First-Order Model Checking on Generalisations of Pushdown Graphs

Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation von Dipl.- Math. Alexander Kartzow aus Gießen
Juli 2011 — Darmstadt — D 17

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Mathematik
Arbeitsgruppe Logik

First-Order Model Checking on Generalisations of Pushdown Graphs

Genehmigte Dissertation von Dipl.- Math. Alexander Kartzow aus Gießen

1. Gutachten: Prof. Dr. Martin Otto
2. Gutachten: Prof. Damian Niwiński
3. Gutachten: Prof. Dr. Stephan Kreutzer

Tag der Einreichung: 09.12.2010
Tag der Prüfung: 11.05.2011

Darmstadt — D 17

Acknowledgement

I am deeply grateful to my supervisor Martin Otto for his support. Beside his mathematical advice, I especially appreciated his lessons in mathematical writing and his efforts for improving my English. I thank my referees Damian Niwiński and Stephan Kreutzer for their valuable comments on this work. Furthermore, I thank Achim Blumensath and Dietrich Kuske for many helpful comments and the opportunity to discuss some of my ideas. I am grateful to Alex Kreuzer and my wife Franziska for spell checking parts of this thesis. Finally, I thank my wife and my family for the moral support and the DFG for the financial support during the last years.

German Summary / Zusammenfassung

In dieser Arbeit untersuchen wir das Model-Checking-Problem für Pushdown-Graphen. Ein Model-Checking-Algorithmus für eine Logik $\mathscr{L}$ und eine Klasse von Strukturen $\mathscr{C}$ ist ein Algorithmus, der bei Eingabe eines Paares $(\mathfrak{A}, \varphi)$ mit $\mathfrak{A} \in \mathscr{C}$ und $\varphi \in \mathscr{L}$ entscheidet, ob die Struktur $\mathfrak{A}$ die Formel φ erfüllt.

In dieser Arbeit konzentrieren wir uns größtenteils auf die Entwicklung von Model-Checking-Algorithmen für die Logik erster Stufe (im folgenden FO abgekürzt) und ihrer Erweiterung um Erreichbarkeitsprädikate auf Klassen verallgemeinerter Pushdown-Graphen.

Ein Pushdown-Graph ist der Konfigurationsgraph eines Kellerautomaten. Kellerautomaten, die auch Pushdown-Systeme genannt werden, sind endliche Automaten erweitert um die Speicherstruktur eines Stacks. Ein klassisches Resultat von Muller und Schupp [53] beweist die Entscheidbarkeit des Model-Checking-Problems für die monadische Logik zweiter Stufe (im folgenden MSO abgekürzt) auf der Klasse der Pushdown-Graphen. Insbesondere gibt es also auch einen Model-Checking-Algorithmus für die Logik erster Stufe auf der Klasse der Pushdown-Graphen.

In den letzten Jahren haben Verallgemeinerungen der Pushdown-Graphen großes Interesse im Bereich der automatischen Verifikation von funktionalen Programmiersprachen erlangt. Pushdown-Graphen wurden im wesentlichen auf zwei Arten erweitert.

Die erste Erweiterung führt zum Konzept eines Pushdown-Systems höherer Ordnung. Hierbei wird der Stack eines Kellerautomaten ersetzt durch eine Struktur ineinander geschachtelter Stacks. Die Verschachtelungstiefe dieser Stacks wird dabei als die Stufe des Systems bezeichnet. Ein Pushdown-System der Stufe 2 hat also einen Stack aus Stacks, ein System der Stufe 3 einen Stack aus Stacks aus Stacks und analog für jede Stufe $n \in \mathbb{N}$. Auf jeder Stufe $i \leq n$ dieser Schachtelung gibt es entsprechende Stack-Operationen um den obersten Eintrag des Stufe i Stacks zu manipulieren. Mit diesem Ansatz wurden zwei Hierarchien verallgemeinerter Pushdown-Graphen definiert. Die Hierarchie der "Higher-Order-Pushdown-Graphen" und die der "Collapsible-Pushdown-Graphen". Die beiden Klassen unterscheiden sich in den verwendeten Stack-Operationen. Die Pushdown-Systeme, die Collapsible-Pushdown-Graphen erzeugen, erweitern die Pushdown-Systeme, die Higher-Order-Pushdown-Graphen erzeugen, um eine neue Operation, die "Collapse" genannt wird. Trotz der ähnlichen Definition dieser beiden Hierarchien von Graphen haben die Hierarchien sehr unterschiedliche modelltheoretische Eigenschaften.

Die Hierarchie der Higher-Order-Pushdown-Graphen fällt mit der Caucal-Hierarchie zusammen. Diese Klasse von Graphen ist definiert durch iteriertes Anwenden von MSO-Interpretationen und Abwicklungen beginnend von der Klasse der endlichen Graphen. Da sowohl Abwicklungen als auch MSO-Interpretationen die Entscheidbarkeit von monadischer Logik zweiter Stufe erhalten, ist MSO-Model-Checking auf der Klasse der Higher-Order-Pushdown-Graphen entscheidbar.

Die Klasse der Collapsible-Pushdown-Graphen hat dagegen ganz andere modelltheoretische Eigenschaften. Schon auf der zweiten Stufe dieser Hierarchie gibt es Graphen mit unentscheidbarer MSO-Theorie. Hingegen ist der modale μ-Kalkül auf der Klasse der Collapsible-Pushdown-Graphen entscheidbar. Dieses unterschiedliche Verhalten in Bezug auf MSO und μ-Kalkül tritt nur bei wenigen natürlichen Strukturklassen auf.

Eine weitere Klasse mit dieser Eigenschaft erhalten wir durch die zweite Verallgemeinerung von Pushdown-Graphen. Abwicklungen von Pushdown-Graphen haben sich in der Software-Verifikation als nützliche Abstraktion von Programmabläufen herausgestellt. Hierbei wird auf dem Stack vor allem der Aufruf von Funktionen und die Rückkehr zum aufrufenden Programm verwaltet. Viele interessante Eigenschaften von Programmen lassen sich so durch MSO-Model-Checking auf der Abwicklung eines Pushdown-Graphen überprüfen und nachweisen. Allerdings ist es in diesem Modell nicht möglich, den Zustand des Programms vor einem Funktionsaufruf mit dem Zustand am Ende dieser Funktion zu vergleichen, denn in monadischer Logik zweiter Stufe kann man bei unbeschränkt verschachteltem Aufruf von Funktionen die zusammenhgehörenden Positionen von Funktionsaufruf und Funktionsende nicht definieren.

Um dieses Problem zu umgehen haben Alur et al. [2] die Klasse der "Nested-Pushdown-Trees" eingeführt (Warnung: wir bezeichnen diese bewusst nicht als "Nested-Pushdown-Bäume", weil es keine Bäume sind). Ein Nested-Pushdown-Tree ist die Abwicklung eines Pushdown-Graphen mit einer zusätzlichen Relation $\hookrightarrow$. Diese verbindet eine Push-Operation des Kellerautomaten mit der dazugehörigen Pop-Operation. Wenn man einen Pushdown-Graphen also als abstraktes Modell des Programmablaufs eines Computerprogramms sieht, wird der Funktionsaufruf über $\hookrightarrow$ mit dem Ende der aufgerufenen Funktion verbunden. Mit diesem Modell kann man also die oben erwähnten Nachteile der Pushdown-Graphen überwinden. Alur et al. konnten zeigen, dass für die Klasse der Nested-Pushdown-Trees das μ-Kalkül-Model-Checking entscheidbar ist. Jedoch gibt es einen Nested-Pushdown-Tree mit unentscheidbarer MSO-Theorie.

Da die monadische Logik zweiter Stufe für Collapsible-Pushdown-Graphen und für Nested-Pushdown-Trees unentscheidbar ist, stellt sich die natürliche Frage, welche Fragmente der monadischen Logik zweiter Stufe auf diesen Klassen entscheidbar sind.

In unserer Arbeit geben wir dafür die folgenden partiellen Antworten.

1. Auf der zweiten Stufe der Hierarchie der Collapsible-Pushdown-Graphen ist das FO-Model-Checking-Problem entscheidbar. Genauer ist die Erweiterung von FO um reguläre Erreichbarkeitsprädikate und Ramsey-Quantoren entscheidbar. Wir beweisen dies, indem wir eine baumautomatische Repräsentation (vgl. Punkt 4) für jeden Collapsible-Pushdown-Graphen der zweiten Stufe erzeugen.

2. Das FO-Model-Checking-Problem auf der Klasse der Nested-Pushdown-Trees ist in zweifach exponentiellem Platz entscheidbar. Zusätzlich kann jeder Nested-Pushdown-Tree durch eine FO-Interpretation aus einem Collapsible-Pushdown-Graphen der Stufe 2 erzeugt werden. Mithilfe dieser Interpretation können wir auch die Theorie der Logik erster Stufe erweitert um das Erreichbarkeitsprädikat für jeden Nested-Pushdown-Tree entscheiden.

Neben diesen Resultaten über bekannte Erweiterungen von Pushdown-Graphen beinhaltet diese Arbeit auch die folgenden Ergebnisse.

3. Durch die Kombination der Idee der geschachtelten Stacks mit der Definition der Nested-Pushdown-Trees definieren wir eine neue Hierarchie der Nested-Pushdown-Trees höherer Ordnung. Ein Nested-Pushdown-Tree der Stufe l ist die Abwicklung eines Pushdown-Graphen der Stufe l erweitert um eine neue Relation $\hookrightarrow$, die zusammengehörende Push- und Pop-Operationen verbindet. Wir beweisen, dass diese neue

Hierarchie eng verwandt mit den Hierarchien der Higher-Order-Pushdown-Graphen und der Collapsible-Pushdown-Graphen ist. Alle Abwicklungen von Higher-Order-Pushdown-Graphen sind in der neuen Hierarchie enthalten. Außerdem lassen sich alle Higher-Order-Nested-Pushdown-Trees durch FO-Interpretationen aus der Klasse der Collapsible-Pushdown-Graphen erzeugen. Durch diese Interpretation kann man Higher-Order-Nested-Pushdown-Trees der Stufe l als besonders einfache Collapsible-Pushdown-Graphen der Stufe $l + 1$ betrachten. Wir zeigen dann, dass für die zweite Stufe dieser neuen Hierarchie ein FO-Model-Checking-Algorithmus existiert.

4. Wir zeigen in dieser Arbeit auch, dass die Erweiterung der Logik erster Stufe um Ramsey-Quantoren auf baumautomatischen Strukturen entscheidbar ist. Baumautomatische Strukturen sind Strukturen, die sich durch endliche Baumautomaten repräsentieren lassen. Ein Ramsey-Quantor ist von der Gestalt $\mathsf{Ram}^n \bar{x}(\varphi(\bar{x}))$. Eine solche Formel wird von einer Struktur $\mathfrak{A}$ erfüllt, wenn es eine unendliche Teilmenge $M \subseteq \mathfrak{A}$ gibt, so dass jedes n-Tupel aus M, von dem je zwei Elemente paarweise verschieden sind, die Formel φ erfüllt. Unser Beweis, der in Zusammenarbeit mit Dietrich Kuske entstand, verallgemeinert ein analoges Resultat für die Klasse der wortautomatischen Strukturen.

Contents

1 Introduction

In this thesis, we investigate the first-order model checking problem for generalisations of pushdown graphs. Our work is a contribution to the classification of all graphs that have decidable first-order theories. The classes of graphs that we study are collapsible pushdown graphs and nested pushdown trees. These classes of graphs have the following interesting model-theoretic properties. The monadic second-order theory of a graph from these classes is not decidable in general, while its modal μ-calculus theory is always decidable. Most other classes of graphs do not share these properties. In most cases, a natural class of graphs will either have decidable monadic second-order and modal μ-calculus theories or undecidable monadic second-order and modal μ-calculus theories. We start by briefly recalling the history of generalisations of pushdown graphs. These classes of graphs arise naturally in the field of software verification for higher-order functional programmes.

1.1 Verification and Model Checking

Verification of hard- and software is concerned with the problem of proving that a certain piece of hard- or software fulfils the task for which it was designed. Since computer systems are more and more used in safety critical areas, failure of a system can have severe consequences. Thus, verification of these systems is very important. The most successful approach to verification is the model checking paradigm introduced by Clarke and Emerson [18]. In model checking, one derives an abstract structure $\mathfrak{A}$ as a model of some piece of hard- or software and one specifies the requirements of the system in a formula φ from some logic $\mathscr{L}$. The problem whether the system is correct then reduces to the problem whether the abstract model $\mathfrak{A}$ of the system satisfies the formula. This is called a model checking problem. If the model satisfies the formula, we write $\mathfrak{A} \models \varphi$. In this terminology, the $\mathscr{L}$ model checking problem on some class $\mathscr{C}$ of structures asks on input a structure $\mathfrak{A} \in \mathscr{C}$ and a formula $\varphi \in \mathscr{L}$ whether $\mathfrak{A} \models \varphi$. Since the 1980's, model checking on finite structures has been developed and is nowadays used for real-world hardware verification problems. For hardware, it is sufficient to consider finite structures. Each piece of hardware has a finite amount of storage capacity whence it can always be modelled as a finite state system. On the other hand, software verification requires the use of infinite models as abstractions because the storage capacity of the underlying hardware is a priori unbounded. Hence, software verification naturally leads to model checking problems on infinite structures. Of course, model checking on infinite structures is only possible for certain classes of structures. Since we expect an algorithm to process the structures involved as input, we need finite descriptions of these infinite structures. Hence, model checking on infinite structures is only interesting for classes of finitely representable structures. A further restriction is imposed by the question of decidability of the model checking problem. A very expressive logic on a large class of finitely represented structures will result in an undecidable model checking problem (the halting problem can be formulated as a special version of model checking on structures representing Turing machines). Thus, there is a tradeoff between the choice of the class $\mathscr{C}$ and the logic $\mathscr{L}$. It is important to identify those pairs $(\mathscr{C}, \mathscr{L})$ for which a model checking algorithm exists, i.e., for which the $\mathscr{L}$ model checking on $\mathscr{C}$ is decidable.

Various techniques have been developed to finitely represent infinite structures. According to Bárány et al. [4], these may be classified into the following approaches.

- Algebraic representations: a structure is described as the least solution of some recursive equation in some appropriate algebra of structures. An example of this class are vertex replacement equational graphs [20].
- Transformational or logical representations: the structure is described as the result of applying finitely many transformations to some finite structure. A transformation in this sense is, e.g., the tree-unfolding, the Muchnik-Iteration, or some logical interpretation (see [9] for a survey).
- Internal representations: an isomorphic copy of the structure is explicitly described using transducers or rewriting techniques. In most cases a set of words or trees is used as the universe of the structure. The relations are then represented by rewriting rules or by transducers that process tuples of elements from this set. Rewriting rules often appear in the disguise of transitions of some computational model. In this case the universe consists of configurations of some computational model. There is an edge from one configuration to another configuration if one step of the computation leads form the first to the second configuration.

There is no clear separation between the approaches because there are many classes of structures that may be represented using techniques from different approaches.

In this thesis we will only deal with structures that have internal representations. We investigate configuration graphs of different types of automata. The universe of such a graph consists of the set of configurations of an automaton and the relations are given by the transitions from one configuration to another. Automata that may be used for this approach are, e.g., Turing machines, finite automata, pushdown systems or collapsible pushdown systems. In this thesis we study configuration graphs of collapsible pushdown systems. We will introduce these systems later in detail. A pushdown system can be seen as a finite automaton equipped with a stack. A collapsible pushdown system uses a nested stack, i.e., a stack of stacks of stacks of ... of stacks instead of the ordinary stack. On each stack level, the collapsible pushdown system can manipulate the topmost entry of its stack.

Another concept that plays a major role within this thesis is the concept of a tree generated by some pushdown system. This tree is obtained by applying a graph unfolding to the configuration graph. This can be seen as a transformational representation of the graph that starts from the underlying configuration graph. On the other hand, it can also be seen as an internal representation: the nodes of a graph are represented by the set of runs of the given automaton and the relations of the structure are defined by rewriting rules that transform a run of length n into a run of length $n+1$ that extends the first run. If a graph is the configuration graph of some automaton, we will refer to the unfolding of this graph as the tree generated by this automaton. This notion becomes important when we discuss nested pushdown trees. These are trees generated by pushdown systems expanded by a so-called jump relation. We will present this concept at the end of the next section.

The second form of internal representation for infinite structures that we will use are tree-automatic structures. A structure is tree-automatic if it can be represented as a regular set of trees such that for each relation there is a finite tree-automaton that accepts those tuples of trees from the universe that form a tuple of the relation. We provide a more detailed

introduction to tree-automatic structures as well as some notes concerning the history of tree-automatic structures in Section 2.5. The class of tree-automatic structures is a nice class because first-order model checking is decidable on this class: there are automata constructions that correspond to negation, conjunction and existential quantification. Thus, for any tree-automatic structure and any first-order formula, one can construct a tree-automaton that accepts an input (representing a tuple of parameters from the structure) if and only if the structure satisfies the formula (where the free variables of φ are assigned to the parameters represented by the input).

1.2 Collapsible Pushdown Graphs and Nested Pushdown Trees

The history of software verification is closely connected to two important results on model checking. In 1969, Rabin [55] proved the decidability of monadic second-order logic (MSO) on the infinite binary tree. In terms of model checking, his result states that the MSO model checking is decidable for the class that only consists of one structure, namely, the full binary tree. Sixteen years later, Muller and Schupp [53] showed the decidability of the MSO model checking on pushdown graphs. This was a very important step towards automated software verification because pushdown graphs proved to be very suitable for modelling procedural programmes with calls of first-order recursive procedures. The function calls and returns are modelled using the stack. At a function call, the state of the programme is pushed onto the stack and at a return the old context is restored using a pop operation.

Collapsible pushdown systems can be seen as the result of the search for a similar result for higher-order functional programming languages. Already in the 1970's Maslov was the first to consider so-called higher-order pushdown systems as accepting devices for word languages. A higher-order pushdown system is a generalisation of a pushdown system where one replaces the stack by a nested stack of stacks of stacks of ... stacks. For each stack level the higher-order pushdown system can use a push and a pop operation. In the last years, these automata have become an important topic of interest because of two results.

1. Carayol and Wöhrle [16] showed that the class of graphs generated by ε-contractions of configuration graphs of higher-order pushdown systems coincide with the class of graphs in the Caucal hierarchy. Caucal [17] defined this class as follows. The initial level in the hierarchy contains all finite graphs. A graph in the next level is obtained by applying an unfolding and an MSO-interpretation to a graph in the previous level. Since both operations preserve the MSO decidability, MSO model checking on higher-order pushdown graphs is decidable. In fact, the Caucal hierarchy is one of the largest classes where the MSO model checking is known to be decidable.

2. Knapik et al. [41] studied higher-order pushdown systems as generators of trees. They proved that the class of trees generated by higher-order pushdown systems coincides with the class of trees generated by safe higher-order recursion schemes (safe higher-order functional programmes). Safety is a rather syntactic condition on the types of in- and outputs to functions that are used in a recursion scheme.

The second result initiated a lot of study on the question whether there is some computational model whose generated trees form exactly the class of trees generated by arbitrary higher-order recursion schemes and whether the trees generated by safe recursion schemes

form a proper subclass of the class of trees generated by arbitrary recursion schemes. For instance, Aehlig et al. [1] showed that safety is no restriction for string languages defined by level 2 recursion schemes. Hague et al. [27] introduced collapsible pushdown systems. The concept of a collapsible pushdown system is a stronger variant of the concept of a higher-order pushdown systems. They showed that these are as expressive as arbitrary higher-order recursion schemes, i.e., a tree is generated by a level n recursion scheme if and only if it is generated by some level n collapsible pushdown system. Furthermore, they showed the decidability of modal μ-calculus ($L\mu$) model checking on collapsible pushdown graphs. Recently, Kobayashi [43] designed an $L\mu$ model checker for higher-order recursion schemes and successfully applied this model checker to the verification of higher-order functional programmes. Even though the connection to higher-order recursion schemes turns collapsible pushdown systems into a very interesting class of structure for model checking purposes, there are few things known about the structure of the trees and graphs generated by these systems. For example, it is conjectured – but not proved – that the class of trees generated by collapsible pushdown systems properly extends the class of trees generated by higher-order pushdown systems. The same conjecture in terms of recursion schemes says that there is a tree generated by some unsafe higher-order recursion scheme that is not generated by any safe higher-order recursion scheme.

Concerning model checking, Hague et al. proved another interesting fact about the class of graphs generated by collapsible pushdown systems: they presented a collapsible pushdown system of level 2 that has undecidable MSO theory. In terms of model checking, this is a proof of the fact that MSO model checking on the class of collapsible pushdown graphs is undecidable.

From a theoretical point of view, this turns collapsible pushdown graphs into an interesting class of graphs. Besides the class of nested pushdown trees it is the only known natural class of graphs that has decidable $L\mu$ model checking but undecidable MSO model checking. Thus, a better understanding of this class of graphs may also give insight into the difference between $L\mu$ and MSO. In fact, this thesis tries to identify larger fragments of MSO that are still decidable on collapsible pushdown graphs. The most prominent fragment of MSO is, of course, first-order logic (FO). The author was the first to investigate first-order model checking on collapsible pushdown graphs. In STACS'10 [35], we proved that the model checking problem for the extension of FO by reachability predicates on the class of collapsible pushdown graphs of level 2 is decidable. In this thesis, we present a slightly extended version of this result: if one enriches the graphs by $L\mu$-definable predicates, model checking is still decidable. Furthermore, we may also enrich FO by Ramsey quantifiers. Very recently, Broadbent [12] matched our result with a tight upper bound: the first-order model checking on level 3 collapsible pushdown graphs is undecidable. Moreover, Broadbent presented a fixed formula $\varphi \in \mathsf{FO}$ such that the question whether φ is satisfied by some level 3 collapsible pushdown graph is undecidable. Furthermore, he provided an example of a level 3 collapsible pushdown graph with undecidable FO-theory.

We now turn to the history of nested pushdown trees. Alur et al. [2] introduced the concept of so-called jump edges in order to overcome the following weakness of model checking on pushdown graphs. Recall that pushdown systems are useful abstractions of programmes which call first-order recursive functions. Function calls and returns are handled by using push and pop operations. But interesting properties of some programme may include statements about the situation before a function call happens in comparison to the situation at

the end of this function, i.e., at the return of this function. Unfortunately, even strong logics like MSO cannot express such properties. They cannot "find" the exact corresponding pop operation for a given push operation in general. As soon as a potentially unbounded nesting of function calls may occur, MSO like many other logics cannot keep track of the number of nestings in the call and return structure. But this would be necessary for identifying the pop operation that corresponds to a given push.

Alur et al. wanted to make this correspondence of push and pop operations explicit. Thus, a nested pushdown tree is defined to be the unfolding of a pushdown graph enriched by jump edges that connect each push operation with the corresponding pop operation. Unfortunately, this expansion of trees generated by pushdown systems leads to undecidability of the MSO model checking [2]. Anyhow, Alur et al. were able to prove that $L\mu$ model checking is still decidable on nested pushdown trees. Thus, the class of nested pushdown trees is the second natural class of structures with undecidable MSO but decidable $L\mu$ model checking. We were able to provide an elementary FO model checking algorithm for nested pushdown trees. This result was first presented in MFCS'09 [34].

The similar behaviour of the class of nested pushdown trees and collapsible pushdown graphs with respect to model checking has an easy explanation. Nested pushdown trees are first-order interpretable in collapsible pushdown graphs of level 2. Furthermore, the interpretation is quite simple and uniform.

1.3 Goal and Outline of this Thesis

This thesis is concerned with various model checking problems. Our most important results provide model checking algorithms for first-order logic (and slight extensions) on various classes of structures. The main focus is on structures defined by higher-order (collapsible) pushdown systems. On the one hand, we study the hierarchy of collapsible pushdown graphs that was introduced by Hague et al. [27]. On the other hand, we study a new hierarchy of higher-order nested pushdown trees. This hierarchy is the class obtained by the straightforward generalisation of the concept of a nested pushdown tree to trees generated by higher-order pushdown systems. We consider the expansions of these trees by jump-edges that connect corresponding push and pop transitions (at the highest level of the underlying higher-order pushdown system). This new hierarchy forms a class of graphs that contains the class of trees generated by higher-order pushdown systems and that is contained (via uniform FO-interpretations) in the class of collapsible pushdown graphs. Thus, we hope that the study of this new hierarchy can reveal some insights into the differences between these two hierarchies.

In this thesis, we obtain the following results on the model checking problems for these hierarchies.

1. The second level of the collapsible pushdown hierarchy is tree-automatic and its FO(REACH) model checking is decidable.

2. First-order model checking on nested pushdown trees is in 2-EXPSPACE.

3. First-order model checking on level 2 nested pushdown trees is decidable.

In order to prove these claims, we develop various new techniques.

All of these proofs rely on a structural analysis of runs of higher-order collapsible pushdown systems. This analysis provides a characterisation of the reachability of one configuration from another.

The second ingredient for our first result is a clever encoding of configurations in trees which turns the set of reachable configurations into a regular set of trees.

The other two results use a new application of Ehrenfeucht-Fraïssé games to the model checking problem. We analyse strategies in the Ehrenfeucht-Fraïssé game that are subject to certain restrictions. The existence of such restricted winning strategies on a class of structures can be used to provide a model checking algorithm on this class. The basic idea is as follows: assume that Duplicator has a strategy that only requires to consider finitely many elements in a structure. Model checking on this structure can then be reduced to model checking on a finite substructure, namely, on the substructure induced by those elements that are relevant for Duplicator's strategy.

Using our analysis of runs of collapsible pushdown systems, we show that there are such restricted strategies on the first two levels of the nested pushdown hierarchy.

Motivated by the tree-automaticity of level $\mathbf{2}$ collapsible pushdown graphs, we also study the model checking problem on the class of all tree-automatic structures. We provide an extension of the known first-order model checking algorithm to Ramsey quantifiers. These are also called Magidor-Malitz quantifiers because these generalised quantifiers were first introduced by Magidor and Malitz [50].

The proof of this result is given by an explicit automata-construction that corresponds to this quantifier. For the string-automatic structures, such a proof was given by Rubin [57] using the concept of word-combs. Rubin then proved that, on string-automatic structures, each set witnessing a Ramsey quantifier contains a word-comb. Using the theory of ω-string-automata, he then uses word-combs to design a finite string-automaton corresponding to the Ramsey quantifier. In joint work with Dietrich Kuske, we extended this result to the tree-case. We define the concept of a tree-comb and use ω-tree-automata in order to provide a finite tree-automata construction that corresponds to the Ramsey quantifier on a tree-automatic structure. We stress that our result is a nontrivial adaption of Rubin's work. The technical difference between the string and the tree case is based on the fact that strings have a uniquely defined length, while the lengths of paths in a tree are not necessarily uniform.

Outline of this Thesis

In Chapter 2, we first review all basic concepts that are necessary for understanding this thesis. Namely, we review different logics, logical interpretations and the concepts of trees and words. We also revisit the theory of Ehrenfeucht-Fraïssé games and develop a new model checking approach based on the analysis of restricted strategies in these games. After these preliminaries, we introduce our objects of study. In Section 2.3, we introduce higher-order pushdown systems, collapsible pushdown systems and nested pushdown trees. After this, we provide some technical results on runs of collapsible pushdown systems in Section 2.4. These results concern the existence and computability of certain runs of level $\mathbf{2}$ collapsible pushdown systems. The technical lemmas provided in this section play a crucial role in proving our results concerning level $\mathbf{2}$ (collapsible) pushdown systems. In Section 2.5, we review the basic concepts and results on tree-automatic structures. At the beginning of Chapter 3 we briefly present our main results in the following order.

1. The second level of the collapsible pushdown hierarchy is tree-automatic and its FO+REACH theory is decidable.
2. First-order model checking on nested pushdown trees is in 2-EXPSPACE.
3. First-order model checking on level 2 nested pushdown trees is decidable.
4. The model checking problem for FO extended by Ramsey quantifiers on tree-automatic structures is decidable.

For each of these results there is one section in Chapter 3 providing the details of the proof and some discussion on related topics. Note that we postpone the formal definition of the hierarchy of higher-order nested pushdown trees to Section 3.3. In that section, we relate this new hierarchy to the hierarchy of higher-order pushdown graphs and to the hierarchy of collapsible pushdown graphs. Finally, Chapter 4 contains concluding remarks and some open problems.

2 Basic Definitions and Technical Results

In the first part of this chapter, we review different kinds of logics and logical interpretations that will play a role in this thesis. Most of this part is assumed to be known to the reader and is merely stated for fixing notation. An exception to this rule is the part on Ehrenfeucht-Fraïssé games. First, we briefly recall the definition and some well-known facts about Ehrenfeucht-Fraïssé games. Afterwards, we introduce a new application of these games to first-order model checking problems. We develop an approach for model checking via the analysis of restricted strategies in the Ehrenfeucht-Fraïssé game played on two identical copies of a fixed structure. If Duplicator has winning strategies that satisfy certain restrictions on each structure of some class $\mathscr{C}$, then we can turn these strategies into an FO model checking algorithm on $\mathscr{C}$.

In Section 2.2 we review the notions of grids and trees. Grids only play a minor role for our results. We use a certain grid-like structure as a counterexample in an undecidability proof. In contrast, trees play a crucial role for our first two main results.

Section 2.3 is an introduction to collapsible pushdown graphs and nested pushdown trees. The main focus of this thesis is on model checking algorithms for the classes of these graphs. As a preparation for the development of these algorithms, we present the most important tool for our results in Section 2.4. In that section we give a detailed analysis of the structure of runs of collapsible pushdown graphs of level $\mathbf{2}$. Finally, in Section 2.5 we recall the necessary notions concerning tree-automatic structures. Note that tree-automatic structures play two different roles in this thesis: our first main result studies the class of tree-automatic structures on its own. We provide a model checking algorithm for first-order logic extended by Ramsey quantifiers (also called Magidor-Malitz quantifiers) on this class. Our algorithm extends the known first-order model checking algorithm on tree-automatic structures.

In the second main result, we use tree-automaticity as a tool. We show that collapsible pushdown graphs of level $\mathbf{2}$ are tree-automatic. Thus, they inherit the decidability of the first-order model checking problem from the general theory of tree-automatic structures.

2.1 Logics and Interpretations

In this section we briefly recall the definitions of the logics we are concerned with. These are classical first-order logic and its extensions by monadic second-order quantifiers, certain generalised quantifiers, reachability predicates, or least fixpoint operators. Furthermore, we present basic modal logic and the modal μ-calculus (denoted by $L\mu$), which is the extension of modal logic by least fixpoint operators. The last part of this section also fixes our notation concerning logical interpretations.

2.1.1 First-Order Logic, Locality and Ehrenfeucht-Fraïssé Games

Vocabularies and Structures

For reasons of convenience, we only introduce relational vocabularies and relational structures because we are only concerned with such structures. A vocabulary (or signature)

$\sigma = ((R_i)_{i\in I})$ consists of relation symbols R_i. Each relation symbol R_i has a fixed arity $\mathsf{ar}(R_i) \in \mathbb{N}$.

A σ-structure $\mathfrak{A}$ is a tuple $(A, (R_i^{\mathfrak{A}})_{i\in I})$ where A is a set called the universe of $\mathfrak{A}$, and $R_i^{\mathfrak{A}} \subseteq A^{\mathsf{ar}(R_i)}$ is a relation of arity $\mathsf{ar}(R_i)$ for each $i \in I$. We denote structures with the letters $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$, and so on. We silently assume that the universe of $\mathfrak{A}$ is a set A, the universe of $\mathfrak{B}$ is a set B, etc.

We introduce the following notation concerning elements of the universe of a structure. For some structure $\mathfrak{A}$, we use the notation $a \in \mathfrak{A}$ for stating that a is some element of the universe A of $\mathfrak{A}$. Furthermore, we use a sloppy notation for tuples of elements. We write $\bar{a} := a_1, a_2, \ldots, a_n \in \mathfrak{A}$ for $\bar{a} := (a_1, a_2, \ldots, a_n) \in A^n$.

First-Order Logic

Let σ be a vocabulary. We denote by $\mathsf{FO}(\sigma)$ first-order logic over the vocabulary σ. Formulas of $\mathsf{FO}(\sigma)$ are composed by iterated use of the following rules:

1. for x, y some variable symbols, $x = y$ is a formula in $\mathsf{FO}(\sigma)$,
2. for $R_i \in \sigma$ a relation of arity $r := \mathsf{ar}(R_i)$ and variable symbols $x_1, x_2, \ldots x_r$, $R_i x_1 x_2 \ldots x_r$ is a formula in $\mathsf{FO}(\sigma)$,
3. for $\varphi, \psi \in \mathsf{FO}(\sigma)$, $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\neg\varphi$ are formulas in $\mathsf{FO}(\sigma)$,
4. for $\varphi \in \mathsf{FO}(\sigma)$ and x a variable symbol, $\exists x\varphi$ and $\forall x\varphi$ are formulas in $\mathsf{FO}(\sigma)$.

Let $\varphi \in \mathsf{FO}(\sigma)$ be a formula. We write $\mathsf{Var}(\varphi)$ for the set of variable symbols occurring in φ. The semantics of first-order formulas is defined as follows. Let $\mathfrak{A}$ be a σ-structure with universe A and $I : \mathsf{Var}(\varphi) \to A$ some function (called the variable assignment or interpretation) we write $\mathfrak{A}, I \models \varphi$, and say $\mathfrak{A}, I$ is a model of φ (or $\mathfrak{A}, I$ satisfies φ), if one of the following holds.

1. φ is of the form $x = y$ where x, y are variable symbols and $I(x) = I(y)$.
2. φ is of the form $R_i x_1 x_2 \ldots x_r$ and $(I(x_1), I(x_2), \ldots I(x_r)) \in R_i^{\mathfrak{A}}$.
3. φ is of the form $\psi \vee \chi$ for $\psi, \chi \in \mathsf{FO}(\sigma)$ and $\mathfrak{A}, I\restriction_{\mathsf{Var}}(\psi) \models \psi$ or $\mathfrak{A}, I\restriction_{\mathsf{Var}}(\chi) \models \chi$.
4. φ is of the form $\psi \wedge \chi$ for $\psi, \chi \in \mathsf{FO}(\sigma)$ and $\mathfrak{A}, I\restriction_{\mathsf{Var}}(\psi) \models \psi$ and $\mathfrak{A}, I\restriction_{\mathsf{Var}}(\chi) \models \chi$.
5. φ is of the form $\neg\psi$ for $\psi \in \mathsf{FO}(\sigma)$ and $\mathfrak{A}, I \not\models \psi$.
6. φ is of the form $\exists x\psi$ and there is some $a \in A$ such that $\mathfrak{A}, I_{x\mapsto a} \models \psi$ where $I_{x\mapsto a} : \mathsf{Var}(\psi) \to A$ with $I_{x\mapsto a}(y) := \begin{cases} I(y) & \text{for } y \neq x, \\ a & \text{for } y = x. \end{cases}$
7. φ is of the form $\forall x\psi$ and $\mathfrak{A}, I_{x\mapsto a} \models \psi$ for all $a \in A$.

We denote by $\mathsf{Free}(\varphi) \subseteq \mathsf{Var}(\varphi)$ the set of variables occurring free in φ. A variable x does not occur free if it only occurs under the scope of quantifiers $\exists x$ or $\forall x$. If $\mathsf{Free}(\varphi) \subseteq \{x_1, x_2, \ldots, x_n\}$ we use the notation

$$\mathfrak{A}, a_1, a_2, \ldots, a_n \models \varphi(x_1, x_2, \ldots, x_n)$$

for $\mathfrak{A}, I \models \varphi$ if I maps x_i to a_i. Furthermore, if $x_1, x_2, \ldots, x_n$ are clear from the context, we also use the notation $\mathfrak{A} \models \varphi(a_1, a_2, \ldots, a_n)$.

In the following, we write FO instead of $\mathsf{FO}(\sigma)$ whenever σ is clear from the context or if a statement does not depend on the concrete σ. We may assign to each formula in FO its quantifier rank. This is the maximal nesting depth of existential and universal quantifications in this formula. We write FO_ρ for the restriction of FO to formulas of quantifier rank up to ρ.

Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures. For n parameters $\bar{a} \in \mathfrak{A}$ and n parameters $\bar{b} \in \mathfrak{B}$, we write $\mathfrak{A}, \bar{a} \equiv_\rho \mathfrak{B}, \bar{b}$ for the fact that $\mathfrak{A} \models \varphi(\bar{a})$ if and only if $\mathfrak{B} \models \varphi(\bar{b})$ for all $\varphi \in \mathsf{FO}_\rho$ with free variables among $x_1, x_2, \ldots, x_n$.

We conclude the section on first-order logic by recalling two important concepts for the analysis of first-order theories. Firstly, we present the concepts of Gaifman locality and Gaifman graphs. Afterwards, we present Ehrenfeucht-Fraïssé games which are a classical tool for the analysis of $\equiv_\rho$. In this thesis, we develop a nonstandard application of these games for the design of model checking algorithms.

Gaifman-Locality

First-order logic has a local nature, i.e., first-order formulas can only express properties about local parts of structures. For example, reachability along a path of some relation E is not first-order expressible. Gaifman introduced the notions of Gaifman graphs and local neighbourhoods in order to give a precise notion of the local nature of first-order logic. Let us start by recalling these notions.

Definition 2.1.1. Let $\sigma = (R_1, R_2, \ldots, R_n)$ be a finite relational signature and let $\mathfrak{A}$ be a σ-structure. The Gaifman graph $\mathcal{G}(\mathfrak{A})$ is the graph (A, E) where A is the universe of $\mathfrak{A}$ and $E \subseteq A^2$ is the relation defined as follows. E connects two distinct elements of A if they appear together in a tuple of some relation of $\mathfrak{A}$, i.e., $(a_1, a_2) \in E$ for $a_1 \neq a_2$ if and only if there is an $1 \leq i \leq n$ and tuples $\bar{x}, \bar{y}, \bar{z} \in A$ such that

$$\mathfrak{A} \models R_i \bar{x} a_1 \bar{y} a_2 \bar{z} \text{ or}$$
$$\mathfrak{A} \models R_i \bar{x} a_2 \bar{y} a_1 \bar{z}.$$

For $a_1, a_2 \in A$ we say a_1 and a_2 have distance n in $\mathfrak{A}$, written $\mathsf{dist}(a_1, a_2) = n$, if their distance in $\mathcal{G}(\mathfrak{A})$ is n. Analogously, we use the terminology $\mathsf{dist}(a_1, a_2) \leq n$ with the obvious meaning. Note that $\mathsf{dist}(x_1, x_2) \leq n$ is first-order definable for each fixed $n \in \mathbb{N}$.[1]

We define the n-local neighbourhood of some tuple $\bar{a} = a_1, a_2, \ldots, a_n \in A$ inductively by

$$\mathcal{N}_0(\bar{a}) := \{a_1, a_2, \ldots, a_n\} \text{ and}$$
$$\mathcal{N}_{n+1}(\bar{a}) := \{a \in A : \exists a' \in \mathcal{N}_n(\bar{a}) \text{ such that } \mathsf{dist}(a, a') \leq 1\}.$$

When we say that "first-order logic is Gaifman-local", we refer to the fact that for each quantifier rank ρ, there is a natural number n such that for each formula $\varphi \in \mathsf{FO}_\rho$ the question whether a structure $\mathfrak{A}$ is a model of φ only depends on the n-local neighbourhoods of the elements in the structure $\mathfrak{A}$. More precisely, any first-order formula is a boolean combination of local formulas and local sentences which we introduce next.

[1] Note that the restriction to finite vocabulary is essential for this statement.

Definition 2.1.2. Let σ be a finite vocabulary and let $\varphi \in \mathsf{FO}$ be some formula with free variables $\bar{x}$. We write φ^n for the relativisation of φ to the n-local neighbourhood of $\bar{x}$. φ^n is obtained from φ by replacing each quantifier $\exists y(\psi)$ by

$$\exists y(\mathsf{dist}(y, \bar{x}) \leq n \wedge \psi)$$

and each quantifier $\forall y(\psi)$ by

$$\forall y(\mathsf{dist}(y, \bar{x}) \leq n \rightarrow \psi).$$

This means that for each variable assignment I that maps $\bar{x} \mapsto \bar{a} \in A$, $\mathfrak{A}, I \models \varphi^n(\bar{x})$ if and only if $\mathfrak{A}\restriction_{\mathscr{N}_n(\bar{a})}, I \models \varphi$.

We call $\varphi(x)$ an n-local formula if $\varphi(x) \equiv \varphi^n(x)$ and we call it local if it is n-local for some $n \in \mathbb{N}$.

We call a sentence local if it is of the form

$$\exists x_1 \ldots \exists x_n \bigwedge_{1 \leq i < j \leq n} \mathsf{dist}(x_i, x_j) > 2l \wedge \bigwedge_{1 \leq i \leq n} \psi^l(x_i)$$

for some formula ψ and some $l \in \mathbb{N}$. Such a sentence asserts that there are n elements far apart from one another each satisfying the l-local formula ψ^l.

Using this notation we can state Gaifman's Lemma.

Lemma 2.1.3 ([25]). Each first-order formula is equivalent to a boolean combination of local sentences and local formulas.

This lemma has an interesting consequence. For each quantifier rank $\rho \in \mathbb{N}$ there is a natural number $n \in \mathbb{N}$ such that the following hold: if $\bar{a} \in A$ and $a, a' \in A$ are such that $\mathsf{dist}(a, \bar{a}) > 2\rho + 1$ and $\mathsf{dist}(a', \bar{a}) > 2\rho + 1$ and there is an isomorphism $\mathscr{N}_\rho(a) \simeq \mathscr{N}_\rho(a')$ mapping a to a', then $\mathfrak{A}, \bar{a}, a \equiv_\rho \mathfrak{A}, \bar{a}, a'$.

In Section 3.2.2, we develop a lemma of a similar style. But in contrast to Gaifman's Lemma, this new lemma is tailored towards an application on graphs of small diameter. Due to the small diameter, all elements a and $\bar{a}$ satisfy $\mathsf{dist}(a, \bar{a}) \leq 2n + 1$ whence we cannot use Gaifman's Lemma itself. Nevertheless, in that section we need a lemma that provides $\equiv_\rho$-equivalence for certain tuples $\bar{a}, a$ and $\bar{a}, a'$ in certain graphs of small diameter. We obtain this lemma using Ehrenfeucht-Fraïssé games which we introduce in the following.

Ehrenfeucht-Fraïssé Games and First-Order Model Checking

The equivalence $\equiv_\rho$ of first-order logic up to quantifier rank ρ has a nice characterisation via Ehrenfeucht-Fraïssé games. Based on the work of Fraïssé [24], Ehrenfeucht [22] introduced these games which have become one of the most important tools for proving inexpressibility of properties in first-order logic. This tool is especially important in the context of finite model theory where other methods, e.g. compactness, fail. The game is played by two players, who are called Spoiler and Duplicator. They play on two σ-structures $\mathfrak{A}_1$ and $\mathfrak{A}_2$. The players alternatingly choose elements in the two structures. At the end of the game, Duplicator has won if there is a partial isomorphism between the elements chosen in each of the structures. Thus, Spoiler's goal is to choose elements in such a way that no choice of Duplicator yields a partial isomorphism between the elements chosen so far. The precise definitions are as follows.

Definition 2.1.4. Let $\mathfrak{A}_1$ and $\mathfrak{A}_2$ be σ-structures. For

$$\bar{a}^1 = a_1^1, a_2^1, \ldots, a_m^1 \in A_1^m \text{ and}$$
$$\bar{a}^2 = a_1^2, a_2^2, \ldots, a_m^2 \in A_2^m$$

we write $\bar{a}^1 \mapsto \bar{a}^2$ for the map that maps a_i^1 to a_i^2 for all $1 \leq i \leq m$.

In the n-round Ehrenfeucht-Fraïssé game on $\mathfrak{A}_1, a_1^1, a_2^1, \ldots, a_m^1$ and $\mathfrak{A}_2, a_1^2, a_2^2, \ldots, a_m^2$ for $a_i^j \in A_j$ there are two players, Spoiler and Duplicator, which play according to the following rules. The game is played for n rounds. The i-th round consists of the following steps

1. Spoiler chooses one of the structures, i.e., he chooses $j \in \{1, 2\}$.
2. Then he chooses one of the elements of his structure, i.e., he chooses some $a_{m+i}^j \in A_j$.
3. Now, Duplicator chooses an element in the other structure, i.e., for $k := 3 - j$, she chooses some $a_{m+i}^k \in A_k$.

Having executed n rounds, Spoiler and Duplicator defined tuples

$$\bar{a}^1 := a_1^1, a_2^1, \ldots, a_{m+n}^1 \in A_1^{m+n} \text{ and}$$
$$\bar{a}^2 := a_1^2, a_2^2, \ldots, a_{m+n}^2 \in A_2^{m+n}.$$

Duplicator wins the play if $f : \bar{a}^1 \mapsto \bar{a}^2$ is a partial isomorphism, i.e., if f satisfies the following conditions.

1. $a_i^1 = a_j^1$ if and only if $a_i^2 = a_j^2$ for all $1 \leq i \leq j \leq m + n$ and
2. for each $R_i \in \sigma$ of arity r the following hold: if $i_1, i_2, \ldots i_r$ are numbers between 1 and $m + n$, $\mathfrak{A}_1, \bar{a}^1 \models R_i x_{i_1} x_{i_2} \ldots x_{i_r}$ if and only if $\mathfrak{A}_2, \bar{a}^2 \models R_i x_{i_1} x_{i_2} \ldots x_{i_r}$.

Definition 2.1.5. Let $\mathfrak{A}_1$, $\mathfrak{A}_2$ be structures and $\bar{a}^1 \in \mathfrak{A}^n$, $\bar{a}^2 \in \mathfrak{A}_2$. We write $\mathfrak{A}_1, \bar{a}^1 \simeq_\rho \mathfrak{A}_2, \bar{a}^2$ if Duplicator has a winning strategy in the ρ-round Ehrenfeucht-Fraïssé game on $\mathfrak{A}_1, \bar{a}^1$ and $\mathfrak{A}_2, \bar{a}^2$.

Our interest in Ehrenfeucht-Fraïssé games stems from the following relationship of $\simeq_\rho$ and $\equiv_\rho$ (recall that $\equiv_\rho$ is equivalence with respect to FO formulas up to quantifier rank ρ).

Lemma 2.1.6 ([24],[22]). For all σ-structures $\mathfrak{A}_1$, $\mathfrak{A}_2$, and for all tuples $\bar{a}^1 \in A_1^n$, and $\bar{a}^2 \in A_2^n$,

$$\mathfrak{A}_1, \bar{a}^1 \simeq_\rho \mathfrak{A}_2, \bar{a}^2 \text{ iff } \mathfrak{A}_1, \bar{a}^1 \equiv_\rho \mathfrak{A}_2, \bar{a}^2,$$

i.e., Duplicator has a winning strategy in the ρ round Ehrenfeucht-Fraïssé game on $\mathfrak{A}_1, \bar{a}^1$ and $\mathfrak{A}_2, \bar{a}^2$ if and only if $\mathfrak{A}_1, \bar{a}^1$ and $\mathfrak{A}_2, \bar{a}^2$ are indistinguishable by first-order formulas of quantifier rank ρ.

Remark 2.1.7. We want to give some brief comments on the proof.

If $\mathfrak{A}_1, \bar{a}^1 \not\equiv_\rho \mathfrak{A}_2, \bar{a}^2$, then there is a formula φ in negation normal form (i.e., negation only occurs in negated atomic formulas) such that $\mathfrak{A}_1, \bar{a}^1 \models \varphi$ but $\mathfrak{A}_2, \bar{a}^2 \not\models \varphi$. By induction on the structure of φ one can prove that there is a winning strategy for Spoiler. Basically, for every subformula starting with an existential quantification, Spoiler chooses a witness

for this quantification in $\mathfrak{A}$ and for each universal quantification, he chooses an element in $\mathfrak{B}$ witnessing the negation of the subformula. Due to the fact that the second structure does not satisfy φ, Duplicator must eventually respond with an element not satisfying the existential claim made by Spoiler. By clever choice of further elements, Spoiler can then point out this difference and Duplicator will loose the game.

On the other hand, if the two structures cannot be distinguished by quantifier rank ρ formulas, then Duplicator just has to preserve the equivalence of the quantifier rank m types of the elements chosen in both structures, where m is the number of rounds left to play. Note that the resulting partial map is a partial isomorphism if and only if it preserves all quantifier-free formulas. Thus, Duplicator wins the game using the strategy indicated above.

Ehrenfeucht-Fraïssé games are usually used to show that first-order logic cannot express certain properties. We stress that our main application of these games is nonstandard. Nevertheless, we first present an example of this classical application. In Section 3.3 we use the result of this example.

Example 2.1.8. We present a proof that there are only finitely many types of coloured finite successor structures that are distinguishable by FO_ρ. This example will also illustrate how the concept of Gaifman locality can be fruitfully applied to the analysis of Ehrenfeucht-Fraïssé games.[2] In general, the analysis of Ehrenfeucht-Fraïssé games is difficult because one has to consider too many possible choices for Spoiler. But if the structure of the local neighbourhoods is simple, this can be used to analyse Duplicator's strategies in the game.

A finite successor structure is up to isomorphism a structure of the form

$$\mathfrak{A} := (\{1, 2, \dots n\}, \mathsf{succ}, P_1, P_2, \dots, P_m)$$

for some $n, m \in \mathbb{N}$ where $\mathsf{succ} = \{(k, k+1) : 1 \leq k < n\}$ is the successor relation on the natural numbers up to n and $P_1, \dots, P_m$ are unary predicates (which we call colours). We are going to show that for fixed $m \in \mathbb{N}$ there are at most $(\rho + 2^\rho)^{(2^m+1)^{2^{\rho+1}+1}}$ successor structures with m colours that are pairwise not $\simeq_\rho$-equivalent.

In order to prove this claim, we consider a successor structure $\mathfrak{A}$ with m colours and n elements. We will make use of the 2^l-local neighbourhood $\mathcal{N}_{2^l}(a)$ of the elements $a \in \mathfrak{A}$.

Note that $\mathcal{N}_{2^l}(a)$ is a successor structure with exactly $2^{l+1} + 1$ many elements unless $a \leq 2^l$ or $a \geq n - 2^l$. Since there are at most 2^m many possibilities to colour a node with m colours, there are at most $(2^m + 1)^{2^{l+1}+1}$ many distinct 2^l-local neighbourhoods up to isomorphism. The base $2^m + 1$ is due to the fact that elements may be undefined (if $a \leq 2^l$ or $a \geq n - 2^l$) or coloured in one of the 2^m possibilities.

We claim that the number of occurrences of each 2^ρ-local neighbourhood type counted up to threshold $2^\rho + \rho$ determines the $\simeq_\rho$-type of a successor structure.

In order to prove this, we use Ehrenfeucht-Fraïssé games. Before we explain Duplicator's strategy, note the following facts.

1. Counting the occurrences of each 2^l-local neighbourhood type up to some threshold $t \in \mathbb{N}$ determines the occurrences of 2^{l-1}-neighbourhood types up to threshold t. Furthermore, the 2^l-local neighbourhood of an element $a \in \mathfrak{A}$ determines the 2^{l-1}-local neighbourhood of the elements $a - 2^{l-1}, a - 2^{l-1} + 1, \dots, a + 2^{l-1} - 1, a + 2^{l-1}$.

[2] Our example is in fact an application of Hanf's Lemma (cf. [28]).

2. For any $k < l$, the union of the 2^{l-k-1}-local neighbourhoods of k elements contains at most $k(2^{l-k}+1) = k + 2^{\ln(k)+l-k} \leq k + 2^l < l + 2^l$ many elements.

3. The 2^l-local neighbourhood types of the 2^l first and the 2^l last elements of a successor structure $\mathfrak{A}$ occur exactly once in $\mathfrak{A}$ because they are determined by the number of elements that exist to their left, respectively, right.

Let $\mathfrak{A}$ and $\mathfrak{B}$ be structures that have, up to threshold $2^\rho + \rho$, the same number of occurrences of each 2^ρ-local neighbourhood type.

Duplicator has the following strategy in the ρ round Ehrenfeucht-Fraïssé game. Without loss of generality, Spoiler chooses at first some element $a_1 \in \mathfrak{A}$. Duplicator may respond with any element $b_1 \in \mathfrak{B}$ such that $\mathcal{N}_{2^{\rho-1}}(a)$ and $\mathcal{N}_{2^{\rho-1}}(b)$ are isomorphic.

For the following $\rho - 1$ rounds, we distinguish between local and global moves of Spoiler. Assume that in the i-th round the game is in position $(a_1, a_2, \ldots, a_{i-1}) \mapsto (b_1, b_2, \ldots, b_{i-1})$ such that the following holds.

1. For each $1 \leq j \leq i-1$, $\mathcal{N}_{2^{\rho-(i-1)}}(a_j)$ and $\mathcal{N}_{2^{\rho-(i-1)}}(b_2)$ are isomorphic.

2. Up to threshold $1 + 2^{\rho-(i-1)}$, the distance of a_j from a_k agrees with the distance of b_j from b_k, i.e., a_j is the n-th successor of a_k for some $n \leq 2^{\rho-(i-1)}$ if and only if b_j is the n-th successor of b_k.

Due to symmetry we may assume that Spoiler chooses some $a_i \in \mathfrak{A}$. We call this move local, if there is some $j < i$ such that the distance between a_i and a_j is at most $2^{\rho-i}$. In this case, Duplicator chooses the element b_i that has the same distance to b_j as a_i to a_j. Since the $2^{\rho-(i-1)}$-local neighbourhood of a_j and b_j coincide, the $2^{\rho-i}$-local neighbourhood of a_i and b_i agree. Furthermore, note that the distances of a_i from each a_k and the distances of b_i from the corresponding b_k agree up to threshold $2^{\rho-i}$.

If Spoiler chooses some $a_i \in \mathfrak{A}$ such that the distance between a_i and a_j for all $j < i$ is more than $2^{\rho-i}$, we call the move global. In this case, Duplicator chooses an element b_i such that $\mathcal{N}_{2^{\rho-i}}(a_i) \simeq \mathcal{N}_{2^{\rho-i}}(b_i)$ and such that the distance from b_i to any b_j is more than $2^{\rho-i}$ for all $j < i$. Such an element b_i exists due to the following facts.

1. The $2^{\rho-i}$-local neighbourhoods of $a_1, a_2, \ldots, a_{i-1}$ contain less than $\rho + 2^\rho$ many elements.

2. For each $j < i$, $\mathcal{N}_{2^{\rho-(i-1)}}(a_j) \simeq \mathcal{N}_{2^{\rho-(i-1)}}(b_j)$. Thus, the $2^{\rho-i}$-local neighbourhoods of the elements of distance at most $2^{\rho-i}$ from one of the a_j are isomorphic to the corresponding elements that are close to b_j. Hence, for each isomorphism-type of a $2^{\rho-i}$-local neighbourhood the number of elements that realise this type and that are close to one of the a_j coincide with the number of elements that realise this type and that are close to one of the b_j. Let k be the number of elements a close to one of the a_j such that $\mathcal{N}_{2^{\rho-i}}(a) \simeq \mathcal{N}_{2^{\rho-i}}(a_i)$.

3. Since a_i is far away from all a_j, there are at least $k+1 \leq \rho + 2^\rho$ many elements of neighbourhood type $\mathcal{N}_{2^{\rho-i}}(a)$ in $\mathfrak{A}$. Due to our assumptions on $\mathfrak{A}$ and $\mathfrak{B}$ and on the neighbourhoods of the elements chosen so far, there are at least $k+1$ elements of the neighbourhood type $\mathcal{N}_{2^{\rho-i}}(a)$ in $\mathfrak{B}$ of which exactly k have distance at most $2^{\rho-i}$ of one of the b_j. Thus, there is an element $b_i \in \mathfrak{B}$ such that $\mathcal{N}_{2^{\rho-i}}(b_i) \simeq \mathcal{N}_{2^{\rho-i}}(a_i)$ that is far away from all the b_j for $j < i$.

It is straightforward to see that, after ρ rounds, we end up with a partial map

$$f : (a_1, a_2, \ldots, a_\rho) \mapsto (b_1, b_2, \ldots, b_\rho) \text{ such that}$$

1. for each $1 \leq j \leq \rho$, $\mathcal{N}_1(a_j) \simeq \mathcal{N}_1(b_j)$ whence the colours of a_j and the colours of b_j are equal, and
2. up to threshold 2, the distance of a_j from a_k agrees with the distance of b_j from b_k, i.e., f and f^{-1} preserve the successor relation.

Thus, f is a partial isomorphism and Duplicator wins the game.

Note that counting 2^ρ-local neighbourhoods up to threshold $\rho + 2^\rho$ assigns to each successor structure with m colours a function $(2^m+1)^{2^{\rho+1}+1} \to (\rho + 2^\rho)$. We have seen that if these functions agree for two structures $\mathfrak{A}$ and $\mathfrak{B}$, then Duplicator wins the ρ round Ehrenfeucht-Fraïssé game on these two structures whence $\mathfrak{A} \equiv_\rho \mathfrak{B}$. Thus, there are at most $(\rho + 2^\rho)^{(2^m+1)^{2^{\rho+1}+1}}$ many m-coloured successor structures that can be distinguished by quantifier rank ρ first-order formulas.

We will use the result of the previous example in Section 3.3.5. But beside this classical application of Ehrenfeucht-Fraïssé games, a nonstandard application of Ehrenfeucht-Fraïssé games plays a much more important role in this thesis. This application gives rise to FO model checking algorithms on certain classes of structures. Ferrante and Rackoff[23] were the first to mention the general approach of using Ehrenfeucht-Fraïssé analysis for the the decidability of FO theories.

We consider the game played on two copies of the same structure, i.e., the game on $\mathfrak{A}, \bar{a}^1$ and $\mathfrak{A}, \bar{a}^2$ with identical choice of the initial parameter $\bar{a}^1 = \bar{a}^2 \in \mathfrak{A}$. At a first glance, this looks quite uninteresting because Duplicator has of course a winning strategy in this setting: he can copy each move of Spoiler. But we want to look for winning strategies with certain constraints. In our application the constraint will be that Duplicator is only allowed to choose elements that are represented by short runs of certain automata, but the idea can be formulated more generally.

Definition 2.1.9. Let $\mathscr{C}$ be a class of structures. Assume that $S^{\mathfrak{A}}(m) \subseteq \mathfrak{A}^m$ is a subset of the m-tuples of the structure $\mathfrak{A}$ for each $\mathfrak{A} \in \mathscr{C}$ and each $m \in \mathbb{N}$. Set $S := (S^{\mathfrak{A}}(m))_{m \in \mathbb{N}, \mathfrak{A} \in \mathscr{C}}$. We call S a constraint for Duplicator's strategy and we say Duplicator has an S-preserving winning strategy if she has a strategy for each game played on two copies of $\mathfrak{A}$ for some $\mathfrak{A} \in \mathscr{C}$ with the following property. Let $\bar{a}^1 \mapsto \bar{a}^2$ be a position reached after m rounds where Duplicator used her strategy. If $\bar{a}^2 \in S(m)$, then Duplicator's strategy chooses an element a^2_{m+1} such that $\bar{a}^2, a^2_{m+1} \in S^{\mathfrak{A}}(m+1)$ for each challenge of Spoiler in the first copy of $\mathfrak{A}$.

Remark 2.1.10. We write $S(m)$ for $S^{\mathfrak{A}}(m)$ if $\mathfrak{A}$ is clear from the context.

Recall the following fact: if Duplicator uses a winning strategy in the n round game, her choice in the $(m+1)$-st round is an element a^2_{m+1} such that $\mathfrak{A}, \bar{a}^1, a^1_{m+1} \equiv_{n-m-1} \mathfrak{A}, \bar{a}^2, a^2_{m+1}$.

This implies that if Duplicator has an S-preserving winning strategy, then for every formula $\varphi(x_1, x_2, \ldots, x_{m+1}) \in \mathrm{FO}_{n-m-1}$ and for all $\bar{a} \in A^m$ with $\bar{a} \in S(m)$ the following holds:

$$\begin{aligned}
&\text{there is an element } a \in A \text{ such that } \bar{a}, a \in S(m+1) \text{ and } \mathfrak{A}, \bar{a}, a \models \varphi \\
&\text{iff there is an element } a \in A \text{ such that } \mathfrak{A}, \bar{a}, a \models \varphi \\
&\text{iff } \mathfrak{A}, \bar{a} \models \exists x_{m+1} \varphi.
\end{aligned}$$

Replacing existential quantification with universal quantification we obtain directly that this statement is equivalent to

$$\begin{aligned} &\text{for all } a \in A \text{ such that } \bar{a}, a \in S(m+1) \text{ we have } \mathfrak{A}, \bar{a}, a \models \varphi \\ &\text{iff } \mathfrak{A}, \bar{a} \models \forall x \varphi(\bar{y}, x). \end{aligned}$$

Algorithm: $\texttt{ModelCheck}(\mathfrak{A}, \bar{a}, \varphi(\bar{x}))$

Input: a structure $\mathfrak{A}$,a formula $\varphi \in \mathrm{FO}_\rho$, an assignment $\bar{x} \mapsto \bar{a}$

if φ is an atom or negated atom then
  if $\mathfrak{A}, \bar{a} \models \varphi(\bar{x})$ then accept else reject;
if $\varphi = \varphi_1 \vee \varphi_2$ then
  if $\texttt{ModelCheck}(\mathfrak{A}, \bar{a}, \varphi_1)$ = accept then accept else
    if $\texttt{ModelCheck}(\mathfrak{A}, \bar{a}, \varphi_2)$= accept then accept else reject;
if $\varphi = \varphi_1 \wedge \varphi_2$ then
  if $\texttt{ModelCheck}(\mathfrak{A}, \bar{a}, \varphi_1)$= $\texttt{ModelCheck}(\mathfrak{A}, \bar{a}, \varphi_2)$= accept then accept else reject;
if $\varphi = \exists x \varphi_1(\bar{x}, x)$ then
  check whether there is an $a \in \mathfrak{A}$ such that $\texttt{ModelCheck}(\mathfrak{A}, \bar{a}a, \varphi_1)$= accept;
if $\varphi = \forall x_i \varphi_1$ then
  check whether $\texttt{ModelCheck}(\mathfrak{A}, \bar{a}a, \varphi_1)$= accept holds for all $a \in \mathfrak{A}$;

Algorithm 1: The general FO-model checking as pseudo-code

Now, we want to make use of this observation in a general approach to first-order model checking. The pseudo-algorithm in Algorithm 1 is a correct description of first-order model checking as it just proceeds by syntactic induction on the first-order formula in order to determine whether the given structure is a model of the given formula. But of course, in general this is no algorithm. As soon as $\mathfrak{A}$ is infinite and a quantification occurs in φ, this pseudo-algorithm would not terminate because it would have to check infinitely many variable assignments. Nevertheless, it is correct in the sense that if we consider a class of structures where we could check these infinitely many variable assignments in finite time, then it would correctly determine the answer to the model checking problem. Using S-preserving strategies, we want to turn the pseudo-algorithm into a proper algorithm for certain classes of structures. For this purpose, we first introduce the following notation.

Definition 2.1.11. Given a class $\mathscr{C}$ of finitely represented structures, we call a constraint S for Duplicator's strategy finitary on $\mathscr{C}$, if for each $\mathfrak{A} \in \mathscr{C}$ we can compute a function $f_{\mathfrak{A}}$ such that for all $n \in \mathbb{N}$

- $S^{\mathfrak{A}}(n)$ is finite,
- for each $\bar{a} \in S^{\mathfrak{A}}(n)$, we can represent $\bar{a}$ in space $f_{\mathfrak{A}}(n)$, and
- $\bar{a} \in S^{\mathfrak{A}}(n)$ is effectively decidable.

Using such a finitary constraint, we can rewrite the model checking algorithm from above into Algorithm 2. The condition of a finitary constraint is exactly what is needed to guarantee termination of this algorithm. Furthermore, our observation on S-preserving constraints implies that this algorithm is correct for all structures from a class $\mathscr{C}$ where Duplicator has an S-preserving winning strategy for every $\mathfrak{A} \in \mathscr{C}$. We apply this idea in Sections 3.2 and

3.3. There, we represent elements of certain structures $\mathfrak{A}$ by runs of some automaton. The sets $S^{\mathfrak{A}}(n)$ consist of runs that have length bounded by some function $f_{\mathfrak{A}}$ that is computable from the automaton representing $\mathfrak{A}$.

Algorithm: $\texttt{SModelCheck}(\mathfrak{A}, \bar{a}, \varphi(\bar{x}))$
Input: a structure $\mathfrak{A}$, a formula $\varphi \in \mathsf{FO}_\rho$, an assignment $\bar{x} \mapsto \bar{a}$ for tuples $\bar{x}, \bar{a}$ of arity m such that $\bar{a} \in S(m)$
if φ is an atom or negated atom then
  if $\mathfrak{A}, \bar{a} \models \varphi(\bar{x})$ then accept else reject;
if $\varphi = \varphi_1 \vee \varphi_2$ then
  if $\texttt{SModelCheck}(\mathfrak{A}, \bar{a}, \varphi_1)$ = accept then accept else
    if $\texttt{SModelCheck}(\mathfrak{A}, \bar{a}, \varphi_2)$= accept then accept else reject;
if $\varphi = \varphi_1 \wedge \varphi_2$ then
  if $\texttt{SModelCheck}(\mathfrak{A}, \bar{a}, \varphi_1)$= $\texttt{SModelCheck}(\mathfrak{A}, \bar{a}, \varphi_2)$= accept then accept else reject;
if $\varphi = \exists x \varphi_1(\bar{x}, x)$ then
  check whether there is an $a \in \mathfrak{A}$ such that $\bar{a}, a \in S(m+1)$ and
  $\texttt{SModelCheck}(\mathfrak{A}, \bar{a}a, \varphi_1)$= accept;
if $\varphi = \forall x_i \varphi_1$ then
  check whether $\texttt{SModelCheck}(\mathfrak{A}, \bar{a}a, \varphi_1)$= accept holds for all $a \in \mathfrak{A}$ such that $\bar{a}, a \in S(m+1)$;

Algorithm 2: FO-model checking on S-preserving structures

2.1.2 Extensions of First-Order Logic

In many cases the expressive power of FO is too weak. For example, due to the local nature of first-order logic, simple reachability questions cannot be formalised in FO. In order to overcome this weakness, there have been proposed a lot of different extensions tailored for different applications. In the following, we present those extensions that we use later.

Monadic Second-Order Logic

Perhaps the most classical extension of first-order logic is monadic second-order logic (abbreviated MSO). The formulas of this logic are defined using the same rules as for FO but additionally adding quantification over subsets. For this, we fix a set $X_1, X_2, \ldots$ of set variable symbols. We extend the formation rules of first-order logic by the following two rules.

- $X_i x$ is an MSO formula for any variable symbol x and any set variable symbol X_i.
- If φ is an MSO formula then $\exists X_i \varphi$ and $\forall X_i \varphi$ are also MSO formulas.

For the semantics, we extend the variable assignment I to the set of set variable symbols occurring in a formula φ. Now, I maps each symbol X_i to a subset of the structure $\mathfrak{A}$. We then set

- $\mathfrak{A}, I \models X_i x$ if $I(x) \in I(X_i)$,
- $\mathfrak{A}, I \models \exists X \varphi$ if there is some $M \subseteq A$ such that $\mathfrak{A}, I_{X \mapsto M} \models \varphi$ where $I_{X \mapsto M}$ is identical to I but maps X to M, and

- $\mathfrak{A}, I \models \forall X \varphi$ if $\mathfrak{A}, I_{X \mapsto M} \models \varphi$ for all $M \subseteq A$.

MSO is the most expressive logic that we are going to consider. But this expressive power comes at a prize. The MSO model checking on collapsible pushdown graphs and nested pushdown trees is undecidable. Thus, we look for weaker extensions of first-order logic that are still decidable in our setting.

Monadic Least Fixpoint Logic

Another approach for extending first-order logic is the use of fixpoint-operators. Here, we present the monadic least fixpoint logic (MLFP). Consider an MSO formula φ without quantification over sets, with a free variable x and a free set variable X that only occurs positively, i.e., that only occurs under an even scope of negations. For each structure $\mathfrak{A}$, each variable assignment I, and $M \subseteq A$, we write I^M for $I_{X \mapsto M}$, the variable assignment that is identical to I but maps the set variable X to $M \subseteq A$. Now, φ defines a monotone operator

$$f^\varphi : 2^A \to 2^A$$
$$f^\varphi(M) := \{a \in A : \mathfrak{A}, I^M_{x \mapsto a} \models \varphi\}.$$

Due to the theorem of Knaster and Tarski [42], f^φ has a unique least fixpoint $M \subseteq A$, i.e., there is a minimal set $M^\varphi \subseteq A$ such that $f^\varphi(M^\varphi) = M^\varphi$. MLFP is the extension of FO by the rule that $[\mathrm{lfp}_{x,X}\varphi](y)$ is an MLFP formula where φ is a formula as described above and where y is a free variable. The semantics of $\psi = [\mathrm{lfp}_{x,X}\varphi](y)$ is defined by $\mathfrak{A}, I \models \psi$ iff $I(y) \in M^\varphi$.

It is clear that the expressive power of MLFP is between the expressive power of FO and that of MSO. Each MLFP formula can be translated into an equivalent MSO formula because the least fixpoint of $\varphi(x,X)$ is defined by the formula

$$\psi(Z) := \big(\forall Y \forall y \ (Yy \leftrightarrow \varphi(y,Y))\big) \to Z \subseteq Y$$

were $X \subseteq Y$ is an abbreviation for $\forall x(Xx \to Yx)$. Thus, $[\mathrm{lfp}_{x,X}\varphi](y)$ can be translated into $\exists Z(\psi(Z) \wedge Zy)$. The expressive power of MLFP is strictly greater than that of FO because fixpoints can be used to formalise reachability queries. For example, the fixpoint induced by the formula $\varphi(x,X) := Px \vee \exists y(Exy \wedge Xy)$ contains all elements for which an E-path to an element in P exists. Due to the local nature of first-order logic, this is not expressible with an FO formula.

The least fixpoint operator is a very strong extension of FO in the sense that MLFP is much more expressive than FO. As in the case of MSO, MLFP is too powerful on those structures we are interested in. We will show that the MLFP-theory of a certain collapsible pushdown graph of level 2 is undecidable.

Thus, in order to find logics with a decidable model checking problems on collapsible pushdown graphs, we look at logics with strictly weaker expressive power than that of MLFP.

FO + Reachability Predicates

During the last decade another extension of first-order logic has been studied and successfully applied for model checking. If one looks at verification problems, the most important properties that one wants to verify often involve reachability of certain states. Thus, for classes of graphs where MSO and MLFP are undecidable, one may study the weakest extension of first-order logic that allows to express reachability questions. We call this logic FO(REACH) and we introduce it formally in the following definition.

Definition 2.1.12. Let $\sigma = (E_1, E_2, \ldots, E_n)$ be a relational signature and E_i a binary relation symbol for each $1 \leq i \leq n$. Let $\mathsf{FO(REACH)}$ denote the smallest set generated by the formation rules of first-order logic plus the rule that $\mathsf{REACH}xy$ is an $\mathsf{FO(REACH)}$ formula for each pair of variables x, y.

$\mathsf{FO(REACH)}$ inherits its semantics mainly from FO. If we consider $\mathfrak{A}$ as a graph with edge relation $E := \bigcup_{1 \leq j \leq n} E_j^{\mathfrak{A}}$ where each edge is labelled with a nonempty subset of $\{1, 2, \ldots, n\}$ then REACH is interpreted as the transitive closure of the edge relation E.

Similar to FO extended by reachability we now introduce FO extended by regular reachability. Let $\sigma, \mathfrak{A}$ and E be defined as before. For simplicity, we assume that each edge in $\mathfrak{A}$ is labelled by exactly one label from $\{1, 2, \ldots, n\}$, i.e., for all $a, a' \in A$, $(a, a') \in E_i$ and $(a, a') \in E_j$ implies $i = j$.

We write $\mathsf{FO}(\mathrm{Reg})$ for the extension of FO by atomic formulas $\mathsf{REACH}_L xy$ for each regular language $L \subseteq \{1, 2, \ldots, n\}^*$ and all variable symbols x, y.

For $a, b \in \mathfrak{A}$, $\mathsf{REACH}_L ab$ holds if there is a path $a = a_1, a_2, a_3, \ldots, a_k = b$ such that $(a_i, a_{i+1}) \in E$ for all $1 \leq i < k$ and the word formed by the labels of the edges along this path form a word of L.

FO + Generalised Quantifiers

Lastly, we present the extensions of first-order logic by generalised quantifiers. The idea of generalised quantifiers was first introduced by Mostowski [52] and then further developed to full generality by Lindström [49]. We briefly recall the general notion. Afterwards, we present the generalised quantifiers that occur in this thesis: the infinite existential quantifier $\exists^\infty$; the modulo counting quantifiers $\exists^{(k,n)}$; and the Ramsey- or Magidor-Malitz quantifier Ram^n, first introduced by Magidor and Malitz [50].

Definition 2.1.13 ([49]). Let σ be some vocabulary. A collection of σ-structures Q which is closed under isomorphisms is called a generalised quantifier.

Let Q be some generalised quantifier. $\mathsf{FO}(Q)$ denotes the extension of first-order logic by this quantifier. The formulas of $\mathsf{FO}(Q)$ are defined using the formation rules of FO and the following rule. If $\varphi_i(x_1^i, x_2^i, \ldots x_{a_i}^i, \bar{y})$ is a formula in $\mathsf{FO}(Q)$ for each $1 \leq i \leq n$, then

$$Qx_1^1, \ldots x_{a_1}^1, x_1^2, \ldots, x_{a_2}^2, \ldots, x_1^n, \ldots, x_{a_n}^n (\varphi_1)(\varphi_2)\ldots(\varphi_n)$$

is a formula of $\mathsf{FO}(Q)$.

The semantics of this formula is defined as follows. Let $\mathfrak{A}$ be some structure and I some variable assignment. We set

$$R_i^I := \{\bar{a} \in A^{a_i} : \mathfrak{A}, I_{\bar{x}^i \mapsto \bar{a}} \models \varphi_i\} \text{ where}$$
$$\bar{x}^i = x_1^i, x_2^i, \ldots, x_{a_i}^i.$$

Now, we define the semantics of the quantifier by

$$\mathfrak{A}, I \models Qx_1^1, \ldots x_{a_1}^1, x_1^2, \ldots, x_{a_2}^2, \ldots, x_1^n, \ldots, x_{a_n}^n (\varphi_1)(\varphi_2)\ldots(\varphi_n) \text{ if } (A, R_1^I, R_2^I, \ldots, R_n^I) \in Q.$$

Example 2.1.14.

1. Let $\exists$ consist of all structures $\mathfrak{A} = (A, P)$ for P a unary predicate $\emptyset \neq P \subseteq A$. The generalised quantifier defined by $\exists$ is the usual existential quantifier.

2. Analogously, let $\forall$ consist of all structures $\mathfrak{A} = (A, P)$ for P the unary predicate $P = A$. The generalised quantifier defined by $\forall$ is the usual universal quantifier.

3. The infinite existential quantifier $\exists^\infty$ is defined by the collection of structures $\mathfrak{A} = (A, P)$ where $P \subseteq A$ is infinite. Some structure $\mathfrak{B}$ with some variable assignment I satisfies $\mathfrak{B}, I \models \exists^\infty x \varphi$ if there are infinitely many pairwise distinct elements $b_1, b_2, b_3 \cdots \in B$ such that $\mathfrak{B}, I_{x \to b_i} \models \varphi$ for all $i \in \mathbb{N}$.

4. The modulo counting quantifier $\exists^{(k,m)}$ is defined by the collection of structures $\mathfrak{A} = (A, P)$ where $P \subseteq A$ and $|P| = k \mod m$. Some structure $\mathfrak{B}$ with some variable assignment I satisfies $\mathfrak{B}, I \models \exists^{(k,m)} x \varphi$ if there are $k \mod m$ many pairwise distinct $b_i \in B$ such that $\mathfrak{B}, I_{x \to b_i} \models \varphi$.

5. Finally, we introduce the Ramsey quantifier of arity i. Let us say that a set $S \subseteq A^n$ contains an infinite box if there is an infinite subset $A' \subseteq A$ such that S contains all n-tuples of pairwise distinct elements from A'. Let Ram^n contain all structures $\mathfrak{A} = (A, P)$ where $P \subseteq A^n$ contains an infinite box. This means that $\mathsf{Ram}^1 = \exists^\infty$ and $\mathsf{Ram}^2 xy(Exy)$ is the formula that states that there is an infinite clique with respect to the binary relation E.

For a more detailed introduction to generalised quantifiers we refer the reader to the survey of Väänänen [62].

Definition 2.1.15. We denote by $\mathsf{FO}(\exists^{\mathrm{mod}})$ the extension of FO by modulo counting quantifiers. By $\mathsf{FO}((\mathsf{Ram}^n)_{n \in \mathbb{N}})$ we denote the extension of FO by Ramsey quantifiers. Analogously, $\mathsf{FO}(\exists^{\mathrm{mod}}, (\mathsf{Ram}^n)_{n \in \mathbb{N}})$ denotes the extension of FO by both types of quantifiers.

Remark 2.1.16. Note that $\exists^\infty$ is expressible in $\mathsf{FO}(\exists^{\mathrm{mod}})$: $\exists^\infty x(\varphi(x))$ is equivalent to $\neg(\exists^{0,2} x(\varphi(x)) \vee \exists^{1,2} x(\varphi(x)))$. Thus, we will use the quantifier $\exists^\infty$ as an abbreviation in $\mathsf{FO}(\exists^{\mathrm{mod}})$.

We come back to the generalised quantifiers $\exists^{\mathrm{mod}}$ and Ram^n in Section 3.4. We will show that first-order logic extended by these quantifiers is decidable on tree-automatic structures.

2.1.3 Basic Modal Logic and $L\mu$

Beside the classical logics like FO, MSO, and their extensions there is a another class of logics of great importance in the field of model checking: Basic modal logic and its extensions.

Almost a century ago, C. I. Lewis [48] introduced a modal operator for the first time. Since then, modal operators and modal logics have been studied intensively and found applications in very different fields like philosophy, mathematics, linguistics, computer science, and economic game theory. For an introduction to modal logics we refer the reader to the introductory chapters of [6].

This thesis is mainly concerned with model checking for classical logics. Nevertheless, we use some results concerning model checking for basic modal logic and modal μ-calculus. Thus, we will briefly recall the basic definitions and introduce our notation.

We fix a signature $\sigma = (E_1, E_2, \ldots, E_n, P_1, P_2, \ldots, P_m)$ of binary relations E_i and unary relations P_j called propositions.

Definition 2.1.17. Modal logic over the signature σ consists of the formulas generated by iterated use of the following rules.

1. True and False are modal formulas.
2. p_j is a modal formula for $1 \leq j \leq m$.
3. For φ, ψ modal formulas, their conjunction, disjunction and negation are modal formulas, i.e., $\varphi \wedge \psi, \varphi \vee \psi, \neg\varphi$ are modal formulas.
4. If φ is a modal formula, then $\langle E_i \rangle \varphi$ and $[E_i]\varphi$ are modal formulas.

In the modal terminology, one calls σ-structures Kripke structures. A Kripke structure $\mathfrak{A}$ together with a distinguished element $a \in \mathfrak{A}$ is called a pointed Kripke structure. The semantics of modal formulas is inductively defined according to the following rules.

1. $\mathfrak{A}, a \models \mathsf{True}$ and $\mathfrak{A}, a \not\models \mathsf{False}$ for all pointed Kripke structures $\mathfrak{A}, a$.
2. For $1 \leq j \leq m$, $\mathfrak{A}, a \models p_j$ if $a \in P_j$.
3. For formulas of the form $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\neg\varphi$ we use the standard interpretation of the logical connectives.
4. For $\varphi = \langle E_i \rangle \psi$, we set $\mathfrak{A}, a \models \varphi$ if there is some $a' \in A$ such that $(a, a') \in E_i$ and $\mathfrak{A}, a' \models \psi$. For $\varphi = [E_i]\psi$, we set $\mathfrak{A}, a \models \varphi$ if $\mathfrak{A}, a' \models \psi$ for all $a' \in A$ such that $(a, a') \in E_i$.

The expressive power of modal logic is strictly contained in that of first-order logic. This can be seen immediately when applying the so-called standard translation. The basic idea is that $\langle E_i \rangle \varphi$ is translated into a formula $\exists x(E_i y x \wedge \hat{\varphi}(x))$ where $\hat{\varphi}(x)$ is the standard translation of φ. $[E_i]\varphi$ is translated using the duality of $\langle E_i \rangle$ and $[E_i]$, i.e., replacing $[E_i]\varphi$ by $\neg\langle E_I \rangle \neg\varphi$. By clever reuse of variable names, it suffices to use 2 variables in this translation. The popularity of modal logic in model checking stems from its algorithmic tractability. Each satisfiable modal formula has a model which is a finite tree. Since trees are algorithmically well-behaved, one can develop very efficient algorithms for model checking of modal formulas. But this comes at the cost that the expressive power of modal logic is quite low. Thus, there have been many proposals how to extend the expressive power of modal logic while keeping the good algorithmic behaviour. One of the most powerful extensions of modal logic is the modal μ-calculus. This is the extension of modal logic by fixpoint operators analogously to the extension MLFP of first-order logic.

Definition 2.1.18. In order to define the modal μ-calculus, we fix set variables $X, Y, Z, \ldots$ The modal μ-calculus (denoted as $L\mu$) over the signature σ is the set of formulas generated by the following rules.

1. We may use all the rules that are used to generate the formulas of modal logic.
2. Additionally, X is a formula for each set variable X.

3. Finally, if X occurs only positively in an $L\mu$-formula φ, i.e., under the scope of an even number of negations, then $\mu X.\varphi(X)$ is a formula of $L\mu$.

Fix a σ-structure $\mathfrak{A}$, a variable assignment $I : V \to A$ and a point $a \in A$. We say $\mathfrak{A}, I, a \models X$ for $X \in V$ if $a \in I(X)$. For $\varphi = \mu X.\psi(X)$, we say $\mathfrak{A}, I, a \models \varphi$ if $a \in M^\psi$ for $M^\psi \subseteq A$ the least fixpoint of the operator that maps any subset $B \subseteq A$ to $\{a \in A : \mathfrak{A}, I_{X \to B}, a \models \psi(X)\}$. The rules for all other formulas are inherited from the semantics of modal logic in the obvious way.

$L\mu$ can be embedded into MLFP, i.e., for each $L\mu$-formula there is an equivalent MLFP formula. One extends the standard translation of modal logic to FO by the obvious translation of fixpoints in $L\mu$ to fixpoints in MLFP. $L\mu$ is a very powerful modal logic. Its expressive power encompasses many modal logics like linear time logic (LTL), computation tree logic (CTL) or CTL*.

2.1.4 Logical Interpretations

Logical interpretations are a formal framework to identify a structure that "lives" in another structure. This concept is used widely in mathematics. For instance, if one investigates the multiplicative group of a field, this is in fact the interpretation of a group within a field. This is one of the easiest examples of an interpretation. The following example from linear algebra illustrates a slightly more involved application of the concept of interpretations.

Example 2.1.19. Let $(V, +, \cdot)$ be some n-dimensional vectorspace. It is commonly known that the endomorphisms of V with concatenation $\circ$ and pointwise addition form a ring $\mathsf{End}(V)$. This ring is isomorphic to the ring of $n \times n$-dimensional matrices with addition and multiplication.

In this representation, $\mathsf{End}(V)$ is interpretable in V: The domain of this interpretation are all n^2-tuples from V where the k-th element of this tuple is considered as the entry in the $\left\lceil \frac{k}{n} \right\rceil$-th row and the $(k \mod n)$-th column. Addition and composition of the endomorphisms can then be reduced to computations on these n^2-tuples. Addition of two morphisms corresponds to pointwise addition of the n^2-tuples and composition can be reduced using the known formulas for matrix multiplication.

In logical terms, this is an n^2-dimensional first-order interpretation of the ring $\mathsf{End}(V)$ in the vectorspace V.

We call an interpretation logical if it is defined using formulas from some logic. The idea of using logical interpretations goes back to Tarski who used this concept to obtain undecidability results. Since then, the use of interpretations for decidability or undecidability proofs for the theories of certain structures has been a fruitful approach. For a detailed survey on logical interpretations we recommend the article of Blumensath et al. [9]. We briefly introduce our notation concerning interpretations and the important results that we are going to use.

Given some structure $\mathfrak{A}$, we can use formulas of some logic L to define a new structure $\mathfrak{B}$ from $\mathfrak{A}$. The idea is to obtain the domain of $\mathfrak{B}$ as an L definable subset of A^n. Then we define relations in this new structure via formulas in the signature of the old structure. If we obtain some structure $\mathfrak{B}$ in this way from another structure $\mathfrak{A}$, we say that $\mathfrak{B}$ is interpretable

in $\mathfrak{A}$. If $\mathfrak{B}$ is interpretable in $\mathfrak{A}$ this can be used to reduce the model checking problem on input $\mathfrak{B}$ to the model checking problem on $\mathfrak{A}$. In this thesis we will use FO-interpretations and one-dimensional MSO-interpretations. Let us start with introducing FO-interpretations formally.

Definition 2.1.20. Let $\sigma := (E_1, E_2, \ldots, E_n)$ and $\tau := (F_1, F_2, \ldots, F_m)$ be relational signatures. For $n \in \mathbb{N}$, an (n-dimensional-σ-τ) FO-interpretation is given by a tuple of FO(σ)-formulas $I := (\varphi, \psi_{F_1}, \psi_{F_2}, \ldots, \psi_{F_n})$ where φ has n free variables and each ψ_{F_i} has $r_i \cdot n$ free variables where r_i is the arity of F_i.

The interpretation I induces two maps: one from σ-structures to τ-structures and another from τ-formulas to σ-formulas.

Let Str_I be the map that maps a σ-structure $\mathfrak{A} := (A, E_1^{\mathfrak{A}}, \ldots, E_n^{\mathfrak{A}})$ to the τ-structure $\mathfrak{B} := (B, F_1^{\mathfrak{B}}, \ldots, F_m^{\mathfrak{B}})$ where

$$B := \left\{\bar{a} \in A^n : \mathfrak{A} \models \varphi(\bar{a})\right\} \text{ and}$$
$$F_i^{\mathfrak{B}} := \left\{(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_{r_i}) \in A^{n \cdot r_i} : \bar{a}_1, \bar{a}_2, \ldots, \bar{a}_{r_i} \in B \text{ and } \mathfrak{A} \models \psi_{F_i}(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_{r_i})\right\}.$$

Let Frm_I be the map that maps an FO(τ) formula α to the formula $\mathsf{Frm}_I(\alpha)$ which is obtained by the following rules:

- If $\alpha = F_i x^1 x^2 \ldots x^k$ for variable symbols x^i, then set

  $$\mathsf{Frm}_I(\alpha) := \psi_{F_i}(x_1^1, x_2^1, \ldots, x_n^1, x_1^2, \ldots, x_n^2, \ldots, x_1^k, \ldots x_n^k)$$

  where n is the dimension of I.

- Boolean connectives are preserved, i.e., if $\alpha = \alpha_1 \vee \alpha_2$ then

  $$\mathsf{Frm}_I(\alpha) = \mathsf{Frm}_I(\alpha_1) \vee \mathsf{Frm}_I(\alpha_2)$$

  and analogously for $\neg$ and $\wedge$.

- If $\alpha = \exists x \alpha_1$, then $\mathsf{Frm}_I(\alpha) := \exists x_1 \exists x_2 \ldots \exists x_n (\varphi(x_1, x_2, \ldots, x_n) \wedge \mathsf{Frm}_I(\alpha_1))$.
  If $\alpha = \forall x \alpha_1$, then $\mathsf{Frm}_I(\alpha) := \forall x_1 \forall x_2 \ldots \forall x_n (\varphi(x_1, x_2, \ldots, x_n) \rightarrow \mathsf{Frm}_I(\alpha_1))$.

