_{s,h}^{X,\alpha}$.

Proof of (F). Analogous to the proof of (F) in the previous case of the proof (dealing with $\text{Cycl}[S]_{s,h}^X(\beta)$ instead of $\text{Pred}[S]_{s,h}^X(x)$).

Thanks to (A)–(F), we conclude that $Q \cup \text{Pred}[S]_{s,h''}^X(x) = \text{Pred}[S]_{s,h}^X(x)$, and so by $\text{card}(Q) = \text{card}(S) - \text{card}(T)$, we have $\text{card}(\text{Pred}[S]_{s,h''}^X(x)) = \text{card}(T)$. Moreover, directly from the fact that $s(u) \notin Q$,

- (f) $s(u) \in \text{Pred}[S]_{s,h}^X(x)$ if and only if $s(u) \in \text{Pred}[S]_{s,h''}^X(x)$.

We show $(s, h) \approx_{X,\alpha}^S (s, h'')$, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$, and $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$. By transitivity of the hop relation $\leftrightarrow_{X,\alpha}^S$, the last two memberships imply $(s, h) \leftrightarrow_{X,\alpha}^S (s', h')$.

Proof of $(s, h) \approx_{X,\alpha}^S (s, h'')$. By (A)–(F) and (f) (e.g. see the proof of Lemma 5.39).

Proof of $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$. Directly from the properties (A)–(F), we conclude that $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Pred}[S]_{s,h}^X(x)\} = h'' \setminus \{(\ell, \ell') \in h'' \mid \ell \in \text{Pred}[S]_{s,h''}^X(x)\}$.

Thus, by (A) and $(s, h) \approx_{X,\alpha}^S (s, h'')$, we can apply Lemma 5.40(I) and derive that $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$.

Proof of $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$. As $\approx_{X,\alpha}^S$ is an equivalence relation, $(s, h) \approx_{X,\alpha}^S (s, h'')$ allows us to derive that $(s, h'') \approx_{X,\alpha}^S (s', h')$. From the properties (A)–(F), and $\text{card}(\text{Pred}[S]_{s,h''}^X(x)) = \text{card}(T)$, we derive $[(s, h'') \#_{X,\alpha} (s', h')] < [(s, h) \#_{X,\alpha} (s', h')]$.

By induction hypothesis, $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$.

case: $(S, T) = (\text{Path}[S]_{s,h}^X(t), \text{Path}[S]_{s',h'}^X(t))$, for some $t \in T[S]^X$. Again, without loss of generality we assume $\text{card}(S) > \text{card}(T)$. Compared to the previous cases, this case is slightly more involved. In what follows, we write $S(\alpha)$, for the upper bound on β in the core formulae of the form $\text{sees}_X(t, t') \geq \beta$. Similarly, we write $S_{\text{left}}(\alpha)$ and $S_{\text{right}}(\alpha)$ for the upper bound on β_1 and β_2 , respectively, on the core formulae of the form $u \in \text{sees}_X(t, t') \geq (\beta_1, \beta_2)$. So, $S_{\text{left}}(\alpha) = \frac{1}{6}\alpha(\alpha+1)(\alpha+2)+1$, $S_{\text{right}}(\alpha) = \frac{1}{2}\alpha(\alpha+3)$,

$$S(\alpha) = S_{\text{left}}(\alpha) + S_{\text{right}}(\alpha) = \frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3). \quad (\dagger)$$

Since $\text{card}(S) > \text{card}(T)$ and $(s, h) \approx_{X,\alpha}^S (s', h')$, we have

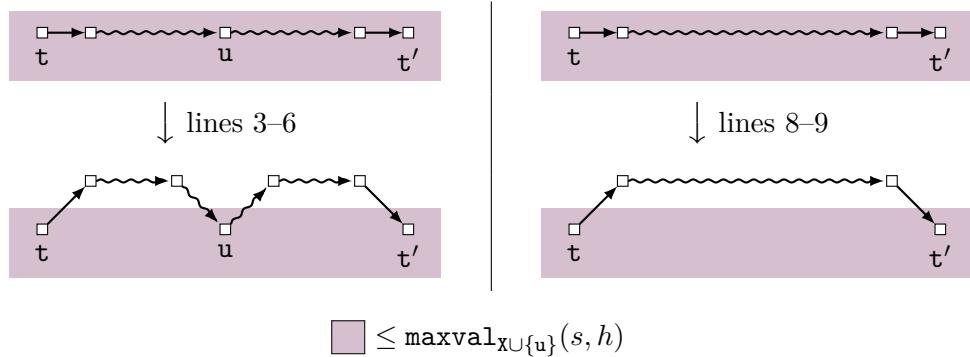
$$\text{card}(\text{Path}[S]_{s,h}^X(t)) = \text{card}(S) > \text{card}(T) = \text{card}(\text{Path}[S]_{s',h'}^X(t)) \geq S(\alpha).$$

Let $t' \in \text{Lab}[S]_{s,h}^X$ be a term such that $\llbracket t' \rrbracket_{s,h}^X = \text{sby}_{s,h}^X(t)$. Informally, we want to modify the path in h , going from $\llbracket t \rrbracket_{s,h}^X$ to $\llbracket t' \rrbracket_{s,h}^X$ so that it becomes of length $\text{card}(T)$. The resulting heap, say h'' , should satisfy $(s, h) \approx_{X,\alpha}^S (s, h'')$. When defining h'' , we must

```

1:  $\hat{h} \leftarrow h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Path}[S]_{s,h}^X(t)\}.$ 
2:  $n \leftarrow \text{card}(\text{Path}[S]_{s',h'}^X(t)).$ 
3: if  $(s, h) \models u \in \text{sees}_X(t, t') \geq (1, 1)$  then
4:   let  $\delta \in [1, n - 1]$  such that  $h'^\delta([\![t]\!]_{s',h'}^X) = s(u).$ 
5:   let  $\tilde{\ell}_1, \dots, \tilde{\ell}_{\delta-1}, \tilde{\ell}_{\delta+1}, \dots, \tilde{\ell}_{n-1}$  be  $n-2$  distinct locations greater than  $\text{maxval}_{X \cup \{u\}}(s, h).$ 
6:    $\tilde{h} \leftarrow \{[\![t]\!]_{s,h}^X \mapsto \tilde{\ell}_1 \mapsto \dots \mapsto \tilde{\ell}_{\delta-1} \mapsto s(u) \mapsto \tilde{\ell}_{\delta+1} \mapsto \dots \mapsto \tilde{\ell}_{n-1} \mapsto [\![t']\!]_{s,h}^X\}.$ 
7: else
8:   let  $\tilde{\ell}_1, \dots, \tilde{\ell}_{n-1}$  be  $n-1$  distinct locations greater than  $\text{maxval}_{X \cup \{u\}}(s, h).$ 
9:    $\tilde{h} \leftarrow \{[\![t]\!]_{s,h}^X \mapsto \tilde{\ell}_1 \mapsto \dots \mapsto \tilde{\ell}_{n-1} \mapsto [\![t']\!]_{s,h}^X\}.$ 
10:  $h'' \leftarrow \hat{h} + \tilde{h}.$ 

```

Figure C.1: Strategy defining h'' .

deal with the possible occurrence of $s(u)$ in $\text{Path}[S]_{s,h}^X(t)$, so that (s, h'') agrees with (s, h) on the satisfaction of the core formulae of the form $u \in \text{sees}_X(t, t') \geq (\beta_1, \beta_2)$. All of this can be done rather easily, by defining h'' following the strategy in Figure C.1. As done in the strategy (line 1), in what follows we write \hat{h} for the heap $h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Path}[S]_{s,h}^X(t)\}$. The strategy is split into two cases, depending on whether $(s, h) \models u \in \text{sees}_X(t, t') \geq (1, 1)$. In both cases, we define (lines 6 and 9) a heap \tilde{h} such that

- (ρ_1) $\text{dom}(\tilde{h})$ is a minimal set describing a path $(\tilde{\ell}_0, \dots, \tilde{\ell}_n)$ in \tilde{h} , from $[\![t]\!]_{s,h}^X$ to $[\![t']\!]_{s,h}^X$,
- (ρ_2) $n = \text{card}(\tilde{h}) = \text{card}(\text{Path}[S]_{s',h'}^X(t))$,
- (ρ_3) for all $j \in [1, n - 1]$, $\tilde{\ell}_j = s(u)$ if and only if $h'^\delta([\![t]\!]_{s',h'}^X) = s(u)$.
- (ρ_4) for all $j \in [1, n - 1]$, $\tilde{\ell}_j > \text{maxval}_{X \cup \{u\}}(s, h)$ or $\tilde{\ell}_j = s(u) \in \text{Path}[S]_{s,h}^X(t) \setminus \{[\![t]\!]_{s,h}^X\}$.

The proofs of (ρ_1)–(ρ_4) are straightforward and follow from the definition of \tilde{h} . They are left to the reader. Notice that (ρ_4) implies that \tilde{h} and \hat{h} are disjoint. Indeed, by definition of \hat{h} for all $\ell \in \text{dom}(\hat{h})$ we have $\ell \leq \text{maxval}_{X \cup \{u\}}(s, \hat{h})$ and $\ell \notin \text{Path}[S]_{s,h}^X(t)$. As shown in line 10, h'' is defined as $\hat{h} + \tilde{h}$. Notice that the definitions of \hat{h} and h'' lead to the equality:

$$h'' \setminus \tilde{h} = \hat{h} = h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Path}[S]_{s,h}^X(t)\}.$$

For simplicity, below we write $(\tilde{\ell}_0, \dots, \tilde{\ell}_n)$ for the path of h'' (and \tilde{h}) described by $\text{dom}(\tilde{h})$. Similarly, $(\ell_0^P, \dots, \ell_m^P)$ denotes the path in h described by $\text{Path}[S]_{s,h}^X(t)$,

where $m = \text{card}(\text{Path}[s]_{s,h}^X(t))$. We show two essential properties of h'' :

- (ρ_5) Every $\ell \in \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$ does not belong to $\text{Lab}[s]_{s,h''}^X$.
- (ρ_6) Let $x \in X$ and $\ell \in \text{LOC}$. h'' witnesses a path from $s(x)$ to $\ell \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$ if and only if h witnesses a path from $s(x)$ to $\ell \notin \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$.

Proof of (ρ_5). Let $\ell \in \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, that is $\ell = \tilde{\ell}_j$ for some $j \in [1, n - 1]$.

According to (ρ_4), we either have

- $\tilde{\ell}_j > \text{maxval}_{X \cup \{u\}}(s, h)$, or
- $\tilde{\ell}_j = s(u)$ and $s(u) \in \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$.

The first case is rather straightforward. By definition of maximum value, it cannot be that $\tilde{\ell}_j \in \text{ran}(h)$ and so, by definition of \tilde{h} , $\tilde{\ell}_j \notin \text{ran}(\tilde{h})$. Then, from (ρ_1) there is exactly one location $\ell' \in \text{dom}(\tilde{h})$ such that $\tilde{h}(\ell') = \tilde{\ell}_j$. So, by definition of h'' , ℓ' is the only location in $\text{dom}(h'')$ such that $h''(\ell') = \tilde{\ell}_j$. As $\tilde{\ell}_j \in \text{dom}(\tilde{h}) \subseteq \text{dom}(h'')$, by Lemma 5.52 we conclude that $\tilde{\ell}_j \notin \text{Lab}[s]_{s,h''}^X$.

We consider the second case, i.e. $\tilde{\ell}_j = s(u)$ and $s(u) \in \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. *Ad absurdum*, suppose $s(u) \in \text{Lab}[s]_{s,h''}^X$. The proof splits depending on whether $s(u) = s(x)$, $s(u) = \llbracket e(x) \rrbracket_{s,h''}^X$ or $s(u) = \llbracket m(x, y) \rrbracket_{s,h''}^X$, for some $x, y \in X$. All cases lead to a contradiction. So, $s(u) \notin \text{Lab}[s]_{s,h''}^X$, which ends the proof.

case: $s(u) = s(x)$. This case is obvious. $s(u) = s(x)$ implies $s(u) \in \text{Lab}[s]_{s,h}^X$, which contradicts $s(u) \in \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. Indeed, we recall that $\llbracket t \rrbracket_{s,h}^X$ is the only labelled location in $\text{Path}[s]_{s,h}^X(t)$.

case: $s(u) = \llbracket e(x) \rrbracket_{s,h''}^X$. As $s(u) \in \text{dom}(\tilde{h})$, from the semantics of end-point variables, h'' witnesses two disjoint non-empty paths $\rho = (\ell_0, \dots, \ell_{k_1})$ and $\rho' = (\ell'_0, \dots, \ell'_{k_2})$, where ρ goes from $s(x)$ to $s(u)$, whereas ρ' goes from $s(u)$ to itself. Notice that, by definition of \tilde{h} , this implies that h'' witnesses a non-empty path going from $\tilde{\ell}_n = \text{sby}_{s,h}^X(\ell)$ to $\tilde{\ell}_j = s(u)$, i.e. there is $i_2 \in [1, k_2 - 1]$ such that $\ell'_{i_2} = \text{sby}_{s,h}^X(\ell)$. We divide the proof depending on whether $\llbracket t \rrbracket_{s,h}^X$ appears in ρ .

case: $\llbracket t \rrbracket_{s,h}^X$ belongs to ρ . This implies that there is $i_1 \in [0, k_1 - 1]$ such that $\ell_{i_1} = \llbracket t \rrbracket_{s,h}^X$. In particular, the path ρ is of the form

$$(\ell_0 = s(x), \dots, \ell_{i_1} = \tilde{\ell}_0 = \llbracket t \rrbracket_{s,h}^X, \dots, \ell_{k_1} = \tilde{\ell}_j = s(u)),$$

where $\ell_0, \dots, \ell_{i_1-1}$ belong to $\text{dom}(\tilde{h})$. Informally, this means that ρ is made of a (possibly empty) prefix of locations in $\text{dom}(\tilde{h})$ and a non-empty suffix of locations in $\text{dom}(\tilde{h})$. Similarly, the path ρ' is of the form

$$(\ell'_0 = \tilde{\ell}_j = s(u), \dots, \ell'_{i_2} = \tilde{\ell}_n = \text{sby}_{s,h}^X(t), \ell'_{i_2+1}, \dots, \ell'_{k_2} = s(u))$$

where $\ell'_{i_2}, \dots, \ell'_{k_2-1}$ belong to $\text{dom}(\tilde{h})$. From $s(u) \in \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, there is $j' \in [1, m - 1]$ such that $s(u) \in \ell'^P_{j'}$. We recall that $(\ell'^P_0, \dots, \ell'^P_m)$ is the path described by $\text{Path}[s]_{s,h}^X(t)$, going from $\llbracket t \rrbracket_{s,h}^X$ to $\text{sby}_{s,h}^X(t)$. From $\tilde{h} \subseteq h$, we conclude that h witnesses the following two disjoint non-empty paths:

$$(\ell_0 = s(x), \dots, \ell_{i_1} = \ell^P_0 = \llbracket t \rrbracket_{s,h}^X, \dots, \ell^P_{j'} = s(u)),$$

$$(\ell^P_{j'} = s(u), \dots, \ell^P_m = \text{sby}_{s,h}^X(t) = \ell'_{i_2}, \ell'_{i_2+1}, \dots, \ell'_{k_2} = s(u)).$$

From the semantics of end-point variables this allows us to conclude that $s(u) \in \llbracket e(x) \rrbracket_{s,h}^X$. Again, this contradicts $s(u) \in \text{Path}[s]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$.

case: $\llbracket t \rrbracket_{s,h}^X$ does not belong to ρ . First of all, we show that the locations $\ell_0, \dots, \ell_{k_1-1}$ belongs to $\text{dom}(\tilde{h})$. From (ρ_4), every location of $\text{dom}(\tilde{h})$ that is not

$\llbracket t \rrbracket_{s,h}^x$ nor $s(u)$ is greater than $\text{maxval}_{X \cup \{u\}}(s, h)$. In particular, as $\ell_0 = s(x)$, this means that $\ell_0 \notin \text{dom}(\tilde{h})$. By definition of h'' , we conclude that $\ell_0 \in \text{dom}(\tilde{h})$. *Ad absurdum*, suppose $k' \in [1, k_1 - 1]$ be the smallest index such that $\ell_{k'} \in \text{dom}(\tilde{h})$. So, $\ell_{k'-1} \in \text{dom}(\tilde{h})$. We know that $\ell_{k'} \neq s(u)$ and $\ell_{k'} \neq \llbracket t \rrbracket_{s,h}^x$. By (p4), this implies $\ell_{k'} > \text{maxval}_{X \cup \{u\}}(s, h)$. However, is contradictory, since $\ell_{k'-1} \in \text{dom}(\tilde{h})$ and $\tilde{h} \subseteq h$ imply that $h''(\ell_{k'-1}) = \tilde{h}(\ell_{k'-1}) \in \text{ran}(h)$. Therefore, the locations $\ell_0, \dots, \ell_{k_1-1}$ do not belong to $\text{dom}(\tilde{h})$ and, as they belong to $\text{dom}(h'')$, we conclude that they belong to $\text{dom}(\tilde{h})$. By definition of \tilde{h} , we conclude that ρ is a non-empty path in h , going from $s(x)$ to $s(u)$, and that all the locations $\ell_0 = s(x), \dots, \ell_{k_1-1}$ do not belong to $\text{Path}[s]_{s,h}^x(t)$. In particular, this means that h witnesses a (possibly empty) path going from $s(x)$ to ℓ_{k_1-1} , where $h(\ell_{k_1-1}) = s(u)$. This is contradictory, as from $s(u) \in \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$ it allows us to conclude that $\ell_{k_1-1} \in \text{Path}[s]_{s,h}^x(t)$. For this, we simply apply the following result, that is shown during the proof Lemma 5.40(II).

(κ) Consider a location $\ell_1 \in \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, for some memory state (s, h) . Let ℓ_2 be a location such that $h(\ell_2) = \ell_1$ and h witnesses a path going from $s(x)$ to ℓ_2 , for some $x \in X$. Then, $\ell_2 \in \text{Path}[s]_{s,h}^x(t)$.

case: $s(u) = \llbracket m(x, y) \rrbracket_{s,h''}^x$. By definition, this implies that h'' witnesses two non-empty disjoint paths $\rho = (\ell_0, \dots, \ell_{k_1})$ and $\rho' = (\ell'_0, \dots, \ell'_{k_2})$, where ρ goes from $s(x)$ to $s(u)$, whereas ρ' goes from $s(y)$ to $s(u)$, respectively. Moreover, $s(u)$ does not belong to a cycle in h'' . Without loss of generality we assume that if $\llbracket t \rrbracket_{s,h}^x$ belongs to one of the two paths, it belongs to ρ , i.e. it is in $\{\ell_0, \dots, \ell_{k_1}\}$. From the disjointness of ρ and ρ' , we conclude that for every $i \in [0, k_2 - 1]$, $\ell'_i \neq \llbracket t \rrbracket_{s,h}^x$. Analogously to the previous case, since $\ell'_0 = s(y)$ and, by (p4), apart from $\llbracket t \rrbracket_{s,h}^x$ and $s(u)$, every location in $\text{dom}(\tilde{h})$ is greater than $\text{maxval}_{X \cup \{u\}}(s, h)$, we conclude that for every $i \in [0, k_2 - 1]$, $\ell'_i \notin \text{dom}(\tilde{h})$. By definition of h'' and \tilde{h} , this implies that ρ describes a non-empty path in h , going from $s(y)$ to $s(u)$ and such that for every $i \in [0, k_2 - 1]$, $\ell'_i \notin \text{Path}[s]_{s,h}^x(t)$. In particular, this means that h witnesses a (possibly empty) path, going from $s(u)$ to ℓ'_{k_2-1} , where $h(\ell'_{k_2-1}) = s(u)$. However, this is contradictory, as from $s(u) \in \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$ it allows us to conclude that $\ell'_{k_2-1} \in \text{Path}[s]_{s,h}^x(t)$, by applying (κ).

Below, we recall that a non-empty path (ℓ_0, \dots, ℓ_k) is minimal in a heap h whenever h does not witness a shorter non-empty path going from ℓ_0 to ℓ_k . Equivalently, $\ell_j \neq \ell_k$, for all $j \in [1, k - 1]$.

Proof of (p6). (\Rightarrow): Let $x \in X$ and $\ell \in \text{LOC}$. Clearly, if $s(x) = \ell$ then the statement trivially holds from the fact that $s(x) \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$ (in this case, h witnesses an empty path from $s(x)$ to ℓ). Otherwise, let $\rho = (\ell_0, \dots, \ell_k)$ be the minimal non-empty path in h'' , going from $\ell_0 = s(x)$ to $\ell_k = \ell \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. We divide the proof depending on whether $\ell_i = \llbracket t \rrbracket_{s,h}^x$, for some $i \in [0, k - 1]$.

case: $\ell_i = \llbracket t \rrbracket_{s,h}^x$, for some $i \in [0, k - 1]$. Since $\ell_k \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, we conclude that the path described by $\text{dom}(\tilde{h})$ must be completely included in ρ . That is, we have $i + n \leq k$ and for every $j \in [0, n]$, $\ell_{i+j} = \tilde{\ell}_j$. So, $\ell_{i+n} = \tilde{\ell}_n = \text{sby}_{s,h}^x(t)$, and the $\{\ell_i, \dots, \ell_{i+n-1}\} = \text{dom}(\tilde{h})$. Since ρ is the minimal non-empty path in h'' going from $s(x)$ to ℓ , the locations $\ell_0, \dots, \ell_{k-1}$ are all distinct. This implies that for every $j \in [0, i-1] \cup [i+n, k-1]$ we have

$\ell_j \notin \text{dom}(\tilde{h})$. By definition of h'' and since these locations are in $\text{dom}(h'')$, we conclude that for every $j \in [0, i - 1] \cup [i + n, k - 1]$, $\ell_j \in \text{dom}(\hat{h})$. By definition of \hat{h} , this implies that (ℓ_0, \dots, ℓ_i) and $(\ell_{i+n}, \dots, \ell_k)$ are two paths in h . Together with $\ell_i = \llbracket t \rrbracket_{s,h}^x$ and $\ell_{i+n} = \text{sby}_{s,h}^x(t)$, this implies that h witnesses a path going from $\ell_0 = s(x)$ to $\ell_k = \ell$. Lastly, let us prove that $\ell_k = \ell \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. If $k = i + n$ then the result trivially follows from $\ell = \text{sby}_{s,h}^x(t)$. Otherwise, $i + n < k$ implies that $[i + n, k - 1]$ is non-empty and so $\ell_{k-1} \in \text{dom}(\hat{h})$. By definition of \hat{h} , this implies $h(\ell_{k-1}) = \ell$ and $\ell_{k-1} \notin \text{Path}[s]_{s,h}^x(t)$. By (κ), we derive that $\ell \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$.

case: $\ell_i \neq \llbracket t \rrbracket_{s,h}^x$, for all $i \in [0, k - 1]$. We show that for every $i \in [0, k - 1]$, $\ell_i \in \text{dom}(\hat{h})$. Ad absurdum, suppose $i \in [0, k - 1]$ to be the smallest index such that $\ell_i \in \text{dom}(\hat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. According (p4), we either have

- $\ell_i > \text{maxval}_{x \cup \{u\}}(s, h)$, or
- $\ell_i = s(u)$ and $s(u) \in \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$.

We divide the proof depending on these two cases.

case: $\ell_i > \text{maxval}_{x \cup \{u\}}(s, h)$. In this case, it cannot be that $i = 0$, as $\ell_0 = s(x) \leq \text{maxval}_{x \cup \{u\}}(s, h)$. Therefore, $\ell_1 \in \text{dom}(\hat{h})$. As i is assumed to be the smallest index such that $\ell_i \in \text{dom}(\hat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, this implies $\ell_{i-1} \in \text{dom}(\hat{h})$. However, by definition of h'' , this implies $\ell_i = h''(\ell_{i-1}) \in \text{ran}(h)$, in contradiction with $\ell_i > \text{maxval}_{x \cup \{u\}}(s, h)$.

case: $\ell_i = s(u)$ and $s(u) \in \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. Again, it cannot be that $i = 0$, as $\ell_0 = s(x) \in \text{Lab}[s]_{s,h}^x$ and therefore $\ell_0 \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. As i is assumed to be the smallest index such that $\ell_i \in \text{dom}(\hat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, this implies that $\ell_{i-1} \in \text{dom}(\hat{h})$, that \hat{h} witnesses a path going from $\ell_0 = s(x)$ to ℓ_{i-1} and that $\hat{h}(\ell_{i-1}) = \ell_i = s(x)$. However, by definition of \hat{h} , this implies that $\ell_{i-1} \notin \text{Path}[s]_{s,h}^x(t)$, that h witnesses a path going from $\ell_0 = s(x)$ to ℓ_{i-1} and that $h(\ell_{i-1}) = s(x)$. By (κ), this is contradictory.

In both cases we reached a contradiction, which leads us to derive that no $i \in [0, k - 1]$ can be such that $\ell_i \in \text{dom}(\hat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. Since $\ell_i \neq \llbracket t \rrbracket_{s,h}^x$, this implies $\ell_i \notin \text{dom}(\tilde{h})$. Since $\ell_i \in \text{dom}(h'')$, by definition of h'' , we derive $\ell_i \in \text{dom}(\hat{h})$. Then, directly by definition of \hat{h} , we conclude that $\rho = (\ell_0, \dots, \ell_k)$ is a path in h , going from $s(x)$ to ℓ . Moreover, $\ell_{k-1} \notin \text{Path}[s]_{s,h}^x(t)$. By (κ), this implies $\ell_k = \ell \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$.

(\Leftarrow): Analogous to the other direction, the only difference being how we derive that $\ell \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. Briefly, suppose $\rho = (\ell_0, \dots, \ell_k)$ to be the minimal non-empty path in h , going from $\ell_0 = s(x)$ to $\ell_k = \ell \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$ (similarly to the other direction, the case for $s(u) = \ell$ is trivial). The proof splits depending on whether $\ell_i = \llbracket t \rrbracket_{s,h}^x$, for some $i \in [0, k - 1]$.

case: $\ell_i = \llbracket t \rrbracket_{s,h}^x$, for some $i \in [0, k - 1]$. Symmetrically to the other direction, we are able to conclude that the paths (ℓ_0, \dots, ℓ_i) and $(\ell_{i+m}, \dots, \ell_k)$, where $\ell_{i+m} = \text{sby}_{s,h}^x(t) = \tilde{\ell}_n$, are two paths in both \hat{h} and h'' . As $\ell_i = \llbracket t \rrbracket_{s,h}^x = \tilde{\ell}_0$, by (p1), we derive that h'' witnesses a non-empty path from ℓ_0 to ℓ_k . Now, the path $(\ell_{i+m}, \dots, \ell_k)$ is either empty, and so $\ell_k = \text{sby}_{s,h}^x(t)$, or it is non-empty, and so $\ell_{k-1} \in \text{dom}(\hat{h}) \subseteq \text{dom}(h)$. In both cases, $\ell_k \leq \text{maxval}_{x \cup \{u\}}(s, h)$. By $\ell_k \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$ and (p4), we derive $\ell_k \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$.

case: $\ell_i \neq \llbracket t \rrbracket_{s,h}^x$, for all $i \in [0, k-1]$. Symmetrically to the other direction, we are able to show that, for every $i \in [0, k-1]$, $\ell_i \in \text{dom}(\widehat{h})$. This implies that ρ is a path in h'' . From $\ell_{k-1} \in \text{dom}(h)$ we have $\ell_k = h(\ell_{k-1}) \leq \text{maxval}_{X \cup \{u\}}(s, h)$. Again, by (p4) we conclude that $\ell_k \notin \text{dom}(\widehat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$.