The well-known connection between Str_I and Frm_I is given in the following lemma.

Lemma 2.1.21. Let I be an n-dimensional-σ-τ FO-interpretation, $\mathfrak{A}$ some σ-structure and φ some FO(τ) sentence. Then

$$\mathsf{Str}_I(\mathfrak{A}) \models \varphi \text{ iff } \mathfrak{A} \models \mathsf{Frm}_I(\varphi).$$

The proof is by induction on the structure of φ.

For FO model checking purposes, interpretations can be used as follows. Fix an interpretation I and two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ of structures. Assume that there is a computable function Str_I^{-1} that maps each $\mathfrak{A} \in \mathscr{C}_1$ to a structure $\mathfrak{B} \in \mathscr{C}_2$ such that $\mathsf{Str}_I(\mathfrak{B}) = \mathfrak{A}$. Then we can reduce the model checking problem for $\mathscr{C}_1$ to the model checking problem for $\mathscr{C}_2$. For $\mathfrak{A} \in \mathscr{C}_1$, we decide whether $\mathfrak{A} \models \varphi$ as follows: Firstly, we compute $\mathfrak{B} := \mathsf{Str}_I^{-1}(\mathfrak{A})$. Secondly, we solve the model checking problem $\mathfrak{B} \models \mathsf{Frm}_I(\varphi)$.

Similar to FO-interpretations we can define MSO-interpretations: simply replace the FO formulas in I by MSO formulas. Again, these can be used to reduce the MSO model checking on one class of structures to another class, but only if the interpretation is one-dimensional. If we use an n-dimensional MSO-interpretation for $n > 1$, the resulting transformation Frm_I translates MSO formulas into second-order formulas as quantification over unary relations is turned into quantification over n-ary relations. As long as we stick to one-dimensional MSO-interpretations, the transformation Frm_I turns an $\mathsf{MSO}(\tau)$ formula into an $\mathsf{MSO}(\sigma)$ formula and analogously to the previous lemma one obtains the following statement.

Lemma 2.1.22. Let I be a 1-dimensional-σ-τ MSO-interpretation, $\mathfrak{S}$ some σ-structure and φ some $\mathsf{MSO}(\tau)$ sentence. Then

$$\mathsf{Str}_I(\mathfrak{A}) \models \varphi \text{ iff } \mathfrak{A} \models \mathsf{Frm}_I(\varphi).$$

2.2 Grids and Trees

2.2.1 A Grid-Like Structure

Grid-like structures often play a crucial role in undecidability results for model checking problems. In this section, we introduce a certain grid-like structure, namely, the bidirectional half-grid. It is a version of the upper half of the $\mathbb{N} \times \mathbb{N}$ grid with an edge-relation for each direction, i.e., there are relations for the left, right, upward, and downward successor.

Definition 2.2.1. The half-grid is the structure $\mathfrak{H} := (H, \rightarrow, \leftarrow, \downarrow, \uparrow)$ where

$$\begin{aligned}
H &:= \{(i,j) \in \mathbb{N} \times \mathbb{N} : i \leq j\},\\
\rightarrow &:= \left\{\big((i,j),(k,l)\big) \in H^2 : i = k, j = l-1\right\},\\
\leftarrow &:= \left\{\big((i,j),(k,l)\big) \in H^2 : i = k, j = l+1\right\},\\
\downarrow &:= \left\{\big((i,j),(k,l)\big) \in H^2 : i = k-1, j = l\right\}, \text{ and}\\
\uparrow &:= \left\{\big((i,j),(k,l)\big) \in H^2 : i = k+1, j = l\right\},
\end{aligned}$$

See Figure 2.1 for a pictures of $\mathfrak{H}$.

Many MSO model checking results can be reduced to the question of tree-likeness or grid-likeness of the underlying graphs. On the one hand, if a class of structures consists only of structures that are similar to trees, e.g, structures with small tree-width, then the MSO model checking is effectively decidable. On the other hand, if a class contains a grid-like structure then the MSO model checking is undecidable. We do not want to go into the details what grid-likeness means exactly. But for our purposes, the crucial observation is that the upper half of a grid is, of course, grid-like whence $\mathfrak{H}$ has undecidable MSO-theory. In fact, we can even show undecidability of the $L\mu$-theory of this structure.

Lemma 2.2.2. $L\mu$ model checking is undecidable on the bidirectional half-grid $\mathfrak{H}$.

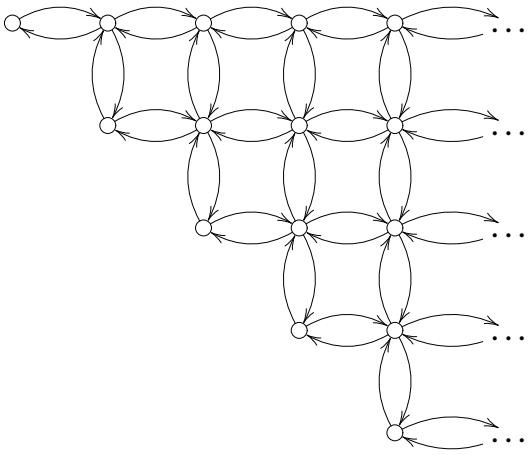


Figure 2.1.: The bidirectional half-grid.

Remark 2.2.3. Note that we consider $L\mu$ on the naked half-grid, i.e., without any additional propositions. Although this result is not very surprising for people familiar with $L\mu$, we have not found any proof of this lemma in the literature. The interested reader may find a detailed proof of this result in Appendix A where we reduce the halting problem for Turing machines to $L\mu$ model checking on $\mathfrak{H}$.

Since $L\mu$ may be seen as a fragment of MLFP and of MSO, the following corollaries follow immediately.

Corollary 2.2.4. MLFP and MSO are undecidable on $\mathfrak{H}$.

These results play a crucial role in Section 3.1.5, where we investigate the $L\mu$-theory of FO-interpretations of collapsible pushdown graphs.

2.2.2 Words and Trees

If Σ is a finite set (called alphabet) then Σ^* denotes the set of finite words over the alphabet Σ. For words $w_1, w_2 \in \Sigma^*$, we write $w_1 \leq w_2$ if w_1 is a prefix of w_2. We write $w_1 < w_2$ for $w_1 \leq w_2$ and $w_1 \neq w_2$. We denote by $w_1 \circ w_2$ (or simply $w_1 w_2$) the concatenation of w_1 and w_2. Furthermore, we write $w_1 \sqcap w_2$ for the greatest common prefix of w_1 and w_2. If $|w| = n$, we set w_{-i} for $0 \leq i \leq n$ to be the prefix of w of length $n - i$.

We now turn to trees. In this thesis we only consider binary trees. Most of the time we are concerned with finite trees, but in Section 3.4 we have to treat infinite trees as well. We use the word "tree" only for finite trees unless we explicitly say otherwise.

We call a set $D \subseteq \{0,1\}^*$ a tree domain, if D is prefix closed, i.e., for each $d \in D$ and $d' \in \Sigma^*$ we have $d' \in D$ if $d' \leq d$.

A Σ-labelled tree is a mapping $T : D \to \Sigma$ for D some tree domain. T is called finite, if D is finite; otherwise T is an infinite tree.

For $d \in D$ we denote the subtree rooted at d by $(T)_d$. This is the tree defined by $(T)_d(e) := T(de)$. For T_1, T_2 trees, we write $T_1 \preceq T_2$ if T_1 is an initial segment of T_2, i.e., if $\mathsf{dom}(T_1) \subseteq \mathsf{dom}(T_2)$ and $T_2 \restriction_{\mathsf{dom}(T_1)} = T_1$.

We denote the depth of the tree T by $\mathsf{dp}(T) := \max\{|t| : t \in \mathsf{dom}(T)\}$.

For T some tree with domain D, let D^+ denote the set of minimal elements of the complement of D, i.e.,

$$D^+ = \{e \in \{0,1\}^* \setminus D : \text{ all proper ancestors of } e \text{ are contained in } D\}.$$

In particular, note that $\emptyset^+ = \{\varepsilon\}$. Under the same assumptions, we write $D^\oplus$ for $D \cup D^+$. Note that $D^\oplus$ is the extension of the tree domain D by one layer.

Sometimes it is useful to define trees inductively by describing the subtrees rooted at 0 and 1. For this purpose we fix the following notation. Let $\hat{T}_0$ and $\hat{T}_1$ be Σ-labelled trees and $\sigma \in \Sigma$. Then we write $T := \hat{T}_0 \leftarrow \sigma \rightarrow \hat{T}_1$ for the Σ-labelled tree T with the following three properties

$$1.\ T(\varepsilon) = \sigma, \qquad 2.\ (T)_0 = \hat{T}_0, \text{ and} \qquad 3.\ (T)_1 = \hat{T}_1.$$

We call $(T)_0$ the left subtree of T and $(T)_1$ the right subtree of T.

We denote by Tree_Σ the set of all finite Σ-labelled trees and by $\mathsf{Tree}_\Sigma^\omega$ the set of all infinite Σ-labelled trees. We set $\mathsf{Tree}_\Sigma^{\leq\omega} := \mathsf{Tree}_\Sigma \cup \mathsf{Tree}_\Sigma^\omega$ to be the set of all finite or infinite Σ-labelled trees. We call the elements of Tree_Σ trees without referring to finiteness. This convention is useful because we use infinite trees only in Section 3.4. In that section we always clarify whether we talk about finite or infinite trees. The elements of $\mathsf{Tree}_\Sigma^\omega$ are called infinite trees

For $T \in \mathsf{Tree}_\Sigma^{\leq\omega}$ a finite or infinite tree, we write $T^\square$ for its lifting to the domain $\{0,1\}^*$ by padding with a special symbol $\square$, i.e.,

$$T^\square : \{0,1\} \to \Sigma \cup \{\square\}, T^\square(d) := \begin{cases} T(d) & \text{if } d \in \mathsf{dom}(T), \\ \square & \text{otherwise.} \end{cases}$$

Note that we consider a Σ-word w as a Σ-tree t_w with domain $\{0^i : 0 \leq i \leq |w|-1\}$ where $t_w(0^{i-1})$ is labelled by the i-th letter of w.

2.3 Generalised Pushdown Graphs

In this section we introduce the objects of our study, namely, the class of collapsible pushdown graphs and the class of nested pushdown trees. Both classes generalise the class of pushdown graphs. It will turn out that nested pushdown trees may be seen as a subclass of the collapsible pushdown graphs which has a nicer algorithmic behaviour than the class of all collapsible pushdown graphs. We start by recalling the well-known basics on pushdown systems. Then we present nested pushdown trees (NPT) and in the last part we present collapsible pushdown graphs (CPG).

2.3.1 Pushdown Graphs

A pushdown system is a finite automaton extended by a stack. These systems were first developed in formal language theory. Used as word- or tree acceptors, pushdown systems recognise exactly the context-free languages.

We are interested in the model checking properties of graphs generated by generalisations of pushdown system. The graph of a pushdown system is the graph of all reachable configurations where the edge-relation is induced by the transition relation of the pushdown system.

We briefly recall the definitions and present some classical results on pushdown systems.

Definition 2.3.1. A pushdown system is a tuple $\mathscr{S} = (Q, \Sigma, \Gamma, q_I, \Delta)$ satisfying the following conditions. Q is finite and it is called the set of states. It contains the initial state $q_I \in Q$. Σ is finite and is called the set of stack symbols. There is a special symbol $\bot \in \Sigma$ which is called the bottom-of-stack symbol. Γ is finite and it is called the input alphabet.

$$\Delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \mathrm{OP}$$

is the transition relation where

$$\mathrm{OP} := \{\mathrm{pop}_1, \mathrm{id}\} \cup \{\mathrm{push}_\sigma : \sigma \in \Sigma \setminus \{\bot\}\}.$$

The elements of OP are called stack operations. Each stack operation induces a function $\Sigma^+ \to \Sigma^*$ as follows.

- Let $w, w' \in \Sigma^+$ be words and $\sigma \in \Sigma$ a letter such that $w = w'\sigma$. Then $\mathrm{pop}_1(w) := w'$.
- id is the identity on Σ^+.
- Let $w \in \Sigma^+$. For each $\sigma \in \Sigma \setminus \{\bot\}$, we set $\mathrm{push}_\sigma(w) := w\sigma$.

A configuration of $\mathscr{S}$ is a tuple $(q, s) \in Q \times \Sigma^+$. Let $\delta = (q, \sigma, \gamma, q', \mathrm{op}) \in \Delta$. We call δ a γ-labelled transition. δ connects the configuration (q, s) with the configuration (q', s') if $s = \mathrm{op}(s')$. We set $(q, s) \vdash^\gamma (q', s')$ if there is a γ-labelled transition $\delta \in \Delta$ that connects (q, s) with (q', s').

We call $\vdash := \bigcup_{\gamma \in \Gamma} \vdash^\gamma$ the transition relation of $\mathscr{S}$

The configuration graph of $\mathscr{S}$ (also called the graph generated by $\mathscr{S}$) consists of all configurations that are reachable from the initial configuration $(q_0, \bot)$ via a path along $\vdash$.

Remark 2.3.2. We call a graph $\mathfrak{A}$ a pushdown graph if it is the graph generated by some pushdown system $\mathscr{S}$.

Without loss of generality, we assume that there is no transition of the form $(q, \bot, \gamma, q', \mathrm{pop}_1) \in \Delta$. This means that we never remove the bottom-of-stack symbol from the stack. Thus, we never have to deal with an empty stack.

Definition 2.3.3. Let $\mathscr{S}$ be a pushdown system. Let C be the set of configurations of $\mathscr{S}$ and $\vdash$ its transition relation A run ρ of $\mathscr{S}$ is a sequence of configurations that are connected by transitions, i.e., a sequence $c_0 \vdash^{\gamma_1} c_1 \vdash^{\gamma_2} c_2 \vdash^{\gamma_3} \ldots \vdash^{\gamma_n} c_n$.

We call $\rho(i) := c_i$ the configuration of ρ at position i. We call ρ a run from $\rho(0)$ to $\rho(n)$ and say that the length of ρ is $\ln(\rho) := n$.

We write $\mathrm{Runs}(\mathscr{S})$ for the set of all runs of $\mathscr{S}$.

For runs ρ_1, ρ_2 of a pushdown system we write $\rho_1 \preceq \rho_2$ for the fact that ρ_1 is an initial segment of ρ_2. We write $\rho_1 \prec \rho_2$ if ρ_1 is a proper initial segment, i.e., $\rho_1 \preceq \rho_2$ and $\ln(\rho_1) < \ln(\rho_2)$.

For runs $\rho = c_0 \vdash^{\gamma_1} c_1 \vdash^{\gamma_2} c_2 \vdash^{\gamma_3} \ldots \vdash^{\gamma_n} c_n$ and $\rho' = c'_0 \vdash^{\gamma'_1} c'_1 \vdash^{\gamma'_2} c'_2 \vdash^{\gamma'_3} \ldots \vdash^{\gamma'_m} c'_m$ where $c_n = c'_0$ we define

$$\pi := \rho \circ \rho' := c_0 \vdash^{\gamma_1} c_1 \vdash^{\gamma_2} c_2 \vdash^{\gamma_3} \ldots \vdash^{\gamma_n} c_n \vdash^{\gamma'_1} c'_1 \vdash^{\gamma'_2} c'_2 \vdash^{\gamma'_3} \ldots \vdash^{\gamma'_m} c'_m$$

and we call $\rho \circ \rho'$ the composition of ρ and ρ'. We also say that π decomposes as $\pi = \rho \circ \rho'$.

Remark 2.3.4. Note that a run does not necessarily start in the initial configuration. This convention is useful for the analysis of decompositions of runs because every restriction $\rho \restriction_{[i,j]}$ of a run ρ with $0 \leq i \leq j \leq \mathsf{ln}(\rho)$ is again a run.

In the following, we will often identify a run ρ of length n with a function from $\{0, 1, 2, \ldots, n\}$ to C that maps i to $\rho(i)$. This is a sloppy notation because there may be two different transitions $(q, \sigma, \gamma, q', \mathrm{op})$ and $(q, \sigma, \gamma', q', \mathrm{op})$ with $\gamma \neq \gamma'$ that give rise to two runs $(q, w) \vdash^{\gamma} (q', \mathrm{op}(w))$ and $(q, w) \vdash^{\gamma} (q', \mathrm{op}(w))$. In this case we would identify both runs with the same function f where $f(0) = (q, w)$ and $f(1) = (q', \mathrm{op}(w))$. For simplicity, we will always assume that the configurations of a run already determine the whole run.

Perhaps the most important theorem concerning pushdown systems and formal languages is the so-called uvxyz-theorem or pumping lemma of Bar-Hillel et al. [3]. It is a classical tool for proving that a language is not context-free. The uvxyz-theorem states the following. Given a context-free language L there is a natural number $n \in \mathbb{N}$ such that for all words from $w \in L$ of length at least n, there is a decomposition $w = uvxyz$ such that $uv^i xy^i z \in L$ for all $i \in \mathbb{N}$. There are elegant proofs of this theorem using context-free grammars.

In Chapter 3.2, we are interested in the runs of pushdown systems. Especially, we need to find a short run that is similar to a given long run. Thus, we are interested in a version of the uvxyz-theorem where we look at the run corresponding to a word w. We want to find a decomposition such that we can remove certain parts from the run and obtain a valid run (corresponding to some word uxz where $w = uvxyz$).

In this form, the proof of the lemma is slightly more complicated than in the version of context free languages. Thus, we start by giving an auxiliary lemma. It says that the run of a pushdown system does not depend on a prefix of the stack that is never read. A generalised version of this lemma for higher-order pushdown systems can be found in [8].

Definition 2.3.5. Let $w \in \Sigma^*$. Let ρ be a run of a pushdown system. We set $(q_i, w_i) := \rho(i)$ for all $i \in \mathsf{dom}(\rho)$. If $w \leq w_i$ for all $i \in \mathsf{dom}(\rho)$, we write $w \trianglelefteq \rho$ and say that ρ is prefixed by w.

Lemma 2.3.6. Let ρ be a run of some pushdown system $\mathscr{S}$ and let $w \in \Sigma^*$ be some word such that $w \trianglelefteq \rho$. For each $i \in \mathsf{dom}(\rho)$, let v_i denote the suffix of $\rho(i)$ such that $\rho(i) = (q_i, wv_i)$ for some state $q_i \in Q$.

If $w' \in \Sigma^*$ ends with the same letter as w then the function

$$\begin{aligned} \rho[w/w'] : \mathsf{dom}(\rho) &\to Q \times \Sigma^* \\ \rho[w/w'](i) &:= (q_i, w'v_i) \end{aligned}$$

is a run of $\mathscr{S}$.

The proof of this lemma is straightforward: just observe that any stack operation commutes with the prefix replacement. The claim follows by induction on $\mathsf{dom}(\rho)$. We are now prepared to state the uvxyz-theorem in a version for pushdown systems.

Lemma 2.3.7 ([3]). Let $\mathscr{S}$ be some pushdown system. There is a constant $n \in \mathbb{N}$ such that for every run ρ of length greater than n that starts in the initial configuration at least one of the following holds.

1. There is a decomposition $\rho = \rho_1 \circ \rho_2 \circ \rho_3$, words $w_1 < w_2$, and a state $q \in Q$ such that $\rho_2(0) = (q, w_1)$, $\rho_3(0) = (q, w_2)$, $\ln(\rho_2) \geq 1$, and $\rho' := \rho_1 \circ \rho_3[w_2/w_1]$ is a run of $\mathscr{S}$.

2. There is a decomposition $\rho = \rho_1 \circ \rho_2 \circ \rho_3 \circ \rho_4 \circ \rho_5$, words $w_1 < w_2$ with equal topmost letter and states $q, q' \in Q$ such that $\rho_2(0) = (q, w_1)$, $\rho_3(0) = (q, w_2)$, $\rho_4(0) = (q', w_2)$, $\rho_5(0) = (q', w_1)$, $\ln(\rho_2) + \ln(\rho_4) \geq 1$, and $\rho' := \rho_1 \circ \rho_3[w_2/w_1] \circ \rho_5$ is a run of $\mathscr{S}$.

3. There is a decomposition $\rho = \rho_1 \circ \rho_2 \circ \rho_3$ with $\ln(\rho_2) \geq 1$ such that $\rho' := \rho_1 \circ \rho_3$ is a run of $\mathscr{S}$.

Proof. We assume that $n \in \mathbb{N}$ is some large natural number (what large means can be obtained from the proof). Let ρ be some run such that $m := \ln(\rho) > n$. In order to prove this claim, we look for configurations in the run that share the same state and share the same topmost element on their stack. There are the following cases.

1. The run ends with a large stack: assume that ρ ends in a stack w with $|w| > |\Sigma \times Q|$. For each $i \leq |w|$, let w_i be the prefix of w of length i. Let $n_i \leq \ln(\rho)$ be maximal such that the stack at $\rho(n_i)$ is w_i. Set $(q_i, w_i) := \rho(n_i)$. By pigeon-hole principle there are $j < k < \ln(\rho)$ such that $q_j = q_k$ and $\mathrm{top}_1(w_j) = \mathrm{top}_1(w_k)$. Since n_k is maximal, the run

$$\rho' := \rho\restriction_{[0,n_j]} \circ \rho\restriction_{[n_k,\ln(r)]}[w_k/w_j]$$

   is well defined and satisfies the lemma.

2. The run passes a large stack but ends in a small one: assume that ρ ends in some word of length at most $|\Sigma \times Q|$. Furthermore, assume that ρ passes a word of length greater than $|\Sigma \times Q| + |Q \times Q \times \Sigma|$.

   Let $i_{\max} \in \mathrm{dom}(\rho)$ be a position such that the word at $\rho(i_{\max})$ has maximal length in ρ.

   By assumption, it follows that for $\rho(i_{\max}) =: (q_{\max}, w_{\max})$,

$$|w_{\max}| > |\Sigma \times Q| + |Q \times Q \times \Sigma|.$$

   For each $i \leq |w_{\max}|$, let w_i be the prefix of $w_{\max}$ of length i. For each

$$|\Sigma \times Q| \leq i \leq |\Sigma \times Q| + |Q \times Q \times \Sigma|,$$

   let $n_i \leq i_{\max}$ be maximal such that $\rho(n_i) = (q_i, w_i)$ for some $q_i \in Q$. Analogously, let $m_i \geq i_{\max}$ be minimal such that $\rho(m_i + 1) = (\hat{q}_i, \mathrm{pop}_1(w_i))$ for some $\hat{q}_i \in Q$.

   Note that w_i is the stack at $\rho(m_i)$ and $w_i \trianglelefteq \rho\restriction_{[n_i,m_i]}$ due to the definition of n_i and m_i.

   By the pigeon-hole principle, there are $|\Sigma \times Q| \leq j < k \leq |\Sigma \times Q| + |Q \times Q \times \Sigma|$ such that $q_j = q_k, \hat{q}_j = \hat{q}_k$ and $\mathrm{top}_1(w_j) = \mathrm{top}_1(w_k)$.

   Then the run $\rho' := \rho\restriction_{[0,n_j]} \circ \rho\restriction_{[n_k,m_k]}[w_k/w_j] \circ \rho\restriction_{[m_j,\ln(\rho)]}$ satisfies the lemma.

3. The run never visits a large stack, i.e., a stack of size greater than $|Q \times \Sigma| + |Q \times Q \times \Sigma|$. Since there are only finitely many stacks of size smaller than this bound, in a long run of this form there is a configuration which is visited twice and the subrun in between may be omitted. □

Pushdown graphs form a class of finitely represented infinite graphs with good model checking properties. Almost fifty years ago, Buchi [14] showed that the reachability problem on pushdown graphs is decidable. This result was notably extended by Muller and Schupp in the 80's as follows.

Theorem 2.3.8 ([53]). The MSO-theory of every pushdown graph is decidable.

This result was important for the development of software verification because of the following fact. A pushdown graph naturally arises as the abstraction of some programme using (first-order recursive) functions. Given a programme, one can design a pushdown system that simulates the behaviour of this programme. Every run of the pushdown system corresponds to a possible execution of the programme. Here, the state of the pushdown system stores the programme counter. This means that the state of the pushdown system stores the line number that is executed by the programme in this step. If a function call occurs, the pushdown system does the following. It writes the programme counter onto the stack, and the new state is the first line of the function which is called. When this function eventually terminates, the programme counter is restored by reading the stack. While the programme counter is restored, the topmost element of the stack is deleted.

Using this reduction, many problems occurring in software verification can be reduced to model checking on pushdown graphs. But this approach has a severe limitation: in the language of the pushdown graph, MSO cannot be used to define a function return corresponding to a function call. Defining a return that corresponds to a certain call is equivalent to defining a subrun of the pushdown system that starts at this function call and forms a well-bracketed word (where we interpret push operations as opening brackets and pop operations as closing brackets). But it is well known that MSO cannot define the language of well-bracketed words (the so-called Dyck-languages).

Thus, if one wants to verify properties of a programme that involves a comparison of the situation just before a function call with the situation exactly after the return of the function, one cannot reduce this problem to a model checking problem on pushdown graphs.

In the next section we present nested pushdown trees. These generalise trees generated by pushdown systems in such a way that pairs of corresponding calls and returns become definable even in first-order logic. Therefore, nested pushdown trees are suitable abstractions for programmes if one wants to verify properties involving the pairs of corresponding function calls and returns.

2.3.2 Nested Pushdown Trees

Alur et al. [2] proposed the study of the model checking problem on nested pushdown trees. A nested pushdown tree is the tree generated by a pushdown system where the pairs of corresponding push and pop operations are marked by a new relation $\hookrightarrow$. This new relation is called jump-relation. We stress that due to this new relation, a nested pushdown tree is no tree.

Definition 2.3.9. Let $\mathscr{S} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a pushdown system. Then the nested pushdown tree generated by $\mathscr{S}$ is

$$\mathsf{NPT}(\mathscr{S}) := (R, (\vdash^\gamma)_{\gamma\in\Gamma}, \hookrightarrow)$$

where $(R, (\vdash^\gamma)_{\gamma\in\Gamma})$ is the unfolding of the configuration graph of $\mathscr{S}$. R is the set of all runs of $\mathscr{S}$ starting at the configuration $(q_0, \bot)$. For two runs $\rho_1, \rho_2 \in R$, we have $\rho_1 \vdash^\gamma \rho_2$ if ρ_2 extends ρ_1 by exactly one γ-labelled transition. The binary relation $\hookrightarrow$ is called jump-relation and is defined as follows: let $\rho_1, \rho_2 \in R$ with $\mathsf{ln}(\rho_i) = n_i$ and $\rho_1(n_1) = (q, w) \in Q \times \Sigma^*$. Then $\rho_1 \hookrightarrow \rho_2$ if ρ_1 is an initial segment of ρ_2, $\rho_2(n_2) = (q', w)$ for some $q' \in Q$ and w is a proper prefix of all stacks between $\rho_1(n_1)$ and $\rho_2(n_2)$, i.e., $w < \rho_2(i)$ for all $n_1 < i < n_2$.

Alur et al. proved the following results concerning the model checking properties of the class of nested pushdown trees.

Theorem 2.3.10 ([2]). The $L\mu$ model checking problem for nested pushdown trees is in EXPTIME.

Lemma 2.3.11 ([2]). The MSO model checking problem for nested pushdown trees in undecidable.

Proof. Let $\mathscr{S} := (\{0,1\}, \{a, \bot\}, \{A, P\}, (0, \bot), \Delta)$ with

$$\Delta = \{(0, \bot, A, 0, \mathrm{push}_a), (0, a, A, 0, \mathrm{push}_a), (0, a, P, 1, \mathrm{pop}_1), (1, a, P, 1, \mathrm{pop}_1)\}.$$

Figure 2.2 shows the nested pushdown tree generated by $\mathscr{S}$. We now show that the bidi-

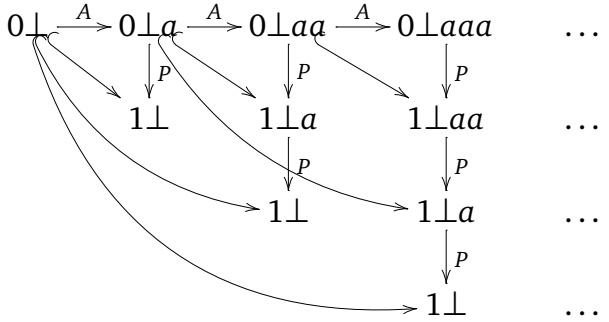


Figure 2.2.: Example of a nested pushdown tree.

rectional halfgrid $\mathfrak{H}$ is MSO-interpretable in this graph. Application of Lemmas 2.2.2 and 2.1.22 then directly yields the claim.

As an abbreviation, we use the binary relation REACH_{P^*} which holds for configurations (c_1, c_2) if and only if c_2 is reachable from c_1 by a P-labelled path. This predicate is clearly MSO-definable and in the structure $\mathsf{NPT}(\mathscr{S})$ it describes reachability along the columns. Now, we define the next-column relation by

$$\varphi_{nc}(x, y) := \exists z_1, z_2(\mathsf{REACH}_{P^*} z_1 x \wedge \mathsf{REACH}_{P^*} z_2 y \wedge z_1 \vdash^A z_2).$$

Similarly, we can define a next-diagonal relation by

$$\varphi_{nd}(x, y) := \exists z_1, z_2(z_1 \hookrightarrow x \wedge z_2 \hookrightarrow y \wedge z_1 \vdash^A z_2).$$

We conclude that $\varphi_{nc} \wedge \varphi_{nd}(x,y)$ holds if and only if y is the right neighbour of x in the half-grid. Thus, $\downarrow$ coincides with $\vdash^{\mathrm{pop}_1}$ and $x \to y$ is defined by $\varphi_{nc}(x,y) \wedge \varphi_{nd}(x,y)$. Switching the roles of x and y, we can also define $\leftarrow$ and $\uparrow$.

This completes the interpretation of $\mathfrak{H}$ in $\mathsf{NPT}(\mathscr{S})$. Lemmas 2.2.2 and 2.1.22 then yield the claim. □

Remark 2.3.12. Even though MSO model checking for nested pushdown trees is undecidable, nested pushdown trees form an interesting class for software verification. Many interesting properties of programmes are expressible in $L\mu$. Moreover, the jump-relation allows to use $L\mu$ in order to express properties concerning corresponding push and pop operations. Such properties are not expressible when using MSO on pushdown graphs.

We have seen that MSO is undecidable on nested pushdown trees while $L\mu$ model checking is decidable. This difference concerning decidability of MSO and $L\mu$ model checking turns nested pushdown trees into an interesting class of structures from a model theoretic point of view. Natural classes of graphs tend to have either decidable MSO and $L\mu$ model checking or undecidable MSO and $L\mu$ model checking. Beside the class of nested pushdown trees we only know of one other natural class that does not follow this general rule: the class of collapsible pushdown graphs. In Section 3.2 we will show that nested pushdown trees and collapsible pushdown graphs are closely related via a simple FO-interpretation. This relationship between nested pushdown trees and collapsible pushdown graphs (of level 2) will motivate our definition of the hierarchy of higher-order nested pushdown trees in Section 3.3 in analogy to the hierarchy of collapsible pushdown graphs. But before we come to this generalisation of the concept of a nested pushdown tree, let us introduce collapsible pushdown graphs.

2.3.3 Collapsible Pushdown Graphs

Before we introduce Collapsible pushdown graphs (CPG) in detail, we fix some notation. Then, we informally explain collapsible pushdown systems. Afterwards, we formally introduce these systems and the graphs generated by them. We conclude this section with some basic results on runs of collapsible pushdown systems. In Chapter 3.1 we will then investigate FO model checking on collapsible pushdown graphs.

For some alphabet Σ, we inductively define Σ^{*n} and Σ^{+n} for all $n \in \mathbb{N} \setminus \{0\}$ as follows. We set $\Sigma^{*1} := \Sigma^*$, i.e., Σ^{*1} is the set of all finite words over alphabet Σ. Then we set $\Sigma^{*(n+1)} := (\Sigma^{*n})^*$. Analogously, we write $\Sigma^{+1} := \Sigma^+$ for the set of all nonempty finite words over alphabet Σ and we set $\Sigma^{+(n+1)} := (\Sigma^{+n})^+$. Each element of Σ^{*n} is called an n-word. Stacks of a level n collapsible pushdown system are certain nonempty n-words over a special alphabet.

Let us fix a word $s \in \Sigma^{*(n+1)}$ of level $n+1$. s consists of an ordered list $w_1, w_2, \ldots, w_m$ of n-words, i.e., $w_1, w_2, \ldots, w_m \in \Sigma^{*n}$. If we want to state this list of n-words explicitly, we separate them by colons writing $s = w_1 : w_2 : \cdots : w_m$. By $|s|$ we denote the number of n-words s consists of, i.e., $|s| = m$. We say $|s|$ is the width of s. We also use the notion of the height of an $(n+1)$-word. The height of s is $\mathrm{hgt}(s) := \max\{|w_i| : 1 \leq i \leq m\}$ which is the width of the widest n-word occurring in s.

Let s' be another word of level $n+1$ such that $s' = w'_1 : w'_2 : \ldots w'_l \in \Sigma^{*(n+1)}$. We write $s : s'$ for the concatenation $w_1 : w_2 : \ldots w_m : w'_1 : w'_2 : \cdots : w'_l$.

If $s \in \Sigma^{*n}$, we denote by $[s]$ the $n+1$ word that only consists of a list of one n word which is s. We regularly omit the brackets if no confusion arises.

Let Σ be some finite alphabet. A level n stack s is an n-word where each letter carries a link to some substack. Each link has a certain level $1 \leq i \leq n$. A level i link points to some $(i-1)$-word of the topmost level i stack of s. Now, we first define the initial level n stack; afterwards we describe some stack operations that are used to generate all level n stacks from the initial one.

Definition 2.3.13. Let Σ be some finite alphabet with a distinguished bottom-of-stack symbol $\perp \in \Sigma$. The initial stack of level l over Σ is inductively defined as follows. The initial level 1 stack is $\perp_1 := \perp$. For the higher levels, we set $\perp_n := [\perp_{n-1}]$ to be the initial stack of level n.

We informally describe the operations that can be applied to a level n stack. There are the following stack operations:

- The push operation of level 1, denoted by $\mathsf{push}_{\sigma,k}$ for $\sigma \in \Sigma$ and $1 \leq k \leq n$, writes the symbol σ onto the topmost level 1 stack and attaches a link of level k. This link points to the next to last entry of the topmost level k stack.
- For $2 \leq i \leq n$, the push operation of level i is denoted by clone_i. It duplicates the topmost entry of the topmost level i stack. The links are preserved by clone_i in the following sense. Let s be some stack. Let a be a letter in the topmost level i stack of s. Assume that a has a link of level j. Let a' be the copy of a in $\mathsf{clone}_i(s)$. Then the link of a' points to the unique level $j-1$ stack in the topmost level j stack of $\mathsf{clone}_i(s)$ that is a clone of the $j-1$ stack to which the link of a points. This means that for $j \geq i$, a and a' carry links to the same stack. For $j < i$, the link of a' points to the clone of the stack to which the link of a points.
- The level i pop operation pop_i for $1 \leq i \leq n$ removes the topmost entry of the topmost level i stack. Note that the pop_1 operation corresponds to the ordinary pop in a pushdown system that just removes the topmost symbol from the stack.
- The last operation is $\mathsf{collapse}$. The result of $\mathsf{collapse}$ is determined by the link attached to the topmost letter of the stack. If we apply collapse to a stack s where the link level of the topmost letter is i, then $\mathsf{collapse}$ replaces the topmost level i stack of s by the level i stack to which the link points. Note that the application of a collapse is equivalent to the application of a sequence of pop_i operations where the link of the topmost letter controls how long this sequence is.

In the following, we formally introduce collapsible pushdown stacks and the stack operations. We represent such a stack of letters with links as n-words over the alphabet $(\Sigma \cup (\Sigma \times \{2,\ldots,n\} \times \mathbb{N}))^{+n}$. We consider elements from Σ as elements with a link of level 1 and elements (σ,l,k) as letters with a link of level l. In the latter case, the third component specifies the width of the substack to which the link points. For letters with link of level 1, the position of this letter within the stack already determines the stack to which the link points. Thus, we need not explicitly specify the link in this case.

Remark 2.3.14. Other equivalent definitions, for instance in [27], use a different way of storing the links: they also store symbols (σ,i,n) on the stack, but here n denotes the number of

pop_i transitions that are equivalent to performing the collapse operation at a stack with topmost element (σ, i, n). The disadvantage of that approach is that the clone_i operation cannot copy stacks. Instead, it can only copy the symbols stored in the topmost stack and has to alter the links in the new copy. A clone of level i must replace all links (σ, i, n) by $(\sigma, i, n+1)$ in order to preserve the links stored in the stack.

Before we give a formal definition of the stack operations, we introduce some auxiliary functions.

Definition 2.3.15. For $s = w_1 : w_2 : \cdots : w_n \in (\Sigma \cup (\Sigma \times \{2, \dots l\} \times \mathbb{N}))^{+l}$, we define the following auxiliary functions:

- For $1 \leq k \leq l$, the topmost level $k-1$ word of s is $\mathsf{top}_k(s) := \begin{cases} w_n & \text{if } k = l, \\ \mathsf{top}_k(w_n) & \text{otherwise.} \end{cases}$
- For $\mathsf{top}_1(s) = (\sigma, i, j) \in \Sigma \times \{2, 3, \dots, l\} \times \mathbb{N}$, we define the topmost symbol $\mathsf{Sym}(s) := \sigma$, the collapse level of the topmost element $\mathsf{CLvl}(s) := i$, and the collapse link of the topmost element $\mathsf{CLnk}(s) := j$.

  For $\mathsf{top}_1(s) = \sigma \in \Sigma$, we define the topmost symbol $\mathsf{Sym}(s) := \sigma$, the collapse level of the topmost element $\mathsf{CLvl}(s) := 1$, and the collapse link of the topmost element $\mathsf{CLnk}(s) := |\mathsf{top}_2(s)| - 1$.
- For $m \in \mathbb{N}$, we define $\mathrm{p}_{\sigma,k,m}(s) := \begin{cases} s(\sigma, k, m) & \text{if } l = 1, \\ w_1 : w_2 : \cdots : w_{n-1} : \mathrm{p}_{\sigma,k,m}(w_n) & \text{otherwise.} \end{cases}$

These auxiliary functions are useful for the formalisation of the stack operations.

Definition 2.3.16. For $s = w_1 : w_2 : \cdots : w_n \in (\Sigma \cup (\Sigma \times \{2, 3, \dots l\} \times \mathbb{N}))^{+l}$, for $\sigma \in \Sigma \setminus \{\bot\}$, for $1 \leq k \leq l$ and for $2 \leq j \leq l$, we define the stack operations

$$\mathsf{clone}_j(s) := \begin{cases} w_1 : w_2 : \cdots : w_{n-1} : w_n : w_n & \text{if } j = l \geq 2, \\ w_1 : w_2 : \cdots : w_{n-1} : \mathsf{clone}_j(w_n) & \text{otherwise.} \end{cases}$$

$$\mathsf{push}_{\sigma,k}(s) := \begin{cases} s\sigma & \text{if } k = l = 1, \\ \mathrm{p}_{\sigma,k,n-1}(s) & \text{if } k = l \geq 2, \\ w_1 : w_2 : \cdots : w_{n-1} : \mathsf{push}_{\sigma,k}(w_n) & \text{otherwise.} \end{cases}$$

$$\mathsf{pop}_k(s) := \begin{cases} w_1 : w_2 : \cdots : w_{n-1} : \mathsf{pop}_k(w_n) & \text{if } k < l, \\ w_1 : w_2 : \cdots : w_{n-1} & \text{if } k = l, n > 1, \\ \text{undefined} & \text{otherwise, i.e., } k = l, n = 1. \end{cases}$$

$$\mathsf{collapse}(s) := \begin{cases} w_1 : w_2 : \cdots : w_m & \text{if } \mathsf{CLvl}(s) = l, \mathsf{CLnk}(s) = m > 0, \\ w_1 : w_2 : \cdots : w_{n-1} : \mathsf{collapse}(w_n) & \text{if } \mathsf{CLvl}(s) < l, \\ \text{undefined} & \text{if } \mathsf{CLnk}(s) = 0. \end{cases}$$

The set of level l operations is

$$\mathsf{OP}_l := \{(\mathsf{push}_{\sigma,k})_{\sigma \in \Sigma, k \leq l}, (\mathsf{clone}_k)_{2 \leq k \leq l}, (\mathsf{pop}_k)_{1 \leq k \leq l}, \mathsf{collapse}\}.$$

The set of level l stacks, $\mathsf{Stacks}_l(\Sigma)$, is the smallest set that contains $\bot_l$ and is closed under application of operations from OP_l.

Remark 2.3.17. It is sometimes convenient to assume that the identity

$$\mathsf{id} : \mathsf{Stacks}_l(\Sigma) \to \mathsf{Stacks}_l(\Sigma)$$

is also a stack operation. Whenever this assumption is useful, we assume id to be a stack operation.

We illustrate the definition of the stack operations with the following example.

Example 2.3.18. We start with the level 3 stack $s_0 := [\bot] : [\bot : \bot]$. We have

$$\begin{aligned}
\mathsf{push}_{a,2}(s_0) &= [\bot] : [\bot : \bot(a,2,1)] =: s_1 \\
\mathsf{push}_{b,3}(s_1) &= [\bot] : [\bot : \bot(a,2,1)(b,3,1)] =: s_2 \\
\mathsf{clone}_3(s_2) &= [\bot] : [\bot : \bot(a,2,1)(b,3,1)] : [\bot : \bot(a,2,1)(b,3,1)] =: s_3 \\
\mathsf{clone}_2(s_3) &= s_2 : [\bot : \bot(a,2,1)(b,3,1) : \bot(a,2,1)(b,3,1)] =: s_4 \\
\mathsf{collapse}(s_4) &= [\bot] \\
\mathsf{pop}_1(s_4) &= s_2 : [\bot : \bot(a,2,1)(b,3,1) : \bot(a,2,1)] =: s_5 \\
\mathsf{collapse}(s_5) &= s_2 : [\bot] = [\bot] : [\bot : \bot(a,2,1)(b,3,1)] : [\bot]\,.
\end{aligned}$$

Note that $\mathsf{collapse}$ and pop_k operations are only allowed if the resulting stack is nonempty. This avoids the special treatment of empty stacks. Furthermore, any $\mathsf{collapse}$ that works on level 1 is equivalent to one pop_1 operation: level 1 links always point to the preceding letter because there is no clone_1 operation. Furthermore, every $\mathsf{collapse}$ that works on a level $i \geq 2$ is equivalent to a sequence of pop_i operations.

Let us now define the substack relation on collapsible pushdown stacks. It is the natural generalisation of the prefix order on words.

Definition 2.3.19. Let $s, s' \in \mathsf{Stacks}_l(\Sigma)$. We say that s' is a substack of s if there are $n_i \in \mathbb{N}$ for $1 \leq i \leq l$ such that $s' = \mathsf{pop}_1^{n_1}(\mathsf{pop}_2^{n_2}(\ldots(\mathsf{pop}_l^{n_l}(s))))$. We write $s' \leq s$ if s' is a substack of s.

Now, it is time to formally define collapsible pushdown systems. These are defined completely analogously to pushdown systems but using a level l stack and all the level l stack operations.

Definition 2.3.20. A collapsible pushdown system of level l (l-CPS) is a tuple

$$\mathscr{S} = (Q, \Sigma, \Gamma, \Delta, q_0)$$

where Q is a finite set of states, Σ a finite stack alphabet with a distinguished bottom-of-stack symbol $\bot \in \Sigma$, Γ a finite input alphabet, $q_0 \in Q$ the initial state, and

$$\Delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \mathrm{OP}_l$$

the transition relation.

A level l configuration is a pair (q, s) where $q \in Q$ and $s \in \mathsf{Stacks}_l(\Sigma)$. For $q_1, q_2 \in Q$ and $s, t \in \mathsf{Stacks}_l(\Sigma)$ we define a γ-labelled transition $(q_1, s) \vdash^\gamma (q_2, t)$ if there is a $(q_1, \sigma, \gamma, q_2, op) \in \Delta$ such that $\mathsf{op}(s) = t$ and $\mathsf{Sym}(s) = \sigma$.

We call $\vdash := \bigcup_{\gamma \in \Gamma} \vdash^\gamma$ the transition relation of $\mathscr{S}$. We set $C(\mathscr{S})$ to be the set of all configurations that are reachable from $(q_0, \bot_l)$ via $\vdash$ and call $C(\mathscr{S})$ the set of reachable or valid configurations. The collapsible pushdown graph (CPG) generated by $\mathscr{S}$ is

$$\mathsf{CPG}(\mathscr{S}) := \left(C(\mathscr{S}), (C(\mathscr{S})^2 \cap \vdash^\gamma)_{\gamma \in \Gamma}\right)$$

Remark 2.3.21.

- Note that the transitions of a collapsible pushdown system only depend on the state and the topmost symbol, but not on the topmost collapse level and collapse link. The latter are only used to handle the result of a collapse operation.

- In the following, we always assume that the label of each transition carries information about the stack operation and the state that is reached, i.e., we assume that there is a map $f : \Gamma \to Q \times \mathsf{OP}$ such that for each transition $(q, \sigma, \gamma, q', \mathsf{op}) \in \Delta$ we have $f(\gamma) = (q', \mathsf{op})$. It is obvious that each collapsible pushdown system can be transformed into one that satisfies this assumption: use $\Gamma \times Q \times \mathsf{OP}$ as new input alphabet; then $\vdash^{\gamma} = \bigcup_{q \in Q, \mathsf{op} \in \mathsf{OP}} \vdash^{(\gamma,q,\mathsf{op})}$. In this sense, we will write $\vdash^{q,\mathsf{op}} := \bigcup_{f(\gamma)=(q,\mathsf{op})} \vdash^{\gamma}$ and also $\vdash^{q} := \bigcup_{\mathsf{op} \in \mathsf{OP}} \vdash^{q,\mathsf{op}}$ and $\vdash^{\mathsf{op}} := \bigcup_{q \in Q} \vdash^{q,\mathsf{op}}$.

- An higher-order pushdown system is a collapsible pushdown system that does not use the collapse operation.

  To be more precise, we call a collapsible pushdown system with transition relation Δ an higher-order pushdown system if

$$\Delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \left(\mathsf{OP}_l \setminus \left(\{\mathsf{collapse}\} \cup \{\mathsf{push}_{\sigma,i} : i \geq 2\}\right)\right),$$

  i.e., if it does not use the collapse operation and the links of level i for all $i > 1$.

Example 2.3.22. The following example of a collapsible pushdown graph $\mathfrak{G}$ of level 2 is taken from [27]. Let $Q := \{0, 1, 2\}, \Sigma := \{\bot, a\}$, $\Gamma := \{\mathsf{Cl}, A, A', P, \mathsf{Co}\}$. Δ is given by $(0, -, \mathsf{Cl}, 1, \mathsf{clone}_2)$, $(1, -, A, 0, \mathsf{push}_{a,2})$, $(1, -, A', 2, \mathsf{push}_{a,2})$, $(2, a, P, 2, \mathsf{pop}_1)$, and $(2, a, \mathsf{Co}, 0, \mathsf{collapse})$, where $-$ denotes any letter from Σ.

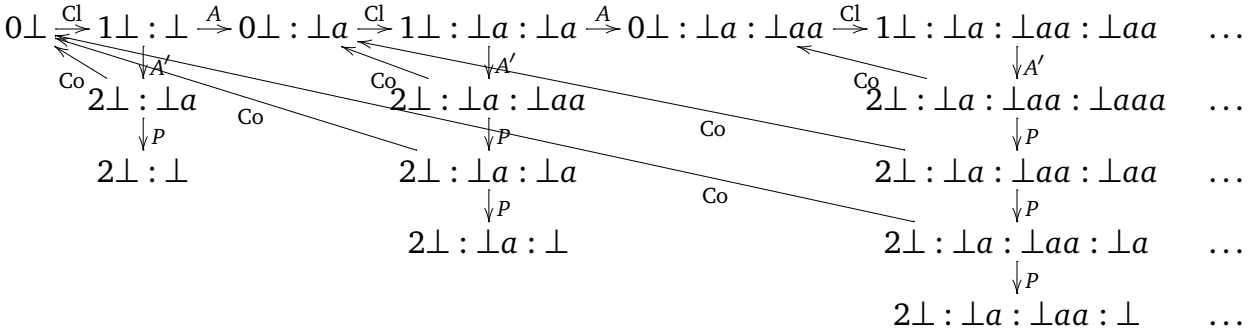


Figure 2.3.: Example of the 2-CPG $\mathfrak{G}$ (the level 2 links of the letters a are omitted due to space restrictions).

The next two theorems summarise the known results concerning model checking on collapsible pushdown graphs.

Theorem 2.3.23 ([27]). There is a collapsible pushdown graph of level 2 with undecidable MSO model checking.

Proof. The graph from figure 2.3 is an example. Note that the graph from figure 2.2 is clearly FO-interpretable in this graph; one merely has to reverse the collapse-edges in order to obtain the jump-edges and to omit every second node in the topmost line. Hence, the corresponding MSO undecidability result from theorem 2.3.11 applies also to this collapsible pushdown graph. □

Theorem 2.3.24 ([27]). $L\mu$ model checking on level n collapsible pushdown graphs is n-EXPTIME complete.

We briefly sketch the proof idea for this theorem. The proof uses parity-games on collapsible pushdown graphs. It is commonly known that $L\mu$ model checking and the calculation of winning regions in a parity-game are equivalent. In order to solve parity-games on a collapsible pushdown graph of level $l+1$, Hague et al. reduce this problem to another parity game on a collapsible pushdown graph of level l. Their proof consists of two steps.

1. First, they prove that for each collapsible pushdown graph there is another one of the same level that is rank aware. A level l collapsible pushdown graph is rank aware if it "knows" at each configuration (q,s) with $\mathsf{CLvl}(s) = l$ the minimal rank (or priority) that was visited since the last occurrence of the stack $\mathsf{collapse}(s)$. One can show that for each parity game on a collapsible pushdown graph, one can construct a parity game on a rank aware collapsible pushdown graph such that every winning strategy in this new game can be transformed into a winning strategy for the original game.

2. In the second step, Hague et al. reduce the problem of solving a parity game on a rank aware collapsible pushdown graph of level $l+1$ to the problem of solving a game on a graph of level l. The basic idea is to simulate only the topmost level l stack of the level $l+1$ graph and to handle the attempt to use a clone_{l+1} at a certain stack s in the following way: if one player would perform a clone_{l+1} operation in the original game, Verifier has to make a certain claim about her winning strategy in the original game. According to her winning strategy, for each priority i, there is a set of states Q_i such that whenever the game returns to s and the minimal priority between the clone_{l+1} operation and this new occurrence of the stack s is i, then the stack s is visited again in one of the states from Q_i. Now, in the new game, Verifier moves to a state representing the set $(Q_i)_{i\in P}$ where P is the finite set of priorities. Falsifier now has two choices. Either he believes Verifier or he does not believe that Verifier's claim is correct.

   If he believes her, he chooses one of the $i \in P$ and a $q \in Q_i$. The new game continues in (q,s) after visiting an auxiliary state of priority i.

   Otherwise, the new game continues with the stack $\mathsf{top}_{l-1}(s)$ and Falsifier has to show that there is some position where he could use a pop_l- or $\mathsf{collapse}$ operation (in the original game) and return to some state q and stack s such that $q \notin Q_i$ for i the least priority visited since Verifier had made this claim. At this point, rank awareness comes into play. At each position where a pop_2- or $\mathsf{collapse}$ operation may be performed, rank-awareness allows to determine the minimal priority i since Verifier had made her claim. Thus, due to rank-awareness, we can check whether Falsifier managed to reach a position (q,s) where $q \notin Q_i$. In this case, Falsifier wins the game. If $q \in Q_i$ then the game ends and Verifier wins.

Using this reduction $l-1$ times, one derives a parity game on a level 1 pushdown graph such that a strategy on this game can be used in order to compute a strategy in the original parity game. Walukiewicz [63] showed the solvability of parity games on level 1 pushdown graphs. Now, the decidability of $L\mu$ model checking on collapsible pushdown graphs follows by induction on the level of the graph.

Since MSO model checking is undecidable for collapsible pushdown graphs, it is interesting to investigate model checking for fragments of MSO. What is the largest fragment such that model checking on collapsible pushdown graphs is decidable? In Chapter 3.1, we make a first step towards an answer to this question. We prove the decidability of the first-order model checking on level 2 collapsible pushdown graphs extended by $L\mu$-definable predicates.

Now, we come to the notion of a run of a collapsible pushdown system. This definition is completely analogous to the definition of a run of a pushdown system (cf. Definition 2.3.3).

Definition 2.3.25. Let $\mathcal{S}$ be a collapsible pushdown system. A run ρ of $\mathcal{S}$ is a sequence of configurations that are connected by transitions, i.e., a sequence

$$c_0 \vdash^{\gamma_1} c_1 \vdash^{\gamma_2} c_2 \vdash^{\gamma_3} \cdots \vdash^{\gamma_n} c_n.$$

Remark 2.3.26. As in the case of pushdown systems, we identify a run ρ of length n with the function that maps a number i to the configuration occurring just after the i-th transition in ρ for each $0 \leq i \leq n$, i.e., $\rho(i)$ denotes the configuration after the i-th transition of ρ and especially $\rho(0)$ is the first configuration of ρ.

The final part of this section consists of some basic results concerning runs of collapsible pushdown systems of level 2. We focus on level 2 because all of our main results only treat pushdown systems of level 2.

First, we come to the question whether certain runs can create links to certain stacks. Consider some configuration (q,s) of a level 2 collapsible pushdown system. If $|s| = n$ then a $\mathsf{push}_{\sigma,2}$ transition applied to (q,s) creates a letter with a link to the substack of width $n-1$. Thus, links to the substack of width $n-1$ in some word above the n-th one are always created by a clone_2 operation. A direct consequence of this fact is the following lemma.

Lemma 2.3.27. Let s be some level 2 stack with $\mathsf{top}_1(s) = (\sigma, 2, k)$. Let ρ be a run of a pushdown system of level 2 that starts with stack s, that passes $\mathsf{pop}_1(s)$, and that ends in s. If $k < |s| - 1$ then ρ passes $\mathsf{pop}_2(s)$.

The proof is left to the reader. Later we often use the contraposition of this statement. We use the fact that a certain run to s does not visit $\mathsf{pop}_2(s)$ and conclude that it cannot visit $\mathsf{pop}_1(s)$.

The next result deals with the decision problem for configurations: given a collapsible pushdown system $\mathcal{S}$, and a configuration (q,s), is $(q,s) \in \mathsf{CPG}(\mathcal{S})$? We can solve this problem using the decidability of $L\mu$ model checking on collapsible pushdown systems.

In the following we reduce the decision problem for configurations for a level 2 collapsible pushdown system $\mathcal{S}$ to the $L\mu$ model checking on a variation of $\mathcal{S}$. The proof is based on the idea that a stack is uniquely determined by its top element and the information which substacks can be reached via $\mathsf{collapse}$ and pop_i.

We can compute a variant $\mathcal{S}^{(q,s)}$ of a given CPS $\mathcal{S}$ such that $\mathcal{S}^{(q,s)}$ satisfies a certain $L\mu$ formula if and only if (q,s) is a configuration of the graph generated by $\mathcal{S}$. The new

pushdown system is the extension of $\mathscr{S}$ by a testing device for the configuration (q,s). Let us describe this testing device.

Assume that we want to define a testing device for the configuration (q,s). Furthermore, assume that for each configuration (q',s') where s' is a proper substack of s, there already is a testing device for configuration (q',s'). The testing device for (q,s) works as follows.

Whenever the system is in some configuration $(q,\hat{s})$, it switches to $(q_s,\hat{s})$ where q_s is a new "testing state". In q_s, the system checks whether $\mathsf{top}_1(\hat{s}) = \mathsf{top}_1(s)$. If this is the case, then the following happens. Let $\hat{s}'$ be the stack obtained from $\hat{s}$ by removing the topmost element and let s' be the stack obtained from s by removing the topmost element. Now, we start the testing device for the substack s' on the stack $\hat{s}'$. If this testing device returns that $\hat{s}'$ is s', then $\hat{s} = s$ and the new testing device was started in (q,s).

For each configuration (q,s), there is an $L\mu$ formula such that this formula is satisfied at some configuration of $\mathscr{S}^{(q,s)}$ if and only if this configuration is (q,s).

Before we go into the details of this proof, we recall the terminology concerning $L\mu$ on collapsible pushdown graphs. The binary relations on such a graph are labelled by symbols from the input alphabet Γ and we use expressions as $\langle\gamma\rangle\varphi$ for the formula saying "there is a γ-labelled edge leading to a node where φ holds". As an abbreviation we use $\Diamond\varphi$ for the formula saying "there is an arbitrary labelled edge leading to a node where φ holds", i.e., as an abbreviation for $\bigvee_{\gamma\in\Gamma}\langle\gamma\rangle\varphi$.

Lemma 2.3.28. Given some CPS $\mathscr{S} = (Q,\Sigma,\Gamma,\Delta,q_0)$ of level 2, some $q \in Q$ and some stack s, it is decidable whether (q,s) is a reachable configuration of $\mathscr{S}$, i.e., whether (q,s) is a vertex of $\mathsf{CPG}(\mathscr{S})$.

Proof. For $q \in Q$ and s a stack, we define a system $\mathscr{S}^{(q,s)}$ and a formula $\psi_{(q,s)} \in L\mu$, such that

$$\mathscr{S}^{(q,s)}, (q_0,\bot_2) \models \psi_{(q,s)} \quad \text{iff} \quad (q,s) \in \mathsf{CPG}(\mathscr{S}).$$

We set

$$\begin{aligned}
&\mathscr{S}^{(q,s)} := (Q',\Sigma,\Gamma',\Delta^{(q,s)},q_0) \text{ with}\\
&Q' := Q \cup \{q_t : t \leq s\} \cup \{q_\emptyset\}, \text{ and}\\
&\Gamma' := \Gamma \cup \{(q_t,\mathrm{op}) : t \leq s, \mathrm{op} \in \mathsf{OP}\} \cup (\{q_\emptyset\} \times \mathsf{OP}),
\end{aligned}$$

where q_t is a new state for every substack t of the stack s we are looking for and $q_\emptyset$ is used for checking that certain operations can or cannot be performed on a configuration.

In the following we define $\Delta^{(q,s)} \supseteq \Delta$ by induction on the size of s such that $\Delta^{(q_t,t)} \subseteq \Delta^{(q,s)}$ for all proper substacks $t < s$.

1. For $s = \bot_2$ we set

$$\Delta^{(q,s)} := \Delta \cup \{(q,\bot,(q_\emptyset,\mathsf{clone}_2),q_\emptyset,\mathsf{clone}_2),(q,\bot,(q_\emptyset,\mathsf{pop}_2),q_\emptyset,\mathsf{pop}_2)\}.$$

   Additionally, we set $\varphi_{(q_0,s)} := \langle q_\emptyset,\mathsf{clone}_2\rangle\mathsf{True} \wedge [q_\emptyset,\mathsf{pop}_2]\mathsf{False}$. Note that the first part of this formula is satisfied in $\mathscr{S}^{(q,s)}$ at some configuration c if the state is q and the topmost symbol is $\bot$. At such a configuration c, the second part can only be satisfied if no pop_2 is possible, i.e., if the width of the stack is 1.

2. Assume that $|s| > 1$ and $\mathsf{Sym}(s) = \bot$ for some stack s. Then we set $t = \mathsf{pop}_2(s)$ and

$$\Delta^{(q,s)} := \Delta^{(q_t,t)} \cup \{(q, \bot, (q_t, \mathsf{pop}_2), q_t, \mathsf{pop}_2)\}$$

and $\varphi_{(q,s)} := \langle q_t, \mathsf{pop}_2 \rangle \varphi_{(q_t,t)}$.

3. The next case is $\mathsf{Sym}(s) \neq \bot$ and $\mathsf{CLnk}(s) = 0$. Then we set $t := \mathsf{pop}_1(s)$ and

$$\begin{aligned}\Delta^{(q,s)} :=& \Delta^{(q_t,t)} \\ &\cup \{(q, \mathsf{Sym}(s), (q_t, \mathsf{pop}_1), q_t, \mathsf{pop}_1), (q, \mathsf{Sym}(s), (q_\emptyset, \mathsf{collapse}), q_\emptyset, \mathsf{collapse})\}\end{aligned}$$

and

$$\varphi_{(q,s)} := \langle q_t, \mathsf{pop}_1 \rangle \varphi_{(q_t,t)} \wedge [q_\emptyset, \mathsf{collapse}]\mathsf{False}.$$

4. In all other cases we set $t := \mathsf{pop}_1(s)$ and $u := \mathsf{collapse}(s)$. We set

$$\begin{aligned}\Delta^{(q,s)} :=& \Delta^{(q_t,t)} \cup \{(q, \mathsf{Sym}(s), (q_t, \mathsf{pop}_1), q_t, \mathsf{pop}_1)\} \\ &\cup \{(q, \mathsf{Sym}(s), (q_U, \mathsf{collapse}), q_u, \mathsf{collapse})\}\end{aligned}$$

and

$$\varphi_{(q,s)} := \langle q_t, \mathsf{pop}_1 \rangle \varphi_{(q_t,t)} \wedge \langle q_u, \mathsf{collapse} \rangle \varphi_{(q_u,u)}.$$

We show by induction that for all $q \in Q$ and stacks s

$$\mathsf{CPG}(\mathscr{S}^{(q,s)}), c \models \varphi_{(q,s)} \text{ iff } c = (q,s).$$

The initial stack $s = \bot_2 = [\bot]$ is characterised by the facts that the top symbol of the stack is the bottom-of-stack symbol and that pop_2 is undefined. The first conjunct of

$$\varphi_{(q_0,s)} = \langle q_\emptyset, \mathsf{clone}_2 \rangle \mathsf{True} \wedge [q_\emptyset, \mathsf{pop}_2]\mathsf{False}$$

is only satisfied if the top symbol is $\bot$ and the second conjunct is satisfied if and only if pop_2 is undefined. Thus, $\varphi_{(q,\bot_2)}$ and $\mathscr{S}^{(q,\bot_2)}$ satisfy our claim.

For the induction step, note that $\mathsf{collapse}$ is defined if and only if the collapse link of the topmost symbol is not 0. If $\mathsf{collapse}$ is defined for some stack s and $u := \mathsf{collapse}(s), t = \mathsf{pop}_1(s)$ then $\Delta^{(q_u,u)} \subseteq \Delta^{(q_t,t)} \subseteq \Delta^{(q_s,s)}$ because $u \leq t$. With these observations the induction step is straightforward by case distinction on the topmost symbol of s and on the fact whether $\mathsf{collapse}(s)$ is defined. Let (q,s) be some configuration and let l be the minimal level such that $\mathsf{pop}_l(s)$ is defined. On the graph generated by $\mathscr{S}^{(q,s)}$, the formula $\varphi_{(q,s)}$ asserts that this pop_l operation is defined and, by induction hypothesis, results in the stack $\mathsf{pop}_l(s)$. The analogous argument applies to the result of a collapse operation if the operation is defined on s. If it is undefined, i.e., $\mathsf{CLnk}(s) = 0$ then the formula $\varphi_{(q,s)}$ asserts that the collapse is undefined.

Now, we set $\psi_{(q,s)} := \mu Z.(\Diamond Z \vee \varphi_{(q,s)})$ which is just the formula asserting reachability of some point where $\varphi_{(q,s)}$ holds. Thus,

$$(q,s) \in \mathsf{CPG}(\mathscr{S}) \qquad \text{iff} \qquad \mathsf{CPG}(\mathscr{S}^{(q,s)}), (q_0, \bot_2) \models \psi_{(q,s)} \ .$$

The latter problem is decidable due to Theorem 2.3.24. □

Remark 2.3.29. This lemma extends to systems of higher level. But in the case of higher levels, the proof needs some further preparation. The underlying problem that one faces on higher levels is the following. Consider the level 3 stacks $s_2 := [[\bot(\sigma,2,0)]]$ and $s_3 := [[\bot(\sigma,3,0)]]$. For any sequence of stack operations, the result of the application of this sequence to s_2 is defined if and only if its application to s_3 is defined. Furthermore, the resulting stacks are identical except for the replacement of level 2 links of value 0 by level 3 links of value 0.

Thus, our approach cannot distinguish between s_2 and s_3.

In order to make our approach work, we have to transform $\mathscr{S}$ into a new pushdown system $\mathscr{S}'$ over a new alphabet Σ' which is level aware. Level awareness is defined as follows. There is a mapping $f : \Sigma' \to \{1,2,3,\ldots,l\}$ such that for each stack generated by $\mathscr{S}'$, $\mathsf{Sym}(s) = \sigma$ implies that $\mathsf{CLvl}(s) = f(\sigma)$. This system can be obtained by replacing Σ by $\Sigma \times \{1,2,3,\ldots,l\}$ and by using $\mathsf{push}_{(\sigma,k),k}$ instead of $\mathsf{push}_{\sigma,k}$.

Then we can apply the generalisation of the approach of level 2 to this new system and solve the decision problem for configurations.

We now turn to a quantitative version of the decision problem for configurations. We want to compute how many runs to a given configuration exist up to a given threshold $k \in \mathbb{N}$. In the next lemma we show that this question can be reduced to the decision problem for configurations.

Lemma 2.3.30. There is an algorithm solving the following problem. Given a level 2 collapsible pushdown system $\mathscr{S} = (Q,\Sigma,\Gamma,q_0,\Delta)$, a state $q \in Q$, a stack $s \in \mathsf{Stacks}_2(\Sigma)$ and a threshold $k \in \mathbb{N}$, how many runs from the initial configuration to (q,s) exist up to threshold k?

Proof. First of all, by Lemma 2.3.28, it is decidable whether (q,s) is a node of $\mathsf{Grph}(\mathscr{S})$. If this is the case then there is at least one run of the desired form. Otherwise there are 0 runs of this form.

Assume that $(q,s) \in \mathsf{Grph}(\mathscr{S})$. Since the runs of $\mathscr{S}$ are recursively enumerable, we can compute the length-lexicographically smallest run ρ_1 to (q,s).[3]

In the following we show how to decide whether there is a second run of the desired form. For this purpose, let $l := \mathsf{ln}(\rho_1)$ and let δ_i be the transition between $\rho_1(i)$ and $\rho_1(i+1)$ for all $0 \leq i < l$. Furthermore, let q_i be the state at $\rho_1(i)$. Now, we construct a new pushdown system $\tilde{\mathscr{S}} := (\tilde{Q},\Sigma,\Gamma,\tilde{q}_0,\tilde{\Delta})$ where

$$\tilde{Q} := Q \cup \{\tilde{q}_0,\tilde{q}_1,\ldots,\tilde{q}_l\}$$

for new states $\tilde{q}_0,\ldots,\tilde{q}_l$ and

$$\begin{aligned}\tilde{\Delta} :=& \Delta \cup \{(\tilde{q}_i,\sigma,\gamma,\tilde{q}_{i+1},\mathrm{op}) : \delta_i = (q,\sigma,\gamma,q',\mathrm{op})\} \\ &\cup \{(\tilde{q}_i,\sigma,\gamma,q',\mathrm{op}) : (q_i,\sigma,\gamma,q',\mathrm{op}) \in \Delta \setminus \{\delta_i\},\}\,.\end{aligned}$$

This system copies the behaviour of every initial segment of ρ_1 and stays within the new states. As soon as it simulates one of the transitions of Δ that do not extend the run to another initial segment of ρ_1, it changes to the correct original state in Q. From this point

[3] We assume that the transition relation of $\mathscr{S}$ is a totally ordered set.

on, the system behaves exactly like $\mathscr{S}$. Note that the run corresponding to ρ_1 in $\tilde{\mathscr{S}}$ ends in configuration $(\tilde{q}_l, s)$ (for s the final stack of ρ_1). Hence, the corresponding run is no witness for the reachability of (q,s) in the new system. Thus, if $\mathsf{CPG}(\tilde{\mathscr{S}})$ contains (q,s), then there are two different runs in $\mathscr{S}$ from the initial configuration to (q,s).

Repeating this construction up to k times, we compute the runs to (q,s) up to threshold k. □

Remark 2.3.31. We have no elementary bound on the complexity of this algorithm. This is due to the fact that we cannot derive a polynomial bound on the length of the run ρ_1. Hence, the size of the pushdown system under consideration may increase too much in each iteration. Since we use the $L\mu$ model checking algorithm on each of the pushdown systems we construct, the resulting algorithm is doubly exponential in the size of the largest pushdown system that we construct.

In the last part of this section, we recall a lemma of Blumensath from [8] concerning the substitution of prefixes of stacks. The original lemma was stated for higher-order pushdown systems (without collapse) of arbitrary level. Here, we only recall the result for level 2 pushdown systems and we present a straightforward adaption to the case of level 2 collapsible pushdown systems. We start by defining a prefix relation on stacks. Note that this relation does not coincide with the substack relation.

Definition 2.3.32. For some level 2 stack t and some substack $s \leq t$ we say that s is a prefix of t and write $s \trianglelefteq t$, if there are $n \leq m \in \mathbb{N}$ such that $s = w_1 : w_2 : \cdots : w_{n-1} : w_n$ and $t = w_1 : w_2 \cdots : w_{n-1} : v_n : v_{n+1} : \cdots : v_m$ such that $w_n \leq v_j$ for all $n \leq j \leq m$.

For some run ρ, we write $s \trianglelefteq \rho$ if $s \trianglelefteq \rho(i)$ for all $i \in \mathsf{dom}(\rho)$.

Remark 2.3.33. Note that $s \trianglelefteq t$ obtains if s and t agree on the first $|s| - 1$ words and the last word of s is a prefix of all other words of t. Especially, s has to be a substack of t ands $|s| \leq |t|$.

Now, we introduce a function that replaces the prefix of some stack by some other.

Definition 2.3.34. Let s, t, u be level 2 stacks such that $s \trianglelefteq t$. Assume that

$$\begin{aligned} s &= w_1 : w_2 : \cdots : w_{n-1} : w_n, \\ t &= w_1 : w_2 \cdots : w_{n-1} : v_n : v_{n+1} : \cdots : v_m, \text{ and} \\ u &= x_1 : x_2 : \cdots : x_p \end{aligned}$$

for numbers $n, m, p \in \mathbb{N}$ such that $n \leq m$. For each $n \leq i \leq m$, let $\hat{v}_i$ be the unique word such that $v_i = w_n \circ \hat{v}_i$. We define

$$t[s/u] := x_1 : x_2 : \cdots : x_{p-1} : (x_p \circ \hat{v}_n) : (x_p \circ \hat{v}_{n+1}) : \cdots : (x_p \circ \hat{v}_m)$$

and call $t[s/u]$ the stack obtained from t by replacing the prefix s by u.

Remark 2.3.35. Note that for t some stack with level 2 links, the resulting object $t[s/u]$ may be no stack. Take for example the stacks

$$\begin{aligned} t &= \bot(a,2,0) : \bot(a,2,0), \\ s &= \bot(a,2,0) : \bot \text{ and} \\ u &= \bot : \bot. \end{aligned}$$

Then $t[s/u] = \bot : \bot(a,2,0)$. This list of words cannot be created from the initial stack using the stack operation because an element $(a,2,0)$ in the second word has to be a clone of some element in the first one. But $(a,2,0)$ does not occur in the first word.

If $t \in \Sigma^{+2}$, i.e., if t does not contain links of level 2, then $t[s/u]$ is always a stack. Thus, the prefix replacement for stacks of higher-order pushdown systems always results in a well-defined stack while prefix replacement for stacks of collapsible pushdown systems may result in objects that are not stacks.

In the following we study the compatibility of prefix replacement with the stack operations.

Lemma 2.3.36. Let s,t be stacks such that $s \trianglelefteq t$. Let op be some operation. If $s \not\trianglelefteq \mathsf{op}(t)$, then one of the following holds:

1. $\mathsf{op}(t) = \mathsf{pop}_2^k(s)$ for some $k \in \mathbb{N}$ or
2. $\mathsf{op}(t) = \mathsf{pop}_1(s)$, $\mathsf{top}_2(t) = \mathsf{top}_2(s)$ and $\mathsf{top}_2(\mathsf{op}(t)) = \mathsf{pop}_1(\mathsf{top}_2(s))$.

Proof. If op is clone_2 or $\mathsf{push}_{\sigma,i}$ for some $\sigma \in \Sigma$ and $i \in \{1,2\}$, then $s \trianglelefteq t$ implies $s \trianglelefteq \mathsf{op}(t)$.

If $\mathsf{op} = \mathsf{pop}_2$ and $s \not\trianglelefteq \mathsf{op}(t)$ then $|t| = |s|$ and $\mathsf{op}(t) = \mathsf{pop}_2(s)$.

If $\mathsf{op} = \mathsf{pop}_1$, $s \trianglelefteq t$ and $s \not\trianglelefteq \mathsf{op}(t)$ implies that $\mathsf{top}_2(s) \leq \mathsf{top}_2(t)$ but $\mathsf{top}_2(s) \not\leq \mathsf{top}_2(\mathsf{op}(t))$. One immediately concludes that $\mathsf{top}_2(t) = \mathsf{top}_2(s)$ and $\mathsf{top}_2(\mathsf{op}(t)) = \mathsf{pop}_1(\mathsf{top}_2(s))$.

If $\mathsf{op} = \mathsf{collapse}$, we have to distinguish two cases. If $\mathsf{CLvl}(t) = 1$, then we apply the same argument as in the case of $\mathsf{op} = \mathsf{pop}_1$. Otherwise, $\mathsf{op}(t) = \mathsf{pop}_2^m(t)$ for some $m \in \mathbb{N}$ and one reasons analogously to the case of $\mathsf{op} = \mathsf{pop}_2$. □

Lemma 2.3.37. Let s,t be stacks such that $s \trianglelefteq t$ and $|t| > |s|$. Then it holds that $s \trianglelefteq \mathsf{pop}_2(t)$.

Proof. Just note that $t = \mathsf{pop}_2(s) : t'$ for t' a stack where each word is prefixed by $\mathsf{top}_2(s)$. Furthermore, $|t| > |s|$ implies that $|t'| \geq 2$. Hence, $\mathsf{pop}_2(t) = \mathsf{pop}_2(s) : t'$ for t' a stack of width at least 1 where each word is $\mathsf{top}_2(s)$ prefixed by $\mathsf{top}_2(s)$. Thus, $s \trianglelefteq t$ holds. □

Blumensath showed the following important compatibility of prefix replacement and stack operations in the case of level 2 pushdown systems (without collapse!).

Lemma 2.3.38 ([8]). Let ρ be a run of some pushdown system $\mathscr{S}$ of level 2 and let $s,u \in \Sigma^{+2}$ be stacks such that the following conditions are satisfied:

1. $s \trianglelefteq \rho$,
2. $\mathsf{top}_2(s) < \mathsf{top}_2(\rho(i))$ for all $i < \mathsf{ln}(\rho)$ or $\mathsf{Sym}(u) = \mathsf{Sym}(s)$.

Under these conditions, the function $\rho[s/u]$ defined by $\rho[s/u](i) := \rho(i)[s/u]$ is a run of $\mathscr{S}$.

Proof (sketch). One proves this lemma by induction on $\mathsf{dom}(\rho)$. The transitions performed in ρ can be carried over one by one to the transitions of $\rho[s/u]$. □

Now, we present an adaption of this idea to collapsible pushdown systems.

Lemma 2.3.39. Let ρ be a run of some collapsible pushdown system $\mathscr{S}$ of level 2 and let s and u be stacks such that the following conditions are satisfied:

1. $s \trianglelefteq \rho$,

2. $\mathrm{top}_2(s) < \mathrm{top}_2(\rho(i))$ for all $i < \ln(\rho)$ or $\mathrm{top}_1(u) = \mathrm{top}_1(s)$,

3. $|s| = |u|$, and

4. for $\rho(0) = (q, t)$, $t[s/u]$ is a stack.

Under these conditions the function $\rho[s/u]$ defined by $\rho[s/u](i) := \rho(i)[s/u]$ is a run of $\mathscr{S}$.

Proof. The proof is again by induction on $\mathrm{dom}(\rho)$. For all operations, except for collapse, the proof of this lemma is analogous to the proof of the previous lemma. For each such operation op occurring at position i in ρ one shows that $\rho(i+1)[s/u] = \mathrm{op}(\rho(i)[s/u])$.

For the collapse operation, assume that there is a position i such that

$$\rho(i+1) = \mathrm{collapse}(\rho(i))$$

and such that $\rho(i)[s/u]$ is defined. Due to condition 2, the topmost symbol and the collapse level of $\rho(i)$ and $\rho(i)[s/u]$ agree. Thus, if the collapse level is 1, then the collapse acts on both configurations like a pop_1. In this case, the compatibility of this collapse with the prefix replacement follows from the proof of the case of pop_1. Otherwise, the collapse level of the topmost element of both stacks is 2. In this case the collapse links of the topmost elements also agree by definition. Furthermore, due to $|s| = |u|$ the width of $\rho(i)$ and $\rho(i)[s/u]$ agrees. Hence, there is some $k \in \mathbb{N}$ such that the collapse applied to both configurations results in $\mathrm{pop}_2^k(\rho(i))$ and $\mathrm{pop}_2^k(\rho(i))[s/u]$, respectively. Thus, the reduction to the iterated use of the case of pop_2 proves the claim. □

2.4 Technical Results on the Structure of Collapsible Pushdown Graphs

In this section, we develop the technical background for our main results that are presented in Sections 3.1 and 3.3.

As in the end of the previous section, this section is only concerned with collapsible pushdown systems of level 2. Hence, if we write collapsible pushdown system, we always mean one of level 2.

The overall goal of this section is the following: finite automata can be used to determine how many[4] runs from the initial configuration to some configuration (q, s) exist. In order to prove this result, we introduce three notions: returns, loops, and generalised milestones[5]. We motivate these notions from the last to the first.

Let s and s' be stacks. We call s' a generalised milestone of s if every run from the initial configuration to a configuration with stack s has to pass s' at some intermediate step. Thus, it follows directly from this definition that the reachability of a certain stack from the initial configuration decomposes into the analysis of the reachability of milestones from other milestones of this stack. We will see that every run to s passes all the milestones of s'

[4] For the rest of this section, the question "how many?" is meant up to a certain threshold $k \in \mathbb{N}$, i.e., "how many runs to (q, s) exist" stands for "given a threshold $k \in \mathbb{N}$, how many runs to (q, s) exist up to threshold k?".

[5] The term "generalised" refers to the fact that this notion is a generalisation of the notion "milestone" which we introduced in [35].

in a certain order. Thus, the question "how many runs to s exist?" can be reduced to the question "how many runs from one milestone of s to the next exist?".

A closer analysis of this decomposition shows that the run from one milestone to the next is always a loop followed by exactly one transition. A loop is a run from some configuration (q,s) to some configuration (q',s) not passing a substack of $\mathsf{pop}_2(s)$. This means that a run starts and ends with the same stack s and it does not "look into" the content of $\mathsf{pop}_2(s)$.

Using this result, the question "how many runs to (q,s) exist?" can be reduced to the question "how many loops of each generalised milestone of s exist?".

In order to show that a finite automaton can answer the last question, we introduce the notion of a return. A run ρ is called return if it is a run from some stack s to the stack $\mathsf{pop}_2(s)$ that satisfies the following conditions:

1. before the last position, no substack of $\mathsf{pop}_2(s)$ is passed, and
2. the collapse links of level 2 stored in $\mathsf{top}_2(s)$ are not used by ρ.

It turns out that returns naturally appear as subruns of loops. In the following we first introduce generalised milestones and develop their theory. Then we define loops and returns and show their connection to generalised milestones in Section 2.4.2. In Section 2.4.3 we develop the theory of counting returns. Finally, we develop the analogous theory of loops in Section 2.4.4.

2.4.1 Milestones and Loops

Recall that $w \sqcap v$ denotes the greatest common prefix of the words w and v (cf. Section 2.2.2). We start with a formal definition of generalised milestones. Afterwards, we show that this definition fits the informal description given before.

Definition 2.4.1. Let $s = w_1 : w_2 : \cdots : w_k$ be a stack. We call a stack m a generalised milestone of s if m is of the form

$$\begin{aligned} &m = w_1 : w_2 : \cdots : w_i : v_{i+1} \text{ where } 0 \leq i < k, \\ &w_i \sqcap w_{i+1} \leq v_{i+1} \text{ and} \\ &v_{i+1} \leq w_i \text{ or } v_{i+1} \leq w_{i+1}. \end{aligned}$$

We denote by $\mathsf{GMS}(s)$ the set of all generalised milestones of s.

For a generalised milestone m of s, we call m a milestone of s if m is a substack of s. We write $\mathsf{MS}(s)$ for the set of all milestones of s.

Remark 2.4.2. In the following we are mainly concerned with generalised milestones. Only in Section 3.1 the concept of milestones appears as a useful concept on its own.

A simple observation is that we can derive a bound on the number of generalised milestones from the height and the width of a stack.

Lemma 2.4.3. For each stack s there are less than $2 \cdot \mathsf{hgt}(s) \cdot |s|$ many generalised milestones.

In our informal description of generalised milestones, we said that the generalised milestones of s are those stacks that every run to s has to pass. In order to show this, we use

a result of Carayol [15]. He showed the following. For each higher-order pushdown stack s there is a unique minimal sequence of stack operations that creates s from the initial stack. On level two, this sequence creates the stack word by word, i.e., it starts with a sequence of push operations writing the first word onto the stack, then there is a clone operation, after this there is a sequence of pop_1 transitions followed by a sequence of push transitions that create the second word of the stack, then there follows a clone and so on. Furthermore, the topmost word reached after the n-th of the pop_1 sequences is exactly the greatest common prefix of the n-th and the $(n-1)$-st word of the stack. This result directly carries over to collapsible pushdown stacks due to the following fact: on level two the result of a collapse operation is either the same as applying a pop_1 or a sequence of pop_2 operations. Carayol's result shows that for any sequence containing a pop_2 operation, there is a shorter one where this pop_2 is eliminated. Hence, we can eliminate in the same way any $\mathsf{collapse}$ of link level 2. Finally, any other $\mathsf{collapse}$ can be treated like a pop_1 operation. We describe Carayol's result more formally in the following lemma.

Lemma 2.4.4 ([15]).

- For each collapsible pushdown stack s of level 2 there is a minimal sequence of operations $\mathsf{op}_1, \mathsf{op}_2, \ldots, \mathsf{op}_n \in \{\mathsf{push}_{\sigma,i}, \mathsf{pop}_1, \mathsf{clone}_2\}$ such that $s = \mathsf{op}_n(\mathsf{op}_{n_1}(\ldots(\mathsf{op}_1(\bot_2))))$.
- For $\mathsf{op}_1, \mathsf{op}_2, \ldots, \mathsf{op}_n$ the minimal sequence generating a stack s, the stack

  $$\mathsf{op}_j(\mathsf{op}_{j-1}(\ldots \mathsf{op}_0(\bot_2)))$$

  is a generalised milestone of s for each $0 \leq j \leq n$.

  Furthermore, for each generalised milestone m of s there is a $0 \leq j \leq n$ such that $m = \mathsf{op}_j(\mathsf{op}_{j-1}(\ldots \mathsf{op}_0(\bot_2)))$.
- Every run ρ to some stack s passes all generalised milestones of s.

Remark 2.4.5. From the minimality of the sequence $\mathsf{op}_1, \mathsf{op}_2, \ldots, \mathsf{op}_n$ generating s it follows that there is a bijection between the initial subsequences $\mathsf{op}_1, \mathsf{op}_2, \ldots, \mathsf{op}_j$ and the milestones of s. From now on, we call $\mathsf{op}_j(\mathsf{op}_{j-1}(\ldots \mathsf{op}_0(\bot_2)))$ the j-th milestone of s.

Note that if $i \leq j$ then the i-th milestone m_i of s is a milestone of the j-th milestone m_j of s. If we restrict this order to the set of milestones $\mathsf{MS}(s)$, then it coincides with the substack relation.

We want to conclude the analysis of generalised milestones with a lemma that characterises runs connecting generalised milestones in terms of loops. Thus, we first give a precise definition of loops. Then we prove this characterisation. A loop is a run that starts and ends in the same stack and which satisfies certain restrictions concerning the substacks that are passed.

Definition 2.4.6. A loop from (q,s) to (q',s) is a run λ that does not pass a substack of $\mathsf{pop}_2(s)$ and that may pass $\mathsf{pop}_1^k(s)$ only if the k topmost elements of $\mathsf{top}_2(s)$ are letters with links of level 1. This means that for all $i \in \mathsf{dom}(\lambda)$, if $\lambda(i) = (q_i, \mathsf{pop}_1^k(s))$ then $\mathsf{CLvl}(\mathsf{pop}_1^{k'}(s)) = 1$ for all $0 \leq k' < k$.

If λ is a loop from (q,s) to (q',s) such that $\lambda(1) = \mathsf{pop}_1(s)$ and $\lambda(\mathsf{ln}(\lambda) - 1) = \mathsf{pop}_1(s)$, then we call λ a low loop.

If λ is a loop from (q,s) to (q',s) that never passes $\mathsf{pop}_1(s)$, then we call λ a high loop.

Remark 2.4.7. If λ is a loop from (q,s) to (q',s) such that the stack at i and at j is s for $i \leq j \in \mathsf{dom}(\lambda)$, then $\lambda{\restriction_{[i,j]}}$ is a loop.

We now characterise runs connecting milestones in terms of loops.

Lemma 2.4.8. Let ρ be a run from the initial configuration to the stack $s = w_1 : w_2 : \cdots : w_k$. Furthermore, let n be the number of generalised milestones of s. For all $i \leq n$, let m_i be the i-th generalised milestone of s. Furthermore, let n_i denote the maximal position such that the stack of $\rho(n_i)$ is m_i. We write q_i for the state of $\rho(n_i)$, i.e., $\rho(n_i) = (q_i, m_i)$. For all $i < n$, there is some state q'_{i+1} such that there is a transition from $\rho(n_i)$ to $(q'_{i+1}, m_{i+1}) = \rho(n_i+1)$ and $\rho{\restriction_{[n_i+1,n_{i+1}]}}$ is a loop of m_{i+1}. Furthermore, $\rho{\restriction_{[0,n_1]}}$ is a loop of $\bot_2$.

Proof. Fix some $i \in \mathbb{N}$. We prove the claim for m_i and m_{i+1}. We distinguish the following cases.

- Assume that $m_{i+1} = \mathsf{clone}_2(m_i)$. In this case $m_i = w_1 : w_2 : \cdots : w_{|m_i|}$. Thus, at the last position $j \in \mathsf{dom}(\rho)$ where $|\rho(j)| = |m_i|$, the stack at $\rho(j)$ is m_i (because ρ never changes the first $|m_i|$ many words after passing $\rho(j)$). Hence, $j = n_i$ by definition. Since $|s| > |m_i|$, it follows directly that the operation at n_i is a clone_2 leading to m_{i+1}. Note that ρ never passes a stack of width $|m_i|$ again. Thus, it follows from Lemma 2.3.27 that $\rho{\restriction_{[n_i+1,n_{i+1}]}}$ satisfies the restriction that it never visits $\mathsf{pop}_1^k(m_{i+1})$ if $\mathsf{CLvl}(\mathsf{pop}_1^{k-1}(m_{i+1})) = 2$. Thus, we conclude that this restriction is a loop.

- Assume that $m_{i+1} = \mathsf{pop}_1(m_i)$. In this case, $m_i = w_1 : w_2 : \cdots : w_{|m_i|-1} : w$ for some w such that $w_{|m_i|-1} \sqcap w_{|m_i|} < w \leq w_{|m_i|-1}$. Thus, $w \not\leq w_{|m_i|}$ and creating $w_{|m_i|}$ as the $|m_i|$-th word on the stack requires passing $w_1 : w_2 : \cdots : w_{|m_i|-1} : w_{|m_i|-1} \sqcap w_{|m_i|}$. This is only possible via applying pop_1 or $\mathsf{collapse}$ of level 1 to m_i. Since we assumed n_i to be maximal, the operation at n_i must be pop_1 or $\mathsf{collapse}$ of level 1 and leads to m_{i+1}.

  We still have to show that $\rho{\restriction_{[n_i+1,n_{i+1}]}}$ is a loop. By definition of n_{i+1}, $\rho{\restriction_{[n_i+1,n_{i+1}]}}$ starts and ends in m_{i+1}. By maximality of n_i, $\rho{\restriction_{[n_i+1,n_{i+1}]}}$ does not visit the stack $\mathsf{pop}_2(m_i) = \mathsf{pop}_2(m_{i+1})$. Furthermore, note that $\mathsf{top}_1(m_{i+1})$ is a cloned element. Hence, Lemma 2.3.27 implies that $\rho{\restriction_{[n_i+1,n_{i+1}]}}$ may only visit $\mathsf{pop}_1^k(m_{i+1})$ in case that $\mathsf{CLvl}(\mathsf{pop}_1^{k-1}(m_{i+1})) = 1$. Thus, $\rho{\restriction_{[n_i+1,n_{i+1}]}}$ is a loop.

- The last case is $m_{i+1} = \mathsf{push}_{\sigma,l}(m_i)$ for $(\sigma, l) \in \Sigma \times \{1,2\}$. In this case,

  $$m_i = w_1 : w_2 : \cdots : w_{|m_i|-1} : w$$

  for some w such that $w_{|m_i|-1} \sqcap w_{|m_i|} \leq w < w_{|m_i|}$. Creating $w_{|m_i|}$ on the stack requires pushing the missing symbols onto the stack as they cannot be obtained via clone operation from the previous word. Since n_i is maximal, the operation at n_i is some $\mathsf{push}_{\sigma,l}$ leading to m_{i+1}. $\rho{\restriction_{[n_i+1,n_{i+1}]}}$ is a high loop due to the maximality of n_i (this part of ρ never visits $m_i = \mathsf{pop}_1(m_{i+1})$ or any other proper substack of m_{i+1}). □

We conclude this section by rephrasing this result in terms of milestones. We will use it in this form in Chapter 3.1.

Corollary 2.4.9. Let ρ be a run from the initial configuration to the configuration (q,s) where s decomposes as

$$s = w_1 : w_2 : \cdots : w_k.$$

Let n_i denote the maximal position such that $\rho(n_i) = (q, m_i)$ for some $q \in Q$ and m_i the i-th milestone of s. We define $q_i \in Q$ such that $\rho(n_i) = (q_i, m_i)$. Then one of the following applies.

1. There is a $\mathsf{push}_{\sigma,j}$ transition from $\rho(n_i) = (q_i, m_i)$ to $(q'_{i+1}, m_{i+1}) := \rho(n_i + 1)$ and $\rho\restriction_{[n_i+1, n_{i+1}]}$ is a loop of m_{i+1}, or
2. there is a clone_2 transition followed by a sequence $\lambda_0 \circ \pi_1 \circ \lambda_1 \cdots \circ \pi_n \circ \lambda_n$ where the λ_i are loops and the π_i are runs that perform exactly one pop_1 operation or collapse of level 1 each.

Furthermore, we have

3. $\rho(n_1) = (q_1, [\bot])$, i.e., $\rho\restriction_{[0,n_1]}$ is a loop of $[\bot]$. If m is the number of milestones of s, then $\rho(n_m) = (q, s)$ is the final configuration of ρ.

As another direct corollary of the lemma, we obtain that the linear order of the milestones induced by the substack relation coincides with the order in which the milestones appear for the last time in a given run.

Corollary 2.4.10. For an arbitrary run ρ from the initial configuration to some stack s, the function

$$\begin{aligned} f : \mathrm{MS}(s) &\to \mathrm{dom}(\rho) \\ s' &\mapsto \max\{i \in \mathrm{dom}(\rho) : \rho(i) = (q, s') \text{ for some } q \in Q\} \end{aligned}$$

is an order embedding.

We have seen that generalised milestones induce a uniform decomposition of all runs to a given stack. Furthermore, the parts of the run that connect generalised milestones always consist of a loop plus one further transition. In order to understand the existence of runs to certain configurations, we investigate the theory of loops in the following.

2.4.2 Loops and Returns

Recall that we have already defined loops in Definition 2.4.6. Next, we define returns which are runs from a stack $s : w$ to s without visiting substacks of s. Our interest in returns stems from the fact that they appear as subruns of high loops whence they play an important role in finding loops for a given stack.

Definition 2.4.11. Let $t = s : w$ be some stack with topmost word w. A return from t to s is a run ρ from t to s such that ρ never visits a substack of s except for the last stack of ρ and such that one of the following holds:

1. the last operation in ρ is pop_2,
2. the last operation in ρ is a $\mathsf{collapse}$ and $w < \mathrm{top}_2(\rho(\ln(\rho) - 1))$, i.e., ρ pushes at first some new letters onto t and then performs a collapse of one of these new letters, or
3. there is some $i \in \mathrm{dom}(\rho)$ such that $\rho\restriction_{[i,\ln(\rho)]}$ is a return from $\mathsf{pop}_1(t)$ to s.

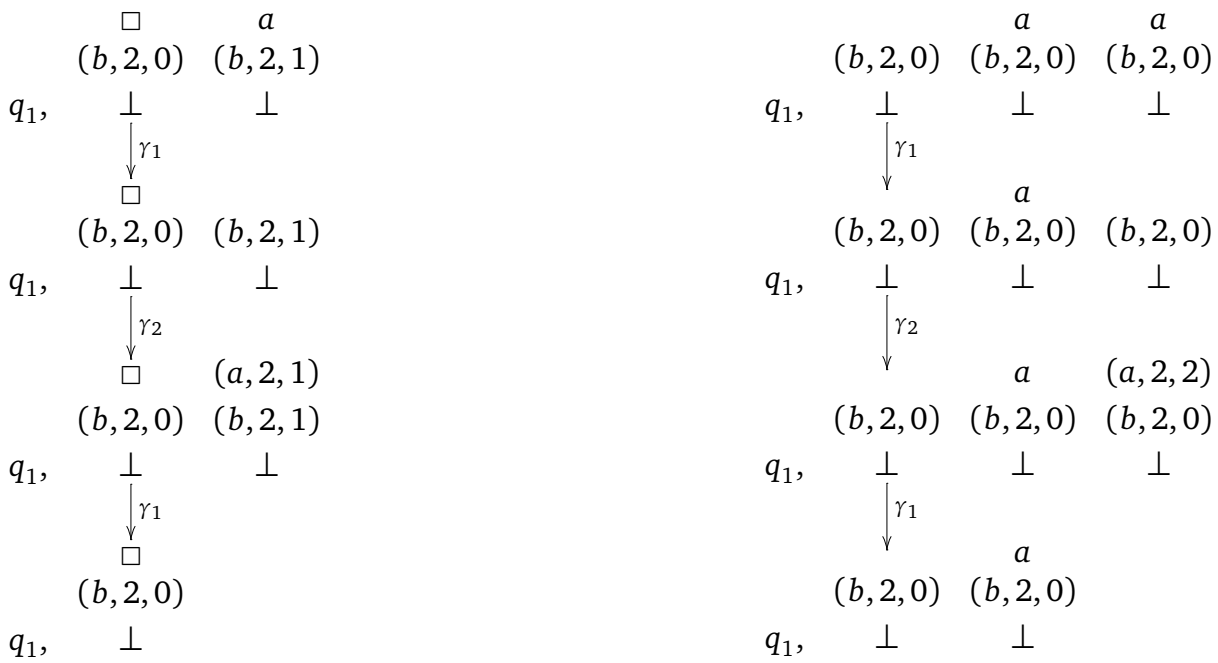


Figure 2.4.: The run ρ from (q_1, s) to $(q_1, \mathsf{pop}_2(s))$ on the left side and the run $\rho' = \lambda\restriction_{[1,4]}$ from (q_1, s') to $(q_1, \mathsf{pop}_2(s'))$ on the right side.

Remark 2.4.12. A return from t to $\mathsf{pop}_2(t)$ is a run ρ from t to $\mathsf{pop}_2(t)$ such that ρ never visits a substack of $\mathsf{pop}_2(t)$ except for the last stack of ρ and that does not use the level 2 links stored in $\mathsf{top}_2(t)$.

We first give an example for this definition, afterwards we discuss its motivation.

Example 2.4.13. Consider a collapsible pushdown system $\mathscr{S}$ over the alphabet $\{\bot, \top, a, b\}$ with the transitions $(q_0, a, \gamma_0, q_1, \mathsf{clone}_2), (q_1, a, \gamma_1, q_1, \mathsf{collapse})$ and $(q_1, b, \gamma_2, q_1, \mathsf{push}_{a,2})$. Consider the stack

$$s := \bot(b,2,0)\square : \bot(b,2,1)a.$$

The transitions induce a unique run ρ from (q_1, s) to $(q_1, \mathsf{pop}_2(s))$ of length 3. ρ is depicted on the left side of Figure 2.4. $\rho\restriction_{[1,3]}$ is a return from

$$(q_1, \bot(b,2,0)\square : \bot(b,2,1)) \text{ to}$$
$$(q_1, [\bot(b,2,0)\square])$$

because it satisfies the second item of the definition of a return. Hence, ρ is a return because it satisfies the third item of the definition.

We want to consider a second example that shows how returns occur as subruns of loops. The transitions induce a loop λ from

$$(q_0, \bot(b,2,0) : \bot(b,2,0)a) \text{ to}$$
$$(q_1, \bot(b,2,0) : \bot(b,2,0)a).$$

The run passes $(q_1, \bot(b,2,0) : \bot(b,2,0)a : \bot(b,2,0)a)$ and continues from there as depicted on the right side of Figure 2.4 (the figure shows λ without its first configuration

because this final part of λ plays a role in the next remark). Note that $\lambda\restriction_{[2,4]}$ is a return starting from a stack with topmost word $\mathsf{pop}_1(\mathsf{top}_2(\lambda(0)))$. Later, when we analyse loops in detail we will see that this is a typical occurrence of a return. Any loop of a stack s decomposes into parts prefixed by s and parts that are returns of stacks with topmost word $\mathsf{pop}_1(\mathsf{top}_2(s))$.

Remark 2.4.14. A return is a run from some stack s to $\mathsf{pop}_2(s)$ that depends on the symbols and link levels of $\mathsf{top}_2(s)$, but not on any other content of s in the following sense. A return from s to $\mathsf{pop}_2(s)$ consists of a sequence of transitions. For any stack s' with $|s'| \geq 2$ such that the topmost words of s and s' coincide on their symbols and link levels, this sequence can be applied to s'. The resulting run induced by this sequence is then a return from s' to $\mathsf{pop}_2(s')$.

We explain this idea with some examples. Let $\mathscr{S}$ be the pushdown system and ρ the return from s to $\mathsf{pop}_2(s)$ as in Example 2.4.13. Consider the stack

$$s' := \bot(b,2,0) : \bot(b,2,0)a : \bot(b,2,0)a.$$

Note that the symbols and link levels of $\mathsf{top}_2(s)$ and $\mathsf{top}_2(s')$ agree while their links differ. There is a return ρ' from (q_1,s') to $(q_1,\mathsf{pop}_2(s'))$ which is obtained by starting in (q_1,s') and copying the transitions of ρ one by one. The resulting return ρ' is depicted on the right side of Figure 2.4.

This is not by accident, but by intention: whenever two stacks s and s' coincide on the symbols and link levels of their topmost words, we can copy a return from s to $\mathsf{pop}_2(s)$ transition by transition and obtain a return from s' to $\mathsf{pop}_2(s')$. This is due to two facts.

Firstly, a return from s to $\mathsf{pop}_2(s)$ never looks into $\mathsf{pop}_2(s)$ before its last configuration. Thus, the words below the topmost word have no influence on this run. Secondly, the restriction of the use of collapse links ensures that a return only uses collapse links of level 2 if these were created during the run ρ. If such a link points to $\mathsf{pop}_2(s)$, it is created by a push operation at some position i in ρ on a stack of width $|s|$. But then ρ', the one to one copy of the transitions of ρ with starting stack s', uses a push transition at position i on a stack of width $|s'|$. Thus, the link created in this step points to $\mathsf{pop}_2(s')$. Hence, if ρ uses the created collapse link and collapses the stack to $\mathsf{pop}_2(s)$, then ρ' uses the copy of this link and collapses to $\mathsf{pop}_2(s')$.

We defined returns in such a way that they are runs from some stack s to $\mathsf{pop}_2(s)$ that are independent of the links and the words below the topmost one. The next example shows that the restricted use of the collapse operation in the definition of returns is crucial for this property. We present a run from some stack $\hat{s}$ to $\mathsf{pop}_2(\hat{s})$ that does not look into the substacks of $\mathsf{pop}_2(\hat{s})$ before the final position but that lacks the independence of the level 2 links of the topmost word.

Consider the stacks

$$\hat{s} := \bot(b,2,0)(b,2,0) : \bot(a,2,1)a \text{ and}$$
$$\hat{s}' := \bot(b,2,0) : \bot(a,2,1) : \bot(a,2,1)a.$$

We still consider the transitions given in Example 2.4.13. Using these transitions, there are runs $\hat{\rho}$ from $(q_1,\hat{s})$ to $(q_1,\mathsf{pop}_2(\hat{s}))$ and $\hat{\rho}'$ from $(q_1,\hat{s}')$ to $(q_1,[\bot(b,2,0)])$ as depicted in Figure 2.5.

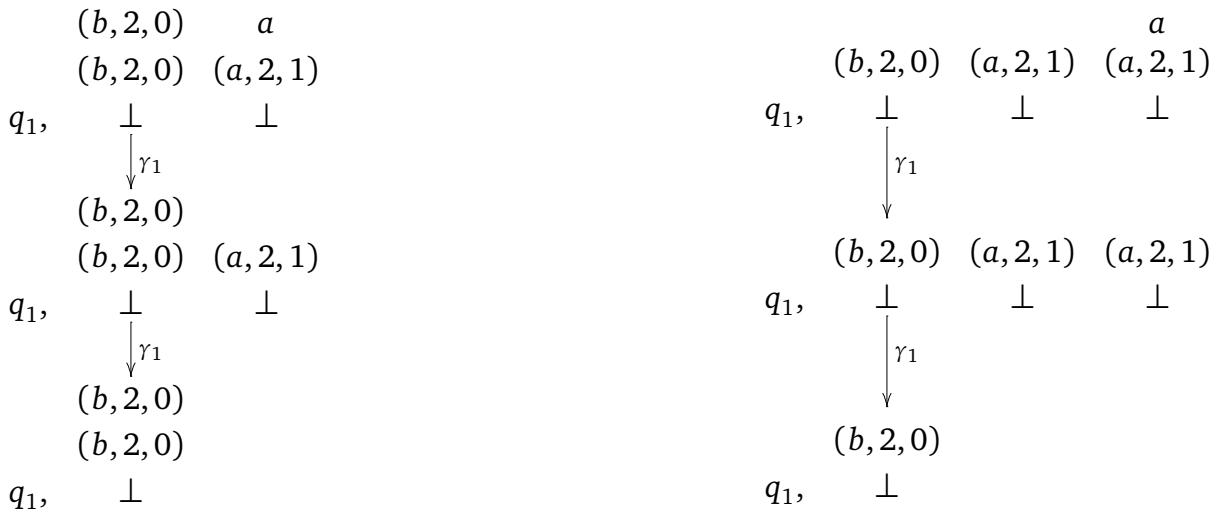


Figure 2.5.: The run $\hat{\rho}$ from $(q_1, \hat{s})$ to $(q_1, \mathsf{pop}_2(\hat{s}))$ on the left side and the run $\hat{\rho}'$ from $(q_1, \hat{s}')$ to $(q_1, \mathsf{pop}_2(\mathsf{pop}_2(\hat{s}')))$ on the right side.

Note that $\hat{\rho}$ is no return because it uses the level 2 collapse link stored in $\mathsf{top}_2(\hat{s})$. Furthermore, $\mathsf{top}_2(\hat{s}) = \mathsf{top}_2(\hat{s}')$ and $\hat{\rho}'$ copies $\hat{\rho}$ transition by transition. Nevertheless, $\hat{\rho}'$ does not end with the stack $\mathsf{pop}_2(\hat{s}')$ but with $\mathsf{pop}_2^2(\hat{s}')$.

Thus, if we drop the restriction on the use of collapse links, then we obtain runs from some stack s to $\mathsf{pop}_2(s)$ that cannot be transferred into runs from stacks s' to $\mathsf{pop}_2(s')$ even though $\mathsf{top}_2(s) = \mathsf{top}_2(s')$.

2.4.3 Computing Returns

As already mentioned in the previous section, the theory of returns is important for the theory of loops. Thus, we first study the theory of returns on its own. Later we apply this theory to the theory of loops. Our main goal in this part is to provide a finite automaton that calculates on input $\mathsf{top}_2(s)$ the number of returns from (q, s) to $(q', \mathsf{pop}_2(s))$ up to a given threshold $k \in \mathbb{N}$. We start by introducing appropriate notation for this purpose.

Definition 2.4.15. Let $\mathscr{S}$ be a collapsible pushdown system of level 2. We set

$$\#\mathsf{Ret}^k_{\mathscr{S}}(s) : Q \times Q \to \{0, 1, \ldots, k\}$$

$$(q, q') \mapsto \begin{cases} i & \text{if there are exactly } i \leq k \text{ different returns of } \mathscr{S} \text{ from } (q, s) \text{ to } (q', \mathsf{pop}_2(s)) \\ k & \text{otherwise.} \end{cases}$$

Remark 2.4.16. This function maps (q, q') to the number i of returns from (q, s) to (q', s) if $i \leq k$ and it maps (q, q') to k otherwise. In this sense k stands for the class of at least k returns. Thus, the answer to the question "how many returns from (q, s) to $(q', \mathsf{pop}_2(s))$ exist up to threshold k?" is exactly the value of $\#\mathsf{Ret}^k_{\mathscr{S}}(s)(q, q')$. If $\mathscr{S}$ is clear from the context, we will omit it and write $\#\mathsf{Ret}^k$ instead of $\#\mathsf{Ret}^k_{\mathscr{S}}$.

As already indicated in the examples, it turns out that we can copy returns between stacks which agree on their topmost words. Lemma 2.4.27 proves this fact. A corollary of this lemma is that the number of returns from (q, s) to $(q', \mathsf{pop}_2(s))$ only depend on the topmost word of s. Hence, the following definition is well-defined.

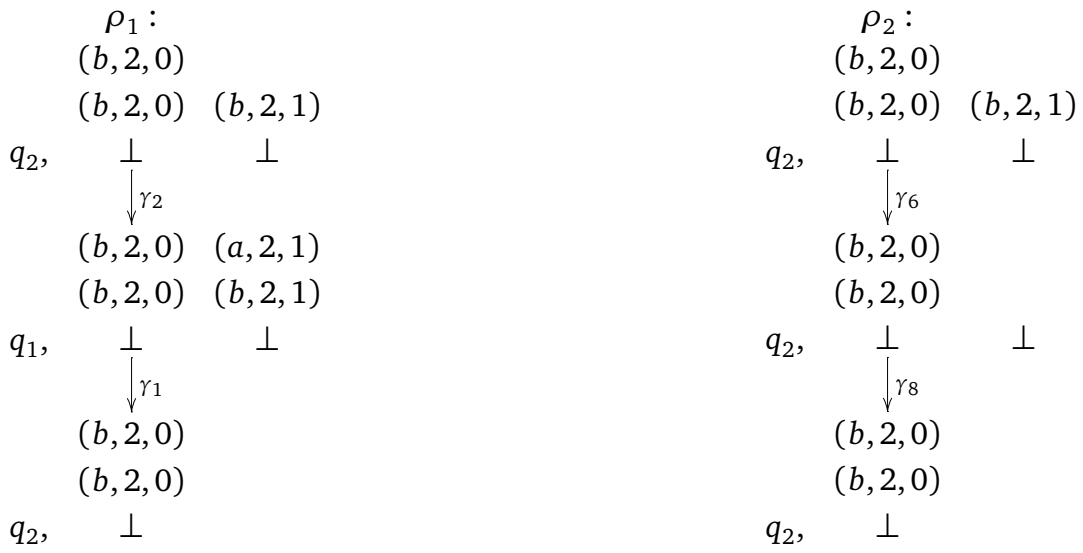


Figure 2.6.: The two returns from (q_2, s') to $(q_2, \mathrm{pop}_2(s'))$.

Definition 2.4.17. For w an arbitrary word, let $\#\mathrm{Ret}^k(w)$ be $\#\mathrm{Ret}^k(s)$ for an arbitrary stack s with $\mathrm{top}_2(s) = w$ and $|s| \geq 2$.

The next part of this section aims at a better understanding of the dependence of $\#\mathrm{Ret}^k(s)$ from $\mathrm{top}_2(s)$. Let w be the topmost word of the stack s. It will turn out that $\#\mathrm{Ret}^k(w)$ only depends on $\#\mathrm{Ret}^k(\mathrm{pop}_1(w))$, on $\mathrm{Sym}(w)$ and on $\mathrm{CLvl}(w)$. This means that the topmost element of w and the number of returns of stacks with topmost word $\mathrm{pop}_1(w)$ already determine the number of returns of s. This implies that $\#\mathrm{Ret}^k(s)$ can be computed as follows. First, we compute the number of returns of stacks with topmost word $\bot$. Then we compute $\#\mathrm{Ret}^k(w_i)$ where w_i is the prefix of w of length i from $i = 2$, $i = 3$, ... until we have computed $\#\mathrm{Ret}^k(w_i)$ for $w_i = w$ or equivalently, for $i = |w|$. Before we prove this claim in detail, let us give an example.

Example 2.4.18. Consider the pushdown system $\mathcal{S}$ given by the transitions

$$(q_0, a, \gamma_0, q_2, \mathrm{collapse}), \quad (q_1, a, \gamma_1, q_2, \mathrm{collapse}), \quad (q_2, b, \gamma_2, q_1, \mathrm{push}_{a,2}),$$
$$(q_0, a, \gamma_3, q_3, \mathrm{push}_{c,2}), \quad (q_3, c, \gamma_4, q_2, \mathrm{clone}_2), \quad (q_2, c, \gamma_5, q_2, \mathrm{pop}_1),$$
$$(q_2, b, \gamma_6, q_2, \mathrm{pop}_1), \quad (q_2, a, \gamma_7, q_2, \mathrm{pop}_1), \quad \text{and } (q_2, \bot, \gamma_8, q_2, \mathrm{pop}_2).$$

Consider the stacks

$$s = \bot(b,2,0)(b,2,0) : \bot(b,2,1)a \qquad \text{and} \qquad s' := \mathrm{pop}_1(s).$$

There are exactly two returns of $\mathcal{S}$ from (q_2, s') to $(q_2, \mathrm{pop}_2(s'))$. These are depicted in Figure 2.6. We call them ρ_1 and ρ_2.

We explain how returns from (q_0, s) to $(q_2, \mathrm{pop}_2(s))$ depend on those from (q_2, s') to $(q_2, \mathrm{pop}_2(s'))$. First of all note that there is a return π_1 from (q_0, s) to $(q_2, \mathrm{pop}_2(s))$ as depicted on the left side of Figure 2.7. This return π_1 decomposes as $\pi_1 = \pi_1{\restriction}_{[0,1]} \circ \rho_1$. If we replace ρ_1 by the other return ρ_2, then we obtain again a return which we call $\pi_2 := \pi_1{\restriction}_{[0,1]} \circ \rho_2$. This run is depicted on the right side of Figure 2.7. In the following, we

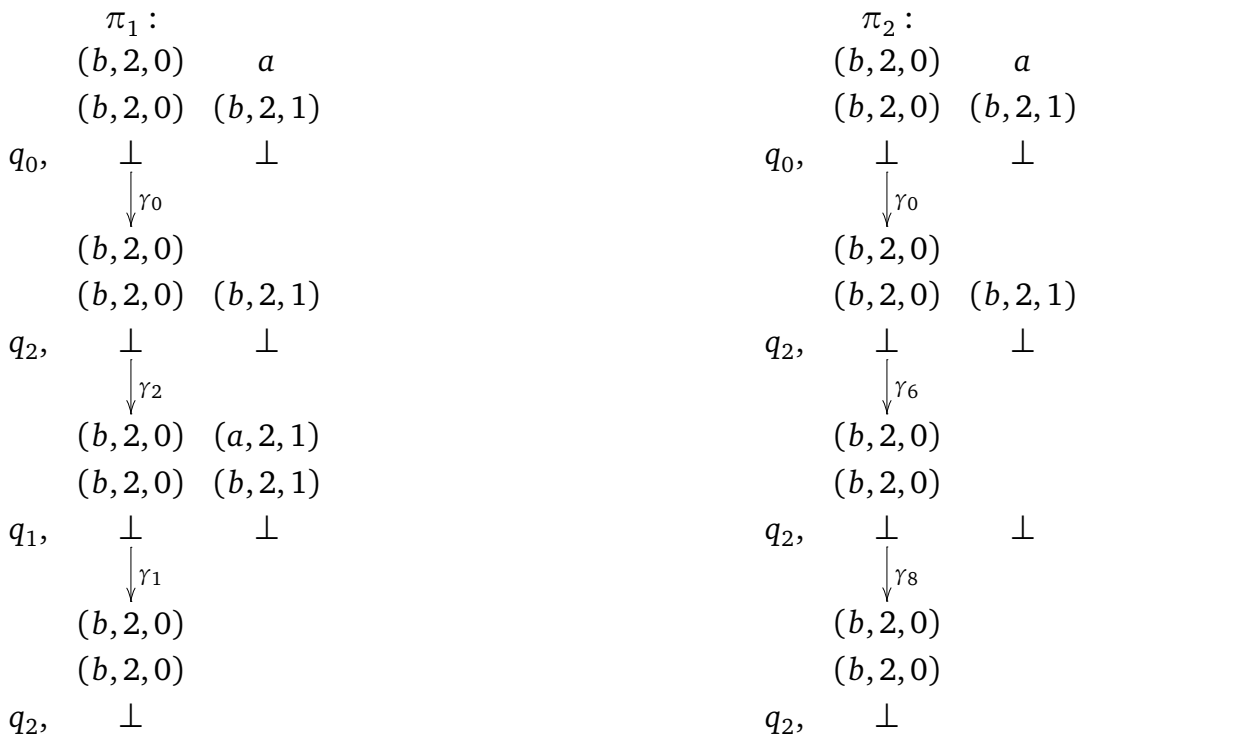


Figure 2.7.: Two returns from (q_0, s) to $(q_2, \mathsf{pop}_2(s))$.

consider $\pi_1 \restriction_{[0,1]}$ as a representative for the returns π_1 and π_2 because both returns can be obtained from $\pi_1 \restriction_{[0,1]}$ by attaching a return with topmost word $\mathsf{pop}_1(\mathsf{top}_2(s))$. Furthermore, the existence of $\pi_1 \restriction_{[0,1]}$ only depends on $\mathsf{Sym}(s)$ and $\mathsf{CLvl}(s)$: on any stack with topmost symbol a of link level 1, we can perform the sequence of transitions $\pi_1 \restriction_{[0,1]}$ consists of. Let us now turn to the other returns from (q_0, s) to $(q_2, \mathsf{pop}_2(s))$. Figure 2.8 depicts another return π_3. Note that $\pi_3 \restriction_{[4,6]}$ and $\pi_3 \restriction_{[8,10]}$ are returns starting at stacks with topmost words $\mathsf{pop}_1(\mathsf{top}_2(s))$. $\pi_3 \restriction_{[8,10]}$ is ρ_2 and $\pi_3 \restriction_{[4,6]}$ copies the transitions of ρ_2 one by one. We can replace each of these parts of π_3 by the return ρ_1 (or by a one by one copy of its transitions) and obtain another return. We can also replace both parts by copies of the return ρ_1 and obtain a fourth return. Thus, we obtain 4 different returns from (q_0, s) to $(q_2, \mathsf{pop}_2(s))$ from the pair $(\pi_3 \restriction_{[0,4]}, \pi_3 \restriction_{[6,8]})$ by plugging in different returns of topmost word $\mathsf{pop}_1(\mathsf{top}_2(s))$ after each element of this pair.

It is again an important observation that $(\pi_3 \restriction_{[0,4]}, \pi_3 \restriction_{[6,8]})$ only depends on $\mathsf{Sym}(s)$ and $\mathsf{CLvl}(s)$ in the following sense. Given any other stack t with topmost symbol a of link level 1, there is a run $\hat{\pi}_3 \restriction_{[0,4]}$ that copies the transitions of $\pi_3 \restriction_{[0,4]}$ and that ends in a stack t' with $\mathsf{top}_2(t') = \mathsf{pop}_1(\mathsf{top}_2(t))$. Similarly, we can copy the transitions of $\pi_3 \restriction_{[6,8]}$ to a run starting at $\mathsf{pop}_2(t')$ and which ends again in a stack with topmost word $\mathsf{pop}_1(\mathsf{top}_2(t))$.

It is easy to see that there are no other returns from (q_0, s) to $(q_2, \mathsf{pop}_2(s))$ than the ones we discussed above.

Thus, the tuple $\pi_1 \restriction_{[0,1]}, (\pi_3 \restriction_{[0,4]}, \pi_3 \restriction_{[6,8]})$ represents all returns from the configuration (q_0, s) to $(q_2, \mathsf{pop}_2(s))$ in the following sense.

1. $\pi_1 \restriction_{[0,1]}$ can be turned into a return from (q_0, s) to $(q_2, \mathsf{pop}_2(s))$ by appending a return of a stack with topmost word $\mathsf{pop}_1(\mathsf{top}_2(s))$.

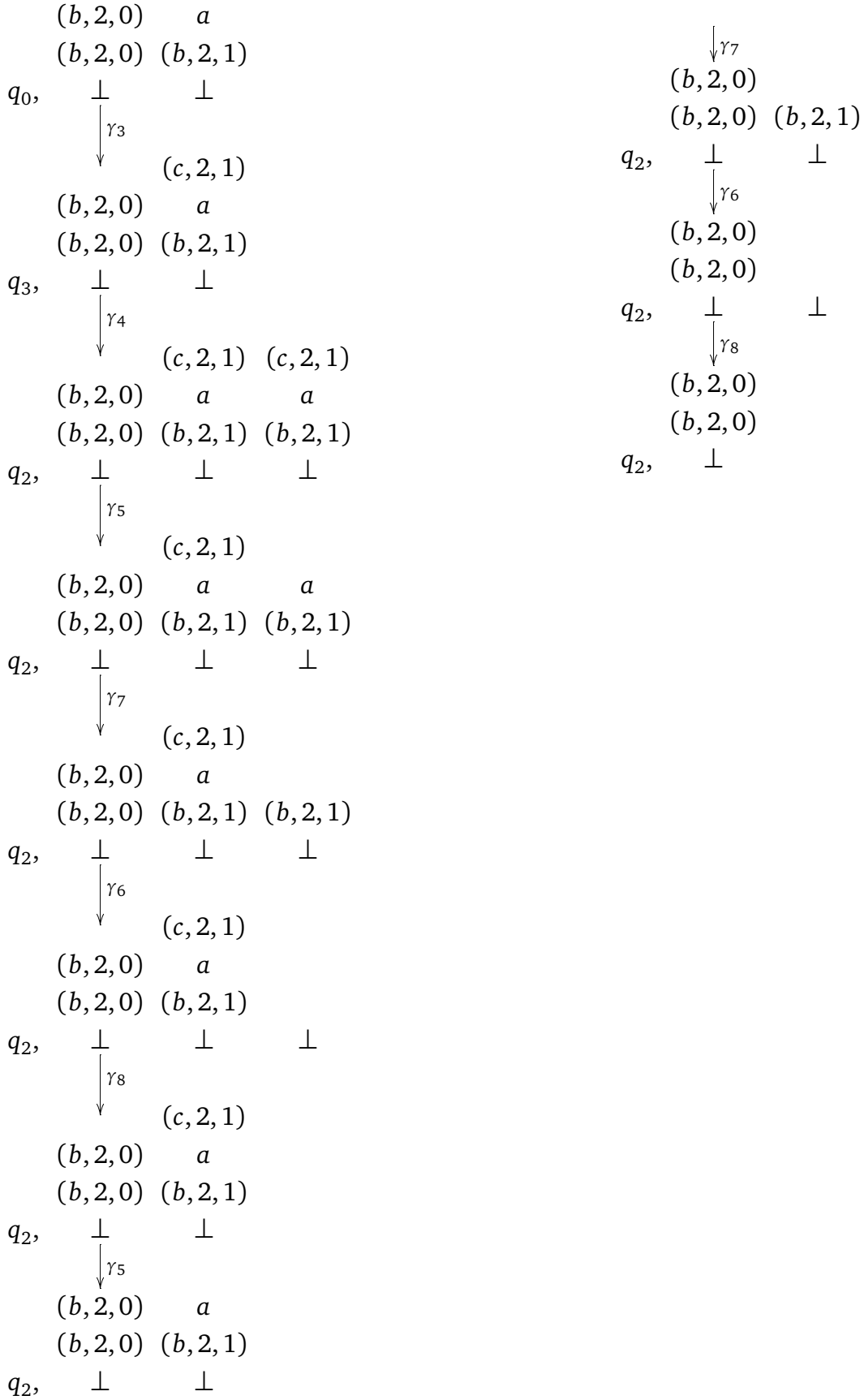


Figure 2.8.: The return π_3 from (q_0, s) to $(q_2, \mathrm{pop}_2(s))$.

2. $(\pi_3{\restriction}_{[0,4]}, \pi_3{\restriction}_{[6,8]})$ can be turned into a return from (q_0, s) to $(q_2, \mathsf{pop}_2(s))$ by plugging in one return of a stack with topmost word $\mathsf{pop}_1(\mathsf{top}_2(s))$ between the two runs and by appending such a return to the end of $\pi_3{\restriction}_{[6,8]}$.

3. All returns from (q_0, s) to $(q_2, \mathsf{pop}_2(s))$ are induced by this tuple in the sense of items 1 and 2.

Since there are 2 returns from state (q_2, t) to $(q_2, \mathsf{pop}_2(t))$ for any stack t of width at least 2 and topmost word $\mathsf{top}_2(t) = \mathsf{pop}_1(\mathsf{top}_2(s))$, we conclude that there are $2 + 2 \cdot 2 = 6$ different returns.

This form of computing the number of returns works for all stacks. Take for example the stack

$$\hat{s} := \bot : \bot(b, 2, 1)(b, 2, 1)a.$$

This stack has the same topmost symbol and link level as s. Thus, we can copy transition by transition the runs $\pi_1{\restriction}_{[0,1]}$ and $\pi_3{\restriction}_{[0,4]}$ to runs $\hat{\pi}_1$ and $\hat{\pi}_2$ starting from $\hat{s}$. Furthermore, note that the stack of $\pi_3(6)$ is $\mathsf{pop}_2(\pi_3(4))$. For $\hat{t}$ the stack obtain via a pop_2 from the last stack of $\hat{\pi}_2$ we can copy $\pi_3{\restriction}_{[6,8]}$ transition by transition to a run $\hat{\pi}_3$ starting at $\hat{t}$. The resulting runs are depicted in Figure 2.9.

Again, we can turn $\hat{\pi}_1$ and the pair $(\hat{\pi}_2, \hat{\pi}_3)$ into returns from $(q_0, \hat{s})$ to $(q_2, \mathsf{pop}_2(s))$. For this purpose, we have to plug in returns with topmost word $\bot(b, 2, 1)(b, 2, 1)$ from state q_2 to state q_2 after $\hat{\pi}_1$, $\hat{\pi}_2$, and $\hat{\pi}_3$.

There are exactly three returns from $(q_2, \bot : \bot(b, 2, 1)(b, 2, 1))$ to $(q_2, [\bot])$. The first performs $\vdash^{\gamma_2}$ and then $\vdash^{\gamma_1}$, the second performs $\vdash^{\gamma_6}, \vdash^{\gamma_2}$ and $\vdash^{\gamma_1}$, and the last one performs $\vdash^{\gamma_6}$, $\vdash^{\gamma_6}$ and $\vdash^{\gamma_8}$.

Since we have to append such a return to $\hat{\pi}_1$ in order to obtain a return of $\hat{s}$, $\hat{\pi}_1$ induces 3 different returns from $(q_0, \hat{s})$ to $(q_2, \mathsf{pop}_2(\hat{s}))$. Moreover, using 2 of these returns, we can turn the pair $(\hat{\pi}_2, \hat{\pi}_3)$ into a return from $(q_0, \hat{s})$ to $(q_2, \mathsf{pop}_2(\hat{s}))$. Hence, there are $3 \cdot 3 = 9$ possibilities to turn this pair into a return. We conclude that there are $3 + 9 = 12$ returns from $(q_0, \hat{s})$ to $(q_2, \mathsf{pop}_2(\hat{s}))$. We leave it as an exercise to figure out that there are exactly 12 returns from $(q_0, \hat{s})$ to $(q_2, \mathsf{pop}_2(\hat{s}))$.

The previous example pointed to a connection between returns of a stack with topmost word w and the returns of stacks with topmost word $\mathsf{pop}_1(w)$. The main result of this section is that this connection can be used to define a finite automaton that calculates on input $\mathsf{top}_2(s)$ the function $\#\mathsf{Ret}^k(s)$ for a given $k \in \mathbb{N}$. Furthermore, this dependence can be used to calculate a bound on the length of returns in dependence of the length of the topmost word of a stack. We first state these two results, afterwards we provide the technical background for the proofs.

Proposition 2.4.19. There is an algorithm that, given a collapsible pushdown system $\mathcal{S}$ of level 2, computes a deterministic finite automaton $\mathcal{A}_{\text{ret}}$ with the following property. $\mathcal{A}_{\text{ret}}$ computes $\#\mathsf{Ret}^k(s : w)$ on input $\pi(w)$ where $\pi(w)$ denotes the projection of w to its symbols and link levels.

Proposition 2.4.20. There is an algorithm that, on input some 2-CPG $\mathcal{S}$ and a natural number k, computes a function $\mathsf{BRL}_k^{\mathcal{S}} : \mathbb{N} \to \mathbb{N}$ with the following properties.

$\hat{\pi}_1$:

$$
\begin{array}{lc}
 & a \\
 & (b,2,1) \\
 & (b,2,1) \\
q_0, & \bot \quad \bot \\
 & \downarrow \gamma_0 \\
 & (b,2,1) \\
 & (b,2,1) \\
q_3, & \bot \quad \bot
\end{array}
$$

$\hat{\pi}_2$:

$$
\begin{array}{lc}
 & a \\
 & (b,2,1) \\
 & (b,2,1) \\
q_0, & \bot \quad \bot \\
 & \downarrow \gamma_3 \\
 & (c,2,1) \\
 & a \\
 & (b,2,1) \\
 & (b,2,1) \\
q_3, & \bot \quad \bot \\
 & \downarrow \gamma_4 \\
 & (c,2,1) \quad (c,2,1) \\
 & a \quad a \\
 & (b,2,1) \quad (b,2,1) \\
 & (b,2,1) \quad (b,2,1) \\
q_2, & \bot \quad \bot \quad \bot \\
 & \downarrow \gamma_5 \\
 & (c,2,1) \\
 & a \quad a \\
 & (b,2,1) \quad (b,2,1) \\
 & (b,2,1) \quad (b,2,1) \\
q_2, & \bot \quad \bot \quad \bot \\
 & \downarrow \gamma_7 \\
 & (c,2,1) \\
 & a \\
 & (b,2,1) \quad (b,2,1) \\
 & (b,2,1) \quad (b,2,1) \\
q_2, & \bot \quad \bot \quad \bot
\end{array}
$$

$\hat{\pi}_3$:

$$
\begin{array}{lc}
 & (c,2,1) \\
 & a \\
 & (b,2,1) \\
 & (b,2,1) \\
q_2, & \bot \quad \bot \\
 & \downarrow \gamma_5 \\
 & a \\
 & (b,2,1) \\
 & (b,2,1) \\
q_2, & \bot \quad \bot \\
 & \downarrow \gamma_7 \\
 & (b,2,1) \\
 & (b,2,1) \\
q_2, & \bot \quad \bot
\end{array}
$$

Figure 2.9.: $\hat{\pi}_1$ corresponding to $\pi_1 \restriction_{[0,1]}$, $\hat{\pi}_2$ corresponding to $\pi_3 \restriction_{[0,4]}$, and $\hat{\pi}_3$ corresponding to $\pi_3 \restriction_{[6,8]}$.

1. For each stack s, for states q_1, q_2 and for $i := \#\mathrm{Ret}^k(s)(q_1, q_2)$, the length-lexicographically shortest returns $\rho_1, \ldots, \rho_i$ from (q_1, s) to $(q_2, \mathrm{pop}_2(s))$ satisfy

$$\ln(\rho_j) \leq \mathrm{BRL}_k^{\mathscr{S}}(|\mathrm{top}_2(s)|) \text{ for all } 1 \leq j \leq i.$$

2. If there is a return ρ from (q_1, s) to $(q_2, \mathrm{pop}_2(s))$ with $\ln(\rho) > \mathrm{BRL}_k^{\mathscr{S}}(|\mathrm{top}_2(s)|)$, then there are k returns from (q_1, s) to $(q_2, \mathrm{pop}_2(s))$ of length at most $\mathrm{BRL}_k^{\mathscr{S}}(|\mathrm{top}_2(s)|)$.

For any stack s with topmost word $\perp$, we will calculate the number of returns of s using Lemma 2.3.30.

We can inductively calculate the returns of some stack s as follows. Assume that we already know how to calculate returns of stacks with topmost word of size $|\mathrm{top}_2(s)| - 1$. Any return of s splits into those parts that only depend on its topmost symbol and link level and those parts that are returns from stacks with smaller topmost word (cf. Example 2.4.18). By induction hypothesis we already counted the latter parts. Hence, we have to focus on the other parts. Here again, we can reduce the counting of these runs to an application of Lemma 2.3.30. This reduction to Lemma 2.3.30 is uniformly in the length of the topmost word from s. Due to this uniformity, we can then compute a finite automaton that calculates the number of returns.

The reader who is not interested in the technical details of the proofs of the propositions may safely skip this part and continue reading Section 2.4.4.

We start the analysis of returns with a general observation. By definition, there are returns ρ where there is some $i \in \mathrm{dom}(\rho)$ such that $\rho\restriction_{[i,\ln(\rho)]}$ is a return from $\mathrm{pop}_1(\rho(0))$. Our first lemma shows that a run ρ which visits $\mathrm{pop}_1(\rho(0))$ is a return if and only if a suffix of ρ is a return from $\mathrm{pop}_1(\rho(0))$ to $\mathrm{pop}_2(\rho(0))$.

Lemma 2.4.21. For ρ a return from s to $\mathrm{pop}_2(s)$ and for $i \in \mathrm{dom}(\rho)$ minimal such that $\rho(i) = \mathrm{pop}_1(s)$, the restriction $\rho\restriction_{[i,\ln(\rho)]}$ is a return from $\mathrm{pop}_1(s)$ to $\mathrm{pop}_2(s)$.

Proof. If ρ ends with a pop_2 transition, there is nothing to show. Now assume that ρ ends with a collapse operation. If $\mathrm{top}_2(s) \leq \mathrm{top}_2(\rho(\ln(\rho) - 1))$ then

$$\mathrm{top}_2(\mathrm{pop}_1(s)) < \mathrm{top}_2(s) \leq \mathrm{top}_2(\rho(\ln(\rho) - 1))$$

immediately yields the claim. The last possible case is that there is some $j \in \mathrm{dom}(\rho)$ such that $\rho\restriction_{[j,\ln(\rho)]}$ is a return of $\mathrm{pop}_1(s)$. Since $i \leq j$, this immediately implies the claim as $\rho\restriction_{[i,j]}$ is a run from $\mathrm{pop}_1(s)$ to $\mathrm{pop}_1(s)$ that never visits $\mathrm{pop}_2(s)$. But the class of returns is closed under prefixing by such runs. □

The previous observation gives rise to a classification of returns into low and high ones.

Definition 2.4.22. Let ρ be some return. We call ρ a low return, if there is some $i \in \mathrm{dom}(\rho)$ such that $\rho(i) = \mathrm{pop}_1(\rho(0))$. Otherwise we call ρ a high return.

Remark 2.4.23. Due to 2.4.21, a low return decomposes as a run to $\mathrm{pop}_1(\rho(0))$ followed by a return of $\mathrm{pop}_1(\rho(0))$. High returns never pass $\mathrm{pop}_1(\rho(0))$. Hence, low returns pass "lower" stacks than high returns.

In fact, the analysis of high returns and low returns is very similar. But there are small differences which provoke a lot of case distinctions when dealing with both types at the same time. In order to avoid these case distinctions, some of our lemmas will concentrate on high returns and we will only remark the differences to the case of low returns.

Next, we show that the notion $\#\mathsf{Ret}^k(w)$ (cf. Definition 2.4.17) is well-defined for every word w. For this purpose let us first introduce auxiliary notation.

Definition 2.4.24. The word $w{\downarrow_0}$ is obtained from $w \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^*$ by replacing every occurrence of $(\sigma, 2, j)$ in w by $(\sigma, 2, 0)$ for all $\sigma \in \Sigma$ and all $j \in \mathbb{N}$.

Remark 2.4.25. Later, it is important that $w{\downarrow_0}$ is a word over the finite alphabet $\Sigma \cup (\Sigma \times \{2\} \times \{0\})$.

Definition 2.4.26. Let s, s' be stacks such that $\mathsf{top}_2(s){\downarrow_0} = \mathsf{top}_2(s'){\downarrow_0}$. Let ρ be a return from (q_1, s) to $(q_2, \mathsf{pop}_2(s))$ and ρ' be a return from (q_1, s') to $(q_2, \mathsf{pop}_2(s'))$. We say ρ and ρ' are equivalent returns if they consist of the same sequence of transitions.

For an example, note that the returns ρ and ρ' in Figure 2.4 are equivalent. The crucial observation is that different stacks whose topmost words agree on their symbols and link levels have the same returns modulo this equivalence of returns.

Lemma 2.4.27. Let s and s' be stacks of width at least 2 such that $\mathsf{top}_2(s){\downarrow_0} = \mathsf{top}_2(s'){\downarrow_0}$. If ρ is a return from (q_1, s) to $(q_2, \mathsf{pop}_2(s))$ then there is an equivalent return ρ' from (q_1, s') to $(q_2, \mathsf{pop}_2(s'))$.

Proof. We assume that there is a symbol $\square \in \Sigma$ not occurring in any of the transitions of the pushdown system. Let s be some stack and $w := \mathsf{top}_2(s)$.

Now, we define $s' := w{\downarrow_0}\square : w{\downarrow_0}$. This definition is tailored towards the fact that s' is minimal with the following two properties.

1. The assumption that $\square$ does not appear within the transitions implies that an arbitrary run from s' to $\mathsf{pop}_2(s')$ is a return.[6]

2. $\mathsf{top}_2(s){\downarrow_0} = \mathsf{top}_2(s'){\downarrow_0}$.

In order to prove the lemma, it suffices to show that for every return starting in s there is an equivalent one starting in s' and vice versa.

The proof of this lemma is as follows. Let $t := \mathsf{pop}_2(s)$, $t' := \mathsf{pop}_2(s')$ and let $k := |t| - |t'| = |t| - 1$. Furthermore, let ρ be some return from (q, s) to $(\hat{q}, t)$. We define ρ' to be the largest run which starts in (q, s') and copies an initial part of ρ transition by transition.

We prove that $\mathsf{dom}(\rho') = \mathsf{dom}(\rho)$ by showing a stronger claim. For some word v let v^{-k} denote the word that is obtained from v by replacing every link l of level 2 by the link $l - k$.

Claim. The domains of ρ' and ρ agree. Furthermore, for each $i \in \mathsf{dom}(\rho)$ the following holds.

1. The states of $\rho(i)$ and $\rho'(i)$ agree

[6] This fact is not important for the proof of this lemma, but this fact gets important in the next lemmas.

2. The stack of $\rho(i)$ decomposes as $t : w_1 v_1 : w_2 v_2 : \cdots : w_n v_n$ and the stack of $\rho'(i)$ decomposes as $t' : w_1{\downarrow_0}v_1^{-k} : w_2{\downarrow_0}v_2^{-k} : \cdots : w_n{\downarrow_0}v_n^{-k}$ where the w_i are chosen in such a way that $w_i{\downarrow_0}$ is a maximal prefixes of $\mathsf{top}_2(s){\downarrow_0}$.

3. If the operation at i is a collapse of link level 2, then v_n is nonempty.

This claim can be proved by induction on the domain of ρ. This is tedious but straightforward. The construction of ρ' can be seen as the application of a prefix replacement as follows: $\rho' = \rho[t : \bot / t'\bot]$ where we manually repair the links of level 2.

The lemma then follows as a direct corollary of the claim: just note that the last operation of ρ is a collapse of level 2 or a pop_2 and yields a stack of width $|s| - 1$. In both cases it follows directly from the statements of the claim that the same transition is applicable to $\rho'(i)$ and results in a stack of width $|s| - 1 - |s| + 2 = 1$. Since ρ' never changed the first word of the stack, this stack is $t' = \mathsf{pop}_2(s')$. □

The previous lemma shows that $\#\mathsf{Ret}^k(w)$ is well defined (cf. Definition 2.4.17) and $\#\mathsf{Ret}^k(s) = \#\mathsf{Ret}^k(\mathsf{top}_2(s){\downarrow_0})$ for all stacks s of width at least 2. Thus, if we want to compute $\#\mathsf{Ret}^k(s)$, we can concentrate of the returns of a fixed stack with topmost word $w := \mathsf{top}_2(s){\downarrow_0}$. We will do this by choosing the stack $s' := w{\downarrow_0}\Box : w{\downarrow_0}$ to be the representative of any stack with $\mathsf{top}_2(s) = w$. s' is the smallest stack with topmost word $w{\downarrow_0}$ such that any run from s' to $\mathsf{pop}_2(s')$ is a return.

The following lemma contains the observation that every return from some stack s to $\mathsf{pop}_2(s)$ decomposes into parts that are prefixed by s and parts that are returns of stacks with topmost word $\mathsf{pop}_1(\mathsf{top}_2(s))$. This lemma shows that the decomposition of the returns in Example 2.4.18 can be generalised to decompositions of all returns.

Lemma 2.4.28. Let ρ be some high return of some stack s with topmost word $w := \mathsf{top}_2(s)$. Then there is a well-defined sequence

$$0 := j_0 < i_1 < j_1 < i_2 < j_2 < \cdots < i_n < j_n < i_{n+1} := \ln(\rho) - 1$$

with the following properties.

1. For $1 \leq k \leq n+1$, $s \trianglelefteq \rho{\restriction_{[j_{k-1}, i_k]}}$.

2. For all $1 \leq k \leq n$, $\mathsf{top}_2(\rho(i_k)) = w$ and the operation at i_k in ρ is a pop_1 or a collapse of level 1.

3. Either w is a proper prefix of $\mathsf{top}_2(\rho(i_{n+1}))$ and the operation at i_{n+1} is a collapse of level 2 or w is a prefix of $\mathsf{top}_2(\rho(i_{n+1}))$ and the operation at i_{n+1} is a pop_2.

4. For each $1 \leq k \leq n$, there is a stack s_k with $\mathsf{top}_2(s_k) = \mathsf{pop}_1(w)$ such that $\rho{\restriction_{[i_k+1, j_k]}}$ is a return from s_k to $\mathsf{pop}_2(s_k)$.

Remark 2.4.29. If ρ is a low return, a completely analogous lemma holds. We just have to omit i_{n+1}, i.e., the sequence ends with $j_n = \ln(\rho)$. Then statements 1, 2, and 4 hold for this sequence $0 := j_0 < i_1 < j_1 < i_2 < j_2 < \cdots < i_n < j_n = \ln(\rho)$.

Proof. Set $j_0 := 0$. Let $i_1 \in \mathsf{dom}(\rho)$ be the minimal position such that $s \trianglelefteq \rho(i_1)$ but $s \not\trianglelefteq \rho(i_1 + 1)$. If $i = \ln(\rho) - 1$ we set $n := 0$ and we are done: due to the definition of a high

return, the last operation is either a collapse of level 2 and $w < \mathsf{top}_2(\rho(i_1))$ or it is a pop_2 and $w \leq \mathsf{top}_2(\rho(i_1))$.

So let us assume that $i_1 < \mathsf{ln}(\rho) - 1$. By definition, $s \trianglelefteq \rho\restriction_{[j_0,i_1]}$.

Since ρ is a return of s and $i_1 + 1 < \mathsf{ln}(\rho))$, $|\rho(i_1 + 1)| \geq |s|$. Lemma 2.3.36 then implies that $\mathsf{top}_2(\rho(i_1)) = w$ and $\mathsf{top}_2(\rho(i_1 + 1)) = \mathsf{pop}_1(w)$.

Since $|\rho(i_1 + 1)| \geq |s| > |\rho(\mathsf{ln}(\rho))|$, there is some minimal j_1 such that $i_1 < j_1$ and $|\rho(j_1)| < |\rho(i_1)|$. We want to prove the following claim.

Claim. For s' the stack at $\rho(i_1 + 1)$, $\rho\restriction_{[i_1+1,j_1]}$ is a return from s' to $\mathsf{pop}_2(s')$.

First observe that by definition of j_1 for all $i_1 + 1 \leq k < j_1, |\rho(k)| \geq |s'|$. The operation at $j_1 - 1$ has to be a pop_2 or a $\mathsf{collapse}$ (of link level 2) because it decreases the width of the stack.

If it is pop_2, then we conclude that $\rho(j_1) = \mathsf{pop}_2(s')$ and the claim is satisfied.

Now, we consider the case that the operation before j_1 is a collapse of level 2. Since ρ is a high return, $\rho(i_1 + 1) \neq \mathsf{pop}_1(s)$. Thus, $|s'| > |s|$. Since $\mathsf{top}_2(s') = \mathsf{pop}_1(w)$, all elements in $\mathsf{top}_2(s')$ are clones of elements in the topmost word w of s. Thus, their level 2 links point to stacks t with $|t| < |s| < |s'|$. Heading for a contradiction, let us first assume that $\mathsf{top}_2(\rho(j_1 - 1))$ is a prefix of $\mathsf{top}_2(s')$, i.e., $\mathsf{top}_2(\rho(j_1 - 1)) \leq \mathsf{top}_2(s') < w$. In this case, $|\rho(j_1)| < |s|$ whence $j_1 = \mathsf{ln}(\rho)$ and the operation at $j_1 - 1$ is the last collapse operation in ρ. But $\mathsf{pop}_1(s)$ does not occur within ρ because ρ is a high return. We conclude that the last operation of ρ is $\mathsf{collapse}$, but neither $w < \mathsf{top}_2(\rho(j_1 - 1))$ nor a final segment of ρ is a return of $\mathsf{pop}_1(s)$. This contradicts the definition of a return.

Thus, we conclude that the topmost element of $\rho(j_1 - 1)$ was pushed onto the stack between $i_1 + 1$ and $j_1 - 1$. Since $|\rho(k)| \geq |s'|$ for all $\rho(k)$ with $i_1 + 1 \leq k \leq j_1 - 1$, the link of this element is at least $|s'| - 1$. But by definition of j this link also points below s', whence the link is $|s'| - 1$. But then $\rho\restriction_{[i,j]}$ satisfies all requirements of a return of s' and we are done.

This completes the claim.

Thus, we have obtained that i_1 and j_1 are candidates for the initial elements of the sequence required by the lemma. Note that the proof yields even more information. We have seen that $j_1 < \mathsf{ln}(\rho)$. Thus, $|\rho(i_1)| > |\mathsf{pop}_2(\rho(i_1))| = |\rho(j_1)| \geq |s|$. Lemma 2.3.37 implies that $s \trianglelefteq \rho(j_1)$ because $s \trianglelefteq \rho(i_1)$.

Hence, we can use the same arguments (restricted to $\rho\restriction_{[j_1,\mathsf{ln}(\rho)]}$) to show that for the minimal $i_2 > j_1$ such that $s \not\trianglelefteq \rho(i_2 + 1)$, we have $\mathsf{top}_2(\rho(i_2 + 1)) = \mathsf{pop}_1(w)$. By induction one concludes that the whole run ρ decomposes into parts prefixed by s and returns of stacks with topmost word $\mathsf{pop}_1(w)$ as desired. □

Remark 2.4.30. For low returns the proof is analogous. The only difference is the following. When defining inductively $0 = j_0 < i_1 < j_1 < \cdots < i_k$ at some point, we will obtain that $\rho(i_k) = \mathsf{pop}_1(s)$. In this case, we set $j_k := \mathsf{ln}(\rho)$. Lemma 2.4.21 shows that $\rho\restriction_{[i_k,j_k]}$ is a return. Thus, this definition satisfies the claim of the lemma for the case of low returns.

Lemma 2.4.31. In Lemma 2.4.28, the sequence $0 = j_0 < i_1 < j_1 < i_2 < j_2 < \cdots < i_{n+1}$ is uniquely defined by conditions 1 and 4: assume that there is another sequence

$$0 = l_0 < k_1 < l_1 < k_2 < l_2 < \cdots < k_{n+1} = \mathsf{ln}(\rho)$$

satisfying these conditions. Then $l_0 = j_0, i_1 = k_1, j_1 = l_1, \ldots, i_{n+1} = k_{n+1}$.

Proof. If $i_1 < k_1$ then $\rho\restriction_{[l_0,k_1]}$ contains $\rho(i_1)$ but $s \not\trianglelefteq \rho(i_1)$ which is a contradiction. If $k_1 < i_1$, we derive the contradiction $s \not\trianglelefteq \rho(k_1)$ analogously. Now, $l_1 = j_1$ follows from the fact that a return of $\rho(k_1)$ has to visit a stack s' with $|s'| < |\rho(k_1)|$ at its last position but it is not allowed to do so before. But j_1 is the minimal position where such a stack is reached whence $l_1 = j_1$. The claim follows by induction. □

Before we continue our analysis of returns, it is useful to fix an enumeration of all runs of a pushdown system.

Assumption 2.4.32. Let $\mathcal{S}$ be some pushdown system and Δ its transition relation. From now on, we assume that Δ is a linearly ordered set. Thus, all runs of $\mathcal{S}$ that start in a fixed configuration are well-ordered via the length-lexicographic ordering of the transitions that they use.

The rest of this section is concerned with the question "How can we determine $\#\mathrm{Ret}^k(s)$ for some collapsible pushdown system $\mathcal{S}$ using a finite automaton?". The technical tools that we use in order to answer this question are the notions of a return simulator and a simulation of a return. We start with an informal description. Afterwards, we precisely define these notions. A return simulator is a copy of the pushdown system $\mathcal{S}$ enriched by transitions that simulate each return of $\mathrm{pop}_1(s)$ in one transition. The simulation of a return from s to $\mathrm{pop}_2(s)$ is a return of this return simulator from the special stack

$$s' := \bot\top\mathrm{top}_1(s)\Box : \bot\top\mathrm{top}_1(s)$$

to $\mathrm{pop}_2(s')$. $\top$ is a new symbol representing $\mathrm{top}_2(\mathrm{pop}_1(s))$ and $\Box$ is a symbol not occurring in the transitions of the return simulator. $\Box$ is used to stop the computation once we reached $\mathrm{pop}_2(s')$. This guarantees that any run from s' to $\mathrm{pop}_2(s')$ is a return. Figure 2.10 shows the simulations of the return π_1 and π_3 from figures 2.7 and 2.8.

Before we introduce simulations and simulators formally, we want to explain the connection between a run and its simulation. For this purpose, we fix some notation. Let ρ be some return and ρ' its simulation (which is also a return). According to Lemma 2.4.28, there is a sequence

$$0 = j_0 < i_1 < j_1 < \ldots j_n \leq i_{n+1} = \ln(\rho) - 1$$

such that, for all $1 \leq k \leq n$, $s \trianglelefteq \rho\restriction_{[j_{k-1},i_k]}$ and $\rho\restriction_{[i_k+1,j_k]}$ is a return from some stack with topmost word $\mathrm{top}_2(\mathrm{pop}_1(s'))$. Analogously, there is a sequence

$$0 = j'_0 < i'_1 < j'_1 < \ldots j'_n \leq i'_{n+1} = \ln(\rho') - 1$$

such that, for $1 \leq k \leq n$, $s' \trianglelefteq \rho'\restriction_{[j'_{k-1},i'_k]}$ and $\rho'\restriction_{[i'_k+1,j'_k]}$ is a return from some stack with topmost word $\bot\top = \mathrm{top}_2(\mathrm{pop}_1(s'))$. The run ρ and its simulation ρ' are connected as follows:

$\rho'\restriction_{[j'_{k-1},i'_k]} = \rho\restriction_{[j_{k-1},i_k]}[s/s']$, i.e., $\rho'\restriction_{[j'_{k-1},i'_k]}$ copies $\rho\restriction_{[j_{k-1},i_k]}$ transition by transition but starts in a different stack. Furthermore, $\rho'\restriction_{[i'_k+1,j'_k]}$ is a return of length 1, i.e., it is a run that only consists of one pop_2 operation.

Thus, the simulation induces a decomposition of a run into those parts prefixed by its initial stack s and those parts that form returns which are equivalent to a return from $\mathrm{pop}_1(s)$ to $\mathrm{pop}_2(s)$. Using this decomposition we prove the inductive computability of $\#\mathrm{Ret}^k(s)$ from $\#\mathrm{Ret}^k(\mathrm{pop}_1(s))$. We first define the notion of a return simulator. Afterwards, we introduce the notion of a simulation of a return.

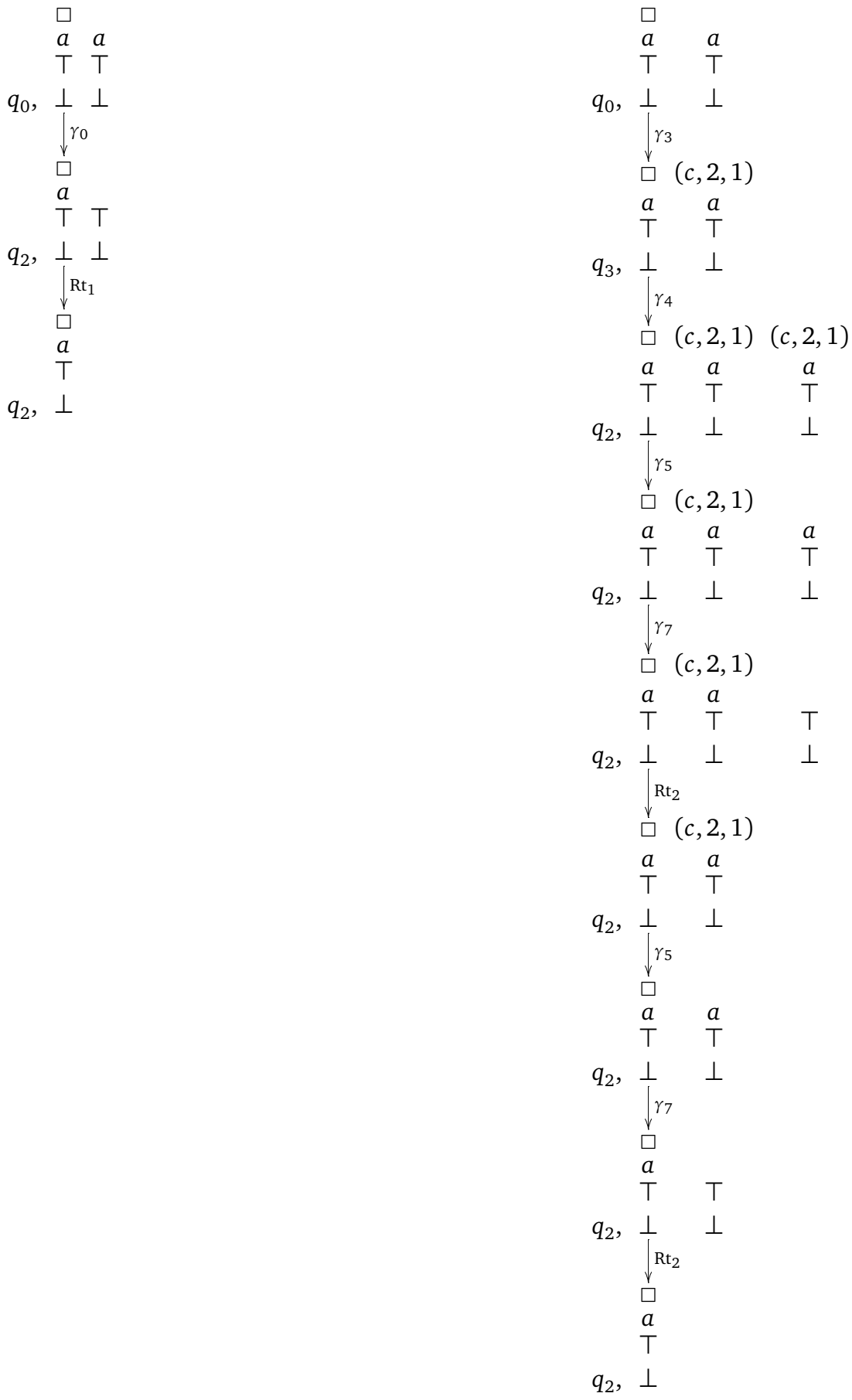


Figure 2.10.: Simulation of π_1 on the left and π_3 on the right.

Definition 2.4.33. Let $k \in \mathbb{N}$ be a threshold, $\mathscr{S} = (Q, \Sigma, \Gamma, q_0, \Delta)$ a collapsible pushdown system, and w some word. Let $s := w{\downarrow_0}\Box : w{\downarrow_0}$. The return simulator with respect to $(k, \mathscr{S}, s)$, denoted by $\mathrm{Rt}_s^k(\mathscr{S})$, is the tuple $(Q, \Sigma, \Gamma \cup \{\mathrm{Rt}_i : i \leq k\}, q_0, \hat{\Delta})$ where the $\mathrm{Rt}_i \notin \Gamma$ are new edge labels and

$$\hat{\Delta} := \Delta \cup \{(q_1, \top, \mathrm{Rt}_i, q_2, \mathrm{pop}_2) : i \leq \#\mathrm{Ret}^k(\mathrm{pop}_1(w))(q_1, q_2)\}.$$

We also use the notation $\mathrm{Rt}_\rho^k(\mathscr{S})$ for $\mathrm{Rt}_s^k(\mathscr{S})$ if ρ is a return starting at stack s.

Remark 2.4.34. Before we continue, let us make some remarks concerning this definition.

- The return simulator copies the behaviour of $\mathscr{S}$ as long as the topmost symbol of a stack is not $\top$.
- We consider $\top$ as an abbreviation for the word $\mathrm{pop}_1(\mathrm{top}_2(s))$. A run starting in the stack

  $$s' := \bot\top\mathrm{top}_1(s)\Box : \bot\top\mathrm{top}_1(s)$$

  reaches a stack with topmost symbol $\top$ if and only if the equivalent run that starts in s reaches a stack with topmost word $\mathrm{pop}_1(\mathrm{top}_2(s))$. Recall that a return of s always continues with a return if it reaches a stack with topmost word $\mathrm{pop}_1(\mathrm{top}_2(s))$.
- By definition $\#\mathrm{Ret}^k_{\mathrm{Rt}_s^k(\mathscr{S})}(\bot\top)$ agrees with $\#\mathrm{Ret}^k_{\mathscr{S}}(\mathrm{pop}_1(s))$. On topmost symbol $\top$ the applicable transitions of the return simulator are only pop_2 transitions. Hence, a return from $(q_1, \mathrm{pop}_1(s'))$ to $(q_2, \mathrm{pop}_2(s'))$ consists by definition of only one pop_2 transition. If $\#\mathrm{Ret}^k_{\mathscr{S}}(\mathrm{pop}_1(s))(q_1, q_2) = i$, then there are i such transitions which are labelled by $\mathrm{Rt}_1, \ldots, \mathrm{Rt}_i$. Each of these induces exactly one return whence $\#\mathrm{Ret}^k_{\mathrm{Rt}_s^k(\mathscr{S})}(\bot\top)(q_1, q_2) = i$.

The last two observations will lead to the result that the number of returns of $\mathscr{S}$ from s and the returns of the simulator from s' agree up to threshold k.

Definition 2.4.35. Let $\mathscr{S}$, s, s' and k as in Definition 2.4.33. We call any run of $\mathrm{Rt}_s^k(\mathscr{S})$ from (q_1, s') to $(q_2, \mathrm{pop}_2(s'))$ a simulation of a return from (q_1, s) to $(q_2, \mathrm{pop}_2(s))$.

Lemma 2.4.36. Let $\mathscr{S}, k, s$ and s' be as in Definition 2.4.33. If ρ is a simulation of a return from (q, s') to $(q', \mathrm{pop}_2(s'))$, then ρ is in fact a return of the return simulator.

Proof. Let i be minimal in $\mathrm{dom}(\rho)$ such that the stack at $\rho(i)$ is a substack of $\mathrm{pop}_2(s')$. Due to $|s'| = 2$, $\rho(i) = \mathrm{pop}_2(s')$. Furthermore $\mathrm{Sym}(\mathrm{pop}_2(s')) = \Box$. Since $\mathrm{Rt}_s^k(\mathscr{S})$ does not contain any transition of the form $(q_1, \Box, \gamma, q_2, \mathrm{op})$, $\rho(i)$ cannot be extended. Thus, $i = \ln(\rho)$.

Furthermore, $\mathrm{top}_2(s') = \bot\top\mathrm{top}_1(s)$. If $\mathrm{CLvl}(s) = 2$, then $\mathrm{CLnk}(s) = 0$ by definition of s. Thus, $\mathrm{top}_2(s')$ does not contain any defined level 2 link. One easily concludes that ρ is a return. □

In the following, we justify the term simulation of a return. To each simulation of the return simulator $\mathrm{Rt}_s^k(\mathscr{S})$ with initial state q and final state q' we associate a return from (q, s) to $(q', \mathrm{pop}_2(s))$.

Definition 2.4.37. Let $\mathscr{S}$ be a collapsible pushdown system, $k \in \mathbb{N}$ a threshold, and w some word. Let

$$s := w{\downarrow_0}\Box : w{\downarrow_0} \text{ and}$$
$$s' := \bot\top\mathsf{top}_1(s)\Box : \bot\top\mathsf{top}_1(s)$$

We define a function sTr_s that maps every run ρ' of $\mathrm{Rt}^k_s(\mathscr{S})$ from (q,s') to $(q',\mathsf{pop}_2(s'))$ to a return $\mathsf{sTr}_s(\rho')$ of $\mathscr{S}$ from (q,s) to $(q',\mathsf{pop}_2(s))$.

In order to explain sTr_s, we fix a run ρ' of $\mathrm{Rt}^k_s(\mathscr{S})$ from (q_1,s') to $(q_2,\mathsf{pop}_2(s'))$. Due to the previous lemma, ρ' is a return. We assume that it is a high return. Let

$$0 = j_0 < i_1 < j_1 < \cdots < j_n < i_{n+1}$$

be the sequence corresponding to ρ' according to Lemma 2.4.28. Without loss of generality, we assume that $n > 0$.

We set $\pi'_k := \rho'{\restriction_{[j_{k-1},i_k]}}$ for all $1 \le k \le n+1$ and $\rho'_k := \rho'{\restriction_{[i_k+1,j_k]}}$ for all $1 \le k \le n$. Now, we write $c'_k = (q'_k,s'_k)$ for the last configuration of π'_k and $\hat{c}'_k = (\hat{q}'_k,\hat{s}'_k)$ for the configuration following c'_k in ρ'. Lemma 2.4.28 implies the following.

1. $s' \trianglelefteq \pi'_k$ for all $1 \le k \le n$.
2. For all $1 \le k \le n$, $\mathsf{top}_2(s'_k) = \bot\top\mathsf{top}_1(s)$ and $\hat{s}'_k = \mathsf{pop}_1(s'_k)$. Thus, $\mathsf{top}_2(\hat{s}'_k) = \bot\top$.
3. $\mathsf{top}_2(s') \le \mathsf{top}_2(c'_{n+1})$ and c'_{n+1} is connected to $\hat{c}'_{n+1}$ via a pop_2 or $\mathsf{collapse}$ of level 2. In the latter case, $\mathsf{top}_2(s') < \mathsf{top}_2(c'_{n+1})$.
4. For $1 \le i \le n$, ρ'_k is a return from $\hat{c}'_k$ to $\mathsf{pop}_2(\hat{c}'_k)$. Thus, ρ'_k is a return of a stack with topmost word $\bot\top$.

We now define iteratively runs π_i, ξ_i and ρ_i whose composition then forms $\mathsf{sTr}_s(\rho')$.

Due to condition 1, $\pi_1 := \pi_1[s'/s]$ is well-defined. π_1 ends with stack $s_1 := s'_1[s'/s]$. Due to condition 2, $\mathsf{top}_2(s'_1) = \mathsf{top}_2(s')$ whence $\mathsf{top}_2(s_1) = w$. This implies that $\mathsf{top}_1(s'_1) = \mathsf{top}_1(w) = \mathsf{top}_1(s_1)$. Furthermore, the transition connecting $\rho'(i_1) = (q'_1,s'_1)$ with $\rho'(i_1+1) = (\hat{q}'_1,\hat{s}'_1)$ performs a pop_1 or a collapse of level 1. Let ξ_1 be the run of length 1 that applies this transition to the last configuration of π_1, i.e., ξ_1 is a run $(q'_1,s_1) \vdash (\hat{q}'_1,\mathsf{pop}_1(s_1))$.

Due to the observation in Remark 2.4.34, the form of ρ'_1 is $(\hat{q}'_1,\hat{s}'_1) \vdash^{\mathrm{Rt}_n} (\tilde{q}'_1,\mathsf{pop}_2(\hat{s}'_1))$ for some $n \le \#\mathrm{Ret}^k(\mathsf{pop}_1(s))(\hat{q}'_1,\tilde{q}'_1)$. Thus, we can define $\hat{\rho}_1$ to be the n-th return from $(\hat{q}'_1,\mathsf{pop}_1(s))$ to $(\tilde{q}'_1,\mathsf{pop}_2(s))$ in length-lexicographic order.

Recall that $\mathsf{top}_2(\mathsf{pop}_1(s_1)) = \mathsf{pop}_1(s)$ is the topmost word of the last configuration of ξ_1. Hence, there is a return ρ_1 that is equivalent to $\hat{\rho}_1$ and starts in the last configuration of ξ_1.

ρ_1 ends in configuration $(\tilde{q}'_1,\mathsf{pop}_2(s_1))$ where $\tilde{q}'_1$ is by definition the state of the initial configuration of π'_2. Furthermore the stack of π'_2 is $\mathsf{pop}_2(s'_1)$. Since $s' \trianglelefteq \pi'_2$, $s'_1[s'/s]$ is a well-defined stack. Due to $s_1 = s'_1[s'/s]$, we conclude that $\mathsf{pop}_2(s_1) = \mathsf{pop}_2(s'_1)[s'/s]$.

Thus, we can repeat this construction for $2,3,4,\ldots,n$ and obtain runs π_k, ξ_k, ρ_k such that $\pi_1 \circ \xi_1 \circ \rho_1 \circ \pi_2 \circ \xi_2 \circ \rho_2 \circ \cdots \circ \pi_n \circ \xi_n \circ \rho_n$ is a well-defined run from (q,s) to $\pi'_{n+1}(0)[s'/s]$.

We set $\pi_{n+1} := \pi_{n+1}[s'/s]$. Due to condition 3, $w \le s_{n+1} := s'_{n+1}[s'/s]$ which is the last stack of π_{n+1}. As in the cases $i \le n$, it follows that $\mathsf{top}_1(s_{n+1}) = \mathsf{top}_1(s'_{n+1})$. Thus, the

last transition δ of ρ' is also applicable to the last configuration of π_{n+1}. By definition, δ connects the last configuration of π'_{n+1} with $(q', \mathsf{pop}_2(s'))$. Since $|s'_{n+1}| = |s_{n+1}|$ and $\mathsf{CLvl}(s_{n+1}) = \mathsf{CLvl}(s'_{n+1})$, the application of this transition to the last configuration of π_{n+1} results in $(q', \tilde{s})$ where $\tilde{s}$ is a stack of width 1 such that $\tilde{s} = \mathsf{pop}_2^m(s_{n+1})$ for some $m \in \mathbb{N}$. But this is by definition $(q', w{\downarrow_0}\Box)$. Let ξ_{n+1} be the run that applies δ to the last configuration of π_{n+1}. We define

$$\mathsf{sTr}_s(\rho') := \pi_1 \circ \xi_1 \circ \rho_1 \circ \pi_2 \circ \xi_2 \circ \rho_2 \circ \cdots \circ \pi_n \circ \xi_n \circ \rho_n \circ \pi_{n+1} \circ \xi_{n+1}.$$

We say $\mathsf{sTr}_s(\rho')$ is the return simulated by ρ'.

Remark 2.4.38. For low returns ρ', $\mathsf{sTr}_s(\rho')$ is defined completely analogous. We define π_i, ξ_i, and ρ_i for all $1 \leq i \leq n$ as before. Then

$$\mathsf{sTr}_s(\rho') := \pi_1 \circ \xi_1 \circ \rho_1 \circ \pi_2 \circ \xi_2 \circ \rho_2 \circ \cdots \circ \pi_n \circ \xi_n \circ \rho_n.$$

Lemma 2.4.39. Let $s = w{\downarrow_0}\Box : w{\downarrow_0}$ and ρ' a simulation of a return as in the previous definition. Then $\mathsf{sTr}_s(\rho')$ is a return from s to $\mathsf{pop}_2(s)$.

Proof. By definition, $\mathsf{sTr}_s(\rho')$ is a run from s to $\mathsf{pop}_2(s)$ that does not pass any substack of $\mathsf{pop}_2(s)$ before its final configuration. If its last operation is pop_2 we are done.

Otherwise, the last operation is a $\mathsf{collapse}$ of level 2. Then we distinguish the following cases:

First consider the case that ρ' is a high return. Recall that by definition of π_{n+1}, we have that $\mathsf{top}_2(s)$ is a proper prefix of the topmost word of the last stack of π_{n+1}. But then the use of the last collapse in $\mathsf{sTr}_s(\rho')$ satisfies the restrictions from the definition of a return.

Now, consider the case that ρ' is a low return. By definition, $\mathsf{sTr}_s(\rho')$ ends with ρ_n. But ρ_n was defined to be a return from $\mathsf{pop}_1(s)$ to $\mathsf{pop}_2(s)$. Thus, $\mathsf{sTr}_s(\rho')$ is a return due to Lemma 2.4.21. □

Lemma 2.4.40. Let $s = w{\downarrow_0}\Box : w{\downarrow_0}$ as in Definition 2.4.37. Then sTr_s is injective.

Proof (Sketch). The proof is by contradiction. Assume that there are two runs ρ'_1 and ρ'_2 such that $\rho'_1 \neq \rho'_2$. We write $\rho_i := \mathsf{sTr}_s(\rho_i)$ for $i \in \{1, 2\}$.

Then there is a minimal $i' \in \mathsf{dom}(\rho'_1)$ such that the transition δ'_1 at position i' in ρ'_1 is not the transition δ'_2 at position i' in ρ'_2. Set $\pi' := \rho'_1{\restriction_{[0,i']}} = \rho'_2{\restriction_{[0,i']}}$. Now, π' induces a common initial segment π of ρ_1 and ρ_2 of length i. By this we mean that $\rho_1{\restriction_{[0,i]}} = \rho_2{\restriction_{[0,i]}}$ and that δ'_1 and δ'_2 determine the transition of ρ_1 and ρ_2 at position i.

We distinguish two cases.

1. Assume that $\mathsf{top}_1(\rho_1(i)) = \mathsf{top}_1(\rho_2(i)) \neq \top$. By definition of ρ_1 and ρ_2, this implies that the transition at i in ρ_j is δ'_j for $j \in \{1, 2\}$. Since $\delta'_1 \neq \delta'_2$, this implies that ρ_1 and ρ_2 differ in the transition applied at i whence $\rho_1 \neq \rho_2$.

2. Otherwise, assume that $\mathsf{top}_1(\rho_1(i)) = \mathsf{top}_1(\rho_2(i)) = \top$. Then we directly conclude that $\delta'_1 = (q, \top, \mathsf{Rt}_{j_1}, q'_1, \mathsf{pop}_2)$ and $\delta'_2 = (q, \top, \mathsf{Rt}_{j_2}, q'_2, \mathsf{pop}_2)$ where either $q'_1 \neq q'_2$ or $j_1 \neq j_2$.

   If $q'_1 \neq q'_2$, then there are $i_1 > i$ and $i_2 > i$ such that $\rho_1{\restriction_{[i,i_1]}}$ is a return from state q to state q'_1 while $\rho_2{\restriction_{[i,i_i]}}$ is a return from state q to state q'_2. Thus, $\rho_1 \neq \rho_2$.

Otherwise, $j_1 \neq j_2$ and $q_1' = q_2'$. By definition, ρ_1 continues with the j_1-th return from $\rho_1(i)$ to $(q_1', \mathrm{pop}_2(\rho_1(i)))$ and ρ_2 continues with the j_2-th return from $\rho_2(i) = \rho_1(i)$ to $(q_1', \mathrm{pop}_2(\rho_1(i)))$. Since $j_1 \neq j_2$, these returns differ whence the runs ρ_1 and ρ_2 differ. □

The last fact that we prove about sTr_s is a characterisation of its image. Consider a run ρ of $\mathrm{Rt}_s^k(\mathcal{S})$ in the domain of sTr_s. By definition of sTr_s, $\mathrm{sTr}_s(\rho)$ is a return from s to $\mathrm{pop}_2(s)$ that satisfies the following restriction: let ρ' be a subrun of sTr_s that is a return from some stack s' with $\mathrm{top}_2(s') = \mathrm{top}_2(\mathrm{pop}_1(s))$. Then ρ' is one of the k smallest returns of s' (with respect to length-lexicographic order).

The following lemma shows that this condition already defines the image of sTr_s. We only state the lemma for high returns, but for low returns the analogous statement holds.

Lemma 2.4.41. Let $\mathcal{S}$ be some collapsible pushdown system, s some stack of the form $s = w{\downarrow_0}\Box : w{\downarrow_0}$ and k some threshold. Furthermore, let ρ be a high return from (q,s) to $(q', \mathrm{pop}_2(s))$ of $\mathcal{S}$ and let

$$0 = j_0 < i_1 < j_1 < \cdots < i_n < j_n < i_{n+1}$$

be the sequence corresponding to ρ according to Lemma 2.4.28. Let $\rho_m := \rho{\restriction_{[i_m, j_m]}}$ for each $1 \leq m \leq n$. ρ_m is a return from some (q_m, s_m) to $(q_m', \mathrm{pop}_2(s_m))$. If for all $1 \leq m \leq n$, ρ_m is one of the m length-lexicographically smallest returns from (q_m, s_m) to $(q_m', \mathrm{pop}_2(s_m))$, then there is a run ρ' of $\mathrm{Rt}_s^k(\mathcal{S})$ such that $\mathrm{sTr}_s(\rho') = \rho$.

Proof. Let ρ be a high return from s to $\mathrm{pop}_2(s)$ satisfying the properties required in the lemma. For each $1 \leq m \leq n+1$ let δ_m be the transition connecting $\rho(i_m)$ and $\rho(i_m + 1)$. Furthermore, for $1 \leq m \leq n$ let l_m be the number such that $\rho{\restriction_{[i_m+1, j_m]}}$ is the l_m-th return from $\rho(i_m + 1)$ to $\rho(j_m)$ in length-lexicographic order. Set $s' := \bot\top\mathrm{top}_1(s)\Box : \bot\top\mathrm{top}_1(s)$. For $1 \leq m \leq n+1$, set $\pi_m' := \rho{\restriction_{[j_{m-1}, i_m]}}[s/s']$.

We define ρ' to be the run

$$\rho' := \pi_1' \circ \xi_1' \circ \rho_1' \circ \pi_2' \circ \cdots \circ \rho_n' \circ \pi_{n+1}' \circ \xi_{n+1}'$$

where ξ_m' applies δ_m to the last configuration of π_m' and ρ_m' applies an Rt_{l_m}-labelled transition to the last configuration of ξ_m'.

It is now easy to check that ρ' is a well defined run of $\mathrm{Rt}_s^k(\mathcal{S})$ from s' to $\mathrm{pop}_2(s)$ and that $\rho = \mathrm{sTr}_s(\rho')$. □

A corollary of the previous lemma is that there are at least as many returns of a pushdown system $\mathcal{S}$ from (q,s) to $(q', \mathrm{pop}_2(s))$ as there are runs of $\mathrm{Rt}_s^k(\mathcal{S})$ from (q,s') to $(q', \mathrm{pop}_2(s'))$ for $s' = \bot\top\mathrm{top}_1(s)\Box : \bot\top\mathrm{top}_1(s)$.

In fact, we want to prove that these two numbers agree up to threshold k. We obtain this result as a corollary of the following lemma. Again we only formulate the lemma for high returns, but the corresponding statement for low returns is proved analogously.

Lemma 2.4.42. Let $\mathcal{S}$ be a collapsible pushdown system and $s = w{\downarrow_0}\Box : w{\downarrow_0}$ for some word w. Let ρ be a return of $\mathcal{S}$ from (q,s) to $(q', \mathrm{pop}_2(s))$. Let

$$0 = j_0 < i_1 < j_1 < \cdots < i_n < j_n < i_{n+1}$$

be the sequence corresponding to ρ according to Lemma 2.4.28. If there is a $1 \leq k \leq n$ such that $\rho_k := \rho{\restriction_{[i_k+1, j_k]}}$ is not one of the minimal k returns from $\rho(i_k+1)$ to $\rho(j_k)$, then there are more than k returns of $\mathcal{S}$ from (q,s) to $(q', \mathrm{pop}_2(s))$.

Proof. Let $1 \leq k \leq n$ be a number such that $\rho_k := \rho\restriction_{[i_k+1,j_k]}$ satisfies the requirements of the lemma. Then there are k returns from $\rho(i_k+1)$ to $\rho(i_j)$ that are length-lexicographically smaller than ρ_k. Now, let $\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_k$ be an enumeration of these runs. For $1 \leq i \leq k$, the run $\pi_i := \rho\restriction_{[0,i_k+1]} \circ \hat{\rho}_i \circ \rho\restriction_{[j_k,i_{n+1}+1]}$ is a return from (q,s) to $(q', \mathrm{pop}_2(s))$. The π_i are pairwise distinct and distinct from ρ. Thus, there are at least $k+1$ returns from (q,s) to $(q', \mathrm{pop}_2(s))$. □

As a direct corollary of the previous two lemmas, we obtain that the runs of the return simulator and the returns from a configuration (q,s) to $(q', \mathrm{pop}_2(s))$ agree up to threshold k.

Corollary 2.4.43. Let $\mathscr{S}$ be a collapsible pushdown system, $k \in \mathbb{N}$ some threshold and w some word. For $s := w\downarrow_0 \Box : w\downarrow_0$ and $s' := \bot\top\mathrm{top}_1(s)\Box : \bot\top\mathrm{top}_1(s)$ and for all $q, q' \in Q$ let $M_{q,q'}$ be the set of runs of $\mathrm{Rt}^k_s(\mathscr{S})$ from (q,s') to $(q', \mathrm{pop}_2(s'))$. For all $q, q' \in Q$,

$$\#\mathrm{Ret}^k_{\mathscr{S}}(w)(q,q') = \min\{k, |M_{q,q'}|\} = \#\mathrm{Ret}^k_{\mathrm{Rt}^k_s(\mathscr{S})}(\mathrm{top}_2(s'))(q,q').$$

The last corollary shows that we can count simulations of a return simulator in order to calculate $\#\mathrm{Ret}^k_{\mathscr{S}}(s)$. Now, we use this result in order to obtain a proof of Proposition 2.4.19. Recall that this proposition asserts that there is a finite automaton $\mathscr{A}_{\mathrm{ret}}$ that calculates $\#\mathrm{Ret}^k(s)$ for each stack s on input $\mathrm{top}_2(s)\downarrow_0$.