We strengthen (p6) into the two following results:

- (p7) Let (ℓ_0, \dots, ℓ_k) , where $\ell_0, \ell_k \notin \text{dom}(\widehat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, be a minimal non-empty path in h'' . Suppose that h'' witnesses a (possibly empty) path from $s(x)$ to ℓ_0 , for some $x \in X$. Then, either:

- (ℓ_0, \dots, ℓ_k) is a minimal non-empty path in h . For all $i \in [0, k-1]$, $\ell_i \neq \llbracket t \rrbracket_{s,h}^x$.
- There are $i \in [0, k-1]$ and $j \in [1, k]$ s.t. $\ell_i = \llbracket t \rrbracket_{s,h}^x$, $\ell_j = \text{sby}_{s,h}^x(t)$. The path $(\ell_0, \dots, \ell_i = \ell_0^P, \ell_2^P, \dots, \ell_{m-1}^P, \ell_m^P = \ell_j, \dots, \ell_k)$ is minimal non-empty in h .

Moreover, $\ell_0, \ell_k \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$.

Let us consider the other direction.

- (p8) Let (ℓ_0, \dots, ℓ_k) , where $\ell_0, \ell_k \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, be a minimal non-empty path in h . Suppose that h witnesses a (possibly empty) path from $s(x)$ to ℓ_0 , for some $x \in X$. Then, either:

- (ℓ_0, \dots, ℓ_k) is a minimal non-empty path in h'' . For all $i \in [0, k-1]$, $\ell_i \neq \llbracket t \rrbracket_{s,h}^x$.
- There are $i \in [0, k-1]$ and $j \in [1, k]$ s.t. $\ell_i = \llbracket t \rrbracket_{s,h}^x$, $\ell_j = \text{sby}_{s,h}^x(t)$. The path $(\ell_0, \dots, \ell_i = \widetilde{\ell}_0, \widetilde{\ell}_2, \dots, \widetilde{\ell}_{n-1}, \widetilde{\ell}_n = \ell_j, \dots, \ell_k)$ is minimal non-empty in h'' .

Moreover, $\ell_0, \ell_k \notin \text{dom}(\widehat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$.

The proofs of (p7) and (p8) are very similar. Below, we show the proof of (p7).

Proof of (p7). Notice that $\ell_0, \ell_k \notin \text{Path}[s]_{s,h}^x(t) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. holds directly from (p6), as h'' witnesses paths from $s(x)$ to both $\ell_0, \ell_k \notin \text{dom}(\widehat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$. As done in (p6), we divide the proof depending on whether $\ell_i = \llbracket t \rrbracket_{s,h}^x$, for some $i \in [0, k-1]$.

case: $\ell_i = \llbracket t \rrbracket_{s,h}^x$, for some $i \in [0, k-1]$. Since $\ell_k \notin \text{dom}(\widehat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, we conclude that the path described by $\text{dom}(\widehat{h})$ must be completely included in ρ . Exactly as in (p6), This allows us derive that (ℓ_0, \dots, ℓ_i) and $(\ell_{i+n}, \dots, \ell_k)$, where $\ell_{i+n} = \text{sby}_{s,h}^x(t)$, are two disjoint paths, of both \widehat{h} and h . For every $j \in [0, i-1] \cup [i+n, k-1]$, $\ell_j \in \text{dom}(\widehat{h})$, and therefore $\ell_j \notin \text{Path}[s]_{s,h}^x(t)$. Together with $\ell_i = \llbracket t \rrbracket_{s,h}^x$ and $\ell_{i+n} = \text{sby}_{s,h}^x(t)$, this allows us to conclude that the following is a minimal path in h , going from ℓ_0 to ℓ_k ,

$$(\ell_0, \dots, \ell_i = \ell_0^P, \ell_2^P, \dots, \ell_{m-1}^P, \ell_m^P = \ell_j, \dots, \ell_k),$$

where we recall that $(\ell_0^P, \dots, \ell_m^P)$ is the path described by $\text{Path}[s]_{s,h}^x(t)$.

case: $\ell_i \neq \llbracket t \rrbracket_{s,h}^x$, for all $i \in [0, k-1]$. As $\ell_0 \notin \text{dom}(\widehat{h}) \setminus \{\llbracket t \rrbracket_{s,h}^x\}$, we conclude that $\ell_0 \in \text{dom}(\widehat{h})$. Afterwards, exactly as in the proof of (p6), we are able to conclude that for every $i \in [0, k-1]$, $\ell_i \in \text{dom}(\widehat{h})$. This implies that ρ is a path in h , hence concluding the proof.

As done in the previous cases, we discuss the properties (A)–(F) below.

- A. for all $t'' \in T[s]^X$, $\llbracket t'' \rrbracket_{s,h}^X$ and $\llbracket t'' \rrbracket_{s,h''}^X$ are equidefined. If defined, $\llbracket t \rrbracket_{s,h}^X = \llbracket t \rrbracket_{s,h''}^X$.

Proof of (A). The proof is straightforward when t'' is a program variable. For the cases of meet-point variables and end-point variables (below), the proof essentially relies on (p7) and (p8).

case: $t'' = m(x, y)$. Suppose $\llbracket m(x, y) \rrbracket_{s,h}^X = \ell$. By definition, h witnesses two disjoint non-empty paths ρ and ρ' , where $\rho = (\ell_0, \dots, \ell_{k_1})$ goes from $s(x)$ to ℓ , whereas $\rho' = (\ell'_0, \dots, \ell'_{k_2})$ goes from $s(y)$ to ℓ . Moreover, ℓ does not belong to a cycle of h . Without loss of generality, we assume that if $\llbracket t \rrbracket_{s,h}^X$ belongs to one of these two paths, then it belongs to ρ , i.e. it is in $\{\ell_0, \dots, \ell_{k_1}\}$. Since the two paths are disjoint, this implies that $\llbracket t \rrbracket_{s,h}^X$ does not belong to ρ' . As $\ell \in \text{Lab}[S]_{s,h}^X$, we have $\ell \notin \text{Path}[S]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. We apply (ρ8): ρ' is a path in h'' , and so is one path among ρ or

$$(\ell_0, \dots, \ell_i = \tilde{\ell}_0, \tilde{\ell}_2, \dots, \tilde{\ell}_{n-1}, \tilde{\ell}_n = \ell_j, \dots, \ell_{k_1}),$$

say ρ'' . Moreover, $\ell \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. The paths ρ and ρ'' of h'' are disjoint, with ρ (resp. ρ'') going from $s(x)$ (resp. $s(y)$) to ℓ . In order to conclude that $\ell = \llbracket m(x, y) \rrbracket_{s,h''}^X$, it is sufficient to show that ℓ does not belong to a cycle in h'' . *Ad absurdum*, suppose that this is not the case. Since $\ell \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$ and, in h'' , $s(x)$ reaches ℓ , by (ρ7) we derive that ℓ belongs to a cycle in h . This contradicts the fact that $\ell = \llbracket m(x, y) \rrbracket_{s,h}^X$. Therefore, $\ell = \llbracket m(x, y) \rrbracket_{s,h''}^X$.

The proof of the other direction, i.e. if $\llbracket m(x, y) \rrbracket_{s,h''}^X = \ell$ then $\ell = \llbracket m(x, y) \rrbracket_{s,h}^X$, is symmetrical. One notice that if $\llbracket m(x, y) \rrbracket_{s,h''}^X = \ell$, then, by (ρ5), $\ell \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. This allows us to rely on (ρ7) and (ρ8), as done above.

case: $t'' = e(x)$. Suppose $\llbracket e(x) \rrbracket_{s,h}^X = \ell$. By definition, h witnesses a non-empty path $\rho = (\ell_0, \dots, \ell_{k_1})$ going from $s(x)$ to ℓ . Moreover, if $\ell \in \text{dom}(h)$, then h witnesses a non-empty path $\rho' = (\ell'_0, \dots, \ell'_{k_2})$, disjoint from ρ and going from ℓ to ℓ . As $\ell_{k_1} \in \text{Lab}[S]_{s,h}^X$, we have $\ell_{k_1} \notin \text{Path}[S]_{s,h}^X(t) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. We apply (ρ8), and deduce that either ρ or

$$(\ell_0, \dots, \ell_i = \tilde{\ell}_0, \tilde{\ell}_2, \dots, \tilde{\ell}_{n-1}, \tilde{\ell}_n = \ell_j, \dots, \ell_{k_1}),$$

is a path in h'' . Let ρ'' be this path. Moreover, $\ell_{k_1} \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$. From the semantics of end-point variables, to show that $\ell_{k_1} = \ell = \llbracket e(x) \rrbracket_{s,h''}^X$, we must check that if $\ell_{k_1} \in \text{dom}(h'')$, then there is a non-empty path in h'' , disjoint from ρ'' and going from ℓ_{k_1} to ℓ_{k_1} . Suppose $\ell_{k_1} \in \text{dom}(h'')$. By definition of h'' , either $\ell_{k_1} \in \text{dom}(\tilde{h})$ or $\ell_{k_1} \in \text{dom}(\hat{h})$. In the first case, we conclude that $\ell_{k_1} = \llbracket t \rrbracket_{s,h}^X$ and, as $\text{Path}[S]_{s,h}^X(t) \neq \emptyset$, $\ell_{k_1} \in \text{dom}(h)$. In the second case, again, $\ell_{k_1} \in \text{dom}(h)$, as $\hat{h} \subseteq h$. Therefore, h witnesses the non-empty path $\rho' = (\ell'_0, \dots, \ell'_{k_2})$, disjoint from ρ and going from ℓ_{k_1} to itself. Since ρ' is disjoint from ρ , if $\llbracket t \rrbracket_{s,h}^X$ belongs to ρ , i.e. $\llbracket t \rrbracket_{s,h}^X \in \{\ell_0, \dots, \ell_{k_1-1}\}$, then it does not belong to ρ' . We apply (ρ8), and conclude that h'' witnesses a non-empty path going from ℓ_{k_1} to ℓ_{k_1} , that is disjoint from ρ'' . By definition of end-point variables, $\ell_{k_1} = \ell = \llbracket e(x) \rrbracket_{s,h''}^X$.

As in the case of meet-point variables, the proof in the order direction is symmetrical. Again, we notice that $\llbracket e(x) \rrbracket_{s,h''}^X = \ell$ implies $\ell \notin \text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}$, by (ρ5). This allows us to rely on (ρ7) and (ρ8) to show the result.

Fundamentally, (A) shows that $\llbracket t \rrbracket_{s,h''}^X = \llbracket t \rrbracket_{s,h}^X$ and $\text{sby}_{s,h}^X(t) \in \text{Lab}[S]_{s,h''}^X$. Since, by (ρ5), $\text{Lab}[S]_{s,h''}^X \cap (\text{dom}(\tilde{h}) \setminus \{\llbracket t \rrbracket_{s,h}^X\}) = \emptyset$, by (ρ1) we conclude that

$$\text{Path}[S]_{s,h''}^X(t) = \text{dom}(\tilde{h}). \tag{ρ9}$$

B. For every $t'' \in T[S]^X$,

(a) $\text{sby}_{s,h}^X(t'')$ and $\text{sby}_{s,h''}^X(t'')$ are equidefined. If defined, they are equal,

- (b) If $\llbracket t'' \rrbracket_{s,h}^X \neq \llbracket t \rrbracket_{s,h}^X$ then $\text{Path}[s]_{s,h}^X(t'') = \text{Path}[s]_{s,h''}^X(t'')$. Otherwise, $\text{card}(\text{Path}[s]_{s,h}^X(t)) > \text{card}(\text{Path}[s]_{s,h''}^X(t)) \geq \mathcal{L}(\alpha)$.
- (c) If $h^{\delta_1}(\llbracket t'' \rrbracket_{s,h_j}^X) = s(u)$ for some $\delta_1 \in [0, \text{card}(\text{Path}[s]_{s,h}^X(t''))]$, then there is $\delta_2 \in [0, \text{card}(\text{Path}[s]_{s,h''}^X(t''))]$ such that $h''^{\delta_2}(\llbracket t'' \rrbracket_{s,h''}^X) = s(u)$ and $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha))$, $\min(\text{card}(\text{Path}[s]_{s,h}^X(t'')) - \delta_1, \mathcal{S}_{\text{right}}(\alpha)) = \min(\text{card}(\text{Path}[s]_{s,h''}^X(t'')) - \delta_2, \mathcal{S}_{\text{right}}(\alpha))$.
- (d) If $h''^{\delta_2}(\llbracket t'' \rrbracket_{s,h''}^X) = s(u)$ for some $\delta_2 \in [0, \text{card}(\text{Path}[s]_{s,h''}^X(t''))]$, then there is $\delta_1 \in [0, \text{card}(\text{Path}[s]_{s,h}^X(t''))]$ such that $h^{\delta_1}(\llbracket t'' \rrbracket_{s,h}^X) = s(u)$ and $\min(\delta_1, \mathcal{S}_{\text{left}}(\alpha)) = \min(\delta_2, \mathcal{S}_{\text{left}}(\alpha))$, $\min(\text{card}(\text{Path}[s]_{s,h}^X(t'')) - \delta_1, \mathcal{S}_{\text{right}}(\alpha)) = \min(\text{card}(\text{Path}[s]_{s,h''}^X(t'')) - \delta_2, \mathcal{S}_{\text{right}}(\alpha))$.

We divide the proofs of (a)–(d) depending on whether $\llbracket t \rrbracket_{s,h}^X = \llbracket t'' \rrbracket_{s,h}^X$.

case: $\llbracket t \rrbracket_{s,h}^X = \llbracket t'' \rrbracket_{s,h}^X$. From (A), we deduce $\llbracket t'' \rrbracket_{s,h''}^X = \llbracket t \rrbracket_{s,h''}^X$, which in turn shows that $\text{Path}[s]_{s,h''}^X(t'') = \text{Path}[s]_{s,h''}^X(t)$. Afterwards,

Proof of (a). Directly from (ρ₁) and (ρ₉).

Proof of (b). Follows by (ρ₂) and (ρ₉), together with the assumption

$$\text{card}(\text{Path}[s]_{s,h}^X(t)) \geq \text{card}(\text{Path}[s]_{s',h'}^X(t)) \geq \mathcal{S}(\alpha).$$

Proof of (c) and (d). Follows from (ρ₃). Indeed, (s, h) and (s', h') satisfy the same core formulae of the form $u \in \text{sees}_x(t, t') \geq (\beta_1, \beta_2)$, where $\beta_1 \in [1, \mathcal{S}_{\text{left}}(\alpha)]$ and $\beta_2 \in [1, \mathcal{S}_{\text{right}}(\alpha)]$. From the semantics of these core formulae, we deduce that (s, h) and (s', h') satisfy statements analogous to (c) and (d) (where h' replaces h''). Then, (ρ₃) allows us to show (c) and (d).

case: $\llbracket t \rrbracket_{s,h}^X \neq \llbracket t'' \rrbracket_{s,h}^X$. Let $\rho = (\ell_0 = \llbracket t'' \rrbracket_{s,h}^X, \dots, \ell_k = \text{sby}_{s,h}^X(t''))$, be the minimal path in h that is described by $\text{Path}[s]_{s,h}^X(t'')$. As $\llbracket t \rrbracket_{s,h}^X \neq \llbracket t'' \rrbracket_{s,h}^X$ we have $\text{Path}[s]_{s,h}^X(t) \cap \text{Path}[s]_{s,h}^X(t'') = \emptyset$, and so the locations $\ell_0, \dots, \ell_{k-1}$ do not belong to $\text{Path}[s]_{s,h}^X(t)$. As they belong to $\text{dom}(h)$, by definition of \widehat{h} , they belong to $\text{dom}(\widehat{h})$. By $\widehat{h} \subseteq h''$, we conclude that ρ is a path in h'' . From (A), $\llbracket t'' \rrbracket_{s,h''}^X = \llbracket t \rrbracket_{s,h}^X = \ell_0$, for every $j \in [0, k-1]$, $\ell_j \notin \text{Lab}[s]_{s,h''}^X$ and $\ell_k \in \text{Lab}[s]_{s,h''}^X$. By definition, $\text{Path}[s]_{s,h''}^X(t'')$ is a minimal set describing the path ρ . We conclude that $\text{Path}[s]_{s,h}^X(t'')$ and $\text{Path}[s]_{s,h''}^X(t'')$ describe the same path in h and h'' , respectively. This property generalises (a)–(d).

- C. For every $x \in X$, $\text{Pred}[s]_{s,h}^X(x) = \text{Pred}[s]_{s,h''}^X(x)$

Proof of (C). (\Rightarrow): Suppose $\ell \in \text{Pred}[s]_{s,h}^X(x)$. So, $h(\ell) = s(x)$ and h does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. Since $\ell \notin \text{Path}[s]_{s,h}^X(t)$, we have $\ell \in \text{dom}(\widehat{h})$ and $\widehat{h}(\ell) = s(x)$. By $\widehat{h} \subseteq h''$, $h''(\ell) = s(x)$. From (ρ₆), h'' does not witness a path going from $s(y)$ to ℓ , for any $y \in X$. Therefore, $\ell \in \text{Pred}[s]_{s,h''}^X(x)$.

(\Leftarrow): Analogous to the other direction.

- D. For every $\beta \in [1, \alpha]$, $\text{Cycl}[s]_{s,h}^X(\beta) = \text{Cycl}[s]_{s,h''}^X(\beta)$.

Proof of (D). (\Rightarrow): Let $L \in \text{Cycl}[s]_{s,h}^X(\beta)$. So, L is a minimal set describing an unlabelled cycle in h , of length β . Since $L \cap \text{Path}[s]_{s,h}^X(t) = \emptyset$ and $L \subseteq \text{dom}(h)$, by definition of \widehat{h} we have $L \subseteq \text{dom}(\widehat{h})$, and L is a minimal set describing a cycle in \widehat{h} . By $\widehat{h} \subseteq h''$, L is a minimal set describing a cycle in h'' . From (A), $L \cap \text{Lab}[s]_{s,h''}^X = \emptyset$. So $L \in \text{Cycl}[s]_{s,h''}^X(\beta)$.

(\Leftarrow): Symmetrical to the other direction.

$$\text{E. } \uparrow\text{Cycl}[S]_{s,h}^{X,\alpha} = \uparrow\text{Cycl}[S]_{s,h''}^{X,\alpha}.$$

Proof of (E). Analogous to the proof of (D).

$$\text{F. } \text{Rem}[S]_{s,h''}^{X,\alpha} = \text{Rem}[S]_{s,h}^{X,\alpha}.$$

Proof of (F). (\Rightarrow): Let $\ell \in \text{Rem}[S]_{s,h}^{X,\alpha}$. So, $\ell \in \text{dom}(h)$ and moreover

$$\begin{aligned} \ell &\notin \text{Lab}[S]_{s,h}^X, & \ell &\notin \text{Path}[S]_{s,h}^X(t''), & \ell &\notin \text{Pred}[S]_{s,h}^X(x), \\ \ell &\notin [\text{Cycl}[S]_{s,h}^X(\beta')]^\flat, & \ell &\notin [\uparrow\text{Cycl}[S]_{s,h}^X]^\flat, \end{aligned}$$

where $t'' \in T[S]^X$, $x \in X$ and $\beta' \in [1, \alpha]$. From (A)–(E) we conclude that

$$\begin{aligned} \ell &\notin \text{Lab}[S]_{s,h}^X, & \ell &\notin \text{Path}[S]_{s,h''}^X(t''), & \ell &\notin \text{Pred}[S]_{s,h''}^X(x), \\ \ell &\notin [\text{Cycl}[S]_{s,h''}^X(\beta')]^\flat, & \ell &\notin [\uparrow\text{Cycl}[S]_{s,h''}^{X,\alpha}]^\flat, \end{aligned}$$

where $x \in X$, $\beta' \in [1, \alpha]$, and $t'' \in T[S]^X$ is such that $\llbracket t'' \rrbracket_{s,h''}^X \neq \llbracket t \rrbracket_{s,h}^X$.

Moreover, from $\ell \notin \text{Path}[S]_{s,h}^X(t)$, we have $\ell \in \text{dom}(\hat{h})$. Together with (ρ₉), this implies $\ell \in \text{dom}(h'')$ and $\ell \notin \text{Path}[S]_{s,h''}^X(t)$. So, $\ell \in \text{Rem}[S]_{s,h''}^{X,\alpha}$.

(\Leftarrow): Analogous to the other direction.

We show $(s, h) \approx_{X,\alpha}^S (s, h'')$, $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$, and $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$. By transitivity of the hop relation $\leftrightarrow_{X,\alpha}^S$, the last two memberships imply $(s, h) \leftrightarrow_{X,\alpha}^S (s', h')$.

Proof of $(s, h) \approx_{X,\alpha}^S (s, h'')$. By (A)–(F) (see for instance the proof of Lemma 5.39).

Proof of $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$. Directly from the properties (A)–(F), we conclude that

$$h \setminus \{(\ell, \ell') \in h \mid \ell \in \text{Path}[S]_{s,h}^X(t)\} = h'' \setminus \{(\ell, \ell') \in h'' \mid \ell \in \text{Path}[S]_{s,h''}^X(t)\}.$$

In particular, notice that this heaps corresponds to the heap \hat{h} built in the strategy used to define h'' . By (A) and $(s, h) \approx_{X,\alpha}^S (s, h'')$, we can apply Lemma 5.40(II) and derive that $(s, h) \leftrightarrow_{X,\alpha}^S (s, h'')$.

Proof of $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$. As $\approx_{X,\alpha}^S$ is an equivalence relation, by $(s, h) \approx_{X,\alpha}^S (s, h'')$ we conclude that $(s, h'') \approx_{X,\alpha}^S (s', h')$. From the properties (A)–(F), we derive $[(s, h'') \#_{X,\alpha} (s', h')] < [(s, h) \#_{X,\alpha} (s', h')]$. In particular, (ρ₂) and (ρ₉) imply $\text{card}(\text{Path}[S]_{s,h''}^X(t)) = \text{card}(T)$. By induction hypothesis, $(s, h'') \leftrightarrow_{X,\alpha}^S (s', h')$. \square

D

Appendix of Chapter 6

Contents

Proof of Lemma 6.12	553
Proof of Lemma 6.19	554

Proof of Lemma 6.12

Lemma 6.12. The following formulae are theorem of $\mathcal{H}_C(*)$:

- ($I_{6.12.1}^*$) $x \sim y \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge x \sim y) * \psi \quad \blacktriangleleft [\sim \in \{=, \neq\}]$
- ($I_{6.12.2}^*$) $x = y \wedge ((\varphi \wedge x \hookrightarrow -) * \psi) \Rightarrow (\varphi \wedge y \hookrightarrow -) * \psi$
- ($I_{6.12.3}^*$) $(\varphi \wedge x \hookrightarrow -) * \psi \Rightarrow \varphi * (\psi \wedge \neg x \hookrightarrow -)$
- ($I_{6.12.4}^*$) $\neg x \hookrightarrow - \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge \neg x \hookrightarrow -) * \psi$
- ($I_{6.12.5}^*$) $x \hookrightarrow - \wedge (\varphi * (\neg x \hookrightarrow - \wedge \psi)) \Rightarrow (\varphi \wedge x \hookrightarrow -) * (\neg x \hookrightarrow - \wedge \psi)$
- ($I_{6.12.6}^*$) $x \hookrightarrow y \wedge ((\varphi \wedge x \hookrightarrow -) * \psi) \Rightarrow (\varphi \wedge x \hookrightarrow y) * \psi$
- ($I_{6.12.7}^*$) $\neg x \hookrightarrow y \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge \neg x \hookrightarrow y) * \psi.$

The proofs of ($I_{6.12.1}^*$), ($I_{6.12.3}^*$) and ($I_{6.12.5}^*$) were already established during Chapter 6.

Proof of ($I_{6.12.2}^$).*

1	$x \hookrightarrow - \wedge x = y \Rightarrow y \hookrightarrow -$	(\equiv_{SUB})
2	$x = y \wedge ((\varphi \wedge x \hookrightarrow -) * \psi) \Rightarrow ((\varphi \wedge x \hookrightarrow - \wedge x = y) * \psi)$	$(I_{6.12.1}^*)$
3	$(\varphi \wedge x \hookrightarrow - \wedge x = y) * \psi \Rightarrow (\varphi \wedge y \hookrightarrow -) * \psi$	PC, (*), 1
4	$x = y \wedge ((\varphi \wedge x \hookrightarrow -) * \psi) \Rightarrow (\varphi \wedge y \hookrightarrow -) * \psi$	$(\Rightarrow \text{TR})$, 2, 3 \square

Proof of ($I_{6.12.4}^$).*

1	$\varphi \Rightarrow (\varphi \wedge x \hookrightarrow -) \vee (\varphi \wedge \neg x \hookrightarrow -)$	PC
2	$\varphi * \psi \Rightarrow ((\varphi \wedge x \hookrightarrow -) \vee (\varphi \wedge \neg x \hookrightarrow -)) * \psi$	(*), 1
3	$((\varphi \wedge x \hookrightarrow -) \vee (\varphi \wedge \neg x \hookrightarrow -)) * \psi \Rightarrow$ $((\varphi \wedge x \hookrightarrow -) * \psi) \vee ((\varphi \wedge \neg x \hookrightarrow -) * \psi)$	(I_3^*)
4	$\varphi \wedge x \hookrightarrow - \Rightarrow x \hookrightarrow -$	PC
5	$\psi \Rightarrow \top$	PC
6	$(\varphi \wedge x \hookrightarrow -) * \psi \Rightarrow (x \hookrightarrow - * \top)$	$(*\text{I}_{LR})$, 4, 5
7	$x \hookrightarrow - * \top \Rightarrow x \hookrightarrow -$	(I_5^*)
8	$\varphi * \psi \Rightarrow x \hookrightarrow - \vee ((\varphi \wedge \neg x \hookrightarrow -) * \psi)$	PC, 2, 3, 6, 7
9	$\neg x \hookrightarrow - \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge \neg x \hookrightarrow -) * \psi$	PC, 8 \square

Proof of ($I_{6.12.6}^$).*

1	$\varphi \wedge x \hookrightarrow - \Rightarrow (\varphi \wedge x \hookrightarrow - \wedge x \hookrightarrow y) \vee (\varphi \wedge x \hookrightarrow - \wedge \neg x \hookrightarrow y)$	PC
2	$(\varphi \wedge x \hookrightarrow -) * \psi \Rightarrow ((\varphi \wedge x \hookrightarrow - \wedge x \hookrightarrow y) \vee (\varphi \wedge x \hookrightarrow - \wedge \neg x \hookrightarrow y)) * \psi$	(*), 1

3	$(\varphi \wedge x \hookrightarrow _) * \psi \Rightarrow ((\varphi \wedge x \hookrightarrow _ \wedge x \hookrightarrow y) * \psi) \vee ((\varphi \wedge x \hookrightarrow _ \wedge \neg x \hookrightarrow y) * \psi)$	$(I_3^*), (\Rightarrow \text{TR}), 2$
4	$\varphi \wedge x \hookrightarrow _ \wedge \neg x \hookrightarrow y \Rightarrow x \hookrightarrow _ \wedge \neg x \hookrightarrow y$	PC
5	$\psi \Rightarrow \top$	PC
6	$(\varphi \wedge x \hookrightarrow _ \wedge \neg x \hookrightarrow y) * \psi \Rightarrow (x \hookrightarrow _ \wedge \neg x \hookrightarrow y) * \top$	$(*\text{I}_{LR})$
7	$(x \hookrightarrow _ \wedge \neg x \hookrightarrow y) * \top \Rightarrow \neg x \hookrightarrow y$	(\neg_{PTO}^*)
8	$(\varphi \wedge x \hookrightarrow _) * \psi \Rightarrow ((\varphi \wedge x \hookrightarrow _ \wedge x \hookrightarrow y) * \psi) \vee \neg x \hookrightarrow y$	PC, 3, 6, 7
9	$x \hookrightarrow y \wedge ((x \hookrightarrow _ \wedge \varphi) * \psi) \Rightarrow (\varphi \wedge x \hookrightarrow _ \wedge x \hookrightarrow y) * \psi$	PC, 8
10	$\varphi \wedge x \hookrightarrow _ \wedge x \hookrightarrow y \Rightarrow \varphi \wedge x \hookrightarrow y$	PC
11	$(\varphi \wedge x \hookrightarrow _ \wedge x \hookrightarrow y) * \psi \Rightarrow (\varphi \wedge x \hookrightarrow y) * \psi$	$(*), 10$
12	$x \hookrightarrow y \wedge ((x \hookrightarrow _ \wedge \varphi) * \psi) \Rightarrow (\varphi \wedge x \hookrightarrow y) * \psi$	$(\Rightarrow \text{TR}), 9, 11 \quad \square$

Proof of $(I_{6.12.7}^)$.* Similar to the proof of $(I_{6.12.4}^*)$, by replacing $x \hookrightarrow _$ with $x \hookrightarrow y$.