Proof of Proposition 2.4.19. In the following, we define the finite automaton

$$\mathscr{A}_{\mathrm{ret}} := (Q_{\mathrm{ret}}, \Sigma \cup (\Sigma \times \{2\} \times \{0\}), a_0, \Delta_{\mathrm{ret}}).$$

Let $Q_{\mathrm{ret}} := \{a_0\} \cup \{0,1,\ldots,k\}^{Q \times Q}$ where Q is the set of states of $\mathscr{S}$ and a_0 is an extra initial state distinct from all other states. Thus, beside the initial state all functions from $Q \times Q$ to $\{0,1,\ldots,k\}$ are states of $\mathscr{A}_{\mathrm{ret}}$.

We define Δ_{ret} in such a way that the run of $\mathscr{A}_{\mathrm{ret}}$ on some word $w = \mathrm{top}_2(s)\downarrow_0$ ends in a state $a = \#\mathrm{Ret}^k(s)$. We compute the transitions of $\mathscr{A}_{\mathrm{ret}}$ iteratively.

We start with the transitions from the initial state. Recall that all words occurring in some stack start with the letter $\bot$. Thus, the only transition at a_0 should be of the form $(a_0, \bot, a)$ where a must satisfy $a = \#\mathrm{Ret}^k(\bot) = \#\mathrm{Ret}^k([\bot\Box : \bot])$. Due to Lemma 2.3.30, the value of a is computable.

Now, we repeat the following construction. Assume that for all reachable states $a \in Q_{\mathrm{ret}} \setminus \{a_0\}$ every path from a_0 to a is labelled by some word w such that $a = \#\mathrm{Ret}^k(w)$.

For each $\sigma \in \Sigma$, we want to compute the value $a' = \#\mathrm{Ret}^k(w\sigma)$. Let $s := w\downarrow_0\sigma\Box : w\downarrow_0\sigma$. Recall that $\mathrm{Rt}^k_s(\mathscr{S})$ is computable from $\mathscr{S}, a = \#\mathrm{Ret}^k(w)$, and k. Due to Lemma 2.3.30, we can count the number i_{q_1,q_2} of runs of $\mathrm{Rt}^k_s(\mathscr{S})$ from $(q_1, \bot\top\sigma\Box : \bot\top\sigma)$ to $(q_2, [\bot\top\sigma\Box])$ for each pair $q_1, q_2 \in Q$ up to threshold k. Finally, Corollary 2.4.43 shows that $i_{q_1,q_2} = \#\mathrm{Ret}^k(w\sigma)(q_1,q_2)$.

Thus, $a' := \#\mathrm{Ret}^k(w\sigma)$ is computable and we add the transition (a, σ, a') to Δ_{ret}. By induction hypothesis and by Corollary 2.4.43, all nonempty paths to some state $\hat{a}$ are now labelled by words v such that $\hat{a} = \#\mathrm{Ret}^k(v)$ (this can be proved by induction on the length of the path).

For words of the form $w(\sigma,2,0)$ the transitions $(a, (\sigma,2,0), a')$ are defined completely analogous.

After finitely many iterations of this process, we cannot add any new transitions to $\mathscr{A}_{\text{ret}}$. Then the construction of $\mathscr{A}_{\text{ret}}$ is finished.

We claim that the resulting automaton $\mathscr{A}_{\text{ret}}$ calculates $\#\text{Ret}^k(s)$ on input $\text{top}_2(s)\downarrow_0$ for every stack s.

The claim is proved by contradiction. Assume that there is some stack s such that there is no run of $\mathscr{A}_{\text{lp}}$ on $w := \text{top}_2(s)\downarrow_0$. By minimality there is a run on $\text{pop}_1(w)$. Now, we could add a transition from the final state of this run which is labelled by $\text{top}_1(w)$. This contradicts the assumption that we added all possible transitions.

Thus, there is a run of $\mathscr{A}_{\text{lp}}$ on $w := \text{top}_2(s)\downarrow_0$ for all stacks s. By construction, the run on w calculates $\#\text{Ret}^k(w)$. □

We conclude this section by proving Proposition 2.4.20. Recall that this proposition asserts the existence of a function $\text{BRL}_k^{\mathscr{S}}$ that bounds the length of the shortest returns of every stack. We first define $\text{BRL}_k^{\mathscr{S}}$, then we prove the properties asserted in the lemma.

Let $\mathscr{S}$ be some collapsible pushdown system and let $\mathscr{A}_{\text{ret}}$ be the corresponding finite automaton that calculates the returns of $\mathscr{S}$ up to threshold k. Recall that Δ_{ret} denotes the transition relation of $\mathscr{A}_{\text{ret}}$.

Recall that for each $\delta = (a, \tau, b) \in \Delta_{\text{ret}}$, it holds that $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \{0\})$ and $a, b : Q \times Q \to \{0, 1, \ldots, k\}$ are functions such that there is some word $w_\delta \in (\Sigma \cup (\Sigma \times \{2\} \times \{0\}))^+$ with $\#\text{Ret}^k(w_\delta) = a$ and $\#\text{Ret}^k(w_\delta \tau) = b$. In the following, we fix a w_δ for each $\delta \in \Delta_{\text{ret}}$. For each w_δ, we define the stack $s'_\delta := \bot\top\tau\square : \bot\top\tau$. Due to Corollary 2.4.43, there are (up to threshold k) $b(q_1, q_2)$ many simulations of returns from (q_1, s'_δ) to $(q_2, \text{pop}_2(s'_\delta))$. Using Lemma 2.3.30 we can compute the $b(q_1, q_2)$ many lexicographically smallest such simulations. We call these $\rho_1^{\delta,q_1,q_2}, \rho_2^{\delta,q_1,q_2}, \ldots, \rho_{b(q_1,q_2)}^{\delta,q_1,q_2}$.

Let

$$l^\delta_{q_1,q_2} := \max\left\{\ln(\rho_1^{\delta,q_1,q_2}), \ln(\rho_2^{\delta,q_1,q_2}), \ldots, \ln(\rho_{b(q_1,q_2)}^{\delta,q_1,q_2})\right\} \text{ and}$$
$$\#\top^\delta_{q_1,q_2} := \max\left\{|\{j \in \text{dom}(\rho_i^{\delta,q_1,q_2}) : \text{Sym}(\rho_i^{\delta,q_1,q_2}(j)) = \top\}| : 1 \le i \le b(q_1, q_2)\right\}$$

be the maximal length of any of these simulations and the maximal number of occurrences of $\top$ as topmost symbol in any of these returns, respectively. Now, set

$$l := \max\{l^\delta_{q_1,q_2} : q_1, q_2 \in Q, \delta \in \Delta_{\text{ret}}\} \text{ and}$$
$$\#\top := \max\{\#\top^\delta_{q_1,q_2} : q_1, q_2 \in Q, \delta \in \Delta_{\text{ret}}\}.$$

Definition 2.4.44. We define

$$\text{BRL}_k^{\mathscr{S}} : \mathbb{N} \to \mathbb{N} \text{ by}$$
$$\text{BRL}_k^{\mathscr{S}}(0) = 0 \text{ and}$$
$$\text{BRL}_k^{\mathscr{S}}(n+1) := l + \#\top \cdot \text{BRL}_k^{\mathscr{S}}(n).$$

Remark 2.4.45. The following idea underlies this definition. Assume that there is some word w of length $n-1$ such that the length of the shortest $\#\text{Ret}^k(w)(q_1, q_2)$ returns from $(q_1, w\square : w)$ to $(q_2, w\square)$ is bound by $\text{BRL}_k^{\mathscr{S}}(n-1)$. Furthermore, let s be a stack such that $w = \text{top}_2(\text{pop}_1(s))$.

Due to the definition of l, the lexicographically smallest $\#\mathrm{Ret}^k(s)(q,q')$ many returns from (q,s) to $(q',\mathrm{pop}_2(s))$ have simulations of length at most l.

Now, sTr_s translates these simulations into $\#\mathrm{Ret}^k(s)(q,q')$ many returns by copying all transitions one by one except for transitions on topmost symbol $\top$. The latter are replaced by lexicographically small returns equivalent to those from $(q_1, w\Box : w)$ to $(q_2, w\Box)$. Since this replacement happens at at most $\#\top$ many positions, we obtain $\#\mathrm{Ret}^k(s)(q,q')$ many returns from (q,s) to $(q',\mathrm{pop}_2(s))$ of length at most $l + \#\top \cdot \mathrm{BRL}_k^{\mathscr{S}}(n-1) = \mathrm{BRL}_k^{\mathscr{S}}(n)$.

Next, we prove Proposition 2.4.20.

Proof. Recall that we have to show the following two properties of $\mathrm{BRL}_k^{\mathscr{S}}$.

1. For each stack s, for all states q_1, q_2 and for $i := \#\mathrm{Ret}^k(s)(q_1,q_2)$, the length-lexicographically shortest returns $\rho_1, \ldots, \rho_i$ from (q_1,s) to $(q_2,\mathrm{pop}_2(s))$ satisfy $\ln(\rho_j) \leq \mathrm{BRL}_k^{\mathscr{S}}(|\mathrm{top}_2(s)|)$ for all $1 \leq j \leq i$.
2. If there is a return ρ from (q_1,s) to $(q_2,\mathrm{pop}_2(s))$ with $\ln(\rho) > \mathrm{BRL}_k^{\mathscr{S}}(|\mathrm{top}_2(s)|)$, then there are k returns from (q_1,s) to $(q_2,\mathrm{pop}_2(s))$ of length at most $\mathrm{BRL}_k^{\mathscr{S}}(|\mathrm{top}_2(s)|)$.

Note that the previous remark already contains a proof of the first part. The second part is proved by induction on k.

Let ρ be a return from (q_1,s) to $(q_2,\mathrm{pop}_2(s))$ with $\ln(\rho) > \mathrm{BRL}_k^{\mathscr{S}}(|\mathrm{top}_2(s)|)$. Then we conclude that $\#\mathrm{Ret}^k(s) \geq 1$. Due to the first statement, the lexicographically shortest return ρ_1 from (q_1,s) to $(q_2,\mathrm{pop}_2(s))$ satisfies $\ln(\rho_1) \leq \mathrm{BRL}_k^{\mathscr{S}}(|\mathrm{top}_2(s)|)$. Thus, $\rho_1 \neq \rho$ and we conclude that $\#\mathrm{Ret}^k(s) \geq 2$ (if $k \geq 2$).

We can iterate this argument k times and obtain $\rho_1, \rho_2 \ldots, \rho_k$ many short returns from (q_1,s) to $(q_2,\mathrm{pop}_2(s))$ as desired. □

2.4.4 Computing Loops

This section investigates the computability of loops. In fact, it lifts the results on returns to analogous results on loops. Again, we start by defining the functions we are interested in.

Definition 2.4.46. Let $\mathscr{S}$ be some collapsible pushdown system of level 2, $k \in \mathbb{N}$ some threshold and s some stack. We define

$$\#\mathrm{Loop}_{\mathscr{S}}^k(s) : Q \times Q \to \{0,1,\ldots,k\}$$
$$(q,q') \mapsto \begin{cases} i & \text{if there are exactly } i \leq k \text{ different loops of } \mathscr{S} \text{ from } (q,s) \text{ to } (q',s) \\ k & \text{otherwise.} \end{cases}$$

This function maps (q,q') to the number i of loops from (q,s) to (q',s) if $i \leq k$ and it maps (q,q') to k otherwise. In this sense k stands for the class of at least k loops.

Analogously, we define $\#\mathrm{LLoop}_{\mathscr{S}}^k(s) : Q \times Q \to \{0,1,\ldots,k\}$ to be the function that maps (q,q') to the number i of low loops from (q,s) to (q',s) if $i \leq k$ and that maps (q,q') to k otherwise.

Finally, we define $\#\mathrm{HLoop}_{\mathscr{S}}^k(s) : Q \times Q \to \{0,1,\ldots,k\}$ to be the function that maps (q,q') to the number i of high loops from (q,s) to (q',s) if $i \leq k$ and that maps (q,q') to k otherwise.

If $\mathscr{S}$ is clear from the context, we omit it and write $\#\mathrm{Loop}^k$ for $\#\mathrm{Loop}_{\mathscr{S}}^k$, etc.

Analogously to the theory of returns, we want to show that $\#\mathrm{Loop}^k$, $\#\mathrm{HLoop}^k$, and $\#\mathrm{LLoop}^k$ can be calculated by a finite automaton. Furthermore, we also want to prove bounds on the length of short loops analogously to Proposition 2.4.20. We start by stating these two propositions.

Proposition 2.4.47. There is an algorithm that, given a collapsible pushdown system $\mathcal{S}$ of level 2, computes a deterministic finite automaton $\mathcal{A}_{\mathrm{loop}}$ that computes $\#\mathrm{Loop}^k(s:w)$ on input $w{\downarrow_0}$.

In the same sense, there are automata that compute $\#\mathrm{HLoop}^k$ and $\#\mathrm{LLoop}^k$.

Proposition 2.4.48. There is an algorithm that computes on input some 2-CPG $\mathcal{S}$ and a natural number k a function $\mathrm{BLL}_k^{\mathcal{S}} : \mathbb{N} \to \mathbb{N}$ such that the following holds.

1. For every stack s, for $q_1, q_2 \in Q$ and for $i := \#\mathrm{Loop}^k(s)(q_1, q_2)$, the length-lexicographically shortest loops $\lambda_1, \ldots, \lambda_i$ from (q_1, s) to (q_2, s) satisfy

$$\ln(\lambda_j) \leq \mathrm{BLL}_k^{\mathcal{S}}(|\mathrm{top}_2(s)|)$$

   for all $1 \leq j \leq i$.

2. If there is a loop λ from (q_1, s) to (q_2, s) with $\ln(\lambda) > \mathrm{BLL}_k^{\mathcal{S}}(|\mathrm{top}_2(s)|)$, then there are k loops from (q_1, s) to $(q_2, \mathrm{pop}_2(s))$ of length at most $\mathrm{BLL}_k^{\mathcal{S}}(|\mathrm{top}_2(s)|)$.

Analogously, there are functions $\mathrm{BHLL}_k^{\mathcal{S}} : \mathbb{N} \to \mathbb{N}$ and $\mathrm{BLLL}_k^{\mathcal{S}} : \mathbb{N} \to \mathbb{N}$ that satisfy the same assertions but for the set of high loops or low loops, respectively.

Before we prove these propositions, we present two corollaries of the previous Proposition that play a crucial role in Section 3.3.

Corollary 2.4.49. Let $\mathcal{S}$ be some level 2 collapsible pushdown system. Furthermore, let (q, s) be some configuration and $\rho_1, \ldots, \rho_n$ be pairwise distinct runs from the initial configuration to (q, s). There is a run $\hat{\rho}_1$ from the initial configuration to (q, s) such that the following holds.

1. $\hat{\rho}_1 \neq \rho_i$ for $2 \leq i \leq n$ and
2. $\ln(\hat{\rho}_1) \leq 2 \cdot \mathrm{wdt}(s) \cdot \mathrm{hgt}(s)(1 + \mathrm{BLL}_n^{\mathcal{S}}(\mathrm{hgt}(s)))$.

Proof. If $\ln(\rho_1) \leq 2 \cdot |s| \cdot \mathrm{hgt}(s)(1 + \mathrm{BLL}_n^{\mathcal{S}}(\mathrm{hgt}(s)))$, set $\hat{\rho}_1 := \rho_1$ and we are done. Assume that this is not the case. Due to Lemmas 2.4.3 and 2.4.4, ρ_1 decomposes as

$$\rho_1 = \lambda_0 \circ \mathrm{op}_1 \circ \lambda_1 \circ \cdots \circ \lambda_{m-1} \circ \mathrm{op}_m \circ \lambda_m$$

where every λ_i is a loop and every op_i is a run of length 1 such that $m \leq 2 \cdot |s| \cdot \mathrm{hgt}(s)$. Proposition 2.4.48 implies the following: If $\ln(\lambda_i) > \mathrm{BLL}_n^{\mathcal{S}}(\mathrm{hgt}(s)))$, then there are n loops from $\lambda(0)$ to $\lambda(\ln(\lambda))$ of length at most $\mathrm{BLL}_n^{\mathcal{S}}(\mathrm{hgt}(s)))$. At least one of these can be plugged into the position of λ_i such that the resulting run does not coincide with any of the $\rho_2, \rho_3, \ldots, \rho_n$. In other words, there is some loop λ_i' of length at most $\mathrm{BLL}_n^{\mathcal{S}}(\mathrm{hgt}(s)))$ such that

$$\hat{\rho}_1 := \lambda_0 \circ \mathrm{op}_1 \circ \lambda_1 \circ \cdots \circ \mathrm{op}_i \circ \lambda_i' \circ \mathrm{op}_{i+1} \circ \lambda_{i+1} \circ \cdots \circ \lambda_{m-1} \circ \mathrm{op}_m \circ \lambda_m$$

is a run to (q, s) distinct from $\rho_2, \rho_3 \ldots, \rho_n$ and shorter than ρ_1. Iterated replacement of large loops results in a run ρ_1' with the desired properties. □

Now, we state a second corollary that is quite similar to the previous one but deals with runs of a different form.

Corollary 2.4.50. Let $\hat{\rho}_1, \hat{\rho}_2, \ldots \hat{\rho}_n$ be runs from the initial configuration to some configuration (q,s). Furthermore, let w be some word and $\rho_1, \rho_2, \ldots \rho_n$ be runs from (q,s) to $(q',s:w)$ that do not visit proper substacks of s. If $\hat{\rho}_1 \circ \rho_1, \hat{\rho}_2 \circ \rho_2, \ldots, \hat{\rho}_n \circ \rho_n$ are pairwise distinct, then there is a run ρ_1' from (q,s) to $(q',s:w)$ that satisfies the following.

1. ρ_1'does not visit a proper substack of s,
2. $\ln(\rho_1') \leq 2 \cdot \mathrm{hgt}(s:w) \cdot (1 + \mathrm{BLL}_n^{\mathscr{S}}(\mathrm{hgt}(s:w)))$, and
3. $\hat{\rho}_1 \circ \rho_1'$ is distinct from each $\hat{\rho}_i \circ \rho_i$ for $2 \leq i \leq n$.

Proof. It is straightforward to see that ρ_1 decomposes as

$$\rho_1 = \lambda_0 \circ \mathrm{op}_1 \circ \lambda_1 \circ \cdots \circ \lambda_{m-1} \circ \mathrm{op}_m \circ \lambda_m$$

where every λ_i is a loop and every op_i is a run of length 1 such that $m \leq 2 \cdot \mathrm{hgt}(s:w)$. We then proceed completely analogous to the previous corollary. □

We now come to the proofs of the main propositions on loops. The proofs of these two propositions are analogous to the proofs for the return case. The reader who is not interested in these rather technical proofs should skip the rest of this section and continue reading Section 2.5.

We now prepare the proofs of the two main propositions on loops. Analogously to the return case, the first important observation is that $\#\mathrm{Loop}^k(s)$, $\#\mathrm{HLoop}^k(s)$, and $\#\mathrm{LLoop}^k(s)$ only depend on the symbols and link levels of the topmost word of the stack s. In order to show this, we first define the notion of equivalent loops analogously to the notion of equivalent returns in Definition 2.4.26.

Definition 2.4.51. Let s, s' be stacks such that $\mathrm{top}_2(s)\downarrow_0 = \mathrm{top}_2(s')\downarrow_0$. Let λ be a loop from (q_1,s) to (q_2,s) and λ' be a loop from (q_1,s') to (q_2,s'). We say λ and λ' are equivalent loops if they consist of the same sequence of transitions.

The crucial observation is that different stacks whose topmost words agree on their symbols and link levels have the same loops modulo this equivalence relation.

Lemma 2.4.52. Let s and s' be stacks such that $\mathrm{top}_2(s)\downarrow_0 = \mathrm{top}_2(s')\downarrow_0$. If λ is a loop from (q_1,s) to (q_2,s) then there is an equivalent loop λ' from (q_1,s') to (q_2,s').

Remark 2.4.53. The proof of this lemma is analogous to the proof of Lemma 2.4.27. Furthermore, it is straightforward to see that a low loop can only be equivalent to a low loop (analogously, a high loops can only be equivalent to a high loop).

Since the lemma shows that loops of a given stack only depend on its topmost word, it is a meaningful concept to speak about the loops of some word.

Definition 2.4.54. For w some word, let $\#\mathrm{Loop}^k(w)$ be $\#\mathrm{Loop}^k(s)$ for some stack s with $\mathrm{top}_2(s) = w$. Analogously, we define the notions $\#\mathrm{HLoop}^k(w)$ and $\#\mathrm{LLoop}^k(w)$.

The next step towards the proof of our main propositions is a characterisation of $\#\mathsf{Loop}^k(w)$ in terms of $\#\mathsf{Loop}^k(\mathsf{pop}_1(w))$ and $\#\mathsf{Ret}^k(\mathsf{pop}_1(w))$ analogously to the result of Lemma 2.4.28 for returns. We do this in the following three lemmas. First, we present a unique decomposition of loops into high and low loops. Afterwards, we characterise low loops and high loops.

Lemma 2.4.55. Let λ be a loop from (q,s) to (q',s). λ is either a high loop or it has a unique decomposition as $\lambda = \lambda_0 \circ \lambda_1 \circ \lambda_2$ where λ_0 and λ_2 are high loops and λ_1 is a low loop.

Proof. Assume that λ is no high loop. Since it is a loop, it visits $\mathsf{pop}_1(s)$ at some position. Let $i \in \mathsf{dom}(\lambda)$ be the minimal position just before the first occurrence of $\mathsf{pop}_1(s)$ and $j \in \mathsf{dom}(\lambda)$ be the position directly after the last occurrence of $\mathsf{pop}_1(s)$. It is straightforward to see that $\lambda\restriction_{[i+1,j-1]}$ is by definition a loop of $\mathsf{pop}_1(s)$ and the initial and final part of λ are loops of s. We conclude by noting that $\lambda\restriction_{[i,j]}$ is then a low loop of s. □

Remark 2.4.56. An important consequence of this lemma is the fact that $\#\mathsf{Loop}^k(s)$ is determined by $\#\mathsf{HLoop}^k(s)$ and $\#\mathsf{LLoop}^k(s)$. $\#\mathsf{Loop}^k(s)(q,q')$ counts the high loops from (q,s) to (q',s) and those loops that consists of a high loop from (q,s) to $(\hat{q},s)$ followed by a low loop from $(\hat{q},s)$ to $(\hat{q}',s)$ followed by a loop from $(\hat{q}',s)$ to (q',s). Thus, writing $H(s)$ for $\#\mathsf{HLoop}^k(s)$ and $L(s)$ for $\#\mathsf{LLoop}^k(s)$, we obtain that

$$\#\mathsf{Loop}^k(s)(q,q') = \min\left\{ k, H(s)(q,q') + \sum_{\hat{q},\hat{q}' \in Q} H(s)(q,\hat{q}) \cdot L(s)(\hat{q},\hat{q}') \cdot H(s)(\hat{q}',q') \right\}.$$

In the following, we first explain how low loops depend on the loops of smaller stacks, afterwards we explain how high loops depend on returns of smaller stacks.

Lemma 2.4.57. Let λ be a low loop starting and ending in stack s. Then $\lambda\restriction_{[1,\mathsf{ln}(\lambda)-1]}$ is a loop starting and ending in $\mathsf{pop}_1(s)$. The operation at 0 is a pop_1 or a $\mathsf{collapse}$ of level 1. The operation at $\mathsf{ln}(\lambda)-1$ is a push_σ where $\mathsf{top}_1(s) = \sigma \in \Sigma$.

Proof. Let $\mathsf{top}_1(s) = \sigma$. The lemma follows directly from the observation that a push_σ transition followed by a loop of s followed by a pop_1 or $\mathsf{collapse}$ forms a loop of $\mathsf{pop}_1(s)$. □

The following lemma provides the analysis of high loops. Every high loop decomposes into parts that are prefixed by its initial stack s and parts that are returns of stacks with topmost word $\mathsf{pop}_1(\mathsf{top}_2(s))$. Note the similarity of this characterisation and its proof with the characterisation of returns in Lemma 2.4.28.

Lemma 2.4.58. Let λ be some high loop of some stack s with topmost word $w = \mathsf{top}_2(s)$. Then there is a sequence $0 =: j_0 < i_1 < j_1 < i_2 < j_2 < \cdots < i_n < j_n \leq i_{n+1} := \mathsf{ln}(\lambda)$ such that

1. for $1 \leq k \leq n+1$, $s \trianglelefteq \lambda\restriction_{[j_{k-1},i_k]}$ and
2. for each $1 \leq k \leq n$, there is a stack s_k with $\mathsf{top}_2(s_k) = \mathsf{pop}_1(w)$ such that $\lambda\restriction_{[i_k+1,j_k]}$ is a return of s_k.

Proof. This is completely analogous to the proof of Lemma 2.4.28. Assume that λ is a high loop and let i_1 be the minimal position in $\mathsf{dom}(\lambda)$ such that $s \not\trianglelefteq \lambda(i_1+1)$. For exactly the same reasons as in the return case, $\mathsf{top}_2(\lambda(i_1+1)) = \mathsf{pop}_1(w)$. Since λ is a high loop, $\lambda(i_1+1) \neq \mathsf{pop}_1(s)$ whence $|\lambda(i_1+1)| > |s|$. Since λ ends in stack s, there is a minimal $j_1 > i_1+1$ with $|\lambda(j_1)| < |\lambda(i_1+1)|$. Now, completely analogous to the return case one concludes that $\lambda{\restriction}_{[i_1+1,j_1]}$ is a return: just note that all level 2 links in $\mathsf{top}_2(\lambda(i_1+1))$ are clones of $\mathsf{top}_2(s)$ whence they point to stacks of width smaller than s. By definition of a loop, λ cannot use any of these links. Thus, it is clear that $\lambda(j_1) = \mathsf{pop}_2(\lambda(i_1+1))$. Furthermore, it is easy to see that $s \trianglelefteq \lambda(j_1)$. Thus, an inductive definition of the i_k and j_k provides a proof of the lemma. □

Remark 2.4.59. Completely analogous to the decomposition of returns, one proves that the sequence $j_0 < i_1 < \cdots < j_n \leq i_{n+1}$ is unique.

Having obtained this decomposition of high loops we show that the computation of $\#\mathsf{HLoop}^k$ can be done analogously to the computation of $\#\mathsf{Ret}^k$: we use certain runs of the return simulator $\mathrm{Rt}^k_s(\mathscr{S})$ as simulations of high loops with initial and final stack s.

Definition 2.4.60. Let $\mathscr{S}$ be a collapsible pushdown system of level 2, $k \in \mathbb{N}$ some threshold, and s some stack of the form $s = w{\downarrow_0}\Box : w{\downarrow_0}$. We set $s' := \bot\top\mathsf{top}_1(s)\Box : \bot\top\mathsf{top}_1(s)$. We call any run of $\mathrm{Rt}^k_s(\mathscr{S})$ from (q_1, s') to (q_2, s') a simulation of a high loop from (q_1, s) to (q_2, s).

This terminology is justified for the same reasons as in the case of returns. Analogously, to the function sTr_s, we next define a function sTl_s that translates simulations of loops into loops with initial and final stack s.

Definition 2.4.61. Let $\mathscr{S}, k$, s and s' be as in the previous definition. Let λ' be a simulation of a high loop from (q_1, s') to (q_2, s') where $q_1, q_2 \in Q$. Due to the definition of the return simulator $\mathrm{Rt}^k_s(\mathscr{S})$, the run λ' cannot pass any substack of $\mathsf{pop}_1(s')$ (there are no transitions that allow to return to s' once the run reaches $\mathsf{pop}_1(s')$ or $\mathsf{pop}_2(s')$). Thus, λ' is a high loop and there is a sequence $j_0 < i_1 < j_1 < \cdots < j_n \leq i_{n+1}$ according to Lemma 2.4.58.

We set $\pi'_k := \lambda'{\restriction}_{[j_{k-1},i_k]}$ for all $1 \leq k \leq n+1$ and $\rho'_k := \lambda'{\restriction}_{[i_k+1,j_k]}$ for all $1 \leq k \leq n$.

Completely analogous to what we did in Definition 2.4.37 we can define runs π_k, ξ_k, and ρ_k and set

$$\mathsf{sTl}_s(\rho') := \pi_1 \circ \xi_1 \circ \rho_1 \circ \pi_2 \circ \xi_2 \circ \rho_2 \circ \cdots \circ \pi_n \circ \xi_n \circ \rho_n \circ \pi_{n+1}.$$

We say $\mathsf{sTl}_s(\lambda')$ is the high loop simulated by λ'.

We omit the details of the following claims because they are completely analogous to the return case. One can show that sTl_s is an injective function (cf. Lemma 2.4.40). The image of sTl_s contains exactly all those high loops with initial and final stack s that use length-lexicographic small returns of stacks with topmost word $\mathsf{top}_2(\mathsf{pop}_1(s))$ in the following sense. Let λ be in the image of sTl_s. If λ contains a subrun that is a return from $(q_1, \hat{s})$ to $(q_2, \mathsf{pop}_2(\hat{s}))$ with $\mathsf{top}_2(\hat{s}) = \mathsf{pop}_1(\mathsf{top}_2(s))$ then this subrun is equivalent to one of the k length-lexicographically smallest returns from $(q_1, \mathsf{pop}_1(s))$ to $(q_2, \mathsf{pop}_2(s))$. The proof of this claim is analogous to the proof of Lemma 2.4.41. Finally, if there is a high loop from (q, s) to (q', s) that is not the image of sTl_s, then there are k high loops from (q, s) to (q', s) in the image of sTl_s (cf. Lemma 2.4.42).

Analogous to Lemma 2.4.43, these results imply that the number of simulations of high loops is up to threshold k the number of high loops.

Corollary 2.4.62. Let $\mathcal{S}$ be a collapsible pushdown system, $k \in \mathbb{N}$ some threshold and w some word. For $s := w\downarrow_0 \Box : w\downarrow_0$, $s' := \bot\top\mathrm{top}_1(s)\Box : \bot\top\mathrm{top}_1(s)$ and for $q, q' \in Q$, let $M_{q,q'}$ be the set of runs of $\mathrm{Rt}^k_s(\mathcal{S})$ from (q, s') to (q', s'). For all $q, q' \in Q$,

$$\#\mathrm{HLoop}^k_{\mathcal{S}}(w)(q,q') = \min\{k, |M_{q,q'}|\} = \#\mathrm{HLoop}^k_{\mathrm{Rt}^k_s(\mathcal{S})}(\mathrm{top}_2(s'))(q,q').$$

We will soon see that this corollary can be used to define a finite automaton that computes $\#\mathrm{HLoop}^k$. But before we come to this result, we briefly examine how we can compute the number of low loops of a given stack.

In Lemma 2.4.57 we proved that the number of low loops of a stack s depends on the number of loops of $\mathrm{pop}_1(s)$. The following lemma shows how we can use this dependence in order to compute $\#\mathrm{LLoop}^k(s)$ from $\#\mathrm{Loop}^k(\mathrm{pop}_1(s))$.

Lemma 2.4.63. Let $\mathcal{S}$ be some collapsible pushdown system of level 2, $k \in \mathbb{N}$ some threshold. There is a function that computes $\#\mathrm{LLoop}^k(w)$ on input $\mathrm{Sym}(w)$, $\mathrm{CLvl}(w)$, $\mathrm{Sym}(\mathrm{pop}_1(w))$, $\#\mathrm{Loop}^k(\mathrm{pop}_1(w))$ and the transition relation Δ of $\mathcal{S}$.

Proof. A low loop from (q,s) to (q',s) with $w = \mathrm{top}_2(s)$ can only exists if $\mathrm{CLvl}(w) = 1$. Hence, we only have to consider the case $\mathrm{CLvl}(w) = 1$. Due to Lemma 2.4.57, a low loop from (q,s) to (q',s) starts with a pop_1 or collapse (of level 1) and ends with $\mathrm{push}_{\mathrm{Sym}(w)}$. Between these two transitions, the low loop performs a loop of $\mathrm{pop}_1(s)$. We set

$$M_{q_1,q_2} := \{(q_1, \mathrm{Sym}(w), \gamma, q_2, \mathrm{op}) \in \Delta : \mathrm{op} = \mathrm{pop}_1 \text{ or } \mathrm{op} = \mathrm{collapse}\} \text{ and}$$
$$N_{q_1,q_2} := \{(q_1, \mathrm{Sym}(\mathrm{pop}_1(w)), \gamma, q_2, \mathrm{push}_{\mathrm{Sym}(w)}) \in \Delta\}.$$

Then, $\#\mathrm{LLoop}^k(s)(q,q') = \min\left\{k, \sum_{\hat{q},\hat{q}' \in Q} |M_{q,\hat{q}}| \cdot \#\mathrm{Loop}^k(s)(\hat{q},\hat{q}') \cdot |N_{\hat{q}',q'}|\right\}$. □

By now, we are prepared to prove Proposition 2.4.47. Recall that we have to provide automata that calculate $\#\mathrm{Loop}^k$, $\#\mathrm{HLoop}^k$ and $\#\mathrm{LLoop}^k$. In fact, we provide one automaton that calculates these functions and $\#\mathrm{Ret}^k$ at the same time.

Proof of Proposition 2.4.47. Let $\mathcal{S} = (Q, \Sigma, \Gamma, q_0, \Delta)$ be a collapsible pushdown system of level 2.

We want to define a finite automaton $\mathcal{A}_{\mathrm{lp}} := (Q_{\mathrm{lp}}, \Sigma \cup (\Sigma \times \{2\} \times \{0\}), a_0, \Delta_{\mathrm{lp}})$ that computes $\#\mathrm{Ret}^k(s), \#\mathrm{HLoop}^k(s), \#\mathrm{LLoop}^k(s)$, and $\#\mathrm{Loop}^k(s)$ on input $w := \mathrm{top}_2(s)\downarrow_0$.

Recall the following facts.

1. $\#\mathrm{Ret}^k(s) = \#\mathrm{Ret}^k(w)$, $\#\mathrm{HLoop}^k(s) = \#\mathrm{HLoop}^k(w)$, etc.
2. Due to Proposition 2.4.19, $\#\mathrm{Ret}^k(w)$ is computable by some automaton.
3. Due to the proof of corollary 2.4.62, we can compute a function f such that for all words w and all $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$, we have

$$\#\mathrm{HLoop}^k(w\tau) = f(\#\mathrm{Ret}^k(w), \mathrm{Sym}(\tau), \mathrm{CLvl}(\tau)).$$

4. Due to Lemma 2.4.63, we can compute a function g such that

$$\#\mathrm{LLoop}^k(w\tau) = g(\mathrm{Sym}(\tau), \mathrm{CLvl}(\tau), \mathrm{Sym}(w), \#\mathrm{Loop}^k(w)).$$

5. Due to Remark 2.4.56 there is a function h such that

$$\#\mathrm{Loop}^k(w\tau) = h(\#\mathrm{HLoop}^k(w\tau), \#\mathrm{LLoop}^k(w\tau)).$$

Analogously to the proof of Proposition 2.4.19, we can use these observations in order to define an automaton that computes $\#\mathrm{Ret}^k(s), \#\mathrm{HLoop}^k(s), \#\mathrm{LLoop}^k(s)$, and $\#\mathrm{Loop}^k(s)$ on input $w := \mathrm{top}_2(s)\downarrow_0$. □

Finally, we have to prove Proposition 2.4.48. Analogously to the case of returns, we first define functions $\mathrm{BLL}_k^{\mathscr{S}}$, $\mathrm{BHLL}_k^{\mathscr{S}}$, and $\mathrm{BLLL}_k^{\mathscr{S}}$. Then we show that these functions satisfy the conditions of Proposition 2.4.48. We first prepare the definition of $\mathrm{BHLL}_k^{\mathscr{S}}$. Afterwards, we define the functions mentioned above.

Let $\mathscr{S}$ be some collapsible pushdown system and $\mathscr{A}_{\mathrm{lp}}$ the corresponding finite automaton that calculates the returns, high loops, low loops, and loops of $\mathscr{S}$. Recall that we write Δ_{lp} for the transition relation of $\mathscr{A}_{\mathrm{lp}}$.

Recall that the transitions are labelled by elements of $\Sigma \cup (\Sigma \times \{2\} \times \{0\})$. The run of the automaton on some word w over this alphabet leads to a state a such that a encodes $\mathrm{Sym}(w), \#\mathrm{Ret}^k(w), \#\mathrm{HLoop}^k(w)$, etc.

For each transition $\delta(a, \tau, b)$ we fix a word w_δ such that the run on w_δ ends in state a.

For each w_δ, we define the stack $s'_\delta := \bot\top\tau\Box : \bot\top\tau$. Due to Corollary 2.4.43, there are (up to threshold k) $\#\mathrm{HLoop}^k_{\mathscr{S}}(s'_\delta)(q_1, q_2)$ many simulations of high loops from (q_1, s'_δ) to (q_2, s'_δ). Using Lemma 2.3.30 we can compute the $\#\mathrm{HLoop}^k_{\mathscr{S}}(s'_\delta)(q_1, q_2)$ many lexicographically smallest such simulations. We call these $\lambda_1^{\delta,q_1,q_2}, \lambda_2^{\delta,q_1,q_2}, \ldots, \lambda_{\#\mathrm{HLoop}^k(s'_\delta)(q_1,q_2)}^{\delta,q_1,q_2}$. Let

$$l^\delta_{q_1,q_2} := \max\left\{\ln(\lambda_1^{\delta,q_1,q_2}), \ln(\lambda_2^{\delta,q_1,q_2}), \ldots, \ln(\lambda_{\#\mathrm{HLoop}^k(s'_\delta)(q_1,q_2)}^{\delta,q_1,q_2})\right\} \text{ and}$$
$$\#\top^\delta_{q_1,q_2} := \max\left\{\left|\left\{j \in \mathrm{dom}(\lambda_i^{\delta,q_1,q_2}) : \mathrm{Sym}(\lambda_i^{\delta,q_1,q_2}(j)) = \top\right\}\right| :\leq \#\mathrm{HLoop}^k(s'_\delta)(q_1, q_2)\right\}$$

be the maximal length of any of these simulations and the number of occurrences of $\top$ as topmost symbol in any of these simulations, respectively. Now, set

$$l := \max\{l^\delta_{q_1,q_2} : q_1, q_2 \in Q, \delta \in \Delta_{\mathrm{lp}}\} \text{ and}$$
$$\#\top := \max\{\#\top^\delta_{q_1,q_2} : q_1, q_2 \in Q, \delta \in \Delta_{\mathrm{lp}}\}.$$

Definition 2.4.64. We define

$$\mathrm{BHLL}_k^{\mathscr{S}} : \mathbb{N} \to \mathbb{N} \text{ via}$$
$$\mathrm{BHLL}_k^{\mathscr{S}}(0) = 0 \text{ and}$$
$$\mathrm{BHLL}_k^{\mathscr{S}}(n+1) = l + \#\top \cdot \mathrm{BRL}_k^{\mathscr{S}}(n).$$

Furthermore, we define $\mathrm{BLLL}_k^{\mathscr{S}} : \mathbb{N} \to \mathbb{N}$ and $\mathrm{BLL}_k^{\mathscr{S}} : \mathbb{N} \to \mathbb{N}$ simultaneously via

$$\begin{aligned}
&\mathrm{BLLL}_k^{\mathscr{S}}(0) := 0,\\
&\mathrm{BLL}_k^{\mathscr{S}}(0) := 0,\\
&\mathrm{BLLL}_k^{\mathscr{S}}(n+1) := 2 + \mathrm{BLL}_k^{\mathscr{S}}(n) \text{ and}\\
&\mathrm{BLL}_k^{\mathscr{S}}(n+1) := \mathrm{BLLL}_k^{\mathscr{S}}(n+1) + 2 \cdot \mathrm{BHLL}_k^{\mathscr{S}}(n+1).
\end{aligned}$$

Remark 2.4.65. The following idea underlies the definition of $\mathrm{BHLL}_k^{\mathscr{S}}$. Let w be some word w of length $n-1$. Assume that we have already proved that the lengths of the shortest $\#\mathrm{Ret}^k(w)(q_1,q_2)$ many returns from $(q_1, w\Box : w)$ to $(q_2, [w\Box])$ are bound by $\mathrm{BRL}_k^{\mathscr{S}}(n-1)$.

Now, let s be a stack such that $w = \mathrm{top}_2(\mathrm{pop}_1(s))$. Due to the definition of l, the lexicographically smallest $\#\mathrm{HLoop}^k(s)(q,q')$ many high loops from (q,s) to (q',s) have simulations of length at most l.

sTl_s translates these simulations into $\#\mathrm{HLoop}^k(s)(q,q')$ many high loops by copying all transitions one by one but by replacing transitions on topmost symbol $\top$ by lexicographically small returns equivalent to those from $(q_1, w\Box : w)$ to $(q_2, w\Box)$. Since this replacement happens at at most $\#\top$ many positions, we obtain $\#\mathrm{HLoop}^k(s)(q,q')$ many returns from (q,s) to $(q', \mathrm{pop}_2(s))$ of length at most $l + \#\top \cdot \mathrm{BRL}_k^{\mathscr{S}}(n-1) = \mathrm{BHLL}_k^{\mathscr{S}}(n)$.

The other two functions are motivated as follows. A low loop of a word $w\sigma$ consists of its initial and final transition plus a loop of w. Hence, a short low loop consists of a short loop of w plus 2 transitions.

Due to Lemma 2.4.55, a loop of $w\sigma$ is either a high loop or consists of a high loop followed by a low loop followed by a high loop. Thus, short loops consists of at most three short loops, one a low the two others high ones.

In analogy to Remark 2.4.45, the previous remark already contains the first half of the proof of Proposition 2.4.48. The second half is proved completely analogous to the return case.

2.5 Automatic Structures

For over 50 years finite automata have been playing a crucial role in theoretical computer science and have found various applications in very different fields. In this chapter we recall the basic notions and techniques concerning finite tree-automata and tree-automatic structures. In general finite automata come in different flavours. On one hand automata can be used as acceptors for strings or for trees and on the other hand one can consider the variants for inputs of finite or infinite length. We mainly focus on finite tree-automata for finite binary trees because we will use these automata as one of the crucial tools in Section 3.1. Nevertheless, in Section 3.4 we will also use finite ω-tree-automata on infinite trees as tools for our proof. But only basic facts concerning ω-tree-automata are actually needed to understand that section.

For automata on strings, most of the facts we present here are folklore. Their analogues for tree-automata are mostly straightforward generalisations.

This section is organised as follows: we first recall the notions of a finite tree-automaton and a finite ω-tree-automaton, then we introduce tree-automatic structures as a form of internal representation for infinite structures. Finally, we recall the known decidability results for model checking on tree-automatic structures.

2.5.1 Finite Automata

In this section, we present the basic theory of tree-automata and tree-automatic structures. For a more detailed introduction, we refer the reader to [19]. We start by fixing our notation concerning tree-automata.

Definition 2.5.1. A finite tree-automaton is a tuple $\mathscr{A} = (Q, \Sigma, q_I, F, \Delta)$ where Q is a finite nonempty set of states, Σ is a finite alphabet, $q_I \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\Delta \subseteq Q \times Q \times \Sigma \times Q$ is the transition relation.

Remark 2.5.2. In the following, we simply write automaton for "finite tree-automaton".

We next define the concept of a run of an automaton on a tree. Before we state the definition, recall that for any tree t, t^+ denotes the minimal elements of $\{0,1\}^* \setminus \mathrm{dom}(t)$ and $t^\oplus = \mathrm{dom}(t) \cup t^+$ (cf. Section 2.2.2).

Definition 2.5.3. A run of $\mathscr{A}$ on a binary Σ-labelled tree t is a map $\rho : \mathrm{dom}(t)^\oplus \to Q$ such that

- $\rho(d) = q_I$ for all $d \in \mathrm{dom}(t^+)$, and
- $\big(\rho(d0), \rho(d1), t(d), \rho(d)\big) \in \Delta$ for all $d \in \mathrm{dom}(t)$.

ρ is called accepting if $\rho(\varepsilon) \in F$. We say t is accepted by $\mathscr{A}$ if there is an accepting run of $\mathscr{A}$ on t. With each automaton $\mathscr{A}$, we associate the language

$$L(\mathscr{A}) := \{t : t \text{ is accepted by } \mathscr{A}\}$$

accepted (or recognised) by $\mathscr{A}$. The class of languages accepted by automata is called the class of regular languages.

Remark 2.5.4. Recall that we can consider any string as a tree where each node has at most one successor. Using this idea a finite string-automaton is just the corresponding special case of a finite automaton.

One of the reasons for the success of the concept of automata in computer science is the robustness of this model with respect to determinisation. We call an automaton $\mathscr{A}$ bottom-up deterministic, if Δ is the graph of a function $Q \times Q \times \Sigma \to Q$.

Lemma 2.5.5 (see [19]). For each automaton $\mathscr{A}$ there is a bottom-up deterministic automaton $\mathscr{A}'$ that accepts exactly the same trees as $\mathscr{A}$.

This correspondence between deterministic and nondeterministic automata is one of the reasons why the class of regular languages has very strong closure properties.

Lemma 2.5.6 (see [19]). The regular languages are closed under conjunction, disjunction, complementation, and projection.

While the proof of closure under complementation is straightforward using deterministic automata (just use $Q \setminus F$ as set of accepting states), the closure under projection is easily shown using nondeterministic automata.

In the next section we will see how these closure properties can be used to turn automata into a useful tool for first-order model checking purposes using the concept of an automatic

structure. Beforehand, we recall some more facts about automata. First, we present the generalisation of the pumping lemma for regular string languages to the tree case. Instead of the length of a string, one uses the depth of a tree. One obtains the completely analogous result which states that if a regular language of trees contains a tree of large depth, then the language contains infinitely many trees and some of these have smaller depth than the tree considered initially.

Lemma 2.5.7 (see [19]). Let $\mathscr{A} = (Q, \Sigma, q_I, F, \Delta)$ be an automaton recognising the language L. For each tree $t \in L$ with $\mathsf{dp}(t) > |Q|$, there are nodes $d, d' \in \mathsf{dom}(t)$ with $d \leq d'$ such that the following holds. If we replace in t the subtree rooted at d by the subtree rooted at d', the tree t_0 resulting from this replacement satisfies $t_0 \in L$. Furthermore, let $t_1, t_2, t_3, \ldots$ be the infinite sequence of trees where $t_1 = t$ and t_{i+1} arises from t_i by replacing the subtree rooted at d' in t_i by the subtree rooted at d in t_i, then $t_i \in L$ for all $i \in \mathbb{N}$.

Proof. Take an accepting run ρ_1 of $\mathscr{A}$ on t. Since $\mathsf{dp}(t) > |Q|$, there are $d, d' \in \mathsf{dom}(t)$ such that $\rho_1(d) = \rho_1(d')$ and $d \leq d'$. Now, we have to show that the trees $t_0, t_1, t_2, \ldots$ are accepted by $\mathscr{A}$, i.e., we have to define accepting runs for these trees. For t_0 consider the run

$$\rho_0 : \mathsf{dom}(t_0)^{\oplus} \to Q \text{ where } \rho_0(e) := \begin{cases} \rho_1(e) & \text{if } d \not\leq e, \\ \rho_1(d'f) & \text{if } e = df. \end{cases}$$

It is easy to see that ρ_0 is a run of $\mathscr{A}$ on t_0. It is accepting because ρ was accepting and we did not alter the label of the root. For $i \geq 1$ we use the same trick the other way round, setting

$$\rho_{i+1}(e) := \begin{cases} \rho_i(e) & \text{if } d' \not\leq e, \\ \rho_i(df) & \text{if } e = d'f. \end{cases}$$

Again one easily sees that this defines an accepting run of $\mathscr{A}$ on t_{i+1}. □

As a direct corollary of the pumping lemma we obtain that finiteness of regular languages is decidable because finiteness of such a language is equivalent to not containing a tree of depth between $|Q|$ and $2|Q|$. The latter can be checked by exhaustive search.

Corollary 2.5.8. Given an automaton $\mathscr{A}$, it is decidable whether $L(\mathscr{A})$ is finite. If this is the case, we can compute $|L(\mathscr{A})|$.

We conclude this introduction to automata on trees by recalling a well known characterisation of regular classes of trees in terms of MSO-definability.

Lemma 2.5.9 ([59], [21]). For a set T of finite Σ-labelled trees, there is an automaton recognising T if and only if T is MSO definable.

Beside the successful applications of automata on finite strings or trees in many areas of computer science, the lifting of the underlying ideas to the case of infinite inputs had a mayor impact on the importance of automata theory for computer science. Rabin[55] played a prominent role in the development of this theory.

In order to give a meaningful definition of an automaton processing an infinite tree, we have to reverse the direction in which the automaton works. Up to now, we have considered bottom-up automata, i.e., automata which start to label a tree at the leaves and then process the tree up to the root. Of course, one can also imagine an automaton that starts labelling the root and then labels top-down all the nodes from the root to the leaves. For the determinisation result we presented, it is very important to think of a bottom-up automaton. Deterministic top-down automata are strictly weaker than nondeterministic ones: there is a regular language which is not the language recognised by any top-down deterministic automaton. Top-down automata become important as soon as we look at infinite trees. Since there are infinite trees without leaves, the bottom-up approach is not meaningful anymore. But the top-down approach generalises from finite to infinite trees. Considered as a device working top-down, an automaton is a device labelling the root with the initial state and then forks into two copies of this automaton – one for each successor of the root. Each of these copies now repeats the same procedure on the corresponding subtree but starting from a different state according to the transition relation. With this view, it is straightforward to generalise the notion of a run from finite trees to infinite trees. We only have to come up with a new concept of an accepting run. We now introduce finite ω-tree automata. Recall that we write $t^\perp$ for the lifting of a (possibly infinite) tree to the domain $\{0,1\}^*$ by padding with $\perp$.

Definition 2.5.10. A finite ω-tree automaton is a tuple $\mathscr{A} = (Q, \Sigma, Q_I, \Delta, \Omega)$, where Q and Σ, and Δ are as in the case of a finite tree-automaton, Q_I is a set (called the set of initial states), and Ω is a function $\Omega : Q \to \mathbb{N}$ (called priority function).

A function $\rho : \{0,1\}^* \to Q$ is a run of $\mathscr{A}$ on an infinite tree $t^\perp$ if $\rho(\varepsilon) \in Q_I$ and ρ respects Δ. We say ρ is a run on an arbitrary finite or infinite tree t if it is a run on $t^\perp$.

Given some run ρ of $\mathscr{A}$ on t, we call ρ accepting if $\liminf_{n\to\infty} \Omega(\rho(b_1 b_2 b_3 \dots b_n))$ is even for all infinite branches $b_1 b_2 b_3 \dots \in \{0,1\}^*$.

Remark 2.5.11. The acceptance condition that we present here is called parity condition. In the literature, several other acceptance conditions for automata on infinite trees are studied, e.g., Buchi-, Muller-, Rabin- or Street-conditions. The parity condition turned out to be the strongest of all these in the sense that all other conditions mentioned can be reformulated in terms of parity conditions, while the parity condition is weak enough in order to transfer most of the important results from the theory of finite trees to the infinite tree case.

In the following we use the term ω-automaton for "finite ω-tree-automaton".

Even though the determinisation result for finite automata does not carry over to ω-automata, the languages accepted by ω-automata have the same good closure properties as in the finite case. Rabin was the first who gave a construction for the complementation of a nondeterministic ω-automaton. In analogy to the finite case, we call the class of languages of (finite and infinite) trees accepted by ω-automata ω-regular languages.

Lemma 2.5.12 ([55]). The ω-regular languages are closed under conjunction, disjunction, complementation, and projection.

The proof of this lemma is through effective constructions of the corresponding ω-automata. Furthermore, the tight correspondence between automata and MSO carries over from the finite to the ω-case.

Theorem 2.5.13 ([55]). A subset $S \subseteq \mathsf{Tree}_\Sigma^{\leq\omega}$ is ω-regular if and only if it is MSO-definable.

We conclude this brief introduction of ω-automata by recalling the connection between regular and ω-regular sets of trees. We show that each regular set has an ω-regular representation via padding with some label $\perp$. Recall that for a Σ-labelled tree t, we write $t^\perp$ for the full binary tree which coincides with t on $\mathsf{dom}(t)$ and is labelled by $\perp$ at all other positions. In the following lemmas, we assume that $\perp \notin \Sigma$.

Lemma 2.5.14. Given an automaton $\mathscr{A} = (Q, \Sigma, q_I, F, \Delta)$, one can construct an ω-automaton $\mathscr{A}^\infty$ such that for all finite Σ-labelled trees t,

$$\mathscr{A} \text{ accepts } t \text{ iff } \mathscr{A}^\infty \text{ accepts } t^\perp.$$

Proof. The construction of $\mathscr{A}^\infty := (Q^\infty, \Sigma \cup \{\perp\}, Q_I^\infty, \Delta^\infty, \Omega)$ is as follows. We add a new state q_{acc} to the set of states by setting $Q^\infty := Q \cup \{q_{\text{acc}}\}$. Set $Q_I^\infty := F$ (since we change from the bottom-up view to the top-down view, the final states of the automaton become the initial state of the ω-automaton). Δ^∞ is a copy of Δ enriched by the following transitions: $\{(q_{\text{acc}}, q_{\text{acc}}, \perp, q_I), (q_{\text{acc}}, q_{\text{acc}}, \perp, q_{\text{acc}})\}$. The priority function $\Omega : Q_I^\infty \to \{1, 2\}$ is defined by

$$\Omega(q) = \begin{cases} 1 & \text{if } q \in Q, \\ 2 & \text{if } q = q_{\text{acc}}. \end{cases}$$

Using these definitions a tree t is accepted if and only if there is a finite initial part $D \subseteq \{0,1\}^*$ such that $\mathscr{A}$ accepts $t\restriction_D$, i.e., it labels $t\restriction_{D^+}$ only with the initial state q_I and all descendants of D^+ are nodes labelled by $\perp$. Thus, $\mathscr{A}$ accepts a tree t' if and only if it is of the form $t' = t^\perp$ for some finite tree t such that $\mathscr{A}$ accepts t. □

Lemma 2.5.15. Given an ω-automaton $\mathscr{A} = (Q, \Sigma, Q_I, \Delta, \Omega)$, one can construct an automaton $\mathscr{A}^{\text{fin}}$ such that for all finite Σ-labelled trees t,

$$\mathscr{A} \text{ accepts } t^\perp \text{ iff } \mathscr{A}^{\text{fin}} \text{ accepts } t.$$

Proof. We construct $\mathscr{A}^{\text{fin}} := (Q^{\text{fin}}, \Sigma, q_I^{\text{fin}}, F^{\text{fin}}, \Delta^{\text{fin}})$ as follows:

- $Q^{\text{fin}} := Q \cup \{q_{\text{init}}\}$ for a new state q_{init} not contained in Q,
- $F^{\text{fin}} := Q_I$,
- $q_I^{\text{fin}} := q_{\text{init}}$, and
- Δ^{fin} is constructed as follows. For each $q \in Q$ we consider the runs of the automaton $\mathscr{A}$ with initial state q, i.e., the automaton $\mathscr{A}_q := (Q, \Sigma, \{q\}, \Delta, \Omega)$, on the $\{\perp\}$-labelled full binary tree $\emptyset^\perp$. We call q good if there is an accepting run of $\mathscr{A}_q$ on $\emptyset^\perp$. Now, for each transition (q_1, q_2, σ, q_3) we add a new transition $(q_{\text{init}}, q_2, \sigma, q_3)$ to Δ^{fin} if q_1 is good. Analogously, we add a transition $(q_1, q_{\text{init}}, \sigma, q_3)$ to Δ^{fin} if q_2 is good. Finally, we add $(q_{\text{init}}, q_{\text{init}}, \sigma, q_3)$ to Δ^{fin} if both q_1 and q_2 are good. Furthermore, Δ^{fin} contains a copy of each transition in Δ, i.e., $\Delta \subseteq \Delta^{\text{fin}}$.

Now, $\mathscr{A}^{\text{fin}}$ copies the behaviour of $\mathscr{A}$ but at any position where one of the successor nodes is labelled by a good state, it can nondeterministically guess that the tree it processes is not defined on this successor. If this guess is right, then $\mathscr{A}$ processes at this successor a tree which is completely labelled by $\perp$. Since the state at this successor is good, the partial run up to this position can be extended in such a way that each path starting at this successor is accepting.

Now, if $\mathscr{A}^{\text{fin}}$ labels some node by its initial state, there is no transition that is applicable at this node. Thus, any run of $\mathscr{A}^{\text{fin}}$ on a tree t labels only those positions by q_{init} that are in $\mathsf{dom}(t)^+$.

By definition, a run of $\mathscr{A}^{\text{fin}}$ on some tree t labels all elements of $\mathsf{dom}(t)^+$ by q_{init} if and only if there is a run of $\mathscr{A}$ on $t^\perp$ that labels all elements of $\mathsf{dom}(t)^+$ by good states. But this is equivalent to the fact that $\mathscr{A}$ accepts $t^\perp$ by the definition of good states.

We conclude that $\mathscr{A}^{\text{fin}}$ satisfies the claim of this lemma. □

2.5.2 Automatic Structures

As already mentioned the algorithmic tractability of problems on an infinite structure depends on a good finite representation. In this section we recall how automata can be used for this purpose. The general underlying idea is the following.

Given some structure $\mathfrak{A}$, one defines a tuple of machines from some fixed model of computation such that these machines can be used to evaluate atomic formulas on $\mathfrak{A}$.

A presentation of some structure $\mathfrak{A} = (A, E_1, E_2, \ldots, E_n)$ using a certain model of computation $\mathscr{M}$ is a tuple of machines $M, M_1, M_2, \ldots, M_n$ from $\mathscr{M}$ and some map f such that the following holds.

- M accepts a set L of strings or trees.
- f is a bijective map from L to A.
- M_i accepts a tuple of elements from L if and only if the image of this tuple under f is in E_i.

The first model of computation that was considered for this approach is that of Turing machines (cf. Appendix A). If one uses Turing machines for representing a structure in this way, one obtains the so-called class of recursive structures (cf. [29]). But for algorithmic issues, Turing machines turned out to be far too strong, resulting in the undecidability of model checking on recursive structures for most logics. In general, it is only possible to evaluate quantifier-free formula on recursive structures.

Automata can be used much more fruitfully as underlying model of computation. This is due to their good computational behaviour. The resulting structures are called automatic structures. Hodgson [31, 32] first proposed this idea. But it took more than 10 years until the systematic investigation of the general notion of automatic structures started. Khoussainov and Nerode [37] reintroduced the notion of string-automatic structures. They obtained the first important results. For instance, they proved that these structures have decidable FO model checking due to the good closure properties of regular languages. Another boost to the study of automatic structures came from the work of Blumensath [7] who developed the theory further and lifted the idea from the finite string case to the cases of finite or infinite

strings or trees. Since then, the field of automatic structures has been an active area of research and many new results have been collected over the years by Blumensath, Grädel, Khoussainov, Kuske, Lohrey, Rubin, et al.(e.g., [37, 10, 39, 44, 11, 40, 38, 57, 36, 45, 47, 46]). In the following, we recall the definitions and important results with a focus on tree-automatic structures (which we simply call automatic-structures in the following). String-automatic structures are obtained by restriction of the accepted languages to languages of strings. We start by introducing the convolution of trees. This is a tool for representing an n-tuple of Σ-trees as a single tree over the alphabet $(\Sigma \cup \{\Box\})^n$ where $\Box$ is a padding symbol $\Box \notin \Sigma$.

Definition 2.5.16. The convolution of two Σ-labelled trees t and s is given by a function

$$t \otimes s : \mathrm{dom}(t) \cup \mathrm{dom}(s) \to (\Sigma \cup \{\Box\})^2$$

where $\Box$ is some new padding symbol, and

$$(t \otimes s)(d) := \begin{cases} (t(d), s(d)) & \text{if } d \in \mathrm{dom}(t) \cap \mathrm{dom}(s), \\ (t(d), \Box) & \text{if } d \in \mathrm{dom}(t) \setminus \mathrm{dom}(s), \\ (\Box, s(d)) & \text{if } d \in \mathrm{dom}(s) \setminus \mathrm{dom}(t). \end{cases}$$

We also use the notation $\bigotimes(t_1, t_2, \ldots, t_n)$ for $t_1 \otimes t_2 \otimes \cdots \otimes t_n$.

Using convolutions of trees we can use a single automaton for defining n-ary relations on a set of trees. Thus, we can then use automata to represent a set and a tuple of n-ary relations on this set. If we can represent the domain of some structure and all its relations by automata, we call the structure automatic.

Definition 2.5.17. We say a relation $R \subseteq \mathrm{Tree}_\Sigma^n$ is automatic if there is an automaton $\mathscr{A}$ such that $L(\mathscr{A}) = \{\bigotimes(t_1, t_2, \ldots, t_n) \in \mathrm{Tree}_\Sigma^n : (t_1, t_2, \ldots, t_n) \in R\}$.

A structure $\mathfrak{B} = (B, E_1, E_2, \ldots, E_n)$ with relations E_i is automatic if there are automata $\mathscr{A}_B, \mathscr{A}_{E_1}, \mathscr{A}_{E_2}, \ldots, \mathscr{A}_{E_n}$ such that for the language $L(\mathscr{A}_B)$ accepted by $\mathscr{A}_B$ the following holds:

1. There is a bijection $f : L(\mathscr{A}_B) \to B$.
2. For $c_1, c_2, \ldots, c_n \in L(\mathscr{A}_B)$, the automaton $\mathscr{A}_{E_i}$ accepts $\bigotimes(c_1, c_2, \ldots, c_n)$ if and only if $(f(c_1), f(c_2), \ldots, f(c_n)) \in E_i$.

In other words, f is a bijection between $L(\mathscr{A}_B)$ and B and the relations E_i are automatic via the automata $\mathscr{A}_{E_i}$. We call f a tree presentation of $\mathfrak{B}$.

Automatic structures form a nice class because automata theoretic techniques may be used to decide first-order formulas on these structures:

Theorem 2.5.18 ([7], [57]). If $\mathfrak{B}$ is automatic, then its $\mathrm{FO}(\exists^{\mathrm{mod}})$-theory is decidable.

Proof. Given some FO formula $\varphi(x_1, \ldots, x_n)$, we can construct effectively an automaton $\mathscr{A}_\varphi$ such that $\mathscr{A}_\varphi$ accepts $t_1 \otimes \cdots \otimes t_n$ if and only if $\mathfrak{B}, f(t_1), \ldots, f(t_n) \models \varphi$ for f the bijection from the previous definition. For atomic formulas, this is clear from the definition of an automatic structure because the automata for the relations are already given in the definition. Conjunction and negation transform into the classical automata constructions of

product and complementation. Finally, existential quantification corresponds to the closure of regular languages under projection.

The decidability of the modulo counting quantifier was first proved for the string-automatic case in [40]. Our presentation follows the ideas of Rubin [57]. He provided a proof for the string-automatic case that allows a straightforward adaption to the case of trees.

For simplicity in the presentation, we assume that $\mathfrak{B}$ is an automatic structure whose presentation is the identity id. Let

$$\varphi(x, y_1, y_2, \ldots, y_n) \in \mathrm{FO}(\exists^{\mathrm{mod}})$$

be some formula which is represented on $\mathfrak{B}$ by the automaton $\mathscr{A} = (Q, \Sigma, q_I, F, \delta)$, i.e.,

$$\mathfrak{B}, t, t_1, t_2, \ldots, t_n \models \varphi(x, y_1, y_2, \ldots, y_n)$$

if and only if $\mathscr{A}$ accepts $\bigotimes(t, t_1, t_2, \ldots, t_n)$.

Given a tuple $t_1 \otimes \cdots \otimes t_n$ representing the assignment of the free variables in a formula $\exists^{(k,m)} x(\varphi(x, y_1, \ldots, y_n))$ for which we want to evaluate the formula, we have to construct an automaton that counts modulo j the number of trees t such that $\mathscr{A}$ accepts $t \otimes \bar{t} := t \otimes t_1 \otimes \cdots \otimes t_n$. Without loss of generality, we assume that $\mathscr{A}$ is a bottom-up deterministic automaton. In this case the number of trees t such that $t \otimes \bar{t}$ is accepted by $\mathscr{A}$ coincides with the number of accepting runs on trees of the form $t \otimes \bar{t}$. We now construct an automaton $\hat{\mathscr{A}}$ that does this counting. The states $\hat{Q}$ of $\hat{\mathscr{A}}$ are functions $Q \to \{0, 1, \ldots, m-1, \infty\}$. $\hat{\mathscr{A}}$ will label a node d of $\bar{t}$ with a function f such that there are (modulo m) $f(q)$ different trees t such that the unique run of $\mathscr{A}$ on $t \otimes (\bar{t})_d$ labels the root with state q. By this we mean that $f(q) = \infty$ iff there are infinitely many such trees t and otherwise $f(q)$ determines the number of such trees modulo m. If we know how to label the successors of some node according to this rule, then some automaton can update this information. The details of the construction are as follows.

We set $\hat{Q} := \{0, 1, \ldots, m-1, \infty\}^Q$, the initial state is $\hat{q}_I := f$ where – modulo m – $f(q)$ is

$$|\{t : \rho(\varepsilon) = q \text{ for } \rho \text{ the run of } \mathscr{A} \text{ on } t \otimes \emptyset^n\}|.$$

Note that q_I is computable due to Corollary 2.5.8. The set of final states $\hat{F}$ consists of those function $f : Q \to \{0, 1, \ldots, m-1, \infty\}$ such that $\sum_{q \in F} f(q) = k \mod m$. The transition relation $\hat{\Delta}$ consists of all tuples (f_0, f_1, σ, f) where f_0, f_1, f are functions $Q \to \{0, 1, \ldots, m-1, \infty\}$ such that

$$f(q) = \left(\sum_{(q_0, q_1, \sigma, q) \in \Delta} f_0(q_0) \cdot f_1(q_1) \right) \mod k$$

holds for all $q \in Q$. Note that for fixed f_0, f_1, and σ the function f is uniquely determined. Thus, the resulting automaton is a deterministic bottom-up automaton.

By an easy induction, one sees that for all $d \in \bar{t}$ the run of $\hat{\mathscr{A}}$ on $(\bar{t})_d$ labels the root with some function $f : Q \to \{0, 1, \ldots, m-1, \infty\}$ such that there are $f(q)$ many different trees t (modulo m) such that the run ρ_t of $\mathscr{A}$ on $t \otimes (\bar{t})_d$ satisfies $\rho_t(\varepsilon) = q$.

From this fact, we directly obtain the desired result, namely, that $\hat{\mathscr{A}}$ accepts $\bar{t}$ if and only if there are k modulo m many trees t such that $\mathscr{A}$ accepts $t \otimes \bar{t}$. □

Remark 2.5.19. The complexity of the FO model checking algorithm for automatic structures is nonelementary. This is due to the following facts.

- Applying a projection to some deterministic automaton yields a nondeterministic one.
- Complementation of an automaton can only be done efficiently if the automaton is deterministic.
- Determinisation of a nondeterministic automaton yields an exponential blow-up.

Thus, the size of the automaton obtained by the construction in the proof is an exponential tower in the number of alternations of existential quantification and negation in the formula (or equivalently the number of alternations of existential and universal quantifications), i.e., if there are n alternations between existential and universal quantification in φ, then the corresponding automaton $\mathscr{A}_\varphi$ may have $\exp_n(c)$ many states (for c some constant)[7].

On the other hand, the algorithm cannot be improved essentially: the extension $(\mathbb{N}, +, |_p)$[8] of Presburger Arithmetic is a string-automatic structure [10] which has nonelementary model checking complexity [26]. This implies that there is a nonelementary lower bound for the FO model checking complexity on automatic structures.

The theory of automatic structures can be naturally extended to the theory of structures that are represented by automata for infinite trees. The structures obtained in this way are called ω-automatic structures. We conclude this section by precisely defining ω-automatic structures.

Definition 2.5.20. We say a relation $R \subseteq (\mathrm{Tree}_\Sigma^{\leq\omega})^n$ is ω-automatic if there is an ω-automaton $\mathscr{A}$ such that

$$L(\mathscr{A}) = \left\{ \bigotimes(t_1, t_2, \ldots, t_n) \in (\mathrm{Tree}_\Sigma^{\leq\omega})^n : (t_1, t_2, \ldots, t_n) \in R \right\}.$$

A structure $\mathfrak{A} = (A, E_1, E_2, \ldots, E_n)$ with relations E_i is ω-automatic if there are ω-automata $\mathscr{A}_A, \mathscr{A}_{E_1}, \mathscr{A}_{E_2}, \ldots, \mathscr{A}_{E_n}$ such that for $L(\mathscr{A}_A)$, the language accepted by $\mathscr{A}_A$, the following holds:

1. There is a bijection $f : L(\mathscr{A}_A) \to A$.
2. For $t_1, t_2, \ldots, t_n \in L(\mathscr{A}_A)$, the automaton $\mathscr{A}_{E_i}$ accepts $\bigotimes(t_1, t_2, \ldots, t_n)$ if and only if $(f(t_1), f(t_2), \ldots, f(t_n)) \in E_i$.

We call f an ω-presentation of $\mathfrak{A}$.

The similarity of Lemma 2.5.6 and Lemma 2.5.12 yields the straightforward extension of Theorem 2.5.18 to the ω-automatic case.

Theorem 2.5.21 ([7, 10, 5]). The $\mathrm{FO}(\exists^\infty)$-theory of every ω-automatic structure is decidable.

[7] We denote by $\exp_i$ the following function. $\exp_0(m) := c$ and $\exp_{n+1}(c) := 2^{\exp_n(m)}$, i.e., $\exp_n(m)$ is an exponential tower of height n with topmost exponent m.

[8] $x|_p y$ if x is a power of p dividing y.

3 Main Results

In this chapter, we present four results concerning model checking on certain graph structures. The first and the last involve automaticity[1] while the other two are based on modularity arguments for Ehrenfeucht-Fraïssé games.

Our first result concerns FO model checking on collapsible pushdown graphs of level 2. The expansion of every level 2 collapsible pushdown graph by regular reachability and $L\mu$-definable predicates is automatic. From the general decidability result for FO on automatic structures, we obtain the following theorem.

Theorem 3.0.1. Let $\mathscr{S} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a collapsible pushdown system of level 2. Let

$$\mathfrak{G} := (\mathsf{CPG}(\mathscr{S}), \mathrm{REACH}_{L_1}, \mathrm{REACH}_{L_2}, \ldots, \mathrm{REACH}_{L_n}, P_1, \ldots, P_m)$$

be an expansion of the collapsible pushdown graph $\mathsf{CPG}(\mathscr{S})$ where $L_1, L_2, \ldots, L_n$ are arbitrary regular languages over Γ and $P_1, \ldots, P_m$ are arbitrary $L\mu$-definable predicates. Then the FO-theory of $\mathfrak{G}$ is decidable.

Using our fourth main theorem, even the $\mathsf{FO}(\exists^\infty, \exists^{\mathrm{mod}}, (\mathrm{Ram}^n)_{n\in\mathbb{N}})$-theory of $\mathfrak{G}$ is decidable. A preliminary version of this theorem was published in [35] and we present the proof of this theorem in Section 3.1.

Next, we turn to modularity arguments for Ehrenfeucht-Fraïssé games on nested pushdown trees. The analysis of restricted strategies in these games lead to model checking algorithms on the class of nested pushdown trees (cf. Section 2.1.1). We obtain the following two results.

Theorem 3.0.2. FO(REACH) model checking on nested pushdown trees is decidable. Furthermore, there is an FO model checking algorithm on nested pushdown trees with the following complexities: Its structure complexity is in EXPSPACE, while its expression complexity and its combined complexity are in 2-EXPSPACE.

Theorem 3.0.3. FO model checking on level 2 nested pushdown trees is decidable.

The concept of a level 2 nested pushdown tree is a combination of the concepts of higher-order pushdown systems and nested pushdown trees. One takes a level 2 pushdown system (without collapse) and enriches the unfolding of its graph by jump-edges connecting corresponding clone and pop operations (of level 2). We formally introduce the hierarchy of higher-order nested pushdown trees in Section 3.3.

The proof of the first theorem, which was published in [34], is contained in Section 3.2. The second theorem is proved in Section 3.3.

Finally, motivated by the automaticity of collapsible pushdown graphs of level 2. We study the model checking problem on the class of automatic structure. We extend the automata-based approach for $\mathsf{FO}(\exists^\infty, \exists^{\mathrm{mod}})$ model checking on automatic structures to $\mathsf{FO}(\exists^\infty, \exists^{\mathrm{mod}}, (\mathrm{Ram}^n)_{n\in\mathbb{N}})$ model checking. In Section 3.4 we prove the following theorem which was developed by Dietrich Kuske and the author.

Theorem 3.0.4. The $\mathsf{FO}(\exists^\infty, \exists^{\mathrm{mod}}, (\mathrm{Ram}^n)_{n\in\mathbb{N}})$-theory of automatic structures is decidable.

[1] We stress that the term "automaton" stands for "finite tree-automaton" and "automatic" stands for "tree-automatic".

3.1 Level 2 Collapsible Pushdown Graphs are Tree-Automatic

In this section, we focus on collapsible pushdown graphs of level 2. Thus, whenever we talk about collapsible pushdown systems or graphs, we mean those of level 2.

The main result of this section is the following theorem.

Theorem 3.1.1. Given a collapsible pushdown graph $\mathsf{CPG}(\mathscr{S}) = (C(\mathscr{S}), (\vdash^\gamma)_{\gamma\in\Gamma})$, regular languages $L_1, L_2, \ldots, L_n \subseteq \Gamma^*$, and $L\mu$-definable predicates $P_1, P_2, \ldots, P_m \subseteq C(\mathscr{S})$, its expansion $(\mathsf{CPG}(\mathscr{S}), \mathsf{REACH}_{L_1}, \mathsf{REACH}_{L_2}, \ldots, \mathsf{REACH}_{L_n}, P_1, P_2, \ldots, P_m)$ is automatic[2].

A direct consequence of this result is the automaticity of the second level of the Caucal hierarchy.

Corollary 3.1.2. The second level of the Caucal hierarchy is automatic.

Proof. The second level of the Caucal hierarchy is obtained by ε-contraction[3] from the class of higher-order pushdown graphs of level 2 (cf. [16]). □

Using Theorem 3.0.4 we obtain the decidability of the first-order theory of collapsible pushdown graphs of level 2:

Corollary 3.1.3. Let $\mathscr{S}$ be a collapsible pushdown system of level 2. Let $\mathfrak{G}$ be the expansion of $\mathsf{CPG}(\mathscr{S})$ by $L\mu$-definable predicates and by regular reachability predicates. Under these conditions, the $\mathsf{FO}(\mathrm{Reg}, \exists^{\mathrm{mod}}, (\mathsf{Ram}^n)_{n\in\mathbb{N}})$-theory of $\mathfrak{G}$ is decidable.

Remark 3.1.4. Note that this corollary is just a reformulation of Theorem 3.0.1.

The main part of this section consists of a proof of theorem 3.1.1. Furthermore, we discuss the limitations of our approach and the limitations of first-order model checking on collapsible pushdown graphs in general.

The section is organised as follows. In Section 3.1.1, we present a function Enc which translates configurations of collapsible pushdown systems into trees. This function Enc yields an automatic representation for every collapsible pushdown graph. We show in Section 3.1.2 that the reachable configurations of a collapsible pushdown system are turned into a regular set of trees by Enc. The proof of this statement takes the results on loops from section 2.4 as a main ingredient. Recall that the loops of a given stack can be calculated by a string-automaton reading the topmost word of the stack. This result carries over to an automaton reading the encoding of a given stack. Since runs from the initial configuration to some configuration c mainly consist of loops, this kind of regularity of loops can be used to show the regularity of the set of reachable configurations. In Section 3.1.3, we prove that the stack operations are regular via Enc. Hence, for each transition relation $\vdash^\gamma$ there is an automaton recognising those encodings of pairs of configurations that are related by $\vdash^\gamma$. Then we show that regular reachability predicates over Γ^* are regular sets via Enc. This is done in Section 3.1.4 as follows. First, we prove that the image of the "ordinary" reachability predicate REACH is a regular relation via Enc. Then we show that collapsible pushdown graphs are closed under products with string-automata. Finally, we reduce the predicate

[2] Recall that "automatic" is an abbreviation for "tree-automatic".

[3] An ε-contraction of a higher-order pushdown graph $G = (V, E_1, E_2, \ldots, E_n)$ is a graph $(V, E'_1, E'_2, \ldots, E'_m)$ for $m \leq n$ where $E'_i := \mathsf{REACH}_{L_i}$ for $L_i = L((E_{m+1} + E_{m+2} + \cdots + E_n)^* E_i)$.

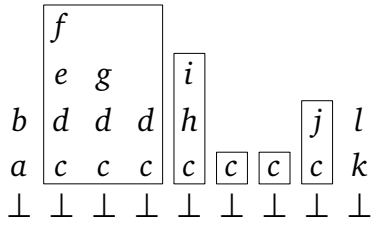


Figure 3.1.: A stack with blocks forming a c-blockline.

REACH_L to the predicate REACH on the product of the collapsible pushdown system and the string-automaton corresponding to L.

Afterwards, we relate our result to other known results. In Section 3.1.5, we first investigate combinations of the known $L\mu$ model checking algorithm with our FO model checking algorithm. Then, in Section 3.1.6, we provide a lower bound for FO model checking on level 2 collapsible pushdown graphs. Recall that the first-order model checking on automatic structures has nonelementary complexity. We show that the complexity of the first-order model checking on collapsible pushdown graphs is also nonelementary. Thus, our model checking algorithm cannot be improved essentially. In the final part we discuss first-order model checking on higher-order collapsible pushdown graphs. Recently, Broadbent [12] showed that first-order model checking is undecidable on level 3 collapsible pushdown graphs. Thus, there is no hope to extend our technique to higher levels of the collapsible pushdown hierarchy.

3.1.1 Encoding of Level 2 Stacks in Trees

In this section we present an encoding of level 2 stacks in trees. The idea is to divide a stack into blocks and to encode different blocks in different subtrees. The crucial observation is that every stack is a list of words that share the same first letter. A block is a maximal list of words occurring in the stack which share the same two first letters. If we remove the first letter of every word of such a block, the resulting 2-word decomposes again as a list of blocks. Thus, we can inductively carry on to decompose parts of a stack into blocks and encode every block in a different subtree. The roots of these subtrees are labelled with the first letter of the block. This results in a tree where every initial left-closed path in the tree represents one word of the stack. A path of a tree is left-closed if its last element has no left successor.

As we already mentioned, the encoding works by dividing stacks into blocks. The following notation is useful for the formal definition of blocks. Let $w \in \Sigma^*$ be some word and $s = w_1 : w_2 : \cdots : w_n \in \Sigma^{*2}$ some stack. We write $s' := w \setminus s$ for $s' = ww_1 : ww_2 : \cdots : ww_n$. Note that $[w]$ is a prefix of s', i.e., in the notation from Definition 2.3.32, $[w] \trianglelefteq w \setminus s$. We say that s' is s prefixed by w.

Definition 3.1.5. Let $\sigma \in \Sigma$ and $b \in \Sigma^{*2}$. We call b a σ-block if $b = [\sigma]$ or $b = \sigma\tau \setminus s'$ for some $\tau \in \Sigma$ and some $s' \in \Sigma^{*2}$. If $b_1, b_2, \ldots, b_n$ are σ-blocks, then we call $b_1 : b_2 : \cdots : b_n$ a σ-blockline. See Figure 3.1 for an example of a blockline with its blocks.

Note that every stack in $\mathsf{Stacks}_2(\Sigma)$ forms a $\bot$-blockline. Furthermore, every blockline l decomposes uniquely as $l = b_1 : b_2 : \cdots : b_n$ of maximal blocks b_i in l. We will call these maximal blocks the blocks of l.

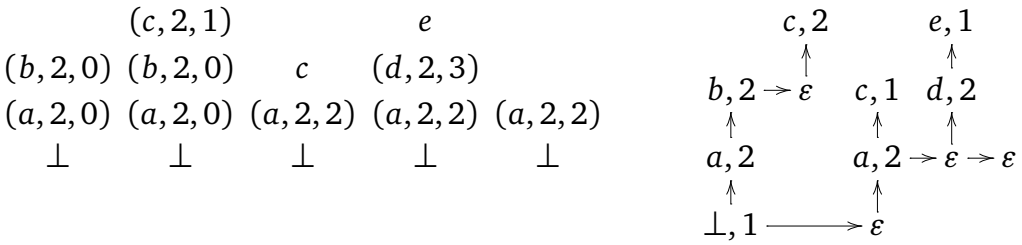


Figure 3.2.: A stack s and its Encoding $\mathsf{Enc}(s)$: right arrows lead to 1-successors (right successors), upward arrows lead to 0-successors (left successors).

Another crucial observation is that a σ-block $b \in \Sigma^{*2} \setminus \Sigma$ decomposes as $b = \sigma \setminus l$ for some blockline l and we call l the blockline induced by b. For a block of the form $[b]$ with $b \in \Sigma$, we define the blockline induced by $[b]$ to be $\emptyset$.

Recall that the symbols of a collapsible pushdown stack (of level 2) come from the set $\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ where Σ is the stack alphabet.

We are now going to define our encoding of stacks in trees. For $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$, we encode a τ-blockline l in a tree as follows. The root of the tree is labelled by $(\mathsf{Sym}(\tau), \mathsf{CLvl}(\tau))$. The blockline induced by the first block of l is encoded in the left subtree and the rest of l is encoded in the right subtree. This means that we only encode explicitly the symbol and the collapse level of each element of the stack, but not the collapse link. We will later see how to decode the collapse links from the encoding of a stack. When we encode a part of a blockline in the right subtree, we do not repeat the label $(\mathsf{Sym}(\tau), \mathsf{CLvl}(\tau))$, but replace it by the empty word ε.

Definition 3.1.6. Let $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Furthermore, let

$$s = w_1 : w_2 : \cdots : w_n \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}$$

be some τ-blockline. Let w'_i be words such that $s = \tau \setminus [w'_1 : w'_2 : \cdots : w'_n]$ and set $s' := w'_1 : w'_2 : \cdots : w'_n$. As an abbreviation we write ${}_i s_k := w_i : w_{i+1} : \cdots : w_k$. Furthermore, let $w_1 : w_2 : \cdots : w_j$ be a maximal block of s. Note that $j > 1$ implies that there is some $\tau' \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ and there are words $w''_{j'}$ for each $j' \leq j$ such that $w_{j'} = \tau\tau' w''_{j'}$.

Now, for arbitrary $\sigma \in (\Sigma \times \{1,2\}) \cup \{\varepsilon\}$, we define recursively the $(\Sigma \times \{1,2\}) \cup \{\varepsilon\}$-labelled tree $\mathsf{Enc}(s, \sigma)$ via

$$\mathsf{Enc}(s,\sigma) := \begin{cases} \sigma & \text{if } |w_1| = 1, n = 1 \\ \sigma \to \mathsf{Enc}({}_2 s_n, \varepsilon) & \text{if } |w_1| = 1, n > 1 \\ \mathsf{Enc}({}_1 s'_n, (\mathsf{Sym}(\tau'), \mathsf{CLvl}(\tau'))) \leftarrow \sigma & \text{if } |w_1| > 1, j = n \\ \mathsf{Enc}({}_1 s'_j, (\mathsf{Sym}(\tau'), \mathsf{CLvl}(\tau'))) \leftarrow \sigma \to \mathsf{Enc}({}_{j+1} s_n, \varepsilon) & \text{otherwise} \end{cases}$$

For every $s \in \mathsf{Stacks}_2(\Sigma)$, $\mathsf{Enc}(s) := \mathsf{Enc}(s, (\bot, 1))$ is called the encoding of the stack s.

Figure 3.2 shows a configuration and its encoding.

Remark 3.1.7. Fix some stack s. For $\sigma \in \Sigma$ and $k \in \mathbb{N}$, every $(\sigma, 2, k)$-block of s is encoded in a subtree whose root d is labelled $(\sigma, 2)$. We can restore k from the position of d in the tree $\mathsf{Enc}(s)$ as follows.

$$k = |\{d' \in \mathsf{dom}\mathsf{Enc}(s) \cap \{0,1\}^* 1 : d' \leq_{\mathrm{lex}} d\}|,$$

where $\leq_{\text{lex}}$ is the lexicographic order. This is due to the fact that every right-successor corresponds to the separation of some block from some other.

This correspondence can be seen as a bijection. Let $s = w_1 : w_2 : \cdots : w_n$ be some stack. We define the set $R := \mathsf{dom}(\mathsf{Enc}(s)) \cap (\{\varepsilon\} \cup \{0,1\}^*1)$. Then there is a bijection $f : \{1,2,3,\ldots,n\} \to R$ such that i is mapped to the i-th element of R in lexicographic order. Each $1 \leq i \leq n$ represents the i-th word of s. f maps the first word of s to the root of $\mathsf{Enc}(s)$ and every other word in s to the element of $\mathsf{Enc}(s)$ that separates this word from its left neighbour in s.

If we interpret ε as empty word, the word from the root to $f(i)$ in $\mathsf{Enc}(s)$ is the greatest common prefix of w_{i-1} and w_i. More precisely, the word read along this path is the projection onto the letters and collapse levels of $w_{i-1} \sqcap w_i$.

Furthermore, set $f'(i) := d0^m \in \mathsf{Enc}(s)$ for $d := f(i)$ such that m is maximal with this property, i.e., $f'(i)$ is the leftmost descendent of $f(i)$. Then the path from $f(i)$ to $f'(i)$ is the suffix w_i' such that $w_i = (w_{i-1} \sqcap w_i) \circ w_i'$ (here we set $w_0 := \varepsilon$). More precisely, the word read along this path is the projection onto the symbols and collapse levels of w_i'.

Having defined the encoding of a stack, we want to encode whole configurations, i.e., a stack together with a state. To this end, we just add the state as new root of the tree and attach the encoding of the stack as left subtree, i.e., for some configuration (q,s) we set

$$\mathsf{Enc}(q,s) := \mathsf{Enc}(s) \leftarrow q.$$

The image of this encoding function contains only trees of a very specific type. We call this class $\mathbb{T}_{\text{Enc}}$. In the next definition we state the characterising properties of $\mathbb{T}_{\text{Enc}}$. This class is MSO-definable whence automata-recognisable (cf. Lemma 2.5.9).

Definition 3.1.8. Let $\mathbb{T}_{\text{Enc}}$ be the class of trees T that satisfy the following conditions.

1. The root of T is labelled by some element of Q ($T(\varepsilon) \in Q$).
2. Every element of the form $\{0,1\}^*0$ is labelled by some $(\sigma, l) \in \Sigma \times \{1,2\}$, especially $T(0) = (\bot, 1)$.
3. Every element of the form $\{0,1\}^*1$ is labelled by ε.
4. $1 \notin \mathsf{dom}(T)$, $0 \in \mathsf{dom}(T)$.
5. For all $t \in T$ we have that $T(t0) = (\sigma, 1)$ implies $T(t10) \neq (\sigma, 1)$.

Remark 3.1.9. Note that all trees in the image of Enc satisfy condition 5 due to the following. $T(t0) = T(t10) = (\sigma, 1)$ would imply that the subtree rooted at t encodes a blockline l such that the first block b_1 of l induces a σ-blockline and the second block b_2 induces also a σ-blockline. This contradicts the maximality of the blocks used in the encoding because all words of b_1 and b_2 have σ as second letter whence $b_1 : b_2$ forms a larger block. Note that for letters with links of level 2 the analogous restriction does not hold. In Figure 3.2 one sees the encoding of a stack s where $\mathsf{Enc}(s)(0) = \mathsf{Enc}(s)(10) = (a, 2)$. Here, the label $(a,2)$ represents two different letters. $\mathsf{Enc}(s)(0)$ encodes the element $(a,2,0)$, while $\mathsf{Enc}(s)(10)$ encodes the element $(a,2,2)$, i.e., the first element encodes a letter a with undefined link and the second encodes the letter a with a link to the substack of width 2.

Having defined the encoding function Enc, we next show that it induces a bijection between the configurations of CPG and $\mathbb{T}_{\mathrm{Enc}}$. The rest of this section is a formal proof of the following lemma.

Lemma 3.1.10. $\mathsf{Enc} : Q \times \mathsf{Stacks}_2(\Sigma) \to \mathbb{T}_{\mathrm{Enc}}$ is a bijection. We denote its inverse by Dec.

The formal proof of this lemma is rather technical. The reader who is not interested in the technical details of this proof may continue with reading Section 3.1.2 directly.

We start our proof of the lemma by explicitly constructing the inverse of Enc. This inverse is called Dec. Since Enc removes the collapse links of the elements in a stack, we have to restore these now. For restoring the collapse links, we use the following auxiliary function. For $g \in N$ and $\tau \in \{\varepsilon\} \cup (\Sigma \times \{1,2\})$, we set

$$f_g(\tau) := \begin{cases} \sigma & \text{if } \tau = (\sigma, 1), \\ (\sigma, 2, g) & \text{if } \tau = (\sigma, 2), \\ \varepsilon & \text{if } \tau = \varepsilon. \end{cases}$$

Later, g will be the width of the stack decoded so far.

Definition 3.1.11. Let $\Gamma := (\Sigma \times \{1,2\}) \cup \{\varepsilon\}$. We define the following function $\mathsf{Dec} : \mathsf{Tree}_\Gamma \times \mathbb{N} \to (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{*2}$ by recursion. Let

$$\mathsf{Dec}(T, g) = \begin{cases} f_g(T(\varepsilon)) & \text{if } \mathrm{dom}(T) = \{\varepsilon\}, \\ f_g(T(\varepsilon)) \setminus \mathsf{Dec}((T)_0, g) & \text{if } 1 \notin \mathrm{dom}(T), \\ f_g(T(\varepsilon)) \setminus (\varepsilon : \mathsf{Dec}((T)_1, g+1)) & \text{if } 0 \notin \mathrm{dom}(T), \\ f_g(T(\varepsilon)) \setminus (\mathsf{Dec}((T)_0, g) : \mathsf{Dec}((T)_1, g + G((T)_0))) & \text{otherwise,} \end{cases}$$

where $G((T)_0) := |\mathsf{Dec}((T)_0, 0)|$ is the width of the stack encoded in $(T)_0$. For a tree $T \in \mathbb{T}_{\mathrm{Enc}}$, the decoding of T is

$$\mathsf{Dec}(T) := (T(\varepsilon), \mathsf{Dec}((T)_0, 0)) \in Q \times (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}.$$

Remark 3.1.12. Obviously, for each $T \in \mathbb{T}_{\mathrm{Enc}}$, $\mathsf{Dec}(T) \in Q \times (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}$. In fact, the image of Dec only consists of configurations, i.e., $\mathsf{Dec}(T) = (q, s)$ such that s is a level 2 stack. The verification of this claim relies on two important observations.

Firstly, $T(0) = (\bot, 1)$ due to condition 2 of Definition 3.1.8. Thus, all words in s start with letter $\bot$.

Now, s is a stack if and only if the link structure of s can be created using the push, clone and pop_1 operations. The proof of this claim can be done by a tedious but straightforward induction. We only sketch the most important observations for this fact.

Every letter a of the form $(\sigma, 2, l)$ occurring in s is either a clone or can be created by the $\mathsf{push}_{\sigma,2}$ operation. We call a a clone if a occurs in s in some word waw' such that the word to the left of this word has wa as prefix. Note that cloned elements are those that can be created by use of the clone_2 and pop_1 operations from a certain substack of s.

If a is not a clone in this sense, then Dec creates the letter a because there is some $(\sigma, 2)$-labelled node in T corresponding to a. Now, the important observation is that Dec defines

$a = f_g((\sigma, 2))$ where $g+1$ is the width of the stack decoded from the lexicographically smaller nodes. Hence, the letter a occurs in the $(g+1)$-st word of s and points to the g-th word. Such a letter a can clearly be created by a $\mathsf{push}_{\sigma,2}$ operation. Thus, all 2-words in the image of Dec can be generated by stack operations from the initial stack. A reformulation of this observation is that the image of Dec only contains stacks.

Now, we prove that Dec is injective on $\mathbb{T}_{\mathrm{Enc}}$. Afterwards, we show that $\mathsf{Dec} \circ \mathsf{Enc}$ is the identity on the set of all configurations. This implies that Dec is a surjective map from $\mathbb{T}_{\mathrm{Enc}}$ to $Q \times \mathsf{Stacks}_2(\Sigma)$. Putting both facts together, we obtain the bijectivity of Enc.

Lemma 3.1.13. Dec is injective on $\mathbb{T}_{\mathrm{Enc}}$.

Proof. Assume that there are trees $T', U' \in \mathbb{T}_{\mathrm{Enc}}$ with $\mathsf{Dec}(T') = \mathsf{Dec}(U') = (q, s)$. Then by definition $T'(\varepsilon) = U'(\varepsilon) = q$. Thus, we only have to compare the subtrees rooted at 0, i.e., $T := (T')_0$ and $U := (U')_0$. From our assumption it follows that $\mathsf{Dec}(T, 0) = \mathsf{Dec}(U, 0)$.

Note that the roots of T and of U are both labelled by $(\bot, 1)$.

Now, the lemma follows from the following claim.

Claim. Let T and U be trees such that there are $T', U' \in \mathbb{T}_{\mathrm{Enc}}$ and $d \in \mathsf{dom}(T') \setminus \{\varepsilon\}$, $e \in \mathsf{dom}(U') \setminus \{\varepsilon\}$ such that $T = (T')_d$ and $U = (U')_e$. If $\mathsf{Dec}(T, m) = \mathsf{Dec}(U, m)$ and either $T(\varepsilon) = U(\varepsilon) = \varepsilon$ or $T(\varepsilon) \in \Sigma \times \{1, 2\}$ and $U(\varepsilon) \in \Sigma \times \{1, 2\}$, then $U = T$.

The proof is by induction on the depth of the trees U and T. If $\mathsf{dp}(U) = \mathsf{dp}(T) = 0$, $\mathsf{Dec}(U, m)$ and $\mathsf{Dec}(T, m)$ are uniquely determined by the label of their roots. A straightforward consequence of the definition of Dec is that $U(\varepsilon) = T(\varepsilon)$ whence $U = T$.

Now, assume that the claim is true for all trees of depth at most k for some fixed $k \in \mathbb{N}$. Let U and T be trees of depth at most $k+1$.

We proceed by a case distinction on whether the left or right subtree of T and U are defined. In fact, $\mathsf{Dec}(T, m) = \mathsf{Dec}(U, m)$ implies that

1. $(T)_0 \neq \emptyset$ if and only if $(U)_0 \neq \emptyset$ and
2. $(T)_1 \neq \emptyset$ if and only if $(U)_1 \neq \emptyset$.

We first prove that $\mathsf{Dec}(T, m) = \mathsf{Dec}(U, m)$ implies $U = T$ in the cases satisfying these conditions. Afterwards, we show that all possible combinations that do not satisfy this condition imply $\mathsf{Dec}(T, m) \neq \mathsf{Dec}(U, m)$.

1. Assume that $(U)_0 = (U)_1 = (T)_0 = (T)_1 = \emptyset$. Then $\mathsf{dp}(T) = \mathsf{dp}(U) = 0$. For trees of depth 0 we have already shown that $\mathsf{Dec}(U, 0) = \mathsf{Dec}(T, 0)$ implies $U = T$.

2. Assume that $(U)_0 = \emptyset$, $(U)_1 \neq \emptyset$, $(T)_0 = \emptyset$ and $(T)_1 \neq \emptyset$. In this case

$$\mathsf{Dec}(U, m) = f_m(U(\varepsilon)) \setminus (\varepsilon : \mathsf{Dec}((U)_1, m+1)) \text{ and}$$
$$\mathsf{Dec}(T, m) = f_m(T(\varepsilon)) \setminus (\varepsilon : \mathsf{Dec}((T)_1, m+1)).$$

   Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$, we can directly conclude that $U(\varepsilon) = T(\varepsilon)$. But then $\mathsf{Dec}(T, m) = \mathsf{Dec}(U, m)$ implies that $\mathsf{Dec}((T)_1, m+1) = \mathsf{Dec}((U)_1, m+1)$. Since $\mathsf{dp}((T)_1) \leq k$ and $\mathsf{dp}((U)_1) \leq k$, the induction hypothesis implies that $(T)_1 = (U)_1$. We conclude that $T = U$.

3. Assume that $(U)_0 \neq \emptyset$, $(U)_1 = \emptyset$, $(T)_0 \neq \emptyset$, and $(T)_1 = \emptyset$. In this case,

$$\mathsf{Dec}(U, m) = f_m(U(\varepsilon)) \setminus \mathsf{Dec}((U)_0, m) \text{ and}$$
$$\mathsf{Dec}(T, m) = f_m(T(\varepsilon)) \setminus \mathsf{Dec}((T)_0, m).$$

Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$, we conclude that $U(\varepsilon) = T(\varepsilon)$ and $\mathsf{Dec}((U)_0, m) = \mathsf{Dec}((T)_0, m)$. Since the depths of $(U)_0$ and of $(T)_0$ are at most k, the induction hypothesis implies $(U)_0 = (T)_0$ whence $U = T$.

4. Assume that $(U)_0 \neq \emptyset$, $(U)_1 \neq \emptyset$, $(T)_0 \neq \emptyset$, and $(T)_1 \neq \emptyset$. Then we have

$$\mathsf{Dec}(U, m) = f_m(U(\varepsilon)) \setminus \left(\mathsf{Dec}((U)_0, m) : \mathsf{Dec}((U)_1, m + m')\right) \text{ and}$$
$$\mathsf{Dec}(T, m) = f_m^n(T(\varepsilon)) \setminus \left(\mathsf{Dec}((T)_0, m) : \mathsf{Dec}((T)_1, m + m'')\right)$$

for some natural numbers $m', m'' > 0$.

Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$ this implies that the roots of U and T coincide. Hence,

$$\mathsf{Dec}((U)_0, m) : \mathsf{Dec}((U)_1, m + m') = \mathsf{Dec}((T)_0, m) : \mathsf{Dec}((T)_1, m + m'')$$

If $\mathsf{Dec}((U)_0, m) = \mathsf{Dec}((T)_0, m)$, then the induction hypothesis yields $(U)_0 = (T)_0$. Furthermore, this implies $\mathsf{Dec}((U)_1, m+m') = \mathsf{Dec}((T)_1, m+m'')$ and $m' = m''$ whence by induction hypothesis $(U)_1 = (T)_1$. In this case we conclude immediately that $T = U$.

The other case is that the width of $\mathsf{Dec}((U)_0, m)$ and the width of $\mathsf{Dec}((T)_0, m)$ do not coincide.

We prove that this case contradicts the assumption that $\mathsf{Dec}(U, m) = \mathsf{Dec}(T, m)$.

Let us assume that $\mathsf{Dec}((U)_0, m) = \mathsf{pop}_2^z\left(\mathsf{Dec}((T)_0, m)\right)$ for some $z \in \mathbb{N} \setminus \{0\}$. Note that this implies that the first word of $\mathsf{Dec}((U)_1, m + m')$ is a word in $\mathsf{Dec}((T)_0, m)$.

Since $U(0)$ is a left successor in some tree from $\mathbb{T}_{\mathrm{Enc}}$, it is labelled by some $(\sigma, l) \in \Sigma \times \{1, 2\}$. We make a case distinction on l.

a) Assume that $U(0) = (\sigma, 2)$ for some $\sigma \in \Sigma$. Then all words in $\mathsf{Dec}((T)_0, m)$ start with the letter $(\sigma, 2, m)$. Thus, the first word of $\mathsf{Dec}((U)_1, m+m')$ must also start with $(\sigma, 2, m)$. But all collapse links of level 2 in $\mathsf{Dec}((U)_1, m + m')$ are at least $m + m' > m$. This is a contradiction.

b) Otherwise, $U(1) = (\sigma, 1)$ for some $\sigma \in \Sigma$. Thus, all words in $\mathsf{Dec}((T)_0, m)$ start with the letter σ. Thus, the first word of $\mathsf{Dec}((U)_0, m)$ and the first word of $\mathsf{Dec}((U)_1, m + m')$ have to start with σ. But this requires that $U(0) = U(10) = (\sigma, 1)$. This contradicts the assumption that U is a proper subtree of a tree from $\mathbb{T}_{\mathrm{Enc}}$ (cf. condition 5 of Definition 3.1.8).

Both cases result in contradictions. Thus, it is not the fact that there is some $z \in \mathbb{N} \setminus \{0\}$ such that

$$\mathsf{Dec}((U)_0, m) = \mathsf{pop}_2^z\left(\mathsf{Dec}((T)_0, m)\right)$$

By symmetry, we obtain that there is no $z \in \mathbb{N} \setminus \{0\}$ such that

$$\mathsf{Dec}((T)_0, m) = \mathsf{pop}_2^z\left(\mathsf{Dec}((U)_0, m)\right).$$

Thus, we conclude that $\mathsf{Dec}((T)_0, m) = \mathsf{Dec}((U)_0, m)$ whence $U = T$ as shown above.

If $\mathsf{Dec}(T,m) = \mathsf{Dec}(U,m)$, one of the previous cases applies: the following case distinction shows that all other cases for the defined or undefined subtrees of T and U imply $\mathsf{Dec}(T,m) \neq \mathsf{Dec}(U,m)$.

1. Assume that $(U)_0 = (U)_1 = (T)_0 = \emptyset$ and $(T)_1 \neq \emptyset$. In this case, $\mathsf{Dec}(U,m)$ is $[\varepsilon]$ or $[\tau]$ for some $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Furthermore,

$$\mathsf{Dec}(T,m) = f_m(T(\varepsilon)) \setminus (\varepsilon : \mathsf{Dec}((T)_1, m+1)).$$

   It follows that $|\mathsf{Dec}(T,m)| \geq 2 > |\mathsf{Dec}(U,m)| = 1$ whence $\mathsf{Dec}(T,m) \neq \mathsf{Dec}(U,m)$.

2. Assume that $(U)_0 = (U)_1 = \emptyset$, $(T)_0 \neq \emptyset$, and $(T)_1 = \emptyset$. In this case, $\mathsf{Dec}(U,m)$ is again $[\varepsilon]$ or $[\tau]$ for some $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Since we assumed that $U(\varepsilon) = T(\varepsilon)$,

$$\mathsf{Dec}(T,m) = f_m(T(\varepsilon)) \setminus f_m(T(0)) \setminus s$$

   for some 2-word s. Since T is a subtree of a tree in $\mathbb{T}_{\mathrm{Enc}}$, $T(0) \in \Sigma \times \{1,2\}$. Thus, $f_m(T(0)) \in \Sigma \cup (\Sigma \times \{1,2\} \times \mathbb{N})$. We conclude that the length of the first word of $\mathsf{Dec}(T,m)$ is greater than the length of the first word of $\mathsf{Dec}(U,m)$. Thus, $\mathsf{Dec}(T,m) \neq \mathsf{Dec}(U,m)$.

3. Assume that $(U)_0 = (U)_1 = \emptyset$, $(T)_0 \neq \emptyset$, and $(T)_1 \neq \emptyset$. Completely analogous to case 1, we conclude that $|\mathsf{Dec}(T,m)| \geq 2 > |\mathsf{Dec}(U,m)| = 1$ whence $\mathsf{Dec}(T,m) \neq \mathsf{Dec}(U,m)$.

4. Assume that $(U)_0 = \emptyset$, $(U)_1 \neq \emptyset$, and $(T)_0 = (T)_1 = \emptyset$. Exchanging the roles of U and T, this is exactly the same as case 1.

5. Assume that $(U)_0 = \emptyset$, $(U)_1 \neq \emptyset$, $(T)_0 \neq \emptyset$, and $(T)_1 = \emptyset$. Analogously to case 2, we derive that the length of the first word of $\mathsf{Dec}(T,m)$ is greater than the length of the first word of $\mathsf{Dec}(U,m)$. Thus, $\mathsf{Dec}(T,m) \neq \mathsf{Dec}(U,m)$.

6. Assume that $(U)_0 = \emptyset$, $(U)_1 \neq \emptyset$, $(T)_0 \neq \emptyset$, and $(T)_1 \neq \emptyset$. Analogously to case 2, we derive that the length of the first word of $\mathsf{Dec}(T,m)$ is greater than the length of the first word of $\mathsf{Dec}(U,m)$. Thus, $\mathsf{Dec}(T,m) \neq \mathsf{Dec}(U,m)$.

7. Assume that $(U)_0 \neq \emptyset$, and $(U)_1 = (T)_0 = (T)_1 = \emptyset$. Exchanging the roles of U and T, this is exactly the case 2.

8. Assume that $(U)_0 \neq \emptyset$, $(U)_1 = (T)_0 = \emptyset$, and $(T)_1 \neq \emptyset$. Exchanging the roles of U and T, this is exactly the case 5.

9. Assume that $(U)_0 \neq \emptyset$, $(U)_1 = \emptyset$, $(T)_0 \neq \emptyset$, and $(T)_1 \neq \emptyset$. In this case,

$$\mathsf{Dec}(U,m) = f_m(U(\varepsilon)) \setminus \mathsf{Dec}((U)_0, m)$$
$$\text{and } \mathsf{Dec}(T,m) = f_m(T) \setminus \big(\mathsf{Dec}((T)_0, m) : \mathsf{Dec}((T)_1, m+m')\big)$$

   for some $m' \in \mathbb{N} \setminus \{0\}$. Since $U(\varepsilon) = \varepsilon$ if and only if $T(\varepsilon) = \varepsilon$, we conclude that $U(\varepsilon) = T(\varepsilon)$. Now,

$$\mathsf{Dec}((U)_0, m) = \tau \setminus u'$$

for $\tau = f_m(U(0)) \in \Sigma \cup (\Sigma \times \{2\} \times \{m\})$ and u' some level 2-word. We distinguish the following cases.

First assume that $\tau = (\sigma, 2, m)$. For all letters in $T' := \mathsf{Dec}((T)_1, m + m')$ of collapse level 2, the collapse link is greater or equal to $m + m'$. Hence, T' does not contain a symbol $(\sigma, 2, m)$ whence $\mathsf{Dec}(U, m) \neq \mathsf{Dec}(T, m)$.

Otherwise, $\tau \in \Sigma$. But then $\mathsf{Dec}(U, m) = \mathsf{Dec}(T, m)$ would imply that

$$\mathsf{Dec}((T)_0, m) = \tau \setminus T'$$
$$\text{and } \mathsf{Dec}((T)_{10}, m + m') = \tau \setminus T''$$

for certain nonempty level 2-words T' and T''. But then $T(0) = T(10) = (\tau, 1)$ which contradicts the fact that T is a subtree of some tree from $\mathbb{T}_{\mathsf{Enc}}$.

Thus, we conclude that $\mathsf{Dec}(T, m) \neq \mathsf{Dec}(U, m)$.

10. Assume that $(U)_0 \neq \emptyset$, $(U)_1 \neq \emptyset$, and $(T)_0 = (T)_1 = \emptyset$. Exchanging the roles of U and T, this is the same as case 3.

11. Assume that $(U)_0 \neq \emptyset$, $(U)_1 \neq \emptyset$, $(T)_0 = \emptyset$, and $(T)_1 \neq \emptyset$. Exchanging the roles of U and T, this is the same as case 6.

12. Assume that $(U)_0 \neq \emptyset$, $(U)_1 \neq \emptyset$, $(T)_0 \neq \emptyset$, and $(T)_1 = \emptyset$. Exchanging the roles of U and T, this is the same as case 9.

Hence, we have seen that $\mathsf{Dec}(T, m) = \mathsf{Dec}(U, m)$ implies that each of the subtrees of T is defined if and only if the corresponding subtree of U is defined. Under this condition, we concluded that $U = T$. Thus, the claim holds and the lemma follows as indicated above. □

Next, we prove that Dec is a surjective map from $\mathbb{T}_{\mathsf{Enc}}$ to $Q \times \mathsf{Stacks}_2(\Sigma)$. This is done by induction on the blocklines used to encode a stack. In this proof we use the notion of left-maximal blocks and good blocklines. Let

$$s : \big(w \setminus (w' : b)\big) : s'$$

be a stack where s and s' are 2-words, w, and w' are words, and b is a τ-block. We call b left maximal in this stack if either $b = [\tau]$ or $b = \tau\tau' \setminus b'$ such that w' does not start with $\tau\tau'$ for some $\tau' \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. We call a blockline in some stack good, if its first block is left maximal. Furthermore, we call the blockline starting with the block b left maximal if w' does not start with τ. Recall that the encoding of stacks works on left maximal blocks and good blocklines.

Lemma 3.1.14. $\mathsf{Dec} \circ \mathsf{Enc}$ is the identity on $Q \times \mathsf{Stacks}_2(\Sigma)$, i.e., $\mathsf{Dec}(\mathsf{Enc}(c)) = c$, for all $c \in Q \times \mathsf{Stacks}_2(\Sigma)$.

Corollary 3.1.15. $\mathsf{Dec} : \mathbb{T}_{\mathsf{Enc}} \to Q \times \mathsf{Stacks}_2(\Sigma)$ is surjective.

Proof of Lemma. Let $c = (q, s)$ be a configuration. Since Dec and Enc encode and decode the state of c in the root of $\mathsf{Enc}(c)$, it suffices to show that

$$\mathsf{Dec}(\mathsf{Enc}(s, (\bot, 1)), 0) = s$$

for all stacks $s \in \mathsf{Stacks}_2(\Sigma)$. We proceed by induction on blocklines of the stack s. For this purpose we reformulate the lemma in the following claim.

Claim. Let s' be some stack which decomposes as $s' = s'' : (w \setminus b) : s'''$ such that $b \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{+2}$ is a good τ-blockline for some $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Then

1. $\mathsf{Dec}(\mathsf{Enc}(b, \varepsilon), |s''|) = b'$ for the unique 2-word b' such that $b = \tau \setminus b'$ and
2. if b is left maximal, then $\mathsf{Dec}(\mathsf{Enc}(b, (\sigma, l)), |s''|) = b$ where $\sigma = \mathsf{Sym}(\tau)$ and $l = \mathsf{CLvl}(\tau)$.

Note that the conditions in the second part require that either $\tau \in \Sigma$ or $\tau = (\sigma, 2, |s''|)$ for some $\sigma \in \Sigma$.

The lemma follows from the second part of the claim because every stack is a left maximal $\bot$-blockline.

We prove both claims by parallel induction on the size of b. As abbreviation we set $g := |s''|$. We write $\overset{(1)}{=}(\overset{(2)}{=}$, respectively) when some equality is due to the induction hypothesis of the first claim (the second claim , respectively). The arguments for the first claim are as follows.

- If $b = [\tau]$ for $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$, the claim is trivially true because $\mathsf{Dec}(\mathsf{Enc}(b, \varepsilon), g) = \mathsf{Dec}(\varepsilon, g) = \varepsilon$.
- If there are $b_1, b_1' \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{*2}$ such that

$$b = [\tau] : b_1 = [\tau] : \left(\tau \setminus b_1'\right) \text{ then}$$

$$\begin{aligned}
&\mathsf{Dec}(\mathsf{Enc}(b, \varepsilon), g) = \mathsf{Dec}(\varepsilon \to \mathsf{Enc}(b_1, \varepsilon), g) \\
=&f_g(\varepsilon) \setminus \left(\varepsilon : \mathsf{Dec}(\mathsf{Enc}(b_1, \varepsilon), g + 1)\right) \\
\overset{(1)}{=}&\varepsilon \setminus (\varepsilon : b_1') = \varepsilon : b_1' = b'.
\end{aligned}$$

- Assume that there is some $\tau' \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ and some $b_1 \in (\Sigma \cup (\Sigma \times \{2\} \times \mathbb{N}))^{*2}$ such that

$$b = \tau\tau' \setminus b_1.$$

  The assumption that b is good implies that the blockline $\tau' \setminus b_1$ is left maximal whence

$$\begin{aligned}
&\mathsf{Dec}(\mathsf{Enc}(b, \varepsilon), g) = \mathsf{Dec}(\mathsf{Enc}(\tau' \setminus b_1, (\mathsf{Sym}(\tau'), \mathsf{CLvl}(\tau'))) \leftarrow \varepsilon, g) \\
=&f_g(\varepsilon) \setminus \mathsf{Dec}(\mathsf{Enc}(\tau' \setminus b_1, (\mathsf{Sym}(\tau'), \mathsf{CLvl}(\tau')), g)) \\
\overset{(2)}{=}&\tau' \setminus b_1 = b'.
\end{aligned}$$

- The last case is that

$$b = \tau \setminus \left((\tau' \setminus b_1) : b_2\right)$$

for b_2 a blockline of s not starting with τ'. By this we mean that $b_2 \neq \tau' w' : b_2'$ for any word w' and any 2-word b_2'. Since b is good, $\tau' \setminus b_1$ is a left maximal blockline. Furthermore, $\tau \setminus b_2$ is a good blockline. Thus,

$$\begin{aligned}
&\mathsf{Dec}(\mathsf{Enc}(b,\varepsilon),g)\\
=&\mathsf{Dec}\left(\mathsf{Enc}\left(\tau' \setminus b_1,(\mathsf{Sym}(\tau'),\mathsf{CLvl}(\tau'))\right) \leftarrow \varepsilon \rightarrow \mathsf{Enc}(\tau \setminus b_2,\varepsilon),g\right)\\
=&f_g(\varepsilon) \setminus\\
&\left(\mathsf{Dec}(\mathsf{Enc}(\tau' \setminus b_1,(\mathsf{Sym}(\tau'),\mathsf{CLvl}(\tau'))),g) : \mathsf{Dec}(\mathsf{Enc}(\tau \setminus b_2,\varepsilon),g+f)\right),
\end{aligned}$$

where

$$f = |\mathsf{Dec}(\mathsf{Enc}(\tau' \setminus b_1,(\mathsf{Sym}(\tau'),\mathsf{CLvl}(\tau'))),g)| \overset{(2)}{=} |b_1|.$$

From this, we obtain that

$$\begin{aligned}
&\mathsf{Dec}(\mathsf{Enc}(b,\varepsilon),g)\\
\overset{(1)}{=}&\varepsilon \setminus \left((\tau' \setminus b_1) : \mathsf{Dec}(\mathsf{Enc}(\tau \setminus b_2,\varepsilon),g+f)\right)\\
\overset{(2)}{=}&(\tau' \setminus b_1) : b_2 = b'.
\end{aligned}$$

For the proof of the second claim, note that the calculations are basically the same, but $f_g(\varepsilon)$ is replaced by $f_g(\sigma,l)$. Thus, if $l = 1$ then $f_g(\sigma,l) = \sigma = \tau$. For the case $l = 2$, recall that $g = |s''|$ whence $f_g(\sigma,l) = (\sigma,2,|s''|)$. Note that $\mathsf{CLnk}(\tau) = |s''|$ due to the left maximality of b.

Thus, one proves the second case using the same calculations, but replacing ε by τ. □

The previous lemmas provide a proof of Lemma 3.1.10: we have shown that Dec is bijective and it is the inverse of Enc.

3.1.2 Recognising Reachable Configurations

In this section, we show that Enc maps the reachable configurations of a given collapsible pushdown system to a regular set.

Fix a configuration $c = (q,s)$. Recall that every run from the initial configuration to some stack s has to pass each of the generalised milestones $\mathsf{GMS}(s)$ of s (cf. Section 2.4.1). Especially, the set of milestones $\mathsf{MS}(s) \subseteq \mathsf{GMS}(s)$ has a close connection to our encoding: with every $d \in \mathsf{Enc}(c)$, we can associate a subtree of $\mathsf{Enc}(c)$ which encodes a milestone. Via this correspondence, the substack relation on the milestones corresponds exactly to the lexicographic order of the elements of $\mathsf{Enc}(c)$.

We show the regularity of the set of encodings of reachable configurations as follows. Given the tree $\mathsf{Enc}(c)$, we annotate each node $d \in \mathsf{Enc}(c)$ with a state q_d. This annotation represents the claim that there is a run from the initial configuration to c that passes the milestone associated with d in state q_d. Then we show that an automaton can check the correctness of such an annotation. Since this annotation can be generated nondeterministically by an automaton, it follows that the set of encodings of reachable configurations is regular.

The correspondence between nodes of $\mathsf{Enc}(s)$ and milestones of s is established via the notion of the left stack induced by $d \in \mathsf{dom}(\mathsf{Enc}(s))$. This left stack is the decoding of the subtree of $\mathsf{Enc}(s)$ which contains all nodes that are lexicographically smaller than d. We show that these left stacks always form milestones and that each milestone can be represented by such an element.

Definition 3.1.16. Let $T \in \mathbb{T}_{\mathsf{Enc}}$ be a tree and $d \in T \setminus \{\varepsilon\}$. Then the left and downward closed tree of d is $LT(d,T) := T\restriction_D$ where $D := \{d' \in T : d' \leq_{\mathrm{lex}} d\} \setminus \{\varepsilon\}$. Then we denote by $\mathsf{LStck}(d,T) := \mathsf{Dec}(LT(d,T),0)$ the left stack induced by d. If T is clear from the context, we omit it.

Remark 3.1.17. We exclude the case $d = \varepsilon$ from the definition because the root encodes the state of the configuration and not a part of the stack. In the following, we are often interested in the stack encoded in a tree, whence we will consider all nodes except for the root of the encoding tree.

Recall that $w := \mathsf{top}_2(\mathsf{LStck}(d,s))\downarrow_0$ is $\mathsf{top}_2(\mathsf{LStck}(d,s))$ where all level 2 links are set to 0 (cf. Definition 2.4.24). Due to the definition of the encoding, for every $d \in \mathsf{dom}(\mathsf{Enc}(s))$, w is determined by the path from the root to d: interpreting ε as empty word, the word along this path contains the pairs of stack symbols and collapse levels of the letters of $\mathsf{top}_2(\mathsf{LStck}(d,s))$. Since all level 2 links in w are 0, w is determined by this path. Thus, Proposition 2.4.47 implies that there is an automaton that calculates at each position $d \in \mathsf{Enc}(q,s)$ the number of possible loops of $\mathsf{LStck}(d,\mathsf{Enc}(q,s))$ with given initial and final state.

Remark 3.1.18. $\mathsf{LStck}(d,\mathsf{Enc}(q,s))$ is a substack of s for all $d \in \mathsf{dom}(\mathsf{Enc}(q,s))$. This observation follows from Remark 3.1.7 combined with the fact that the left stack is induced by a lexicographically downward closed subset.

Lemma 3.1.19. Let $q \in Q$ and $s \in \mathsf{Stacks}_2(\Sigma)$. For each $d \in \mathsf{Enc}(q,s) \setminus \{\varepsilon\}$ we have $\mathsf{LStck}(d,\mathsf{Enc}(q,s)) \in \mathsf{MS}(s)$. Furthermore, for each $s' \in \mathsf{MS}(s)$ there is some $d \in \mathsf{Enc}(q,s) \setminus \{\varepsilon\}$ such that $s' = \mathsf{LStck}(d,\mathsf{Enc}(q,s))$.

Proof. For the first claim, let $d \in \mathsf{dom}(\mathsf{Enc}(q,s)) \setminus \{\varepsilon\}$. We already know that $s_d := \mathsf{LStck}(d,\mathsf{Enc}(q,s))$ is a substack of s.

Recall that the path from the root to s_d encodes $\mathsf{top}_2(s_d)$. Furthermore, by definition of Enc, d corresponds to some maximal block b occurring in s in the following sense: there are 2-words s_1, s_2 and a word w such that $s = s_1 : (w \setminus b) : s_2$ and such that the subtree rooted at d encodes b. Moreover, d encodes the first letter of b, i.e., if b is a τ-block, then the path from the root to d encodes $w\tau$.

Note that by maximality of b, the greatest common prefix of the last word of s_1 and the first word of $w \setminus b$ is a prefix of $w\tau$.

Since the elements that are lexicographically smaller than d encode the blocks to the left of b, one sees that $s_d = s_1 : w\tau$. Setting $k := |s_d|$, we conclude that s_d is a substack of s such that the greatest common prefix of the $(k-1)$-st and the k-th word of s is a prefix of $\mathsf{top}_2(s_d)$.

Recall that this is exactly a characterisation of a milestone of s. Thus, s_d is a milestone of s and we completed the proof of the first claim.

Now, we turn to the second claim. The fact that every milestone $s' \in \mathsf{MS}(s)$ is indeed represented by some node of $\mathsf{Enc}(q,s)$ can be seen by induction on the block structure of s'. Assume that $s' \in \mathsf{MS}(s)$ and that s' decomposes as $s' = b_0 : b_1 : \cdots : b_{m-1} : b'_m$ into

maximal blocks. We claim that s then decomposes as $s = b_0 : b_1 : \cdots : b_{m-1} : b_m : \cdots : b_n$ into maximal blocks. In order to verify this claim, we have to prove that b_{m-1} cannot be the initial segment of a larger block $b_{m-1} : b_m$ in s. Note that if b'_m only contains one letter, then by definition of a milestone the last word of b_{m-1} and the first word occurring in s after b_{m-1}, which is the first word of b_m, can only have a common prefix of length at most 1. Hence, their composition does not form a block. Otherwise, the first word of b'_m contains two letters which do not coincide with the first two letters of the words in b_{m-1}. Since this word is by definition a prefix of the first word in b_m, we can conclude again that $b_{m-1} : b_m$ does not form a block.

Note that all words in the blocks b_i for $1 \leq i \leq n$ and in the block b'_m share the same first letter which is encoded at the position 0 in $\mathsf{Enc}(q,s)$ and in $\mathsf{Enc}(q,s')$. By the definition of $\mathsf{Enc}(q,s)$ the blockline induced by b_i is encoded in the subtree rooted at 01^i0 in $\mathsf{Enc}(q,s)$. For $i < m$ the same holds in $\mathsf{Enc}(q,s')$. We set $d := 01^m$. Note that $\mathsf{Enc}(q,s')$ and $\mathsf{Enc}(q,s)$ coincide on all elements that are lexicographically smaller than d (because these elements encode the blocks $b_1 : b_2 : \ldots b_{m-1}$.

Now, we distinguish the following cases.

1. Assume that $b'_m = [\tau]$ for $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$. Then the block b'_m consists of only one letter. In this case d is the lexicographically largest element of $\mathsf{Enc}(q,s')$ whence $s' = \mathsf{LStck}(d, \mathsf{Enc}(q,s')) = \mathsf{LStck}(d, \mathsf{Enc}(q,s))$.

2. Otherwise, there is a $\tau \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ such that

$$b_m = \tau \setminus (c_0 : c_1 : \cdots : c_{m'} : \cdots : c_{n'}) \text{ and}$$
$$b'_m = \tau \setminus (c_0 : c_1 : \cdots : c_{m'-1} : c'_{m'})$$

   for some $m' \leq n'$ such that $c_0 : c_1 : \cdots : c_{n'}$ are the maximal blocks of the blockline induced by b_m and $c_0 : c_1 : \ldots c_{m'-1} : c'_{m'}$ are the maximal blocks of the blockline induced by b'_m. Now, $c_1 : c_2 : \cdots : c_{m'-1}$ are encoded in the subtrees rooted at $d01^i0$ for $0 \leq i \leq m'-1$ in $\mathsf{Enc}(q,s)$ as well as in $\mathsf{Enc}(q,s')$. $c_{m'+1} : c_{m'+2} : \cdots : c_{n'}$ is encoded in the subtree rooted at $d01^{m'+1}$ in $\mathsf{Enc}(q,s)$ and these elements are all lexicographically larger than $d01^{m'}0$. Hence, we can set $d' := d01^{m'}$ and repeat this case distinction on $d', c'_{m'}$ and $c_{m'}$ instead of d, b'_m and b_m.

Since s' is finite, by repeated application of the case distinction, we will eventually end up in the first case where we find a $d \in \mathsf{Enc}(q,s)$ such that $s' = \mathsf{LStck}(d, \mathsf{Enc}(q,s))$. □

The next lemma states the tight connection between milestones of a stack (with substack relation) and elements in the encoding of this stack (with lexicographic order).

Lemma 3.1.20. The map

$$g : \mathsf{dom}(\mathsf{Enc}(q,s)) \setminus \{\varepsilon\} \to \mathsf{MS}(s)$$
$$d \mapsto \mathsf{LStck}(d,s)$$

is an order isomorphism between $\big(\mathsf{dom}(\mathsf{Enc}(q,s)) \setminus \{\varepsilon\}, \leq_{\mathrm{lex}}\big)$ and $\big(\mathsf{MS}(s), \leq\big)$.

Proof. If the successor of d in lexicographic order is $d0$, then the left stack of the latter extends the former by just one letter. Otherwise, the left and downward closed tree of the successor of d contains more elements ending in 1, whence it encodes a stack of larger width. Since each left and downward closed tree induces a milestone, it follows that g is an order isomorphism. □

Recall that by Lemma 2.4.10, each run to a configuration (q,s) visits the milestones of s in the order given by the substack relation. With the previous lemma, this translates into the fact that the left stacks induced by the elements of $\mathsf{Enc}(q,s)$ are visited by the run in lexicographical order of the elements of $\mathsf{Enc}(q,s)$.

This gives rise to the following algorithm for identifying reachable configurations of a collapsible pushdown system $\mathscr{S}$: we label each node d of the encoding with a state q_d. Let s_d be the left stack induced by each d. Fix a d and let d' be the lexicographical successor of d. Then we check whether there is a run from (q_d, s_d) to $(q_{d'}, s_{d'})$.

In the next section we show that this check depends only on the local structure of the encoding of a configuration. Hence, an automaton can do this check.

Detection of Reachable Configurations

We have already seen that every run to a valid configuration (q,s) passes all the milestones of s. Now, we use the last states in which a run ρ to (q,s) visits the milestones as a certificate for the reachability of (q,s).

Definition 3.1.21. Let (q,s) be some configuration and ρ a run from the initial configuration to (q,s). The certificate for the reachability of (q,s) induced by ρ is the map $C_\rho : \mathsf{dom}(\mathsf{Enc}(q,s)) \setminus \{\varepsilon\} \to Q$ such that $d \mapsto \hat{q}$ if and only if $\rho(i) = (\hat{q}, \mathsf{LStck}(d))$ and i is the maximal position in ρ where $\mathsf{LStck}(d, \mathsf{Enc}(q,s))$ is visited.

Remark 3.1.22. In the following, we identify a function $f : \mathsf{dom}(\mathsf{Enc}(q,s)) \setminus \{\varepsilon\} \to Q$ with the $Q \cup \{\square\}$-labelled tree $\hat{f} : \mathsf{dom}(\mathsf{Enc}(q,s)) \to Q \cup \{\square\}$ where $\hat{f}(d) = \begin{cases} \square & \text{if } d = \varepsilon, \\ f(d) & \text{otherwise.} \end{cases}$

In the following, we analyse the existence of certificates for reachability. The existence of certain loops plays an important role in this analysis. Thus, we first fix some notation concerning the existence of returns and loops. Recall that we defined the functions $\#\mathsf{Ret}^k$, $\#\mathsf{Loop}^k$, etc. (cf. Definitions 2.4.15 and 2.4.46) that count up to threshold k the number of returns and loops starting in a given configuration. Recall that there is a loop from (q,s) to (q',s) if and only if $\#\mathsf{Loop}^k(s)(q,q') \geq 1$ for all $k \geq 1$.

Definition 3.1.23. We set

$$\exists\mathsf{Loops}(s) := \{(q,q') \in Q \times Q : \#\mathsf{Loop}^1(s)(q,q') = 1\}.$$

$\exists\mathsf{Loops}(s)$ contains those pairs of states q, q' such that there exists at least one loop from (q,s) to (q',s). Completely analogously, we set

$$\exists\mathsf{HLoops}(s) := \{(q,q') \in Q \times Q : \#\mathsf{HLoop}^1(s)(q,q') = 1\},$$
$$\exists\mathsf{LLoops}(s) := \{(q,q') \in Q \times Q : \#\mathsf{LLoop}^1(s)(q,q') = 1\} \text{ and}$$
$$\exists\mathsf{Returns}(s) := \{(q,q') \in Q \times Q : \#\mathsf{Ret}^1(s)(q,q') = 1\}.$$

These sets contain the pairs of initial and final states of low loops, high loops and returns starting with stack s.

Remark 3.1.24. Due to Remark 3.1.17 and due to Proposition 2.4.47, the function that assigns

$$d \mapsto \exists\mathrm{Loops}(\mathrm{LStck}(d, \mathrm{Enc}(q,s)))$$

is calculated by some automaton for all configurations (q,s). Analogous, the function that assigns

$$d \mapsto \exists\mathrm{HLoops}(\mathrm{LStck}(d, \mathrm{Enc}(q,s)))$$

is also calculated by some automaton.

Using this notation, we can prove the first important lemma concerning certificates for reachability.

Lemma 3.1.25. For every CPG G, there is an automaton $\mathscr{A}$ that checks for each map

$$f : \mathrm{dom}(\mathrm{Enc}(q,s)) \setminus \{\varepsilon\} \to Q$$

whether f is a certificate for the reachability of (q,s). This means that $\mathscr{A}$ accepts $\mathrm{Enc}(q,s) \otimes f$ if $f = C_\rho$ for some run ρ from the initial configuration to (q,s).

Proof. As before, we identify f with a $Q \cup \{\Box\}$-labelled tree encoding f. Due to the previous remark, it is sufficient to prove that there is an automaton which accepts

$$\mathrm{Enc}(q,s) \otimes f \otimes T_{\mathrm{Lp}} \otimes T_{\mathrm{HLp}}$$
$$\text{if and only if } f = C_\rho \text{ for some run } \rho,$$

where T_{Lp} is a tree encoding the value of $\exists\mathrm{Loops}(\mathrm{LStck}(d, \mathrm{Enc}(q,s)))$ at each node $d \in \mathrm{dom}(\mathrm{Enc}(q,s))$ and T_{HLp} is a tree encoding the value of $\exists\mathrm{HLoops}(\mathrm{LStck}(d, \mathrm{Enc}(q,s))$. We write $T := \mathrm{Enc}(q,s)$ as an abbreviation. We start with an informal description what we have to check at some node $d \in \mathrm{dom}(T)$. According to Corollary 2.4.9, it is sufficient to check the following facts.

1. Assume that $d, d0 \in \mathrm{dom}(t)$. We know that $\mathrm{LStck}(d,T) = \mathrm{pop}_1(\mathrm{LStck}(d0,T))$. By definition, we know that f can only be a certificate for reachability if there is a run ρ' from $\big(f(d), \mathrm{LStck}(d,T)\big)$ to $\big(f(d0), \mathrm{LStck}(d0,T)\big)$ that starts with some push operation followed by a high loop of $\mathrm{LStck}(d0,T)$. This requirement can be checked by an automaton when it reads the labels $t(d)$ and $t(d0)$ as follows.

   We assume that the automaton has stored the information about the topmost symbol σ of $\mathrm{LStck}(d,T)$. When it reads $t(d)$ it guesses nondeterministically a pair $(q', (\sigma', i))$ for $q' \in Q$, $\sigma' \in \Sigma$ and $i \in \{1,2\}$ such that there is a $\mathrm{push}_{\sigma',i}$ transition from state $f(d)$ and topmost symbol σ going to state q'. Reading the label $t(d0)$ it checks whether $\mathrm{Enc}(q,s)(d0) = (\sigma', i)$ and whether $\big(q', f(d0)\big) \in \exists\mathrm{Loops}\,(\mathrm{LStck}(d0,T))$. If this is the case then the automaton guessed the right push transition and there is a run from $(f(d), \mathrm{LStck}(d,T))$ to $(f(d0), \mathrm{LStck}(d0,T))$.

2. Consider the case where $d \in \mathrm{dom}(T)$ but $d0 \notin \mathrm{dom}(T)$ and where d has a successor d' in lexicographic order. This implies that the direct successor of $\mathrm{LStck}(d,T)$ in $\mathrm{MS}(s)$ is of the form

$$s' := \mathrm{pop}_1^m(\mathrm{clone}_2(\mathrm{LStck}(d,T))).$$

In this case there is a maximal prefix $d_0 \leq d$ and some $d_1 \in \{0,1\}^*$ such that $d = d_0 0 d_1$ and $d' = d_0 1 \in \mathrm{dom}(T)$. Due to Lemma 3.1.20, we know that $s' = \mathrm{LStck}(d',T)$.

From our observations about milestones we know that we have to verify that there is some run $\rho' := \rho_0 \circ \lambda_0 \circ \rho_1 \circ \lambda_1 \circ \rho_2 \circ \lambda_2 \ldots \rho_m \circ \lambda_m$ where λ_i is a loop for all $0 \leq i \leq m$ and ρ_0 is a run that performs one clone operation and for $j > 0$ the run ρ_j performs either one pop_1 or one $\mathrm{collapse}$ of level 1 such that ρ' starts in $\big(f(d), \mathrm{LStck}(d,T)\big)$ and ends in $\big(f(d'), \mathrm{LStck}(d',T)\big)$.

An automaton can verify this because the path from d_0 to d encodes the topmost stack symbols and collapse levels of $\mathrm{pop}_1^{m'}(\mathrm{clone}_2(\mathrm{LStck}(d,T)))$ for $m' \leq m$. Since the existence of loops only depends on the topmost word, an automaton can check the existence of ρ' while processing the path from d to d_0.

3. Finally, we have to consider the lexicographically minimal and maximal element in the encoding of the stack. Let d be the rightmost leaf of T. Recall that $\mathrm{LStck}(d,T) = s$. f can only be a certificate for reachability for (q,s) if it labels d with the last state in which s is visited. But if ρ is a run to (q,s) then this last state must be q. Thus, the condition for the rightmost leaf d is that $f(d) = q$.

Recall that $\mathrm{LStck}(0,T) = [\bot]$. Due to Corollary 2.4.9, the run starts with a loop from the initial configuration to some configuration $(\hat{q}, [\bot])$. Hence, we have to check whether $f(0) = \hat{q}$.

The lemma claims that there is an automaton $\mathscr{A}$ checking these conditions. Instead of a concrete construction of $\mathscr{A}$, we present an MSO formula χ that checks at each node $d \in \mathrm{dom}(\mathrm{Enc}(q,s))$ the corresponding condition. Due to the correspondence between MSO definability and automata recognisability, the automaton $\mathscr{A}$ can be constructed from this formula using standard constructions.

1. For the first condition consider the formula

$$\chi_1 := \forall x \forall y (\neg \mathrm{Root}(x) \wedge y = x0) \rightarrow \left(\bigvee_{(q_1,\sigma,\gamma,q_2,\mathrm{push}_{\tau,i}) \in \Delta} \mathrm{Sym}(x) = \sigma \wedge \mathrm{Top}(y) = (\tau,i) \wedge f(x) = q_1 \wedge (q_2, f(y)) \in \mathrm{HLp}(y) \right),$$

where

- $\mathrm{Root}(x)$ is the formula stating that x is the root of the tree, i.e., x has no predecessor,
- $\mathrm{Sym}(x) = \sigma$ is an MSO formula stating that the maximal 1-ancestor z of x satisfies $\mathrm{Enc}(q,s)(z) = (\sigma,i)$ for some $i \in \{1,2\}$,
- $\mathrm{Top}(y) = (\tau,i)$ is a formula stating that $\mathrm{Enc}(q,s)(y) = (\tau,i)$, and

- $(q_2, f(y)) \in \mathrm{HLp}(y)$ asserts that $T_{\mathrm{HLp}}(q_2, f(y)) = 1$, i.e., it asserts that $(q_2, f(y)) \in \exists\mathrm{HLoops}(\mathrm{LStck}(y, T))$.

This formula asserts exactly the conditions of the first case at all nodes x that have a left successor. Note that we exclude the root of the tree because it encodes the state of the configuration and not a part of the stack.

2. For the second case, let $\varphi(x, y, X)$ be an MSO formula that is valid if x does not have a left successor, if y is the successor of x with respect to lexicographic ordering and if X contains the path connecting the predecessor of y with x.

   Assume that there is a triple (x, y, X) that satisfies φ on $\mathrm{Enc}(q, s)$. Then there are a node $z \in \mathrm{dom}(\mathrm{Enc}(q, s))$, a number $k \in \mathbb{N}$ and numbers $n_1, n_2, \ldots, n_k \in \mathbb{N}$ such that $y = z1$ and $x = z01^{n_1}01^{n_2}\ldots01^{n_k}$. Then $X = \{a : z \leq a \leq x\}$. For each node $a \in X$, there is some $0 \leq l \leq k$ and a number $n_l' \leq n_l$ such that $a = y01^{n_1}01^{n_2}\ldots01^{n_{l-1}}01^{n_l'}$. Since the path to a encodes the topmost word of the left stack induced by a, setting $k_a := k - l$ we obtain that

$$\mathrm{top}_2(\mathrm{LStck}(a, \mathrm{Enc}(q, s))) = \mathrm{top}_2(\mathrm{pop}_1^{k_a}(\mathrm{clone}_2(\mathrm{LStck}(x, \mathrm{Enc}(q, s))))).$$

   Furthermore,

$$\mathrm{LStck}(y, \mathrm{Enc}(q, s)) = \mathrm{pop}_1^{k}(\mathrm{clone}_2(\mathrm{LStck}(x, \mathrm{Enc}(q, s)))).$$

   We will use the following abbreviations:

$$s_a := \mathrm{LStck}(a, \mathrm{Enc}(q, s)) \text{ and}$$
$$\hat{s}_a := \mathrm{pop}_1^{k_a}(\mathrm{clone}_2(\mathrm{LStck}(x, \mathrm{Enc}(q, s)))).$$

   By definition, $\exists\mathrm{Loops}(s_a) = \exists\mathrm{Loops}(\hat{s}_a)$. We use a as the representative for $\hat{s}_a$.

   We next define a formula χ_2. χ_2 asserts the existence of a function $g : X \to Q$ that labels each node $a \in X$ with a state q_a such that there is a pop_1 or collapse of level 1 followed by a loop which connects $(q_a, \hat{s}_a)$ with $(q_b, \hat{s}_b)$ for $b \leq a$ some node such that $k_b = k_a + 1$. Furthermore, the formula asserts that there is a run from $(f(x), s_x)$ to $(g(x), \hat{s}_x) = (g(x), \mathrm{clone}_2(s_x))$ and it asserts that $g(z) = f(y)$. Note that such a labelling g is exactly a witness for a run $\rho' = \rho_0 \circ \lambda_0 \circ \rho_1 \circ \lambda_1 \circ \rho_2 \circ \lambda_2 \ldots \rho_m \circ \lambda_m$ as described above.

   Let χ_2 be the formula

$$\forall x, y \forall X \Big(\varphi(x, y, X) \rightarrow$$
$$\exists g : X \to Q \left(\bigvee_{(q_1, \sigma, \gamma, q_2, \mathrm{clone}_2) \in \Delta} \big(\mathrm{Sym}(x) = \sigma \wedge f(x) = q_1 \wedge (q_2, g(x)) \in \mathrm{Lp}(x)\big) \right.$$
$$\left. \wedge \psi(g, X) \wedge \exists z(z1 = y \wedge f(y) = g(z)) \right) \Big)$$

where

$$\psi(g,X) := \forall v,z \in X \left((z = v1 \to g(z) = g(v)) \wedge \left(z = v0 \to (\psi_p \vee \psi_c) \right) \right),$$

$$\psi_p(v,z) := \bigvee_{(q_1,\sigma,\gamma,q_2,\mathrm{pop}_1)\in\Delta} \left(\mathrm{Sym}(z) = \sigma \wedge g(z) = q_1 \wedge (q_2, g(v)) \in \mathrm{Lp}_s(v) \right) \text{ and}$$

$$\psi_p(v,z) := \bigvee_{(q_1,\sigma,\gamma,q_2,\mathrm{collapse})\in\Delta} \left(\mathrm{Top}(z) = (\sigma,1) \wedge g(z) = q_1 \wedge (q_2, g(v)) \in \mathrm{Lp}_s(v) \right).$$

Note that the function g has finite range whence it may be encoded in a finite number of set-variables. Thus, χ_2 can be formalised in MSO.

3. Let χ_3 be the formula asserting that
   a) the rightmost leaf d of $\mathrm{Enc}(q,s)$ satisfies $f(d) = q$, and that
   b) $(q_0, f(0)) \in \exists \mathrm{Loops}([\bot])$, i.e., if $T_{\mathrm{Lp}}(0)(q_0, f(0)) = 1$.

Now, $\mathrm{Enc}(q,s) \otimes f \otimes T_{\mathrm{Lp}} \otimes T_{\mathrm{HLp}} \models \chi := \chi_1 \wedge \chi_2 \wedge \chi_3$ if and only if $f = C_\rho$ for some run ρ from the initial configuration to (q,s). □

Since regular tree-languages are closed under projection, there is an automaton that non-deterministically guesses the existence of a certificate for reachability for each encoding of a reachable configuration.

Corollary 3.1.26. For every collapsible pushdown system $\mathscr{S}$ of level 2, there is an automaton $\mathscr{A}$ that accepts a tree T if and only if $T = \mathrm{Enc}(q,s)$ for a reachable configuration (q,s) of $\mathscr{S}$.

Proof. Note that $T = \mathrm{Enc}(q,s)$ for an arbitrary configuration if and only if $T \in \mathbb{T}_{\mathrm{Enc}}$ which is a regular set. Furthermore, the set of encodings of reachable configurations forms a regular subset of $\mathbb{T}_{\mathrm{Enc}}$ due to the previous lemma and due to the closure of regular languages under projection. □

3.1.3 Regularity of the Stack Operations

In the previous section, we have seen that the function Enc translates the reachable configurations of a collapsible pushdown graph $\mathscr{S}$ (of level 2) into a regular tree language. In order to prove that $\mathrm{CPG}(\mathscr{S})$ is automatic, we have to define automata recognising the transition relations $\vdash^\gamma$ for every $\gamma \in \Gamma$. In fact, we will prove that for each transition $(q,\sigma,\gamma,q',\mathrm{op}) \in Q \times \Sigma \times \Gamma \times Q \times \mathrm{OP}$ the set

$$\{(\mathrm{Enc}(q,s), \mathrm{Enc}(q',s')) : \mathrm{Sym}(s) = \sigma \text{ and } \mathrm{op}(s) = s'\}$$

is regular. In preparation of this proof, we analyse the relationship between the encodings of the stack s and the stack $s' := \mathrm{pop}_2(s)$.

Lemma 3.1.27. Let $c = (q,s)$ and $c' = (q',s')$ be configurations of a pushdown system $\mathscr{S}$ such that $s' = \mathrm{pop}_2(s)$. There is a unique element $t \in \mathrm{Enc}(c')$ such that $t \in \mathrm{Enc}(c) \setminus \mathrm{Enc}(c')$ For $D := \{d \in \mathrm{dom}(\mathrm{Enc}(c)) : t1 \not\leq d\}$, we have

$$\mathrm{dom}(\mathrm{Enc}(c)) \setminus \mathrm{dom}(\mathrm{Enc}(c')) \subseteq t10^*$$
$$\text{and } \mathrm{Enc}(c') = \mathrm{Enc}(c)\restriction_D \text{ (see Figure 3.3).}$$

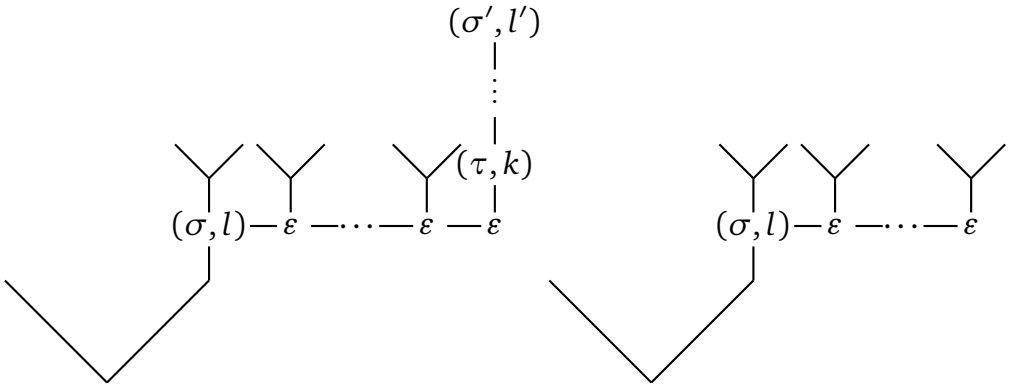


Figure 3.3.: pop_2 operation in the tree-encoding.

Proof. The proof is by induction on the structure of $\mathsf{Enc}(s, (\bot, 1))$ and $\mathsf{Enc}(s', (\bot, 1))$. In fact, we prove the following stronger claim.

Claim. Let $\tau \in (\Sigma \times \{1, 2\}) \cup \{\varepsilon\}$. Let t be the maximal element such that

- t is in the rightmost path of $\mathsf{dom}(\mathsf{Enc}(s, \tau))$,
- $t \in \mathsf{dom}(\mathsf{Enc}(s', \tau))$ and
- $t1 \in \mathsf{dom}(\mathsf{Enc}(s, \tau)) \setminus \mathsf{dom}(\mathsf{Enc}(s', \tau))$.

Set $D := \{d \in \mathsf{dom}(\mathsf{Enc}(s, \tau)) : t1 \not\leq d\}$. It holds that

$$\mathsf{dom}(\mathsf{Enc}(s, \tau)) \setminus \mathsf{dom}(\mathsf{Enc}(s', \tau)) \subseteq t10^*$$
$$\text{and } \mathsf{Enc}(s', \tau) = \mathsf{Enc}(s, \tau)\restriction_D \text{ (see Figure 3.3).}$$

Recall that for some stack consisting of just one word w_1, its encoding $\mathsf{Enc}(w_1, \varepsilon)$ is a path with 0-edges only, i.e., $\mathsf{dom}(\mathsf{Enc}(w_1, \varepsilon)) \subseteq \{0\}^*$.

Let $s := w_1 : w_2 : \cdots : w_n : w_{n+1}$ and correspondingly $s' := w_1 : w_2 : \cdots : w_n$. In the case that $|w_1| \geq 1$, let $\tau_1, \tau_2 \in \Sigma \cup (\Sigma \times \{2\} \times \mathbb{N})$ and w_1' some word such that $w_1 = \tau_1\tau_2 w_1'$. We prove the lemma by induction on the size of s. We distinguish the following cases.

1. For all $i \leq n$ there are words w_i' such that $w_i = \tau_1\tau_2 w_i'$, but $\tau_1\tau_2 \not\leq w_{n+1}$. Then the root in $\mathsf{Enc}(s', \tau)$ has only a left successor and $\mathsf{Enc}(s, \tau)$ extends $\mathsf{Enc}(s', \tau)$ by a right subtree of the root which is $\mathsf{Enc}(w_{n+1}, \varepsilon)$. Due to our initial remark on the structure of the encoding of a single word, the claim follows immediately.

2. For all $i \leq n + 1$, there are words w_i' such that $w_i = \tau_1\tau_2 w_i'$. In this case $\mathsf{Enc}(s, \tau)$ and $\mathsf{Enc}(s', \tau)$ coincide on their roots, these roots do not have right successors and the subtrees induced by the left successor are

$$\mathsf{Enc}\left(\tau_2 \setminus (w_1' : \cdots : w_n' : w_{n+1}'), \left(\mathsf{Sym}(\tau_2), \mathsf{CLvl}(\tau_2)\right)\right)$$
$$\text{and } \mathsf{Enc}\left(\tau_2 \setminus (w_1' : \cdots : w_n'), \left(\mathsf{Sym}(\tau_2), \mathsf{CLvl}(\tau_2)\right)\right).$$

   Now, we apply again the same case distinction to the subtrees encoding these parts of the stacks.

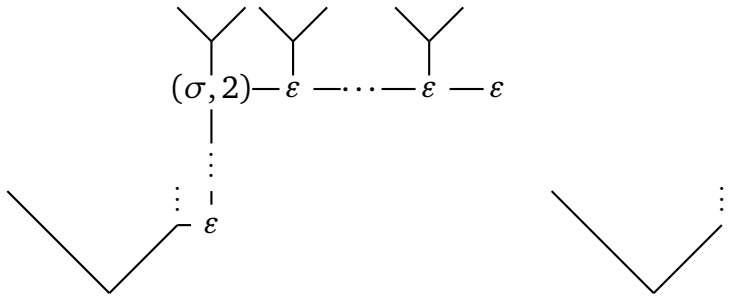


Figure 3.4.: collapse operation of level 2 (if the collapse is of level 1 then it is identical to the pop_1 operation).

3. There is some $j < n$ such that $\tau_1\tau_2 \leq w_i$ for all $i \leq j$ and $\tau_1\tau_2 \not\leq w_i$ for all $i > j$. In this case the claim of the lemma reduces to the claim that the lemma holds for $t := w_{j+1} : w_{j+2} : \cdots : w_n : w_{n+1}$ and $t' := w_{j+1} : w_{j+2} : \cdots : w_n$. Since the left subtrees of the encodings of s and s' agree and their right subtrees encode t and t', respectively, we can apply again this case distinction to t and t'.

4. The last case is that $|w_1| = 1$. If $n > 1$, the claim reduces to the claim that the lemma holds for $t := w_2 : w_3 : \cdots : w_n : w_{n+1}$ and $t' := w_2 : w_3 : \cdots : w_n$ because w_1 is encoded in the root of $\mathsf{Enc}(s, \tau)$ and $\mathsf{Enc}(s', \tau)$ and the right subtree of the trees encode t and t', respectively.

   If $n = 1$, this leads to the fact that $\mathsf{Enc}(s', \sigma)$ is only a tree of one element and $\mathsf{Enc}(s, \sigma)$ extends this root by a right subtree, namely $\mathsf{Enc}(w_{n+1}, \varepsilon)$. In this case the lemma holds due to our initial remark.

In each iteration of the case distinction, the stacks get smaller. Thus, we eventually reach the first case or the last case with condition $n = 1$. This observation completes the proof of the lemma. □

Analogously to the case of pop_2, one proves a similar result for the collapse operation:

Lemma 3.1.28. Let s, s' be stacks of a pushdown system $\mathscr{S}$ such that $\mathsf{CLvl}(s) = 2$ and $s' := \mathsf{collapse}(s)$. Let t' be the maximal element in the rightmost path of $\mathsf{Enc}(s, (\bot, 1))$ which is labelled by some $(\sigma, 2)$ for $\sigma \in \Sigma$. Furthermore, let t be the maximal ancestor of t' such that $t1 \leq t'$. For $D := \{d \in \mathsf{dom}(\mathsf{Enc}(s, (\bot, 1))) : t1 \not\leq d\}$, it holds that

$$\mathsf{Enc}(s', (\bot, 1)) = \mathsf{Enc}(s, (\bot, 1))\restriction_D \text{ (see Figure 3.4).}$$

Proof. Note that the rightmost leaf of $\mathsf{Enc}(s, (\bot, 1))$ is of the form $t'1^n$ for some $n \in \mathbb{N}$. Hence, the topmost element of s is a clone of the element encoded at t'. Thus,

$$\mathsf{collapse}(s) = \mathsf{pop}_2(\mathsf{LStck}(t', \mathsf{Enc}(s, (\bot, 1)))).$$

Using the previous lemma, the claim follows immediately. □

With these auxiliary lemmas we can now prove that Enc turns the relations of collapsible pushdown graphs into automatic relations.

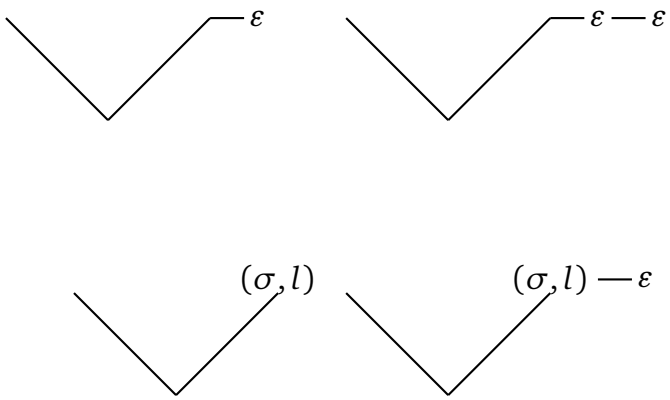


Figure 3.5.: The two versions of clone_2 operations.

Lemma 3.1.29. Let $\mathscr{S} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a collapsible pushdown system. For each $\delta \in \Delta$, there is an automaton $\mathscr{A}_\delta$ such that for all configurations c_1 and c_2

$$\mathscr{A}_\delta \text{ accepts } \mathsf{Enc}(c_1) \otimes \mathsf{Enc}(c_2) \quad \text{iff} \quad c_1 \vdash^\gamma c_2.$$

Proof. Consider a transition $\delta := (q, \sigma, \gamma, q', \mathsf{op})$. We show that there is an automaton that accepts $\mathsf{Enc}(c_1) \otimes \mathsf{Enc}(c_2)$ if and only if δ induces a transition from c_1 to c_2. Thus, we have to define an automaton that accepts $\mathsf{Enc}(c_1) \otimes \mathsf{Enc}(c_2)$ if and only if the following conditions are satisfied.

1. $c_1 = (q, s_1)$ for some stack s_1,
2. $c_2 = (q', s_2)$ for some stack s_2,
3. $\mathsf{Sym}(c_1) = \sigma$, and
4. $\mathsf{op}(s_1) = s_2$.

The states of c_1 and c_2 may be checked directly at the root of $\mathsf{Enc}(c_1) \otimes \mathsf{Enc}(c_2)$. $\mathsf{Sym}(c_1)$ is encoded in the last node of the rightmost path in $\mathsf{Enc}(c_1)$ that is not labelled ε. Hence, the remaining problem is to construct an automaton for each stack operation op which recognises $\mathsf{Enc}(s_1, (\bot, 1)) \otimes \mathsf{Enc}(s_2, (\bot, 1))$ if and only if $s_2 = \mathsf{op}(s_1)$.

We proceed by a case distinction on the stack operation.

- If $s_2 = \mathsf{push}_{\sigma,2}(s_1)$ or $s_2 = \mathsf{clone}_2(s_1)$, then $\mathsf{Enc}(s_1, (\bot, 1))$ and $\mathsf{Enc}(s_2, (\bot, 1))$ differ only in one node, which is the rightmost leaf of $\mathsf{Enc}(s_2, (\bot, 1))$ (cf. Figures 3.5 and 3.7). This can easily be checked by an automaton.
- If $s_2 = \mathsf{push}_{\sigma,1}(s_1)$, we have to distinguish two cases. In most cases, this operation behaves analogous to $\mathsf{push}_{\sigma,2}$ and $\mathsf{Enc}(s_2, (\bot, 1))$ is the extension of $\mathsf{Enc}(s_1, (\bot, 1))$ by a left successor of the rightmost leaf of $\mathsf{Enc}(s_1, (\bot, 1))$. This new node is labelled $(\sigma, 1)$.

But there is one case that is different, namely, when s_1 decomposes as

$$s_1 = s_1' : \big(w \setminus \big(\sigma w_1 : \sigma w_2 : \cdots : \sigma w_n : \varepsilon\big)\big)\,.$$

This case is depicted in Figure 3.6. In this case,

$$\mathsf{top}_1(w)\sigma w_1 : \mathsf{top}_1(w)\sigma w_2 : \cdots : \mathsf{top}_1(w)\sigma w_n$$

forms a block b of the stack s_1. $\mathsf{top}_1(w) \setminus \varepsilon$ forms another block which is encoded in the rightmost leaf of $\mathsf{Enc}(s_1, (\bot, 1))$. Now,

$$s_2 = s_1' : (w \setminus \sigma w_1 : \sigma w_2 : \cdots : \sigma w_n : \sigma)$$

i.e., in $\mathsf{Enc}(s_2, (\bot, 1))$ the whole block $\mathsf{top}_1(w) \setminus \sigma w_1 : \sigma w_2 : \cdots : \sigma w_n : \sigma$ is encoded in a single subtree. This subtree extends the subtree encoding the block b by exactly one ε-labelled node as depicted in Figure 3.6. Thus, $s_2 = \mathsf{push}_{\sigma,1}(s_1)$ if the following conditions are satisfied:

1. there is a node $d1 \in \mathsf{Enc}(s_1, (\bot, 1)) \otimes \mathsf{Enc}(s_2, (\bot, 1))$ such that $d1$ is the rightmost leaf of $\mathsf{Enc}(s_1, (\bot, 1))$,
2. $d1 \notin \mathsf{Enc}(s_2, (\bot, 1))$,
3. $\mathsf{Enc}(s_2, (\bot, 1))$ extends $\mathsf{Enc}(s_1, (\bot, 1))$ by one node of the form $d01^m$,
4. $d0$ is labelled by $(\sigma, 1)$ in $\mathsf{Enc}(s_1, (\bot, 1))$ and $\mathsf{Enc}(s_2, (\bot, 1))$, and
5. the two trees coincide on all nodes but $d1$ and $d01^m$.

These conditions are clearly MSO-definable whence there is an automaton recognising these pairs of trees.

Note that the case distinction is also MSO-definable. For $d1$ the rightmost leaf of $\mathsf{Enc}(s_1, (\bot, 1))$, the second case applies if and only if $d0$ has label $(\sigma, 1)$ in $\mathsf{Enc}(s_1, (\bot, 1))$. Again, the correspondence between MSO and automata yields an automaton that accepts $\mathsf{Enc}(s_1, (\bot, 1)) \otimes \mathsf{Enc}(s_2, (\bot, 1))$ if and only if $s_2 = \mathsf{push}_{\sigma,1}(s_1)$.

- Consider $s_2 = \mathsf{pop}_1(s_1)$. Since pop_1 is a kind of inverse of $\mathsf{push}_{\sigma,i}$, we make a similar case distinction as in that case.

  The different possibilities are depicted in the Figures 3.8 and 3.9. Note the similarity of Figure 3.8 and of Figure 3.6, as well as the similarity of Figure 3.9 and Figure 3.7.

  Both cases can be distinguished by an automaton. $\mathsf{Enc}(s_1, (\bot, 1))$ and $\mathsf{Enc}(s_2, (\bot, 1))$ are as in Figure 3.8 if and only if the rightmost leaf of $\mathsf{Enc}(s_1, (\bot, 1))$ is a right successor.

  Analogously to the push case, we conclude that there is an automaton that recognises $\mathsf{Enc}(s_1, (\bot, 1)) \otimes \mathsf{Enc}(s_2, (\bot, 1))$ if and only if $s_2 = \mathsf{pop}_1(s_1)$.

- For the case of pop_2, recall Lemma 3.1.27 and Figure 3.3. An automaton recognising the pop_2 operation only has to guess the set D from Lemma 3.1.27 and check whether the second tree is the restriction of the first tree to D. Note that the last element of D along the rightmost path may be guessed nondeterministically and then the automaton may check that its guess was right.

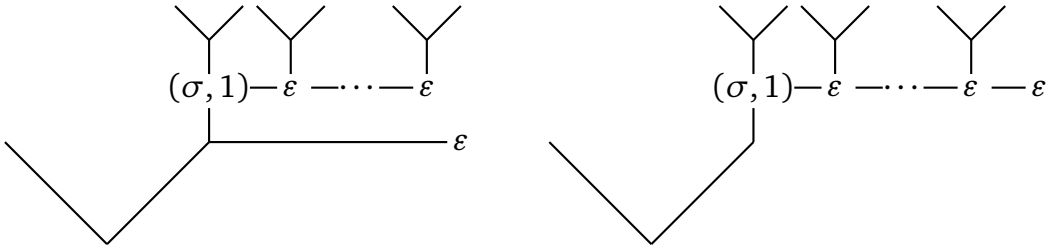


Figure 3.6.: $\mathsf{push}_{\sigma,1}$ operation with $\mathsf{top}_2(\mathsf{push}_{\sigma,1}(s_1)) \leq \mathsf{top}_2(\mathsf{pop}_2(s_1))$.

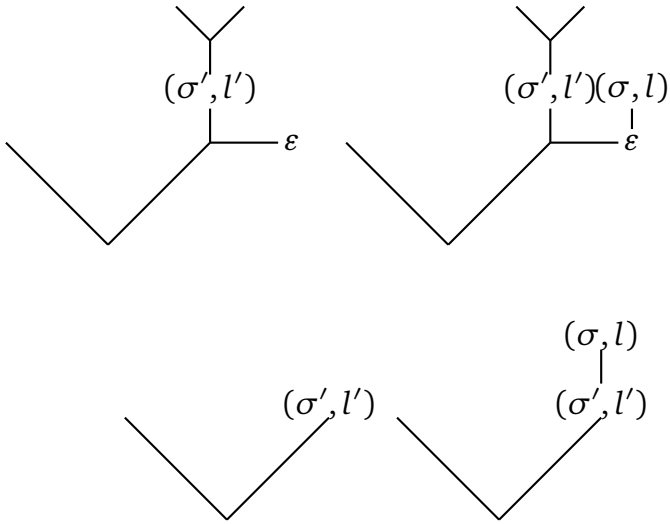


Figure 3.7.: The two versions of $\mathsf{push}_{\sigma,l}$ operation otherwise.

- For the case of $\mathsf{collapse}$, we have a case distinction due to the collapse level of the stack s_1. Either $\mathsf{CLvl}(s_1) = 1$ or $\mathsf{CLvl}(s_2) = 2$. If it is 1, the collapse operation on s_1 is equivalent to a pop_1 operation. Otherwise, the collapse level of s_1 is 2. This case can be treated as in the case of a pop_2, but using Lemma 3.1.28 instead of Lemma 3.1.27.

  Since the case distinction only depends on the collapse level stored in the label of the maximal node in the rightmost path of $\mathsf{Enc}(s_1, (\bot, 1))$ which is not labelled ε, an automaton may nondeterministically guess which case applies and verify its guess during the run on $\mathsf{Enc}(s_1, (\bot, 1)) \otimes \mathsf{Enc}(s_2, (\bot, 1))$. □

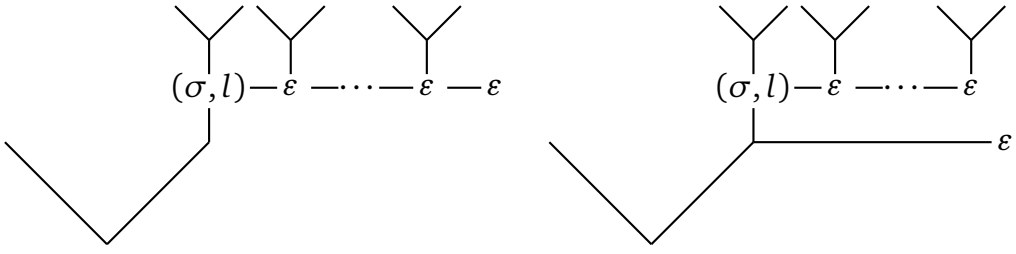


Figure 3.8.: pop_1 operation on a cloned element.

Figure 3.9.: pop_1 operation otherwise.

We have seen that for each collapsible pushdown system the class of encodings of valid configurations of this system is a set of regular trees. Furthermore, all operations of a collapsible pushdown system are automata-recognisable in this encoding. Putting these facts together we obtain the following theorem.

Theorem 3.1.30. Given a collapsible pushdown system $\mathscr{S}$ of level 2, one can effectively compute an automatic presentation of the collapsible pushdown graph generated by $\mathscr{S}$.

A direct corollary of this theorem is the decidability of the first-order model checking on collapsible pushdown graphs (cf. Theorem 3.0.4).

Corollary 3.1.31. The $\mathsf{FO}(\exists^\infty, \exists^{\mathsf{mod}}, (\mathsf{Ram}^n)_{n\in\mathbb{N}})$-theory of every level 2 collapsible pushdown graph is decidable.

3.1.4 Tree-Automaticity of Regular Reachability Predicates

In this section we show that regular reachability predicates are also automatic via Enc. In the first part, we expand a collapsible pushdown graph $\mathsf{CPG}(\mathscr{S})$ by the binary relation REACH (cf. Definition 2.1.12) and prove that this predicate is automatic in our encoding. In the second part, we use the closure of collapsible pushdown systems under products with automata in order to provide the automaticity of all regular reachability predicates.

In order to show the regularity of the reachability predicate, we start with an observation about the general form of a run between two configurations. Let $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$. For every run ρ from c_1 to c_2 there are configurations (q_3, s_3), (q_4, s_4), (q_5, s_5), positions $i_3 \leq i_4 \leq i_5 \in \mathsf{dom}(\rho)$, and numbers $m_2, m_3, m_5 \in \mathbb{N}$ such that the following holds:

1. $\rho(i_3) = (q_3, s_3)$ and $s_3 = \mathsf{pop}_2^{m_2}(s_1)$,
2. $\rho(i_4) = (q_4, s_4)$, $s_4 = \mathsf{pop}_1^{m_3}(s_3)$ and s_4 is a common substack of s_1 and s_2,
3. $\rho(i_5) = (q_5, s_5)$, $s_4 = \mathsf{pop}_1^{n_5}(s_5)$ and $s_5 = \mathsf{pop}_2^{m_5}(s_2)$, and
4. ρ does not visit any proper substack of s_4.

For any run ρ, s_4 is found as follows: it is the minimal substack of s_1 that is visited by ρ. i_4 is then an arbitrary position in ρ that visits s_4. The existence of i_5 follows directly from the fact that s_5 is a milestone of s_2 and the fact that ρ visits s_4, which is a substack of s_5. The existence of i_3 is clear from the fact that the run ρ has to reach a stack of width $|s_4|$ at first, before it can change the $|s_4|$-th word of the stack, i.e., before it can reach s_3.

We use this decomposition for proving the regularity of REACH as follows.

Definition 3.1.32. Given a collapsible pushdown system $\mathscr{S}$, we define the following four relations on the configurations of $\mathscr{S}$:

1. Let $A \subseteq \mathsf{CPG}(\mathscr{S}) \times \mathsf{CPG}(\mathscr{S})$ be the relation containing those pairs of configurations (c_1, c_2) with $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$ such that
   a) $s_2 = \mathsf{pop}_2^m(s_1)$,
   b) there is a run ρ from c_1 to c_2 and
   c) ρ does not visit a proper substack of s_2.
2. Let $B \subseteq \mathsf{CPG}(\mathscr{S}) \times \mathsf{CPG}(\mathscr{S})$ be the relation containing those pairs of configurations (c_1, c_2) with $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$ such that
   a) $s_2 = \mathsf{pop}_1^m(s_1)$,
   b) there is a run ρ from c_1 to c_2 and
   c) ρ does not visit a proper substack of s_2.
3. Let $C \subseteq \mathsf{CPG}(\mathscr{S}) \times \mathsf{CPG}(\mathscr{S})$ be the relation containing those pairs of configurations (c_1, c_2) with $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$ such that
   a) $s_1 = \mathsf{pop}_1^m(s_2)$,
   b) there is a run ρ from c_1 to c_2 and
   c) ρ does not visit a proper substack of s_1.
4. Let $D \subseteq \mathsf{CPG}(\mathscr{S}) \times \mathsf{CPG}(\mathscr{S})$ be the relation containing those pairs of configurations (c_1, c_2) with $c_1 = (q_1, s_1)$ and $c_2 = (q_2, s_2)$ such that
   a) $s_1 = \mathsf{pop}_2^m(s_2)$,
   b) there is a run ρ from c_1 to c_2 and
   c) ρ does not visit a substack of s_1 after its initial configuration.

Remark 3.1.33. Since we allow runs of length 0, the relations A, B, C and D are reflexive, i.e., for all configurations c, $(c,c) \in A$, $(c,c) \in B$, $(c,c) \in C$ and $(c,c) \in D$.

The relation REACH can be expressed via A, B, C and D in the sense that for arbitrary configurations c_1, c_2, $(c_1, c_2) \in \mathsf{REACH}$ holds if and only if there are configurations x, y, z such that $(c_1, x) \in A$, $(x, y) \in B$, $(y, z) \in C$ and $(z, c_2) \in D$. Since projections of regular sets are regular, REACH is an automatic relation via the encoding Enc if the relations A, B, C and D are automatic via Enc. Proving the regularity of these relations is our next goal. We first prove the regularity of A. This proof requires an analysis of runs from some stack s to some stack $\mathsf{pop}_2^n(s)$ for every $n \in \mathbb{N}$. We obtain a characterisation of these runs that can be checked by an automaton.

Regularity of the Relation A

At a first glance one might think that a run from some stack s to a stack $\mathsf{pop}_2^n(s)$ only consists of a sequence of returns. But this is only true if we do not use the collapse operation. A collapsible pushdown system may start by writing a lot of information with clone_2 and $\mathsf{push}_{\sigma,l}$ operations onto the stack, then use a couple of pop_1 operations to come to an element with a small collapse link and finally use the collapse to jump to a very small substack of s

without using any other substack of s in between. Such a run does not contain any returns at all.

In order to cope with such runs, we introduce the notion of a level-1-loop. A level-1-loop is a kind of loop of the topmost word which increases the number of words on the level 2 stack. We prove that the pairs of initial and final states of these new loops are computable in a similar way as for ordinary loops. Furthermore, we show that every run from s to $\mathsf{pop}_2^n(s)$ decomposes mainly into parts that are basically returns, loops or 1-loops. These parts are connected by application of either a pop_1 or a $\mathsf{collapse}$ operation. First, we introduce 1-loops. Then we show the decomposition result we mentioned above. Finally, we use this decomposition for showing the regularity of the relation A. For this purpose, we introduce certificates for substack reachability. We consider a certificate as the abstract representation of the decomposition of some (potentially existing) run. The certificate consists of the final state of each part of the decomposition of this run. Using these certificates, we reduce the problem whether a run exists to the problem whether the subruns that form the parts of the decomposition exist. This is a much simpler problem because each of these subruns can only have a very special form. Finally, we show that an automaton can check the existence of these subruns while processing the certificate and the trees encoding the initial and final configuration of the run.

Definition 3.1.34. Let s be some stack and w some word. A run λ of length n is called a level-1-loop (or 1-loop) of $s : w$ if the following conditions are satisfied.

1. $\lambda(0) = (q_0, s : w)$ for some $q_0 \in Q$,
2. $\lambda(n) = (q_n, s : s' : w)$ for some nonempty stack s' and some state $q_n \in Q$,
3. for every $i \in \mathsf{dom}(\lambda)$, $|\lambda(i)| > |s|$, and
4. for every $i \in \mathsf{dom}(\lambda)$ such that $w \leq \mathsf{top}_2(\lambda(i-1))$ and $\mathsf{top}_2(\lambda(i)) = \mathsf{pop}_1(w)$, there is some $j > i$ such that $\lambda{\restriction}_{[i,j]}$ is a return.

Remark 3.1.35. Under condition 2, condition 3 is equivalent to the condition that λ never passes the stack s. An example of a 1-loop can be found in Figure 3.10. Note that the last two conditions imply that a 1-loop does never visit a proper substack of $s : w$.

Definition 3.1.36. For a fixed collapsible pushdown system $\mathcal{S}$ and some stack s we denote by $\exists 1\text{-}\mathsf{Loops}_{\mathcal{S}}(s)$ the set

$$\{(q_1, q_2) \in Q \times Q : \text{ there is an } s' \in \mathsf{Stacks}(\Sigma) \text{ and a 1-loop of } \mathcal{S} \text{ from } (q_1, s) \text{ to } (q_2, s')\}.$$

If $\mathcal{S}$ is clear from the context, we omit it.

We use this rather technical definition of a 1-loop due to two important properties. Firstly, we obtain a similar computational behaviour of 1-loops as for loops and returns: $\exists 1\text{-}\mathsf{Loops}(s)$ only depends on the returns of $\mathsf{pop}_1(s)$, $\mathsf{CLvl}(s)$ and $\mathsf{Sym}(s)$. Secondly, this notion is strong enough to capture all parts of a run from a stack s to $\mathsf{pop}_2^n(s)$ that are not captured by the notions of loops and returns. This idea is made precise in Lemma 3.1.39.

Lemma 3.1.37. There is an algorithm that determines for every stack s the set $\exists 1\text{-}\mathsf{Loops}(s)$ from the input $\mathsf{Sym}(s)$, $\mathsf{CLvl}(s)$ and $\exists \mathsf{Returns}(\mathsf{top}_2(\mathsf{pop}_1(s)))$.

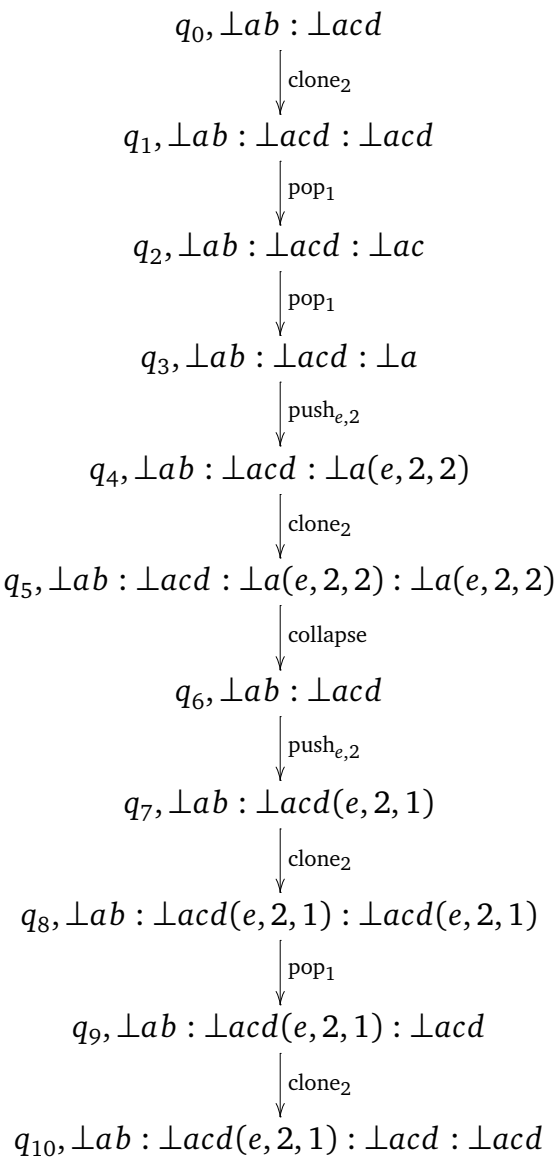


Figure 3.10.: Example of a 1-loop of $s := \bot ab : \bot acd$. The part between q_2 and q_6 forms a return of a stack with topmost word $\mathsf{top}_2(\mathsf{pop}_1(s))$. Note that the run up to q_9 also forms a 1-loop.

Proof (sketch). First of all note the similarity of the claim to the corresponding lemmas concerning returns, loops, low loops and high loops. The main ingredients of this proof are variants of Lemma 2.3.28 and Lemma 2.4.43.

- Analogously to Lemma 2.3.28, it is decidable whether there is some reachable configuration of the form $c = (q, s')$ with $|s'| \geq 3$ and $|\mathsf{top}_2(s')| = 3$. Note that by definition of the return-simulator $\mathsf{Rt}^k_s(\mathscr{S})$, $|\mathsf{top}_2(s')| = 3$ is equivalent to $\mathsf{top}_2(s') = \bot\top(\mathsf{Sym}(s), \mathsf{CLvl}(s), \kappa(\mathsf{CLvl}(s)))$ for all configurations of some return simulator for all stacks s and s'. The decidability follows by reduction to $L\mu$ model checking. We equip the pushdown system with a testing device. This testing devices first tries to perform two pop_1 operations. If it then reaches the bottom of stack, it tries to perform two pop_2-operations. If this is possible, then the stack is of the desired form.