1	$\varphi \Rightarrow (\varphi \wedge x \hookrightarrow y) \vee (\varphi \wedge \neg x \hookrightarrow y)$	PC
2	$\varphi * \psi \Rightarrow ((\varphi \wedge x \hookrightarrow y) * \psi) \vee ((\varphi \wedge \neg x \hookrightarrow y) * \psi)$	$(*), 1, (I_3^*)$
3	$\varphi \wedge x \hookrightarrow y \Rightarrow x \hookrightarrow y$	PC
4	$\psi \Rightarrow \top$	PC
5	$(\varphi \wedge x \hookrightarrow y) * \psi \Rightarrow (x \hookrightarrow y * \top)$	$(*\text{I}_{LR}), 3, 4$
6	$x \hookrightarrow y * \top \Rightarrow x \hookrightarrow y$	(MONO^*)
7	$\varphi * \psi \Rightarrow x \hookrightarrow y \vee ((\varphi \wedge \neg x \hookrightarrow y) * \psi)$	PC, 2, 5, 6
8	$\neg x \hookrightarrow y \wedge (\varphi * \psi) \Rightarrow (\varphi \wedge \neg x \hookrightarrow y) * \psi$	PC, 7 $\quad \square$

Proof of Lemma 6.19

Lemma 6.19. The following axioms and rules are admissible in $\mathcal{H}_C(*, -*)$:

$(I_{6.19.1}^-) \quad (\perp \circledast \varphi) \Rightarrow \perp$ $(I_{6.19.2}^-) \quad (\varphi \circledast \perp) \Rightarrow \perp$ $(I_{6.19.3}^-) \quad \varphi * (\varphi \multimap \psi) \Rightarrow \psi$ $(I_{6.19.4}^-) \quad \frac{\varphi \Rightarrow \psi}{(\varphi \circledast \chi) \Rightarrow (\psi \circledast \chi)}$ $(I_{6.19.5}^-) \quad \frac{\varphi \Rightarrow \psi}{(\chi \circledast \varphi) \Rightarrow (\chi \circledast \psi)}$	$(I_{6.19.6}^*) \quad (\varphi \vee \psi) \circledast \chi \Leftrightarrow (\varphi \circledast \chi) \vee (\psi \circledast \chi)$ $(I_{6.19.7}^*) \quad \chi \circledast (\varphi \vee \psi) \Leftrightarrow (\chi \circledast \varphi) \vee (\chi \circledast \psi)$ $(I_{6.19.8}^*) \quad \varphi \circledast (\psi \circledast \chi) \Leftrightarrow (\varphi * \psi) \circledast \chi$ $(I_{6.19.9}^*) \quad (\varphi \circledast \psi) \wedge (\varphi \multimap \chi) \Rightarrow (\varphi \circledast \psi \wedge \chi)$ $(I_{6.19.10}^*) \quad x \sim y \wedge (\varphi \circledast \psi) \Rightarrow (\varphi \wedge x \sim y \circledast \psi) \quad \blacktriangleleft [\sim \in \{=, \neq\}]$
--	---

The proofs of $(I_{6.19.3}^-)$, $(I_{6.19.4}^-)$, $(I_{6.19.6}^*)$ and $(I_{6.19.10}^*)$ were already established during Chapter 6.

Proof of $(I_{6.19.1}^-)$.

1	$\perp * \top \Rightarrow \perp$	(I_4^*)	4	$\top \Rightarrow (\perp * \neg \varphi)$	$(\text{COM}^*), (\neg_2)$
2	$\perp \Rightarrow \neg \varphi$	PC	5	$\top \Rightarrow \neg(\perp * \varphi)$	Def. $\neg *$, PC
3	$\perp * \top \Rightarrow \neg \varphi$	$(\Rightarrow \text{TR})$, 1, 2	6	$(\perp * \varphi) \Rightarrow \perp$	5, PC \square

Proof of $(I_{6.19.2}^)$.*

1	$\top * \varphi \Rightarrow \top$	PC	3	$\neg(\varphi * \top) \Rightarrow \perp$	PC, 2
2	$\top \Rightarrow (\varphi * \top)$	(\neg_2)	4	$(\varphi * \perp) \Rightarrow \perp$	Def. $\neg *$, PC

Note that implicitly, we have assumed that we can replace $\neg \top$ by \perp in the scope of $\neg *$ or $*$, which is possible as the replacement of equivalents holds in the calculus $\mathcal{H}_C(*, \neg *)$ (see the proof of Theorem 6.22). \square

Proof of $(I_{6.19.5}^)$.*

1	$\varphi \Rightarrow \psi$	Hypothesis
2	$\neg \psi \Rightarrow \neg \varphi$	PC, 1
3	$\chi * (\chi * \neg \psi) \Rightarrow \neg \psi$	$(I_{6.19.3}^*)$
4	$\chi * (\chi * \neg \psi) \Rightarrow \neg \varphi$	$(\Rightarrow \text{TR})$, 3, 2
5	$(\chi * \neg \psi) * \chi \Rightarrow \chi * (\chi * \neg \psi)$	(COM^*)
6	$(\chi * \neg \psi) * \chi \Rightarrow \neg \varphi$	$(\Rightarrow \text{TR})$, 4, 5
7	$(\chi * \neg \psi) \Rightarrow (\chi * \neg \varphi)$	(\neg_2) , 6
8	$\neg(\chi * \neg \varphi) \Rightarrow \neg(\chi * \neg \psi)$	PC, 7
9	$(\chi * \varphi) \Rightarrow (\chi * \psi)$	Def. $* \neg$ \square

Proof of $(I_{6.19.7}^)$.* We handle each implication separately, and we follow a pattern similar to the one used in the proof of $(I_{6.19.6}^*)$.

1	$\chi * (\chi * \neg \varphi) \Rightarrow \neg \varphi$	$(I_{6.19.3}^*)$
2	$((\chi * \neg \varphi) \wedge (\chi * \neg \psi)) \Rightarrow \chi * \neg \varphi$	PC
3	$\chi * ((\chi * \neg \varphi) \wedge (\chi * \neg \psi)) \Rightarrow \chi * (\chi * \neg \varphi)$	(ILR^*) , 2
4	$\chi * ((\chi * \neg \varphi) \wedge (\chi * \neg \psi)) \Rightarrow \neg \varphi$	$(\Rightarrow \text{TR})$, 3, 1
5	$\chi * (\chi * \neg \psi) \Rightarrow \neg \psi$	$(I_{6.19.3}^*)$
6	$((\chi * \neg \varphi) \wedge (\chi * \neg \psi)) \Rightarrow \chi * \neg \psi$	PC
7	$\chi * ((\chi * \neg \varphi) \wedge (\chi * \neg \psi)) \Rightarrow \chi * (\chi * \neg \psi)$	(ILR^*) , 6
8	$\chi * ((\chi * \neg \varphi) \wedge (\chi * \neg \psi)) \Rightarrow \neg \psi$	$(\Rightarrow \text{TR})$, 7, 5

9	$\chi * ((\chi \rightarrow \neg \varphi) \wedge (\chi \rightarrow \neg \psi)) \Rightarrow \neg(\varphi \vee \psi)$	PC, 4, 8
10	$((\chi \rightarrow \neg \varphi) \wedge (\chi \rightarrow \neg \psi)) * \chi \Rightarrow \neg(\varphi \vee \psi)$	$(_{\text{COM}}^*)$, $(\Rightarrow \text{TR})$, 9
11	$((\chi \rightarrow \neg \varphi) \wedge (\chi \rightarrow \neg \psi)) \Rightarrow (\chi \rightarrow \neg(\varphi \vee \psi))$	(\neg_2) , 10
12	$\neg(\chi \rightarrow \neg(\varphi \vee \psi)) \Rightarrow \neg(\chi \rightarrow \neg \varphi) \vee \neg(\chi \rightarrow \neg \psi)$	PC, 11
13	$\chi \dashv (\varphi \vee \psi) \Rightarrow (\chi \dashv \varphi) \vee (\chi \dashv \psi)$	Def. \dashv , 12

The derivation of the other implication can be found below.

1	$\varphi \Rightarrow \varphi \vee \psi$	PC
2	$\psi \Rightarrow \varphi \vee \psi$	PC
3	$(\chi \dashv \varphi) \Rightarrow (\chi \dashv \varphi \vee \psi)$	$(I_{6.19.5}^*)$, 1
4	$(\chi \dashv \psi) \Rightarrow (\chi \dashv \varphi \vee \psi)$	$(I_{6.19.5}^*)$, 2
5	$((\chi \dashv \varphi) \vee (\chi \dashv \psi)) \Rightarrow (\chi \dashv \varphi \vee \psi)$	PC, 3, 4 \square

Proof of $(I_{6.19.8}^)$.* By definition of the subtraction operator \dashv , $(I_{6.19.8}^*)$ is equivalent to

$$(\varphi \rightarrow (\psi \rightarrow \neg \chi)) \Leftrightarrow ((\varphi * \psi) \rightarrow \neg \chi).$$

This equivalence is provable in $\mathcal{H}_C(*, \rightarrow)$, thanks to the adjunction rules.

1	$(\varphi * \psi) * ((\varphi * \psi) \rightarrow \neg \chi) \Rightarrow \neg \chi$	$(I_{6.19.3}^*)$
2	$\psi * (\varphi * ((\varphi * \psi) \rightarrow \neg \chi)) \Rightarrow \neg \chi$	$(_{\text{COM}}^*)$, $(_{\text{ASSOC}}^*)$, 1
3	$\varphi * ((\varphi * \psi) \rightarrow \neg \chi) \Rightarrow (\psi \rightarrow \neg \chi)$	(\neg_2) , 2
4	$((\varphi * \psi) \rightarrow \neg \chi) \Rightarrow (\varphi \rightarrow (\psi \rightarrow \neg \chi))$	(\neg_2) , 3, $(_{\text{COM}}^*)$
5	$\varphi * (\varphi \rightarrow (\psi \rightarrow \neg \chi)) \Rightarrow (\psi \rightarrow \neg \chi)$	$(I_{6.19.3}^*)$
6	$\psi * (\varphi * (\varphi \rightarrow (\psi \rightarrow \neg \chi))) \Rightarrow \neg \chi$	(\neg_1) , 5, $(_{\text{COM}}^*)$, $(_{\text{ASSOC}}^*)$
7	$(\varphi * \psi) * (\varphi \rightarrow (\psi \rightarrow \neg \chi)) \Rightarrow \neg \chi$	$(_{\text{COM}}^*)$, $(_{\text{ASSOC}}^*)$, 6
8	$(\varphi \rightarrow (\psi \rightarrow \neg \chi)) \Rightarrow ((\varphi * \psi) \rightarrow \neg \chi)$	(\neg_2) , 7
9	$(\varphi \rightarrow (\psi \rightarrow \neg \chi)) \Leftrightarrow ((\varphi * \psi) \rightarrow \neg \chi)$	PC, 4, 8 \square

Proof of $(I_{6.19.9}^)$.*

1	$\varphi * (\varphi \rightarrow \chi) \Rightarrow \chi$	$(I_{6.19.3}^*)$
2	$\varphi * (\varphi \rightarrow \neg(\psi \wedge \chi)) \Rightarrow \neg(\psi \wedge \chi)$	$(I_{6.19.3}^*)$
3	$(\varphi * (\varphi \rightarrow \chi)) \wedge (\varphi * (\varphi \rightarrow \neg(\psi \wedge \chi))) \Rightarrow \neg \psi$	PC, 1, 2
4	$\varphi * ((\varphi \rightarrow \chi) \wedge (\varphi \rightarrow \neg(\psi \wedge \chi))) \Rightarrow (\varphi * (\varphi \rightarrow \chi)) \wedge (\varphi * (\varphi \rightarrow \neg(\psi \wedge \chi)))$	$(*\text{ILR})$, PC

5	$\varphi * ((\varphi -* \chi) \wedge (\varphi -* \neg(\psi \wedge \chi))) \Rightarrow \neg\psi$	$(\Rightarrow \text{TR}), 4$
6	$(\varphi -* \chi) \wedge (\varphi -* \neg(\psi \wedge \chi)) \Rightarrow (\varphi -* \neg\psi)$	$(\text{COM}^*), (\neg*_2), 5$
7	$(\varphi -* \chi) \wedge \neg(\varphi -* \neg\psi) \Rightarrow \neg(\varphi -* \neg(\psi \wedge \chi))$	PC
8	$(\varphi -* \chi) \wedge (\varphi \dashv \psi) \Rightarrow (\varphi \dashv (\psi \wedge \chi))$	Def. \dashv , 7 \square

E

Appendix of Chapter 7

Contents

Proof of Lemma 7.4	561
Proof of Lemma 7.15.	562

Proof of Lemma 7.4.

To prove Lemma 7.4, we first introduce the notion of (w_1, w_2) -isomorphic finite forests, and show that isomorphic finite forests satisfy the same formulae of $\text{ML}(\|)$.

Definition E.1 $((w_1, w_2)$ -isomorphic forests). Let $\mathcal{K}_1 = (\mathcal{W}_1, R_1, \mathcal{V}_1)$ and $\mathcal{K}_2 = (\mathcal{W}_2, R_2, \mathcal{V}_2)$ be two Kripke-style finite forests. Let $w_1 \in \mathcal{W}_1$ and $w_2 \in \mathcal{W}_2$. \mathcal{K}_1 and \mathcal{K}_2 are said to be (w_1, w_2) -isomorphic whenever there is a bijection $f : w_1 \rightarrow w_2$ such that

1. for every $w \in \mathcal{W}_1$ and $p \in \text{AP}$, $w \in \mathcal{V}_1(p)$ if and only if $f(w) \in \mathcal{V}_2(f(w))$.
2. $f(w_1) = w_2$,
3. for every $w \in R_1^*(w_1)$ and $w' \in R_1(w)$, $f(w') \in R_2(f(w))$.
4. for every $w \in R_2^*(w_2)$ and $w' \in R_2(w)$, $f^{-1}(w') \in R_1(f^{-1}(w))$.

The bijection f is said to be a (w_1, w_2) -isomorphism from \mathcal{K}_1 to \mathcal{K}_2 .

Lemma E.2. Let (\mathcal{K}_1, w_1) and (\mathcal{K}_2, w_2) be two pointed forests, where $\mathcal{K}_1 = (\mathcal{W}_1, R_1, \mathcal{V}_1)$ and $\mathcal{K}_2 = (\mathcal{W}_2, R_2, \mathcal{V}_2)$, and suppose the two Kripke-style finite forests \mathcal{K}_1 and \mathcal{K}_2 to be (w_1, w_2) -isomorphic. For every formula φ in $\text{ML}(\|)$,

$$(\mathcal{K}_1, w_1) \models \varphi \text{ if and only if } (\mathcal{K}_2, w_2) \models \varphi.$$

Proof. We show the left to right direction of the lemma (the other direction holds by symmetry). Let f be the (w_1, w_2) -isomorphism from \mathcal{K}_1 to \mathcal{K}_2 . The proof carries out with a standard structural induction on φ . We omit the trivial cases of \top and Boolean connectives.

base case: $\varphi = p$ for some $p \in \text{AP}$. Directly from the properties (1) and (2) of f .

induction step: $\varphi = \Diamond\psi$. Suppose $(\mathcal{K}_1, w_1) \models \Diamond\psi$. Thus, there is $w'_1 \in R_1(w_1)$ such that $(\mathcal{K}_1, w'_1) \models \psi$. From the property (3) of f , There is $w'_2 \in R_2(w_2)$ such that $f(w'_1) = w'_2$. Clearly, f is a (w'_1, w'_2) -isomorphism from \mathcal{K}_1 to \mathcal{K}_2 . By induction hypothesis, $(\mathcal{K}_1, w'_1) \models \psi$. We conclude that $(\mathcal{K}_2, w_2) \models \Diamond\psi$.

induction step: $\varphi = \psi \parallel \chi$. Suppose $(\mathcal{K}_1, w_1) \models \psi \parallel \chi$, and thus there are two Kripke-style finite forests $\mathcal{K}'_1 = (\mathcal{W}_1, R'_1, \mathcal{V}_1)$ and $\mathcal{K}''_1 = (\mathcal{W}_1, R''_1, \mathcal{V}_1)$ such that $\mathcal{K}_1 = \mathcal{K}'_1 +_{w_1} \mathcal{K}''_1$, $(\mathcal{K}'_1, w_1) \models \psi$ and $(\mathcal{K}''_1, w_1) \models \chi$. We consider the two relations R'_2 and R''_2 defined below

$$R'_2 = \{(w, w') \in R_2 \mid (f^{-1}(w), f^{-1}(w')) \in R'_1\} \quad R''_2 = R_2 \setminus R'_1.$$

Clearly, $R'_2 \cap R''_2 = \emptyset$ and $R'_2 \cup R''_2 = R_2$. Let $\mathcal{K}'_2 = (\mathcal{W}_2, R'_2, \mathcal{V}_2)$ and $\mathcal{K}''_2 = (\mathcal{W}_2, R''_2, \mathcal{V}_2)$. Since f is a (w_1, w_2) -isomorphism from \mathcal{K}_1 to \mathcal{K}_2 , by definition of \mathcal{K}'_2 and \mathcal{K}''_2 it is easy to check that f is also a (w_1, w_2) -isomorphism from \mathcal{K}'_1 to \mathcal{K}'_2 , as well as a (w_1, w_2) -isomorphism from \mathcal{K}''_1 to \mathcal{K}''_2 . Together with $\mathcal{K} = \mathcal{K}'_1 +_{w_1} \mathcal{K}''_1$, we conclude that $\mathcal{K}_2 = \mathcal{K}'_2 +_{w_2} \mathcal{K}''_2$. By induction hypothesis, $(\mathcal{K}'_2, w_2) \models \psi$ and $(\mathcal{K}''_2, w_2) \models \chi$. Thus, $(\mathcal{K}_2, w_2) \models \psi \parallel \chi$. \square

Lemma E.2 directly implies Lemma 7.4, as showed below.

Lemma 7.4. Let $w \in \mathcal{W}$. Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ be two Kripke-style finite forests such that $\mathcal{K}' \subseteq_w \mathcal{K}$. For every $w' \in R'(w)$ and every formula φ in $\text{ML}(\|)$ we have

$$(\mathcal{K}, w') \models \varphi \text{ if and only if } (\mathcal{K}', w') \models \varphi.$$

Proof. By definition of $+_w$, the tree rooted at w' in \mathcal{K} is the one rooted at w' in \mathcal{K}' , i.e.

$$\{(w_1, w_2) \in R \mid w_1 \in R^*(w')\} = \{(w_1, w_2) \in R' \mid w_1 \in R'^*(w')\}.$$

Thus, let f be a bijection from \mathcal{W} to \mathcal{W} such that for every $w'' \in R^*(w')$, $f(w'') = w''$. f is a (w', w') -isomorphism from \mathcal{K} to \mathcal{K}' . By Lemma E.2, $(\mathcal{K}, w') \models \varphi$ if and only if $(\mathcal{K}', w') \models \varphi$. \square

Proof of Lemma 7.15.

Lemma 7.15. The following axioms and rules are admissible in $\mathcal{H}_{\text{GML}}(\mathbf{I})$:

$$\begin{array}{lll} (\Rightarrow \text{TR}) \quad \frac{\varphi \Rightarrow \chi \quad \chi \Rightarrow \psi}{\varphi \Rightarrow \psi} & (I_{7.15.1}^*) \quad \Diamond_{\geq k_1} \varphi \parallel \Diamond_{\geq k_2} \varphi \Rightarrow \Diamond_{\geq k_1 + k_2} \varphi \\ & (I_{7.15.2}^*) \quad \Diamond_{\geq k} \varphi \Rightarrow \Diamond_{=k} \varphi \parallel \top \\ (\mathbf{I}_{LR}) \quad \frac{\varphi \Rightarrow \varphi' \quad \psi \Rightarrow \psi'}{\varphi \parallel \psi \Rightarrow \varphi' \parallel \psi'} & (I_{7.15.3}^*) \quad \Box \varphi \wedge (\psi \parallel \chi) \Rightarrow (\psi \wedge \Box \varphi) \parallel (\chi \wedge \Box \varphi) \\ & (I_{7.15.4}^*) \quad \Box \varphi_1 \parallel \dots \parallel \Box \varphi_n \parallel (\Box \neg \varphi_1 \wedge \dots \wedge \Box \neg \varphi_n) \end{array}$$

The rule $(\Rightarrow \text{TR})$ is admissible by propositional reasoning.

Proof of (\mathbf{I}_{LR}) .

1	$\varphi \Rightarrow \varphi'$	Hypothesis	5	$\varphi' \parallel \psi \Rightarrow \psi \parallel \varphi'$	(C_{COM})
2	$\psi \Rightarrow \psi'$	Hypothesis	6	$\psi' \parallel \varphi' \Rightarrow \varphi' \parallel \psi'$	(C_{COM})
3	$\varphi \parallel \psi \Rightarrow \varphi' \parallel \psi$	(C) , 1	7	$\varphi \parallel \psi \Rightarrow \psi \parallel \varphi'$	$(\Rightarrow \text{TR})$, 3, 5
4	$\psi \parallel \varphi' \Rightarrow \psi' \parallel \varphi'$	(C) , 2	8	$\varphi \parallel \psi \Rightarrow \varphi' \parallel \psi'$	$(\Rightarrow \text{TR})$ twice, 7, 4, 6 \square

Proof of $(I_{7.15.1}^)$.* The proof is divided in the following four cases:

- $k_1 + k_2 = 0$,
- $k_1 \geq 1$ and $k_2 \geq 1$.
- $k_1 \geq 1$ and $k_2 = 0$,
- $k_2 \geq 1$ and $k_1 = 0$.

case: $k_1 + k_2 = 0$. By definition, $\Diamond_{\geq k_1 + k_2} \varphi$ is \top . $(I_{7.15.1}^*)$ follows by propositional reasoning.

case: $k_1 \geq 1$ and $k_2 \geq 1$. In this case, $\Diamond_{\geq k_1} \varphi \parallel \Diamond_{\geq k_2} \varphi$ and $\Diamond_{\geq k_1 + k_2} \varphi$ are syntactically equivalent, up to associativity of \parallel . So, $(I_{7.15.1}^*)$ is derivable by propositional reasoning and $(\text{C}_{\text{ASSOC}})$.

case: $k_1 \geq 1$ and $k_2 = 0$. We have $\Diamond_{\geq k_2} \varphi = \top$. We show $\vdash_{\mathcal{H}_{\text{GML}}(\mathbf{I})} \Diamond_{\geq k_1} \varphi \parallel \top \Rightarrow \Diamond_{\geq k_1} \varphi$ by induction on k_1 .

base case: $k_1 = 1$. In this case, $(I_{7.15.1}^*)$ corresponds to (C_{MONO}) , instantiated with $e = \Diamond \varphi$.

induction step: $k_1 \geq 2$.

1	$\Diamond_{\geq k_1 - 1} \varphi \parallel \top \Rightarrow \Diamond_{\geq k_1 - 1} \varphi$	Induction Hypothesis
2	$\Diamond_{\geq k_1} \varphi \Rightarrow \Diamond \varphi \parallel \Diamond_{\geq k_1 - 1} \varphi$	$(\text{C}_{\text{ASSOC}})$, def. of $\Diamond_{\geq k_1}$
3	$\Diamond_{\geq k_1} \varphi \parallel \top \Rightarrow (\Diamond \varphi \parallel \Diamond_{\geq k_1 - 1} \varphi) \parallel \top$	(C) , 2
4	$(\Diamond \varphi \parallel \Diamond_{\geq k_1 - 1} \varphi) \parallel \top \Rightarrow \Diamond \varphi \parallel (\Diamond_{\geq k_1 - 1} \varphi \parallel \top)$	$(\text{C}_{\text{ASSOC}})$
5	$\Diamond \varphi \parallel (\Diamond_{\geq k_1 - 1} \varphi \parallel \top) \Rightarrow \Diamond \varphi \parallel \Diamond_{\geq k_1 - 1} \varphi$	(C_{COM}) , (C) , 1
6	$\Diamond \varphi \parallel \Diamond_{\geq k_1 - 1} \varphi \Rightarrow \Diamond_{\geq k_1} \varphi$	Previous case of the proof, as $k_1 - 1 \geq 1$
7	$\Diamond_{\geq k_1} \varphi \parallel \top \Rightarrow \Diamond_{\geq k_1} \varphi$	$(\Rightarrow \text{TR})$, 3, 4, 5, 6

case: $k_2 \geq 1$ and $k_1 = 0$. Symmetrical to the previous case, thanks to the commutativity of the composition operator (axiom (COM)). \square

Proof of $(I_{7.15.2}^)$.* By induction on k , with base cases for $k = 0$ and $k = 1$.

base case: $k = 0$. In this case, $(I_{7.15.2}^*)$ corresponds to $\top \Rightarrow (\top \wedge \neg \Diamond_{\geq 1}\varphi) \parallel \top$. Notice that, the formula $\Box \perp \Rightarrow \top \wedge \neg \Diamond_{\geq 1}\varphi$ is valid in GML, and thus it is derivable in $\mathcal{H}_{\text{GML}}(\mathbf{I})$ (since this system extends \mathcal{H}_{GML}). We have,

1	$\Box \perp \Rightarrow \top \wedge \neg \Diamond_{\geq 1}\varphi$	See above
2	$\Diamond_{\geq 0}\varphi \Rightarrow \Diamond_{\geq 0}\varphi \parallel \Box \perp$	(ID)
3	$\Diamond_{\geq 0}\varphi \parallel \Box \perp \Rightarrow \Box \perp \parallel \Diamond_{\geq 0}\varphi$	(COM)
4	$\Diamond_{\geq 0}\varphi \Rightarrow \top$	PC
5	$\Box \perp \parallel \Diamond_{\geq 0}\varphi \Rightarrow (\top \wedge \neg \Diamond_{\geq 1}\varphi) \parallel \top$	(ILR) , 1, 4
6	$\Diamond_{\geq 0}\varphi \Rightarrow (\top \wedge \neg \Diamond_{\geq 1}\varphi) \parallel \top$	$(\Rightarrow \text{TR})$, 2, 3, 5

base case: $k = 1$. In this case, $(I_{7.15.2}^*)$ is axiom (ATOM) .

induction step: $k \geq 2$.