- Analogously to the return case 2.4.43, one proves that $(q_1, q_2) \in \exists 1\text{-}\mathsf{Loops}(s)$ if and only if the graph of the return simulator $\mathsf{Rt}^k_s(\mathscr{S})$ contains some run from $(q_1, \bot\top\mathsf{top}_1(s)\Box : \bot\top\mathsf{top}_1(s))$ to (q_2, s') where $|s'| \geq 3$ and $\mathsf{top}_2(s') = \bot\top\mathsf{top}_1(s)$. Analogously to the return case, each such run corresponds to a 1-loop starting in (q_1, s) and ending in state q_2. Again, we copy the transitions of such a simulation one to one to a run starting in (q_1, s). Whenever we come to a transition on topmost symbol $\top$, we replace the following pop_2-transition by a return of some stack with topmost word $\mathsf{pop}_1(\mathsf{top}_2(s))$.

Putting these two facts together, we obtain that $\exists 1\text{-}\mathsf{Loops}(s)$ can be computed from $\mathsf{Rt}^k_s(\mathscr{S})$, $\mathsf{Sym}(s)$ and $\mathsf{CLvl}(s)$. But the definition of the return simulator only depends on $\exists\mathsf{Returns}(\mathsf{pop}_1(s))$. This concludes the proof. □

Of course, we can turn the previous proof into a definition of an automaton calculating the 1-loops of all milestones of a stack. This is completely analogous to Propositions 2.4.19 and 2.4.47.

Corollary 3.1.38. For each collapsible pushdown system $\mathscr{S}$ of level 2, we can compute an automaton $\mathscr{A}$ that calculates for each configuration c at each $d \in \mathsf{Enc}(c)$ the set $\exists 1\text{-}\mathsf{Loops}_{\mathscr{S}}(\mathsf{LStck}(d, \mathsf{Enc}(c)))$.

Now, we analyse the form of any run from some stack s to some substack $s' = \mathsf{pop}^n_2(s)$.

Lemma 3.1.39. Let s and s' be stacks such that $s' = \mathsf{pop}^m_2(s)$ for some $m \in \mathbb{N}$. Let ρ be a run from s to s' such that ρ does not visit a proper substack of s'. Then ρ decomposes as $\rho_1 \circ \rho_2 \circ \cdots \circ \rho_n \circ \lambda$ where λ is a high loop of s' and each ρ_i is of one of the following forms.

F1. ρ_i is a return,

F2. ρ_i is a 1-loop followed by a collapse of collapse level 2,

F3. ρ_i is a loop followed by a collapse of collapse level 2,

F4. ρ_i is a loop followed by a pop_1 (or a collapse operation of collapse level 1) and there is a $j > i$ such that ρ_j is of the form F2 or F3 and there is no $i < k < j$ such that ρ_k is of the form F1,

F5. ρ_i is a 1-loop followed by a pop_1 operation (or a collapse of collapse level 1) and there is a $j > i$ such that ρ_j is of the form F2 or F3 and there is no $i < k < j$ such that ρ_k is of the form F1.

Proof. Let $s' = \mathsf{pop}_2^n(s)$ for $n \geq 0$ and ρ a run from s to s' not passing any proper substack of s'.

First of all, note that the case $n = 0$ is trivial. If $n = 0$, ρ is by definition a high loop of s.

For the case $n > 0$, we proceed by induction on the length of ρ. We write (q_i, s_i) for the configuration $\rho(i)$. Firstly, consider the case where there is some $m \in \mathsf{dom}(\rho)$ such that $\rho_1 := \rho\restriction_{[0,m]}$ is a return. Then ρ_1 is of the form F1. By induction hypothesis, $\rho\restriction_{[m,\mathsf{ln}(\rho)]}$ decomposes as desired.

Otherwise, assume that there is no $m \in \mathsf{dom}(\rho)$ such that $\rho\restriction_{[0,m]}$ is a return.

Nevertheless, there is a minimal $m \in \mathsf{dom}(\rho)$ such that for all $i < m$, it holds that $|s_i| \geq |s|$ and $|s_m| < |s|$. The last operation of $\hat{\rho} := \rho\restriction_{[0,m]}$ is a collapse such that $\mathsf{top}_2(s_{m-1}) \leq \mathsf{top}_2(s)$ (otherwise $\hat{\rho}$ would be a return).

Writing $w := \mathsf{top}_2(s_{m-1})$, we distinguish two cases.

1. First consider the case that $w = \mathsf{top}_2(s)$. Note that this implies $\mathsf{CLvl}(s) = 2$ because the last operation of $\hat{\rho}$ is a collapse of level 2.

   Furthermore, we claim that $\hat{\rho}$ does not visit $\mathsf{pop}_1(s)$. Heading for a contradiction, assume that $\hat{\rho}(i) = \mathsf{pop}_1(s)$ for some $i \in \mathsf{dom}(\hat{\rho})$. Since $\hat{\rho}$ does not visit $\mathsf{pop}_2(s)$ between i and $m-1$, $\mathsf{top}_2(\hat{\rho}(m-1)) = w$ is only possible if $\mathsf{CLnk}(w) = |s|-1$ (smaller links cannot be restored by $\hat{\rho}$). But then $\hat{\rho}\restriction_{[i,m]}$ is a return of $\mathsf{pop}_1(s)$ whence $\hat{\rho}$ is a return of s. This contradicts the assumption that $\hat{\rho}$ is no return.

   Hence, $\hat{\rho}$ does not pass $\mathsf{pop}_1(s)$ and we distinguish the following cases

   - Assume that the stack of $\hat{\rho}(m-1)$ is s. Then $\hat{\rho}$ is a high loop followed by a collapse: the stack at $\hat{\rho}(0)$ and $\hat{\rho}(m-1)$ is s and the run does not visit $\mathsf{pop}_2(s)$ or $\mathsf{pop}_1(s)$ in between whence its restriction to $[0, m-1]$ is a high loop. Thus, $\rho_1 := \hat{\rho}$ is of the form F3 and the claim follows by induction hypothesis.
   - Assume that the stack of $\hat{\rho}(m-1)$ is $s' = s : t : w$ for some nonempty 2-word t. We claim that $\hat{\rho}$ is a 1-loop plus a collapse operation: We have already seen that $\hat{\rho}$ does not visit any proper substack of s. Thus, it suffices to show that $\hat{\rho}$ reaches a stack with topmost word $\mathsf{pop}_1(w)$ only at positions where a return starts.

     Let i be some position such that $w \leq \hat{\rho}(i-1)$ and $\mathsf{top}_2(\hat{\rho}(i)) = \mathsf{pop}_1(w)$. Recall that $\mathsf{top}_2(s_{m-1}) = w$, $\mathsf{CLnk}(w) = 2$ and $\mathsf{CLvl}(w) \leq |s| - 1$. Since $|\hat{\rho}(i)| > |s|$, we cannot restore $\mathsf{top}_1(w)$ by a push operation. Thus, there is some minimal position $j > i$ such that $|\hat{\rho}(j)| < |\hat{\rho}(i)|$. Since the level 2 links of w point below $\mathsf{pop}_2(s)$ and no proper substack of s is reached by $\hat{\rho}\restriction_{[0,m-1]}$, the links stored in w are not used in $\hat{\rho}\restriction_{[i,j]}$. It follows immediately that $\hat{\rho}\restriction_{[i,j]}$ is a return.

     Thus, $\rho_1 := \hat{\rho}$ is of the form F2.

2. For the other case, assume that $w < \mathsf{top}_2(s)$. Then there is a minimal $i \in \mathsf{dom}(\hat{\rho})$ such that $\mathsf{top}_2(\hat{\rho}(i)) = \mathsf{pop}_1(w)$ and there is no $j > i$ such that $\hat{\rho}\restriction_{[i,j]}$ is a return.

   We claim the following: if $\hat{\rho}(i) = \mathsf{pop}_1(s)$, then $\hat{\rho}_1$ is of the form F4, otherwise $\hat{\rho}$ is of the form F5. Due to the definition, $\rho_1 := \hat{\rho}\restriction_{[0,i]}$ is a loop or 1-loop followed by a pop_1 or a collapse. Hence, it suffices to check the side conditions on the segments following in the decomposition of ρ. For this purpose set $\rho' := \rho\restriction_{[i,\mathsf{ln}(\rho)]}$. By induction hypothesis ρ' decomposes as $\rho' = \rho_2 \circ \rho_3 \circ \cdots \circ \rho_n \circ \lambda$ where the ρ_i and λ satisfy the claim of the lemma.

$$
\begin{array}{rccccc}
 & & & & a & a \\
 & & c & b & (a,2,3) & (a,2,3) \\
 & & (c,2,1) & (c,2,1) & (c,2,1) & (c,2,1) \\
s := & \bot & \bot & \bot & \bot & \bot
\end{array}
$$

Figure 3.11.: The stack s of example 3.1.40.

Now, by definition of i, ρ' does not start with a return. Thus, ρ_2 is of one of the forms F2–F5. But all these forms require that there is some $j \geq 2$ such that ρ_j is of form F2 or F3 and for all $2 \leq k < j$, ρ_k is not of the form F1.

From this condition, it follows directly that $\rho = \rho_1 \circ \rho' = \rho_1 \circ \rho_2 \circ \rho_3 \circ \cdots \circ \rho_n \circ \lambda$ and ρ_1 is of the form F4 or F5. □

The following example illustrates the lemma.

Example 3.1.40. Consider the stack s in Figure 3.11 and the following transitions:

1. $(q_1, a, q_2, \mathsf{clone}_2)$,
2. $(q_2, a, q_3, \mathsf{collapse})$,
3. $(q_3, a, q_2, \mathsf{clone}_2)$,
4. $(q_3, b, q_2, \mathsf{push}_{b,2})$,
5. $(q_1, c, q_4, \mathsf{push}_{b,1})$,
6. $(q_2, b, q_1, \mathsf{collapse})$,
7. $(q_2, c, q_1, \mathsf{collapse})$,
8. $(q_4, b, q_2, \mathsf{pop}_1)$.

Since these transitions form a deterministic relation, there is a unique run starting in (q_1, s). This run ρ is generated by using the transitions in the following order: (1), (2), (3), (2), (4), (6), (5), (8), (7), (5), (8), (7). The run ρ ends in the configuration $(q_1, \bot_2) = (q_1, \mathsf{pop}_2^4(s))$. According to the decomposition of Lemma 3.1.39, $\rho \restriction_{[0,2]}$ is of the form F5, $\rho \restriction_{[2,4]}$ is of the form F2, $\rho \restriction_{[4,6]}$ is of the form F1, $\rho \restriction_{[6,9]}$ is of the form F4, and $\rho \restriction_{[9,12]}$ is of the form F3.

The previous lemma tells us that any run to a substack decomposes into subruns of three forms:

1. returns,
2. subruns that decrease the length of the topmost word by one, or
3. subruns that end in a collapse of level 2 applied to some stack with the same topmost word as their initial stack.

If subruns of the second case occur, then they are followed by a subrun of the third form before any return occurs. Since runs of the third form end in a collapse of level 2, it does not matter whether runs of the second or third form have increased the width of the stack in

between: eventually we perform a collapse operation on a prefix of the initial topmost word. This collapse then deletes all the new words that were created in between.

The decomposition of a run according to Lemma 3.1.39 is the starting point for deciding whether there is a run from some configuration (q,s) to some $(q',\mathrm{pop}_2^n(s))$. The basic idea is that we guess the form and the final state of each segment the run consists of. We then attach this guess to the encoding of the two configurations. We will call such a guess certificate for substack reachability. Finally, we prove that there is an automaton that can check whether a certificate for substack reachability actually encodes some run from (q,s) to $(q',\mathrm{pop}_2^n(s))$.

This approach is quite similar to the proof that the reachable configurations of a given collapsible pushdown system form a regular set. Let us first recall the basic idea of that proof. We used each node of $d \in \mathrm{Enc}(q,s)$ as representative for the milestone $\mathrm{LStck}(d,\mathrm{Enc}(q,s))$ and a certificate for reachability labelled every node with the state in which some run visited the corresponding milestone.

Now, we do a similar thing. Given a run ρ from (q_1,s) to $(q_2,\mathrm{pop}_2^m(s))$, let $\rho = \rho_1 \circ \rho_2 \circ \cdots \circ \rho_n \circ \lambda$ be its decomposition according to Lemma 3.1.39. We want to find a representative for the initial configuration of each of the ρ_j and label this representative with a description of ρ_j. In fact, we label the representative with the final state of ρ_j and the type of ρ_j according to the classification from Lemma 3.1.39.

Let us first explain the system of representation. Let $d \in \mathrm{Enc}(q,s)$ be some node. We write $s_d := \mathrm{LStck}(d,\mathrm{Enc}(q,s))$ for the milestone induced by d. Now, we will use d as a representative for any stack $\hat{s}_d$ that has the following two properties:

1. $\mathrm{pop}_2(s_d) = \mathrm{pop}_2^k(\hat{s}_d)$ for some $k \in \mathbb{N}$ and
2. $\mathrm{top}_2(s_d) = \mathrm{top}_2(\hat{s}_d)$.

This implies that d may represent s_d or some stack $\mathrm{pop}_2(s_d) : s' : \mathrm{top}_2(s_d)$ for s' an arbitrary $\mathrm{top}_2(s_d)$ prefixed stack.

Let us explain why this form of representation is sufficient for our purpose. Recall that the existence of 1-loops and loops only depends on the topmost word of a stack. Thus, we only need to know the topmost word of some stack in order to verify the existence of 1-loops or loops for certain pairs of initial and final states. Furthermore, if we know the topmost word of some stack, we can easily derive the topmost word of the stack reached via pop_1 or collapse of level 1. Moreover, if d is a representative for some stack $\hat{s}_d$, then a collapse of level 2 from $\hat{s}_d$ and from s_d result in the same stack: if d represents $\hat{s}_d$ then $\mathrm{top}_2(s_d)$ and $\mathrm{top}_2(\hat{s}_d)$ coincide. Thus, level 2 collapse links in the topmost word of $\hat{s}_d$ point to some substack of $\mathrm{pop}_2(s_d)$ (because the links of s_d have this property by definition of a stack). Since $\mathrm{pop}_2(s_d) = \mathrm{pop}_2^k(\hat{s}_d)$, the collapse link of s_d and of $\hat{s}_d$ point to the same substack of $\mathrm{pop}_2(s_d)$.

Hence, the representatives that we use are sufficiently similar to the represented stacks in the following sense. The existence of subruns of the forms F2–F5 can be decided by considering the representatives. Note that subruns of the form F1, which are returns, occur as an initial part of the run or after the application of some collapse of level 2. At such positions, the corresponding node d represents a stack $\hat{s}_d$ such that $\hat{s}_d = s_d$. Thus, $\mathrm{pop}_2(\hat{s}_d) = \mathrm{pop}_2(s_d)$ is determined by d. Hence, we can find a node d' such that $s_{d'} = \hat{s}_{d'} = \mathrm{pop}_2(\hat{s}_d)$.

Having explained the system of representation, let us introduce certificates for substack reachability. Before we come to the formal definition, we explain the underlying idea.

Given a tree $\mathsf{Enc}(q_1, s_1) \otimes \mathsf{Enc}(q_2, s_2)$ such that $s_2 = \mathsf{pop}_2^n(s_1)$, we want to label this tree with information witnessing the existence of a run from (q_1, s_1) to (q_2, s_2). Assume that there is such a run ρ. Let $\rho = \rho_1 \circ \rho_2 \circ \ldots \rho_n \circ \lambda$ be its decomposition into parts according to Lemma 3.1.39. Recall that the rightmost leaf d of $\mathsf{Enc}(q_1, s_1)$ represents the stack s_1. Since ρ_1 starts with stack s_1, this d is the position in $\mathsf{Enc}(q_1, s_1) \otimes \mathsf{Enc}(q_2, s_2)$ which we want to label with information concerning ρ_1. We will label this node with the final state of ρ_1 and the type of this run according to the classification from Lemma 3.1.39. If ρ_1 is of the form F1 (i.e., ρ_1 is a return), we label it by 1, if it is of the form F2, we label it by 2, etc.

Now, assume that there is some node d that represents some stack s' such that s' is the initial stack of ρ_j for some $1 \leq j \leq n$. The type of ρ_j defines a representative for the initial stack of ρ_{j+1}, which is the final stack of ρ_j, as follows.

1. If ρ_j is a return, the initial stack of ρ_{j+1} is $\mathsf{pop}_2(s_d)$. There is a node d' such that $s_{d'} = \mathsf{pop}_2(s_d)$. This node is the representative of ρ_{j+1}.

2. If ρ_j ends in a collapse of level 2 (from a stack with topmost word $\mathsf{top}_2(s_d)$) the initial stack of ρ_{j+1} is $\mathsf{collapse}(s_d)$. There is a node d' such that $s_{d'} = \mathsf{collapse}(s_d)$.

3. Finally, if ρ_j ends in a pop_1 or a $\mathsf{collapse}$ of level 1, we need to find a representative d' such that $\mathsf{top}_2(s_{d'}) = \mathsf{top}_2(\mathsf{pop}_1(s_d))$. We take the lexicographically maximal node d' such that $\mathsf{LStck}(d', \mathsf{Enc}(q_1, s_1))$ is a milestone of s_d with topmost word $\mathsf{top}_2(\mathsf{pop}_1(s_d))$.

We call the representative d' of the initial stack of ρ_{j+1} the successor of d. Keep in mind that this successor depends on the label of d'. Furthermore, note that the successor is MSO-definable on $\mathsf{Enc}(q_1, s_1)$ if the label of d is known: the pop_2 or $\mathsf{collapse}$ successor of s_d is clearly definable due to the regularity of the operations pop_2 and $\mathsf{collapse}$. For the third case, note that the successor of d is the unique ancestor of d such that $d = d'01^m$ for some $m \in \mathbb{N}$.

Since we have found a representative for ρ_{j+1}, we label it again by the final state of ρ_{j+1} and by the type of ρ_{j+1}. We continue this process until we have defined a representative for each segment of the run ρ. We will soon see that an automaton can check whether an arbitrary labelling of the nodes of $\mathsf{Enc}(q_1, s_1) \otimes \mathsf{Enc}(q_2, s_2)$ is indeed a labelling corresponding to an existing run from (q_1, s_1) to (q_2, s_2) in this sense.

Let us now formally introduce certificates for substack reachability. We will call such a certificate valid if it witnesses the existence of a run from the larger configuration to the smaller one.

Definition 3.1.41. Let $c_1 = (q_1, s_1), c_2 = (q_2, s_2)$ be configurations such that $s_2 = \mathsf{pop}_2^n(s_1)$ for some $n \in \mathbb{N}$. We call a function

$$f_{\mathsf{CSR}} : \mathsf{dom}(\mathsf{Enc}(c_1)) \setminus \mathsf{dom}(\mathsf{Enc}(c_2)) \to \{1, 2, 3, 4, 5\} \times Q$$

a certificate for substack reachability for c_1 and c_2.

Remark 3.1.42. Due to the finite range of a certificate for substack reachability, we can express quantification over certificates for substack reachability for c_1 and c_2 on the structure $\mathsf{Enc}(c_1) \otimes \mathsf{Enc}(c_2)$ in MSO.

Even though these certificates are defined on domain $\mathrm{dom}(\mathrm{Enc}(c_1)) \setminus \mathrm{dom}(\mathrm{Enc}(c_2))$, we will only use some of the information, namely those labels assigned to nodes that represent one of the stacks we pass on some run from c_1 to c_2. The first component represents a guess on the kind of segment starting at the corresponding stack. The numbers correspond to the enumeration in Lemma 3.1.39. The second component asserts the final state of the corresponding segment.

As already mentioned, for each encoding of two configurations and each certificate for substack reachability on this encoding, there is a successor function. This successor function chooses, according to the label of one representative, the representative for the next stack.

Definition 3.1.43. Let c_1, c_2 be configurations such that $c_2 = \mathrm{pop}_2^n(c_1)$ for some $n \in \mathbb{N}$. Furthermore, let f be a partial function from $\mathrm{dom}(\mathrm{Enc}(c_1))$ to $\{1,2,3,4,5\} \times Q$. For $d \in \mathrm{dom}(f)$, the successor of d with respect to f is defined by case distinction on the first component of $f(d)$, denoted by $\pi_1(f(d))$, as follows.

1. $\pi_1(f(d)) = 1$: If $d \in \{0\}^*$, there is no successor of d with respect to f.

   Otherwise, let d' be the ancestor of d such that $d = d'10^m$ for some number $m \in \mathbb{N}$. Let $d'' \in \{\varepsilon\} \cup \{0\{0,1\}^*\}$ be the lexicographically maximal word such that $d'd'' \in \mathrm{dom}(\mathrm{Enc}(c_1))$ ($d'd''$ is the maximal element in the subtree rooted at $d'0$ if $d'0$ is in the tree, otherwise we have $d'd'' = d'$). We say $d'd''$ is the successor of d with respect to f.

2. $\pi_1(f(d)) \in \{2,3\}$: If $d \in \{0\}^*\{1\}^*$ the successor of d with respect to f is undefined.

   Otherwise, let d' be the element such that $d = d'10^m1^n$ where $m > 0$ and $n \in \mathbb{N}$. Let $d'' \in \{\varepsilon\} \cup \{0\{0,1\}^*\}$ be the lexicographically maximal word such that $d'd'' \in \mathrm{dom}(\mathrm{Enc}(c_1))$. We say $d'd''$ is the successor of d with respect to f.

3. $\pi_1(f(d)) \in \{4,5\}$: If $d \in \{\varepsilon\} \cup \{0\}\{1\}^*$, then the successor of d with respect to f_{CSR} is undefined.

   Otherwise, let d' be the unique element such that $d = d'01^n$ for some $n \in \mathbb{N}$. Then d' is the successor of d with respect to f.

Remark 3.1.44. As already said in the informal description, the motivation of the previous definition are the following observations.

1. If $\pi_1(f(d)) = 1$, then the successor $\hat{d}$ of d is chosen such that

   $$\mathrm{LStck}(\hat{d}, \mathrm{Enc}(c_1)) = \mathrm{pop}_2(\mathrm{LStck}(d, \mathrm{Enc}(c_1))).$$

   If this is not possible, i.e., if $|\mathrm{LStck}(d, \mathrm{Enc}(c_1))| = 1$, the successor is undefined.

2. If $\pi_1(f(d)) \in \{2,3\}$, then the successor $\hat{d}$ is chosen such that

   $$\mathrm{LStck}(\hat{d}, \mathrm{Enc}(c_1)) = \mathrm{collapse}(\mathrm{LStck}(d), \mathrm{Enc}(c_1))$$

   (assuming that $\mathrm{CLvl}(\mathrm{LStck}(d, \mathrm{Enc}(c_1)))$ is 2). If such an element does not exist, then the successor is undefined.

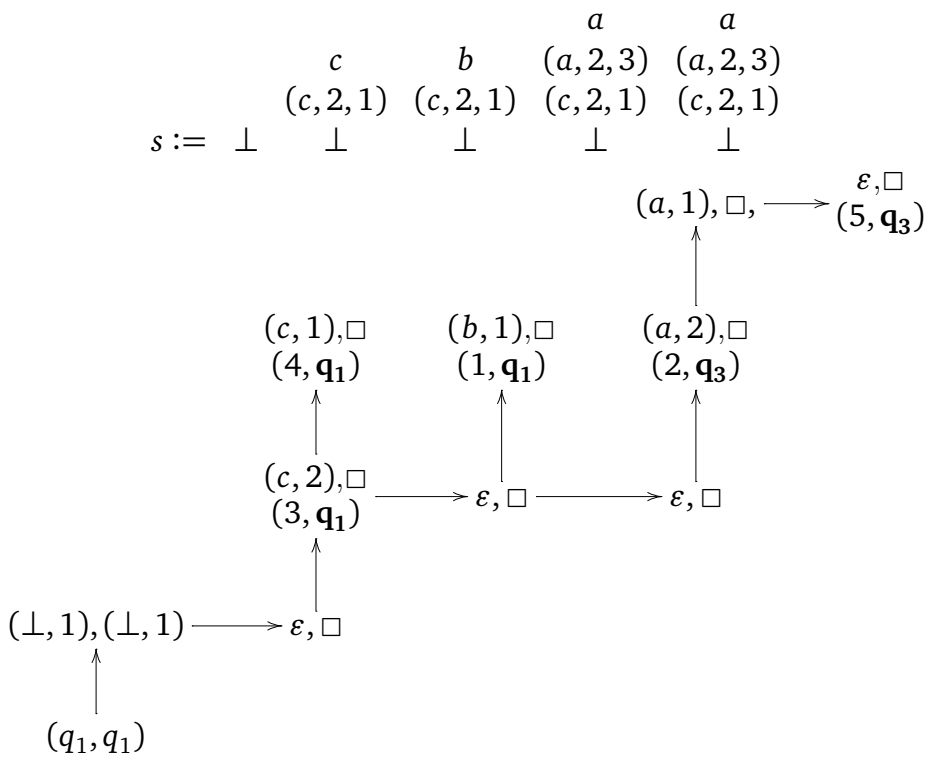


Figure 3.12.: Example of a valid certificate for substack reachability for $\mathrm{Enc}(q_1, s) \otimes \mathrm{Enc}(q_1, \bot_2)$.

3. If $\pi_1(f(d)) \in \{4, 5\}$, then the successor $\hat{d}$ of d is chosen such that $\mathrm{LStck}(\hat{d}, \mathrm{Enc}(c_1))$ is the maximal milestone of $\mathrm{LStck}(d, \mathrm{Enc}(c_1))$ satisfying

$$\mathrm{top}_2(\mathrm{LStck}(\hat{d}, \mathrm{Enc}(c_1))) = \mathrm{top}_2(\mathrm{pop}_1(\mathrm{LStck}(d, \mathrm{Enc}(c_1)))).$$

If this is not possible, i.e., if $\mathrm{top}_2(\mathrm{LStck}(d, \mathrm{Enc}(c_1))) = \bot$, then the successor is undefined.

Example 3.1.45. Recall the run ρ from example 3.1.40.

The decomposition of ρ induces a certificate for substack reachability on

$$\mathrm{Enc}(q_1, s) \otimes \mathrm{Enc}(q_1, \bot_2).$$

This certificate is depicted in Figure 3.12 (the bold labels are the values of the certificate). We only state the values of the certificate on the rightmost leaf of $\mathrm{Enc}(q_1, s)$ and the chain of successors with respect to this certificate. These are the important values that witness the existence of ρ.

We will show that there is a close connection between runs from some configuration (q, s) to another configuration $(\hat{q}, \hat{s})$ where $\hat{s} = \mathrm{pop}_2^n(s)$ and certificates for substack reachability. We prove that every such run induces a certificate with certain properties. Since these properties are rather technical, we postpone the detailed description of these properties for a short while. In the following, we first explain how to obtain such a certificate from the run. Then we present the characterising properties of these certificates. Finally, we show

that each certificate with these properties actually represents a run from (q,s) to $(\hat{q},\hat{s})$. Hence, deciding the existence of such a run reduces to deciding whether there is such a certificate. We then show that the latter problem is MSO-definable on the encoding of the configurations. From this, the regularity of the relation A follows.

Lemma 3.1.46. Let $c = (q,s)$ and $\hat{c} = (\hat{q},\hat{s})$ be configurations such that $\hat{s} = \mathrm{pop}_2^m(s)$ for some $m \in \mathbb{N}$. Let ρ be a run from c to $\hat{c}$ such that ρ does not pass a proper substack of $\hat{s}$. Assume that ρ decomposes as $\rho = \rho_1 \circ \rho_2 \circ \cdots \circ \rho_n \circ \lambda$ according to Lemma 3.1.39. Then there is a certificate for substack reachability f^{ρ}_{CSR} on $\mathrm{Enc}(c) \otimes \mathrm{Enc}(\hat{c})$ such that the following conditions hold.

There is a finite sequence $\bar{t} = t_1, t_2, \ldots, t_n \in \mathrm{dom}(\mathrm{Enc}(c)) \setminus \mathrm{dom}(\mathrm{Enc}(\hat{c}))$ such that

1. t_1 is the rightmost leaf of $\mathrm{Enc}(c) \otimes \mathrm{Enc}(\hat{c})$,
2. for each $1 \leq i \leq n$, $f^{\rho}_{\mathrm{CSR}}(t_i) = (k_i, q_i)$ where k_i is the form of ρ_i according to Lemma 3.1.39 and q_i is the final state of ρ_i.
3. t_{i+1} is the successor of t_i with respect to f^{ρ}_{CSR} for every $1 \leq i < n$, and
4. the successor of t_n with respect to f^{ρ}_{CSR} is the rightmost leaf of $\mathrm{Enc}(\hat{c})$.

Proof. First of all, we define inductively the sequence $t_1, t_2, \ldots, t_n$ and the values of f^{ρ}_{CSR} on these elements.

We write k_i for the form of ρ_i and q_i for the final state of ρ_i. Let t_1 be the rightmost leaf of $\mathrm{Enc}(q,s)$. We define $f^{\rho}_{\mathrm{CSR}}(t_1) := (k_1, q_1)$. Now assume that we have already defined t_i for some $1 \leq i < n$. We define $f^{\rho}_{\mathrm{CSR}}(t_i) := (k_i, q_i)$. If the successor of t_i with respect to f^{ρ}_{CSR} exists, it is uniquely defined and we call it t_{i+1}. We proceed with this definition until $i = n$ or until there is some $l < n$ such that the successor of t_l with respect to f^{ρ}_{CSR} is not defined.

In order to prove that f^{ρ}_{CSR} can be extended to a well-defined certificate for substack reachability satisfying conditions 1–4, we show a stronger claim. For $1 \leq i \leq n+1$ such that t_i is defined, set $s_{t_i} := \mathrm{LStck}(t_i, \mathrm{Enc}(c))$. For $1 \leq i \leq n$ let s_i be the stack of $\rho_i(0)$. Let s_{n+1} be the stack of $\lambda(0)$.

Claim. For all $1 \leq i \leq n$ such that t_i is defined, t_{i+1} is also defined. Furthermore, if t_i is defined for some $1 \leq i \leq n+1$, then s_i and s_{t_i} are similar in the following sense:

1. $\mathrm{top}_2(s_i) = \mathrm{top}_2(s_{t_i})$ and
2. there is an $n_i \geq 1$ such that $\mathrm{pop}_2(s_{t_i}) = \mathrm{pop}_2^{n_i}(s_i)$.
3. Moreover, if $i = 1$ or ρ_{i-1} is of the form F1, F2, or F3. (with respect to Lemma 3.1.39), then $s_i = s_{t_i}$. This means that the substack represented by t_i is the initial stack of ρ_i whenever ρ_{i-1} ended in a pop_2 or collapse of level 2.

Before we prove the claim, let us explain how the lemma follows from the claim. Note that t_1 is defined and $s_{t_1} = \mathrm{LStck}(t_1, \mathrm{Enc}(q,s)) = s = s_1 = \rho_1(0)$. Due to the claim, $t_1, t_2, t_3, \ldots, t_{n+1}$ are defined. According to Lemma 3.1.39, ρ_n is of the form F1, F2, or F3. Thus, the claim implies that $\mathrm{LStck}(t_{n+1}, \mathrm{Enc}(c)) = s_{t_{n+1}} = s_{n+1} = \lambda(0) = \hat{s}$. Thus, t_{n+1} is the rightmost leaf of $\mathrm{Enc}(\hat{c})$. Since the successor with respect to f^{ρ}_{CSR} of some node is always lexicographically smaller than this node, $t_i >_{\mathrm{lex}} t_{n+1}$ for all $1 \leq i \leq n$.

Hence, $\{t_1, t_2, \ldots, t_n\} \subseteq \mathrm{dom}(\mathrm{Enc}(c)) \setminus \mathrm{dom}(\mathrm{Enc}(\hat{c}))$. Thus, we can extend the partial definition of f^{ρ}_{CSR} to a map from $\mathrm{dom}(\mathrm{Enc}(c)) \setminus \mathrm{dom}(\mathrm{Enc}(\hat{c}))$ to $\{1,2,3,4,5\} \times Q$. Furthermore, $\bar{t} = t_1, t_2, \ldots, t_n$ satisfies items 1–3 by definition of the t_i. $\bar{t}$ also satisfies item 4 because we proved that the successor t_{n+1} of t_n is the rightmost leaf of $\mathrm{Enc}(\hat{c})$.

Now, we prove the claim. Assume that there is some $i \leq n$ such that t_i is defined. Furthermore, assume that s_i and s_{t_i} are similar, i.e., s_i and s_{t_i} satisfy conditions 1–3 of the claim. We distinguish the following cases according to the form of t_i:

1. Consider the case $k_i = 1$. In this case, ρ_i is a return. If $i > 1$, then Lemma 3.1.39 implies that ρ_{i-1} is of the form F1, F2, or F3. Thus, item 3 of the claim implies that $s_i = s_{t_i}$.

   Due to $k_i = 1$, the successor of t_i – if defined – is a node t_{i+1} such that

   $$\mathrm{LStck}(t_{i+1}, \mathrm{Enc}(c)) = \mathrm{pop}_2(\mathrm{LStck}(t_i, \mathrm{Enc}(c))) = \mathrm{pop}_2(s_{t_i}).$$

   ρ_i is a return starting at $s_i = s_{t_i}$ whence ρ_i ends in $s_{i+1} := \mathrm{pop}_2(s_{t_i})$. Thus, we conclude that $|s_{t_i}| \geq 2$ whence t_{i+1} is defined. Furthermore, note that

   $$s_{t_{i+1}} = \mathrm{LStck}(t_{i+1}, \mathrm{Enc}(c)) = \mathrm{pop}_2(s_{t_i}) = s_{i+1}$$

   whence t_{i+1} satisfies item 3 of the claim.

2. Consider the case $k_i \in \{2,3\}$. This means that ρ_i ends with a collapse of level 2 from a stack with topmost word $\mathrm{top}_2(s_i)$. Thus, $\mathrm{CLvl}(s_i) = 2$. Since $k_i \in \{2,3\}$, the successor of t_i – if defined – is a node t_{i+1} such that

   $$\mathrm{LStck}(t_{i+1}, \mathrm{Enc}(c)) = \mathrm{collapse}(\mathrm{LStck}(t_i, \mathrm{Enc}(c)) = \mathrm{collapse}(s_{t_i}).$$

   Due to the form of ρ_i, $\mathrm{collapse}(s_i)$ is defined. Due to item 1 of the claim, $\mathrm{top}_1(s_{t_i})$ and $\mathrm{top}_1(s_i)$ coincide. Thus, $\mathrm{collapse}(s_{t_i})$ is also defined. But then t_{i+1} is defined whence the first part of the claim holds. Furthermore, item 2 of the claim implies that that

   $$s_{t_{i+1}} = \mathrm{collapse}(s_{t_i}) = \mathrm{collapse}(s_i) = s_{i+1}$$

   whence t_{i+1} satisfies the second part of the claim.

3. Consider the case $k_i \in \{4,5\}$. This means that ρ_i is a loop or 1-loop followed by a pop_1 or collapse of level 1. Since the topmost word of s_i and the topmost word before the last operation of ρ_i agree, $|\mathrm{top}_2(s_i)| > 1$ holds. Due to item 1 of the claim, $|\mathrm{top}_2(s_{t_i})| > 1$ follows. This implies that there is some node $d \in \mathrm{Enc}(c)$ such that $t_i = d01^l$ for some $l \in \mathbb{N}$. Since $k_i \in \{4,5\}$, d is the successor of t_i with respect to f^{ρ}_{CSR}, i.e., $t_{i+1} = d$ whence the first part of the claim holds.

   For the second part, note that there is some $m \geq 1$ such that $\mathrm{pop}_2(s_i) = \mathrm{pop}_2^m(s_{i+1})$ due to the definition of loops and 1-loops.

   Since the left stack induced by t_{i+1} is a milestone of the one induced by t_i, there is an $n \geq 1$ such that $\mathrm{pop}_2(s_{t_{i+1}}) = \mathrm{pop}_2^n(s_{t_i})$. Thus, by item 2 of the claim, we obtain that $\mathrm{pop}_2(s_{t_{i+1}}) = \mathrm{pop}_2^{n+m+n_i-2}(s_{i+1})$. Since $n_{i+1} := n + m + n_i - 2 \geq 1$, item 2 of the claim holds for t_{i+1}.

   Furthermore, since $\mathrm{top}_2(s_{i+1}) = \mathrm{top}_2(\mathrm{pop}_1(s_i))$ and $\mathrm{top}_2(s_{t_{i+1}}) = \mathrm{top}_2(\mathrm{pop}_1(s_{t_i}))$, item 1 of the claim carries over from t_i to t_{i+1}. This completes the proof that s_{i+1} and $s_{t_{i+1}}$ are similar in the sense of the claim. □

Remark 3.1.47. In the following we say that a certificate for substack reachability f_{CSR} represents ρ if it coincides with f^{ρ}_{CSR} on $\{t_1, t_2, \ldots, t_n\}$.

In the next lemma, we collect important properties of a certificate which represents some run. Afterwards, we turn these properties into the defining conditions of valid certificates. This terminology is justified because each valid certificate represents in fact some run.

Lemma 3.1.48. Let $(q,s),(\hat{q},\hat{s})$ be configurations such that $\hat{s} = \mathsf{pop}_2^m(s)$ for some $m \in \mathbb{N}$. Let f_{CSR} be a certificate representing a run ρ from (q,s) to $(\hat{q},\hat{s})$.

Then there is an $n \in \mathbb{N}$ and a finite sequence $t_1, t_2, \ldots, t_n \in \mathsf{dom}(\mathsf{Enc}(q,s)) \setminus \mathsf{dom}(\mathsf{Enc}(\hat{q},\hat{s}))$ with the following properties (setting $(k_i, q_i) := f_{\mathsf{CSR}}(t_i)$ and $q_0 := q$):

A1. t_1 is the rightmost leaf of $\mathsf{Enc}(q,s)$,

A2. for all $1 \leq i < n$, the successor of t_i with respect to f_{CSR} is t_{i+1},

A3. the successor of t_n with respect to f_{CSR} is the rightmost leaf of $\mathsf{Enc}(\hat{q},\hat{s})$,

A4. $k_n \in \{1,2,3\}$,

A5. $(q_n,\hat{q}) \in \exists\mathsf{HLoops}(\hat{s})$, i.e., there is a high loop from $(q_n,\hat{s})$ to $(\hat{q},\hat{s})$,

A6. if $k_i \in \{4,5\}$ for some $i < n$ then there is a $j > i$ such that $k_j \in \{2,3\}$ and $k_l \neq 1$ for all $i < l < j$,

A7. For each $1 \leq i \leq n$, the stack induced by t_i satisfies in dependence of the value of k_i a certain assertion as follows:

a) if $k_i = 1$ then $(q_{i-1}, q_i) \in \exists\mathsf{Returns}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s)))$,

b) if $k_i = 2$ then $\mathsf{CLvl}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))) = 2$ and there is a $q' \in Q$ and a $\gamma \in \Gamma$ such that

$$(q_{i-1}, q') \in \exists\text{1-}\mathsf{Loops}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))) \text{ and}$$
$$(q', \mathsf{Sym}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))), \gamma, q_i, \mathsf{collapse}) \in \Delta,$$

c) if $k_i = 3$ then $\mathsf{CLvl}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))) = 2$ and there is some $q' \in Q$ and some $\gamma \in \Gamma$ such that

$$(q_{i-1}, q') \in \exists\mathsf{Loops}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))) \text{ and}$$
$$(q', \mathsf{Sym}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))), \gamma, q_i, \mathsf{collapse}) \in \Delta,$$

d) if $k_i = 4$ then there is some $q' \in Q$ and some $\gamma \in \Gamma$ such that

$(q_{i-1}, q') \in \exists\mathsf{Loops}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s)))$ and either
$(q', \mathsf{Sym}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))), \gamma, q_i, \mathsf{pop}_1) \in \Delta$ or
$\mathsf{CLvl}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))) = 1$ and $(q', \mathsf{Sym}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))), \gamma, q_i, \mathsf{collapse}) \in \Delta$.

e) if $k_i = 5$ then there is a $q' \in Q$ such that

$(q_{i-1}, q') \in \exists\text{1-}\mathsf{Loops}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s)))$ and either
$(q', \mathsf{Sym}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))), \gamma, q_i, \mathsf{pop}_1) \in \Delta$ or
$\mathsf{CLvl}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))) = 1$ and $(q', \mathsf{Sym}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))), \gamma, q_i, \mathsf{collapse}) \in \Delta$.

Proof. Let f_{CSR} represent a run ρ.

There is a unique sequence $t_1, t_2, \ldots, t_n$ of maximal length that satisfies A1 and A2. Furthermore, the previous lemma showed that t_n satisfies A3.

From the previous lemma we also know that $f_{\mathrm{CSR}}(t_i)$ encodes the form and the final state of ρ_i where $\rho = \rho_1 \circ \rho_2 \circ \cdots \circ \rho_n \circ \lambda$ is the decomposition of ρ according to Lemma 3.1.39. Thus, k_n is the form of ρ_n. Hence, Lemma 3.1.39 implies that $k_n \in \{1, 2, 3\}$.

q_n is the final state of ρ_n and due to Lemma 3.1.39, λ is a high loop from $(q_n, \hat{s})$ to $(\hat{q}, \hat{s})$. Thus, λ witnesses that $(q_n, \hat{q}) \in \exists\mathsf{HLoops}(\hat{s})$. This is exactly the assertion of A5.

A6 is also a direct consequence of Lemma 3.1.39: if there is some ρ_i of the form F4 or F5, then there is a $j > i$ such that ρ_j is of the form F2 or F3, and for all $i < k < j$, ρ_k is not of the form F1. From the correspondence between the form of ρ_l and the value of k_l for all $1 \leq l \leq n$, A6 follows directly.

A7 is a consequence of the claim in the previous proof. There we showed that

$$\mathrm{top}_2(\mathsf{LStck}(t_i, \mathsf{Enc}(q, s))) = \mathrm{top}_2(\rho_i(0)). \tag{3.1}$$

Thus, $\exists\mathsf{Returns}, \exists\mathsf{Loops}$ and $\exists\mathsf{1\text{-}Loops}$ agree on the stacks $\mathsf{LStck}(t_i, \mathsf{Enc}(q, s))$ and $\rho_i(0)$. We conclude by case distinction on k_i as follows.

$k_i = 1$ This implies that ρ_i is a return from $\rho_i(0)$ to $(q_i, \mathrm{pop}_2(\rho_i(0)))$. By definition, the state of $\rho_i(0)$ is q_{i-1}. Thus, ρ_i witnesses

$$(q_{i-1}, q_i) \in \exists\mathsf{Returns}(\rho_i(0)) = \exists\mathsf{Returns}(\mathsf{LStck}(t_i, \mathsf{Enc}(q, s))).$$

$k_i = 2$ This implies that ρ_i is a 1-loop followed by a $\mathsf{collapse}$ of level 2. Let j be the position just before this $\mathsf{collapse}$, i.e., $j := \mathsf{ln}(\rho_i) - 1$. Let q' be the state of $\rho_i(j)$. Now, $\rho_i{\restriction}_{[o,j]}$ witnesses the existence of a 1-loop from state q_{i-1} to state q' on topmost word $\mathrm{top}_2(\rho_i(0))$. Thus, $(q_{i-1}, q') \in \exists\mathsf{1\text{-}Loops}(\mathsf{LStck}(t_i, \mathsf{Enc}(q, s)))$.

Due to (3.1) and the definition of 1-loops, we have

$$\mathrm{top}_1(\mathsf{LStck}(t_i, \mathsf{Enc}(q, s))) = \mathrm{top}_1(\rho_i(0)) = \mathrm{top}_1(\rho_i(j)).$$

By definition of ρ_i, $\mathsf{CLvl}(\rho_i(j)) = 2$. We conclude that

$$\mathsf{CLvl}(\mathsf{LStck}(t_i, \mathsf{Enc}(q, s))) = \mathsf{CLvl}(\rho_i(j)) = 2.$$

Since ρ_i performs a collapse at j, there is some transition

$$(q', \mathsf{Sym}(\rho(j)), \gamma, q_i, \mathsf{collapse}) \in \Delta.$$

Due to $\mathsf{Sym}(\rho(j)) = \mathsf{Sym}(\mathsf{LStck}(t_i, \mathsf{Enc}(q, s)))$, this transition witnesses that

$$(q', \mathsf{Sym}(\mathsf{LStck}(t_i, \mathsf{Enc}(q, s))), \gamma, q_i, \mathsf{collapse}) \in \Delta.$$

$k_i = 3$ Replacing the role of 1-loops by loops, we can copy the proof from the previous case word by word.

$k_i = 4$ This implies that ρ_i is a loop followed by a pop_1 transition or a $\mathsf{collapse}$ of level 1. We set $j := \mathsf{ln}(\rho_i) - 1$ and q' to be the state of $\rho_i(j)$. Completely analogous to the previous case, one derives that $(q_{i-1}, q') \in \exists\mathsf{Loops}(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s)))$.

The transition at $\rho_i(j)$ is a pop_1 or a $\mathsf{collapse}$ of level 1. Thus, this transition is either $(q', \mathsf{Sym}(\rho_i(j)), \gamma, q_i, \mathsf{pop}_1)$ or $(q', \mathsf{Sym}(\rho_i(j)), \gamma, q_i, \mathsf{collapse})$ and $\mathsf{CLvl}(\rho_i(j)) = 1$. Due to (3.1), $\mathsf{top}_2(\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))) = \mathsf{top}_2(\rho_i(j))$. Thus, this transition is also applicable to $\mathsf{LStck}(t_i, \mathsf{Enc}(q,s))$. This completes the proof in the case $k_i = 4$.

$k_i = 5$ This case is completely analogous to the previous one: we only have to replace loops by 1-loops. □

Definition 3.1.49. Let $c = (q,s)$ and $\hat{c} = (\hat{q}, \hat{s})$ be configurations such that $\hat{s} = \mathsf{pop}_2^n(\bar{s})$. Let $f_{\mathsf{CSR}} : \mathsf{dom}(\mathsf{Enc}(c)) \setminus \mathsf{dom}(\mathsf{Enc}(\hat{c}) \to \{1,2,3,4,5\} \times Q$ be a certificate for substack reachability on c and $\hat{c}$. Setting $q_0 := q$, we call f_{CSR} valid if there is an $n \in \mathbb{N}$ and a finite sequence $t_1, t_2, \ldots, t_n \in \mathsf{dom}(\mathsf{Enc}(c)) \setminus \mathsf{dom}(\mathsf{Enc}(\hat{c}))$ which satisfies conditions A1 – A7 from Lemma 3.1.48.

The next lemma shows the tight correspondence between valid certificates of substack reachability and runs. For $q, \hat{q} \in Q$ two states, s some stack and $\hat{s} = \mathsf{pop}_2^n(s)$ for some $n \in \mathbb{N}$, there is a run from (q,s) to $(\hat{q}, \hat{s})$ if and only if there is a valid certificate for substack reachability for (q,s) and $(\hat{q}, \hat{s})$.

Lemma 3.1.50. Let $c = (q,s)$ and $\hat{c} = (\hat{q}, \hat{s})$ be configurations such that $\hat{s} = \mathsf{pop}_2^m(s)$ for some $m \in \mathbb{N}$. There is a run from c to $\hat{c}$ which does not visit proper substacks of $\hat{s}$ if and only if there is a valid certificate for substack reachability

$$f_{\mathsf{CSR}} : \mathsf{dom}(\mathsf{Enc}(c)) \setminus \mathsf{dom}(\mathsf{Enc}(\hat{c})) \to \{1,2,3,4,5\} \times Q.$$

Proof. The implication from left to right follows from Lemma 3.1.46.

For the proof from right to left assume that f_{CSR} is a valid certificate for substack reachability for c and $\hat{c}$.

Then there is a sequence $t_1, t_2, \ldots, t_n$ in $\mathsf{dom}(f_{\mathsf{CSR}})$ that witnesses the conditions A1–A7.

We now construct runs $\rho_0, \rho_1, \rho_2, \ldots, \rho_n, \rho_{n+1}$ such that ρ_i is an initial segment of ρ_{i+1} for each $i \leq n$. The run ρ_{n+1} is then a run from c to $\hat{c}$.

Before we start the construction, let us define some notation. For all $1 \leq i \leq n$, let $(q_i, k_i) := f_{\mathsf{CSR}}(t_i)$ and let $s_{t_i} := \mathsf{LStck}(t_i, \mathsf{Enc}(c))$. Furthermore, for reasons of convenience, we set $q_0 := q$ and we set t_{n+1} to be the rightmost leaf of $\mathsf{Enc}(\hat{c})$. As soon as ρ_{i-1} is defined for some $1 \leq i \leq n$, we denote by s_i the last stack of ρ_{i-1}.

We define ρ_0 to be a run of length 0 with $\rho_0(0) := c$.

During the construction of ρ_i for $1 \leq i \leq n$, we preserve the following conditions:

1. the last state of ρ_{i-1} is q_{i-1},
2. $\mathsf{top}_2(s_i) = \mathsf{top}_2(s_{t_i})$, and
3. there is an $n_i \geq 1$ such that $\mathsf{pop}_2(s_{t_i}) = \mathsf{pop}_2^{n_i}(s_i)$.
4. Moreover, if $i = 1$ or $k_{i-1} \in \{1,2,3\}$, then $s_i = s_{t_i}$.

Note that $\rho_0(0) = (q, s_0) = (q_0, s_{t_0})$ by definition whence for $i = 1$ these conditions are satisfied.

Now assume that ρ_{i-1} is defined for some $1 \leq i \leq n$ such that these conditions are satisfied. By case distinction on the value of k_i we define ρ_i as follows.

$k_i = 1$ Since f_{CSR} satisfies A6, $i = 1$ or $k_{i-1} \in \{1, 2, 3\}$. Thus, $s_i = s_{t_i}$ by induction hypothesis. Due to A7, we have $(q_{i-1}, q_i) \in \exists\mathsf{Returns}(s_{t_i}) = \exists\mathsf{Returns}(s_i)$. Hence, there is a return $\hat{\rho}$ from (q_{i-1}, s_i) to $(q_i, \mathsf{pop}_2(s_i))$. We set

$$\rho_i := \rho_{i-1} \circ \hat{\rho}.$$

Due to A2 and A3, the successor of t_i with respect to f_{CSR} is t_{i+1}. Since $k_i = 1$, $s_{t_{i+1}} = \mathsf{LStck}(t_{i+1}, \mathsf{Enc}(c)) = \mathsf{pop}_2(s_{t_i}) = \mathsf{pop}_2(s_i) = s_{i+1}$.

$k_i = 2$ Due to A7, $\mathsf{CLvl}(s_{t_i}) = 2$ and there is a $q' \in Q$ and a $\gamma \in \Gamma$ such that

$$(q_{i-1}, q') \in \exists 1\text{-}\mathsf{Loops}(s_{t_i})$$
$$\text{and } \delta := (q', \mathsf{Sym}(s_{t_i}), \gamma, q_i, \mathsf{collapse}) \in \Delta.$$

Since the topmost words of s_i and s_{t_i} agree, there is some stack s' such that there is a 1-loop $\hat{\lambda}$ from (q_{i-1}, s_i) to (q', s'). By definition of a 1-loop,

$$\mathsf{top}_2(s') = \mathsf{top}_2(s_i) = \mathsf{top}_2(s_{t_i}).$$

Thus, $\mathsf{CLvl}(s') = 2$ and $\hat{\lambda}$ can be extended by δ. We write $\hat{\lambda}^+$ for $\hat{\lambda}$ extended by one application of δ. Since $\hat{\lambda}$ is a 1-loop, it does not visit any substacks of $\mathsf{pop}_2(s_i)$. Thus, $\mathsf{collapse}(s_i) = \mathsf{collapse}(s')$. Set $\rho_i := \rho_{i-1} \circ \hat{\lambda}^+$. By assumption, we conclude that

$$s_{t_{i+1}} = \mathsf{collapse}(s_{t_i}) = \mathsf{collapse}(s_i) = \mathsf{collapse}(s') = s_{i+1}.$$

Thus, the last configuration of ρ_i is $(q_i, s_{t_{i+1}})$.

$k_i = 3$ We can copy the argument from the case $k_i = 2$: just replace 1-loops by loops. Then we obtain a run ρ_i that ends in $(q_i, s_{t_{i+1}})$.

$k_i = 4$ Due to condition A7, there is a $q' \in Q$ and a $\gamma \in \Gamma$ such that $(q_{i-1}, q') \in \exists\mathsf{Loops}(s_{t_i})$ and $\delta_p := (q', \mathsf{Sym}(s_{t_i}), \gamma, q_i, \mathsf{pop}_1) \in \Delta$ or $\delta_c := (q', \mathsf{Sym}(s_{t_i}), \gamma, q_i, \mathsf{collapse}) \in \Delta$ and $\mathsf{CLvl}(s_{t_i}) = 1$.

Since the topmost words of s_i and s_{t_i} agree, there is a loop $\hat{\lambda}$ from (q_{i-1}, s_i) to (q', s_i). Due to $\mathsf{top}_2(s_i) = \mathsf{top}_2(s_{t_i})$, it follows that $\mathsf{CLvl}(s_i) = \mathsf{CLvl}(s_{t_i})$. We conclude that δ_c (or δ_p, respectively) can be applied to the last configuration of $\hat{\lambda}$. In both cases, the resulting configuration is $(q_i, \mathsf{pop}_1(s_i))$. Writing $\hat{\lambda}^+$ for $\hat{\lambda}$ extended by δ_p or δ_c, we set $\rho_i := \rho_{i-1} \circ \hat{\lambda}^+$.

By definition of t_{i+1}, $s_{t_{i+1}}$ is the maximal milestone of s_{t_i} such that

$$\mathsf{top}_2(s_{t_{i+1}}) = \mathsf{top}_2(\mathsf{pop}_1(s_{t_i})) = \mathsf{top}_2(s_{i+1}).$$

We still have to show that there is some $n_{i+1} \geq 1$ such that $\mathrm{pop}_2(s_{t_{i+1}}) = \mathrm{pop}_2^{n_{i+1}}(s_{i+1})$.

Since $s_{t_{i+1}}$ is a milestone of s_{t_i}, there is some $j \geq 1$ such that $\mathrm{pop}_2(s_{t_{i+1}}) = \mathrm{pop}_2^{j}(s_{t_{i+1}})$. Since $\hat{\lambda}$ is a loop, we have $\mathrm{pop}_2(s_{i+1}) = \mathrm{pop}_2(s_i)$. By assumption on ρ_{i-1}, we obtain that

$$\mathrm{pop}_2(s_{t_{i+1}}) = \mathrm{pop}_2^{j}(s_{t_i}) = \mathrm{pop}_2^{j+n_i-1}(s_i) = \mathrm{pop}_2^{j+n_i-1}(s_{i+1}).$$

Let $n_{i+1} := j + n_i - 1$. We conclude by noting that $n_{i+1} \geq 1$.

$k_i = 5$ This case is analogous to the previous one. We can replace the loop $\hat{\lambda}$ in the previous case by some 1-loop from (q_{i-1}, s_i) to some (q', s') where $\mathrm{pop}_2(s_i)$ is a substack of s' and $\mathrm{top}_2(s') = \mathrm{top}_2(s_i)$. The rest of the argument is then completely analogous.

Repeating this construction for all $i \leq n$, we define a run ρ_n with last state q_i. Due to A4, the last step in this construction uses one of the first three cases. Thus,

$$s_{n+1} = s_{t_{n+1}} = \mathrm{LStck}(t_{n+1}, \mathrm{Enc}(c)) = \hat{s}.$$

Note that the last equality is due to A3. Thus, ρ_n ends in $(q_n, \hat{s})$.

Due to A5, there is a high loop λ from $(q_n, \hat{s})$ to $(\hat{q}, \hat{s})$. We set $\rho_{n+1} := \rho_n \circ \lambda$. This completes the proof because ρ_{n+1} is a run from (q, s) to $(\hat{q}, \hat{s})$ that does not visit any proper substack of $\hat{s}$. □

We have seen that there is a run from $c = (q, s)$ to $\hat{c} = (\hat{q}, \hat{s})$ for $\hat{s} = \mathrm{pop}_2^n(s)$ if and only if there is a valid certificate for substack reachability on $\mathrm{Enc}(c) \otimes \mathrm{Enc}(\hat{c})$. The final step of the analysis of runs of this form is the following lemma. We prove that the set of all pairs of configurations $(c, \hat{c})$ of the form mentioned above is an automatic relation via the encoding Enc. We show that the set of encodings of such pairs is MSO-definable. The automaticity of the relation follows from the correspondence of MSO and automata on trees.

Lemma 3.1.51. There is a formula in MSO that defines the set

$$S := \{(\mathrm{Enc}(c), \mathrm{Enc}(\hat{c})) : \exists f_{\mathrm{CSR}} : \mathrm{dom}(\mathrm{Enc}(c)) \setminus \mathrm{dom}(\mathrm{Enc}(\hat{c})) \to \{1, \ldots, 5\} \times Q, f_{\mathrm{CSR}} \text{ is valid}\}$$

Proof. Certificates for substack reachability are only defined for configurations $c = (q, s)$ and $\hat{c} = (\hat{q}, \hat{s})$ where $\hat{s} = \mathrm{pop}_2^m(s)$ for some $m \in \mathbb{N}$. Note that this necessary condition is satisfied by a pair $(c, \hat{c})$ if and only if there is some node $d \in \mathrm{dom}(\mathrm{Enc}(c))$ such that $d0 \notin \mathrm{dom}(\mathrm{Enc}(c))$ and $\mathrm{LStck}(d, \mathrm{Enc}(c)) = \hat{s}$. These pairs of configurations are obviously MSO definable.

In this proof, we use the following claim.

Claim. There is an MSO formula φ such that for each certificate for substack reachability f_{CSR} on c and $\hat{c}$ the following holds. f_{CSR} is valid if and only if

$$\mathrm{Enc}(c) \otimes \mathrm{Enc}(\hat{c}) \otimes f_{\mathrm{CSR}} \otimes T_{HL} \otimes T_L \otimes T_{1L} \otimes T_R \models \varphi$$

where T_{HL}, T_L, T_{1L}, T_R are trees such that

$$\begin{aligned}
&T_{HL} \text{ encodes the mapping } d \mapsto \exists\mathsf{HLoops}(\mathsf{LStck}(d, \mathsf{Enc}(c))),\\
&T_L \text{ encodes the mapping } d \mapsto \exists\mathsf{Loops}(\mathsf{LStck}(d, \mathsf{Enc}(c))),\\
&T_{1L} \text{ encodes the mapping } d \mapsto \exists\mathsf{1\text{-}Loops}(\mathsf{LStck}(d, \mathsf{Enc}(c))), \text{ and}\\
&T_R \text{ encodes the mapping } d \mapsto \exists\mathsf{Returns}(\mathsf{LStck}(d, \mathsf{Enc}(c))).
\end{aligned}$$

Before we prove this claim, we show that it implies the lemma. Due to Propositions 2.4.19, 2.4.47, and 3.1.38, the trees T_{HL}, T_L, T_{1L} and T_R are definable on $\mathsf{Enc}(c)$ using MSO.

Furthermore, in Remark 3.1.42 we saw that MSO can express the existence of a certificate for substack reachability on $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$. Thus, given the formula φ from the claim, we can construct a formula ψ asserting that "there is a certificate for substack reachability f_{CSR} on $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$ such that

$$\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c}) \otimes f_{\mathsf{CSR}} \otimes T_{HL} \otimes T_L \otimes T_{1L} \otimes T_R \models \varphi".$$

This MSO formula defines the set S.

Let us now prove the claim. As an abbreviation, we write

$$\mathfrak{A} := \mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c}) \otimes f_{\mathsf{CSR}} \otimes T_{HL} \otimes T_L \otimes T_{1L} \otimes T_R.$$

We provide formulas that assert the conditions A1–A7. The rightmost leaf of $\mathsf{Enc}(c)$ is of course MSO-definable in $\mathfrak{A}$. Furthermore, the successor of a given node d with respect to f_{CSR} is also MSO-definable. Thus, the uniquely defined maximal set $T := \{t_1, t_2, \ldots, t_n\}$ such that the sequence $t_1, t_2, \ldots, t_n$ satisfies condition A1 and A2 is MSO-definable.

Note that $i < j$ is equivalent to $t_j \leq_{\mathsf{lex}} t_i$. Since the lexicographic order on $\mathsf{dom}(\mathsf{Enc}(c))$ is MSO-definable, the successor of t with respect to f_{CSR} is definable for each $t \in T$. But this implies directly that the lexicographically minimal element in T is t_n. Thus, t_n is definable and we can express "the successor of t_n with respect to f_{CSR} is the rightmost leaf of $\mathsf{Enc}(\hat{q}, \hat{s})$" in an MSO formula. This formula expresses A3.

Furthermore, we conclude that "$f_{\mathsf{CSR}}(t_n) = (k_n, q_n)$ such that $k_n \in \{1, 2, 3\}$" is definable because t_n is definable. Thus, we can express A4.

Next, we define a formula expressing A5. The label of the root of $\mathfrak{A}$ encodes the state $\hat{q}$. "$(q_n, \hat{q}) \in \exists\mathsf{HLoops}(\hat{s})$" is expressible because $\hat{q}$ is encoded in the label of t_n and $\exists\mathsf{HLoops}(\hat{s})$ is encoded in the label of the rightmost leaf of $\mathsf{Enc}(\hat{q}, \hat{s})$. This leaf is definable in $\mathfrak{A}$. Thus, we conclude that A5 is expressible by some MSO formula.

Condition A6 says that after some t_i with $k_i \in \{4, 5\}$ there is a t_j with $k_j \in \{2, 3\}$ before the next occurrence of some $k_l = 1$. Since the order of the t_i is definable and since the k_i are encoded in the labels of the t_i, this is clearly MSO-definable.

Finally, note that condition A7 only depends on the values of f_{CSR} on the t_i and on the values of $\exists\mathsf{Returns}$, $\exists\mathsf{Loops}$ and $\exists\mathsf{1\text{-}Loops}$ for $\mathsf{LStck}(t_i, \mathsf{Enc}(c))$. But all these information are encoded in the labels of the t_i whence condition A7 is MSO-definable

Thus, we conclude that the validity of a certificate for substack reachability is expressible in an MSO formula on $\mathfrak{A}$. This proves the claim. The lemma follows from the claim as indicated above. □

The following corollary summarises the results obtained so far.

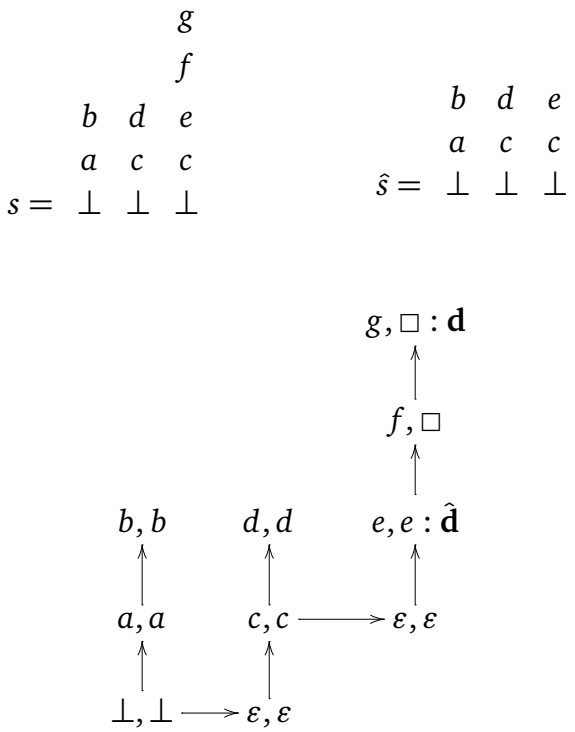


Figure 3.13.: Illustration of the first case of the proof of lemma 3.1.53. For better orientation, we have marked the rightmost leaves of the two encodings by $\mathbf{d}$ and $\hat{\mathbf{d}}$.

Corollary 3.1.52. Let $\mathscr{S}$ be some collapsible pushdown system. The relation A from Definition 3.1.32 is a regular relation via Enc.

Having shown the regularity of A, we prove the regularity of B, C and D in the following. We then obtain the regularity of REACH as a corollary.

Regularity of the Relation **B**

B contains pairs $(c, \hat{c})$ where $c = (q, s)$, $\hat{c} = (\hat{q}, \hat{s})$, and $\hat{s} = \mathsf{pop}_1^m(s)$ such that there is a run from c to $\hat{c}$ not passing any proper substack of $\hat{s}$. By definition, such a run is a composition of high loops and pop_1 or $\mathsf{collapse}$ of level 1.

Recall that we already dealt with a similar problem. In the previous section we investigated milestones m_1, m_2 where $m_2 = \mathsf{pop}_1^n(\mathsf{clone}_2(m_1))$ and proved that the existence of a run from m_1 to m_2 with a given initial and final state is MSO-definable (cf. Lemma 3.1.25). The following lemma adapts the same idea and proves the regularity of B.

Lemma 3.1.53. B is regular via Enc.

Proof. Let us first recall the structure of $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$ where $c = (q, s)$, $\hat{c} = (\hat{q}, \hat{s})$ and $\hat{s} = \mathsf{pop}_1^n(s)$ for some $n \in \mathbb{N}$. There are two cases.

1. Let us first assume that $\hat{s}$ is a milestone of s. Figure 3.13 shows an example of this case. Let $\hat{d}$ be the rightmost leaf in $\mathsf{Enc}(\hat{c})$. $\mathsf{dom}(\mathsf{Enc}(c))$ extends $\mathsf{dom}(\mathsf{Enc}(\hat{c}))$ by the nodes $\hat{d}0, \hat{d}00, \ldots, \hat{d}0^n$. The labels on the path from $\hat{d}0$ to the rightmost leaf $d := \hat{d}0^n$ of $\mathsf{Enc}(c)$ encode the suffix w such that $\mathsf{top}_2(s) = \mathsf{top}_2(\hat{s}) \circ w$.

   Note that this condition on the domains of $\mathsf{Enc}(c)$ and $\mathsf{Enc}(\hat{c})$ is MSO-definable whence the pairs of configurations of this form are regular via Enc.

2. Now assume that $\hat{s}$ is not a milestone of s. Figure 3.14 shows an example of this case. Let d be the rightmost leaf of $\mathsf{Enc}(c)$ and analogously let $\hat{d}$ be the rightmost leaf of $\mathsf{Enc}(\hat{c})$. Let e be the second rightmost element in $\mathsf{Enc}(c)$ without left successor. In fact, e is the second rightmost leaf of $\hat{s}$. There are nodes $\hat{f} < f \leq e$ such that
   a) $\hat{d} = \hat{f}1$,
   b) $d = f10^m$ for some $m < n$ and
   c) $\mathsf{LStck}(e, \mathsf{Enc}(c)) = \mathsf{LStck}(e, \mathsf{Enc}(\hat{c}))$, i.e., the encodings of s and $\hat{s}$ agree on the elements that are lexicographically smaller than e.

   Moreover, the path from $\hat{f}0$ to d encodes the suffix v such that $\mathsf{top}_2(s) = \mathsf{top}_2(\hat{s}) \circ v$.

   Note that these conditions on the domains of $\mathsf{Enc}(c)$ and $\mathsf{Enc}(\hat{c})$ are MSO-definable whence the pairs of configurations of this form are regular via Enc.

We conclude that the encodings of pairs of configurations such that the stack of the second one is obtained from the first one by a sequence of pop_1 operations forms a regular set S.

We show that there is an MSO formula ψ that defines the relation B from Lemma 3.1.32 relatively to S.

We only present the proof for configurations of the second form. The proof for the first case is analogous by replacing $\hat{f}0$ by $\hat{d}0$.

Recall that $\hat{s} = \mathsf{pop}_1^n(s)$. There are nodes

$$\hat{f} =: g_0 < g_1 < g_2 < \cdots < g_{n-1} < g_n \leq d$$

(uniquely determined) such that $g_i \in \{0,1\}^*0$. These are uniquely determined because there are exactly n letters encoded on the path from $\hat{f}$ to d and each left-successor on this path corresponds to one of the letters.

Let w_i be the topmost word of $\mathsf{LStck}(g_i, \mathsf{Enc}(c))$. These form a chain

$$\mathsf{top}_2(\hat{c}) = w_0 < w_1 < w_2 < w_3 < \cdots < w_n = \mathsf{top}_2(c)$$

where w_{i+1} extends w_i by exactly one letter. Thus,

$$\exists \mathsf{HLoops}(\mathsf{pop}_1^m(c)) = \exists \mathsf{HLoops}(\mathsf{LStck}(g_{n-m}, \mathsf{Enc}(c))) \text{ for all } m \leq n.$$

Since $\exists \mathsf{HLoops}(\mathsf{LStck}(g_{n-m}, \mathsf{Enc}(c)))$ is definable at g_{n-m} in $\mathsf{Enc}(c)$, we can MSO-definably access the pairs of initial and final states of all $\mathsf{pop}_1^m(c)$.

Recall that we are looking for a run from c to $\hat{c}$ that do not visit proper substacks of s. Such a run consists of a sequence of high loops combined with pop_1 or $\mathsf{collapse}$ of level 1.

Since the set $\{g_i : 0 \leq i \leq n\}$ is MSO-definable and since their order is also MSO-definable, there is a formula which is satisfied by $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$ if and only if there is a run from c to $\hat{c}$: given a function f that labels each g_i with some state q_i, we can check whether there is a loop followed by a pop_1 or $\mathsf{collapse}$ of level 1 from $(q_{n-m}, \mathsf{pop}_1^m(s))$ to $(q_{n-m-1}, \mathsf{pop}_1^{m+1}(s))$ such that the following holds:

1. $q_n = q$ i.e., q_n is the state of $c = (q, s)$ and
2. there is loop from $(q_0, \hat{s})$ to $\hat{c} = (\hat{q}, \hat{s})$.

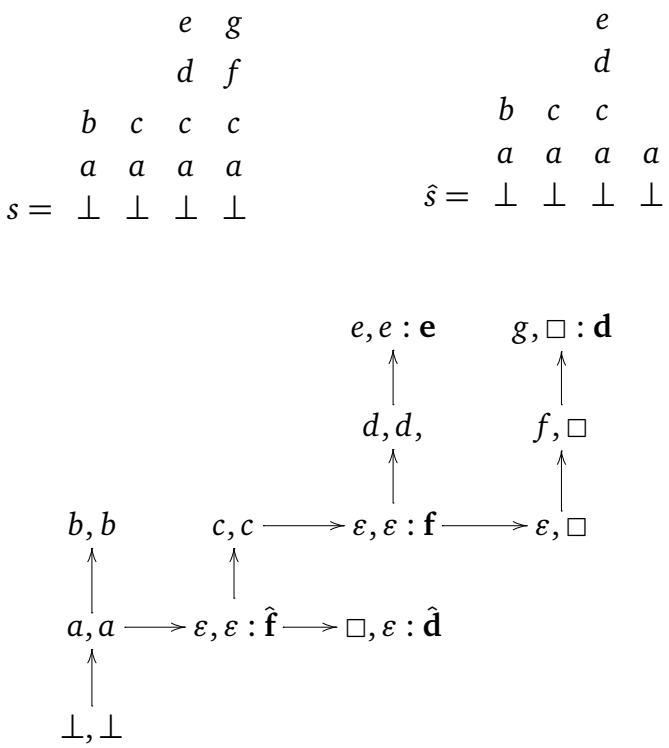


Figure 3.14.: Illustration of the second case of the proof of lemma 3.1.53. For better orientation, we have marked the nodes $\mathbf{d}$, $\hat{\mathbf{d}}$, $\mathbf{e}$, $\mathbf{f}$, and $\hat{\mathbf{f}}$.

The function f can be encoded by $|Q|$ many sets. Thus, there is a formula asserting that there is a function f which satisfies the conditions mentioned above.

For all configurations $(c, \hat{c}) \in S$, $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$ satisfies this formula if and only if $(c, \hat{c}) \in B$, i.e., if there is a run from c to $\hat{c}$ that does not visit a proper substack of $\hat{c}$.

Since we have already seen that S is also MSO-definable, we conclude that the Relation B is regular via Enc. □

Regularity of the Relation $\mathbf{C}$

Recall that the relation C contains a pair of configurations $(c, \hat{c})$ with $c = (q, s)$ and $\hat{c} = (\hat{q}, \hat{s})$ if and only if the following holds: $s = \mathsf{pop}_1^m(\hat{s})$ for some $m \in \mathbb{N}$ and there is a run from c to $\hat{c}$ that does not pass a proper substack of s. We prove the regularity of C analogously to the proof of Lemma 3.1.53.

Lemma 3.1.54. The relation C from Definition 3.1.32 is a regular relation via Enc.

Proof. We proceed completely analogous to Lemma 3.1.53.

Let us first recall the structure of $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$ where $c = (q, s)$, $\hat{c} = (\hat{q}, \hat{s})$ and $\hat{s}$ can be generated from s by a sequence of $\mathsf{push}_{\sigma,i}$ of length $m \in \mathbb{N}$. There are two cases.

1. Let us first assume that s is a milestone of $\hat{s}$. Let d be the rightmost leaf in $\mathsf{Enc}(c)$. $\mathsf{dom}(\mathsf{Enc}(\hat{c}))$ extends $\mathsf{dom}(\mathsf{Enc}(c))$ by the nodes $d0, d00, \ldots, d0^m$. The labels on the path from $d0$ to the rightmost leaf $\hat{d} := d0^m$ of $\mathsf{Enc}(\hat{c})$ encode the suffix w such that $\mathsf{top}_2(\hat{s}) = \mathsf{top}_2(s) \circ w$.

Note that this condition on the domains of $\mathsf{Enc}(c)$ and $\mathsf{Enc}(\hat{c})$ is MSO-definable whence the pairs of configurations of this form are regular via Enc.

2. Now assume that s is not a milestone of $\hat{s}$. Let d be the rightmost leaf of $\mathsf{Enc}(c)$ and $\hat{d}$ be the rightmost leaf of $\mathsf{Enc}(\hat{c})$. Let e be the second rightmost element in $\mathsf{Enc}(\hat{c})$ without left successor. In fact, e is the second rightmost leaf of $\mathsf{Enc}(c)$. There are nodes $f < \hat{f} \leq e$ such that

   a) $d = f1$,

   b) $\hat{d} = \hat{f}10^m$ for some $m < n$, and

   c) $\mathsf{LStck}(e, \mathsf{Enc}(c)) = \mathsf{LStck}(e, \mathsf{Enc}(\hat{c}))$, i.e., the encodings of s and $\hat{s}$ agree on the elements that are lexicographically less or equal to e.

   Moreover, the path from $f0$ to $\hat{d}$ encodes the suffix w such that $\mathsf{top}_2(\hat{s}) = \mathsf{top}_2(s) \circ w$. Note that these conditions are similar to those in the proof of lemma 3.1.53 with exchanged roles for s and $\hat{s}$.

   But in this proof there is one further condition on the encodings of c and $\hat{c}$. The path from $f0$ to $\hat{f}$ may only encode letters with link level 1. This stems from the following fact.

   Since $\mathsf{top}_2(s)$ is a prefix of $\mathsf{top}_2(\hat{s})$ and s is no milestone of $\hat{s}$, $\mathsf{top}_2(s)$ is a proper prefix of the greatest common prefix of the two topmost words of $\hat{s}$, i.e.,

   $$\mathsf{top}_2(s) \leq \mathsf{top}_2(\hat{s}) \sqcap \mathsf{top}_2(\mathsf{pop}_2(\hat{s})).$$

   Furthermore, $\hat{f}$ is defined in such a way that $\mathsf{LStck}(\hat{f}1, \mathsf{Enc}(\hat{c}))$ is the minimal milestone of $\hat{s}$ that has width $|\hat{s}|$. Thus, the elements encoded along the path from $f0$ to $\hat{f}$ are also contained in the second topmost word of $\hat{s}$. Thus, if any of these is of link level 2, then it points strictly below $\mathsf{pop}_2(\hat{s})$. But such a link cannot be constructed from s by application of push operations because a push applied to a stack of width $|\hat{s}|$ cannot generate an element that points below $\mathsf{pop}_2(\hat{s})$.

   Note that these conditions are MSO-definable on $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$.

Thus, the pairs of configurations $(c, \hat{c})$ where the stack of $\hat{c}$ can be generated from the stack of c by a sequence of push operations form a regular set via Enc. Let S denote the set of these pairs of configurations. Next, we show that there is an MSO formula ψ defining the relation C from Definition 3.1.32 with respect to S.

We only present the proof for configurations of the second form. The proof for the first case is analogous, just replace $f0$ by $d0$.

Recall that $n = |\mathsf{top}_2(\hat{s})| - |\mathsf{top}_2(s)|$. By definition of the encoding, there are nodes

$$f0 = g_0 < g_1 < g_2 < \cdots < g_{n-1} < g_n \leq \hat{d}$$

(uniquely determined) such that $g_i \in \{0,1\}^*0$ for all $1 \leq i \leq n$.

Let w_i be the topmost word of $\mathsf{LStck}(g_i, \mathsf{Enc}(\hat{c}))$. We obtain a chain

$$\mathsf{top}_2(s) =: w_0 < w_1 < w_2 < w_3 < \cdots < w_n = \mathsf{top}_2(\hat{s})$$

where w_{i+1} extends w_i by exactly one letter. Thus,

$$\exists \mathsf{HLoops}(\mathsf{pop}_1^m(\hat{s})) = \exists \mathsf{HLoops}(\mathsf{LStck}(g_{n-m}, \mathsf{Enc}(\hat{c}))) \text{ for all } 0 \leq m \leq n.$$

Since $\exists \mathsf{HLoops}(\mathsf{LStck}(g_{n-m}, \mathsf{Enc}(\hat{c})))$ is MSO-definable at g_{n-m} in $\mathsf{Enc}(\hat{c})$, we can MSO-definably access the pairs of initial and final states of high loops of all $\mathsf{pop}_1^m(\hat{s})$.

Since we are looking for runs from c to $\hat{c}$ that do not visit a proper substack of the stack of c, these consist of a sequence of high loops and push-operations.

Since the set of the g_i, $0 \leq i \leq n$, is MSO-definable and since their order is also MSO-definable, there is a formula which is satisfied by $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$ if and only if there is such a run from c to $\hat{c}$ not passing a substack of $\mathsf{pop}_2(c)$: given a function f that labels g_i with a state q_i, a formula can assert that there is a high loop followed by a push operation from $(q_{n-m}, \mathsf{pop}_1^m(\hat{c}))$ to $(q_{n-(m-1)}, \mathsf{pop}_1^{m-1}(c))$ for each $0 \leq m < n$ such that the following holds:

1. $q_0 = q$, i.e., q_0 is the state of c and
2. there is a high loop from $(q_n, \hat{s})$ to $\hat{c} = (\hat{q}, \hat{s})$.

The function f can be encoded by $|Q|$ many sets. Thus, there is a formula ψ which asserts that there is a function f which satisfies the conditions mentioned above.

For all configurations $c, \hat{c}$ such that the stack of $\hat{c}$ can be created from c by a sequence of push transitions, $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c})$ satisfies ψ if and only if there is a run from c to $\hat{c}$ not passing a proper substack of c.

Since we have already seen that the set S of pairs $(c, \hat{c})$ such that $\hat{c}$ can be created from c by a sequence of push transitions is also MSO-definable, we conclude that the Relation C is regular via Enc. □

Regularity of the Relation **D**

Recall that the relation D contains a pair $(c, \hat{c})$ of configurations for $c = (q, s)$ and $\hat{c} = (\hat{q}, \hat{s})$ if and only if the following holds: $s = \mathsf{pop}_2^m(\hat{s})$ and there is a run from c to $\hat{c}$ not passing any substack of s after its initial configuration.

Recall that $s = \mathsf{pop}_2^n(\hat{s})$ implies that s is a milestone of $\hat{s}$. Hence, the existence of a run from c to $\hat{c}$ can be checked in a similar manner as the existence of a run from the initial configuration to $\hat{c}$. Any run of the latter form passes s. If it passes s in state q this is a witness for $(c, \hat{c}) \in D$.

Lemma 3.1.55. D is regular via Enc.

Proof. Fix a collapsible pushdown system $\mathscr{S}$. The set

$$S := \{\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c}) : c = (q, s), \hat{c} = (\hat{q}, \hat{s}) \in \mathsf{CPG}(\mathscr{S}) \text{ and } s = \mathsf{pop}_2^n(\hat{s}) \text{ for some } n \in \mathbb{N}\}$$

is regular (cf. the proof of 3.1.51).

Let $c, \hat{c}$ be configurations such that $(c, \hat{c}) \in S$. We write $c = (q, s)$ and $\hat{c} = (\hat{q}, \hat{s})$. Furthermore, since $c \in \mathsf{CPG}(\mathscr{S})$, there is a run ρ_0 from $(q_0, \bot)$ to c.

There is a run ρ from c to $\hat{c}$ if and only if there is a run $\hat{\rho} := \rho_0 \circ \rho$ from $(q_0, \bot_2)$ to $\hat{c}$ passing c.

From the previous section, we know that there is a certificate for reachability $C_{\hat{\rho}}$ induced by $\hat{\rho}$ which labels each node $e \in \mathsf{Enc}(\hat{c})$ by the last state in which $\hat{\rho}$ passes $\mathsf{LStck}(e, \mathsf{Enc}(\hat{c}))$.

Since s is a milestone of $\hat{s}$, the rightmost leaf d of $\mathsf{Enc}(c)$ is a node in $\mathsf{Enc}(\hat{c})$ such that $s = \mathsf{LStck}(d, \mathsf{Enc}(\hat{c}))$.

If ρ does not visit any substack of c after its initial configuration, then $C_{\hat{\rho}}(d) = q$. On the other hand, if $C_{\hat{\rho}}(d) = q$ then there is a run $\hat{\rho}$ and some $i \in \mathsf{dom}(\hat{\rho})$ such that $\hat{\rho}(i) = (q, s) = c$, $\hat{\rho}$ ends in $\hat{c}$ and after i no substack of s is visited. Thus, $\hat{\rho}\restriction_{[i+1,\ln(\hat{\rho})]}$ witnesses $(c, \hat{c}) \in D$.

Since d is MSO-definable, there is a formula ψ such that $\mathsf{Enc}(c) \otimes \mathsf{Enc}(\hat{c}) \models \psi$ for some $(c, \hat{c}) \in S$, if and only if there is a certificate $C_{\hat{\rho}}$ for $\hat{c}$ such that $C_{\hat{\rho}}(d) = q$. This means that ψ defines D relatively to S. Since S is regular, we conclude that D is also regular. □

Regularity of **Reach**

As already indicated, the regularity of A, B, C, and D directly implies the regularity of REACH. We obtain the following corollary.

Corollary 3.1.56. Let $\mathcal{S}$ be a collapsible pushdown system of level 2. The expansion of the graph of $\mathcal{S}$ by the reachability predicate is automatic, i.e., the graph $(\mathsf{CPG}(\mathcal{S}), \mathsf{REACH})$ is automatic. Thus, the $\mathsf{FO}(\mathsf{REACH})$-theory of $\mathsf{CPG}(\mathcal{S})$ is decidable.

Regularity of $\mathbf{Reach_L}$

In the previous section, we proved that the reachability predicate REACH on collapsible pushdown graphs is automatic via Enc. We improve this result and show that reachability by a path that satisfies a regular expression is automatic.

Recall that for $L \subseteq \Gamma^*$ some regular language, REACH_L is the binary relation that contains configurations $(c, \hat{c})$ if and only if there is a run ρ from c to $\hat{c}$ such that the labels of the transitions used in ρ form a word w such that $w \in L$.

Let L be some regular language. We show that REACH_L is automatic via Enc by constructing a version of the product of $\mathcal{S}$ with the automaton $\mathcal{A}_L$ corresponding to L. We show that $\mathsf{CPG}(\mathcal{S})$ is first-order interpretable in this product and we show that the predicate REACH_L on $\mathsf{CPG}(\mathcal{S})$ can be expressed via REACH on this product.

Before we state the lemma, we introduce some abbreviations. For x a variable and q a state of some collapsible pushdown system $\mathcal{S}$, we write $x \in q$ for the FO formula stating that x is a configuration with state q. This is definable because we assume that the label of an incoming transition encodes the state of the node. Furthermore, all configurations but the initial one have at least one incoming edge. Since the set Q of states is finite, we also write $x \in Q'$ where $Q' \subseteq Q$ for the formula $\bigvee_{q \in Q'} x \in q$.

Lemma 3.1.57. Let $\mathcal{S} = (Q, \Sigma, \Gamma, q_i, \Delta)$ be a 2-CPS. Furthermore, let $L_1, L_2, \ldots, L_n \subseteq \Gamma^*$ be regular languages. Then $(\mathsf{CPG}(\mathcal{S}), (\mathsf{REACH}_{L_i})_{1 \leq i \leq n})$ is automatic via Enc.

Proof. Without loss of generality, we assume that $n = 1$ and write L for L_1. The general case is proved by iterating the following construction. Let $\mathcal{A}_L = (F, \Gamma, f_i, f_f, \Delta_L)$ be the finite string-automaton corresponding to L.

We define the product of $\mathcal{S}$ and $\mathcal{A}_L$ to be the collapsible pushdown system

$$\bar{\mathcal{S}} = (\bar{Q}, \Sigma, \Gamma \cup \{\varepsilon_i, \varepsilon_f\}, q_i, \bar{\Delta}) \text{ where}$$

- $\bar{Q} := Q \cup (Q \times F)$ and

- $\bar{\Delta}$ is the union

$$\begin{aligned}&\Delta\\ &\cup\{(q,\sigma,\varepsilon_i,(q,f_i),\mathrm{id}) : \sigma\in\Sigma, q\in Q\}\\ &\cup\{(q,\sigma,\varepsilon_f,(q,f_f),\mathrm{id}) : \sigma\in\Sigma, q\in Q\}\\ &\cup\{((q,f),\sigma,\gamma,(q',f'),\mathrm{op}) : (q,\sigma,\gamma,q',\mathrm{op})\in\Delta \text{ and } (f,\gamma,f')\in\Delta_L\}.\end{aligned}$$

Note that $\mathsf{CPG}(\mathscr{S})$ is FO definable in $\mathsf{CPG}(\bar{\mathscr{S}})$: both graphs have the same initial configuration and $\bar{\mathscr{S}}$ extends $\mathscr{S}$ only by transitions that lead to configurations with states in $Q\times F$. Hence, the restriction of $\mathsf{CPG}(\bar{\mathscr{S}})$ to the set $\{x\in\mathsf{CPG}(\bar{\mathscr{S}}) : x\in Q\}$ is isomorphic to $\mathsf{CPG}(\mathscr{S})$.

On the other hand, by construction there is a path from $((q,f_i),s)$ to $((q',f_f),s')$ in $\mathsf{CPG}(\bar{\mathscr{S}})$ if and only if there is a path from (q,s) to (q',s') in $\mathsf{CPG}(\bar{\mathscr{S}})$ whose path corresponds to an accepting word of $\mathscr{A}_L$. Hence, $(x,y)\in\mathsf{REACH}_L$ on $\mathsf{CPG}(\mathscr{S})$ corresponds to

$$\exists x'\exists y'\big(x\vdash^{\varepsilon_i} x'\wedge y\vdash^{\varepsilon_f} y'\wedge(x',y')\in\mathsf{REACH}\big)$$

on $\mathsf{CPG}(\bar{\mathscr{S}})$. The closure of automaticity under first-order interpretations yields the desired result. □

3.1.5 Combination of FO and $L\mu$ Model Checking

We have obtained an FO model checking algorithm for collapsible pushdown graphs of level two. Recall that Hague et al. [27] have shown that there is an $L\mu$ model checking algorithm for the class of all collapsible pushdown graphs. It is a natural question whether these two results can be combined. In order to give an answer to this question, we investigate the following three questions.

1. Let $\mathscr{C}_{L\mu}$ be the class of graphs obtained by $L\mu$-interpretation from the class of level 2 collapsible pushdown graphs. Is the FO model checking problem on $\mathscr{C}_{L\mu}$ decidable?

2. Let $\mathscr{C}_{\mathsf{FO}}$ be the class of graphs obtained by FO-interpretation from the class of level 2 collapsible pushdown graphs. Is the $L\mu$ model checking problem on $\mathscr{C}_{\mathsf{FO}}$ decidable?

3. Is MLFP[4] model checking decidable on level 2 collapsible pushdown graphs?

Due to a recent result of Broadbent et al. [13], the first question can be answered positively. They proved the following result:

Theorem 3.1.58 ([13]). The global $L\mu$ model checking for collapsible pushdown graphs is decidable.[5]

[4] Monadic least fixpoint logic (MLFP) is the smallest logic encompassing the expressive power of $L\mu$ and FO that has sensible closure properties.

[5] The global model checking problem asks the following: given a formula $\varphi\in L\mu$ and a graph $\mathfrak{G}$, what are the nodes of $\mathfrak{G}$ that satisfy φ, i.e., what is the set $\{g\in\mathfrak{G} : \mathfrak{G}, g\models\varphi\}$?

For the proof of this theorem, Broadbent et al. introduced an encoding of a collapsible pushdown stack as a word with back-edges. A word with back-edges looks similar to a nested word, but the back-edges are not well nested. Via this encoding, $L\mu$ definable sets of stacks are turned into sets of words with back-edges that are recognised by deterministic automata on such words with back-edges. These automata work like finite automata on ordinary words, but they propagate the state at a position in the word to the next position and to those positions reachable via a back-edge. Broadbent et al. provide a construction of an automaton on words with back-edges that corresponds to a given formula $\varphi \in L\mu$. Using this construction one can then decide the global model checking problem.

For collapsible pushdown graphs of level two, their techniques imply that $L\mu$-definable subsets are automatic via Enc as follows. Let S be a set of level 2 stacks. If the set of words with back-edges encoding S is regular in the sense of Broadbent et al., then S is automatic via Enc.

Hence, from the results of Broadbent et al. the next corollary follows immediately.

Corollary 3.1.59 ([13]). The $L\mu$-definable subsets in a 2-CPG are transformed into regular sets of trees by the encoding function Enc.

From the decidability of model checking on automatic structures, we directly conclude that first-order logic on collapsible pushdown graphs expanded by $L\mu$-definable predicates is decidable.

Corollary 3.1.60. The graph of a collapsible pushdown system of level 2 enriched by $L\mu$-definable predicates is automatic. Hence, its $\mathsf{FO}(\mathrm{Reg}, (\mathsf{Ram}^n)_{n\in\mathbb{N}}, (\exists^{k,m})_{k,m\in\mathbb{N}})$-theory is decidable.

Of course, this result is compatible with Lemma 3.1.57. Thus, Theorem 3.0.1 follows directly from these two results.

After the positive answer to our first question, we give negative answers to the other two questions. We show the undecidability of the $L\mu$ model checking on graphs obtained by first-order interpretations from collapsible pushdown graphs of level 2.

This negative answer to our second question implies also a negative answer to the third: MLFP encompasses FO and $L\mu$ whence for each FO-interpretation I and each $L\mu$-formula φ there is a MLFP formula ψ such that $\mathfrak{A} \models \psi$ if and only if $I_{\mathrm{Str}}(\mathfrak{A}) \models \varphi$ for all structures $\mathfrak{A}$. Since we show the undecidability of the second problem, the first one is also undecidable.

Recall that Lemma 2.2.2 shows the undecidability of the $L\mu$ model checking on the bidirectional half-grid (recall Figure 2.1). Due to this result, the following lemma implies that $L\mu$ model checking is undecidable on $\mathscr{C}_{\mathrm{FO}}$.

Lemma 3.1.61. The bidirectional half-grid $\mathfrak{H}$ is FO interpretable in a certain CPG of level 2.

Proof. Extending the idea for the MSO-undecidability result of Hague et al. [27], we consider the following collapsible pushdown graph.

Let $Q := \{0,1,2\}, \Sigma := \{\bot, a\}$, and Δ is given by $(0,-,\mathsf{Cl}_1,1,\mathsf{clone}_2)$, $(1,-,\mathsf{A},0,\mathsf{push}_{a,2})$, $(0,-,\mathsf{Cl}_2,2,\mathsf{clone}_2)$, $(2,a,\mathsf{P}_1,2,\mathsf{pop}_1)$, $(2,a,\mathsf{Co},0,\mathsf{collapse})$, and $(2,a,\mathsf{P}_2,0,\mathsf{pop}_2)$ where "$-$" denotes any letter from Σ. We call this example graph $\mathfrak{G}$ (cf. Figure 3.15).

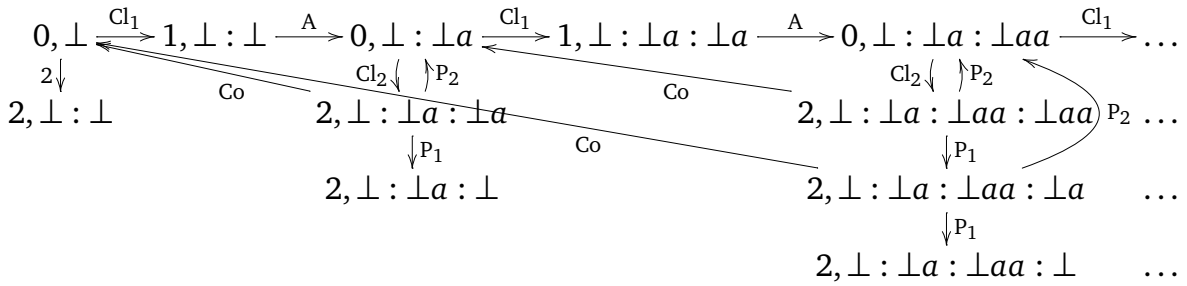


Figure 3.15.: The collapsible pushdown graph $\mathfrak{G}$.

In order to interpret $\mathfrak{H} = (H, \to, \downarrow, \leftarrow, \uparrow)$ in $\mathfrak{G}$, we first have to define the domain of this interpretation. Let

$$\varphi(x) := \exists y \; x \vdash^{P_2} y.$$

This formula defines all elements that are not in the first row of $\mathfrak{G}$ and which have a pop_2 and a collapse successor. Set

$$\varphi_{nd}(x, y) := \exists z \exists z' \left(x \vdash^{Co} z \wedge y \vdash^{P_1} z' \wedge z' \vdash^{Co} z \right).$$

φ_{nd} defines the relation "y is on the diagonal to the right of the diagonal of x". Set

$$\varphi_{nc}(x, y) := \exists z \exists z' \exists z'' \left(z \vdash^{Cl_1} z' \vdash^{A} z'' \wedge x \vdash^{P_2} z \wedge y \vdash^{P_2} z'' \right).$$

This formula defines the relation "y is on the column to the right of the column of x".

Now, y is the right neighbour of x if and only if

$$\varphi_{\to} := \varphi_{nd}(x, y) \wedge \varphi_{nc}(x, y)$$

holds.

Hence, the FO-interpretation $I := (\varphi, \varphi_{\to}, \vdash^{P_1}, \varphi_{\to}^{-1}, (\vdash^{P_1})^{-1})$ yields $\mathfrak{H} = \mathsf{Str}_I(\mathfrak{G})$. □

In the following, we summarise observations concerning the optimality of our result. A first question is whether the complexity of the FO model checking algorithm may be improved. As we mentioned in Chapter 2.3, using automatic representations for model checking purposes leads to nonelementary complexity of the model checking algorithm. In the first part of this section we show a matching lower bound: any FO model checking algorithm has nonelementary complexity. Then we briefly discuss a negative result concerning model checking on higher levels of the collapsible pushdown hierarchy: Broadbent [12] has shown that FO is undecidable on the third level of the collapsible pushdown hierarchy.

3.1.6 Lower Bound for FO Model Checking

Recall that FO model checking on automatic structures has nonelementary complexity. In the following theorem we show that there is no elementary algorithm for FO model checking on collapsible pushdown graphs. The proof is by reduction to the nonemptyness problem for star-free regular expressions. As an auxiliary step, we prove that FO model checking on the full infinite binary tree is nonelementary.

Lemma 3.1.62. The expression complexity of FO model checking on the full infinite binary tree $\mathfrak{T} := (T, \prec, S_1, S_2)$ with prefix order $\prec$ and successor relations S_1, S_2 is nonelementary.

Proof. For each first-order sentence φ, there is a first-order sentence $\varphi'(x)$ such that for all $t \in \mathfrak{T}$, $\mathfrak{T} \models \varphi'(t)$ if and only if $\mathfrak{T}\restriction_{\{t' : t' \prec t\}} \models \varphi$. Note that $\mathfrak{T}\restriction_{\{t' : t' \prec t\}}$ can be considered as a finite word structure over the alphabet $\{1, 2\}$: We identify an incoming S_1 edge with the label 1 and an incoming S_2 edge with the label 2.

In this sense, the model checking problem for the formula $\exists x \varphi'(x)$ on $\mathfrak{T}$ is equivalent to the satisfiability problem for φ with respect to the class of word-structures. Via the classical result of McNaughton and Papert [51] this problem is equivalent to the nonemptyness problem for languages defined by star-free regular expressions. Since the latter problem has nonelementary complexity [58], the claim follows. □

Now, we present a reduction of the FO model checking on the full infinite binary tree to the FO model checking on collapsible pushdown graphs.

Theorem 3.1.63. The expression complexity of any FO model checking algorithm for level 2 collapsible pushdown graphs is nonelementary.

Proof. For the proof of this theorem we modify the graph of example 3.15. Note that $(\omega, <)$ is first order definable in this graph: restrict the domain to all elements with state 0. The order $<$ is then defined via $\varphi_<(x, y) := \exists z \; z \vdash^{\mathrm{pop}_2} y \wedge z \vdash^{\mathrm{collapse}} x$.

In order to obtain a binary tree from a collapsible pushdown graph we create an infinite tree-like graph where every branch is a copy of the graph from example 3.15. The copies are ordered in such a way that the first-order interpretation from above yields the full binary tree when applied to this graph.

To this end, we duplicate the letter a and the label A. We introduce a new letter a' and a new label A'. Furthermore, for each transition where a occurs, we add the corresponding transition where a is replaced by a' as follows: we add the transitions $(1, -, A', 0, \mathrm{push}_{a',2})$, $(2, a', P_1, 2, \mathrm{pop}_1)$, $(2, a', P_2, 0, \mathrm{pop}_2)$, and $(2, a', \mathrm{Co}, 0, \mathrm{collapse})$ where A' is a new edge-label.

On the resulting graph restricted to the configurations with states 0, the formula $\varphi_<(x, y)$ from above defines the prefix order of the full infinite binary tree. Furthermore, the formulas $\varphi_L(x, y) := \exists z \; x \vdash^{\mathrm{Cl}} z \vdash^{A} y$ and $\varphi_R(x, y) := \exists z \; x \vdash^{\mathrm{Cl}} z \vdash^{A'} y$ define the left successor, respectively, the right successor relation.

Lemma 3.1.62 implies the desired result. □

3.1.7 Model Checking on Higher-Order Collapsible Pushdown Graphs

Recently, Broadbent [12] developed a reduction of Post's correspondence problem (PCP,cf. [54]) to FO model checking on collapsible pushdown graphs of level 3. Since the PCP is undecidable, it follows that the FO model checking problem on level 3 collapsible pushdown graphs is undecidable.

In fact, Broadbent's proof comes in two variants: firstly, there is a fixed level 3 collapsible pushdown graph with undecidable FO-theory. On this fixed graph, there is a first-order formula for each instance of the PCP with the following property. The graph satisfies this formula if and only if the corresponding instance of the PCP has a solution. Secondly, Broadbent provides a fixed formula $\varphi \in \mathrm{FO}$ such that there is a class $\mathscr{C}$ of level 3 collapsible

pushdown graphs such that the following holds. For each instance of the PCP there is a graph $\mathfrak{G} \in \mathscr{C}$ such that $\mathfrak{G} \models \varphi$ if and only if this instance of the PCP has a solution.

Thus, FO model checking on level 3 collapsible pushdown graphs is undecidable even for either fixed structure or fixed formula.

3.2 An FO Model Checking Algorithm on Nested Pushdown Trees

This section analyses the FO model checking problem on the class of nested pushdown trees. In the first part we reduce the FO(REACH) model checking problem to the FO(Reg) model checking problem for level 2 collapsible pushdown automata. We show that there is a first-order interpretation I such that for each nested pushdown tree $\mathfrak{N}$ there is a collapsible pushdown graph $\mathfrak{G}$ of level 2 such that $\mathsf{Str}_I(\mathfrak{G}) = \mathfrak{N}$. Furthermore, I transfers the reachability predicate on nested pushdown trees into a certain regular reachability predicate on the collapsible pushdown graph.

In Sections 3.2.2–3.2.4 we have a closer look at the complexity of FO model checking on nested pushdown trees. We develop several versions of the pumping lemma for pushdown systems which are compatible with the jump edges in the following sense: application of these lemmas to a run yields a short run with equivalent first-order type. The bounds obtained by this lemma can be used as a constraint for Duplicator's strategy in the Ehrenfeucht-Fraïssé game on two identical copies of some nested pushdown tree. As indicated in Section 2.1.1, this result can be turned into a model checking algorithm. Using this approach, we show that the complexity of FO model checking on nested pushdown trees is in 2-EXPSPACE.

3.2.1 Interpretation of NPT in CPG

In this section, we show that any nested pushdown tree can be first-order interpreted in some collapsible pushdown graph of level 2. For this purpose, we fix a pushdown system $\mathscr{N} = (Q, \Sigma, \Gamma, \Delta, q_0)$. We show that there is a collapsible pushdown system of level 2 and a first-order interpretation that yields the nested pushdown tree generated by the pushdown system.

The basic idea is the following: every vertex of the nested pushdown tree generated by $\mathscr{N}$ is a run, i.e., a list of configurations that are passed by this run. Every configuration is a level 1-stack s and a state q. We write the state q on top of the stack s and obtain the stack $\mathsf{push}_q(s)$. Then we represent a run $(q_1, s_1) \vdash (q_2, s_2) \vdash \ldots \vdash (q_n, s_n)$ by the stack $\mathsf{push}_{q_1}(s_1) : \mathsf{push}_{q_2}(s_2) : \cdots : \mathsf{push}_{q_n}(s_n)$. Using this encoding, we can simulate every transition of the pushdown system by at most four stack operations of the collapsible pushdown system and the nesting edges can be simulated by reverse collapse edges.

The following definition provides the details of this simulation.

Definition 3.2.1. Let $\mathscr{N} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a pushdown system generating $\mathsf{NPT}(\mathscr{N})$. We define a corresponding collapsible pushdown system

$$C(\mathscr{N}) := (Q_C, \Sigma_C, \Gamma_C, \Delta_C, PUSH(q_0))$$

of level 2 as follows:

- $\Sigma_C := Q \cup \Sigma$.

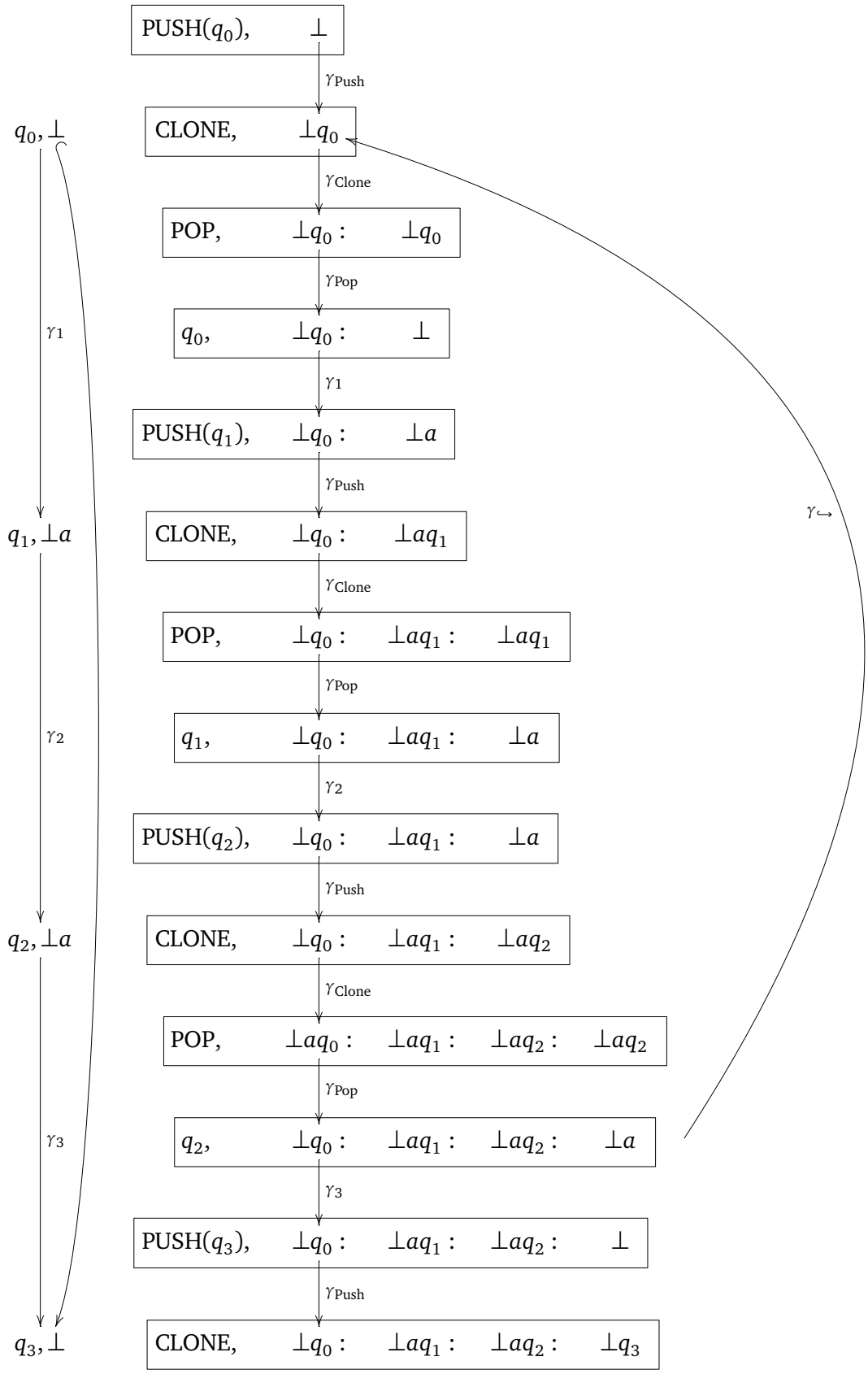


Figure 3.16.: Simulation of a nested pushdown tree in a collapsible pushdown graph of level 2.

- $\Gamma_C := \Gamma \cup \{\gamma_{\hookrightarrow}, \gamma_{\mathrm{Clone}}, \gamma_{\mathrm{Pop}}, \gamma_{\mathrm{Push}}\}$ for $\gamma_{\hookrightarrow}, \gamma_{\mathrm{Clone}}, \gamma_{\mathrm{Pop}}, \gamma_{\mathrm{Push}}$ new symbols not contained in Γ.
- $Q_C := \{\mathrm{POP}, \mathrm{CLONE}\} \cup Q \cup \{\mathrm{PUSH}(q) : q \in Q\}$, where POP and CLONE, and $\mathrm{PUSH}(q)$ are new auxiliary states used to perform exactly the stack operation indicated by the name.
- Δ_C consists of the following transitions:
  - For $q \in Q$ and $\sigma \in \Sigma$, let

$$(\mathrm{PUSH}(q), \sigma, \gamma_{\mathrm{Push}}, \mathrm{CLONE}, \mathrm{push}_{q,1}),$$
$$(\mathrm{CLONE}, q, \gamma_{\mathrm{Clone}}, \mathrm{POP}, \mathrm{clone}_2), \text{ and}$$
$$(\mathrm{POP}, q, \gamma_{\mathrm{Pop}}, q, \mathrm{pop}_1)$$

    be in Δ_C. These transitions are auxiliary transitions that write the state of the run onto the topmost word and create a clone of the topmost word preparing the simulation of the next transition.
  - For $(q, \sigma, \gamma, p, \mathrm{id}) \in \Delta$, set $(q, \sigma, \gamma, \mathrm{PUSH}(p), \mathrm{id}) \in \Delta_C$.
  - For $(q, \sigma, \gamma, p, \mathrm{push}_\tau) \in \Delta$ add $(q, \sigma, \gamma, \mathrm{PUSH}(p), \mathrm{push}_{\tau,2}) \in \Delta_C$.
  - For $(q, \sigma, \gamma, p, \mathrm{pop}_1) \in \Delta$, set $(q, \sigma, \gamma, \mathrm{PUSH(p)}, \mathrm{pop}_1) \in \Delta_C$. This transition simulates the pop_1 transition. Moreover, whenever a pop_1 occurs, we also have to simulate the jump-edge. For this purpose, we set $(q, \sigma, \gamma_{\hookrightarrow}, \mathrm{CLONE}, \mathrm{collapse}) \in \Delta_C$.

Figure 3.16 shows a path in a nested pushdown tree generated by a pushdown system $\mathcal{N}$ and the corresponding path in $C(\mathcal{N})$. The following lemma shows that the original nested pushdown tree is first-order definable in the graph generated by $C(\mathcal{N})$.

Lemma 3.2.2. If $\mathcal{N}$ is a pushdown system that generates a nested pushdown tree $\mathrm{NPT}(\mathcal{N})$, then $\mathrm{NPT}(\mathcal{N})$ is FO_3-interpretable in $\mathrm{CPG}(C(\mathcal{N}))$.

Proof. First of all, note that $C(\mathcal{N})$ is deterministic whenever it is in one of the states $\{\mathrm{POP}, \mathrm{CLONE}\} \cup \{\mathrm{PUSH}(q) : q \in Q\}$.

For all $q \in Q, w \in \Sigma^*$ and s a stack, we say that $(\mathrm{CLONE}, s) \in \mathrm{CPG}(C(\mathcal{N}))$ represents a run to (q, w) of $\mathcal{N}$ if $\mathrm{top}_2(s) = wq$ (in this equality we forget about the links stored in s, of course).

The following holds for all configurations (CLONE, s) that represent some run to some configuration (q, w).

- $(q, \sigma, p, id) \in \Delta$ iff there is a path from (CLONE, s) to $(\mathrm{CLONE}, s : \bar{w})$ for $\bar{w}$ a word such that $(\sigma, s : \bar{w})$ represents a run to (p, w). If such a path exists, it consists of the operations $\mathrm{clone}_2; \mathrm{pop}_1; \mathrm{id}; \mathrm{push}_p$.
- $(q, \sigma, p, \mathrm{push}_\tau) \in \Delta$ iff there is a path from (σ, s) to $(\sigma, s : \bar{w})$ for $\bar{w}$ a word such that $(\sigma, s : \bar{w})$ represents a run to $(p, w\tau)$. If such a path exists, it consists of the operations $\mathrm{clone}_2; \mathrm{pop}_1; \mathrm{push}_{(\tau,2)}; \mathrm{push}_p$. Furthermore note, that τ has a link to the stack s.

- $(q, \sigma, p, \mathsf{pop}_1) \in \Delta$ iff there is a path from (σ, s) to $(\sigma, s : \bar{w})$ for $\bar{w}$ a word such that $(\sigma, s : \bar{w})$ represents a run to $(p, \mathsf{pop}_1(w))$. If such a path exists, it consists of the operations $\mathsf{clone}_2; \mathsf{pop}_1; \mathsf{pop}_1; \mathsf{push}_p$.

From these observations, an easy induction shows that there is a bijection from the domain of $\mathsf{NPT}(\mathcal{N})$ to those configurations of $\mathsf{CPG}(C(\mathcal{N}))$ which are in state CLONE. Furthermore, the transition relation of $\mathsf{NPT}(\mathcal{N})$ is FO_3-definable on this subset of $\mathsf{CPG}(C(\mathcal{N}))$.

Finally, we have to show the FO_3-definability of the jump-edges of $\mathsf{NPT}(\mathcal{N})$ in $\mathsf{CPG}(C(\mathcal{N}))$. For this purpose, note that a push_τ-transition in $\mathsf{NPT}(\mathcal{N})$ corresponds to a $\mathsf{push}_{\tau,2}$-transition in the collapsible pushdown graph. From the analysis of the existence of push_τ-transitions in $\mathcal{N}$, we obtain directly that this $(\tau, 2)$ has a pointer to the configuration representing the run to the configuration precisely before this push_τ-transition is simulated. When we later simulate a pop_1-transition of $\mathsf{NPT}(\mathcal{N})$ that corresponds to this push_τ-transition, then we remove one of the clones of the corresponding $(\tau, 2)$ from the stack. From this, one easily sees that if (CLONE, s) represents a run to some configuration (q, w) such that the last operation of this run was a pop_1, then the prefix of the run up to the step before the corresponding $\mathsf{push}_{\tau,2}$-transition is encoded in the unique configuration (CLONE, s') such that there are configurations c, d such that $c \vdash^{\gamma\hookrightarrow} (\mathsf{CLONE}, s')$ and $c \vdash^{\gamma} d \vdash^{\gamma_{\mathsf{Push}}} (\mathsf{CLONE}, s)$. It is also easy to see that all configurations that satisfy this condition correspond to positions that simulate corresponding push_τ and pop_1-transitions. Hence, the jump-edges are actually FO_2-definable in $\mathsf{CPG}(C(\mathcal{N}))$. □

Corollary 3.2.3. The FO model checking of nested pushdown trees is decidable.

A closer look at the pushdown system $C(\mathcal{N})$ even gives a better result: $\mathsf{FO}(\mathsf{REACH})$ model checking on nested pushdown trees is decidable. First of all observe that reachability in a nested pushdown tree $\mathsf{NPT}(\mathcal{N})$ coincides with reachability in $\mathsf{NPT}(\mathcal{N})$ without the use of jump-edges because jump-edges only connect vertices x and y where y is a run extending the run x. But there is a one-to-one correspondence between reachability along the transitions of the pushdown system $\mathcal{N}$ and reachability in $\mathsf{CPG}(C(\mathcal{N}))$ without use of the collapse transitions. This is due to the fact that all transitions in $C(\mathcal{N})$ that do not perform a $\mathsf{collapse}$ are used to simulate at least one of the transitions of $\mathcal{N}$. Hence, the predicate REACH on $\mathsf{NPT}(\mathcal{N})$ reduces to $\mathsf{REACH}_{(\Gamma_C \setminus \{\gamma_\hookrightarrow\})^*}$ on $\mathsf{CPG}(C(\mathcal{N}))$. Thus, we obtain the following extension of the previous corollary.

Theorem 3.2.4. $\mathsf{FO}(\mathsf{REACH})$ model checking on nested pushdown trees is decidable.

Remark 3.2.5. Moreover, $\mathsf{FO}(\mathsf{REACH}_{L_1}, \mathsf{REACH}_{L_2}, \ldots, \mathsf{REACH}_{L_n})$ is decidable on $\mathsf{NPT}(\mathcal{N})$ if the L_i are regular languages over Γ (i.e., not using $\hookrightarrow$). This is due to the fact that each $\gamma \in \Gamma$ has a direct translation into a fixed sequence of labels in the simulating collapsible pushdown graph.

Having shown that nested pushdown trees are first-order interpretable in collapsible pushdown graphs of level 2, the question arises whether the reverse statement also holds. Are collapsible pushdown graphs interpretable in the class of nested pushdown trees? The answer to this question is negative if we restrict our attention to uniform first-order interpretations.

In Lemma 3.1.62, we proved that the first-order model checking on collapsible pushdown graphs of level 2 has nonelementary complexity. In the next section, we present an elementary first-order model checking algorithm for nested pushdown trees. Since first-order

interpretations can be used to transfer the first-order model checking problem, we obtain the following theorem.

Theorem 3.2.6. There is no first-order interpretation I such that for each collapsible pushdown graph $\mathfrak{G}$ of level 2, there is a nested pushdown tree $\mathsf{NPT}(\mathcal{N})$ such that

$$\mathfrak{G} = \mathsf{Str}_I(\mathsf{NPT}(\mathcal{N})).$$

Proof. Heading for a contradiction, assume that such an interpretation I exists. Fix some collapsible pushdown graph $\mathfrak{G}$ such that its first-order model checking has nonelementary expression complexity. Set $\mathfrak{N} := \mathsf{NPT}(\mathcal{N})$ such that $\mathfrak{G} = \mathsf{Str}_I(\mathfrak{N})$. By definition of a first-order interpretation, for each sentence $\varphi \in \mathsf{FO}$ over the vocabulary of $\mathfrak{G}$, there is a formula $\mathsf{Frm}_I(\varphi)$ such that $\mathfrak{G} \models \varphi$ if and only if $\mathfrak{N} \models \mathsf{Frm}_I(\varphi)$. As we will see in the following section, the question "$\mathfrak{N} \models \mathsf{Frm}_I(\varphi)$?" has elementary expression complexity. By definition of I, $\mathsf{Frm}_I(\varphi)$ has length linear in the length of φ which implies that the algorithm has also elementary complexity in the size of φ. But then we obtain an elementary algorithm deciding $\mathfrak{G} \models \varphi$ by just calculating $\mathsf{Frm}_I(\varphi)$ and solving $\mathfrak{N} \models \mathsf{Frm}_I(\varphi)$. This contradicts our assumption on $\mathfrak{G}$. □

Remark 3.2.7. More generally, we can weaken our assumption on the interpretation I. Assume that there is an elementary algorithm that, on input a collapsible pushdown graph $\mathfrak{G}$ of level 2, computes an interpretation I and a pushdown system $\mathcal{N}$ such that $\mathfrak{G} = \mathsf{Str}_I(\mathsf{NPT}(\mathcal{N}))$. Let f be an elementary bound on the running time of this algorithm in terms of the size of the pushdown system and the formula. Then we obtain the following elementary model checking algorithm on the class of collapsible pushdown graphs of level 2. Given $\mathfrak{G}$ and a formula φ, we compute $\mathcal{N}$, I and $\mathsf{Frm}_I(\varphi)$ such that $\mathfrak{G} = \mathsf{Str}_I(\mathsf{NPT}(\mathcal{N}))$ in time $f(|\mathfrak{G}|, |\varphi|)$. Note that $|\mathcal{N}|$ and the size of $\mathsf{Frm}_I(\varphi)$ are bound by $f(|\mathfrak{G}|, |\varphi|)$. Using the model checking algorithm on nested pushdown trees, we can decide whether $\mathsf{NPT}(\mathcal{N}) \models \mathsf{Frm}_I(\varphi)$ in $\mathsf{exp}(\mathsf{exp}(\mathsf{exp}(f(|\mathfrak{G}|, |\varphi|))))$.

This solves the model checking problem on collapsible pushdown graphs in running time three-fold exponential in the elementary function f. This contradicts the result that FO model checking on collapsible pushdown graphs has nonelementary complexity.

We have seen that first-order interpretations cannot be used to define collapsible pushdown graphs in nested pushdown trees. The question remains open whether there is another logical interpretation that allows to interpret all collapsible pushdown graphs in the class of nested pushdown trees. Before one could give a precise answer to this question, we would have to specify what kind of interpretation we would like to consider. Nevertheless, we conjecture that the answer to this question is negative for all meaningful concepts of logical interpretation. We want to point out two facts that make it hard to imagine an interpretation of all collapsible pushdown graphs in nested pushdown trees.

We already mentioned the gap in the complexity of $L\mu$ model checking between the two classes. Recall Theorem 2.3.24 which states that the $L\mu$ model checking problem of level 2 collapsible pushdown graphs is 2-EXPTIME complete. On the other hand, recall that Theorem 2.3.10 states that the $L\mu$ model checking problem for nested pushdown trees is in EXPTIME. This implies that any such interpretation would have to imply an exponential blowup in the size of the nested pushdown tree that is used to interpret some graph or the interpretation cannot preserve $L\mu$ formulas.

The second fact relies on comparison of the unfoldings of collapsible pushdown graphs and nested pushdown trees. Recall that the class of collapsible pushdown graphs of level 2 encompasses also all higher-order pushdown graphs and these graphs are contained in the second level of the Caucal hierarchy. Furthermore, recall that the third level of the Caucal hierarchy is generated by applying graph unfoldings followed by MSO-interpretations to all graphs in the second level. Hence, applying unfoldings followed by MSO-interpretations to the collapsible pushdown graphs of level 2, we generate a class of graphs that contains the third level of the Caucal hierarchy. If we apply the same transformation to nested pushdown trees, we end up in the second level of the Caucal hierarchy due to the following lemma.

Lemma 3.2.8. The unfolding $\mathfrak{U}$ of a nested pushdown tree $\mathfrak{N}$ is the ε-contraction of the unfolding of a pushdown graph. Thus, any MSO-interpretation on $\mathfrak{U}$ yields a graph in the second level of the Caucal hierarchy.

Proof. Recall that a nested pushdown tree $\mathfrak{N}$ is almost unfolded, in the sense that it is a tree except for the jump-edges. Thus, the unfolding of $\mathfrak{N}$ is obtained by the following operation. We remove each jump-edge $\rho \hookrightarrow \pi$ and we append a new copy of the subtree rooted at π to ρ via a $\hookrightarrow$-edge. Due to the definition of $\hookrightarrow$, the stacks in the last configuration of ρ and π agree and the run from ρ to π does only "see" the topmost element of this stack. Hence, generating the unfolding boils down to the generation of the right number of copies of the configuration (q_2, s) for each run $\rho \in \mathfrak{N}$ ending in (q_1, s) and to attaching the subtrees induced by this configuration via $\hookrightarrow$ to ρ. As we already mentioned, the number of outgoing jump-edges from ρ to some position with state q_2 only depends on the topmost symbol of ρ and the pair (q_1, q_2). Using new states and ε-contraction, we can easily design a pushdown system $\mathscr{S}$ that behaves as the one generating $\mathfrak{N}$, but which furthermore generates the right number of copies of π at each configuration (by writing and removing nondeterministically sufficiently many dummy symbols onto/from the stack). □

We next show that first-order model checking on nested pushdown trees has elementary complexity. More precisely, we present an algorithm that uses doubly exponential space in the size of the pushdown system and the size of the formula. For this purpose, we first investigate variants of the pumping lemma for pushdown systems that are compatible with nested pushdown trees in the following sense. Application of the pumping lemma to some run yields a shorter run such that both runs share the same first-order theory up to a certain quantifier rank. In Section 3.2.4 we apply these lemmas in order to derive a dynamic small-witness property for nested pushdown trees. This means that for any existential quantification that is satisfied by some nested pushdown tree, there is a short run witnessing this quantification. As explained in Section 2.1.1, this property gives rise to a model checking algorithm. We prove that this algorithm is in 2-EXPSPACE.

3.2.2 A Modularity Result for Games on Graphs of Small Diameter

We prepare the pumping lemmas mentioned above by a general result on Ehrenfeucht-Fraïssé games on certain graphs. We show that certain tuples of a given graph have the same $\simeq_\rho$-type. This argument forms the back-bone of the modification of the pumping lemma (Lemma 2.3.7) in order to obtain $\simeq_\rho$-preserving pumping lemmas.

Our lemma looks like a Gaifman-locality argument, but it can be used in situations where ordinary locality arguments fail. It uses a locality argument on induced substructures whence it can be applied to certain graphs that have a small diameter. The crucial property of these graphs is that there are some generic edges that make the diameter small in the sense that a lot of vertices are connected to the same vertex, but when these edges are removed the diameter becomes large. Therefore, on the graph obtained by removing these generic edges we can apply Gaifman-like arguments in order to establish partial isomorphisms and $\simeq_\rho$-equivalence. Since disjoint but isomorphic neighbourhoods in such a graph have generic edges to the same vertices (in the full graph), moving a tuple from one neighbourhood to the other does not change the $\simeq_\rho$-type of the tuple.

We use the following notation.

For some structure $\mathfrak{G} = (V, E_1, E_2, \ldots, E_n)$ with binary relations $E_1, E_2, \ldots, E_n$ and sets $A, B \subseteq V$ we say that A and B touch if $A \cap B \neq \emptyset$ or if there are $a \in A$, $b \in B$ such that $(a, b) \in E_i$ or $(b, a) \in E_i$ for some $i \leq n$. For a tuple $\bar{a} \in A$ we define inductively the l-neighbourhood of $\bar{a}$ with respect to A, denoted $A_n(\bar{a})$, by setting $A_0(\bar{a}) := \{a_i \in \bar{a}\}$, and

$$A_{l+1}(\bar{a}) := A_l(\bar{a}) \cup \{b \in A : \text{there are } i \leq n \text{ and } c \in A_l(\bar{a}) \text{ s.t. } (b, c) \in E_i \text{ or } (c, b) \in E_i\}.$$

In terms of Gaifman-neighbourhoods, $A_l(\bar{a})$ is the l-local neighbourhood of $\bar{a}$ with respect to $\mathfrak{G}\restriction_A$.

We say that A and B are isomorphic over $C \subseteq V$ and write $A \simeq_C B$ if there is some isomorphism $\varphi : G\restriction_A \simeq G\restriction_B$ such that for all $a \in A$, all $c \in C$, and all $1 \leq i \leq n$,

$$(a, c) \in E_i \text{ iff } (\varphi(a), c) \in E_i \qquad \text{and} \qquad (c, a) \in E_i \text{ iff } (c, \varphi(a)) \in E_i.$$

Lemma 3.2.9. Let $\mathfrak{G} = (V, E_1, E_2, \ldots, E_n)$ be some structure, $A, B \subseteq V$ not touching and let $\varphi : A \simeq B$ be an isomorphism of the induced subgraphs. Let $\bar{a} \in A$ and $\bar{c} \in C := V \setminus \big(A_{2\rho}(\bar{a}) \cup B_{2\rho}(\varphi(\bar{a}))\big)$. Then

$$\varphi\restriction A_{2\rho-1}(\bar{a}) : A_{2\rho-1}(\bar{a}) \simeq_C B_{2\rho-1}(\varphi(\bar{a})) \qquad \text{implies} \qquad \mathfrak{G}, \bar{a}, \varphi(\bar{a}), \bar{c} \simeq_\rho \mathfrak{G}, \varphi(\bar{a}), \bar{a}, \bar{c}.$$

Proof. If $\rho = 0$, the claim holds trivially: since A and B do not touch, there are no edges between the elements from $\bar{a}$ and $\varphi(\bar{a})$; furthermore φ preserves all edges between $\bar{a}$ and $\bar{c}$.

We prove the lemma by induction on ρ. We consider the first round of the Ehrenfeucht-Fraïssé-game on $\mathfrak{G}, \bar{a}, \varphi(\bar{a}), \bar{c}$ and $\mathfrak{G}, \varphi(\bar{a}), \bar{a}, \bar{c}$. By symmetry, we may assume that Spoiler extends the left-hand side $\bar{a}, \varphi(\bar{a}), \bar{c}$, by some $d \in V$. We present a winning strategy for Duplicator. The general idea is the following.

If Spoiler has chosen an element in $A \cup B$ that is close to $\bar{a}$ or $\varphi(\bar{a})$, then Duplicator responds with applying the isomorphism φ. Otherwise, Duplicator just responds choosing the same element as Spoiler. The details are as follows:

Local case: if $d \in A_{2\rho-1}(\bar{a})$ set $a' := d$ and if $d \in \varphi(A_{2\rho-1}(\bar{a}))$ set $a' := \varphi^{-1}(d)$. We set $\bar{a}' := \bar{a}, a'$.

Since $A_{2\rho-1}(\bar{a}') \subseteq A_{2\rho}(\bar{a})$, we have

$$\bar{c} \in C' := V \setminus \big(A_{2\rho-1}(\bar{a}') \cup \varphi(A_{2\rho-1}(\bar{a}'))\big).$$

By definition, there is some set

$$D \subseteq \big(A \setminus A_{2^{\rho-1}}(\bar{a}')\big) \cup \big(B \setminus B_{2^{\rho-1}}(\varphi(\bar{a}'))\big)$$
such that $C' = C \cup D$.

We claim that there is no edge between any element in D and any element in $A_{2^{\rho-1}-1}(\bar{a}')$. If some $d \in D$ satisfies $d \in A$, then by definition it has distance at least 2 from any $a \in A_{2^{\rho-1}-1}(\bar{a}')$. If $d \in D$ satisfies $d \in B$ then it has distance at least 2 from $a \in A_{2^{\rho-1}-1}(\bar{a}')$ because A and B do not touch.

Analogously, one proves that there is no edge between elements in D and elements in $\varphi(A_{2^{\rho-1}-1}(\bar{a}'))$.

Thus, we conclude that $A_{2^{\rho-1}-1}(\bar{a}') \simeq_{C'} \varphi(A_{2^{\rho-1}-1}(\bar{a}'))$. By induction hypothesis, it follows that

$$\mathfrak{G}, \bar{a}', \varphi(\bar{a}'), \bar{c} \simeq_{\rho-1} \mathfrak{G}, \varphi(\bar{a}'), \bar{a}', \bar{c}.$$

Nonlocal case: otherwise,

$$d \in C' := V \setminus \big(A_{2^{\rho-1}}(\bar{a}) \cup \varphi(A_{2^{\rho-1}}(\bar{a}))\big)$$

and we set $\bar{c}' := \bar{c}, d$.
Similarly to the local case, we conclude that $A_{2^{\rho-1}-1}(\bar{a}) \simeq_{C'} \varphi(A_{2^{\rho-1}-1}(\bar{a}))$ because A and B do not touch and the distance between elements in $A_{2^{\rho-1}-1}(\bar{a})$ and elements in $C' \cap A$ is at least 2. Hence, by induction hypothesis

$$\mathfrak{G}, \bar{a}, \varphi(\bar{a}), \bar{c}' \simeq_{\rho-1} \mathfrak{G}, \varphi(\bar{a}), \bar{a}, \bar{c}'.$$

Thus, this strategy is winning for Duplicator in the ρ-round game. □

3.2.3 $\simeq_\alpha$-Pumping on NPT

Recall that $\simeq_\alpha$ coincides with $\equiv_\alpha$. Thus, it describes equivalence with respect to FO_α formulas. In this section we want to develop a version of the pumping Lemma for pushdown systems (Lemma 2.3.7) that preserves $\simeq_\alpha$-types in the following sense. Given a tuple $\bar{\rho}$ of runs and another run ρ such that ρ is very long compared to the runs of $\bar{\rho}$, then we want to apply the pumping lemma in such a way that the resulting run $\hat{\rho}$ is shorter than ρ and such that $\rho\bar{\rho} \simeq_\alpha \hat{\rho}\bar{\rho}$.

In order to achieve this, we use the game argument developed in the previous section and we make a clever choice in the pumping argument. Let us first explain this choice: we want to apply the pumping lemma to ρ and obtain a shorter run $\hat{\rho}$. We apply the lemma in such a way that ρ and $\hat{\rho}$ share a long prefix and they share a long suffix in the sense that the last n transitions of ρ and $\hat{\rho}$ agree for some large $n \in \mathbb{N}$. Later we specify what long exactly means, but we first want to explain how this enables us to use the general game argument in order to show that $\rho\bar{\rho} \simeq_\alpha \hat{\rho}\bar{\rho}$.

The 2^α-neighbourhood of ρ divides into two parts. The first part, denoted by A_ρ, consists of runs ρ' that are very similar to ρ in the sense that there is a large common prefix of ρ

and ρ'. The other part, denoted by C_ρ, consists of runs that are only reachable from ρ via paths that pass a very small prefix of ρ. Now, the 2^α-neighbourhood of $\hat{\rho}$ is isomorphic to the one of ρ in the following sense.

The elements in A_ρ are reachable from ρ via a path such that every edge of this path only changes a small final part of the runs connected by this edge. Thus, every intermediate step shares a large initial prefix with ρ. Since $\hat{\rho}$ coincides with ρ on the final transitions, the path from ρ to an element in A_ρ can be copied edge by edge. We obtain an element in the neighbourhood of $\hat{\rho}$ that has a large common prefix with $\hat{\rho}$ because each edge that we use only changes a small final part of the runs connected by this edge. Since this argument applies to all runs in A_ρ, we obtain an isomorphic copy $B_{\hat{\rho}}$ in the neighbourhood of $\hat{\rho}$.

Now, we consider an element $\pi \in C_\rho$. Any path from ρ to π starts with an initial part that is contained in A_ρ and then at some point we use a $\hookrightarrow$-edge that connects an element $\pi' \in A_\rho$ with a short prefix π'' of this element. Since all elements in A_ρ share a large common prefix, π'' is a prefix of ρ. Since ρ and $\hat{\rho}$ agree on an initial part, π'' is also a prefix of $\hat{\rho}$. Now, the crucial observation is that we can copy the path from ρ to π' edge by edge to a path from $\hat{\rho}$ to some $\hat{\pi}'$ such that π'' and $\hat{\pi}'$ are connected by an $\hookrightarrow$-edge. Since this argument applies to all elements in C_ρ, one derives that C_ρ is also part of the neighbourhood of $\hat{\rho}$.

Using the game argument from the previous section, the isomorphism between A_ρ and $B_{\hat{\rho}}$ can be used to show that $\rho\bar{\rho} \simeq_\alpha \hat{\rho}\bar{\rho}$.

In fact, we divide this $\simeq_\alpha$-preserving pumping lemma into three steps. The first translates a given run into an equivalent run that ends in a configuration with small stack. The second step translates such a run with small final stack into an equivalent run that only passes small stacks. The last step translates a run that only uses small stacks into an equivalent short run.

Later, we use the $\simeq_\alpha$-preserving pumping argument in order to derive an elementary bound for the complexity of FO model checking on nested pushdown trees.

In the following, we first state the three pumping lemmas that we want to prove in this section. Afterwards, we will present the proof of each of these lemmas.

Before we state the first pumping lemma, we want to recall the necessary notation. Let ρ be some run of a pushdown system ending in configuration $c = (q, w)$ where $q \in Q$ and $w \in \Sigma^*$. Recall that, e.g., we write $\mathsf{pop}_1(c)$ for $\mathsf{pop}_1(w)$ and similarly we write $\mathsf{top}_1(\rho)$ for $\mathsf{top}_1(c) = \mathsf{top}_1(w)$. Since we only consider level 1 pushdown systems, $\mathsf{top}_2(\rho)$ is the final stack of ρ. Recall that $\mathsf{wdt}(\rho) = \mathsf{wdt}(w)$ denotes the width of the stack, i.e., $\mathsf{wdt}(\rho) = |w|$. Now, the first pumping lemma reduces the size of the last configuration of a given run, while preserving its $\simeq_\alpha$-type.

Lemma 3.2.10. Let $\mathfrak{N} := \mathrm{NPT}(\mathcal{N})$ be a nested pushdown tree. Let $\bar{\rho} = \rho_1, \rho_2, \ldots, \rho_m \in \mathfrak{N}$ be runs and $\rho \in \mathfrak{N}$ another run such that

$$\mathsf{wdt}(\rho) > \mathsf{wdt}(\rho_i) + (2 + 2^{\alpha+1})|Q| \cdot |\Sigma| + 2^\alpha + 1 \quad \text{for all } i \leq m.$$

There is a $\hat{\rho} \in \mathfrak{N}$ such that $\mathsf{wdt}(\hat{\rho}) < \mathsf{wdt}(\rho)$ and $\mathfrak{N}, \bar{\rho}, \rho \simeq_\alpha \mathfrak{N}, \bar{\rho}, \hat{\rho}$.

In the second pumping lemma, we want to bound the size of all the stacks occurring in a run. For this purpose, we define the following notation.

Definition 3.2.11. Let $\max(\rho)$ denote the size of the largest stack occurring within ρ, i.e.,

$$\max(\rho) := \max\{\mathrm{wdt}(\rho(i)) : i \in \mathrm{dom}(\rho)\}.$$

The second pumping lemma takes a run ρ and transforms ρ into an equivalent run $\hat{\rho}$ such that $\max(\hat{\rho})$ is bounded in terms of $\mathrm{wdt}(\hat{\rho}) = \mathrm{wdt}(\rho)$.

Lemma 3.2.12. Let $\bar{\rho} = \rho_1, \rho_2, \ldots, \rho_m \in \mathfrak{N}$ and $\rho \in \mathfrak{N}$ such that

$$\begin{aligned}&\max(\rho) > \max(\rho_i) + |Q|^2|\Sigma| + 1 \text{ for all } 1 \leq i \leq m, \text{ and such that}\\ &\max(\rho) > |\mathrm{wdt}(\rho)| + |Q|^2|\Sigma| + 2^\alpha + 1.\end{aligned}$$

Then there is some $\hat{\rho} \in \mathfrak{N}$ such that

1. $\hat{\rho}$ and ρ agree on their final configuration,
2. $\max(\hat{\rho}) < \max(\rho)$, and
3. $\mathfrak{N}, \bar{\rho}, \rho \simeq_\alpha \mathfrak{N}, \bar{\rho}, \hat{\rho}$.

In the third pumping lemma, we want to translate a run ρ into an equivalent run $\hat{\rho}$ such that the length of $\hat{\rho}$ is bounded in terms of $\max(\hat{\rho}) \leq \max(\rho)$. For this purpose we introduce a new measure Ξ for the length of a run. We first define Ξ. Then we present the pumping lemma that transforms a run ρ into an equivalent run $\hat{\rho}$ such that $\Xi(\hat{\rho})$ is bounded in terms of $\max(\hat{\rho}) \leq \max(\rho)$. Afterwards, we show that the length of a run $\hat{\rho}$ is polynomially bounded in $\max(\hat{\rho})$ and $\Xi(\hat{\rho})$.

Definition 3.2.13. Let ρ be a run of length n of some pushdown system. We denote the number of occurrences of a stack w in ρ by $|\rho|_w := \left|\{i \in \mathbb{N} : \exists q\ \rho(i) = (q, w)\}\right|$. By $\mathrm{SR}(w, \rho) := \{\hat{\rho} : \exists i, j\ \hat{\rho} = \rho\restriction_{[i,j]}, w \trianglelefteq \rho\}$ we denote the set of subruns of ρ whose stacks are all prefixed by w. Then we define the maximal number of connected occurrences of some stack to be

$$\begin{aligned}&\Xi(\rho, w) := \max\left\{|\hat{\rho}|_w : \hat{\rho} \in \mathrm{SR}(w, \rho)\right\} \text{ and}\\ &\Xi(\rho) := \max\{\Xi(\rho, w) : w \in \Sigma^*\}.\end{aligned}$$

We first state the third pumping lemma. Then we show that it indeed bounds the length of a run. The lemma is based on the fact that a long run ρ that does not visit large stacks has to visit some configuration a lot of times. We can then safely delete a subrun $\rho\restriction_{[i,j]}$ that connects this configuration with itself. The crucial observation is that this does not change the isomorphism type of the neighbourhood if $\rho\restriction_{[i,j]}$ is approximately the middle part of ρ.

Lemma 3.2.14. Let $\bar{\rho} = \rho_1, \rho_2, \ldots, \rho_n \in \mathfrak{N} := \mathrm{NPT}(\mathcal{N})$ such that there is a $B_\Xi \in \mathbb{N}$ satisfying $\Xi(\rho_i) \leq B_\Xi$ for all $1 \leq i \leq n$. For $\rho \in \mathfrak{N}$, there is some $\hat{\rho} \in \mathfrak{N}$ such that

1. $\max(\hat{\rho}) \leq \max(\rho)$,
2. ρ and $\hat{\rho}$ agree on their final configuration,
3. $\Xi(\hat{\rho}) \leq B_\Xi + (2^{\alpha+1} + 2)|Q| + 2^\alpha + 1$, and

4. $\mathfrak{N}, \bar{\rho}, \rho \simeq_\alpha \mathfrak{N}, \bar{\rho}, \hat{\rho}$.

We derive a bound on the length of $\hat{\rho}$ from the bound on $\Xi(\hat{\rho})$ by using the following lemma.

Lemma 3.2.15. Let $\mathcal{N}$ be a pushdown system and ρ a run of $\mathcal{N}$ such that $\max(\rho) = h$ and $\Xi(\rho) = b$, then $\ln(\rho) \leq \frac{b^{h+2}-b}{b-1}$.

Proof. Set $m_h := b$. For every $w \in \Sigma^h$ and some subrun $\pi \in \mathsf{SR}(\rho, w)$ we have $\ln(\pi) \leq m_h$ because the width of all stacks in s is h, which implies that all elements in s have stack w.

Now assume that every subrun $\pi' \in \mathsf{SR}(\rho, v)$ for some $v \in \Sigma^{n+1}$ has $\ln(\pi') \leq m_{n+1}$. Let $w \in \Sigma^n$ be an arbitrary word and let $\pi \in \mathsf{SR}(\pi', w)$. Then there are

$$0 = e_1 < e_2 < \cdots < e_f < e_{f+1} = \ln(\pi)$$

such that for $0 \leq i \leq f$, the stack at e_i in π is w and $\pi\restriction_{[e_i+1, e_{i+1}-1]}$ is w_i-prefixed for some $w_i \in \Sigma^{n+1}$. We have $f \leq b$ due to $\Xi(\pi) \leq \Xi(\rho) \leq b$. By assumption we get $\ln(\pi) \leq (1 + m_{n+1})b$. Note that $\rho \in \mathsf{SR}(\rho, \varepsilon)$ whence

$$\ln(\rho) \leq m_0 = b + bm_1 = b + b^2 + b^2 m_2 = \cdots = m_h \sum_{i=0}^{h} b^i = \frac{b^{h+2} - b}{b - 1}.$$

□

The rest of this section is concerned with the proofs of the pumping lemmas. The reader who is not interested in these technical details may skip the rest of this section and continue reading Section 3.2.4.

We start with some auxiliary lemmas. These are concerned with the structure of runs that are connected by a path of a given length n.

The first observation is that the final stack of runs ρ and $\hat{\rho}$ that are connected by an edge differ in at most one letter. Using this observation inductively, we obtain the following lemma.

Lemma 3.2.16. Let ρ and $\hat{\rho}$ be runs that are connected by a path of length n in some nested pushdown trees. Then $|\mathsf{wdt}(\rho) - \mathsf{wdt}(\hat{\rho})| \leq n$.

Next, we state another auxiliary lemma concerning prefixes of connected runs. Recall that, for w some word and ρ some run, $w \trianglelefteq \rho$ holds if w is a prefix of all stacks occurring in ρ.

Lemma 3.2.17. Let ρ and $\hat{\rho}$ be runs of a pushdown system such that the following holds. Setting $n := \ln(\rho)$, there is a word $w \in \Sigma^*$, a letter $\sigma \in \Sigma$, and numbers $i < j \in \mathsf{dom}(\rho)$ such that $\mathsf{top}_2(\rho(i)) = w$, $w \trianglelefteq \rho\restriction_{[i,n]}$ and $w\sigma \trianglelefteq \rho\restriction_{[j,n]}$.

For every $* \in \{\hookrightarrow, \hookleftarrow, \vdash, \dashv\}$, if $\rho * \hat{\rho}$ then $\hat{\rho} = \rho\restriction_{[0,i]} \circ \hat{\rho}'$ for some $\hat{\rho}'$ with $w \trianglelefteq \hat{\rho}'$.

Proof.

- If $\hat{\rho} \vdash \rho$, then it follows immediately from $i < j \leq n$ that $w \trianglelefteq \hat{\rho}' := \hat{\rho}\restriction_{[i,n-1]}$.
- If $\rho \vdash \hat{\rho}$, then $\hat{\rho}$ extends ρ by one configuration. Since each stack operation alters the height of the stack by at most one, $\mathsf{top}_2(\rho(n)) = w\sigma$ implies directly that $w \trianglelefteq \hat{\rho}'\restriction_{[i,\ln(\hat{\rho})]}$.

- If $\rho \hookrightarrow \hat{\rho}$, a similar argument as in the previous case applies. $\hat{\rho}$ extends ρ only by configurations that are prefixed by $\mathsf{top}_2(\rho)$. Since the last stack of ρ is prefixed by w, the claim follows immediately.

- Finally, consider the case that $\hat{\rho} \hookrightarrow \rho$. By definition of $\hookrightarrow$, we have $w\sigma \leq \rho(i)$ for all $i \in \mathsf{dom}(\rho) \setminus \mathsf{dom}(\hat{\rho})$. Furthermore, $\hat{\rho}$ is an initial segment of ρ. Thus, $\rho\restriction_{[0,i]}$ is an initial segment of $\hat{\rho}$. The claim follows because $\hat{\rho}\restriction_{[i,\ln(\hat{\rho})]}$ is an initial segment of $\rho\restriction_{[i,n]}$ whence it is w prefixed. □

Iterated use of the previous lemma yields the following corollary.

Corollary 3.2.18. Let ρ and $\hat{\rho}$ be runs of a pushdown system such that the following holds. Setting $n := \ln(\rho)$, there are words $w, v \in \Sigma^*$ with $|v| \geq m$, and numbers $i < j \in \mathsf{dom}(\rho)$ such that $\mathsf{top}_2(\rho(i)) = w$, $w \trianglelefteq \rho\restriction_{[i,n]}$ and $wv \trianglelefteq \rho\restriction_{[j,n]}$.

If ρ and $\hat{\rho}$ are connected by a path of length m, then $\hat{\rho} = \rho\restriction_{[0,i]} \circ \hat{\rho}'$ such that $w \trianglelefteq \hat{\rho}'$.

Proof. The proof is by induction on m. The case $m = 0$ is trivial and the case $m = 1$ is exactly the previous lemma. Assume that the claim holds for some $m \in \mathbb{N}$. Let ρ and $\hat{\rho}$ be connected by a path of length $m+1$, i.e., $\rho = \rho_1 * \rho_2 * \cdots * \rho_m = \hat{\rho}$ where each $*$ can be replaced by an element of $\{\hookrightarrow, \hookleftarrow, \vdash, \dashv\}$.

For $u := \mathsf{pop}_1(\rho)$, let $k \in \mathsf{dom}(\rho)$ be maximal such that $\mathsf{top}_2(\rho(k)) = u$ for some $q \in Q$. By definition $u \trianglelefteq \rho\restriction_{[k,n]}$. Due to the previous lemma, $\rho\restriction_{[0,k]}$ is an initial segment of ρ_2 and $\rho_2 = \rho\restriction_{[0,k]} \circ \rho_2'$ with $u \trianglelefteq \rho_2'$.

Now, ρ_2 and $\hat{\rho}$ are connected by a path of length $m-1$. Furthermore, $\rho\restriction_{[0,i]}$ is a prefix of ρ_2 and $w \trianglelefteq \rho_2\restriction_{[i,\ln(\rho_2)]}$. Moreover, there is some j such that $\mathsf{pop}_1(wv) \trianglelefteq \rho_2\restriction_{[j,\ln(\rho_2)]}$. By induction hypothesis we conclude that $\rho\restriction_{[0,i]}$ is a prefix of $\hat{\rho}$ and $w \trianglelefteq \hat{\rho}' := \hat{\rho}\restriction_{[i,\ln(\hat{\rho})]}$. □

We now prove the first pumping lemma that translates a given run ρ into an equivalent one with small final stack.

Proof of Lemma 3.2.10. Let $v := \mathsf{top}_2(\rho)$. Using the proof of Lemma 2.3.7, we find $w_1 < w_2 < v$ and numbers $n_1 < n_2 \leq \ln(\rho)$ such that $\rho(n_1) = (q_1, w_1)$, $\rho(n_2) = (q_2, w_2)$ and such that

$$\hat{\rho} := \rho\restriction_{[0,n_1]} \circ \rho\restriction_{[n_2,\ln(r)]}[w_2/w_1]$$

is a valid run. Because of the length of v, we can furthermore choose w_1 and w_2 such that the following holds:

1. $|w_1| > \mathsf{wdt}(\rho_i)$ for each i,
2. $|v| > |w_2| + 2^\alpha$, and
3. $|w_2| - |w_1| > 1 + 2^{\alpha+1}$.

We show that $\mathfrak{N}, \bar{\rho}, \rho \simeq_\rho \mathfrak{N}, \bar{\rho}, \hat{\rho}$.

Recall that we write $\mathfrak{N}_{2^\alpha}(\rho)$ for the 2^α-neighbourhood of ρ. Note that

$$\mathsf{wdt}(\rho) - \mathsf{wdt}(\hat{\rho}) = |w_2| - |w_1| > 1 + 2^{\alpha+1}.$$

Using Lemma 3.2.16, one concludes that $\mathfrak{N}_{2^\alpha}(\rho)$ and $\mathfrak{N}_{2^\alpha}(\hat{\rho})$ do not touch.

Furthermore, due to condition 2 and Lemma 3.2.18 it follows that for all $\pi \in \mathfrak{N}_{2^\alpha}(\rho)$ we have $\pi = \rho\restriction_{[0,n_2]} \circ \pi'$ for some run π' with $w_2 \trianglelefteq \pi'$. Analogously, for all $\pi \in \mathfrak{N}_{2^\alpha}(\hat{\rho})$ we have $\pi = \rho\restriction_{[0,n_1]} \circ \pi'$ for some run π' with $w_1 \trianglelefteq \pi'$. Lemma 2.3.6 and a straightforward induction on the neighbourhoods of ρ and $\hat{\rho}$ show that the function

$$\begin{aligned}\varphi : \mathfrak{N}_{2^\alpha}(\rho) &\to \mathfrak{N}_{2^\alpha}(\hat{\rho}) \\ \pi &\mapsto \rho\restriction_{[0,n_1]} \circ \pi'[w_2/w_1] \text{ where} \\ \pi' &:= \pi\restriction_{[n_2,\ln(\rho)]}\end{aligned}$$

is a well-defined isomorphism between $\mathfrak{N}_{2^\alpha}(\rho)$ and $\mathfrak{N}_{2^\alpha}(\hat{\rho})$.

Finally, since $\mathsf{wdt}(\rho) > \mathsf{wdt}(\hat{\rho}) \geq |w_1| > \mathsf{wdt}(\rho_i) + 2^\alpha$, again by Lemma 3.2.18, ρ_i cannot be in the 2^α-neighbourhood of ρ or $\hat{\rho}$. Hence, we apply Lemma 3.2.9 and obtain that $\mathfrak{N}, \bar{\rho}, \rho \simeq_\alpha \mathfrak{N}, \bar{\rho}, \hat{\rho}$. □

Next, we prove the second $\simeq_\alpha$-type preserving pumping lemma that preserves the last configuration of a run ρ, but reduces $\mathsf{max}(\rho)$. Recall that $\mathsf{max}(\rho)$ denotes the size of the largest stack occurring in ρ.

Proof of Lemma 3.2.12. Let $\rho_1, \rho_2, \ldots, \rho_m$, and ρ be runs such that

$$\begin{aligned}&\mathsf{max}(\rho) > \mathsf{max}(\rho_i) + |Q|^2|\Sigma| + 1 \text{ for all } 1 \leq i \leq m, \text{ and such that} \\ &\mathsf{max}(\rho) > |\mathsf{wdt}(\rho)| + |Q|^2|\Sigma| + 2^\alpha + 1.\end{aligned}$$

We construct $\hat{\rho}$ as follows.

Let $i \in \mathsf{dom}(\rho)$ be such that $\rho(i) = (q, w)$ for some $q \in Q$ and $w \in \Sigma^*$ with $|w| = \mathsf{max}(\rho)$. This implies $|w| > |Q^2||\Sigma| + 2^\alpha + 1 + \mathsf{wdt}(\rho)$.

Now, using the proof of Lemma 2.3.7 we find $w_1 < w_2 \leq w$ and numbers

$$n_1 < n_2 < m_2 < m_1$$

such that

1. $\mathsf{max}(\rho_i) < |w_1|$,
2. $|w_1| > \mathsf{wdt}(\rho) + 2^\alpha + 1$, and
3. $\hat{\rho} := \rho\restriction_{[0,n_1]} \circ \rho\restriction_{[n_2,m_2]}[w_2/w_1] \circ \rho\restriction_{[m_1,\ln(\rho)]}$ is a valid run.

Now, we set

$$\begin{aligned}m_1' &:= m_1 - (n_2 - n_1) - (m_1 - m_2), \\ \rho_A &:= \rho\restriction_{[0,m_1+1]} \text{ and} \\ \rho_B &:= \hat{\rho}\restriction_{[0,m_1'+1]}.\end{aligned}$$

Note that $\hat{\rho} = \rho_B \circ \rho\restriction_{[m_1+1,\ln(\rho)]}$.

We use Lemma 3.2.9 to show that $\mathfrak{N}, \bar{\rho}, \rho \simeq_\alpha \mathfrak{N}, \bar{\rho}, \hat{\rho}$. For this purpose we set

$$\begin{aligned}A &:= \{\pi \in \mathfrak{N}_{2^\alpha}(\rho) : \pi = \rho_A \circ \pi', \pi' \text{ some run}\} \text{ and} \\ B &:= \{\pi \in \mathfrak{N}_{2^\alpha}(\hat{\rho}) : \pi = \rho_B \circ \pi', \pi' \text{ some run}\}.\end{aligned}$$

Observe that $\rho_A \notin A$ and $\rho_B \notin B$: this is due to Lemma 3.2.16 and the fact that

$$\mathsf{wdt}(\rho_A) = |w_1| - 1 > \mathsf{wdt}(\rho) + 2^\alpha.$$

The proof for ρ_B and B is analogous. Furthermore, for all $\pi \in A$ and all $\pi' \in B$ we have

$$\pi(m_1' + 1) = \rho_A(m_1' + 1) \neq \rho_B(m_1' + 1) = \pi'(m_1' + 1).$$

This is due to the fact that ρ_B ends in stack $\mathsf{pop}_1(w_1)$ (at position $m_1' + 1$) and $w_1 \trianglelefteq \rho_A(m_1' + 1)$ because $n_1 \leq m_1' + 1 \leq m_1$.

We conclude that the greatest common prefix of some $a \in A$ and some $b \in B$ is a proper initial prefix of both runs. Hence, a and b are not connected by an edge whence A and B do not touch.

Furthermore, note that $\rho_i \notin A \cup B$ because for all $\pi \in A \cup B$, we have

$$\max(\pi) \geq \max(\rho_B) \geq |w_1| > \max(\rho_i).$$

Recall that $A_{2^\alpha}(\rho)$ denotes the 2^α-neighbourhood of ρ in the subgraph induced by A. We claim that there is an isomorphism φ of the induced subgraphs

$$\begin{aligned} &\varphi : A_{2^\alpha}(\rho) \simeq B_{2^\alpha}(\hat{\rho}) \\ &\rho_A \circ \pi \mapsto \rho_B \circ \pi. \end{aligned}$$

For the proof of this claim, note that for any two runs π', π'' of length at least 1, and for $* \in \{\vdash, \dashv, \hookrightarrow, \hookleftarrow\}$ we have

$$\begin{aligned} &\rho_A \circ \pi' * \rho_A \circ \pi'' \text{ iff} \\ &\rho_B \circ \ \pi' * \rho_B \circ \pi''. \end{aligned}$$

From this observation it follows by induction on the distance from ρ that

$$\varphi(A_{2^\alpha}(\rho)) \subseteq B_{2^\alpha}(\hat{\rho}).$$

Analogously, by induction on the distance from $\hat{\rho}$ one shows that

$$B_{2^\alpha}(\hat{\rho}) \subseteq \varphi(A_{2^\alpha}(\rho)).$$

One concludes immediately that φ is an isomorphism.

In order to apply the game argument, we finally have to show that φ and φ^{-1} preserve edges between $\mathfrak{N} \setminus (A_{2^\alpha}(\rho) \cup B_{2^\alpha}(\hat{\rho}))$ and $A_{2^\alpha - 1}(\rho)$ or $B_{2^\alpha - 1}(\hat{\rho})$, respectively. Assume that $a \in A_{2^\alpha - 1}(\rho)$ and $c \in \mathfrak{N} \setminus \big(A_{2^\alpha}(\rho) \cup B_{2^\alpha}(\hat{\rho})\big)$. We claim that if a and c are connected by some edge, then we have $c \hookrightarrow a$.

Note that $a \vdash c$ or $a \hookrightarrow c$ implies that a is a subrun of c and therefor $c \in A_{2^\alpha}(\rho)$ by definition of A. If $c \vdash a$, then $\mathsf{wdt}(c) \leq \mathsf{wdt}(\rho)| + 2^\alpha < |w_1| - 1$. Hence, $c \neq \rho_A$. Since ρ_A is a proper initial segment of a, this implies $c \in A_{2^\alpha}(\rho)$.

Thus, if $c \in \mathfrak{N} \setminus \big(A_{2^\alpha}(\rho) \cup B_{2^\alpha}(\hat{\rho})\big)$ is connected to a then $c \hookrightarrow a$ and c is a proper initial segment of ρ_A. Since the last stack of a and c agree and $\mathsf{wdt}(a) < |w_1|$, c is an initial segment

of $\rho\restriction_{[0,n_1]}$. Furthermore, if the stack at $a(i)$ is prefixed by some $v < w_1$ for all $n_1 \leq i \leq \ln(a)$, then the stack of $\varphi(a)(j)$ is prefixed by some $v < w_1$ for all $n_1 \leq j \leq \ln(\varphi(a))$. Moreover, $\rho\restriction_{[0,n_1]}$ is an initial segment of $\varphi(a)$ whence $c \hookrightarrow \varphi(a)$.

An completely analogous analysis of φ^{-1} shows that φ^{-1} preserves edges between $B_{2^\alpha-1}(\hat{\rho})$ and $\mathfrak{N} \setminus \big(A_{2^\alpha}(\rho) \cup B_{2^\alpha}(\hat{\rho})\big)$.

Thus, we can apply Lemma 3.2.9 and obtain that

$$\mathfrak{N}, \bar{\rho}, \rho \simeq_\alpha \mathfrak{N}, \bar{\rho}, \hat{\rho}$$

and $\ln(\hat{\rho}) < \ln(\rho)$.

Now, either $\max(\hat{\rho}) < \max(\rho)$ or we can apply the same construction again to $\hat{\rho}$. Since $\ln(\rho)$ is finite and the length decreases in every step, we eventually construct a run $\hat{\rho}$ with $\max(\hat{\rho}) < \max(\rho)$. □

By now, we have shown how to preserve the $\simeq_\alpha$-type of a run while bounding the size of all stacks that occur.

Recall the statement of Lemma 3.2.15: if the size of the stacks that occur in a run ρ is bounded, then a bound on $\Xi(\rho)$ can be used to calculate a bound on the length of ρ. $\Xi(\rho)$ is the maximal number of occurrences of a word w in a w prefixed subrun of ρ.

For the proof of the third pumping lemma, we need some insight into the relationship of $\Xi(\rho, w)$ and $\Xi(\pi, w)$ for runs ρ and π that are connected in $\mathrm{NPT}(\mathscr{N})$. Before we come to these insights, we introduce the following notation.

Definition 3.2.19. For $\hat{\rho} = \rho\restriction_{[i,j]}$ we call $\hat{\rho}$ a left maximal subrun of ρ if $\hat{\rho} \in \mathrm{SR}(w, \rho)$ and $w \not\leq \rho(i-1)$. Analogously, we call $\hat{\rho}$ a right maximal subrun of ρ if $\hat{\rho} \in \mathrm{SR}(w, \rho)$ and $w \not\leq \rho(j+1)$. We call $\hat{\rho}$ maximal if it is left and right maximal.

Lemma 3.2.20. Let $\rho = \rho_1 \circ \rho_2 \circ \rho_3$ be a run such that $\rho_2 \in \mathrm{SR}(w, \rho)$ is maximal for some $w \in \Sigma^*$. If $\rho \hookrightarrow \pi$ or $\rho \vdash \pi$ for some run π, then π decomposes as $\pi = \rho_1 \circ \pi_2 \circ \pi_3$ for $\pi_2 \in \mathrm{SR}(w, \pi)$ maximal. In this case, we have

$$|\pi_2|_w - |\rho_2|_w \in \{0, 1\}.$$

Proof. For $\rho \vdash \pi$, the proof is trivial because π extends ρ by exactly one configuration.

It remains to consider the case $\rho \hookrightarrow \pi$. Due to the maximality of ρ_2, we have $\ln(\rho_3) = 0$ or $\rho_3(1) < w$. If $\rho_3(1) < w$, then $\pi = \rho_1 \circ \rho_2 \circ \rho_3 \circ \pi'$ for some run π' which implies $\pi_2 = \rho_2$.

Otherwise, if $\ln(\rho_3) = 0$, then $\rho = \rho_1 \circ \rho_2$. Hence, $\pi = \rho_1 \circ \rho_2 \circ \pi'$ such that the last stacks of ρ_2 and π' agree and $w \leq \rho_2(\ln(\rho_2)) = \pi'(\ln(\pi')) < \pi'(i)$ for all $1 \leq i < \ln(\pi')$. Thus, if w is the stack of $\rho_2(\ln(\rho_2))$ then $|\rho_2 \circ \pi'|_w = |\rho_2|_w + 1$. Furthermore, if $w < \rho_2(\ln(\rho_2))$, then $|\rho_2 \circ \pi'|_w = |\rho_2|_w$. □

This lemma has two corollaries that we are going to use in the proof of the third pumping lemma.

Corollary 3.2.21. Let ρ, ρ' be runs such that $\rho \vdash^{\rho'}$ or $\rho \hookrightarrow \rho'$. If ρ decomposes as $\rho = \rho_1 \circ \rho_2$ where ρ_2 is a maximal, w-prefixed subrun, then ρ' decomposes as $\rho' = \rho_1 \circ \rho_2 \circ \rho_3'$ such that $\rho_2 \circ \rho_3'$ is maximal and w-prefixed such that

$$|\rho_2 \circ \rho_3'|_w - |\rho_2|_w \in \{0, 1\}.$$

Corollary 3.2.22. Let $\rho = \rho_1 \circ \rho_2 \circ \rho_3$ be a run such that $\rho_2 \in \mathsf{SR}(w, \rho)$ is maximal for some $w \in \Sigma^*$. Let π be a run that is connected to ρ via a path of length n that only visits runs π' such that ρ_1 is a prefix of π', then π decomposes as $\pi = \rho_1 \circ \pi_2 \circ \pi_3$ for $\pi_2 \in \mathsf{SR}(w, \pi)$ maximal. In this case, we have

$$|\pi_2|_w - |\rho_2|_w \leq n.$$

A straightforward induction proves this corollary.

Using these results, we can prove the third pumping lemma, which bounds $\Xi(\rho)$. The proof relies on the fact that for some large run ρ, we find initial segments ρ_1 and ρ_2 of ρ ending in the same configuration (q, w) such that $|\rho_1|_w$ is much smaller than $|\rho_2|_w$ for some word w and some state q.

Proof of Lemma 3.2.14. Assume $\Xi(\rho)$ is too big in the sense that there is a word $w \in \Sigma^*$ such that $\Xi(\rho', w) > B_\Xi + (2^{\alpha+1} + 2)|Q| + 2^\alpha + 1$ for some $\rho' \in \mathsf{SR}(w, \rho)$, i.e., for some w prefixed subrun ρ' of ρ.

Then there is a decomposition of ρ as $\rho = \rho_1 \circ \rho_2 \circ \rho_3 \circ \rho_4 \circ \rho_5$ such that the following holds.

1. $\rho_2 \circ \rho_3 \circ \rho_4 \in \mathsf{SR}(w, \rho)$,
2. $\rho_2(0) = \rho_3(0) = (q, w)$ for some $q \in Q$,
3. $|\rho_2|_w \geq 2^{\alpha+1} + 2$,
4. $|\rho_3|_w > B_\Xi$,
5. $|\rho_4|_w = 2^\alpha$, and
6. ρ_4 is right maximal in $\mathsf{SR}(w, \rho)$, (this implies $\mathsf{ln}(\rho_5) = 0$ or $w < \rho_5(1)$).

We set $\hat{\rho} := \rho_1 \circ \rho_3 \circ \rho_4 \circ \rho_5$ omitting ρ_2 in ρ and claim that $\mathfrak{N}, \bar{\rho}, \rho \simeq_\alpha \mathfrak{N}, \bar{\rho}, \hat{\rho}$. The proof uses again Lemma 3.2.9. Let

$$B := \{\hat{\pi} \in \mathfrak{N} : \hat{\pi} = \rho_1 \circ \rho_3 \circ \hat{\pi}_1 \circ \hat{\pi}_2, \hat{\pi}_1 \in \mathsf{SR}(w, \hat{\pi}) \text{ right maximal and} |\hat{\pi}_1|_w \leq 2^{\alpha+1}\} \text{ and}$$
$$A := \{\pi \in \mathfrak{N} : \pi = \rho_1 \circ \rho_2 \circ \rho_3 \circ \pi_1 \circ \pi_2, \pi_1 \in \mathsf{SR}(w, \pi) \text{ right maximal and} |\pi_1|_w \leq 2^{\alpha+1}\}.$$

First note that for all $1 \leq i \leq n$, $\rho_i \notin A \cup B$ because $\Xi(\rho_i) < B_\Xi < |\rho_3|_w \leq \Xi(\pi)$ for all $\pi \in A \cup B$.

Now, we show that A and B do not touch. Let

$$a = \rho_1 \circ \rho_2 \circ \rho_3 \circ \pi_1 \circ \pi_2 \in A$$

such that π_1 is right maximal in $\mathsf{SR}(a, w)$ and

$$b = \rho_1 \circ \rho_3 \circ \hat{\pi}_1 \circ \hat{\pi}_2 \in B$$

such that $\hat{\pi}_1$ is right maximal in $\mathsf{SR}(b, w)$.

Heading for a contradiction, we assume that there is some edge connecting a and b. There are the following cases.

1. Assume that $a \hookrightarrow b$ or $a \vdash b$. In both cases we have $b = a \circ \pi'$ for some run π'. The assumption implies that $\rho_2 \circ \rho_3$ is a prefix of $\rho_3 \circ \hat{\pi}_1 \circ \hat{\pi}_2$. Note that $\rho_2 \circ \rho_3$ is w prefixed, while $\hat{\pi}_2(1)$ is not w prefixed (if $\ln(\hat{\pi}_2) \geq 1$). Thus, we conclude that $\rho_2 \circ \rho_3$ is a prefix of $\rho_3 \circ \hat{\pi}_1$. But this clearly contradicts

$$|\rho_2 \circ \rho_3|_w \geq B_\Xi + 2^{\alpha+1} + 2 > B_\Xi + 2^{\alpha+1} \geq |\rho_3 \circ \hat{\pi}_1|_w.$$

2. Assume that $b \hookrightarrow a$ or $b \vdash a$. Due to $|\rho_2 \circ \rho_3|_w > |\rho_3 \circ \hat{\pi}_1|_w$, $\rho_3 \circ \hat{\pi}_1$ is a proper prefix of $\rho_2 \circ \rho_3$.

   It follows that $\ln(\hat{\pi}_2) = 0$: otherwise, $\hat{\pi}_2(1) = \rho_3(j)$ for some $j \in \mathsf{dom}(\rho_3)$. But this leads to the contradiction that $w \not\leq \hat{\pi}_2(1) = \rho_3(j)$ due to the right maximality of $\hat{\pi}_1$ but $w \leq \rho_3(j)$ by definition of ρ_3.

   Hence, Corollary 3.2.21 shows that

$$|\rho_2 \circ \rho_3 \circ \pi_1 \circ \pi_2|_w \leq |\rho_3 \circ \hat{\pi}_1| + 1.$$

   But this contradicts the fact that

$$|\hat{\pi}_1|_w + 1 \leq 2^{\alpha+1} + 1 < 2^{\alpha+2} + 2 \leq |\rho_2|_w.$$

Thus, A and B do not touch. Now, the map

$$\begin{aligned} &\varphi : A \to B \\ &\rho_1 \circ \rho_2 \circ \pi \mapsto \rho_1 \circ \pi \end{aligned}$$

is clearly well-defined. Furthermore, it is an isomorphism. For $* \in \{\vdash, \dashv, \hookrightarrow, \hookleftarrow\}$ and for runs π, π' with $\rho_2(0) = \pi(0) = \pi'(0)$ we have

$$\begin{aligned} &(\rho_1 \circ \rho_2 \circ \pi) * (\rho_1 \circ \rho_2 \circ \pi') \\ \text{iff } &\pi * \pi' \\ \text{iff } &(\rho_1 \circ \pi) * (\rho_1 \circ \pi'). \end{aligned}$$

In order to apply Lemma 3.2.9, we have to show that φ and φ^{-1} preserve edges between $\mathfrak{N} \setminus \big(A_{2^\alpha}(\rho) \cup B_{2^\alpha}(\hat{\rho})\big)$ and $A_{2^\rho - 1}(\rho)$ or $B_{2^\rho - 1}(\hat{\rho})$, respectively.

Note that for $k < 2^\alpha$, Corollary 3.2.22 states that $a \in A_k(\rho)$ implies

$$a = \rho_1 \circ \rho_2 \circ \rho_3 \circ \pi_1 \circ \pi_2$$

for some right maximal $\pi_1 \in \mathsf{SR}(w, a)$ such that $|\pi_1|_w \in [2^\alpha - k, 2^\alpha + k]$.

One immediately concludes that $a \vdash c, c \vdash a$, or $a \hookrightarrow c$ implies that $c \in A_{2^\alpha}(\rho)$ because

$$c = \rho_1 \circ \rho_2 \circ \rho_3 \circ \hat{\pi}_1 \circ \hat{\pi}_2$$

for some right maximal subrun $\hat{\pi}_1 \in \mathsf{SR}(w, c)$ with $|\hat{\pi}_1|_w \in [2^\alpha - k - 1, 2^\alpha + k + 1]$. Since this contradicts the assumption that $c \notin A_{2^\alpha}(\rho)$, we only have to consider the case $c \hookrightarrow a$. We analyse three possibilities.

1. If the last stack of a is w prefixed, then Corollary 3.2.21 implies that $c \in A_{2^\alpha}(\rho)$ which contradicts the assumption on c.

2. If the last stack of a is not w prefixed and c is not a proper prefix of ρ_1, then

$$c = \rho_1 \circ \rho_2 \circ \rho_3 \circ \pi_1 \circ \hat{\pi}_2$$

where $\hat{\pi}_2(1) = \pi_2(1)$. But then $c \in A_{2^\alpha}(\rho)$ which again contradicts the assumption on c.

3. Finally, we consider the case that c is a proper prefix of ρ_1. Since the last stack of c is then a proper prefix of w, one concludes immediately that

$$c \hookrightarrow \varphi(a) = \rho_1 \circ \rho_2 \circ \pi_1 \circ \pi_2.$$

Using the analogous arguments with reversed roles for A and B, one shows that φ^{-1} also preserves the edges from $B_{2^\alpha-1}(\hat{\rho})$ to $\mathfrak{N} \setminus \big(A_{2^\alpha}(\rho) \cup B_{2^\alpha}(\hat{\rho})\big)$.

Hence, Lemma 3.2.9 shows that

$$\mathfrak{N}, \bar{\rho}, \rho \simeq_\alpha \mathfrak{N}\bar{\rho}, \hat{\rho}.$$

Iteration of this construction eventually leads to the construction of some $\hat{\rho}$ that satisfies the lemma. □

3.2.4 First-Order Model Checking on NPT is in 2-EXPSPACE

Using the three pumping lemmas we can now establish a dynamic small witness property for nested pushdown trees: let $\varphi(x_1, x_2, \ldots, x_n)$ be an FO formula that is satisfied by some nested pushdown tree $\mathrm{NPT}(\mathscr{N})$ with parameters $\rho_1, \rho_2, \ldots, \rho_n \in \mathrm{NPT}(\mathscr{N})$. Then the outermost existential quantification occurring in φ is witnessed by a small run ρ such that the length of ρ is bounded in terms of the length of $\rho_1, \rho_2, \ldots, \rho_n$. In order to state this fact in a precise manner, we first define the appropriate notion of a small run.

Definition 3.2.23. Let $\mathscr{N} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a pushdown system. For $j \leq k \in \mathbb{N}$ we say that some $\rho \in \mathrm{NPT}(\mathscr{N})$ is (j,k)-small if

$$\mathrm{wdt}(\rho) \leq 6|\mathscr{N}|^2 j 2^k, \qquad \max(\rho) \leq 8|\mathscr{N}|^3 j 2^k, \qquad \text{and } \Xi(\rho) \leq 6|\mathscr{N}| j 2^k.$$

Now, we can put all the pumping lemmas together in order to prove the existence of a small $\simeq_\alpha$-equivalent tuple for every tuple of elements.

Lemma 3.2.24. Let $\mathscr{N} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a pushdown system and

$$\bar{\rho} = \rho_1, \rho_2, \ldots, \rho_{i-1} \in \mathrm{NPT}(\mathscr{N})$$

such that ρ_j is (j,α)-small for all $1 \leq j \leq i-1$ and $1 \leq i \leq \alpha \in \mathbb{N}$. For each $\rho_i \in \mathrm{NPT}(\mathscr{N})$, there is an (i,α)-small $\rho_i' \in \mathrm{NPT}(\mathscr{N})$ such that

$$\mathrm{NPT}(\mathscr{N}), \bar{\rho}, \rho_i \simeq_{\alpha-i} \mathrm{NPT}(\mathscr{N}), \bar{\rho}, \rho_i'.$$

Proof. Given ρ_i, the first pumping lemma (Lemma 3.2.10) shows that there is some $a \in \mathsf{NPT}(\mathscr{N})$ such that

$$\begin{aligned}&\mathfrak{N}, \bar{\rho}, \rho_i \simeq_{\alpha-i} \mathfrak{N}, \bar{\rho}, a \text{ and}\\ &\mathrm{wdt}(a) \leq 6|\mathscr{N}|^2 i 2^\alpha + |Q||\Sigma|(2 + 2^{(\alpha-i)+1}) + 2^{(\alpha-i)} + 1 \leq 6|\mathscr{N}|^2 i 2^\alpha.\end{aligned}$$

Due to the second pumping lemma (Lemma 3.2.12), there is some $b \in \mathfrak{N}$ such that

$$\begin{aligned}&\mathfrak{N}, \bar{\rho}, a \simeq_{\alpha-i} \mathfrak{N}, \bar{\rho}, b,\\ &b(\ln(b)) = (q, w) = a(\ln(a)) \text{ for some } q \in Q, w \in \Sigma^*, \text{ and}\\ &\max(b) \leq 8|\mathscr{N}|^3 i 2^\alpha + |Q|^2|\Sigma| + 1 \leq 8|\mathscr{N}|^3 i 2^\alpha.\end{aligned}$$

Finally, we apply the third pumping lemma (Lemma 3.2.14) and find some $c \in \mathfrak{N}$ such that

$$\begin{aligned}&\mathfrak{N}, \bar{\rho}, b \simeq_{\alpha-i} \mathfrak{N}, \bar{\rho}, c,\\ &c(\ln(c)) = (q, w) = b(\ln(b)) \text{ for some } q \in Q, w \in \Sigma^*,\\ &\max(c) \leq \max(b), \text{ and}\\ &\Xi(c) \leq 6|\mathscr{N}| i 2^\alpha + (2^{\alpha-i+1} + 2)|Q| + 2^{\alpha-i} + 1 \leq 6|\mathscr{N}| i 2^\alpha.\end{aligned}$$

□

In the terminology of Section 2.1.1, the previous lemma shows that there is a finitary constraint S for Duplicator's strategy in the Ehrenfeucht-Fraïssé game. We set

$$S^{\mathsf{NPT}(\mathscr{N})}(m) := \{\rho_1, \rho_2, \ldots, \rho_m \in \mathsf{NPT}(\mathscr{N})^m : \rho_i \text{ is } (i, \alpha)\text{-small for all } i \leq m\}$$

and $S := (S_i)_{i \leq \alpha}$. With this notation, the previous lemma shows that Duplicator has an S-preserving winning strategy in the α-round Ehrenfeucht-Fraïssé-game on two copies of $\mathsf{NPT}(\mathscr{N})$. As explained in Section 2.1.1, such a strategy has a direct translation into a model checking algorithm.

Theorem 3.2.25. The Algorithm 3 (see next page) solves the FO model checking problem on nested pushdown trees, i.e., given a pushdown system $\mathscr{N}$ and a sentence $\varphi \in \mathsf{FO}_\alpha$, NPTModelCheck accepts the input $(\mathscr{N}, \alpha, \emptyset, \varphi)$, if and only if $\mathsf{NPT}(\mathscr{N}) \models \varphi$. The structure complexity of this algorithm is in EXPSPACE, while its expression and combined complexity are in 2-EXPSPACE.