1	$\Diamond_{\geq k-1}\varphi \Rightarrow \Diamond_{=k-1}\varphi \parallel \top$	Induction Hypothesis
2	$\Diamond_{\geq k}\varphi \Rightarrow \Diamond\varphi \parallel \Diamond_{\geq k-1}\varphi$	(ASSOC) , def. of $\Diamond_{\geq k}$, as $k \geq 2$
3	$\Diamond\varphi \Rightarrow \Diamond_{=1}\varphi \parallel \top$	(ATOM)
4	$\Diamond\varphi \parallel \Diamond_{\geq k-1}\varphi \Rightarrow (\Diamond_{=1}\varphi \parallel \top) \parallel \Diamond_{\geq k-1}\varphi$	(C) , 3
5	$(\Diamond_{=1}\varphi \parallel \top) \parallel \Diamond_{\geq k-1}\varphi \Rightarrow \Diamond_{=1}\varphi \parallel (\top \parallel \Diamond_{\geq k-1}\varphi)$	(ASSOC)
6	$\top \parallel \Diamond_{\geq k-1}\varphi \Rightarrow \Diamond_{\geq k-1}\varphi$	$(I_{7.15.1}^*)$
7	$\top \parallel \Diamond_{\geq k-1}\varphi \Rightarrow \Diamond_{=k-1}\varphi \parallel \top$	$(\Rightarrow \text{TR})$, 1, 6
8	$\Diamond_{=1}\varphi \parallel (\top \parallel \Diamond_{\geq k-1}\varphi) \Rightarrow \Diamond_{=1}\varphi \parallel (\Diamond_{=k-1}\varphi \parallel \top)$	(COM) , (C) , 7
9	$\Diamond_{=1}\varphi \parallel (\Diamond_{=k-1}\varphi \parallel \top) \Rightarrow (\Diamond_{=1}\varphi \parallel \Diamond_{=k-1}\varphi) \parallel \top$	(ASSOC)
10	$\Diamond_{=1}\varphi \Rightarrow \Diamond_{\geq 1}\varphi$	PC
11	$\Diamond_{=k-1}\varphi \Rightarrow \Diamond_{\geq k-1}\varphi$	PC
12	$\Diamond_{=1}\varphi \parallel \Diamond_{=k-1}\varphi \Rightarrow \Diamond_{\geq 1}\varphi \parallel \Diamond_{\geq k-1}\varphi$	(ILR) , 10, 11
13	$\Diamond_{\geq 1}\varphi \parallel \Diamond_{\geq k-1}\varphi \Rightarrow \Diamond_{\geq k}\varphi$	(ASSOC) , def. of $\Diamond_{\geq k}$, as $k \geq 2$
14	$\Diamond_{=1}\varphi \Rightarrow \neg \Diamond_{\geq 2}\varphi$	PC
15	$\Diamond_{=k-1}\varphi \Rightarrow \neg \Diamond_{\geq k}\varphi$	PC
16	$\Diamond_{=1}\varphi \parallel \Diamond_{=k-1}\varphi \Rightarrow \neg \Diamond_{\geq 2}\varphi \parallel \neg \Diamond_{\geq k}\varphi$	(ILR) , 14, 15
17	$\neg \Diamond_{\geq 2}\varphi \parallel \neg \Diamond_{\geq k}\varphi \Rightarrow \neg \Diamond_{\geq k+1}$	$(\neg \text{GRAD})$
18	$\Diamond_{=1}\varphi \parallel \Diamond_{=k-1}\varphi \Rightarrow \Diamond_{\geq k}\varphi$	$(\Rightarrow \text{TR})$, 12, 13
19	$\Diamond_{=1}\varphi \parallel \Diamond_{=k-1}\varphi \Rightarrow \neg \Diamond_{\geq k+1}\varphi$	$(\Rightarrow \text{TR})$, 16, 17

20	$\Diamond_{=1}\varphi \mid \Diamond_{=k-1}\varphi \Rightarrow \Diamond_{=k}\varphi$	PC, 18, 19, def. of $\Diamond_{=k}\varphi$
21	$(\Diamond_{=1}\varphi \mid \Diamond_{=k-1}\varphi) \mid \top \Rightarrow \Diamond_{=k}\varphi \mid \top$	(C), 20
22	$\Diamond_{\geq k}\varphi \Rightarrow \Diamond_{=k}\varphi \mid \top$	(\Rightarrow TR), 2, 4, 5, 8, 9, 21 \square

Proof of (I_{7.15.3}^{}).* Recall that $\Box\varphi$ is defined as $\neg\Diamond\neg\varphi$.

1	$\psi \Rightarrow (\psi \wedge \Diamond\neg\varphi) \vee (\psi \wedge \Box\varphi)$	PC
2	$\chi \Rightarrow (\chi \wedge \Diamond\neg\varphi) \vee (\chi \wedge \Box\varphi)$	PC
3	$\psi \mid \chi \Rightarrow ((\psi \wedge \Diamond\neg\varphi) \vee (\psi \wedge \Box\varphi)) \mid ((\chi \wedge \Diamond\neg\varphi) \vee (\chi \wedge \Box\varphi))$	(I _{LR}), 1, 2
4	$((\psi \wedge \Diamond\neg\varphi) \vee (\psi \wedge \Box\varphi)) \mid ((\chi \wedge \Diamond\neg\varphi) \vee (\chi \wedge \Box\varphi)) \Rightarrow$ $((\psi \wedge \Diamond\neg\varphi) \mid (\chi \wedge \Diamond\neg\varphi)) \vee ((\psi \wedge \Box\varphi) \mid (\chi \wedge \Diamond\neg\varphi))$ $\vee ((\psi \wedge \Diamond\neg\varphi) \mid (\chi \wedge \Box\varphi)) \vee ((\psi \wedge \Box\varphi) \mid (\chi \wedge \Box\varphi))$	(DIST ^C , (COM ^C), PC)
5	$\psi \wedge \Diamond\neg\varphi \Rightarrow \Diamond\neg\varphi$	PC
6	$\chi \wedge \Diamond\neg\varphi \Rightarrow \top$	PC
7	$(\psi \wedge \Diamond\neg\varphi) \mid (\chi \wedge \Diamond\neg\varphi) \Rightarrow \Diamond\neg\varphi \mid \top$	(I _{LR}), 5, 6
8	$\Diamond\neg\varphi \mid \top \Rightarrow \Diamond\neg\varphi$	(MONO ^C)
9	$(\psi \wedge \Diamond\neg\varphi) \mid (\chi \wedge \Diamond\neg\varphi) \Rightarrow \Diamond\neg\varphi$	
10	$\psi \wedge \Box\varphi \Rightarrow \top$	PC
11	$\chi \wedge \Diamond\neg\varphi \Rightarrow \Diamond\neg\varphi$	PC
12	$(\psi \wedge \Box\varphi) \mid (\chi \wedge \Diamond\neg\varphi) \Rightarrow \Diamond\neg\varphi \mid \top$	(I _{LR}), 10, 11, (COM ^C)
13	$(\psi \wedge \Box\varphi) \mid (\chi \wedge \Diamond\neg\varphi) \Rightarrow \Diamond\neg\varphi$	(\Rightarrow TR), 8, 12
14	$\chi \wedge \Box\varphi \Rightarrow \top$	PC
15	$(\psi \wedge \Diamond\neg\varphi) \mid (\chi \wedge \Box\varphi) \Rightarrow \Diamond\neg\varphi \mid \top$	(I _{LR}), 5, 14
16	$(\psi \wedge \Diamond\neg\varphi) \mid (\chi \wedge \Box\varphi) \Rightarrow \Diamond\neg\varphi$	(\Rightarrow TR), 8, 15
17	$((\psi \wedge \Diamond\neg\varphi) \vee (\psi \wedge \Box\varphi)) \mid ((\chi \wedge \Diamond\neg\varphi) \vee (\chi \wedge \Box\varphi)) \Rightarrow$ $\Diamond\neg\varphi \vee ((\psi \wedge \Box\varphi) \mid (\chi \wedge \Box\varphi))$	PC, 4, 9, 13, 16
18	$\psi \mid \chi \Rightarrow \Diamond\neg\varphi \vee ((\psi \wedge \Box\varphi) \mid (\chi \wedge \Box\varphi))$	(\Rightarrow TR), 3, 17
19	$\Box\varphi \wedge (\psi \mid \chi) \Rightarrow (\psi \wedge \Box\varphi) \mid (\chi \wedge \Box\varphi)$	PC, 18, def. of $\Box\varphi$ \square

Proof of (I_{7.15.4}^{}).* The proof is by induction on $n \geq 1$.

base case: $n = 1$. In this case, (I_{7.15.4}^{*}) corresponds to the axiom (SPLIT ^C).

induction step: $n \geq 2$.

1	$\Box\varphi_1 \mid \cdots \mid \Box\varphi_{n-1} \mid (\Box\neg\varphi_1 \wedge \cdots \wedge \Box\neg\varphi_{n-1})$	Induction Hypothesis
---	--	----------------------

2	$\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1} \Rightarrow (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1}) \wedge (\square\varphi_n \parallel \neg\square\varphi_n)$	PC, $(\text{SPLIT}^{\text{C}})$
3	$(\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1}) \wedge (\square\varphi_n \parallel \neg\square\varphi_n) \Rightarrow$ $(\square\varphi_n \wedge (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1})) \parallel (\neg\square\varphi_n \wedge (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1}))$	$(I_{7.15.3}^*)$
4	$\square\varphi_n \wedge (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1}) \Rightarrow \square\varphi_n$	PC
5	$\neg\square\varphi_n \wedge (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1}) \Rightarrow \square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_n$	PC
6	$(\square\varphi_n \wedge (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1})) \parallel (\neg\square\varphi_n \wedge (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1})) \Rightarrow$ $\square\varphi_n \parallel (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_n)$	$(\text{I}_{LR}^{\text{C}}), 4, 5$
7	$\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_{n-1} \Rightarrow \square\varphi_n \parallel (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_n)$	$(\Rightarrow\text{TR}), 2, 6$
8	$\square\varphi_1 \parallel \cdots \parallel \square\varphi_{n-1} \parallel (\square\varphi_n \parallel (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_n)) \Rightarrow$ $\square\varphi_1 \parallel \cdots \parallel \square\varphi_{n-1} \parallel (\square\varphi_n \parallel (\square\neg\varphi_1 \wedge \cdots \wedge \square\neg\varphi_n))$	$(\text{COM}^{\text{C}}), (\text{C}), 7 \quad \square$

F

Appendix of Chapter 8

Contents

Proof of Lemma 8.1.	569
Proof of Lemma 8.19.	570
Proof of Lemma 8.20.	571
Proof of Lemma 8.22.	572
Proof of Lemma 8.23.	573

Proof of Lemma 8.1.

Lemma 8.1. Let (\mathcal{K}, w) be a pointed forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, and let φ be a formula in $\text{ML}(\mathbf{I})$, written with atomic propositions from P . Let $\mathcal{K}' = (\mathcal{W}, R, \mathcal{V}[q_1 \leftarrow R(w)])$. $(\mathcal{K}, w) \models \varphi$ in $\text{ML}(\mathbf{I})$ if and only if $(\mathcal{K}', w) \models \tau_1(\varphi)$ in QK .

Proof. We prove a generalisation of this lemma. Consider a formula φ written with atomic proposition from P , where P is disjoint from $Q = \{q_1, q_2, q_3\}$. Given two Kripke-style finite forests $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$ and a world $w \in \mathcal{W}$, we write $\mathcal{K} \triangleright_i^w \mathcal{K}'$ whenever:

1. $\mathcal{W} \subseteq \mathcal{W}'$,
2. $\mathcal{V}'(q_i) \cap R'(w) = R(w)$,
3. for every $w' \in R(w)$, $\{(w_1, w_2) \in R \mid w_1 \in R^*(w')\} = \{(w_1, w_2) \in R' \mid w_1 \in R'^*(w')\}$,
4. for every $p \in P$ and $w' \in \mathcal{W}$, $w' \in \mathcal{V}(p)$ if and only if $w' \in \mathcal{V}'(p)$.

Essentially, $\mathcal{K} \triangleright_i^w \mathcal{K}'$ holds whenever the universe of \mathcal{K}' extends the one of \mathcal{K} (property (1)), and the children of w in \mathcal{K} are the subset of the children of w in \mathcal{K}' that satisfy the atomic proposition q_i (property (2)). The trees rooted in these children are the same in \mathcal{K} and \mathcal{K}' (property (3)). Lastly, the valuation functions \mathcal{V} and \mathcal{V}' agree on the satisfaction of the atomic proposition in P , for every world in \mathcal{W} (property (4)).

One can easily check that $(\mathcal{W}, R, \mathcal{V}) \triangleright_1^w (\mathcal{W}, R, \mathcal{V}[q_1 \leftarrow R(w)])$. Thus, Lemma 8.1 holds as we show that for all $i \in \{1, 2, 3\}$, $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$ and $w \in \mathcal{W}$, such that $\mathcal{K} \triangleright_i^w \mathcal{K}'$:

$$(\mathcal{K}, w) \models \varphi \text{ in } \text{ML}(\mathbf{I}) \text{ if and only if } (\mathcal{K}', w) \models \tau_i(\varphi).$$

The proof carries out by structural induction on φ (trivial cases for \top and Boolean connectives are omitted).

base case: $\varphi = p$. By definition of φ , $p \in P$. From the properties (1) and (4) of \triangleright_i^w , we have $w \in \mathcal{V}(p)$ if and only if $w \in \mathcal{V}'(p)$.

induction step: $\varphi = \Diamond\psi$. (\Rightarrow): Suppose $(\mathcal{K}, w) \models \Diamond\psi$, and thus there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models \psi$. From $w' \in R(w)$ and the property (2) of \triangleright_i^w , we have $w' \in R'(w)$ and $w' \in \mathcal{V}'(q_i)$. Consider the Kripke-style finite forest $\mathcal{K}'' = (\mathcal{W}', R', \mathcal{V}'[q_1 \leftarrow R'(w')])$. We show that $\mathcal{K} \triangleright_1^w \mathcal{K}''$. The satisfaction of the properties (1) and (4) follows directly from $\mathcal{K} \triangleright_i^w \mathcal{K}'$. Indeed, \mathcal{K}' and \mathcal{K}'' share the same universe, and $\mathcal{V}'[q_1 \leftarrow R'(w')](p) = \mathcal{V}'(p)$ holds for all $p \in P$. From the definition of $\mathcal{V}'[q_1 \leftarrow R'(w')]$, the property (2) reduces to showing that $R'(w') = R(w')$, which follows directly from the fact that $w' \in R(w)$ and $\mathcal{K} \triangleright_i^w \mathcal{K}'$ (property (3)). Similarly, the property (3) follows directly from the fact that \mathcal{K} and \mathcal{K}' share the same accessibility relation R' , $w' \in R(w)$ and $\mathcal{K} \triangleright_i^w \mathcal{K}'$ (again, property (3)). By induction hypothesis, $(\mathcal{K}'', w') \models \tau_1(\psi)$. Moreover, from $\mathcal{V}'[q_1 \leftarrow R'(w')]$, $(\mathcal{K}', w') \models \Box q_1$. By definition of \mathcal{K}'' together with the semantics of the propositional quantification $\exists q_1$, we conclude that $(\mathcal{K}', w') \models \exists q_1 (\Box q_1 \wedge \tau_1(\psi))$. Lastly, from $w' \in R'(w)$ and $w' \in \mathcal{V}'(q_i)$, we derive $(\mathcal{K}', w) \models \tau_i(\Diamond\psi)$.

(\Leftarrow): Similar to the other direction. Suppose $(\mathcal{K}', w) \models \tau_i(\Diamond\psi)$. By definition, there are $w' \in R'(w) \cap \mathcal{V}(q_i)$ and $\mathcal{K}'' = (\mathcal{W}', R', \mathcal{V}[q_1 \leftarrow \mathcal{W}_1])$ such that $R'(w) \subseteq \mathcal{W}_1 \subseteq \mathcal{W}'$ and $(\mathcal{K}'', w') \models \tau_1(\psi)$. Exactly as in the left to right direction of the proof, one can show that $\mathcal{K} \triangleright_1^w \mathcal{K}''$. By induction hypothesis, $(\mathcal{K}, w') \models \psi$. From $w' \in R'(w) \cap \mathcal{V}(q_i)$ and $\mathcal{K} \triangleright_i^w \mathcal{K}'$ (property (2)), we have $w' \in R(w)$. Thus, $(\mathcal{K}, w) \models \Diamond\psi$.

induction step: $\varphi = \varphi_1 \parallel \varphi_2$. Let $j, k \in [1, 3]$ such that $j < k$ and $j \neq i \neq k$.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \varphi_1 \mid \varphi_2$. By definition, there are Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ s.t. $\mathcal{K}_1 +_w \mathcal{K}_2 = \mathcal{K}$, $(\mathcal{K}_1, w) \models \varphi_1$ and $(\mathcal{K}_2, w) \models \varphi_2$. Let $\mathcal{K}'' = (\mathcal{W}', R', \mathcal{V}'')$ where $\mathcal{V}'' \stackrel{\text{def}}{=} \mathcal{V}'[q_j \leftarrow R_1(w), q_k \leftarrow R_2(w)]$. Since $\mathcal{K} \triangleright_i^w \mathcal{K}'$ (property (2)), and the sets $R_1(w)$ and $R_2(w)$ partition $R(w)$, we conclude that \mathcal{K}'' is well-defined, and that $R'(w) \cap \mathcal{V}''(q_i)$ is partitioned in $R'(w) \cap \mathcal{V}''(q_j)$ and $R'(w) \cap \mathcal{V}''(q_k)$. Thus $(\mathcal{K}'', w) \models [q_i = q_j \mid q_k]$. We prove that $\mathcal{K}_1 \triangleright_j^w \mathcal{K}''$. The properties (1) and (4) are trivially satisfied (by $\mathcal{K} \triangleright_i^w \mathcal{K}'$), whereas the property (2) follows directly from the definition of \mathcal{V}'' . To show (3), i.e. for every $w' \in R_1(w)$, $\{(w_1, w_2) \in R_1 \mid w_1 \in R_1^*(w')\} = \{(w_1, w_2) \in R' \mid w_1 \in R'^*(w')\}$, it is sufficient to recall that, by definition of $+_w$ and $\mathcal{K}_1 +_w \mathcal{K}_2 = \mathcal{K}$, for every $w \in R_1(w)$, we have $\{(w_1, w_2) \in R_1 \mid w_1 \in R_1^*(w')\} = \{(w_1, w_2) \in R \mid w_1 \in R^*(w')\}$. Then, the property holds directly from $\mathcal{K} \triangleright_i^w \mathcal{K}'$ (property (3)). Analogously, one can prove that $\mathcal{K}_2 \triangleright_k^w \mathcal{K}''$. By induction hypothesis, $(\mathcal{K}'', w) \models \tau_j(\varphi_1) \wedge \tau_k(\varphi_2)$. Together with $(\mathcal{K}'', w) \models [q_i = q_j \mid q_k]$ and by definition of \mathcal{V}'' , we conclude that $(\mathcal{K}'', w) \models \tau_i(\varphi_1 \mid \varphi_2)$.

(\Leftarrow): Suppose $(\mathcal{K}', w) \models \tau_i(\varphi_1 \mid \varphi_2)$. By definition, there is $\mathcal{K}'' = (\mathcal{W}', R', \mathcal{V}'')$, where $\mathcal{V}'' \stackrel{\text{def}}{=} \mathcal{V}'[q_j \leftarrow \mathcal{W}_1, q_k \leftarrow \mathcal{W}_2]$ such that $\mathcal{W}_1, \mathcal{W}_2 \subseteq \mathcal{W}'$ and $(\mathcal{K}'', w) \models [q_i = q_j \mid q_k] \wedge \tau_j(\varphi_1) \wedge \tau_k(\varphi_2)$. From the semantics of $[q_i = q_j \mid q_k]$, \mathcal{W}_1 and \mathcal{W}_2 are disjoint, and their union contains $R(w) \cap \mathcal{V}'(q_i)$. From $\mathcal{K} \triangleright_i^w \mathcal{K}'$ (property (2)), $R(w) \subseteq \mathcal{W}_1 \cup \mathcal{W}_2$. Therefore, let us consider two Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K}_1 +_w \mathcal{K}_2 = \mathcal{K}$, $R_1(w) = R(w) \cap \mathcal{W}_1$ and $R_2(w) = R(w) \cap \mathcal{W}_2$. From the disjointness of \mathcal{W}_1 and \mathcal{W}_2 , together with $R(w) \subseteq \mathcal{W}_1 \cup \mathcal{W}_2$, we conclude that \mathcal{K}_1 and \mathcal{K}_2 are well-defined. Exactly as in the left to right direction of the proof, one can show that $\mathcal{K}_1 \triangleright_j^w \mathcal{K}''$ and $\mathcal{K}_2 \triangleright_k^w \mathcal{K}''$. By induction hypothesis, $(\mathcal{K}_1, w) \models \varphi_1$ and $(\mathcal{K}_2, w) \models \varphi_2$. We derive $(\mathcal{K}, w) \models \varphi_1 \mid \varphi_2$. \square

Proof of Lemma 8.19.

Lemma 8.19. Let $n \in \mathbb{N}$, $P \subseteq_{\text{fin}} \text{AP}$. Let T be an information tree having ambient names from P and let (\mathcal{K}, w) be a pointed forest such that (\mathcal{K}, w) (n, P)-encodes T . For every formula φ in $\text{SAL}(\mid)$ such that $|\varphi| \leq n$ we have, $T \models \varphi$ if and only if $(\mathcal{K}, w) \models \tau(\varphi)$.

Proof. The proof follows from an easy structural induction on φ (trivial cases for \top and Boolean connectives are omitted). Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

base case: $\varphi = 0$. We have,

$$\begin{aligned} T &\models 0 \\ \Leftrightarrow T &\equiv 0 \text{ (by definition of } \models \text{, in } \text{SAL}(\mid)\text{),} \\ \Leftrightarrow R(w) &= \emptyset \text{ (by Definition 8.18, in both cases (1) and (2)),} \\ \Leftrightarrow (\mathcal{K}, w) &\models \Box \perp \text{ (by definition of } \models \text{, in } \text{ML}(\mid)\text{).} \end{aligned}$$

induction step: $\varphi = \mathbf{n}[\psi]$. (\Rightarrow): Suppose $T \models \mathbf{n}[\psi]$, and thus there is T' such that $T \equiv \mathbf{n}[T']$ and $T' \models \psi$. Since $|\varphi| \geq 1$, Since (\mathcal{K}, w) is a (n, P) -encoding of T (Definition 8.18) there is a world w' such that $R(w) = \{w'\}$ and (\mathcal{K}, w') is a $(n-1, p)$ -encoding of T' , and $(\mathcal{K}, w') \models \mathbf{n}$. We have $|\psi| \leq |\varphi| - 1 \leq n - 1$, and thus by induction hypothesis $(\mathcal{K}, w') \models \tau(\psi)$. From $w' \in R(w)$ we conclude that $(\mathcal{K}, w) \models \Diamond(\mathbf{n} \wedge \psi)$. Together with $\text{card}(R(w)) = 1$, which implies $(\mathcal{K}, w) \models \Diamond_{=1} \top$, this allows us to derive that $(\mathcal{K}, w) \models \tau(\varphi)$.

(\Leftarrow): Similar to the order direction. Suppose $(\mathcal{K}, w) \models \tau(\varphi)$, and therefore there is a world w' such that $\{w'\} = R(w)$ and $(\mathcal{K}, w') \models \mathbf{n} \wedge \psi$. Since $|\varphi| \geq 1$ and (\mathcal{K}, w) is a (n, P) -encoding of T , it must be that $T \equiv \mathbf{n}[T']$ for some information tree T' , and that (\mathcal{K}, w')

is a $(n - 1, P)$ -encoding of T' . We have $|\psi| \leq |\varphi| - 1 \leq n - 1$, and thus by induction hypothesis $T' \models \psi$. From $T \equiv \mathbf{n}[T']$, we conclude that $T \models \mathbf{n}[\psi]$.

induction step: $\varphi = \psi \mid \chi$. (\Rightarrow): Suppose $T \models \varphi \mid \psi$ and therefore there are information trees T_1 and T_2 such that $T_1 \mid T_2 \equiv T$, $T_1 \models \varphi$ and $T_2 \models \psi$. Let $\mathbf{n}_1, \dots, \mathbf{n}_k$ and T'_1, \dots, T'_k ($k \in \mathbb{N}$) such that $T \equiv \mathbf{n}_1[T'_1] \mid \dots \mid \mathbf{n}_k[T'_k]$. From $T_1 \mid T_2 \equiv T$ together with the properties of the structural equivalence \equiv (see Figure 8.5), we conclude that $[1, k]$ can be partitioned into two sets of indices $\{i_1, \dots, i_p\}$ and $\{j_1, \dots, j_q\}$ such that $T_1 \equiv \mathbf{n}_{i_1}[T'_{i_1}] \mid \dots \mid \mathbf{n}_{i_p}[T'_{i_p}]$ and $T_2 \equiv \mathbf{n}_{j_1}[T'_{j_1}] \mid \dots \mid \mathbf{n}_{j_q}[T'_{j_q}]$. Since $(\mathcal{K}, \mathbf{w})$ is a (n, P) -encoding of T , we conclude that there are worlds $\mathbf{w}_1, \dots, \mathbf{w}_k$ such that $\{\mathbf{w}_1, \dots, \mathbf{w}_k\} = R(\mathbf{w})$ and for every $i \in [1, k]$, $(\mathcal{K}, \mathbf{w}_i)$ is a $(n - 1, P)$ -encoding of T'_i and, among the atomic propositions in P , \mathbf{w}_i only satisfies \mathbf{n}_i . Let $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ be two Kripke-style finite forests such that $\mathcal{K} = \mathcal{K}_1 +_{\mathbf{w}} \mathcal{K}_2$, $R_1(\mathbf{w}) = \{\mathbf{w}_{i_1}, \dots, \mathbf{w}_{i_p}\}$ and $R_2(\mathbf{w}) = \{\mathbf{w}_{j_1}, \dots, \mathbf{w}_{j_q}\}$. Let $|\varphi| = k_1$ and $|\psi| = k_2$. We have $n \geq |\varphi| > k_1 + k_2$, and so $(\mathcal{K}_1, \mathbf{w})$ is a (k_1, P) -encoding of T_1 , whereas $(\mathcal{K}_2, \mathbf{w})$ is a (k_2, P) -encoding of T_2 . By induction hypothesis, $(\mathcal{K}_1, \mathbf{w}) \models \tau(\psi)$ and $(\mathcal{K}_2, \mathbf{w}) \models \tau(\chi)$. From $\mathcal{K} = \mathcal{K}_1 +_{\mathbf{w}} \mathcal{K}_2$, $(\mathcal{K}, \mathbf{w}) \models \tau(\psi) \mid \tau(\chi)$.

(\Leftarrow): Similar to the other direction. Suppose $(\mathcal{K}, \mathbf{w}) \models \tau(\psi \mid \chi)$, and thus there are Kripke-style finite forests \mathcal{K}_1 and \mathcal{K}_2 such that $\mathcal{K}_1 +_{\mathbf{w}} \mathcal{K}_2 = \mathcal{K}$, $(\mathcal{K}_1, \mathbf{w}) \models \tau(\psi)$ and $(\mathcal{K}_2, \mathbf{w}) \models \tau(\chi)$. Since $|\varphi| \geq 1$ and $(\mathcal{K}, \mathbf{w})$ is a (n, P) -encoding of T , there must be $\mathbf{n}_1, \dots, \mathbf{n}_k$ and information trees T_1, \dots, T_k ($k \in \mathbb{N}$) such that $T \equiv \mathbf{n}_1[T_1] \mid \dots \mid \mathbf{n}_k[T_k]$ and there are worlds $\mathbf{w}_1, \dots, \mathbf{w}_k$ such that $\{\mathbf{w}_1, \dots, \mathbf{w}_k\} = R(\mathbf{w})$ and for every $i \in [1, k]$, $(\mathcal{K}, \mathbf{w}_i)$ is a $(n - 1, P)$ -encoding of T'_i and, among the atomic propositions in P , \mathbf{w}_i only satisfies \mathbf{n}_i . Let $\{i_1, \dots, i_p\}$ and $\{j_1, \dots, j_q\}$ be a partition of the indices in $[1, k]$ such that $R_1(\mathbf{w}) = \{\mathbf{w}_{i_1}, \dots, \mathbf{w}_{i_p}\}$ and $R_2(\mathbf{w}) = \{\mathbf{w}_{j_1}, \dots, \mathbf{w}_{j_q}\}$. Consider the two information trees $T' = \mathbf{n}_{i_1}[T_{i_1}] \mid \dots \mid \mathbf{n}_{i_p}[T_{i_p}]$ and $T'' = \mathbf{n}_{j_1}[T_{j_1}] \mid \dots \mid \mathbf{n}_{j_q}[T_{j_q}]$. From the properties of \equiv , we have $T \equiv T' \mid T''$. Let $|\varphi| = k_1$ and $|\psi| = k_2$. We have $n \geq |\varphi| > k_1 + k_2$, and so $(\mathcal{K}_1, \mathbf{w})$ is a (k_1, P) -encoding of T' , whereas $(\mathcal{K}_2, \mathbf{w})$ is a (k_2, P) -encoding of T'' . By induction hypothesis, $T' \models \psi$ and $T'' \models \chi$, which implies $T \models \psi \mid \chi$ directly from $T \equiv T' \mid T''$. \square

Proof of Lemma 8.20.

Lemma 8.20. Let φ be in $\text{SAL}(\mid)$ built over $P \subseteq_{\text{fin}} \text{AP}$ and $p \notin P$. φ is satisfiable if and only if $\tau(\varphi) \wedge \bigwedge_{i \in [1, |\varphi|]} \Box^i \bigvee_{\mathbf{n} \in P \cup \{p\}} (\mathbf{n} \wedge \bigwedge_{\mathbf{m} \in (P \cup \{p\}) \setminus \{\mathbf{n}\}} \neg \mathbf{m})$ is satisfiable.

Proof. (\Rightarrow): Suppose φ satisfiable, and let T be an information tree satisfying φ . In general, it could be that T contains ambient names that do not appear in φ . However, without loss of generality, we can assume that there is only one name in T that does not appear in φ , and that name is p (as in the statement of this theorem). This assumption relies on the following property of static ambient logic (see [34], Lemma 8):

Let p, q be two ambient names not appearing in φ . Then $T \models \varphi$ iff $T[p \leftarrow q] \models \varphi$, where $T[p \leftarrow q]$ is the tree obtained from T by replacing every occurrence of p with q .

Let $(\mathcal{K}, \mathbf{w})$ be a $(|\varphi|, P \cup \{p\})$ -encoding of T . By Lemma 8.19, $(\mathcal{K}, \mathbf{w}) \models \tau(\varphi)$. From the properties of $(|\varphi|, P \cup \{p\})$ -encodings, for all $i \in [1, |\varphi|]$, every world $\mathbf{w}' \in R^i(\mathbf{w})$ satisfies exactly one atomic propositions in $P \cup \{p\}$. Therefore, $(\mathcal{K}, \mathbf{w}) \models \bigwedge_{i \in [1, |\varphi|]} \Box^i \bigvee_{\mathbf{n} \in P \cup \{p\}} (\mathbf{n} \wedge \bigwedge_{\mathbf{m} \in (P \cup \{p\}) \setminus \{\mathbf{n}\}} \neg \mathbf{m})$.

(\Leftarrow): Suppose $\psi \stackrel{\text{def}}{=} \tau(\varphi) \wedge \bigwedge_{i \in [1, |\varphi|]} \Box^i \bigvee_{\mathbf{n} \in P \cup \{p\}} (\mathbf{n} \wedge \bigwedge_{\mathbf{m} \in (P \cup \{p\}) \setminus \{\mathbf{n}\}} \neg \mathbf{m})$ satisfiable, and let $(\mathcal{K}, \mathbf{w})$, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, be a pointed forest satisfying ψ . From the satisfaction of ψ we deduce that, for every $i \in [1, |\varphi|]$, every world $\mathbf{w}' \in R^i(\mathbf{w})$ satisfies exactly one atomic propositions

in $P \cup \{p\}$. One can easily show that then there is an information tree T written with ambient names from $P \cup \{p\}$ such that (\mathcal{K}, w) is a $(|\varphi|, P \cup \{p\})$ -encoding of T . From $(\mathcal{K}, w) \models \tau(\varphi)$, together with Lemma 8.19, we conclude that $T \models \varphi$. \square

Proof of Lemma 8.22.

Lemma 8.22. Let $P \subseteq_{\text{fin}} \text{AP}$ and $n \geq 1$. Let T be a (n, P) -encoding of a pointed forest (\mathcal{K}, w) . For every formula φ in $\text{ML}(\mathbf{I})$, built over P and with $|\varphi| \leq n$, we have $(\mathcal{K}, w) \models \varphi$ iff $T \models \varphi$.

Proof. Below, let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. The proof follows with a straightforward induction on φ (trivial cases for \top and Boolean connectives are omitted).

base case: $\varphi = p$. We have,

$$\begin{aligned} & (\mathcal{K}, w) \models p \\ \Leftrightarrow & w \models \mathcal{V}(p) \text{ (by definition of } \models, \text{ in } \text{ML}(\mathbf{I})), \\ \Leftrightarrow & \text{there are information trees } T', T'' \text{ such that } T \equiv \text{ap}[p[0] | T' | T'' \text{ (by Definition 8.21),} \\ \Leftrightarrow & T \models \langle \text{ap} \rangle \langle p \rangle \top \text{ (by definition of } \models, \text{ in } \text{SAL}(\mathbf{I})). \end{aligned}$$

induction step: $\varphi = \Diamond \psi$. (\Rightarrow): Suppose $(\mathcal{K}, w) \models \Diamond \psi$, and thus there is $w' \in R(w)$ such that $(\mathcal{K}, w') \models \psi$. Notice that $|\Diamond \psi| = |\psi| + 1 > 1$, and thus, by Definition 8.21, there are information trees T', T'' such that $T \equiv \text{rel}[T' | T'']$, and T' is a $(n-1, P)$ -encoding of (\mathcal{K}, w') . By induction hypothesis, $T' \models \tau(\psi)$. By $T \equiv \text{rel}[T' | T'']$, this implies $T \models \langle \text{rel} \rangle \tau(\psi)$.

(\Leftarrow): Similar to the other direction. Suppose $T \models \langle \text{rel} \rangle \tau(\psi)$. Thus, there are information trees T', T'' such that $T \equiv \text{rel}[T' | T'']$. Notice that $|\Diamond \psi| = |\psi| + 1 > 1$, and thus, by Definition 8.21, there is $w' \in R(w)$ such that T' is a $(n-1, P)$ -encoding of (\mathcal{K}, w') . By induction hypothesis, $(\mathcal{K}, w') \models \psi$. From $w' \in R(w)$, $(\mathcal{K}, w) \models \Diamond \psi$.

induction step: $\varphi = \psi | \chi$. Below, let $R(w) = \{w_1, \dots, w_k\}$. By Definition 8.21, there is $m \geq n$ and there are information trees T_1, \dots, T_k such that $T \equiv [\text{ap}[Q]]^m | \text{rel}[T_1] | \dots | \text{rel}[T_k]$.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \psi | \chi$, and thus there are Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$, $(\mathcal{K}_1, w) \models \psi$ and $(\mathcal{K}_2, w) \models \chi$. Let $\{i_1, \dots, i_p\}$ and $\{j_1, \dots, j_q\}$ be a partition of $[1, k]$ such that $R_1(w) = \{w_{i_1}, \dots, w_{i_p}\}$ and $R_2(w) = \{w_{j_1}, \dots, w_{j_q}\}$. Let m_1 and m_2 be two natural numbers such that $m_1 + m_2 = m$, $m_1 \geq |\psi|$ and $m_2 \geq |\chi|$. Since $m \geq n \geq |\varphi| > |\psi| + |\chi|$, m_1 and m_2 are guaranteed to exist. We consider the two information trees $T_1 = [\text{ap}[Q]]^{m_1} | \text{rel}[T_{i_1}] | \dots | \text{rel}[T_{i_p}]$ and $T_2 = [\text{ap}[Q]]^{m_2} | \text{rel}[T_{j_1}] | \dots | \text{rel}[T_{j_q}]$. From the properties of \equiv , we have $T \equiv T_1 | T_2$. By Definition 8.21, T_1 is a $(|\psi|, P)$ -encoding of (\mathcal{K}_1, w) , whereas T_2 is a $(|\chi|, P)$ -encoding of (\mathcal{K}_2, w) . By induction hypothesis, $T_1 \models \tau(\psi)$ and $T_2 \models \tau(\chi)$. Moreover, by definition of T_1 and T_2 , $T_1 \models \langle \text{ap} \rangle_{\geq |\psi|} \top$ and $T_2 \models \langle \text{ap} \rangle_{\geq |\chi|} \top$. From $T \equiv T_1 | T_2$, we derive $T \models \tau(\psi | \chi)$.

(\Leftarrow): Suppose $T \models (\tau(\psi) \wedge \langle \text{ap} \rangle_{\geq |\psi|} \top) | (\tau(\chi) \wedge \langle \text{ap} \rangle_{\geq |\chi|} \top)$. There are two information trees T_1 and T_2 such that $T \equiv T_1 | T_2$, $T_1 \models \tau(\psi) \wedge \langle \text{ap} \rangle_{\geq |\psi|} \top$ and $T_2 \models \tau(\chi) \wedge \langle \text{ap} \rangle_{\geq |\chi|} \top$. From $T \equiv [\text{ap}[Q]]^m | \text{rel}[T_1] | \dots | \text{rel}[T_k]$, there are two sets of indices $\{i_1, \dots, i_p\}$ and $\{j_1, \dots, j_q\}$ that partition $[1, k]$, such that $T_1 \equiv [\text{ap}[Q]]^{m_1} | \text{rel}[T_{i_1}] | \dots | \text{rel}[T_{i_p}]$ and $T_2 \equiv [\text{ap}[Q]]^{m_2} | \text{rel}[T_{j_1}] | \dots | \text{rel}[T_{j_q}]$. Let us consider the Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$, $R_1(w) = \{w_{i_1}, \dots, w_{i_p}\}$ and $R_2(w) = \{w_{j_1}, \dots, w_{j_q}\}$. By Definition 8.21, T_1 is a $(|\psi|, P)$ -encoding of (\mathcal{K}_1, w) , whereas T_2 is a $(|\chi|, P)$ -encoding of (\mathcal{K}_2, w) . The induction hypothesis applies, allowing us to derive $(\mathcal{K}_1, w) \models \psi$ and $(\mathcal{K}_2, w) \models \chi$. From $\mathcal{K} = \mathcal{K}_1 +_w \mathcal{K}_2$, $(\mathcal{K}, w) \models \psi | \chi$. \square

Proof of Lemma 8.23.

Lemma 8.23. Let φ be a formula in $\text{ML}(\mathbf{|})$ built over $P = \{p_1, \dots, p_k\} \subseteq_{\text{fin}} \text{AP}$. φ is satisfiable if and only if the formula $\tau(\varphi) \wedge \Lambda_{i \in [0, |\varphi|-1]} [\text{rel}]^i (\alpha_{i,|\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P)$ is satisfiable in $\text{SAL}(\mathbf{|})$.

Recall that, given an ambient name n and an information tree T congruent to $n[T'] \mathbin{\parallel} T''$, we say that T' is a n -child of T . Given $i \in \mathbb{N}$, we say that T' is a (i, n) -descendant of T whether there is a sequence of information trees T_0, \dots, T_i such that $T_0 \equiv T$, $T_i \equiv T'$, and for every $j \in [0, i-1]$, T_{j+1} is a n -child. We recall the definition of $\alpha_{i,n}$, β , γ_P and δ_P .

Formula	Informal explanation
$\alpha_{i,n} \stackrel{\text{def}}{=} \langle \text{ap} \rangle_{\geq(n-i)}$	T' is congruent to $\text{ap}[T_1] \mathbin{\parallel} \dots \mathbin{\parallel} \text{ap}[T_m] \mathbin{\parallel} T_{m+1}$, for some $m \geq n-i$ and trees T_1, \dots, T_{m+1} ,
$\beta \stackrel{\text{def}}{=} \neg(\top \mathbin{\parallel} (\neg 0 \wedge \neg \langle \text{rel} \rangle \top \wedge \neg \langle \text{ap} \rangle \top))$	every child of T' is either a rel -child or a ap -child,
$\gamma_P \stackrel{\text{def}}{=} \Lambda_{p \in P} (\langle \text{ap} \rangle \langle p \rangle \top \Rightarrow [\text{ap}] \langle p \rangle \top)$	rel -children of T' agree on the types of p -children they have, where $p \in P$,
$\delta_P \stackrel{\text{def}}{=} [\text{ap}]((p_1[0] \vee 0) \mathbin{\parallel} \dots \mathbin{\parallel} (p_k[0] \vee 0))$	there is $\{p_{i_1}, \dots, p_{i_l}\} \subseteq P$ such that every rel -child of T' is congruent to $p_{i_1}[0] \mathbin{\parallel} \dots \mathbin{\parallel} p_{i_l}[0]$.

Proof. (\Rightarrow): Suppose that φ is satisfiable, and consider a pointed forest $(\mathcal{K}, \mathbf{w})$ satisfying it, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. Let T be a $(|\varphi|, P)$ -encoding of $(\mathcal{K}, \mathbf{w})$. First, by Lemma 8.22 we know that $T \models \tau(\varphi)$. To show that T satisfies $\Lambda_{i \in [0, |\varphi|-1]} [\text{rel}]^i (\alpha_{i,|\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P)$, let us consider a (i, rel) -descendant T' of T , where $i \in [0, |\varphi|-1]$. We show that

$$T' \models \alpha_{i,|\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P.$$

Since T be a $(|\varphi|, P)$ -encoding of $(\mathcal{K}, \mathbf{w})$, by Definition 8.21, there is a world $\mathbf{w}' \in R^i(\mathbf{w})$ such that T' is a $(|\varphi|-i, P)$ -encoding of $(\mathcal{K}, \mathbf{w})$. Let $R(\mathbf{w}') = \{\mathbf{w}_1, \dots, \mathbf{w}_m\}$, and let Q be the set of atomic propositions among P that are satisfied by \mathbf{w}' . Again by Definition 8.21, there are $\delta \geq |\varphi|-i$ and information trees T_1, \dots, T_m such that $T' \equiv [\text{ap}[Q]]^\delta \mathbin{\parallel} \text{rel}[T_1] \mathbin{\parallel} \dots \mathbin{\parallel} \text{rel}[T_m]$.

T' satisfies $\alpha_{i,|\varphi|}$. Directly from the fact that $\delta \geq |\varphi|-i$ and $T' \equiv [\text{ap}[Q]]^\delta \mathbin{\parallel} T''$, for some tree T'' , we conclude that $T' \models \langle \text{ap} \rangle_{\geq(|\varphi|-i)} \top$.

T' satisfies β . From $T' \equiv [\text{ap}[Q]]^\delta \mathbin{\parallel} \text{rel}[T_1] \mathbin{\parallel} \dots \mathbin{\parallel} \text{rel}[T_m]$, every child of T' is either a rel -child or a ap -child. Thus, $T \models \neg(\top \mathbin{\parallel} (\neg 0 \wedge \neg \langle \text{rel} \rangle \top \wedge \neg \langle \text{ap} \rangle \top))$.

T' satisfies γ_P and δ_P . As rel and ap are distinct ambient names, given $Q = \{q_1, \dots, q_{k'}\}$, we have $T' \models [\text{ap}](q_1[0] \mathbin{\parallel} \dots \mathbin{\parallel} q_{k'}[0])$. It is easy to see that this implies both

$$T \models \Lambda_{p \in P} (\langle \text{ap} \rangle \langle p \rangle \top \Rightarrow [\text{ap}] \langle p \rangle \top) \quad \text{and} \quad T \models [\text{ap}]((p_1[0] \vee 0) \mathbin{\parallel} \dots \mathbin{\parallel} (p_k[0] \vee 0)).$$

(\Leftarrow): For the other direction, suppose $T \models \tau(\varphi) \wedge \Lambda_{i \in [0, |\varphi|-1]} [\text{rel}]^i (\alpha_{i,|\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P)$. First of all, let us show that there is a pointed forest $(\mathcal{K}, \mathbf{w})$ such that T is a $(|\varphi|, P)$ -encoding of $(\mathcal{K}, \mathbf{w})$. We reason inductively, by showing that for all $i \in [0, |\varphi|-1]$, every (i, rel) -descendant T' of T is a $(|\varphi|-i, P)$ -encoding of some pointed forest. The induction hypothesis is that every rel -child of T' is a $(|\varphi|-i-1, P)$ -encoding of some pointed forest.

base case: $i = |\varphi|-1$. Let T' be a $(|\varphi|-1, \text{rel})$ -descendant of T . We show that it is a $(1, P)$ -encoding of some pointed forest. By $T \models \Lambda_{i \in [0, |\varphi|-1]} [\text{rel}]^i (\alpha_{i,|\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P)$ we have

$$T' \models \alpha_{|\varphi|-1,|\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P.$$

Following Definition 8.21 (point (1)), we show that $T' \equiv [\text{ap}[Q]]^m \mid \text{rel}[T_1] \mid \dots \mid \text{rel}[T_k]$, for some $m \geq 1$, trees T_1, \dots, T_k and $Q \subseteq P$. From $T' \models \beta$, every child of T' is either a **rel**-child or a **ap**-child. Therefore, there are trees T_1, \dots, T_k and T'_1, \dots, T'_j , where $j, k \in \mathbb{N}$, such that $T \equiv \text{ap}[T'_1] \mid \dots \mid \text{ap}[T'_j] \mid \text{rel}[T_1] \mid \dots \mid \text{rel}[T_k]$. From $T' \models \alpha_{|\varphi|-1} \mid \varphi$, T' has at least one **ap**-child, i.e. $j \geq 1$. It remains to show that $T'_1 \equiv \dots \equiv T'_j$. From $T' \models \delta_P$, every T'_l ($l \in [1, j]$) is congruent to an information tree $p_{l_1}[0] \mid \dots \mid p_{l_p}[0]$, where $\{p_{l_1}, \dots, p_{l_p}\} \subseteq P$. From $T' \models \gamma_P$, given two informations trees T'_l, T'_r ($l, r \in [1, j]$) and $p \in P$, T'_l has an p -child if and only if so does T'_r . We derive that $T'_l \equiv T'_r$, which allows us to conclude that T' is a $(1, P)$ -encoding of some pointed forest.

induction step: $i < |\varphi| - 1$. Let T' be a (i, rel) -descendant of T , which we show being a $(|\varphi| - i, P)$ -encoding of some pointed forest. By $T \models \Lambda_{i \in [0, |\varphi| - 1]} [\text{rel}]^i (\alpha_{i, |\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P)$,

$$T' \models \alpha_{i, |\varphi|} \wedge \beta \wedge \gamma_P \wedge \delta_P.$$

Exactly as in the base case of the proof, this allows us to derive that there are $m \geq |\varphi| - i$, T_1, \dots, T_k and $Q \subseteq P$ such that $T' \equiv [\text{ap}[Q]]^m \mid \text{rel}[T_1] \mid \dots \mid \text{rel}[T_k]$. Moreover, by induction hypothesis, every tree T_j ($j \in [1, k]$) are $(|\varphi| - i - 1, P)$ -encoding of a pointed forest $(\mathcal{K}_j, \mathbf{w}_j)$, where $\mathcal{K}_j = (\mathcal{W}_j, R_j, \mathcal{V}_j)$. Without loss of generality, let us assume that these pointed forests $(\mathcal{K}_1, \mathbf{w}_1), \dots, (\mathcal{K}_k, \mathbf{w}_k)$ do not share any world. We consider the pointed forest $(\mathcal{K}', \mathbf{w}')$, where $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$ and

- $\mathcal{W}' = \{\mathbf{w}'\} \cup \bigcup_{j \in [1, k]} \mathcal{W}_j$,
- $R' = \{(\mathbf{w}', \mathbf{w}_j) \mid j \in [1, k]\} \cup \bigcup_{j \in [1, k]} R_j$,
- for every $p \in \text{AP}$ we have:
 - $\mathbf{w}' \in \mathcal{V}'(p)$ if and only if $p \in Q$,
 - given $j \in [1, k]$ and $\mathbf{w}'' \in \mathcal{W}_j$, $\mathbf{w}'' \in \mathcal{V}'(p)$ if and only if $\mathbf{w}'' \in \mathcal{V}_j(p)$.

From Definition 8.21, it is easy to see that T' is a $(|\varphi| - i, P)$ -encoding of $(\mathcal{K}', \mathbf{w}')$, concluding the induction step.

Thus, we now know that T is a $(|\varphi|, P)$ -encoding of some pointed forest $(\mathcal{K}, \mathbf{w})$. From $T \models \tau(\varphi)$ together with Lemma 8.22, we conclude that $(\mathcal{K}, \mathbf{w}) \models \varphi$. \square

G

Appendix of Chapter 9

Contents

Proof of Lemma 9.11.	577
Proof of Lemma 9.16.	578
Proof of Lemma 9.18.	579
Proof of Lemma 9.29.	580
Proof of Lemma 9.42.	581
Proof of Theorem 9.43.	583
Proof of Lemma 9.44.	583
Proof of Lemma 9.45.	584

Proof of Lemma 9.11.

As usual (see e.g. Section 4.2.2), since $\text{ML}(*)[m, s, P]$ is finite up to logical equivalence, given a pointed forest $(\mathcal{K}, \mathbf{w})$, we can define a finite *characteristic formula* $\Pi(\mathcal{K}, \mathbf{w})_{m,s}^P$ in $\text{ML}(*)[m, s, P]$ that is logically equivalent to the infinite conjunction $\bigwedge\{\varphi \in \text{ML}(*)[m, s, P] \mid (\mathcal{K}, \mathbf{w}) \models \varphi\}$. Notice that $\Pi(\mathcal{K}, \mathbf{w})_{m,s}^P$ is in $\text{ML}(*)[m, s, P]$. Moreover, the following result holds.

Lemma G.1. Let $(\mathcal{K}, \mathbf{w})$ and $(\mathcal{K}', \mathbf{w}')$ be two pointed forests. For every rank (m, s, P) , we have

$$(I) \quad (\mathcal{K}, \mathbf{w}) \models \Pi(\mathcal{K}, \mathbf{w})_{m,s}^P, \quad (II) \quad (\mathcal{K}, \mathbf{w}) \models \Pi(\mathcal{K}', \mathbf{w}')_{m,s}^P \text{ iff } (\mathcal{K}', \mathbf{w}') \models \Pi(\mathcal{K}, \mathbf{w})_{m,s}^P.$$

The proof of this lemma carries out as the one of Lemma 4.17 (Chapter 4), and it is thus omitted. Let us prove Lemma 9.11, whose statement is recalled below.

Lemma 9.11. $(\mathcal{K}, \mathbf{w}) \not\approx_{m,s}^P (\mathcal{K}', \mathbf{w}')$ iff there is φ in $\text{ML}(*)[m, s, P]$ s.t. $(\mathcal{K}, \mathbf{w}) \models \varphi$ and $(\mathcal{K}', \mathbf{w}') \not\models \varphi$.

Proof. The proof is similar to the one of Theorem 4.15. The right to left direction, i.e. the soundness of the games, follows by structural induction on φ . The left to right direction, i.e. the completeness of the games, is by induction on the rank of the formula.

(\Leftarrow): We first prove that the games are sound (right to left direction), by structural induction on φ . The base cases for \top and atomic propositions is trivial, and so are the induction steps for Boolean connectives. Let us look at the modality \Diamond and the separating conjunction.

induction step: $\varphi = \Diamond\psi$. By hypothesis, $(\mathcal{K}, \mathbf{w}) \models \Diamond\psi$ and $(\mathcal{K}', \mathbf{w}') \not\models \Diamond\psi$. Then there is a world w_1 accessible from w and such that $(\mathcal{K}, w_1) \models \psi$. Moreover by definition the modal depth of $\Diamond\psi$ is at least 1 and the spoiler can play a modal move. Then, the spoiler chooses the structure $(\mathcal{K}, \mathbf{w})$ and chooses exactly w_1 . The duplicator has then to reply by choosing a world w'_1 accessible from w' (otherwise the spoiler wins and the result clearly follows). Since $(\mathcal{K}', \mathbf{w}') \not\models \Diamond\psi$, it holds that $(\mathcal{K}', w'_1) \not\models \psi$. By the induction hypothesis, it holds that $(\mathcal{K}, w_1) \not\approx_{m-1,s}^P (\mathcal{K}', w'_1)$. Hence, by choosing w_1 , the spoiler builds a winning strategy for the game $((\mathcal{K}, \mathbf{w}), (\mathcal{K}', \mathbf{w}'), (m, s, P))$.

induction step: $\varphi = \psi * \chi$. By hypothesis, $(\mathcal{K}, \mathbf{w}) \models \psi * \chi$ and $(\mathcal{K}', \mathbf{w}') \not\models \psi * \chi$. Then, there are \mathcal{K}_1 and \mathcal{K}_2 such that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$, $(\mathcal{K}_1, \mathbf{w}) \models \psi$ and $(\mathcal{K}_2, \mathbf{w}) \models \chi$. Moreover, by definition, the number of nested stars in $\psi * \chi$ is at least 1 and therefore the spoiler can play a spatial move. The spoiler chooses the structure $(\mathcal{K}, \mathbf{w})$ and chooses exactly \mathcal{K}_1 and \mathcal{K}_2 . The duplicator has then to reply by choosing two structures \mathcal{K}'_1 and \mathcal{K}'_2 such that $\mathcal{K}'_1 + \mathcal{K}'_2 = \mathcal{K}'$. Since $(\mathcal{K}', \mathbf{w}') \not\models \psi * \chi$, either $(\mathcal{K}'_1, \mathbf{w}') \not\models \psi$ or $(\mathcal{K}'_2, \mathbf{w}') \not\models \chi$. If the former holds, then by the induction hypothesis, $(\mathcal{K}_1, \mathbf{w}) \not\approx_{m,s-1}^P (\mathcal{K}'_1, \mathbf{w}')$. Hence, by choosing to continue the game on $((\mathcal{K}_1, \mathbf{w}), (\mathcal{K}'_1, \mathbf{w}'), (m, s-1, P))$ the spoiler builds a winning strategy for the game $((\mathcal{K}, \mathbf{w}), (\mathcal{K}', \mathbf{w}'), (m, s, P))$. Symmetrically, if instead $(\mathcal{K}'_2, \mathbf{w}') \not\models \chi$ then by the induction hypothesis $(\mathcal{K}_2, \mathbf{w}) \not\approx_{m,s-1}^P (\mathcal{K}'_2, \mathbf{w}')$. Hence, by choosing to continue the game on $((\mathcal{K}_2, \mathbf{w}), (\mathcal{K}'_2, \mathbf{w}'), (m, s-1, P))$, the spoiler builds a winning strategy for the game $((\mathcal{K}, \mathbf{w}), (\mathcal{K}', \mathbf{w}'), (m, s, P))$. In either case, we conclude that $(\mathcal{K}, \mathbf{w}) \not\approx_{m,s}^P (\mathcal{K}', \mathbf{w}')$.

(\Rightarrow): We prove that the games are complete, by induction on (m, s) and by cases on the first move that the spoiler makes in his winning strategy for the game $((\mathcal{K}, \mathbf{w}), (\mathcal{K}', \mathbf{w}'), (m, s, P))$.

base case: $m = 0$ and $s = 0$. Since the spoiler has a winning strategy, in particular it wins the game of rank $(0, 0, P)$ and therefore by definition of the game it must hold that there is a propositional symbol $p \in P$ such that $(\mathcal{K}, \mathbf{w}) \models p$ iff $(\mathcal{K}', \mathbf{w}') \not\models p$. If $(\mathcal{K}, \mathbf{w}) \models p$, then φ (as in the statement) is p . Otherwise (i.e. $(\mathcal{K}', \mathbf{w}') \models p$) we take $\varphi = \neg p$.

Notice that this case also holds for games on arbitrary rank (m, s, P) : the spoiler wins simply from the conditions of the game that are imposed before each round.

Induction case: the spoiler plays a modal move. Notice that then $m \geq 1$. Suppose that, by following its strategy, the spoiler chooses (\mathcal{K}, w) and a child w_1 of w . By Lemma G.1, we have that $(\mathcal{K}, w_1) \models \Pi(\mathcal{K}, w_1)_{m-1,s}^P$. Let φ be defined as the formula $\Diamond \Pi(\mathcal{K}, w_1)_{m-1,s}^P$. By definition, $(\mathcal{K}, w) \models \varphi$ and $\varphi \in \text{ML}(*[m, s, P]$. *Ad absurdum*, assume $(\mathcal{K}', w') \models \varphi$. Then there is a world w'_1 accessible from w' such that $\mathcal{K}', w'_1 \models \Pi(\mathcal{K}, w_1)_{m-1,s}^P$. By Lemma G.1 there is no formula in $\text{ML}(*[m-1, s, P]$ that can discriminate between (\mathcal{K}, w_1) and (\mathcal{K}', w'_1) . As our games are determined, by the induction hypothesis this implies that the duplicator has a winning strategy for the game $((\mathcal{K}, w_1), (\mathcal{K}', w'_1), (m-1, s, P))$. This is contradictory, as by hypothesis the spoiler has a winning strategy and the move it played is part of this strategy. Hence, $(\mathcal{K}, w) \models \varphi$ and $(\mathcal{K}', w') \not\models \varphi$.

The proof is analogous for the case where the spoiler chooses (\mathcal{K}', w') and a world w'_1 accessible from w . In this case we obtain $(\mathcal{K}, w) \not\models \psi$ and $(\mathcal{K}', w') \models \psi$, where ψ is defined as $\Diamond \Pi(\mathcal{K}', w'_1)_{m-1,s}^P$. Hence, we take φ (as in the statement) defined as $\neg \psi$.

Induction case: the spoiler plays a spatial move. Notice that then $s \geq 1$. Suppose that, by following its strategy, the spoiler chooses (\mathcal{K}, w) and two Kripke-style finite forests \mathcal{K}_1 and \mathcal{K}_2 such that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}$. Recall that, by Lemma G.1, $\mathcal{K}_1, w \models \Pi(\mathcal{K}_1, w)_{m,s-1}^P$ and $\mathcal{K}_2, w \models \Pi(\mathcal{K}_2, w)_{m,s-1}^P$. Let φ be defined as $\Pi(\mathcal{K}_1, w)_{m,s-1}^P * \Pi(\mathcal{K}_2, w)_{m,s-1}^P$. By definition $(\mathcal{K}, w) \models \varphi$ and $\varphi \in \text{ML}(*[m, s, P]$. *Ad absurdum*, assume $(\mathcal{K}', w') \models \varphi$. Then there are \mathcal{K}'_1 and \mathcal{K}'_2 s.t. $\mathcal{K}'_1 + \mathcal{K}'_2 = \mathcal{K}'$, $\mathcal{K}'_1, w' \models \Pi(\mathcal{K}_1, w)_{m,s-1}^P$ and $(\mathcal{K}'_2, w') \models \Pi(\mathcal{K}_2, w)_{m,s-1}^P$. Then, by Lemma G.1, there is no formula in $\text{ML}(*[m, s-1, P]$ that can discriminate between (\mathcal{K}_1, w) and (\mathcal{K}'_1, w') , or that can discriminate between (\mathcal{K}_2, w) and (\mathcal{K}'_2, w') . As our games are determined, by the induction hypothesis this implies that the duplicator has a winning strategy for both $((\mathcal{K}_1, w), (\mathcal{K}'_1, w'), (m, s-1, P))$ and $((\mathcal{K}_2, w), (\mathcal{K}'_2, w'), (m, s-1, P))$. This leads to a contradiction, as by hypothesis the spoiler has a winning strategy and the move it played is part of this strategy. Hence, $(\mathcal{K}, w) \models \varphi$ and $(\mathcal{K}', w') \not\models \varphi$.

The proof is analogous for the case where the spoiler chooses (\mathcal{K}', w') and two finite forests \mathcal{K}'_1 and \mathcal{K}'_2 such that $\mathcal{K}'_1 + \mathcal{K}'_2 = \mathcal{K}'$. In this case we obtain $(\mathcal{K}, w) \not\models \psi$ and $(\mathcal{K}', w') \models \psi$ where ψ is defined as $\Pi(\mathcal{K}'_1, w')_{m,s-1}^P * \Pi(\mathcal{K}'_2, w')_{m,s-1}^P$. Hence, we take φ (as in the statement) defined as $\neg \psi$. \square

Proof of Lemma 9.16.

Lemma 9.16. Let $\mathbf{ax} \in \text{Aux}$ and $0 < i \leq j \in \mathbb{N}$. Suppose $(\mathcal{K}, w) \models \text{init}(j)$.

1. $(\mathcal{K}, w) \models \text{nom}_i(\mathbf{ax})$ if and only if \mathbf{ax} is a nominal for the depth i .
2. Suppose $(\mathcal{K}, w) \models \text{nom}_i(\mathbf{ax})$. $(\mathcal{K}, w) \models @_{\mathbf{ax}}^i \varphi$ if and only if the world (say w') corresponding to the nominal \mathbf{ax} for the depth i is such that $(\mathcal{K}, w') \models \varphi$.
3. $(\mathcal{K}, w) \models \text{nom}_i(\mathbf{ax} \neq \mathbf{bx})$ iff \mathbf{ax} and \mathbf{bx} are nominals for the depth i , for two distinct worlds.

Lemma 9.16(1) is proved during Chapter 9. We recall that:

$$@_{\mathbf{ax}}^i \varphi \stackrel{\text{def}}{=} \langle t \rangle^i (\Diamond \mathbf{ax} \wedge \varphi), \quad \text{nom}_i(\mathbf{ax} \neq \mathbf{bx}) \stackrel{\text{def}}{=} \text{nom}_i(\mathbf{ax}) \wedge \text{nom}_i(\mathbf{bx}) \wedge \neg @_{\mathbf{ax}}^i \Diamond \mathbf{bx}.$$

In the following two proofs, let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.

Proof of Lemma 9.16(2). Since $(\mathcal{K}, w) \models \text{init}(j) \wedge \text{nom}_i(\text{ax})$, by Lemma 9.16(1), ax is a nominal for the depth i . In the following, let w' be the world in $R^i(w)$ corresponding to the nominal ax (i.e. w' has an ax -child).

(\Rightarrow) : Suppose $(\mathcal{K}, w) \models @_\text{ax}^i \varphi$. By definition, there is $w'' \in R^i(w)$ s.t. $(\mathcal{K}, w'') \models \Diamond \text{ax} \wedge \varphi$. Since ax is a nominal for the depth i , we conclude that $w' = w''$ and hence $(\mathcal{K}, w'') \models \varphi$.

(\Leftarrow) : Suppose that $(\mathcal{K}, w') \models \varphi$. By definition, w' is the world corresponding to the nominal ax (for the depth i). Hence $(\mathcal{K}, w') \models \Diamond \text{ax}$. Since $w' \in R^i(w)$ by $(\mathcal{K}, w) \models \text{init}(j)$ we conclude that there is a path of t -nodes from w to w' , of length i . Thus, $(\mathcal{K}, w) \models \langle t \rangle^i (\Diamond \text{ax} \wedge \varphi)$. \square

Proof of Lemma 9.16(3). Thanks to Lemma 9.16((1) and (2)), this proof is straightforward.

(\Rightarrow) : Suppose $(\mathcal{K}, w) \models \text{nom}_i(\text{ax} \neq \text{bx})$. By Lemma 9.16(1) ax and bx are nominals for depth i . Let w_{ax} (resp. w_{bx}) be the world in $R^i(w)$ corresponding to the nominal ax (resp. bx). Notice that, in particular, $(\mathcal{K}, w_{\text{bx}}) \models \Diamond \text{bx}$. By $(\mathcal{K}, w) \models \neg @_\text{ax}^i \Diamond \text{bx}$ and Lemma 9.16(2), we conclude that $(\mathcal{K}, w_{\text{ax}}) \not\models \Diamond \text{bx}$. Thus, $w_{\text{ax}} \neq w_{\text{bx}}$.

(\Leftarrow) : This direction is analogous and simply relies on Lemma 9.16((1) and (2)). \square

Proof of Lemma 9.18.

We recall the (standard) characterisation of $\mathbf{n}(w_{\text{ax}}) < \mathbf{n}(w_{\text{bx}})$ and $\mathbf{n}(w_{\text{ax}}) + 1 = \mathbf{n}(w_{\text{bx}})$ in terms of the binary representation of $\mathbf{n}(w_{\text{ax}})$ and $\mathbf{n}(w_{\text{bx}})$, where $\mathbf{n}(w_{\text{ax}}), \mathbf{n}(w_{\text{bx}}) \in [0, 2^n - 1]$. Let b_{ax} and b_{bx} be the n -bit representation of $\mathbf{n}(w_{\text{ax}})$ and $\mathbf{n}(w_{\text{bx}})$, respectively. We have,

$$\mathbf{n}(w_{\text{ax}}) < \mathbf{n}(w_{\text{bx}}) \quad \text{iff} \quad b_{\text{ax}} = b \cdot 0 \cdot b' \text{ and } b_{\text{bx}} = b \cdot 1 \cdot b'' \text{ for some } b, b', b'' \in \{0, 1\}^*,$$

$$\mathbf{n}(w_{\text{ax}}) + 1 = \mathbf{n}(w_{\text{bx}}) \quad \text{iff} \quad b_{\text{ax}} = b \cdot 0 \cdot 1^k \text{ and } b_{\text{bx}} = b \cdot 1 \cdot 0^k, \text{ for some } b \in \{0, 1\}^*, k \in [0, n].$$

where $b_1 \cdot b_2$ is the concatenation of bit vectors, b^0 is the zero size bit vector and $b^{k+1} \stackrel{\text{def}}{=} b \cdot b^k$. Recall that the rightmost bit in b_{ax} and b_{bx} is the least significant one.

Lemma 9.18. Let $\text{ax} \neq \text{bx} \in \text{Aux}$ and $j \geq 1$. Suppose $(\mathcal{K}, w) \models \text{init}(j) \wedge \text{fork}_j^j(\text{ax}, \text{bx})$. Let w_{ax} (resp. w_{bx}) be the world corresponding to the nominal ax (resp. bx) at depth j .

1. $(\mathcal{K}, w) \models [\text{ax} < \text{bx}]_j^j$ if and only if $\mathbf{n}(w_{\text{ax}}) < \mathbf{n}(w_{\text{bx}})$,
2. Let $j = 1$. $(\mathcal{K}, w) \models [\text{bx} = \text{ax}+1]_1$ if and only if $\mathbf{n}(w_{\text{ax}}) + 1 = \mathbf{n}(w_{\text{bx}})$.

Proof of Lemma 9.18(1). Let us recall the definition of $[\text{ax} < \text{bx}]_j^j$:

$$\bigvee_{u \in [1, n]} (@_\text{ax}^j \neg p_u \wedge @_\text{bx}^j p_u \wedge \bigwedge_{v \in [u+1, n]} (@_\text{ax}^j p_v \Leftrightarrow @_\text{bx}^j p_v)).$$

The proof uses the characterisation of $<$, in terms of binary representation, given above. Below, since $(\mathcal{K}, w) \models \text{fork}_j^j(\text{ax}, \text{bx})$, let w_{ax} (resp. w_{bx}) be the world corresponding to the nominal ax (resp. bx) for the depth j .

(\Rightarrow) : Suppose $(\mathcal{K}, w) \models [\text{ax} < \text{bx}]_j^j$, and so there is $u \in [1, n]$ such that

$$(\mathcal{K}, w) \models @_\text{ax}^j \neg p_u \wedge @_\text{bx}^j p_u \wedge \bigwedge_{v \in [u+1, n]} (@_\text{ax}^j p_v \Leftrightarrow @_\text{bx}^j p_v).$$

Let b_{ax} be the bit vector $b_{\text{ax},n} \dots b_{\text{ax},1}$ such that, for every $i \in [1, n]$, $b_{\text{ax},i} = 1$ if and only if $w_{\text{ax}} \in \mathcal{V}(p_i)$. Similarly, let b_{bx} be the bit vector $b_{\text{bx},n} \dots b_{\text{bx},1}$ such that, for every $i \in [1, n]$, $b_{\text{bx},i} = 1$ if and only if $w_{\text{bx}} \in \mathcal{V}(p_i)$. From Lemma 9.16 and $(\mathcal{K}, w) \models @_\text{ax}^j \neg p_u \wedge @_\text{bx}^j p_u$, we conclude that $b_{\text{ax},u} = 0$ and $b_{\text{bx},u} = 1$. From $(\mathcal{K}, w) \models \bigwedge_{v \in [u+1, n]} (@_\text{ax}^j p_v \Leftrightarrow @_\text{bx}^j p_v)$, we conclude that for every $v \in [u+1, n]$, $b_{\text{ax},v} = b_{\text{bx},v}$. So, there are $b, b', b'' \in \{0, 1\}^*$ such that $b_{\text{ax}} = b \cdot 0 \cdot b'$ and $b_{\text{bx}} = b \cdot 1 \cdot b''$. We derive $\mathbf{n}(w_{\text{ax}}) < \mathbf{n}(w_{\text{bx}})$.

(\Leftarrow): This direction follows similar arguments. Briefly, suppose $\mathbf{n}(w_{\text{ax}}) < \mathbf{n}(w_{\text{bx}})$, and so there are $b, b', b'' \in \{0, 1\}^*$ such that $b_{\text{ax}} = b \cdot 0 \cdot b'$ and $b_{\text{bx}} = b \cdot 1 \cdot b''$. Let $b_{\text{ax},n}, \dots, b_{\text{ax},1} \in \{0, 1\}$ be such that $b_{\text{ax}} = b_{\text{ax},n} \dots b_{\text{ax},1}$, and $b_{\text{bx},n}, \dots, b_{\text{bx},1} \in \{0, 1\}$ be such that $b_{\text{bx}} = b_{\text{bx},n} \dots b_{\text{bx},1}$. From $b_{\text{ax}} = b \cdot 0 \cdot b'$ and $b_{\text{bx}} = b \cdot 1 \cdot b''$, there is $u \in [1, n]$ such that $b_{\text{ax},u} = 0$, $b_{\text{bx},u} = 1$ and for all $v \in [u+1, n]$, $b_{\text{ax},v} = b_{\text{bx},v}$. From the encoding of $\mathbf{n}(w_{\text{ax}})$ and $\mathbf{n}(w_{\text{bx}})$ using p_1, \dots, p_n , we derive that $w_{\text{ax}} \notin \mathcal{V}(p_u)$, $w_{\text{bx}} \in \mathcal{V}(p_u)$, and for every $v \in [u+1, n]$, $w_{\text{ax}} \in \mathcal{V}(p_v)$ if and only if $w_{\text{bx}} \in \mathcal{V}(p_v)$. By definition, we conclude that $(\mathcal{K}, w) \models [\text{ax} < \text{bx}]_j^j$. \square

Proof of Lemma 9.18(2). Let us recall the definition of $[\text{bx} = \text{ax}+1]_1$:

$$\bigvee_{u \in [1, n]} (@_{\text{ax}}^1 (\neg p_u \wedge \bigwedge_{v \in [1, u-1]} p_v) \wedge @_{\text{bx}}^1 (p_u \wedge \bigwedge_{v \in [1, u-1]} \neg p_v) \wedge \bigwedge_{v \in [u+1, n]} (@_{\text{ax}}^1 p_v \Leftrightarrow @_{\text{bx}}^1 p_v)).$$

The proof is very similar to the one of Lemma 9.18(1), and uses the characterisation of +1 in terms of binary representation, given above. The proof uses standard properties of numbers encoded in binary. Below, since $(\mathcal{K}, w) \models \text{fork}_j^j(\text{ax}, \text{bx})$, let w_{ax} (resp. w_{bx}) be the world corresponding to the nominal ax (resp. bx) for the depth j .

(\Rightarrow): Suppose $(\mathcal{K}, w) \models [\text{bx} = \text{ax}+1]_1$, and so there is $u \in [1, n]$ such that

$$(\mathcal{K}, w) \models @_{\text{ax}}^1 (\neg p_u \wedge \bigwedge_{v \in [1, u-1]} p_v) \wedge @_{\text{bx}}^1 (p_u \wedge \bigwedge_{v \in [1, u-1]} \neg p_v) \wedge \bigwedge_{v \in [u+1, n]} (@_{\text{ax}}^1 p_v \Leftrightarrow @_{\text{bx}}^1 p_v)$$

Let b_{ax} be the bit vector $b_{\text{ax},n} \dots b_{\text{ax},1}$ such that, for every $i \in [1, n]$, $b_{\text{ax},i} = 1$ if and only if $w_{\text{ax}} \in \mathcal{V}(p_i)$. Similarly, let b_{bx} be the bit vector $b_{\text{bx},n} \dots b_{\text{bx},1}$ such that, for every $i \in [1, n]$, $b_{\text{bx},i} = 1$ if and only if $w_{\text{bx}} \in \mathcal{V}(p_i)$. From Lemma 9.16 and the fact that (\mathcal{K}, w) satisfies $@_{\text{ax}}^1 (\neg p_u \wedge \bigwedge_{v \in [1, u-1]} p_v) \wedge @_{\text{bx}}^1 (p_u \wedge \bigwedge_{v \in [1, u-1]} \neg p_v)$, we conclude that $b_{\text{ax},u} = 0$, $b_{\text{bx},u} = 1$, and for every $v \in [1, u-1]$, $b_{\text{ax},v} = 1$ and $b_{\text{bx},v} = 0$. From $(\mathcal{K}, w) \models \bigwedge_{v \in [u+1, n]} (@_{\text{ax}}^j p_v \Leftrightarrow @_{\text{bx}}^j p_v)$, we conclude that for every $v \in [u+1, n]$, $b_{\text{ax},v} = b_{\text{bx},v}$. So, there are $b \in \{0, 1\}^*$ and $k \in [0, n]$ such that $b_{\text{ax}} = b \cdot 0 \cdot 1^k$ and $b_{\text{bx}} = b \cdot 1 \cdot 0^k$. We derive $\mathbf{n}(w_{\text{ax}}) < \mathbf{n}(w_{\text{bx}})$.

(\Leftarrow): This direction follows similar arguments. Briefly, suppose $\mathbf{n}(w_{\text{ax}}) < \mathbf{n}(w_{\text{bx}})$, and so there are $b \in \{0, 1\}^*$ and $k \in [0, n]$ such that $b_{\text{ax}} = b \cdot 0 \cdot 1^k$ and $b_{\text{bx}} = b \cdot 1 \cdot 0^k$. Let $b_{\text{ax},n}, \dots, b_{\text{ax},1} \in \{0, 1\}$ be such that $b_{\text{ax}} = b_{\text{ax},n} \dots b_{\text{ax},1}$, and $b_{\text{bx},n}, \dots, b_{\text{bx},1} \in \{0, 1\}$ be such that $b_{\text{bx}} = b_{\text{bx},n} \dots b_{\text{bx},1}$. From $b_{\text{ax}} = b \cdot 0 \cdot 1^k$ and $b_{\text{bx}} = b \cdot 1 \cdot 0^k$, there is $u \in [1, n]$ such that $b_{\text{ax},u} = 0$, $b_{\text{bx},u} = 1$, for every $v \in [1, u-1]$, $b_{\text{ax},v} = 1$ and $b_{\text{bx},v} = 0$, and for all $v \in [u+1, n]$, $b_{\text{ax},v} = b_{\text{bx},v}$. From the encoding of $\mathbf{n}(w_{\text{ax}})$ and $\mathbf{n}(w_{\text{bx}})$ using p_1, \dots, p_n , we conclude that $w_{\text{ax}} \notin \mathcal{V}(p_u)$, $w_{\text{bx}} \in \mathcal{V}(p_u)$, for every $v \in [1, u-1]$, $w_{\text{ax}} \in \mathcal{V}(p_v)$ and $w_{\text{bx}} \notin \mathcal{V}(p_v)$, and for every $v \in [u+1, n]$, $w_{\text{ax}} \in \mathcal{V}(p_v)$ if and only if $w_{\text{bx}} \in \mathcal{V}(p_v)$. By definition, $(\mathcal{K}, w) \models [\text{bx} = \text{ax}+1]_1$. \square

Proof of Lemma 9.29.

We recall the definition of $\text{compl}(j)$:

$$\neg \left(\square \perp * \left([t](\text{type}_{\text{1sr}}(j-1) \wedge \diamond y) \wedge \text{nom}_1(x) \wedge @_x^1 \neg 1_j \wedge \neg (\top * (\text{fork}_j^1(x, y) \wedge [y = x+1]_j)) \right) \right).$$

Lemma 9.29. Let (\mathcal{K}, w) be a pointed forest satisfying $\text{init}(j) \wedge \text{aux} \wedge \text{sub}(j)$. $(\mathcal{K}, w) \models \text{compl}(j)$ if and only if (\mathcal{K}, w) satisfies (compl_j) .

Proof. The proof is similar to Lemma 9.20. Recall that (compl_j) states that for every t -node $w_1 \in R(w)$ where $\mathbf{n}_{j-1}(w_1) < t(j, n)-1$, there is a t -node $w_2 \in R(w)$ s.t. $\mathbf{n}_{j-1}(w_2) = \mathbf{n}_{j-1}(w_1)+1$. Notice that from $(\mathcal{K}, w) \models \text{sub}(j)$, every t -child w' of w is such that $(\mathcal{K}, w') \models \text{type}(j-1)$.

(\Rightarrow): Suppose $(\mathcal{K}, w) \models \text{compl}(j)$. This implies that for every subforest $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K} , if $R'(w) = R(w)$ and $(\mathcal{K}', w) \models [t](\text{type}_{1sr}(j-1) \wedge \Diamond y) \wedge \text{nom}_1(x) \wedge @_x^1 \neg 1_j$, then

$$(\mathcal{K}', w) \models \top * (\text{fork}_j^1(x, y) \wedge [y = x+1]_j).$$

Let us pick a t -node $w_x \in R'(w) = R(w)$ such that $\mathbf{n}_{j-1}(w_x) < t(j, n) - 1$. We show that there must be a world $w_y \in R'(w)$ such that $\mathbf{n}_{j-1}(w_y) = \mathbf{n}_{j-1}(w_x) + 1$.

Recall that, by $(\mathcal{K}, w) \models \text{init}(j) \wedge \text{aux}$, every t -child of w has exactly one x -child and one y -child. Consider the subforest $\mathcal{K}'' = (\mathcal{W}, R'', \mathcal{V})$ such that R'' is obtained from R by removing every x -nodes in $R^2(w)$, with the exception of the only x -child of w_x . Since only x -nodes of $R^2(w)$ are lost, every t -child w' of w is such that $(\mathcal{K}', w') \models \text{type}(j-1) \wedge \Diamond y$ and all the t -children of w' have exactly one $\{1, s, r\}$ -child for each proposition among $1, s$ and r . Therefore, $(\mathcal{K}', w) \models \text{type}_{1sr}(j-1) \wedge \Diamond y$. Moreover, in \mathcal{K}' , the world w_x is the only child of w with a x -child, which implies $(\mathcal{K}', w) \models \text{nom}_1(x)$. From $(\mathcal{K}, w) \models \text{type}(j-1)$ and by Lemma 9.15, we conclude that $\mathbf{n}_{j-1}(w_x)$ is the same number in both \mathcal{K} and \mathcal{K}' , and thus from $\mathbf{n}_{j-1}(w_x) < t(j, n) - 1$ we derive $(\mathcal{K}, w) \models @_x^1 \neg 1_j$. From $(\mathcal{K}, w) \models \text{compl}(j)$, we conclude that $(\mathcal{K}', w) \models \top * (\text{fork}_j^1(x, y) \wedge [y = x+1]_j)$, which implies that there is a subforest $\mathcal{K}'' = (\mathcal{W}, R'', \mathcal{V})$ of \mathcal{K}' such that $(\mathcal{K}'', w) \models \text{fork}_j^1(x, y) \wedge [y = x+1]_j$. By Lemmata 9.23 and 9.27, there is $w_y \in R''(w)$ such that $\mathbf{n}_{j-1}(w_y) = \mathbf{n}_{j-1}(w_x) + 1$ (in \mathcal{K}''), and moreover (\mathcal{K}'', w_y) and (\mathcal{K}'', w_y) satisfy $\text{type}(j-1)$. By Lemma 9.15, $\mathbf{n}(w_y) = \mathbf{n}(w_x) + 1$ holds also in \mathcal{K} . Thus, (\mathcal{K}, w) satisfies (compl_j) .

(\Leftarrow): Suppose that (\mathcal{K}, w) satisfies (compl_j) . Ad absurdum assume that $(\mathcal{K}, w) \not\models \text{compl}(j)$. Then, there is a submodel $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$ of \mathcal{K} such that $R'(w) = R(w)$ and

$$(\mathcal{K}', w) \models [t](\text{type}_{1sr}(j-1) \wedge \Diamond y) \wedge \text{nom}_1(x) \wedge @_x^1 \neg 1_j \wedge \neg(\top * (\text{fork}_j^1(x, y) \wedge [y = x+1]_j)).$$

From $(\mathcal{K}', w) \models [t](\text{type}_{1sr}(j-1) \wedge \Diamond y)$ every t -child w' of w is such that $(\mathcal{K}, w') \models \text{type}(j-1)$, it has a y -child, and all the t -children of w' have exactly one $\{1, s, r\}$ -child for each proposition among $1, s$ and r . From $(\mathcal{K}', w) \models \text{nom}_1(x) \wedge @_x^1 \neg 1_j$, there is an t -children w_x of w that corresponds to the nominal (for the depth 1) x , and moreover $\mathbf{n}_{j-1}(w_x) < t(j, n) - 1$. By Lemma 9.15, $\mathbf{n}_{j-1}(w_x) < t(j, n) - 1$ also holds with respect to \mathcal{K} , and thus from (compl_j) we conclude that there is a t -node w_y such that $\mathbf{n}_{j-1}(w_y) = \mathbf{n}_{j-1}(w_x) + 1$ (w.r.t. \mathcal{K}). As $R'(w) = R(w)$, $w_y \in R'(w)$ and so $(\mathcal{K}', w_y) \models \text{type}_{1sr}(j-1) \wedge \Diamond y$. This allows us to consider the subforest $\mathcal{K}'' = (\mathcal{W}, R'', \mathcal{V})$ obtained from \mathcal{K}' by removing all children of w , with the exception of w_x and w_y , and by removing the only y -child of w_x . By Lemma 9.23, the resulting pointed forest is such that $(\mathcal{K}'', w) \models \text{fork}_j^1(x, y)$, where w_x and w_y are the worlds corresponding to the nominals x and y , respectively, and both (\mathcal{K}'', w_x) and (\mathcal{K}'', w_y) satisfy $\text{type}(j-1)$. By Lemma 9.15, this implies that $\mathbf{n}_{j-1}(w_y) = \mathbf{n}_{j-1}(w_x) + 1$ holds with respect to \mathcal{K}'' . However, by Lemma 9.27 this implies $(\mathcal{K}'', w) \models [y = x+1]_j$, which allows us to conclude that (\mathcal{K}', w) satisfies $\top * (\text{fork}_j^1(x, y) \wedge [y = x+1]_j)$: a contradiction. Thus, $(\mathcal{K}, w) \models \text{compl}(j)$. \square

Proof of Lemma 9.42.

Lemma 9.42. Let φ be a formula in $\text{ML}(\ast)$ written without using atomic propositions from $\{q_1, q_2, q_3\}$, and let $i \in [1, 3]$ and $m \geq \text{md}(\varphi)$. Let \mathcal{K} and \mathcal{K}' be two Kripke trees and w be a world, so that $\mathcal{K} \triangleright_{m,i}^w \mathcal{K}'$. $(\mathcal{K}, w) \models \varphi$ in $\text{ML}(\ast)$ if and only if $(\mathcal{K}', w) \models \tau_i(\varphi)$ in QK^t .

We recall that, given two Kripke tree $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$, as well as a world $w \in \mathcal{W}$, $m \in \mathbb{N}$ and $i \in [1, 3]$, we write $\mathcal{K} \triangleright_{m,i}^w \mathcal{K}'$ whenever (Definition 9.41):

- (1) \mathcal{W} is finite, $\mathcal{W} \subseteq \mathcal{W}'$ and $R \subseteq R'$,

- (2) for all $p \in AP \setminus \{q_1, q_2, q_3\}$, the satisfaction of p is preserved, i.e. $\mathcal{V}(p) = \mathcal{V}'(p) \cap \mathcal{W}$,
- (3) the worlds reachable in at most m steps in R are, among the ones reachable in at most m steps in R' , exactly those in $\mathcal{V}'(q_i)$, i.e. $\bigcup_{j \in [1, m]} R^j(\mathbf{w}) = \{\mathbf{w}' \in R'^j(\mathbf{w}) \cap \mathcal{V}'(q_i) \mid j \in [1, m]\}$.

Proof. As usual, this result is proved by structural induction on φ . The base cases for \top and atomic propositions, as well as the induction steps dealing with Boolean connectives, are trivial and thus omitted. Below, we show the induction steps for \Diamond and $*$. Let $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, and $\mathcal{K}' = (\mathcal{W}', R', \mathcal{V}')$.

inductin step: $\varphi = \Diamond\psi$. Notice that $m \geq \text{md}(\varphi) \geq 1$. Moreover, by definition of $\triangleright_{m,i}^\mathbf{w}$, it is quite easy to see that $\mathcal{K} \triangleright_{m,i}^\mathbf{w} \mathcal{K}'$ implies $\mathcal{K} \triangleright_{m-1,i}^{\mathbf{w}'} \mathcal{K}'$, for every $\mathbf{w}' \in R(\mathbf{w})$. We have,

$$\begin{aligned} & (\mathcal{K}, \mathbf{w}) \models \Diamond\psi, \\ \Leftrightarrow & \text{there is } \mathbf{w}' \in R(\mathbf{w}) \text{ such that } (\mathcal{K}, \mathbf{w}') \models \psi, \\ \Leftrightarrow & \text{there is } \mathbf{w}' \in R'(\mathbf{w}) \cap \mathcal{V}(q_i) \text{ such that } (\mathcal{K}', \mathbf{w}') \models \tau_i(\psi), \\ & (\text{from } R \subseteq R', \text{ the property (3) of } \triangleright_{m,i}^\mathbf{w} \text{ with } m \geq 1, \text{ and thanks to } \mathcal{K} \triangleright_{m-1,i}^{\mathbf{w}'} \mathcal{K}' \text{ which} \\ & \text{allows to apply the induction hypothesis}) \\ \Leftrightarrow & (\mathcal{K}', \mathbf{w}) \models \Diamond(q_i \wedge \tau_i(\psi)). \quad (\text{by } \models). \end{aligned}$$

induction step: $\varphi = \varphi_1 * \varphi_2$. Let $j, k \in [1, 3]$, $j < k$ and $j \neq i \neq k$.

(\Rightarrow): Suppose $(\mathcal{K}, \mathbf{w}) \models \varphi_1 * \varphi_2$. There are two Kripke-style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2$, $(\mathcal{K}_1, \mathbf{w}) \models \varphi_1$ and $(\mathcal{K}_2, \mathbf{w}) \models \varphi_2$. We update update the valuation \mathcal{V}' of the Kripke tree \mathcal{K}' , obtaining a Kripke tree $\mathcal{K}'' = (\mathcal{W}', R', \mathcal{V}'')$: we update $\mathcal{V}'(q_j)$ to the set of worlds that satisfy q_i and belong to $\pi_2(R_1)$, i.e. the second component of R_1 . Similarly, we update $\mathcal{V}'(q_k)$ to the set of worlds that satisfy q_i and belong to $\pi_2(R_2)$. Formally,

$$\mathcal{V}'' \stackrel{\text{def}}{=} \mathcal{V}'[q_j \leftarrow \mathcal{V}'(q_i) \cap \pi_2(R_1), q_k \leftarrow \mathcal{V}'(q_i) \cap \pi_2(R_2)].$$

From $R_1 \cap R_2 = \emptyset$ we conclude that $\mathcal{V}''(\mathcal{V}_j) \cap \mathcal{V}''(\mathcal{V}_k) = \emptyset$. From (3), for every world \mathbf{w}' reachable from \mathbf{w} in at most m steps, if \mathbf{w}' belongs to $\mathcal{V}''(\mathcal{V}_j) \cap \mathcal{V}''(\mathcal{V}_k)$ then it belongs to $\mathcal{V}''(q_i) = \mathcal{V}'(q_i)$. From $m \geq \text{md}(\varphi_1 * \varphi_2)$, we conclude that $(\mathcal{K}'', \mathbf{w}) \models [q_i = q_j + q_k]^{\text{md}(\varphi_1 * \varphi_2)}$. Let $\mathcal{K}'_1 = (\mathcal{W}, R_1|_{\mathbf{w}}^{\leq m}, \mathcal{V})$ and $\mathcal{K}'_2 = (\mathcal{W}, R_2|_{\mathbf{w}}^{\leq m}, \mathcal{V})$. By Proposition 9.3 and from $(\mathcal{K}_1, \mathbf{w}) \models \varphi_1$ and $(\mathcal{K}_2, \mathbf{w}) \models \varphi_2$, we derive $(\mathcal{K}'_1, \mathbf{w}) \models \varphi_1$ and $(\mathcal{K}'_2, \mathbf{w}) \models \varphi_2$. By definition of $R_1|_{\mathbf{w}}^{\leq m}$ and $R_2|_{\mathbf{w}}^{\leq m}$, the Kripke-style finite forests \mathcal{K}'_1 and \mathcal{K}'_2 are Kripke trees. Moreover, $\mathcal{K}'_1 \triangleright_{m,j}^\mathbf{w} \mathcal{K}''$ and $\mathcal{K}'_1 \triangleright_{m,k}^\mathbf{w} \mathcal{K}''$. Indeed, the properties (1) and (2) are trivially satisfied, whereas (3) holds directly from the definition of \mathcal{V}'' . By induction hypothesis, we conclude that $(\mathcal{K}'', \mathbf{w}) \models \tau_j(\varphi_1) \wedge \tau_k(\varphi_2)$. From the definition of \mathcal{K}'' and the semantics of the propositional quantifier $\exists p$, this implies $(\mathcal{K}', \mathbf{w}) \models \tau_i(\varphi_1 * \varphi_2)$.

(\Leftarrow): This direction is analogous to the previous one. Suppose $(\mathcal{K}', \mathbf{w}) \models \tau_i(\varphi_1 * \varphi_2)$, and therefore there is a Kripke tree $\mathcal{K}'' = (\mathcal{W}', R', \mathcal{V}'')$, where $\mathcal{V}'' = \mathcal{V}'[q_j \leftarrow \mathcal{W}'_j, q_k \leftarrow \mathcal{W}'_k]$ for some subsets \mathcal{W}'_1 and \mathcal{W}'_2 of \mathbf{w}' , such that

$$(\mathcal{K}'', \mathbf{w}) \models [q_i = q_j + q_k]^{\text{md}(\varphi_1 * \varphi_2)} \wedge \tau_j(\varphi_1) \wedge \tau_k(\varphi_2).$$

Let S_l , where $l \in [1, 3]$, be the set of worlds satisfying q_l and that are reachable from \mathbf{w} in at most $\text{md}(\varphi_1 * \varphi_2)$ steps. From $(\mathcal{K}'', \mathbf{w}) \models [q_i = q_j + q_k]^{\text{md}(\varphi_1 * \varphi_2)}$, the set S_i is partitioned into S_j and S_k . Let us consider two Kripke style finite forests $\mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V})$ and $\mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V})$ such that $\mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2$ and moreover

$$R_1^+(\mathbf{w}) \cap S_k = \emptyset, \quad R_2^+(\mathbf{w}) \cap S_j = \emptyset.$$

It is quite simple to see that these two Kripke forests exist. Indeed, from the property (3) of $\mathcal{K} \triangleright_{m,i}^w \mathcal{K}'$, every world $w' \in R^l(w)$, where $l \in [1, m]$, belongs to $\mathcal{V}'(q_i)$. Therefore, w' belongs to either S_j or S_k . If it belongs to $S_j \subseteq \mathcal{V}''(q_j)$, then we assign the arrow $(w'', w') \in R$ to R_1 , where w'' is the parent of w' in R , otherwise $(w' \in S_k)$, we assign it to R_2 . We consider the two Kripke trees $\mathcal{K}'_1 = (\mathcal{W}, R_1|_{\mathcal{W}}^{\leq \text{md}(\varphi_1 * \varphi_2)}, \mathcal{V})$ and $\mathcal{K}'_2 = (\mathcal{W}, R_2|_{\mathcal{W}}^{\leq \text{md}(\varphi_1 * \varphi_2)}, \mathcal{V})$. We have $\mathcal{K}'_1 \triangleright_{\text{md}(\varphi_1 * \varphi_2), j}^w \mathcal{K}''$ and $\mathcal{K}'_1 \triangleright_{\text{md}(\varphi_1 * \varphi_2), k}^w \mathcal{K}''$: the properties (1) and (2) are trivially satisfied, whereas (3) holds from the definition of S_j and S_k . As $\text{md}(\varphi_1 * \varphi_2) = \max(\text{md}(\varphi_1), \text{md}(\varphi_2))$, the induction hypothesis applies, and thus we derive $(\mathcal{K}'_1, w) \models \varphi_1$ and $(\mathcal{K}'_2, w) \models \varphi_2$. By Proposition 9.3, this implies $(\mathcal{K}_1, w) \models \varphi_1$ and $(\mathcal{K}_2, w) \models \varphi_2$. From $\mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2$, we conclude: $(\mathcal{K}, w) \models \varphi_1 * \varphi_2$. \square

Proof of Theorem 9.43.

Theorem 9.43. The satisfiability problem of QK^t and $\text{ML}(*)$ are TOWER-complete.

In order to prove Theorem 9.43, it is sufficient to show that the translation $\tau_1(\varphi)$ preserves the (un)satisfiability of the formula φ , as formalised below. Recall that the translations τ_i ($i \in [1, 3]$) uses three auxiliary atomic propositions $\{q_1, q_2, q_3\}$ which are assumed to be distinct from the atomic propositions used to write φ .

Lemma G.2. Let φ be in $\text{ML}(*)$, written without using atomic propositional from $\{q_1, q_2, q_3\}$. φ in $\text{ML}(*)$ is satisfiable if and only if $\tau_1(\varphi)$ in QK^t is satisfiable.

Proof. (\Rightarrow): Suppose that φ is satisfiable and let (\mathcal{K}, w) be a Kripke-style finite forest satisfying it, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$. By Proposition 9.3, the Kripke tree $\mathcal{K}' = (\mathcal{W}, R|_{\mathcal{W}}^{\leq \text{md}(\varphi)}, \mathcal{V})$ is such that $(\mathcal{K}', w) \models \varphi$. Let us consider the Kripke tree $\mathcal{K}'' = (\mathcal{W}, R|_{\mathcal{W}}^{\leq \text{md}(\varphi)}, \mathcal{V}[q_1 \leftarrow \mathcal{W}])$. Trivially, $\mathcal{K}' \triangleright_{\text{md}(\varphi), 1}^w \mathcal{K}''$. By Lemma 9.42, $(\mathcal{K}', w) \models \tau_1(\varphi)$.

(\Leftarrow): Suppose that $\tau_1(\varphi)$ is satisfiable, and let us consider a Kripke tree (\mathcal{K}', w) satisfying it, where $\mathcal{K}' = (\mathcal{W}, R', \mathcal{V})$. Let us consider the relation $R = \{(w_1, w_2) \in R' \mid w_2 \in \mathcal{V}(q_1)\}$, i.e. R is obtained from R' by removing all pairs $(w_1, w_2) \in R'$ such that $w_2 \notin \mathcal{V}(q_1)$. The structure $\mathcal{K}'' = (\mathcal{W}, R, \mathcal{V})$ is a Kripke-style finite forest. We define the Kripke tree $\mathcal{K} = (\mathcal{W}, R|_{\mathcal{W}}^{\text{md}(\varphi)}, \mathcal{V})$. By definition of R , it is easy to see that $\mathcal{K} \triangleright_{\text{md}(\varphi), 1}^w \mathcal{K}'$. We apply Lemma 9.42, and derive that $(\mathcal{K}'', w) \models \varphi$, in $\text{ML}(*)$. Therefore, φ is satisfiable. \square

Proof of Lemma 9.44.

Lemma 9.44. Let (\mathcal{K}, w) be a Kripke-style finite forest, where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, and let φ be a formula in $\text{ML}(*)$. $(\mathcal{K}, w) \models \varphi$ in $\text{ML}(*)$ iff $((\mathcal{W}, R^{-1}, \mathcal{V}), w) \models \varphi[\Diamond \leftarrow \Diamond^{-1}]$ in $\text{MSL}(*, \Diamond^{-1})$.

Proof. This result is proved with a straightforward structural induction on φ . The base cases for \top and atomic propositions, as well as the induction steps dealing with Boolean connectives, are trivial and thus omitted. Below, we show the induction steps for $*$ and \Diamond^{-1} .

induction step: $\varphi = \varphi_1 * \varphi_2$. This case essentially relies on the fact that, given two binary relations R_1 and R_2 , $(R_1 \cup R_2)^{-1} = R_1^{-1} \cup R_2^{-1}$. We have,

$$\begin{aligned} & (\mathcal{K}, w) \models \varphi_1 * \varphi_2 \\ \Leftrightarrow & \text{there are Kripke-style finite forests } \mathcal{K}_1 = (\mathcal{W}, R_1, \mathcal{V}) \text{ and } \mathcal{K}_2 = (\mathcal{W}, R_2, \mathcal{V}) \text{ such that} \\ & \mathcal{K}_1 = \mathcal{K}_1 + \mathcal{K}_2, (\mathcal{K}_1, w) \models \varphi_1 \text{ and } (\mathcal{K}_2, w) \models \varphi_2. \text{ Notice that } R = R_1 \cup R_2, \end{aligned}$$

- \Leftrightarrow there are acyclic Kripke-style finite functions $\mathcal{K}'_1 = (\mathcal{W}, R_1^{-1}, \mathcal{V}), \mathcal{K}'_2 = (\mathcal{W}, R_2^{-1}, \mathcal{V})$ s.t. $R^{-1} = R_1^{-1} \cup R_2^{-1}$, $R_1^{-1} \cap R_2^{-1} = \emptyset$, $(\mathcal{K}'_1, w) \models \varphi_1[\Diamond \leftarrow \Diamond^{-1}], (\mathcal{K}'_2, w) \models \varphi_2[\Diamond \leftarrow \Diamond^{-1}]$, (by induction hypothesis and definition of R_1 and R_2),
- $\Leftrightarrow ((\mathcal{W}, R^{-1}, \mathcal{V}), w) \models (\varphi_1 * \varphi_2)[\Diamond \leftarrow \Diamond^{-1}]$.
(by \models and since $(\varphi_1 * \varphi_2)[\Diamond \leftarrow \Diamond^{-1}]$ is equal to $\varphi_1[\Diamond \leftarrow \Diamond^{-1}] * \varphi_2[\Diamond \leftarrow \Diamond^{-1}]$).

induction step: $\varphi = \Diamond\psi$. We have,

- $(\mathcal{K}, w) \models \Diamond\psi$,
- \Leftrightarrow there is $w' \in \mathcal{W}$ such that $(w, w') \in R$ and $(\mathcal{K}, w') \models \psi$,
- \Leftrightarrow there is $w' \in \mathcal{W}$ such that $(w', w) \in R^{-1}$ and $((\mathcal{W}, R^{-1}, \mathcal{V}), w') \models \psi[\Diamond \leftarrow \Diamond^{-1}]$, (by induction hypothesis)
- $\Leftrightarrow ((\mathcal{W}, R^{-1}, \mathcal{V}), w) \models (\Diamond\psi)[\Diamond \leftarrow \Diamond^{-1}]$.
(by \models and since $(\Diamond\psi)[\Diamond \leftarrow \Diamond^{-1}]$ is equal to $\Diamond^{-1}(\psi[\Diamond \leftarrow \Diamond^{-1}])$). \square

Proof of Lemma 9.45.

Lemma 9.45. φ in $\text{ML}(*)$ is satisfiable iff so is $\varphi[\Diamond \leftarrow \Diamond^{-1}] \wedge \Box^{-(\text{md}(\varphi)+1)} \perp$ in $\text{MSL}(*, \Diamond^{-1})$.

Proof. (\Rightarrow): Suppose φ satisfiable, and let (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, be a Kripke-style finite function satisfying it. By Proposition 9.3, The pointed forest $((\mathcal{W}, R|_{\mathcal{W}}^{\leq \text{md}(\varphi)}, \mathcal{V}), w)$ satisfies φ . Notice that, by definition of $R|_{\mathcal{W}}^{\leq \text{md}(\varphi)}$, this pointed forest satisfies $\Box^{\text{md}(\varphi)+1} \perp$. We consider the acyclic Kripke-style finite function $\mathcal{K}' = (\mathcal{W}, (R|_{\mathcal{W}}^{\leq n})^{-1}, \mathcal{V})$. By definition of $R|_{\mathcal{W}}^{\leq n}$, it holds that $(\mathcal{K}', w) \models \Box^{-(\text{md}(\varphi)+1)} \perp$. By Lemma 9.44, $(\mathcal{K}', w) \models \varphi[\Diamond \leftarrow \Diamond^{-1}]$.

(\Leftarrow): Suppose $\varphi[\Diamond \leftarrow \Diamond^{-1}] \wedge \Box^{-(\text{md}(\varphi)+1)} \perp$ satisfiable, and let (\mathcal{K}, w) , where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ is a Kripke-style finite function, be one of its models. One can easily show, with a standard structural induction, that the satisfiability of a formula in $\text{MSL}(*, \Diamond^{-1})$ only depends on the set of worlds reachable backward from the current world, i.e. worlds in $(R^{-1})^*(w)$. Indeed, the logic only features the operator $*$, which does not change the current world, and the modality \Diamond^{-1} , which can only ever consider elements in $(R^{-1})^*(w)$. Therefore, without loss of generality, we can assume that R is such that, for every $(w_1, w_2) \in R$, $w_2 \in (R^{-1})^*(w)$. As $(\mathcal{K}, w) \models \Box^{-(\text{md}(\varphi)+1)} \perp$, this assumption leads to R being acyclic, so that $\mathcal{K}' = (\mathcal{W}, R^{-1}, \mathcal{V})$ is a Kripke-style finite forest. By Lemma 9.44, $(\mathcal{K}', w) \models \varphi$. \square

List of Notations and Symbols

Auxiliary Logic on Trees:

- (\mathcal{F}, t, n) : pointed forest, 78
- $<_{rk}$: rank order, 89
- \blacksquare : sabotage box operator, 79
- \blacklozenge : sabotage operator, 78
- \blacklozenge^k : k-sabotages operator, 79
- \blacklozenge^* : repeated sabotage operator, 78
- \mathbb{C}, N_j : character nodes, 82
- M : main nodes, 82
- \mathcal{F} : finite forest, 78
- $\Gamma_{rk}(\mathcal{F}, t, n)$: characteristic formula, 91
- n : node, 78
- $rk(m, s, k)$: formula rank, 89
- t : target node, 78
- $1st_S$, 86
- Hit : hit predicate, 78
- $Miss$: miss predicate, 78
- $\#\text{child} \geq \beta$, 85
- $\#\text{desc} \geq \beta$, 85
- inDom : domain membership, 79
- $\text{marks}_\Sigma \geq \beta$, 101
- mark_Σ , 101
- $\text{size}(Miss) \geq \beta$, 83
- $\text{size}(\varphi) \geq \beta$, 83
- word_Σ , 86
- \sim_{rk} : EF-games relation, 89
- $\tau_\beta(\varphi)$: translation from PITL, 102
- $\mathcal{F}[Miss]_t$: miss nodes, 79
- $\#\text{markAnc}_\Sigma \geq \beta$, 102

Classical logic:

- \Leftrightarrow : double implication, 21
- \Rightarrow : implication, 21
- \wedge : conjunction, 17
- $\wedge_{e \in S} \psi(e)$, 20
- \neg : negation, 17

\vee : disjunction, 21

$\text{bv}(\cdot)$: bound variables, 17

$\text{fv}(\cdot)$: free variables, 17

$|\varphi|$: size of a formula, 17

\models : satisfaction relation, 19

$\models \varphi$: validity, 24

\perp : false, 21

\top : true, 17

$\varphi, \psi, \chi, \dots$: formulae, 17

$\varphi \equiv \psi$: equivalence, 24

$\varphi \models \psi$: entailment, 24

Complexity Classes:

AEXP_{Pol} : alternating exponential time with polynomially many alternations, 379

NPSPACE : non-det. PSPACE, 225

PSPACE : polynomial space, 29

RE : recursively enumerable, 28

TOWER, F_3 : tower time/space, 29

Core Formulae:

$[S]^b$: flat set of locations, 151

$|\varphi|_m$: memory size, 222

$\approx_{x,\alpha}$: indistinguishability relation, 123

$\llbracket \cdot \rrbracket_{s,h}^X$: terms evaluation, 127

$\leftrightarrow_{x,\alpha}$: hop relation, 125

\mathfrak{P} : \rightarrow -simulation bound, 221

$LIT_{x,\alpha}(s, h)$: core formulae literals, 223

$Cycl[s]_{s,h}^X(\beta)$: bounded cycles, 151

$EV[S]^X$: end-point variables, 146

$Lab[S]_{s,h}^X$: labelled locations, 147

$Lab[W]_{s,h}^X$: labelled locations, 128

$MV[S]^X$: meet-point variables, 147

$NV[W]^X$: next-point variables, 127

$Obs[W](X)$: observed set of w , 129

$Path[S]_{s,h}^X(\ell)$: path set, 151

$\text{Pred}[\mathcal{S}]_{s,h}^X(\ell)$: predecessors set, 151
 $\text{Pred}[\mathcal{W}]_{s,h}^X(x)$: set of predecessors, 128
 $\text{Rem}[\mathcal{S}]_{s,h}^{X,\alpha}$: remainder set, 151
 $\text{Rem}[\mathcal{W}]_{s,h}^X$: remainder set, 128
 $\text{Self}[\mathcal{W}]_{s,h}^X$: set of self-loops, 128
 $\text{Sk}[\mathcal{W}](X, \alpha)$: skeleton set of w , 129
 $T[\mathcal{S}]^X$: terms of the s fragment, 147
 $T[\mathcal{W}]^X$: terms of the weak fragment, 127
 $\text{alldiff}(T)$, 157
 $\text{alloc}_{\mathcal{S}}(t)$, 156
 $e(x)$: end-point variable, 146
 $\text{loop}_{\mathcal{X}}^S(\beta) \geq \beta$, 152
 $m(x, y)$: meet-point variable, 147
 $\text{next}_X(t)$, 157
 $n(x)$: next-point variable, 127
 $\text{pred}_{\mathcal{X}}^{\mathcal{W}}(x) \geq \beta$, 129
 $\text{pred}_{\mathcal{X}}^S(x) \geq \beta$, 152
 $\text{rem}_{\mathcal{X}}^{\mathcal{W}} \geq \beta$, 129
 $\text{rem}_{\mathcal{X},\alpha}^S \geq \beta$, 152
 $\text{sby}_{s,h}^X(\ell)$: seen by location, 150
 $\text{seeslen}_X(t) \geq \beta$, 156
 $\text{sees}_X(t, t') = \beta$, 156
 $\text{sees}_X(t_1, t_2) \geq \beta'$, 152
 $\text{self}_{\mathcal{X}}^{\mathcal{W}} \geq \beta$, 129
 $t =_S n(x)$, 156
 $\text{unlab}(t)$, 157
 $u \in \text{loop}_{\mathcal{X}}^S(\beta)$, 155
 $u \in \text{pred}_{\mathcal{X}}^{\mathcal{W}}(x)$, 130
 $u \in \text{pred}_{\mathcal{X}}^S(x)$, 155
 $u \in \text{rem}_{\mathcal{X}}^{\mathcal{W}}$, 130
 $u \in \text{rem}_{\mathcal{X},\alpha}^S$, 155
 $u \in \text{sees}_X(t_1, t_2) \geq (\beta_1, \beta_2)$, 155
 $u \in \text{self}_{\mathcal{X}}^{\mathcal{W}}$, 130
 $\text{var.sby}(t)$, 157
 $\text{var}_X(t)$, 157
 $\text{Core}[\mathcal{W}](X, \alpha)$: core formulae of w , 129
 $t \hookrightarrow_S x$, 156

Domains:

\mathbb{N} : natural numbers, 12
 \mathcal{N} : (graph) nodes, 78
 \mathcal{W} : worlds, 106
 LOC : set of locations, 17
 VAR : set of variables, 17
 AP : propositional symbols, 106

Logics:

$\text{PL}(\sim)$, 390
 $\text{ML}(\ast)$, 407
 $\text{ML}(\|)$, 342
 $\text{SL}([\exists]_1, \ast, x \hookrightarrow _, \hookrightarrow^\dagger)$, 80
 $\text{SL}([\exists]_2, \ast, \neg\ast)$, 28
 $\text{SL}(\exists, \ast)$, 32
 $\text{SL}(\exists, \ast, \neg\ast)$, 17
 $\text{SL}(\ast, \neg\ast, \hookrightarrow^2, \hookrightarrow^3)$, 65
 QCTL^t , 105
 WMSO^f , 32
 WMSO , 32
 WSO , 30
 s : strong fragment of
 $\text{SL}([\exists]_1, \ast, [\neg\ast, \hookrightarrow^\dagger]_{\mathcal{S}})$, 119
 w : weak fragment of
 $\text{SL}([\exists]_1, \ast, [\neg\ast, \hookrightarrow^\dagger]_{\mathcal{S}})$, 119
 $\text{SL}([\exists]_1, \ast, [\neg\ast, \hookrightarrow^\dagger]_{\mathcal{W}})$, 119
 BBI , 37
 ALT , 77
 CTL , 105
 MSL , 34
 PITL , 97
 QCTL , 105

Modal Logic with $\|$:

2^Φ : conjunctions of formulae in Φ , 351
 $\|$: composition, 344
 $\text{lvl}(d, \varphi)$: formulae at level d , 386
 $\text{Core}(\Phi, k, P)$: core formulae, 350, 384
 $\text{bd}(d, \varphi)$: branching degree, 381
 $\text{cd}(\varphi)$: composition degree, 384
 $\max_{\text{bd}}(\varphi)$: max. branching degree, 381
 $\text{top}_{\text{AP}}(\varphi)$: top level atomic
 propositions, 350
 $\text{top}_{\text{gm}}(\varphi)$: top level graded
 subformulae, 350

Modal and Temporal logics:

$A(\varphi U \varphi)$: all-until, 106
 $E(\varphi M \psi)$: exists-strong-release, 107
 $E(\varphi U \varphi)$: exists-until, 106
 \Box : modality of necessity, 344
 $[U]$: everywhere modality, 79
 \Diamond : modality of possibility, 35
 \Diamond^{-1} : converse of \Diamond , 35
 $\Diamond_{\geq k}$: graded modality, 345
 $\langle \neq \rangle$: elsewhere modality, 35
 $\langle U \rangle$: somewhere modality, 78

\boxplus : bounded all-generally, 435
 $\exists p$: exist. propositional quantifier, 106
 AF : all-finally, 107
 AG : all-generally, 107
 EF : exists-finally, 107
 EG : exists-generally, 107
 EX : exists-next, 106
 $\text{gm}(d, \varphi)$: modalities at depth d , 381
 w : a world, 106
 $\text{nom}(p)$, 36
 $\text{uniq}(\varphi)$, 108
 $\leftrightarrows_{m,k}^P$: g -bisimulation, 412
 p, q, \dots : propositional symbols, 106
 $\text{gr}(\varphi)$: graded rank, 411
 $\text{md}(\varphi)$: modal depth, 381

Prop. Interval Temporal Logic:

$\Delta(w)$: word decomposition, 98
 Σ_\bullet : extended alphabet, 98
 1 : single predicate, 97
 a : head predicate, 97
 $\bar{\Sigma}$: marked alphabet, 98
 \bar{a} : marked symbol, 98
 $|$: composition operator, 97

Propositional Team Logic:

$\dot{\neg}p$: universally false symbol, 390
 $\dot{\vee}$: team disjunction, 390

Separation Logic:

(G, s, h) : generalised memory state, 52
 (s, h) : memory state, 18
 ℓ : location, 18
 $\exists S$: 2nd-order existential quantifier, 30
 $\exists z$: first-order existential quantifier, 17
 $\forall z$: first-order universal quantifier, 21
 \dashv : separating implication, 17
 \vdash : separating conjunction, 17
 $\dashv[n]$: bounded magic wand, 104
 $\dashv\ast$: subtraction, 20
 X, Y, Z, \dots : sets of variables, 25
 $\text{alloc}_y^{-1}(x)$, 65
 emp : empty predicate, 17
 $\text{ls}(x, y)$: list-segment predicate, 24
 $\text{next}(x = y)$, 66
 $\text{next}_z(x \rightarrow y)$, 67
 $n(x) = n(y)$: next-equality, 55
 $\text{strict}(\varphi)$, 24

x, y, z, \dots : variables, 17
 $x = y$: equality predicate, 17
 $x \neq z$: variables inequality, 20
 $\overline{(\cdot)}$: involution, 57, 68
 \bar{Y} : set of copies of variables, 57
 \triangleright_Y^X : encoded-by relation, 54
 \simeq_X : X -heap-isomorphic relation, 25
 \simeq_X^g : g - X -heap-isomorphic relation, 53
 $\text{maxval}_Y(s, h)$: maximum value, 142
 h : heap, 18
 h^* : Kleene closure of a heap, 23
 h^+ : Kleene plus of a heap, 22
 $h_1 \perp h_2$: disjoint heaps, 18
 $h_1 + h_2$: heap-union, 18
 s : store, 18
 $*_{e \in S} \psi(e)$, 26
 $n(x) \hookrightarrow n(y)$: next-points-to, 55
 $\text{size} = \beta$, 21
 $\text{size} \geq \beta$, 21
 $x \hookrightarrow^\delta y$: bounded reachability, 65
 $x \hookrightarrow^* y$: reach-star, 23
 $x \hookrightarrow^+ y$: reach-plus, 22
 $x \hookrightarrow_-$: alloc predicate, 21
 $x \hookrightarrow y$: points-to predicate, 17
 $x \mapsto y$: strict points-to predicate, 24
 $_ \hookrightarrow x$: alloc-back preciate, 22
 $s[z \leftarrow \ell]$: store update, 20

Sets, Relations, Functions:

$(\cdot)^*$: Kleene closure, 12
 $(\cdot)^+$: Kleene plus, 12
 $(\cdot)^\delta$: δ th composition, 12
 $(\cdot)^{-1}$: converse operator, 12
 $2^{(\cdot)}$: powerset, 12
 $=$: equality, 12
 $[i, j]$: natural numbers from i to j , 12
 \cap : set intersection, 12
 \cup : set union, 12
 \in : set membership, 12
 \leq : less than or equal to, 12
 $\rightharpoonup_{\text{fin}}$: partial map with finite domain, 18
 f : map/function, 25
 w : a word, 82
 Σ : an alphabet, 82
 $\text{card}(\cdot)$: set cardinality, 12
 dom : domain of a function, 12

id_S : identity map on the set S , 12
 ran : range of a function, 12
 a, b, \dots : characters/symbols, 82
 \neq : inequality, 12
 $\pi_1(\cdot)$: 1st projection map, 12
 $\pi_2(\cdot)$: 2nd projection map, 12
 \setminus : set difference, 12

\subseteq : set inclusion, 12
 \subseteq_{fin} : finite subset, 54
 \subsetneq : set strict inclusion, 12
 \times : Cartesian product, 12
 \rightarrow : arrow for functions, 18
 \emptyset : empty set, 12
 $i \dot{-} j$: subtraction on $i, j \in \mathbb{N}$, 12

List of Definitions

1.1	Definition (δ th composition)	12
1.2	Definition (Kleene plus and Kleene closure)	12
1.3	Definition (Domain, image of a subset, range)	12
1.4	Definition (Positions, subformulae and substitutions)	13
2.1	Definition (Memory state)	18
2.2	Definition (Path)	18
2.3	Definition (Disjoint heaps and their union)	18
2.4	Definition (Subheap)	18
2.5	Definition (Weakly connected component)	19
2.9	Definition (X-heap-isomorphism)	25
2.14	Definition (WSO structure)	30
2.17	Definition (Kripke-style finite function)	35
2.18	Definition (Disjoint finite functions and their union)	35
2.21	Definition (Non-deterministic monoid, [75])	37
3.1	Definition (Generalised memory state)	52
3.2	Definition (g-X-heap-isomorphism)	53
3.4	Definition (Encoded-by relation)	54
3.21	Definition (MSL - Memory state encoding.)	71
4.1	Definition (Forest)	78
4.2	Definition (Ancestors and Parents)	78
4.4	Definition (Word encoding)	82
4.12	Definition (Rank)	89
4.13	Definition (Rank order)	89
4.21	Definition (Markings)	98
4.22	Definition (Marked word decomposition)	98
4.30	Definition (Forests as heaps)	104
4.34	Definition (Kripke structure)	106
4.35	Definition (Path)	106
4.36	Definition (Kripke tree)	107
4.38	Definition (QCTL ^t - Pointed forests encoding)	108
4.42	Definition (MSL/ _{MLH} - Pointed forest encoding)	113
5.3	Definition (Small-heap property)	117
5.1	Definition (Acyclicity and garbage-freedom)	119

5.3 Definition (Small-heap property)	122
5.10 Definition (Next-point variables and terms)	127
5.11 Definition (Labelled locations)	127
5.12 Definition (Predecessors, self-loops and the remainder)	128
5.16 Definition (w -indistinguishable memory states)	131
5.17 Definition (w -hop relation)	131
5.22 Definition (Maximum value)	142
5.17 Definition (w -hop relation)	145
5.27 Definition (Seen by)	150
5.29 Definition (Predecessors, paths, cycles and the remainder)	151
5.30 Definition (Sets describing paths.)	151
5.34 Definition (s -indistinguishable memory states)	155
5.38 Definition (s -hop relation)	170
5.44 Definition (Memory size)	222
6.1 Definition (Core formulae of $\text{SL}(*, -*)$)	284
6.7 Definition (Core types of $\text{SL}(*, -*)$)	292
7.1 Definition (Kripke-style finite forest)	342
7.2 Definition ($+_w$: the Ambient-like union)	343
7.7 Definition (Core formulae)	349
7.9 Definition (Disjointness)	350
8.5 Definition (Good shape)	381
8.6 Definition	381
7.7 Definition (Core formulae)	384
8.13 Definition (Team encoding)	392
8.18 Definition (Information Trees encoding)	399
8.21 Definition (Pointed forests as information trees)	400
7.1 Definition (Kripke-style finite forest)	407
9.1 Definition ($+$: the Separation-like union)	408
9.5 Definition (g-bisimulation, [51])	411
9.6 Definition (Indistinguishable forests)	412
4.36 Definition (Kripke tree)	459
9.41 Definition (Trees as trees)	460
2.17 Definition (Kripke-style finite function)	462
E.1 Definition $((w_1, w_2)\text{-isomorphic forests})$	561

List of Figures

1.1	The separating conjunction $*$	6
2.1	A memory state	18
2.2	Two disjoint memory states	18
2.3	Satisfaction relation for $\text{SL}(\exists, *, \neg*)$, with respect to a memory state (s, h)	19
2.6	The decision problems of model-checking, satisfiability, validity and entailment	25
2.7	The complexity of Separation Logics	29
2.8	Satisfaction relation for WSO, with respect to a structure (\mathcal{D}, r)	30
2.9	Translating $\text{SL}(\exists, *, \neg*)$ to WSO. H, H_1 and H_2 are syntactically different	31
2.10	Satisfaction relation for WMSO^\dagger , with respect to a structure (\mathcal{D}, r)	32
2.11	Satisfaction relation for MSL, with respect to (\mathcal{K}, w) where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$	36
2.12	The complexity of MSL	37
2.13	Satisfaction relation for BBI	38
2.14	Hilbert-style axiomatisation of BBI [75]	39
2.15	The decision problems for the properties of acyclicity and garbage freedom	44
3.6	Satisfaction relation for $\text{SL}(\exists, \neg*)$, for a generalised memory state	57
3.7	Inductive cases of the translation to $\text{SL}(n(x), *, \neg*)$	59
3.8	Translation from $\text{SL}(*, \neg*, \leftrightarrow^2, \leftrightarrow^3)$ to a fragment of MSL	72
4.1	Subforest relation	79
4.2	Satisfaction relation for ALT, with respect to a pointed forest state (\mathcal{F}, t, n)	79
4.3	A forest encoding the word 1121	83
4.4	Ehrenfeucht-Fraïssé games for ALT	89
4.5	Satisfaction relation for PITL, under locality principle	98
4.6	Satisfaction relation for PITL on marked words	99
4.7	Example of the satisfaction of a formula on marked words	99
4.8	Translation from PITL to ALT	102
4.9	Translation from ALT to $\text{SL}(*, \neg*, \mathbf{1s})$ with bounded magic wand	105
4.10	Satisfaction relation for QCTL	107
4.11	A pointed forest (left) and one of its encoding as a Kripke tree (right)	109
4.12	Translation from ALT to QCTL	109
4.13	Translation from ALT to MSL/MLH	113
5.1	The decision problems for the properties of acyclicity and garbage freedom	120
5.2	Formulae from Section 2.1.1, in $\text{SL}([\exists]_1, *, [\neg*, \leftrightarrow^{\dagger}]_{\mathcal{W}}^{\mathcal{S}})$	121
5.3	Ehrenfeucht-Fraïssé games for $\text{SL}(*)$	125

5.4	A memory state. The partition of the heap is highlighted.	128
5.5	Semantics of the formulae in $\text{Sk}[\mathcal{W}](\mathbf{x}, \alpha)$, with respect to a memory state (s, h) .	130
5.6	Semantics of the formulae in $\text{Obs}[\mathcal{W}](\mathbf{x})$, with respect to a memory state (s, h) .	130
5.7	A memory state. Labelled locations are highlighted.	148
5.8	A memory state (s, h) . The partition of the heap is highlighted ($\alpha = 3$).	152
5.9	Semantics of the formulae in $\text{Sk}[s](\mathbf{x}, \alpha)$, with respect to a memory state (s, h) .	152
5.10	The two heaps h and h' described in Example 5.32.	153
5.11	Semantics of the formulae in $\text{Obs}[s](\mathbf{x}, \alpha)$, with respect to a memory state (s, h) .	155
5.12	Case analysis for the formula $\tau(\text{rem}_{\mathbf{x}}^{\mathcal{W}} \geq \beta)$, assuming $\text{Path}[s]_{s,h}^{\mathbf{x}}(\ell) \cap \text{Rem}[\mathcal{W}]_{s,h}^{\mathbf{x}} \neq \emptyset$.	164
5.13	The formula $\tau(\text{rem}_{\mathbf{x}}^{\mathcal{W}} \geq \beta)$.	165
5.14	Strategy used to define T'_1 , T'_2 and S' .	177
5.15	Strategy to define P'_1 , P'_2 , R'_1 and R'_2 .	186
5.16	Second case of the construction; paths of h .	189
5.17	Possible relations between ρ and ρ' .	204
5.18	Construction for the $\rightarrow*$ -simulation. The heap h'_1 is an answer for h_1 .	222
5.19	A NPSPACE algorithm for the satisfiability problem of $\text{SL}([\exists]_1, *, [\rightarrow, \leftrightarrow]^S_{\mathcal{W}})$.	226
5.20	Properties of the prime sets.	233
5.21	Subheaps of h'_1 .	236
5.22	Splitting ρ depending on its labelled locations.	244
5.23	Connecting next-point variables.	251
5.24	Recap: The complexity of Separation Logics.	271
6.1	Auxiliary formulae of $\text{SL}(*, \neg*)$. Note: $\mathbf{x} \in \text{VAR}$ and $\beta \in \mathbb{N}$.	284
6.2	The Hilbert-style proof system $\mathcal{H}_C(*, \neg*)$.	286
6.3	A proof of $\text{emp} \Rightarrow ((\mathbf{x} \leftrightarrow \perp \wedge \text{size} = 1) \rightarrow* \neg \text{size} \geq 2)$.	287
6.4	A proof of $\text{emp} \Rightarrow (\mathbf{x} \leftrightarrow \perp \wedge \text{size} = 1 \rightarrow* \text{size} = 1)$.	289
6.5	Proof system \mathcal{H}_C for Boolean combinations of core formulae.	291
6.6	The Hilbert-style proof system $\mathcal{H}_C(*)$.	296
6.7	The formula $\langle * \rangle(\varphi, \psi)$.	301
6.8	The Hilbert-style proof system $\mathcal{H}_C(*, \neg*)$ (again).	316
6.9	The formula $\langle \neg* \rangle(\varphi, \psi)$.	320
6.10	The formula $\langle \neg* \rangle(\varphi', \psi)$.	330
7.1	Three information trees T_3, T_1, T_2 (from left to right), such that $T_3 \equiv T_1 \parallel T_2$.	342
7.2	Three finite forests $\mathcal{K}_3, \mathcal{K}_1, \mathcal{K}_2$ (from left to right), such that $\mathcal{K}_3 = \mathcal{K}_1 +_{\text{w}} \mathcal{K}_2$.	343
7.3	Satisfaction relation for $\text{ML}(\mathbf{I})$, with respect to $(\mathcal{K}, \mathbf{w})$ where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$.	344
7.4	The Hilbert-style proof system $\mathcal{H}_{\text{GML}}(\mathbf{I})$.	347
7.5	The Hilbert-style proof system \mathcal{H}_{GML} .	349
7.6	The Hilbert-style proof system $\mathcal{H}_{\text{GML}}(\mathbf{I})$ (again).	352
7.7	the GML formula $\langle \mathbf{I} \rangle(\varphi, \psi)$, where φ in $\text{Core}(\Phi, k_1, P_1)$ and ψ in $\text{Core}(\Phi, k_2, P_2)$.	354
8.1	Satisfaction relation for QK .	378
8.2	Translation from $\text{ML}(\mathbf{I})$ to QK .	379
8.3	Satisfaction relation for $\text{PL}(\sim)$.	391
8.4	Translation from $\text{PL}(\sim)$ to $\text{ML}(\mathbf{I})$.	394
8.5	Interpretation and semantics of $\text{SAL}(\mathbf{I})$.	398

8.6	An information tree (on the left), and one of its possible encodings as a pointed forest $(\mathcal{K}, \mathbf{w})$ (on the right), with $n \geq 2$	399
8.7	A pointed forest $(\mathcal{K}, \mathbf{w})$ (on the left), and one of its possible encodings as an information tree (on the right), for $n \leq 3$. The portion of the information tree encoding atomic propositions is highlighted.	401
9.1	Satisfaction relation for $\text{ML}(\ast)$, with respect to $(\mathcal{K}, \mathbf{w})$ where $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$	408
9.2	A pointed forest $(\mathcal{K}, \mathbf{w})$ (on the left), and a possible decomposition via the union $+$	409
9.3	$R(\mathbf{w}) _{\top}$ (on the left), and $R'(\mathbf{w}') _{\top}$ (on the right).	416
9.4	Ehrenfeucht-Fraïssé games for $\text{ML}(\ast)$	421
9.5	Pointed forests used to prove Lemma 9.12. Only the leftmost one satisfies $\Diamond_{=2}\Diamond_{=1}\top$	422
9.6	Schema of a pointed forest satisfying (I)–(III).	423
9.7	The pointed forests described in (B).	429
9.8	An instance of (Wang) tiling problem, and a possible solution for a 3×3 grid.	433
9.9	Schema of a pointed forest $(\mathcal{K}, \mathbf{w})$ satisfying $\text{type}(j)$, for $j \geq 3$	435
9.10	Schema of a pointed forest $(\mathcal{K}, \mathbf{w})$ satisfying $\text{fork}_j^i(\mathbf{ax}, \mathbf{bx})$	438
9.11	Summary of the formulae to be defined in the next sections.	439
9.12	$\mathbf{n}(\mathbf{w}_{\mathbf{ax}}) < \mathbf{n}(\mathbf{w}_{\mathbf{bx}})$	440
9.13	$\mathbf{n}(\mathbf{w}_{\mathbf{ax}}) + 1 = \mathbf{n}(\mathbf{w}_{\mathbf{bx}})$	440
9.14	Shape of a pointed forest satisfying $\text{lsr}(k)$	445
9.15	Translation from $\text{ML}(\ast)$ to QK	461
9.16	Satisfaction relation for $\text{MSL}(\ast, \Diamond^{-1})$	462
C.1	Strategy defining h''	542

Index

abduction, 33
alphabet
 finite, 82
 marked, 98
 symbol, 82
ambient, 341
 calculus, 341
ambient logic, 341
 static, 341
antiframe, 34
Auxiliary logic on trees, 77

biabduction, 34
branching degree, 381
 maximal, 381
Bunched logics, 37

characteristic formula, 91
compactness, 88
Computation tree logic, 105
 quantified, 105
connectives
 Boolean, 17
 multiplicative, 17
core formulae, 122
 observed set, 129
 skeleton set, 129
core types, 292
cycle, 151
 self-loop, 128
 unbounded, 151
 unlabelled, 151

disjointness (GML), 350
disjunction, 21

Ehrenfeucht-Fraïssé games, 88

completeness, 91
duplicator, 88
game hopping, 126
move, 89
soundness, 91
spoiler, 88
state, 88
winning strategy, 89
encoding
 $\text{SL}(*, \neg, \hookrightarrow^2, \hookrightarrow^3)$ to MSL, 71
 ALT to MSL/MLH, 113
 ALT to $\text{SL}(*, \neg, \text{ls})$, 104
 ALT to QCTL t , 108
 symbol to ALT node, 82
 word to ALT forest, 82
entailment, 24

false, 21
forest
 edge, 78
 finite, 78
 pointed, 78
formula
 atomic, 17
 closed, 17
 equivalent, 24
 positions, 13
 substitution, 13
 tautology, 24
 valid, 24
 well-quantified, 56
frame, 34
function, 12
 δ th composition, 12
 domain, 12
 labelling, 106

- range, 12
- g-bisimulation, 411
- Gaifman, 124
 - locality Theorem, 124
- generalised memory state, 52
 - encoded-by relation, 54
 - g-X-heap-isomorphism, 53
 - well-formed, 58
- graded modal logic, 345
- graded rank, 411
- heap, 18
 - arrow, 18
 - disjoint, 18
 - union, 18
- implication, 21
 - double, 21
- involution, 57
- Kleene
 - closure, 12
 - plus, 12
- Kripke
 - w-subforest, 343
 - finite forest, 342, 408
 - finite function, 35, 462
 - pointed finite function, 35
 - pointed structure, 106
 - pointed tree, 107
 - structure, 106
 - tree, 107, 459
- literal, 284
- local nominals, 437
- location, 17
 - labelled, 127
 - maximum value, 142
 - unlabelled, 128
- magic wand, 17
 - bounded, 104
- main path, 82
- memory cell, 18
- memory state, 18
 - X-garbage-free, 119
 - X-heap-isomorphism, 25
- acyclic, 119
- disjoint, 18
- Modal logic of heaps, 35
- Modal separation logic, 34
- modality
 - all-finally, 107
 - all-generally, 107
 - all-until, 106
 - converse of possibility, 35
 - elsewhere, 35
 - everywhere, 79
 - exists-finally, 107
 - exists-generally, 107
 - exists-next, 106
 - exists-strong-release, 107
 - exists-until, 106
 - graded, 345
 - necessity, 344
 - possibility, 35
 - repeated sabotage, 78
 - sabotage, 78
 - somewhere, 78
 - temporal, 106
- model, 19
- model-checking, 24
- node, 78
 - ancestor, 78
 - character, 82
 - child, 78
 - current evaluation, 78
 - descendant, 78
 - hit, 78
 - main, 82
 - miss, 78
 - parent, 78
 - target, 78
- nominal, 36
- operator
 - composition ($\text{ML}(\mathbb{I})$), 344
 - composition (PITL), 97
- path, 18
 - disjoint, 18
 - maximal, 106
- predecessors, 128

- predicate
 - alloc, 21
 - alloc-back, 22
 - bounded reachability, 65
 - empty, 17
 - equality, 17
 - head, 97
 - hit, 78
 - list-segment, 24
 - miss, 78
 - next-equality, 55
 - next-points-to, 55
 - points-to, 17
 - reach-plus, 22
 - reach-star, 23
 - reachability, 22
 - relational, 30
 - single, 97
 - size, 21
 - strict, 24
- program variable name, 17
- Propositional interval temporal logic, 97
 - locality principle, 97
- Propositional logic in team semantics, 390
- propositional symbol, 106
- quantifier
 - existential, 17
 - existential second-order, 30
 - propositional, 106
 - universal, 21
- rank, 89
 - modal, 89
 - repeated sabotage, 89
 - sabotage, 89
- rank order, 89
- recursively enumerable, 28
- relation, 12
 - accessibility, 106
 - hop, 125
- satisfaction, 19
- satisfiability, 24
- second-order logic
 - WSO structure, 30
 - assignment, 30
- domain, 30
- one unary function symbol, 32
- weak, 30
- weak monadic, 32
- separating
 - conjunction, 17
 - implication, 17
- Separation logic, 17
 - first-order, 17
 - symbolic-heap, 33
- separation logic
 - extensional, 28
 - intensional, 28
- sepraction, 20
 - bounded, 104
- set, 12
 - consistent, 19
- simulation, 124
 - \exists -simulation, 141
 - \neg -simulation, 221
 - $*$ -simulation, 124
- size
 - memory, 222
 - of a formula, 17
- small-heap property, 122
- star, 17
- store, 18
- strict points-to, 24
- subformula, 13
- subheap, 18
 - strict, 18
- team, 390
- term, 127
- tiling problem, 432
- true, 17
- universe, 342, 408
- validity, 24
- variable, 17
 - bound, 17
 - copy, 57
 - end-point, 146
 - free, 17
 - individual, 30
 - meet-point, 147

- next-point, 127
- relation, 30
- unique, 119
- weakly connected component, 19
- word
 - decomposition, 98
 - empty, 99
- finite, 82
- marked, 98
- world, 106
 - accessible, 106
 - current, 106
 - predecessor, 35
 - spy, 71, 487
 - successor, 35

Titre: Logiques de Séparation: Complexité, Expressivité, Systèmes de Preuve

Mots clés: Logique de séparation, complexité, expressivité, axiomatisation

Résumé: Cette thèse propose une étude approfondie de problèmes de décision classiques, tels que la satisfaisabilité et la validité pour des logiques de séparation, langages d'assertion bien connus développés pour la vérification de programmes avec structures dynamiques.

La première partie de la thèse s'intéresse à la notion d'accessibilité pour les logiques de séparation. Notre motivation est double: d'une part, il s'agit de comprendre les frontières de la décidabilité de fragments de la logique de séparation du premier ordre; d'autre part l'intention est de concevoir une logique de séparation aussi expressive que possible, et dont le problème de satisfaisabilité soit décidable avec une complexité relativement modeste.

Dans la seconde partie de la thèse, nous tirons profit des techniques développées dans la

première partie pour définir une axiomatisation à la Hilbert de logiques de séparation et d'autres logiques spatiales. Nous définissons le premier calcul interne pour la logique de séparation sans quantification. En utilisant la même approche, nous définissons une axiomatisation pour une logique modale enrichie d'un opérateur de composition issu d'une logique des ambients. Ces systèmes de preuves mettent en lumière des relations intéressantes entre les deux logiques.

Dans la troisième partie de la thèse, nous approfondissons encore davantage les relations entre les logiques de séparation et les logiques des ambients. Des similarités et des différences sont établies en termes de pouvoir d'expression et de complexité algorithmique. Afin de mener à bien nos comparaisons, nous nous plaçons dans un cadre uniforme issu de la logique modale.

Title: Reasoning with Separation Logics: Complexity, Expressive Power, Proof Systems

Keywords: Separation logic, computational complexity, expressive power, axiomatisation

Abstract: This thesis proposes an in-depth study of classical decision problems, such as satisfiability and validity, for separation logics: well-known assertion languages developed to verify heap-manipulating programs.

The first part of the thesis focuses on the complexity of separation logics featuring reachability predicates. Our goal is to understand the decidability frontier for fragments of first-order separation logic, and devise a very expressive separation logic whose satisfiability problem is decidable with a relatively low complexity.

In the second part of the thesis, we use techniques developed in the first part to design

Hilbert-style axiomatisations for separation logics and other spatial logics. Two proof systems are introduced, one for quantified-free separation logic and the other for a modal logic enriched with the composition operator from ambient logic. These proof systems reveal interesting connections between the two logics.

In the third part of the thesis, we dig deep in the connections between separation logics and ambient logics. Surprising similarities and differences are found, both in terms of expressive power and computational complexity. To carry out our comparison, we devise a suitable framework based on modal logic.