Proof. The correctness of the algorithm follows directly from the correctness of Algorithm 2 and from Lemma 3.2.24.

We analyse the space consumption of this algorithm. Due to Lemma 3.2.15 an (i, α)-small run ρ has bounded length. It can be stored as a list of $\exp(O(i|\mathscr{N}|^4 \alpha \exp(\alpha)))$ many transitions. Thus, we need $\exp(O(i|\mathscr{N}|^4 \alpha \exp(\alpha))) \log(\mathscr{N})$ space for storing one run. Additionally, we need space for checking whether such a list of transitions forms a valid run and for checking the atomic type of the runs. We can do this by simulation of $\mathscr{N}$. The size of

Algorithm: `NPTModelCheck`$(\mathcal{N}, \alpha, \bar{a}, \varphi(\bar{x}))$

Input: a pushdown system $\mathcal{N}$ generating $\mathfrak{N} := \mathrm{NPT}(\mathcal{N})$, $\alpha \in \mathbb{N}$, $\varphi \in \mathrm{FO}_\alpha$, an assignment $\bar{x} \mapsto \bar{a}$ for tuples $\bar{x}, \bar{a}$ of arity m such that $\bar{a}$ is (m, α)-small

if φ is an atom or negated atom then
  if $\mathfrak{N}, \bar{a} \models \varphi(\bar{x})$ then accept else reject;
if $\varphi = \varphi_1 \vee \varphi_2$ then
  if `NPTModelCheck`$(\mathfrak{N}, \alpha, \bar{a}, \varphi_1)$ = accept then accept else
    if `NPTModelCheck`$(\mathfrak{N}, \alpha, \bar{a}, \varphi_2)$= accept then accept else reject;
if $\varphi = \varphi_1 \wedge \varphi_2$ then
  if `NPTModelCheck`$(\mathfrak{N}, \alpha, \bar{a}, \varphi_1)$= `NPTModelCheck`$(\mathfrak{N}, \alpha, \bar{a}, \varphi_2)$= accept then accept else reject;
if $\varphi = \exists x \varphi_1(\bar{x}, x)$ then
  check whether there is an $a \in \mathfrak{N}$ such that a is $(m+1, \alpha)$-small and `NPTModelCheck`$(\mathfrak{N}, \alpha, \bar{a}a, \varphi_1)$= accept;
if $\varphi = \forall x_i \varphi_1$ then
  check whether `NPTModelCheck`$(\mathfrak{N}, \alpha, \bar{a}a, \varphi_1)$= accept holds for all $(m+1, \alpha)$-small $a \in \mathfrak{N}$;

Algorithm 3: FO model checking on nested pushdown trees

the stack is bounded by the size of the runs. Since we have to store up to α many runs at the same time and i is bounded by $\alpha \leq |\varphi|$, the algorithm is in

$$\mathrm{DSPACE}\big(|\varphi| \log(|\mathcal{N}|) \exp(O(|\mathcal{N}|^4 |\varphi|^2 \exp(|\varphi|)))\big) \subseteq$$
$$\mathrm{DSPACE}\big(\exp(O(|\mathcal{N}|^4 \exp(2|\varphi|)))\big) \subseteq \text{2-EXPSPACE}(|\mathcal{N}| + |\varphi|).$$

If the formula φ is fixed, the space consumption of the algorithm is exponential in the size of $\mathcal{N}$. Thus, the structure complexity of first-order model checking on nested pushdown trees is in EXPSPACE. □

Remark 3.2.26. Recall that we proved the existence of a nonelementary FO(REACH) model checking algorithm for nested pushdown trees. There is no hope in finding an elementary algorithm. A straightforward adaption of the proof of Theorem 3.1.63 shows this. As in the case of collapsible pushdown graphs, one can define a nested pushdown tree that is the full binary tree where each branch looks like the graph in Example 2.2. For similar arguments as in the proof of Theorem 3.1.63, FO model checking on the full infinite binary tree can be reduced to FO(REACH) model checking on this nested pushdown tree.

3.3 Higher-Order Nested Pushdown Trees

In this chapter, we propose the study of a new hierarchy of graphs. We combine the idea underlying the definition of nested pushdown trees with the idea of higher-order pushdown systems and obtain a notion of a higher-order nested pushdown tree. We first give a formal definition of this hierarchy. Afterwards, we compare this new hierarchy with the hierarchies of higher-order pushdown graphs and collapsible pushdown graphs.

Recall that nested pushdown trees are FO-interpretable in collapsible pushdown graphs of level 2. We show that this result extends to the whole hierarchy. Every nested pushdown tree of level n is FO-interpretable in some collapsible pushdown graph of level $n+1$.

In the final part of this chapter we then prove the decidability of the first-order model checking on level 2 nested pushdown trees. The approach is an adaption of the idea underlying the decidability proof of the level 1 case: we prove that there is a strategy in the Ehrenfeucht-Fraïssé game such that Duplicator always chooses small runs. But the techniques involved in the proof of the existence of such a strategy are very different from those in the level 1 case.

3.3.1 Definition of Higher-Order Nested Pushdown Trees

We want to define the notion of higher-order nested pushdown trees. Recall that a nested pushdown tree is the unfolding of a pushdown graph extended by a jump-relation $\hookrightarrow$ that connects corresponding push- and pop operations. Extending this idea to higher levels, one has to define what corresponding push- and pop operations in a level n pushdown system are. In order to obtain well-nested jump-edges, we concentrate on the push- and pop operations of the highest level, i.e., for a level n pushdown system we look at corresponding clone_n and pop_n operations.

Definition 3.3.1. Let $\mathcal{N} = (\Sigma, \Gamma, Q, q_0, \Delta)$ be a pushdown system of level n.[6] Then the level n nested pushdown tree $\mathfrak{N} := \mathsf{NPT}(\mathcal{N})$ is the unfolding of the pushdown graph of $\mathcal{N}$ expanded by the relation $\hookrightarrow$ which connects each clone_n operation with the corresponding pop_n operation, i.e., for runs ρ_1, ρ_2 of $\mathcal{N}$ we have $\rho_1 \hookrightarrow \rho_2$ if ρ_2 decomposes as $\rho_2 = \rho_1 \circ \rho$ for some run ρ from (q,s) to (q',s) of length n such that

$$\begin{aligned}
&\rho(0) \vdash^{\mathsf{clone}_n} \rho(1),\\
&\rho(n-1) \vdash^{\mathsf{pop}_n} \rho(n), \text{ and}\\
&\rho(i) \neq (\hat{q}, s) \text{ for all } 1 \leq i < n \text{ and all } \hat{q} \in Q.
\end{aligned}$$

Remark 3.3.2. Another view on the jump edges is the following. Some run ρ_1 is connected via $\hookrightarrow$ to some other run ρ_2 if ρ_2 decomposes as $\rho_2 = \rho_1 \circ \rho$ where ρ consists of a clone_n operation followed by a "level n return". It is straightforward to show that $\rho_1 \hookrightarrow \rho_2$ if and only if $\rho_2 = \rho_1 \circ \rho$ for some run ρ of length at least 2 such that $|\rho(0)| = |\rho(\ln(\rho)|$ and $|\rho(i)| > |\rho(0)|$ for all $0 < i < \ln(\rho)$.

In the following, we write n-NPT for "nested pushdown tree of level n".

3.3.2 Comparison with Known Pushdown Hierarchies

The hierarchy of higher-order nested pushdown trees is a hierarchy strictly extending the hierarchy of trees generated by higher-order pushdown systems. Furthermore, it is first-order interpretable in the collapsible pushdown hierarchy. In fact, this relationship of the hierarchies is level by level. In the following, we prove these claims.

We start by adapting the first-order interpretation of nested pushdown trees in collapsible pushdown graphs of level 2 to the interpretation of nested pushdown trees of level n in collapsible pushdown graphs of level $n+1$. The approach is completely analogous. First of all, each configuration (q,s) of a level n pushdown system $\mathcal{N}$ is identified with the level n

[6] We stress that $\mathcal{N}$ is a pushdown system without links and without collapse-transitions.

stack $\mathsf{push}_{q,1}(s)$. A run ρ of $\mathscr{N}$ is a list of configurations $\rho(0), \rho(1), \ldots, \rho(\ln(\rho))$. This run is identified with the level $n+1$ stack $s_\rho := \rho(0) : \rho(1) : \cdots : \rho(\ln(\rho))$.

Each extension of ρ by one transition $\delta := (q, \sigma, \gamma, q', \mathrm{op})$ can be simulated by a level $n+1$ pushdown system by changing the stack to

$$s_{\rho'} := \mathsf{push}_{q',1}(\mathrm{op}(\mathsf{pop}_1(\mathsf{clone}_{n+1}(s_\rho)))).$$

It is a straightforward observation that $s_{\rho'}$ represents the run ρ' which is ρ extended by δ. Hence, the unfolding of a level n pushdown system can be simulated by some level $n+1$ collapsible pushdown system.

In order to simulate the nested pushdown tree generated by $\mathscr{N}$, we also have to simulate the jump-edges. A jump-edge connects a clone_n transition with the corresponding pop_n transition. Thus, the collapsible pushdown system simulating $\mathscr{N}$ has to keep track of the positions where a clone_n transition was performed.

For this purpose we introduce a clone-marker #. Before the collapsible pushdown system performs a clone_n transition, it applies a $\mathsf{push}_{\#,n+1}$ operation. This means that it writes the symbol # onto the stack. This symbol carries a link to the stack representing the run up to the configuration before the clone_n transition was applied.

Later, when the system simulates a pop_n transition, it finds a clone of this marker # on top of the stack reached by this pop_n. The link of this clone still points to the position in the run where the corresponding clone_n was performed. Thus, using the collapse operation, we can connect any position simulating a pop_n transition with the position that simulated the corresponding clone_n.

The following proposition provides the detailed construction of the simulating collapsible pushdown system.

Proposition 3.3.3. Let $\mathscr{N}$ be a pushdown system of level $n \geq 2$. We can effectively compute a collapsible pushdown system $\mathscr{S}$ of level $n+1$ and a first-order interpretation $I_{\mathscr{N}}$ such that $\mathsf{NPT}(\mathscr{N})$ is first-order interpretable in $\mathsf{CPG}(\mathscr{S})$ via $I_{\mathscr{N}}$.

Moreover, there is a uniform bound on the length of the formulas of $I_{\mathscr{N}}$ for all higher-order pushdown systems $\mathscr{N}$.

Proof (Sketch). We prove this fact by a straightforward extension of the $n = 1$ case (cf. Lemma 3.2.2). Figure 3.17 illustrates the simulation of a 3-NPT in a collapsible pushdown graph of level 4.

Let $\mathscr{N} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a pushdown system of level $n > 1$ generating $\mathfrak{N} := \mathsf{NPT}(\mathscr{N})$. Then we define a collapsible pushdown system of level $n+1$ $C(\mathscr{N}) := (Q_C, \Sigma_C, \Gamma_C, \Delta_C, I)$ as follows.

- $\Sigma_C := Q \cup \Sigma \cup \{\#\}$ for a new symbol # which is used to simulate the jump-edges.
- $\Gamma_C := \Gamma \cup \{\gamma_{\mathsf{Init}}, \gamma_{\mathsf{Clone}}, \gamma_{\mathsf{Pop}}, \gamma_{\mathsf{Push}}, \gamma_{\mathsf{CPP}}, \gamma_{\mathsf{PP}}, \gamma_{\hookrightarrow}, \gamma_\varepsilon\}$ for new symbols not contained in Γ.
- $Q_C := Q \cup \Sigma \cup \{I, \mathsf{POP}, \mathsf{CLONE}\} \cup \{\mathsf{PUSH}(q) : q \in Q\} \cup \{\mathsf{CPP}(q) : q \in Q\} \cup \{\mathsf{PP}(q) : q \in Q\}$, where I is the new initial state, and the other states are new auxiliary states for the simulation process.

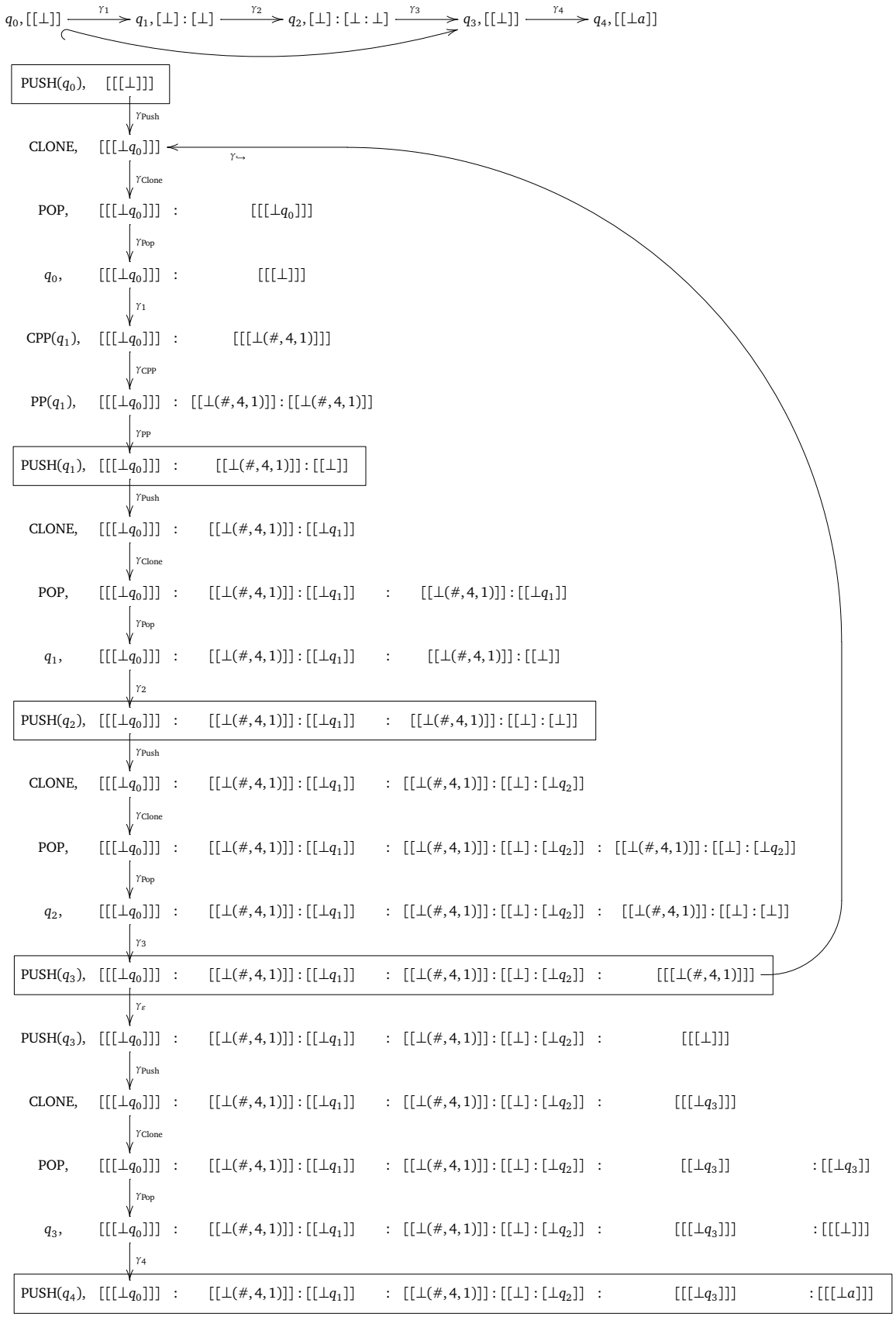


Figure 3.17.: Simulation of 3-NPT in level 4 collapsible pushdown graphs; γ_1 is a clone_3 transition, γ_2 is a clone_2 transition, γ_3 is a pop_3 transition and γ_4 a push_a transition.

- Δ_C consists of the following transitions.
  1. For the initialisation, we add the transition $(I, \bot, \gamma_{\text{Init}}, PUSH(q_0), \text{id}) \in \Delta_C$.
  2. For $q \in Q$ and $\sigma \in \Sigma$, let

$$(PUSH(q), \sigma, \gamma_{\text{Push}}, \text{CLONE}, \text{push}_q),$$
$$(\text{CLONE}, q, \gamma_{\text{Clone}}, \text{POP}, \text{clone}_{n+1}), \text{ and}$$
$$(\text{POP}, q, \gamma_{\text{Pop}}, q, \text{pop}_1)$$

     be in Δ_C.[7] These transitions are auxiliary transitions that write the state of the run onto the topmost level n stack and create a clone of the topmost level n stack preparing the simulation of the next transition.
  3. For $\text{op} \neq \text{clone}_n$ and $(q, \sigma, \gamma, p, \text{op}) \in \Delta$, set $(q, \sigma, \gamma, PUSH(p), \text{op}) \in \Delta_C$.
  4. For $(q, \sigma, \gamma, p, \text{clone}_n) \in \Delta$, set $(q, \sigma, \gamma, \text{CPP}(p), \text{push}_{\#,n+1}) \in \Delta_c$.
  5. We handle the jump-edge marker # with the following transitions. For all $q \in Q$, set

$$(\text{CPP}(q), \#, \gamma_{\text{CPP}}, \text{PP}(q), \text{clone}_n) \in \Delta_C,$$
$$(\text{PP}(q), \#, \gamma_{\text{PP}}, \text{PUSH}(q), \text{pop}_1) \in \Delta_C,$$
$$(\text{PUSH}(q), \#, \gamma_{\varepsilon}, \text{PUSH}(q), \text{pop}_1) \in \Delta_C, \text{ and}$$
$$(\text{PUSH}(q), \#, \gamma_{\hookrightarrow}, \text{CLONE}, \text{collapse}) \in \Delta_C.$$

     The first and the second transition are used to create the jump-edge marker whenever a clone_n is simulated. The third transition is used to remove the marker after the simulation of a pop_n. The last transition is used to simulate the jump-edge.

We use those configurations with state $\text{PUSH}(q)$ for all $q \in Q$ that have no incoming γ_ε-edge as representatives of the runs of $\mathcal{N}$. These configurations are defined by the formula

$$\varphi(x) := \exists y\, x \vdash^{\gamma_\varepsilon} y \vee \left(x \vdash^{\gamma_{\text{Push}}} y \wedge \forall z \neg z \vdash^{\gamma_\varepsilon} x\right).$$

Now, we turn to the formulas that interpret the transitions $\vdash^\gamma$. Let $\rho, \hat{\rho} \in \mathfrak{N}$ be connected by some transition $\delta = (q, \sigma, \gamma, p, \text{op}) \in \Delta$. We denote by ρ' the representative of ρ and by $\hat{\rho}'$ the representative of $\hat{\rho}$ in $C(\mathcal{N})$. We distinguish the following cases.

1. Assume that the last transition of ρ is not a pop_n transition and $\text{op} \neq \text{clone}_n$. Then the transition $\rho \vdash^\gamma \hat{\rho}$ in $\mathfrak{N}$ corresponds to a chain

$$\rho' \vdash^{\gamma_{\text{Push}}} x_1 \vdash^{\gamma_{\text{Clone}}} x_2 \vdash^{\gamma_{\text{Pop}}} x_3 \vdash^\gamma \hat{\rho}'$$

   in $\text{CPG}(C(\mathcal{N}))$.

2. Assume that the last transition of ρ is a pop_n transition and $\text{op} \neq \text{clone}_n$. Then the transition $\rho \vdash^\gamma \hat{\rho}$ in $\mathfrak{N}$ corresponds to a chain

$$\rho' \vdash^{\gamma_\varepsilon} x_4 \vdash^{\gamma_{\text{Push}}} x_1 \vdash^{\gamma_{\text{Clone}}} x_2 \vdash^{\gamma_{\text{Pop}}} x_3 \vdash^\gamma \hat{\rho}'$$

   in $\text{CPG}(C(\mathcal{N}))$.

[7] In the following, we write push_q for $\text{push}_{q,1}$.

3. Assume that the last transition of ρ is not a pop_n transition and $\mathsf{op} = \mathsf{clone}_n$. Then the transition $\rho \vdash^{\gamma} \hat{\rho}$ in $\mathfrak{N}$ corresponds to a chain

$$\rho' \vdash^{\gamma_{\mathrm{Push}}} x_1 \vdash^{\gamma_{\mathrm{Clone}}} x_2 \vdash^{\gamma_{\mathrm{Pop}}} x_3 \vdash^{\gamma} x_5 \vdash^{\gamma_{\mathrm{CPP}}} x_6 \vdash^{\gamma_{\mathrm{PP}}} \hat{\rho}'$$

in $\mathsf{CPG}(C(\mathscr{N}))$.

4. Assume that the last transition of ρ is a pop_n transition and $\mathsf{op} = \mathsf{clone}_n$. Then the transition $\rho \vdash^{\gamma} \hat{\rho}$ corresponds to a chain

$$\rho' \vdash^{\gamma_\varepsilon} x_4 \vdash^{\gamma_{\mathrm{Push}}} x_1 \vdash^{\gamma_{\mathrm{Clone}}} x_2 \vdash^{\gamma_{\mathrm{Pop}}} x_3 \vdash^{\gamma} x_5 \vdash^{\gamma_{\mathrm{CPP}}} x_6 \vdash^{\gamma_{\mathrm{PP}}} \hat{\rho}'$$

in $C(\mathscr{N})$.

Moreover, every chain that starts and ends in nodes defined by φ and that is of one of the forms mentioned in the case distinction corresponds to a transition in $\mathfrak{N}$.

This claim is proved by induction on the length of the shortest path to some node satisfying φ. It is completely analogous to the corresponding proof in Lemma 3.2.2.

Finally, we give an interpretation for the jump-edge relation $\hookrightarrow$. The jump-edges correspond to the edges defined by

$$\varphi_{\hookrightarrow}(x, y) := \exists z (x \vdash^{\gamma_{\mathrm{Push}}} z \wedge y \vdash^{\gamma_{\hookrightarrow}} z).$$

□

The previous proposition shows that higher-order nested pushdown trees are (modulo FO-interpretations) contained in the collapsible pushdown hierarchy. The hierarchy of nested pushdown trees is also an extension of the pushdown tree hierarchy. This is shown in the following lemma.

Lemma 3.3.4. The unfoldings of graphs of level $n-1$ pushdown systems are contained in the n-th level of the nested pushdown tree hierarchy.

Proof. Consider any level $n-1$ pushdown system $\mathscr{S}$ as a level n system that does not use clone_n. Then $\mathscr{S}$ generates a level n nested pushdown tree which coincides with the unfolding of the configuration graph of $\mathscr{S}$. □

Remark 3.3.5. Recall that the unfoldings of higher-order pushdown graphs form the pushdown tree hierarchy. The previous lemma shows that the nested pushdown tree hierarchy is an extension of the pushdown tree hierarchy.

It is an interesting open question what the exact relationship between the hierarchy of pushdown graphs and the hierarchy of nested trees is. Since there are nested pushdown trees that have undecidable MSO-theory (cf. Lemma 2.3.11), the hierarchy of nested pushdown trees is not contained in the hierarchy of pushdown graphs. But it is an open question whether there is some logical interpretation that interprets every nested pushdown tree in some higher-order pushdown graph. Lemma 2.3.11 only implies that there is no 1-dimensional MSO interpretation that interprets nested pushdown trees in higher-order pushdown graphs.

The previous lemma and Proposition 3.3.3 locate the hierarchy of nested pushdown trees between the hierarchy of pushdown trees and the hierarchy of collapsible pushdown graphs. We propose the study of this new hierarchy in order to obtain new insights into the relationship of the hierarchies of collapsible pushdown graphs and higher-order pushdown graphs. In the following, we show that FO model checking on 2-NPT is decidable. Via the interpretation of nested pushdown trees in collapsible pushdown graphs, this can be seen as the first step towards an characterisation of the largest subclass of the class of collapsible pushdown graphs of level 3 on which the FO model checking problem is decidable.

3.3.3 Towards FO Model Checking on Nested Pushdown Trees of Level 2

In the following, we develop an FO model checking algorithm on nested pushdown trees of level 2.

Before we continue, we want to stress that the rest of this chapter deals exclusively with level 2 pushdown systems and not with level 2 collapsible pushdown systems. Thus, stacks do not carry any link structure and the systems never use collapse operations. In this setting, loops and returns play an even more important role than in the setting of collapsible pushdown systems. In runs of pushdown systems of level 2, loops and returns occur almost everywhere in the following sense:

1. every run ρ from some stack s to a substack of $\mathsf{pop}_2(s)$ has an initial part that is a return and
2. every run ρ that starts and ends in stack s and that never visits $\mathsf{pop}_2(s)$ is a loop.

We leave it as an easy exercise to check the correctness of these claims. In the following, we will use these facts without any further explanation.

We want to provide an FO model checking algorithm for the class of nested pushdown trees of level 2. We do this by adapting our approach for first-order model checking on nested pushdown trees of level 1. Fix some pushdown system $\mathcal{N}$ of level 2. We show that every formula of the form $\exists x \varphi$ such that $\mathsf{NPT}(\mathcal{N}), \bar{\rho} \models \exists x \varphi$ has a short witness $\rho \in \mathsf{NPT}(\mathcal{N})$ for the first existential quantification. Here, the size of an element is given by the length of the run of $\mathcal{N}$ representing this element. We consider a run to be short, if its size is bounded in terms of the length of the runs in the tuple $\bar{\rho}$ of parameters.

As in the level 1 case, we prove this dynamic small-witness property via Ehrenfeucht-Fraïssé games. The rough picture of the proof is as follows.

We analyse the α-round Ehrenfeucht-Fraïssé game on two copies of $\mathfrak{N} := \mathsf{NPT}(\mathcal{N})$. We show that Duplicator has a strategy that answers every move of Spoiler by choosing a small element. An element is small if there is a bound on the size of the element in terms of the size of the elements chosen so far in the same copy of $\mathfrak{N}$. Using such a strategy, we obtain a model checking algorithm on nested pushdown trees of level 2 as explained in Section 2.1.1.

On this level of detail, the decidability proof on level 2 is exactly the same as on level 1. But the proof that Duplicator can always choose small runs is completely different.

The main technical tool for this proof is the concept of relevant ancestors. For each element of $\mathfrak{N}$, the relevant l-ancestors are a finite set of initial subruns of this element. Intuitively, the relevant l-ancestors of a run ρ are finitely many ancestors of ρ that give a description of the l-local neighbourhood of ρ. Surprisingly, this finite description is sufficiently complete for the

purpose of preserving partial isomorphisms during the Ehrenfeucht-Fraïssé game. We prove that there is a winning strategy for Duplicator with the following property. Duplicator always chooses small runs whose relevant l-ancestors are isomorphic to the relevant l-ancestors of the element chosen by Spoiler.

In order to find such a strategy for Duplicator, we analyse the structure of relevant ancestors. We show that a relevant ancestor ρ_1 is connected to the next one, say ρ_2, by either a single transition or by a run ρ of a certain kind. This run ρ satisfies the following conditions: ρ_2 decomposes as $\rho_2 = \rho_1 \circ \rho$, the initial stack of ρ is $s : w$ where s is some stack and w is some word. The final stack of ρ is $s : w : v$ for some word v and ρ does never pass a proper substack of $s : w$.

Due to this result, a typical set of relevant ancestors is of the form

$$\rho_1 \prec \rho_2 \prec \rho_3 \prec \cdots \prec \rho_m = \rho,$$

where ρ_{n+1} extends ρ_n by either one transition or by a run that extends the last stack of ρ_n by a new word v. If we want to construct a run ρ' with isomorphic relevant ancestor set, we have to provide runs

$$\rho'_1 \prec \rho'_2 \prec \rho'_3 \prec \cdots \prec \rho'_m = \rho'$$

where ρ'_{n+1} extends ρ'_n in exactly the same manner as ρ_{n+1} extends ρ_n.

We first concentrate on one step of this construction. Assume that ρ_1 ends in some configuration $(q, s : w)$ and ρ_2 extends ρ_1 by a run creating the stack $s : w : v$. How can we find another stack s' and words w', v' such that there is a run ρ'_1 to $(q, s' : w')$ and a run ρ'_2 that extends ρ'_1 by a run from $(q, s' : w')$ to the stack $s' : w' : v'$?

We introduce a family of equivalence relations on words that preserves the existence of such runs. If we find some w' that is equivalent to w with respect to the i-th equivalence relation, then for any run from $s : w$ to $s : w : v$ we can find a run from $s' : w'$ to $s' : w' : v'$ for v and v' equivalent with respect to the $(i-1)$-st equivalence relation.

Let us explain the ingredients of these equivalence relations. Let ρ_1 be a run to some stack $s : w$ and let ρ_2 be a run that extends ρ_1 and ends in a stack $s : w : v$. Recall that the theory of generalised milestones shows that the final segment of ρ_2 is of the form

$$\lambda_n \circ \mathrm{op}_n \circ \lambda_{n-1} \circ \mathrm{op}_{n-1} \circ \cdots \circ \mathrm{op}_1 \circ \lambda_0$$

where the λ_i are loops and $\mathrm{op}_n, \mathrm{op}_{n-1}, \ldots, \mathrm{op}_1$ is the minimal sequence generating $s : w : v$ from $s : w$. Thus, we are especially interested in the loops of each prefix $\mathrm{pop}_1^k(w)$ of w and each prefix $\mathrm{pop}_1^k(w')$ of w'. For this purpose we consider the word models of w and w' enriched by information on runs between certain prefixes of w or w'. Especially, each prefix is annotated with the number of possible loops of each prefix. w and w' are equivalent with respect to the first equivalence relation if the FO_k-types of their enriched word structures coincide. The second, third, etc. equivalence relation is then defined as follows. We enrich every element of the word model of some word w by the equivalence class of the corresponding prefix with respect to the $(i-1)$-st equivalence relation. The i-th equivalence relation then compares the FO_k-types of these enriched word models. This means that two words w and w' are equivalent with respect to the i-th equivalence relation if the FO_k-types of their word models enriched with the $(i-1)$-st equivalence class of each prefix coincide.

This iteration of equivalence of prefixes leads to the following result. Let w and w' be equivalent with respect to the i-th relation. Then we can transfer runs creating i words in the following sense: if ρ is a run creating $w : v_1 : v_2 : \cdots : v_i$ from w, then there is a run ρ' creating $w' : v_1' : v_2' : \cdots : v_i'$ from w' such that v_k and v_k' are equivalent with respect to the $(i-k)$-th relation. This property then allows to construct isomorphic relevant ancestors for a given set of relevant ancestors of some run ρ. We only have to start with a stack $s' : w'$ such that w' is i-equivalent to the topmost word of the minimal element of the relevant ancestors of ρ for some large $i \in \mathbb{N}$.

This observation reduces the problem of constructing runs with isomorphic relevant ancestors to the problem of finding runs whose last configurations have equivalent topmost words (with respect to the i-th equivalence relation for some sufficiently large i) such that one of these runs is always short.

We solve this problem by application of several pumping constructions that respect the equivalence class of the topmost word of the final configuration of a run but which decrease the length of the run.

Putting all these results together, we obtain that Duplicator has an S-preserving strategy on every nested pushdown tree of level 2 where S is a finitary constraint bounding the length of the runs that Duplicator may choose. Then we use the general model checking algorithm from Section 2.1.1 in order to solve the FO model checking problem on nested pushdown trees of level 2.

The outline of the next sections is as follows. In Section 3.3.4 we define the important notion of relevant ancestors and develop some theory concerning these sets. We then define a family of equivalence relations on words and stacks in Section 3.3.5. In Section 3.3.6 we put these things together: the equivalence on stacks gives us a transfer property of relevant ancestors to isomorphic copies. Our analysis of loops (cf. Section 2.4) yields the possibility to bound the length of the runs involved in the isomorphic copy. Thus, preserving isomorphisms between relevant ancestors while choosing small runs is a valid strategy for Duplicator in the Ehrenfeucht-Fraïssé game. This gives us a small-witness property which we use to show the decidability of FO model checking on 2-NPT in Section 3.3.7.

3.3.4 Relevant Ancestors

This section aims at identifying those ancestors of a run ρ in a 2-NPT $\mathfrak{N}$ that are relevant with respect to its FO_k-type. We show that only finitely many ancestors of a certain kind fix the FO_k-type of the l-local neighbourhood of ρ. We call these finitely many ancestors the relevant l-ancestors of ρ.

Before we formally introduce relevant ancestors, we recall some important abbreviations concerning configurations and runs. Abusing notations we apply functions defined on stacks to configurations. For example if $c = (q, s)$ we write $|c|$ for $|s|$ or $\mathsf{pop}_2(c)$ for $\mathsf{pop}_2(s)$.

We further abuse this notation by application of functions defined on stacks to some run ρ, meaning that we apply the function to the last stack occurring in ρ. For example, we write $\mathsf{top}_2(\rho)$ for $\mathsf{top}_2(s)$ and $|\rho|$ for $|s|$ if $\rho(\mathsf{ln}(\rho)) = (q, s)$.

In the same sense one has to understand equations like $\rho(i) = \mathsf{pop}_1(s)$. This equation says that $\rho(i) = (q, \mathsf{pop}_1(s))$ for some $q \in Q$. Keep in mind that $|\rho|$ denotes the width of the last stack of ρ and not the length $\mathsf{ln}(\rho)$ of the run ρ. Recall also that we write $\rho \preceq \rho'$ if the run ρ is an initial segment of the run ρ'.

Definition 3.3.6. Let $\mathfrak{N}$ be some 2-NPT. Define the relation $\overset{+1}{\hookrightarrow} \subseteq \mathfrak{N} \times \mathfrak{N}$ by

$$\rho \overset{+1}{\hookrightarrow} \rho' \text{ if } \rho \prec \rho', |\rho| = |\rho'| - 1, \text{ and } |\pi| > |\rho| \text{ for all } \rho \prec \pi \prec \rho'.$$

We define the relevant l-ancestors of ρ by induction on l. The relevant 0-ancestors of ρ are the elements of the set $\mathrm{RA}_0(\rho) := \{\rho\}$. Inductively, we set

$$\mathrm{RA}_{l+1}(\rho) := \mathrm{RA}_l(\rho) \cup \left\{ \pi \in \mathfrak{N} : \exists \pi' \in \mathrm{RA}_l(\rho)\ \pi \vdash \pi' \text{ or } \pi \hookrightarrow \pi' \text{ or } \pi \overset{+1}{\hookrightarrow} \pi' \right\}.$$

If $\bar{\rho} = (\rho_1, \rho_2, \ldots, \rho_n)$ then we write $\mathrm{RA}_l(\bar{\rho}) := \bigcup_{i=1}^{n} \mathrm{RA}_l(\rho_i)$.

Remark 3.3.7. Note that for each ρ' there is at most one ρ such that $\rho \overset{+1}{\hookrightarrow} \rho'$ while ρ may have arbitrary many $\overset{+1}{\hookrightarrow}$ successors along each branch.

The relation $\overset{+1}{\hookrightarrow}$ can be characterised as follows: For runs ρ, ρ', it holds that $\rho \overset{+1}{\hookrightarrow} \rho'$ if and only if $\rho' = \rho \circ \pi$ for some run π starting at some stack s_ρ and ending in some stack $s_\rho : w$, the first operation of π is a clone and π visits s_ρ only in its initial configuration.

The motivation for these definitions is the following. If there are elements $\rho, \rho' \in \mathfrak{N}$ such that $\rho' \preceq \rho$ and there is a path in $\mathfrak{N}$ of length at most l that witnesses that ρ' is an ancestor of ρ, then we want that $\rho' \in \mathrm{RA}_l(\rho)$. The relation $\overset{+1}{\hookrightarrow}$ is tailored towards this idea. Assume that there are runs $\rho_1 \prec \rho_2 \vdash^{\mathrm{pop}_2} \rho_3$ such that $\rho_2 \vdash^{\mathrm{pop}_2} \rho_3 \hookleftarrow \rho_1$. This path of length 2 witnesses that ρ_1 is a predecessor of ρ_2. By definition, one sees immediately that $\rho_1 \overset{+1}{\hookrightarrow} \rho_2$ whence $\rho_1 \in \mathrm{RA}_1(\rho_2)$. In this sense, $\overset{+1}{\hookrightarrow}$ relates the ancestor ρ_1 of ρ_2 with ρ_2 if ρ_1 may be reachable from ρ_2 via a short path passing a descendant of ρ_2.

In the following, it may be helpful to think of a relevant l-ancestor ρ' of a run ρ as an ancestor of ρ that may have a path of length up to l witnessing that ρ' is an ancestor of ρ. We do not state this idea more precisely, but it may be helpful to keep this picture in mind.

From the definitions, we obtain immediately the following lemmas.

Lemma 3.3.8. Let ρ and ρ' be runs such that $\rho \hookrightarrow \rho'$. Let $\hat{\rho}$ be the predecessor of ρ', i.e., $\hat{\rho}$ is the unique element such that $\hat{\rho} \vdash \rho'$. Then $\rho \overset{+1}{\hookrightarrow} \hat{\rho}$.

Lemma 3.3.9. If $\rho, \rho' \in \mathfrak{N}$ are connected by a single edge $\vdash$ or $\hookrightarrow$ then either $\rho \in \mathrm{RA}_1(\rho')$ or $\rho' \in \mathrm{RA}_1(\rho)$.

Lemma 3.3.10. For all $l \in \mathbb{N}$ and $\rho \in \mathfrak{N}$, $|\mathrm{RA}_l(\rho)| \leq 4^l$.

Lemma 3.3.11. $\mathrm{RA}_l(\rho)$ is linearly ordered by $\preceq$.

Proof. By induction, one obtains easily that $\mathrm{RA}_l(\rho)$ only contains initial segments of the run ρ. These are obviously ordered linearly by $\preceq$. □

In the following we investigate the relationship between relevant ancestors of different runs. First, we characterise the minimal element of $\mathrm{RA}_l(\rho)$.

Lemma 3.3.12. Let $\rho_l \in \mathrm{RA}_l(\rho)$ be minimal with respect to $\preceq$.

$$\text{Either } |\rho_l| = 1 \text{ and } |\rho| \leq l,$$
$$\text{or } \rho_l = \mathrm{pop}_2^l(\rho) \text{ and } |\rho_l| < |\rho'| \text{ for all } \rho' \in \mathrm{RA}_l(\rho) \setminus \{\rho_l\}.$$

Remark 3.3.13. Recall that $|\rho| \leq l$ implies that $\mathrm{pop}_2^l(\rho)$ is undefined.

Proof. The proof is by induction on l. For $l = 0$, there is nothing to show because $\rho_0 = \rho = \mathrm{pop}_2^0(\rho)$. Now assume that the statement is true for some l.

Assume that $|\rho| \leq l + 1$. Then ρ_l satisfies $|\rho_l| = 1$. If ρ_l has no predecessor it is also the minimal element of $\mathrm{RA}_{l+1}(\rho)$ and we are done. Otherwise, there is a maximal ancestor $\hat{\rho} \prec \rho_l$ such that $|\hat{\rho}| = 1$. Either $\hat{\rho} \vdash \rho_l$ or $\hat{\rho} \hookrightarrow \rho_l$ whence $\hat{\rho} \in \mathrm{RA}_{l+1}(\rho)$. Furthermore, no ancestor of $\hat{\rho}$ can be contained in $\mathrm{RA}_{l+1}(\rho)$. We prove this claim by contradiction.

Assume that there is some element $\tilde{\rho} \prec \hat{\rho}$ such that $\tilde{\rho} \in \mathrm{RA}_{l+1}(\rho)$. Then there is some $\tilde{\rho}' \in \mathrm{RA}_l(\rho)$ such that $\tilde{\rho}$ and $\tilde{\rho}'$ are connected by some edge. Due to the definition of $\hat{\rho}$, we have $\tilde{\rho} \prec \hat{\rho} \prec \tilde{\rho}'$. Thus, the edge between $\tilde{\rho}$ and $\tilde{\rho}'$ has to be $\hookrightarrow$ or $\overset{+1}{\hookrightarrow}$. Thus, $\tilde{\rho}$ must have width less than $\hat{\rho}$, i.e., width 0. Since there are no stacks of width 0, this is a contradiction.

Thus, the minimal element of $\mathrm{RA}_{l+1}(\rho)$ is $\rho_{l+1} = \hat{\rho}$. This completes the case $|\rho| \leq l + 1$.

Now assume that $|\rho| > l+1$. Let $\hat{\rho}$ be the maximal ancestor of ρ_l such that $|\hat{\rho}|+1 = |\rho_l|$. Then $\hat{\rho} \overset{+1}{\hookrightarrow} \rho_l$ or $\hat{\rho} \vdash^{\gamma} \rho_1$, whence $\hat{\rho} \in \mathrm{RA}_{l+1}(\rho)$. We have to show that $\hat{\rho}$ is the minimal element of $\mathrm{RA}_{l+1}(\rho)$ and that there is no other element of width $|\hat{\rho}|$ in $\mathrm{RA}_{l+1}(\rho)$. For the second part, assume that there is some $\rho' \in \mathrm{RA}_{l+1}(\rho)$ with $|\rho'| = |\hat{\rho}|$. Then ρ' has to be connected via $\vdash, \overset{+1}{\hookrightarrow}$, or $\hookrightarrow$ to some element $\rho'' \in \mathrm{RA}_l(\rho)$. By definition of these relations $|\rho''| \leq |\rho'| + 1$. By induction hypothesis, this implies $\rho'' = \rho_l$. But then it is immediately clear that $\rho' = \hat{\rho}$ by definition.

Similar to the previous case, the minimality of $\hat{\rho}$ in $\mathrm{RA}_{l+1}(\rho)$ is proved by contradiction. Assume that there is some $\rho' \prec \hat{\rho}$ such that $\rho' \in \mathrm{RA}_{l+1}(\rho)$. Then there is some $\hat{\rho} \prec \rho_l \preceq \rho'' \in \mathrm{RA}_l(\rho)$ such that $\rho' \overset{+1}{\hookrightarrow} \rho''$ or $\rho' \hookrightarrow \rho''$. By the definition of $\hookrightarrow$ and $\overset{+1}{\hookrightarrow}$, we obtain $|\rho''| \leq |\hat{\rho}|$. But this contradicts $|\rho''| \geq |\rho_l| > |\hat{\rho}|$. Thus, we conclude that $\hat{\rho}$ is the minimal element of $\mathrm{RA}_{l+1}(\rho)$, i.e., $\hat{\rho} = \rho_{l+1}$. □

The previous lemma shows that the width of stacks among the relevant ancestors cannot decrease too much. Furthermore, the width cannot grow too much. This is shown in the following corollary.

Corollary 3.3.14. Let $\pi, \rho \in \mathfrak{N}$ such that $\pi \in \mathrm{RA}_l(\rho)$. Then $\big||\rho| - |\pi|\big| \leq l$.

Proof. From the previous lemma, we know that the minimal width of the last stack of an element in $\mathrm{RA}_l(\rho)$ is $|\rho| - l$. We prove by induction that the maximal width is $|\rho| + l$. The case $l = 0$ is trivially true. Assume that $|\pi| \leq |\rho| + l - 1$ for all $\pi \in \mathrm{RA}_{l-1}(\rho)$. Let $\hat{\pi} \in \mathrm{RA}_l(\rho) \setminus \mathrm{RA}_{l-1}(\rho)$. Then there is a $\pi \in \mathrm{RA}_{l-1}(\rho)$ such that $\hat{\pi} \vdash \pi$, $\hat{\pi} \hookrightarrow \pi$, or $\hat{\pi} \overset{+1}{\hookrightarrow} \pi$. In the last two cases the width of $\hat{\pi}$ is smaller than the width of π whence $|\hat{\pi}| \leq |\rho| + l - 1$. In the first case, recall that all stack operations of an level 2 higher order pushdown system alter the width of the stack by at most 1. Thus, $|\hat{\pi}| \leq |\pi| + 1 \leq |\rho| + l$. □

The next lemma shows a kind of triangle inequality of the relevant ancestor relation. If ρ_2 is a relevant ancestor of ρ_1 then all relevant ancestors of ρ_1 that are prefixes of ρ_2 are relevant ancestors of ρ_2.

Lemma 3.3.15. Let $\rho_1, \rho_2 \in \mathfrak{N}$ and let $l_1, l_2 \in \mathbb{N}$. If $\rho_1 \in \mathrm{RA}_{l_1}(\rho_2)$, then

$$\mathrm{RA}_{l_2}(\rho_1) \subseteq \mathrm{RA}_{l_1+l_2}(\rho_2) \text{ and}$$
$$\mathrm{RA}_{l_2}(\rho_2) \cap \{\pi : \pi \preceq \rho_1\} \subseteq \mathrm{RA}_{l_1+l_2}(\rho_1).$$

Proof. The first relation holds directly because of the inductive definition of relevant ancestors.

For the second claim, we proceed by induction on l_2. For $l_2 = 0$ the claim holds because $\mathrm{RA}_0(\rho_2) = \{\rho_2\}$ and $\rho_1 \preceq \rho_2$ imply that $\mathrm{RA}_0(\rho_2) \cap \{\pi : \pi \preceq \rho_1\} \neq \emptyset$ if and only if $\rho_1 = \rho_2$ and $\{\rho_2\} \in \mathrm{RA}_0(\rho_1)$.

For the induction step assume that

$$\mathrm{RA}_{l_2-1}(\rho_2) \cap \{\pi : \pi \preceq \rho_1\} \subseteq \mathrm{RA}_{l_1+l_2-1}(\rho_1).$$

Furthermore, assume that $\pi \in \mathrm{RA}_{l_2}(\rho_2) \cap \{\pi : \pi \preceq \rho_1\}$. We show that $\pi \in \mathrm{RA}_{l_1+l_2}(\rho_1)$. By definition there is some $\pi \prec \hat{\pi}$ such that $\hat{\pi} \in \mathrm{RA}_{l_2-1}(\rho_2)$ and $\pi \in \mathrm{RA}_1(\hat{\pi})$. We distinguish the following cases.

- Consider the case $\hat{\pi} \preceq \rho_1$. Due to the induction hypothesis, $\hat{\pi} \in \mathrm{RA}_{l_1+l_2-1}(\rho_1)$. Thus, $\pi \in \mathrm{RA}_{l_1+l_2}(\rho_1)$.
- Consider the case $\hat{\pi} = \rho_1$. Then $\pi \in \mathrm{RA}_1(\rho_1) \subseteq \mathrm{RA}_{l_1+l_2}(\rho_1)$.
- Finally, consider the case $\pi \prec \rho_1 \prec \hat{\pi} \prec \rho_2$. This implies that $\pi \hookrightarrow \hat{\pi}$ or $\pi \overset{+1}{\hookrightarrow} \hat{\pi}$ whence $|\pi| = |\hat{\pi}| - j < |\rho_1|$ for some $j \in \{0, 1\}$. From Corollary 3.3.14, we know that

  $$\big||\hat{\pi}| - |\rho_2|\big| \leq l_2 - 1 \text{ and } \big||\rho_1| - |\rho_2|\big| \leq l_1.$$

  This implies that $|\rho_1| - |\pi| \leq l_1 + l_2$. By definition of $\hookrightarrow$ and $\overset{+1}{\hookrightarrow}$, there cannot be any element $\pi \prec \pi' \prec \hat{\pi}$ with $|\pi'| = |\pi|$. Thus, π is the maximal predecessor of ρ_1 with $\pi = \mathrm{pop}_2^{|\rho_1|-|\pi|}(\rho_1)$. Application of Lemma 3.3.12 shows that π is the minimal element of $\mathrm{RA}_{|\rho_1|-|\pi|}(\rho_1)$. Hence,

  $$\pi \in \mathrm{RA}_{|\rho_1|-|\pi|}(\rho_1) \subseteq \mathrm{RA}_{l_1+l_2}(\rho_1).$$

□

Corollary 3.3.16. For $\rho \in \mathrm{RA}_l(\rho_1) \cap \mathrm{RA}_l(\rho_2)$, we have $\mathrm{RA}_l(\rho_1) \cap \{\pi : \pi \preceq \rho\} \subseteq \mathrm{RA}_{3l}(\rho_2)$.

Proof. By the previous lemma, $\rho \in \mathrm{RA}_l(\rho_1)$ implies $\mathrm{RA}_l(\rho_1) \cap \{\pi : \pi \preceq \rho\} \subseteq \mathrm{RA}_{2l}(\rho)$. Using the lemma again, $\rho \in \mathrm{RA}_l(\rho_2)$ implies $\mathrm{RA}_{2l}(\rho) \subseteq \mathrm{RA}_{3l}(\rho_2)$. □

The previous corollary shows that if the relevant l-ancestors of two elements ρ_1 and ρ_2 intersect at some point ρ, then all relevant l-ancestors of ρ_1 that are ancestors of ρ are contained in the relevant $3l$-ancestors of ρ_2. Later, we will use the contraposition of this result in order to prove that relevant ancestors of certain runs are disjoint sets.

The following proposition describes how $\mathrm{RA}_l(\rho)$ embeds into the full 2-NPT $\mathfrak{N}$. Successive relevant ancestors of some run ρ are either connected by a single edge or by a $\overset{+1}{\hookrightarrow}$-edge. Later, we will see that this proposition allows to explicitly construct for any run ρ an isomorphic relevant ancestor set that consists of small runs.

Proposition 3.3.17. Let $\rho_1 \prec \rho_2 \prec \rho$ such that $\rho_1, \rho_2 \in \mathrm{RA}_l(\rho)$. If $\pi \notin \mathrm{RA}_l(\rho)$ for all $\rho_1 \prec \pi \prec \rho_2$, then either $\rho_1 \vdash \rho_2$ or $\rho_1 \overset{+1}{\hookrightarrow} \rho_2$.

Proof. Assume that $\rho_1 \nvdash \rho_2$. Consider the set

$$M := \{\pi \in \mathrm{RA}_l(\rho) : \rho_1 \overset{+1}{\hookrightarrow} \pi\}.$$

M is nonempty because there is some $\pi \in \mathrm{RA}_{l-1}(\rho)$ such that either $\rho_1 \overset{+1}{\hookrightarrow} \pi$ (whence $\pi \in M$) or $\rho_1 \hookrightarrow \pi$ (whence the predecessor $\hat{\pi}$ of π satisfies $\hat{\pi} \in M$). Let $\hat{\rho} \in M$ be minimal. It suffices to show that $\hat{\rho} = \rho_2$. For this purpose, we show that $\pi \notin \mathrm{RA}_l(\rho)$ for all $\rho_1 \prec \pi \prec \hat{\rho}$. Since $\hat{\rho} \in \mathrm{RA}_l(\rho)$, this implies that $\hat{\rho} = \rho_2$.

We start with two general observations.

1. For all $\rho_1 \prec \pi \prec \hat{\rho}$, $|\pi| \geq |\hat{\rho}|$ due to the definition of $\rho_1 \overset{+1}{\hookrightarrow} \hat{\rho}$. Furthermore, due to the minimality of $\hat{\rho}$ in M, for all $\rho_1 \prec \pi \prec \hat{\rho}$ with $\pi \in \mathrm{RA}_l(\rho)$, $|\pi| > |\hat{\rho}|$ (otherwise we have $\pi \in M$ contradicting the minimality of $\hat{\rho}$).

2. Note that there cannot exist $\rho_1 \prec \pi \prec \hat{\rho} \prec \hat{\pi}$ with $\pi \hookrightarrow \hat{\pi}$ or $\pi \overset{+1}{\hookrightarrow} \hat{\pi}$ because $|\pi| \geq |\hat{\rho}|$.

Heading for a contradiction, assume that there is some $\rho_1 \prec \pi \prec \hat{\rho}$ such that $\pi \in \mathrm{RA}_l(\rho)$.

Due to observation 2, there is a chain $\pi_0 := \pi, \pi_1, \ldots, \pi_{n-1}, \pi_n := \hat{\rho}$ such that for each $0 \leq i < n$ there is $* \in \{\vdash, \hookrightarrow, \overset{+1}{\hookrightarrow}\}$ such that $\pi_i * \pi_{i+1}$ and $\pi_i \in \mathrm{RA}_{l-i}(\rho)$. By assumption, $n \neq 0$, whence $\hat{\rho} \in \mathrm{RA}_{l-1}(\rho)$. Due to observation 1, we have $|\rho_1| < |\hat{\rho}| < |\pi|$. Since each stack operation alters the width of the stack by at most 1, we conclude that the set

$$M' := \{\pi' : \rho_1 \prec \pi' \prec \hat{\rho}, |\hat{\rho}| = |\pi'|\}$$

is nonempty because on the path from ρ_1 to π there occurs at least one run with final stack of width $|\hat{\rho}|$. But the maximal element $\pi' \in M'$ satisfies $\rho_1 \overset{+1}{\hookrightarrow} \pi' \vdash \hat{\rho}$ or $\rho_1 \overset{+1}{\hookrightarrow} \pi' \hookrightarrow \hat{\rho}$. Since $\hat{\rho} \in \mathrm{RA}_{l-1}(\rho)$, this would imply $\pi' \in M$ which contradicts the minimality of $\hat{\rho}$ in M. Thus, no $\rho_1 \prec \pi \prec \hat{\rho}$ with $\pi \in \mathrm{RA}_l(\rho)$ can exist.

Thus, $\pi \notin \mathrm{RA}_l(\rho)$ for all $\rho_1 \prec \pi \prec \hat{\rho}$ and $\rho_1 \overset{+1}{\hookrightarrow} \hat{\rho} = \rho_2$. □

In the final part of this section, we consider relevant ancestors of two different runs ρ and ρ'. Since we aim at a construction of small runs $\hat{\rho}$ and $\hat{\rho}'$ such that the relevant ancestors of ρ and ρ' are isomorphic to the relevant ancestors of $\hat{\rho}$ and $\hat{\rho}'$, we need to know how sets of relevant ancestors touch each other. Every isomorphism from the relevant ancestors of ρ and ρ' to those of $\hat{\rho}$ and $\hat{\rho}'$ has to preserve edges between a relevant ancestor of ρ and another one of ρ'.

The positions where the relevant l-ancestors of ρ and $\hat{\rho}$ touch can be identified by looking at the intersection of their relevant $(l+1)$-ancestors. This is shown in the following Lemma. For A and B subsets of some 2-NPT $\mathfrak{N}$ and ρ some run of $\mathfrak{N}$, we say A and B touch after ρ if there are runs $\rho \prec \rho_A, \rho \prec \rho_B$ such that $\rho_A \in A$, $\rho_B \in B$ and either $\rho_A = \rho_B$ or $\rho_A * \rho_B$ for some $* \in \{\vdash, \dashv, \hookrightarrow, \hookleftarrow\}$. In this case we say A and B touch at (ρ_A, ρ_B). In the following, we reduce the question whether l-ancestors of two elements touch after some ρ to the question whether the $(l+1)$-ancestors of these elements intersect after ρ.

Lemma 3.3.18. If ρ_1, ρ_2 are runs such that $\mathrm{RA}_{l_1}(\rho_1)$ and $\mathrm{RA}_{l_2}(\rho_2)$ touch after some ρ_0, then $\mathrm{RA}_{l_1+1}(\rho_1) \cap \mathrm{RA}_{l_2+1}(\rho_2) \cap \{\pi : \rho_0 \preceq \pi\} \neq \emptyset$.

Proof. Let ρ_0 be some run, $\rho_0 \prec \hat{\rho}_1 \in \mathrm{RA}_{l_1}(\rho_1)$, and $\rho_0 \prec \hat{\rho}_2 \in \mathrm{RA}_{l_2}(\rho_2)$ such that the pair $(\hat{\rho}_1, \hat{\rho}_2)$ is minimal and $\mathrm{RA}_{l_1}(\rho_1)$ and $\mathrm{RA}_{l_2}(\rho_2)$ touch at $(\hat{\rho}_1, \hat{\rho}_2)$. Then one of the following holds.

1. $\hat{\rho}_1 = \hat{\rho}_2$: there is nothing to prove because $\hat{\rho}_1 \in \mathrm{RA}_{l_1}(\rho_1) \cap \mathrm{RA}_{l_2}(\rho_2) \cap \{\pi : \rho_0 \preceq \pi\}$.

2. $\hat{\rho}_1 \to \hat{\rho}_2$ or $\hat{\rho}_1 \hookrightarrow \hat{\rho}_2$ or $\hat{\rho}_1 \overset{+1}{\hookrightarrow} \hat{\rho}_2$: this implies that $\hat{\rho}_1 \in \mathrm{RA}_{l_2+1}(\rho_2) \cap \mathrm{RA}_{l_1}(\rho_1)$.

3. $\hat{\rho}_2 \to \hat{\rho}_1$ or $\hat{\rho}_2 \hookrightarrow \hat{\rho}_1$ or $\hat{\rho}_2 \overset{+1}{\hookrightarrow} \hat{\rho}_1$: this implies that that $\hat{\rho}_2 \in \mathrm{RA}_{l_1+1}(\rho_1) \cap \mathrm{RA}_{l_2}(\rho_2)$.

□

Corollary 3.3.19. If ρ and ρ' are runs such that $\mathrm{RA}_{l_1}(\rho)$ and $\mathrm{RA}_{l_2}(\rho')$ touch after some run ρ_0 then there exists some $\rho_0 \prec \rho_1 \in \mathrm{RA}_{l_1+1}(\rho) \cap \mathrm{RA}_{l_2+1}(\rho')$ such that

$$\mathrm{RA}_{l_1+1}(\rho) \cap \{x : x \preceq \rho_1\} \subseteq \mathrm{RA}_{l_2+2l_1+3}(\rho').$$

Proof. Use the previous lemma and Lemma 3.3.15. □

3.3.5 A Family of Equivalence Relations on Words and Stacks

In this section we introduce a family of equivalence relations on words. The basic idea is to classify words according to the FO_k-type of the word model associated to the word w enriched by information about certain runs between prefixes of w. This additional information describes

1. the number of possible loops and returns with certain initial and final state of each prefix $v \leq w$, and

2. the number of runs from (q, w) to (q', v) for each prefix $v \leq w$ and all pairs q, q' of states.

It turns out that this equivalence has the following property: if w and w' are equivalent and ρ is a run starting in (q, w) and ending in $(q', w : v)$, then there is a run from (q, w') to $(q, w' : v')$ such that the loops and returns of v and v' agree. This is important because runs of this kind connect consecutive elements of relevant ancestor sets (cf. Proposition 3.3.17).

In order to copy relevant ancestors, we want to apply this kind of transfer property iteratively, e.g., we want to take a run from (q_1, w_1) via $(q_2, w_1 : w_2)$ to $(q_3, w_1 : w_2 : w_3)$ and translate it into some run from (q_1, w_1') via $(q_2, w_1' : w_2')$ to $(q_3, w_1' : w_2' : w_3')$ such that the loops and returns of w_3 and w_3' agree. Analogously, we want to take a run creating n new words and transfer it to a new run starting in another word and creating n words such that the last words agree on their loops and returns. If we can do this, then we can transfer the whole set of relevant ancestors from some run to another one. Using the results of Section 2.4, this allows us to construct isomorphic relevant ancestors that consist only of short runs.

The family of equivalence relations that we define have the following transfer property. Words that are equivalent with respect to the n-th relation allow a transfer of runs creating

n new words. The idea of the definition is as follows. Assume that we have already defined the $(i-1)$-st equivalence relation. We take the word model of some word w and annotate each prefix of the word by its equivalence class with respect to the $(i-1)$-st relation. Then we define two words to be equivalent with respect to the i-th relation if the FO_k-types of their enriched word models agree.

These equivalence relations and the transfer properties that they induce are an important tool in the next section. There we apply them to an arbitrary set of relevant ancestors S in order to obtain isomorphic copies of the substructure induced by S. For the next definition, recall that w_{-n} is an abbreviation for $\mathrm{pop}_1^n(w)$.

Definition 3.3.20. Fix a level 2 pushdown system $\mathcal{N}$. Let $w \in \Sigma^*$ be some word. We are going to define expanded word models $\mathfrak{Lin}_n^{k;z}(w)$ by induction on n. Note that for $n = 0$ the structure will be independent of the parameter k but for greater n this parameter influences with which kind of information the structure is enriched. Let $\mathfrak{Lin}_0^{k;z}(w)$ be the expanded word model

$$\mathfrak{Lin}_0^{k;z}(w) := (\{0,1,\dots,|w|-1\}, \mathrm{succ}, (P_\sigma)_{\sigma\in\Sigma}, (S^j_{q,q'})_{(q,q')\in Q^2, j\le z}, (R_j)_{j\in J}, (L_j)_{j\in J}, (H_j)_{j\in J})$$

such that for $0 \le i < |w|$ the following holds.

- succ and P_σ form the standard word model of w in reversed order, i.e., succ is the successor relation on the domain and $i \in P_\sigma$ if and only if $\mathrm{top}_1(w_{-i}) = \sigma$,
- $i \in S^j_{q,q'}$, if there are j pairwise distinct runs $\rho_1, \dots, \rho_j$ starting in (q, w) and ending in (q', w_{-i}) such that for all $1 \le k \le j$ and $0 \le l < \mathrm{ln}(\rho_k)$ the stack at $\rho_k(l)$ is not w_{-i}.
- The predicates R_j encode at every position i the function $\#\mathrm{Ret}^z(w_{-i})$ (cf. Definition 2.4.15).
- The predicates L_j encode at every position i the function $\#\mathrm{Loop}^z(w_{-i})$ (cf. Definition 2.4.46).
- The predicates H_j encode at every position i the function $\#\mathrm{HLoop}^z(w_{-i})$.

Now, set $\mathrm{Type}_0^{k;z}(w) := \mathrm{FO}_k[\mathfrak{Lin}_0^{k;z}(w)]$, the quantifier rank k theory of $\mathfrak{Lin}_0^{k;z}(w)$. We call it the $(0,k,z)$-type of w. Note that there are only finitely many $(0,k,z)$-types (cf. example 2.1.8).

Inductively, we define $\mathfrak{Lin}_{n+1}^{k;z}(w)$ to be the expansion of $\mathfrak{Lin}_n^{k;z}(w)$ by predicates describing $\mathrm{Type}_n^{k;z}(v)$ for each prefix $v \le w$. More formally, fix a maximal list $\theta_1, \theta_2, \dots, \theta_m$ of pairwise distinct FO_k-types that are realised by some $\mathfrak{Lin}_n^{k;z}(w)$. We define predicates $T_1, T_2, \dots, T_m$ such that $i \in T_j$ if $\mathrm{Type}_n^{k;z}(w_{-i}) = \theta_j$ for all $0 \le i \le n$. Now, let $\mathfrak{Lin}_{n+1}^{k;z}(w)$ be the expansion of $\mathfrak{Lin}_n^{k;z}(w)$ by the predicates $T_1, T_2, \dots, T_m$. We conclude the inductive definition by setting $\mathrm{Type}_{n+1}^{k;z}(w) := \mathrm{FO}_k[\mathfrak{Lin}_{n+1}^{k;z}(w)]$.

Remark 3.3.21. Each element of $\mathfrak{Lin}_n^{k;z}(w)$ corresponds to a prefix of w. In this sense, we write $v \in S^j_{q,q'}$ for some prefix $v \le w$ if $v = w_{-l}$ and $\mathfrak{Lin}_n^{k;z}(w) \models l \in S^j_{q,q'}$.

It is an important observation that $\mathfrak{Lin}_n^{k;z}(w)$ is a finite successor structure with finitely many colours. Thus, there are only finitely many (n,k,z)-types for each $n,k,z \in \mathbb{N}$ (cf. Example 2.1.8).

For our application, k and z can be chosen to be some fixed large numbers, depending on the number of rounds we are going to play in the Ehrenfeucht-Fraïssé game. Furthermore, it will turn out that the conditions on k and z coincide whence we will assume that $k = z$. This is due to the fact that both parameters are counting thresholds in some sense: z is the threshold for counting the existence of loops and returns, while k can be seen as the threshold for distinguishing different prefixes of w which have the same atomic type. Thus, we identify k and z in the following definition of the equivalence relation induced by $\mathrm{Type}_n^{k;z}$.

Definition 3.3.22. For words $w, w' \in \Sigma^*$, we write $w \equiv_n^z w'$ if $\mathrm{Type}_n^{z;z}(w) = \mathrm{Type}_n^{z;z}(w')$.

As a first step, we want to show that $\equiv_n^z$ is a right congruence. We prepare the proof of this fact in the following lemma.

Lemma 3.3.23. Let $n \in \mathbb{N}$, $z \geq 2$ and $\mathcal{N}$ be some pushdown system of level 2. Let w be some word and $\sigma \in \Sigma$ some letter. For each $0 \leq i < |w|$, the atomic type of i and of 0 in $\mathfrak{Lin}_n^{z;z}(w)$ determines the atomic type of $i+1$ in $\mathfrak{Lin}_n^{z;z}(w\sigma)$.

Proof. Recall that $i \in \mathfrak{Lin}_n^{z;z}(w)$ represents w_{-i} and $i+1 \in \mathfrak{Lin}_n^{z;z}(w\sigma)$ represents $w\sigma_{-(i+1)}$. Since $w_{-i} = w\sigma_{-(i+1)}$, it follows directly that the two elements agree on $(P_\sigma)_{\sigma\in\Sigma}$, $(R_j)_{j\in J}$, $(L_j)_{j\in J}$, and $(H_j)_{j\in J}$ and that $w_{-i} \equiv_{n-1}^z w\sigma_{-(i+1)}$ (recall that the elements in $\mathfrak{Lin}_n^{z;z}(w)$ are coloured by $\equiv_{n-1}^z$-types).

We claim that the function $\#\mathrm{Ret}^z(w)$ and the set

$$\{(j,q,q') \in \mathbb{N} \times Q \times Q : j \leq z, \mathfrak{Lin}_n^{z;z}(w) \models i \in S^j_{q,q'}\}$$

determine whether $\mathfrak{Lin}_n^{z;z}(w\sigma) \models (i+1) \in S^j_{q,q'}$. Recall that the predicates $S^j_{q,q'}$ in $\mathfrak{Lin}_n^{z;z}(w)$ encode at each position l the number of runs ρ from (q,w) to (q',w_{-l}) that do not pass w_{-l} before $\ln(\rho)$. We now want to determine the number of runs ρ from $(q,w\sigma)$ to $(q',w\sigma_{-(i+1)}) = (q',w_{-i})$ that do not pass w_{-i} before $\ln(\rho)$.

It is clear that such a run starts with a high loop from $(q,w\sigma)$ to some $(\hat{q},w\sigma)$. Then it performs some transition of the form $(\hat{q},\sigma,\gamma,\hat{q}',\mathrm{pop}_1)$ and then it continues with a run from $(\hat{q}',w)$ to (q',w_{-i}) that do not pass w_{-i} before its last configuration.

In order to determine whether $\mathfrak{Lin}_n^{z;z}(w\sigma) \models (i+1) \in S^j_{q,q'}$, we have to count whether j runs of this form exist. To this end, we define the numbers

$$\begin{aligned}
k_{(\hat{q},\hat{q}')} &:= \#\mathrm{HLoop}^z(w\sigma)(q,\hat{q}),\\
j_{(\hat{q},\hat{q}')} &:= |\{(\hat{q},\sigma,\gamma,\hat{q}',\mathrm{pop}_1) \in \Delta\}|, \text{ and}\\
i_{(\hat{q},\hat{q}')} &:= \max\{k : \mathfrak{Lin}_n^{z;z}(w) \models w_{-i} \in S^k_{(\hat{q}',q')}\}
\end{aligned}$$

for each pair $\bar{q} = (\hat{q},\hat{q}') \in Q^2$. It follows directly that there are $\sum_{\bar{q}\in Q^2} i_{\bar{q}} j_{\bar{q}} k_{\bar{q}}$ many such runs up to threshold z. Note that $j_{\bar{q}}$ only depends on the pushdown system. Due to Corollary 2.4.62, $\#\mathrm{HLoop}^z(w\sigma)$ is determined by σ and $\#\mathrm{Ret}^z(w)$. Thus, $k_{\bar{q}}$ is determined by the atomic type of 0 in $\mathfrak{Lin}_n^{z;z}(w)$. $i_{\bar{q}}$ only depends on the atomic type of i in $\mathfrak{Lin}_n^{z;z}(w)$. These observations complete the proof. □

Corollary 3.3.24. Let $n, z \in \mathbb{N}$ such that $z \geq 2$. Let w_1 and w_2 be words such that $w_1 \equiv_n^z w_2$. Any strategy of Duplicator in the z round Ehrenfeucht-Fraïssé game on $\mathfrak{Lin}_n^{z;z}(w_1)$ and $\mathfrak{Lin}_n^{z;z}(w_2)$ translates directly into a strategy of Duplicator in the z round Ehrenfeucht-Fraïssé game on $\mathfrak{Lin}_n^{z;z}(w_1\sigma)\restriction_{[1,|w_1\sigma|]}$ and $\mathfrak{Lin}_n^{z;z}(w_2\sigma)\restriction_{[1,|w_2\sigma|]}$.

Proof. It suffices to note that the existence of Duplicators strategy implies that the atomic types of 0 in $\mathfrak{Lin}_n^{z;z}(w_1)$ and $\mathfrak{Lin}_n^{z;z}(w_2)$ agree. Hence, the previous lemma applies. Thus, if the atomic type of $i \in \mathfrak{Lin}_n^{z;z}(w_1)$ and $j \in \mathfrak{Lin}_n^{z;z}(w_2)$ agree, then the atomic types of $i+1 \in \mathfrak{Lin}_n^{z;z}(w_1\sigma)$ and $j+1 \in \mathfrak{Lin}_n^{z;z}(w_2\sigma)$ agree. Hence, we can obviously translate Duplicator's strategy on $\mathfrak{Lin}_n^{z;z}(w_1)$ and $\mathfrak{Lin}_n^{z;z}(w_2)$ into a strategy on $\mathfrak{Lin}_n^{z;z}(w_1\sigma)\restriction_{[1,|w_1\sigma|]}$ and $\mathfrak{Lin}_n^{z;z}(w_2\sigma)\restriction_{[1,|w_2\sigma|]}$. □

The previous corollary is the main ingredient for the following lemma. It states that $\equiv_n^z$ is a right congruence.

Lemma 3.3.25. For $z \geq 2$, $\equiv_n^z$ is a right congruence, i.e., if $\mathsf{Type}_n^{z;z}(w_1) = \mathsf{Type}_n^{z;z}(w_2)$ for some $z \geq 2$, then $\mathsf{Type}_n^{z;z}(w_1 w) = \mathsf{Type}_n^{z;z}(w_2 w)$ for all $w \in \Sigma^*$.

Proof. It is sufficient to prove the claim for $w = \sigma \in \Sigma$. The lemma then follows by induction on $|w|$. First observe that

$$\begin{aligned}
&\#\mathsf{Loop}^z(w_1\sigma) = \#\mathsf{Loop}^z(w_2\sigma),\\
&\#\mathsf{HLoop}^z(w_1\sigma) = \#\mathsf{HLoop}^z(w_2\sigma), \text{ and}\\
&\#\mathsf{Ret}^z(w_1\sigma) = \#\mathsf{Ret}^z(w_2\sigma),
\end{aligned}$$

because these values are determined by the values of the corresponding functions at w_1 and w_2 (cf. Propositions 2.4.19 and 2.4.47). These functions agree on w_1 and w_2 because the first elements of $\mathfrak{Lin}_n^{z;z}(w_1)$ and $\mathfrak{Lin}_n^{z;z}(w_2)$ are $\mathsf{FO}_2 \subseteq \mathsf{FO}_z$ definable.

For $i \in \{1,2\}$, $\mathfrak{Lin}_n^{z;z}(w_i\sigma) \models 0 \in S^j_{(q,q')}$ if and only if $j = 1$ and $q = q'$ because $S^j_{(q,q')}$ counts at position 0 the runs ρ from $(q, w_i\sigma)$ to $(q', w_i\sigma)$ that do not pass $w_i\sigma$ before $\ln(\rho)$ and, apparently, this implies $\ln(\rho) = 0$. Since $\#\mathsf{HLoop}^z(w_1\sigma) = \#\mathsf{HLoop}^z(w_2\sigma)$, we conclude that the atomic types of the first elements of $\mathfrak{Lin}_0^{z;z}(w_1\sigma)$ and of $\mathfrak{Lin}_0^{z;z}(w_2\sigma)$ coincide.

Due to the previous corollary, we know that Duplicator has a strategy in the z round Ehrenfeucht-Fraïssé game on $\mathfrak{Lin}_n^{z;z}(w_1\sigma)\restriction_{[1,|w_1|]}$ and $\mathfrak{Lin}_n^{z;z}(w_2\sigma)\restriction_{[1,|w_2|]}$.

Standard composition arguments for Ehrenfeucht-Fraïssé games on word structures directly imply that $\mathfrak{Lin}_0^{z;z}(w_1\sigma) \simeq_z \mathfrak{Lin}_0^{z;z}(w_2\sigma)$. But this directly implies that the atomic types of the first elements of $\mathfrak{Lin}_1^{z;z}(w_1\sigma)$ and of $\mathfrak{Lin}_1^{z;z}(w_2\sigma)$ coincide. If $n \geq 1$, we can apply the same standard argument and obtain that $\mathfrak{Lin}_1^{z;z}(w_1\sigma) \simeq_z \mathfrak{Lin}_1^{z;z}(w_2\sigma)$. By induction one concludes that $\mathfrak{Lin}_n^{z;z}(w_1\sigma) \simeq_z \mathfrak{Lin}_n^{z;z}(w_2\sigma)$. But this is the definition of $w_1\sigma \equiv_n^z w_2\sigma$. □

The next lemma can be seen as the inverse direction of the previous lemma. Instead of appending a word, we want to remove the topmost symbols from the word. For this operation, we cannot preserve the equivalence at the same level but at one level below.

Lemma 3.3.26. Let $m < 2^{z-1} - 1$ and $w, w' \in \Sigma^*$. If $w \equiv_n^z w'$ then $w_{-m} \equiv_{n-1}^z w'_{-m}$.

Proof. Quantifier rank z suffices to define the m-th element of a word structure. Hence, $w \equiv_n^z w'$ implies that $\mathsf{Type}_{n-1}^{z;z}(w_{-m}) = \mathsf{Type}_{n-1}^{z;z}(w'_{-m})$. But this is equivalent to $w_{-m} \equiv_{n-1}^z w'_{-m}$. □

The previous lemmas can be seen as statements concerning the compatibility of the stack operations push_σ and pop_1 with the equivalences $\equiv_n^z$. Later, we need a compatibility result

of the equivalences with all level 2 stack operations. For this purpose, we first lift these equivalences to equivalences on level 2 stacks. We compare the stacks word-wise beginning with the topmost word, then the word below the topmost one, etc. up to some threshold m. The following definition introduces the precise notion of these equivalence relations on stacks.

Definition 3.3.27. Let s, s' be stacks. We write $s \ {}_m\!\equiv^z_n s'$ if for all $0 \leq i \leq m$

$$\mathrm{top}_2\left(\mathrm{pop}_2^i(s)\right) \equiv^z_n \mathrm{top}_2\left(\mathrm{pop}_2^i(s')\right).$$

Remark 3.3.28. If $\mathsf{wdt}(s) \leq m$ or $\mathsf{wdt}(s') \leq m$ then $\mathrm{pop}_2^m(s)$ or $\mathrm{pop}_2^m(s')$ is undefined. In this case we write $s \ {}_m\!\equiv^z_n s'$ iff $\mathsf{wdt}(s) = \mathsf{wdt}(s')$ and $s \ {}_{m'}\!\equiv^z_n s'$ for $m' := \mathsf{wdt}(s) - 1$.

Next, we prove that these equivalence relations on stacks are compatible with all stack operations.

Proposition 3.3.29. Let $z \geq 2$ and let s_1, s_2, s_1', s_2' be stacks such that $s_1' = \mathsf{op}(s_1)$ and $s_2' = \mathsf{op}(s_2)$ for some stack operation op. If $s_1 \ {}_m\!\equiv^z_n s_2$ then the following hold:

- for $\mathsf{op} = \mathsf{push}_\sigma$, $s_1' \ {}_m\!\equiv^z_n s_2'$,
- for $\mathsf{op} = \mathsf{pop}_1$, $s_1' \ {}_m\!\equiv^z_{n-1} s_2'$,
- for $\mathsf{op} = \mathsf{clone}_2$, $s_1' \ {}_{m+1}\!\equiv^z_n s_2'$, and
- for $\mathsf{op} = \mathsf{pop}_2$, $s_1' \ {}_{m-1}\!\equiv^z_n s_2'$.

Proof. For $\mathsf{op} = \mathsf{pop}_1$, we use Lemma 3.3.26. For $\mathsf{op} = \mathsf{push}_\sigma$ we use Lemma 3.3.25. For clone_2 and pop_2, the claim follows directly from the definitions. □

The previous proposition shows that the equivalence relations on stacks are compatible with the stack operations. Recall that successive relevant ancestors of a given run ρ are runs $\rho_1 \prec \rho_2 \preceq \rho$ such that ρ_2 extends ρ_1 by either a single transition or by some run that creates some new word on top of the last stack of ρ_1 (cf. Proposition 3.3.17). In the next section, we are concerned with the construction of a short run $\hat{\rho}$ such that its relevant ancestors are isomorphic to those of ρ. A necessary condition for a run $\hat{\rho}$ to be short is that it only passes small stacks. We construct $\hat{\rho}$ using the following construction. Let $\rho_0 \prec \rho_1 \prec \rho_2 \ldots \prec \rho$ be the set of relevant ancestors of ρ. We then first define a run $\hat{\rho}_0$ that ends in some small stack that is equivalent to the last stack of ρ_0. Then, we iterate the following construction. If ρ_{i+1} extends ρ_i by a single transition, then we define $\hat{\rho}_{i+1}$ to be the extension of $\hat{\rho}_i$ by the same transition. Due to the previous proposition this preserves equivalence of the topmost stacks of ρ_i and $\hat{\rho}_i$. Otherwise, ρ_{i+1} extends ρ_i by some run that creates a new word w_{i+1} on top of the last stack of ρ_i. Then we want to construct a short run that creates a new word w'_{i+1} on top of the last stack of $\hat{\rho}_i$ such that w_{i+1} and w'_{i+1} are equivalent and w'_{i+1} is small. Then we define $\hat{\rho}_{i+1}$ to be $\hat{\rho}_i$ extended by this run.

Finally, this procedure defines a run $\hat{\rho}$ that corresponds to ρ in the sense that the relevant ancestors of the two runs are isomorphic but $\hat{\rho}$ is a short run.

In the following, we prepare this construction. We show that for any run ρ_0 there is a run $\hat{\rho}_0$ that ends in some small stack that is equivalent to the last stack of ρ_0. This is done in Corollary 3.3.37. Furthermore, we show that for runs ρ_i and $\hat{\rho}_i$ that end in equivalent

stacks, any run that extends the last stack of ρ_i by some word w can be transferred into a run that extends $\hat{\rho}_i$ by some small word that is equivalent to w. This is shown in Proposition 3.3.39.

The proofs of Corollary 3.3.37 and Proposition 3.3.39 are based on the property that prefixes of equivalent stacks share the same number of loops and returns for each pair of initial and final states. Recall that our analysis of generalised milestones showed that the existence of loops with certain initial and final states has a crucial influence on the question whether runs between certain stacks exist.

In the following, we first state three main lemmas concerning the reachability of small stacks that are equivalent to some given stack. Together, these lemmas directly imply the Corollary 3.3.37. Afterwards, we present the Proposition 3.3.39. In the end of this section, we provide the technical details for the proofs of the main lemmas and the proposition.

The first lemma allows to translate an arbitrary run ρ into another run ρ' that ends in a stack with a small topmost word such that the topmost words of ρ and ρ' are equivalent. We first define a function that is used to define what small means in this context.

Definition 3.3.30. Let $\mathcal{N} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a pushdown system of level 2. Set

$$\begin{aligned} \mathrm{BTW} &: \mathbb{N}^2 \to \mathbb{N} \\ \mathrm{BTW}(n, z) &= |Q| \cdot |\Sigma^* / \equiv_n^z| + 1, \end{aligned}$$

where $|\Sigma^* / \equiv_n^z|$ is the number of equivalence classes of $\equiv_n^z$.

Lemma 3.3.31. For all $z, n \in \mathbb{N}$ with $z \geq 2$ and for each run ρ with $|\mathrm{top}_2(\rho)| > \mathrm{BTW}(n, z)$ there is some run $\hat{\rho}$ with

$$\begin{aligned} &|\mathrm{top}_2(\rho)| - \mathrm{BTW}(n, z) \leq |\mathrm{top}_2(\hat{\rho})| < |\mathrm{top}_2(\rho)| \text{ and} \\ &\mathrm{top}_2(\rho) \equiv_n^z \mathrm{top}_2(\hat{\rho}). \end{aligned}$$

The previous lemma gives the possibility to replace a given run by some run that ends in an equivalent but small topmost word. After bounding the topmost word, we want to bound the height of all the words occurring in the last configuration of some run ρ. This is done with the next lemma.

Definition 3.3.32. Let $\mathcal{N} = (Q, \Sigma, \Gamma, \Delta, q_0)$ be a pushdown system of level 2. Set

$$\mathrm{B}_{\mathrm{hgt}} := |\Sigma^* / \equiv_0^2| \cdot |Q|^2.$$

Lemma 3.3.33. If ρ is some run with $\mathrm{hgt}(\rho) > |\mathrm{top}_2(\rho)| + \mathrm{B}_{\mathrm{hgt}}$, then there is a run $\hat{\rho}$ with

$$\begin{aligned} &\mathrm{hgt}(\rho) - \mathrm{B}_{\mathrm{hgt}} \leq \mathrm{hgt}(\hat{\rho}) < \mathrm{hgt}(\rho) \text{ and} \\ &\mathrm{top}_2(\rho) = \mathrm{top}_2(\hat{\rho}). \end{aligned}$$

Finally, we want to bound the width of the last stack of some run in terms of its height while preserving the topmost word. This is done in the following lemma.

Definition 3.3.34. Set

$$\begin{aligned} \mathrm{BWW} &: \mathbb{N} \to \mathbb{N} \\ n &\mapsto |Q| \cdot (|\Sigma| + 1)^n. \end{aligned}$$

Remark 3.3.35. $\mathrm{BWW}(n)$ is an upper bound for the number of pairs of states and words of length up to n.

Lemma 3.3.36. For every run ρ with $\mathrm{wdt}(\rho) > \mathrm{BWW}(\mathrm{hgt}(\rho))$ there is a run $\hat{\rho}$ with

$$\mathrm{wdt}(\rho) - \mathrm{BWW}\left(\mathrm{hgt}(\rho)\right) \leq \mathrm{wdt}(\hat{\rho}) < \mathrm{wdt}(\rho) \text{ and}$$
$$\mathrm{top}_2(\hat{\rho}) = \mathrm{top}_2(\rho).$$

The previous three lemmas are summarised in the following corollary. It asserts that for every run there is a run ending in a small stack with equivalent topmost word.

Corollary 3.3.37. For each run ρ starting in the initial configuration, there is a run ρ' starting in the initial configuration such that

$$|\mathrm{top}_2(\rho')| \leq \mathrm{BTW}(n, z),$$
$$\mathrm{hgt}(\rho') \leq |\mathrm{top}_2(\rho')| + \mathrm{B}_{\mathrm{hgt}},$$
$$\mathrm{wdt}(\rho') \leq \mathrm{BWW}(\mathrm{hgt}(\rho')) \text{ and}$$
$$\mathrm{top}_2(\rho) \equiv_n^z \mathrm{top}_2(\rho').$$

The previous corollary deals with the reachability of some stack from the initial configuration. The following proposition is concerned with the extension of a given stack by just one word. We first define the function that is used to bound the size of the new word.

Definition 3.3.38. Let $\mathcal{N}$ be a level 2 pushdown system with state set Q. Set

$$\mathrm{BH}_1 : \mathbb{N}^4 \to \mathbb{N}$$
$$(a, b, c, d) \mapsto b + a(|Q||^{\Sigma^*/\equiv_c^d}|).$$

Before we state the proposition, we explain its meaning. The proposition says that given two equivalent words w and $\hat{w}$ and a run ρ from $(q, s : w)$ to $(q', s : w : w')$ that does not pass a substack of $s : w$, then, for each stack $\hat{s} : \hat{w}$, we find a run $\hat{\rho}$ from $(q, \hat{s} : \hat{w})$ to $(q', \hat{s} : \hat{w} : \hat{w}')$ for some short word $\hat{w}'$ that is equivalent to w'. Furthermore, this transfer of runs works simultaneously on a tuple of such runs, i.e., given m runs starting at $s : w$ of the form described above, we find m corresponding runs starting at $\hat{s} : \hat{w}$. This simultaneous transfer becomes important when we search an isomorphic copy of the relevant ancestors of several runs. In this case the simultaneous transfer allows to copy the relevant ancestors of a certain run while avoiding an intersection with relevant ancestors of other given runs.

Proposition 3.3.39. Let $\mathcal{N}$ be a level 2 pushdown system and $n, z, m \in \mathbb{N}$ such that $n \geq 1$, $z > m$, and $z \geq 2$. Let $c = (q, s : w), \hat{c} = (q, \hat{s} : \hat{w})$ be configurations such that $w \equiv_n^z \hat{w}$. Let $\rho_1, \ldots, \rho_m$ be pairwise distinct runs such that for each i, $|\rho_i(j)| > |s : w|$ for all $j \geq 1$ and such that ρ_i starts at c and ends in $(q_i, s : w : w_i)$. Analogously, let $\hat{\rho}_1, \ldots, \hat{\rho}_{m-1}$ be pairwise distinct runs such that each $\hat{\rho}_i$ starts at $\hat{c}$ and ends in $(q_i, \hat{s} : \hat{w} : \hat{w}_i)$ and $|\hat{\rho}_i(j)| > |\hat{s} : \hat{w}|$ for all $j \geq 1$. If

$$w_i \equiv_{n-1}^z \hat{w}_i \text{ for all } 1 \leq i \leq m-1,$$

then there is some run $\hat{\rho}_m$ from $\hat{c}$ to $(q_0, \hat{s} : \hat{w} : \hat{w}_m)$ such that

$$w_m \equiv_{n-1}^z \hat{w}_m,$$
$$\hat{\rho}_m \text{ is distinct from each } \hat{\rho}_i \text{ for } 1 \leq i < m, \text{ and}$$
$$|\hat{w}_m| \leq \mathrm{BH}_1(m, |\hat{w}|, n, z).$$

The rest of this section is concerned with the proofs of Lemmas 3.3.36, 3.3.33, and 3.3.31 and with the proof of Proposition 3.3.39. The reader who is not interested in the technical details of these proofs may skip the rest of this section and continue reading Section 3.3.6. In that section show how the results of this section can be used to construct isomorphic relevant ancestors that consist of runs ending in small stacks.

Prefix Replacement Revisited

Recall that we defined the prefix replacement for runs that are prefixed by a certain stack (cf. Lemma 2.3.38). We want to extend the notion of prefix replacement to runs that are only prefixed at the beginning and at the end by some stack s and that never visit the substack $\mathsf{pop}_2(s)$. We apply this new form of prefix replacement in the proofs of Lemmas 3.3.36, 3.3.33 and 3.3.31. The following lemma prepares this new kind of prefix replacement.

Lemma 3.3.40. Let $\mathcal{N}$ be some level 2 pushdown system and let ρ be a run of $\mathcal{N}$ of length n. Let s be a stack with topmost word $w := \mathsf{top}_2(s)$ such that

1. $s \trianglelefteq \rho(0)$,
2. $s \trianglelefteq \rho(n)$, and
3. $|s| \leq |\rho(i)|$ for all $0 \leq i \leq n$.

There is a unique sequence $0 = i_0 \leq j_0 < i_1 \leq j_1 < \cdots < i_{m-1} \leq j_{m-1} < i_m \leq j_m = n$ such that

1. $s \trianglelefteq \rho\restriction_{[i_k,j_k]}$ for all $0 \leq k \leq m$ and
2. $\mathsf{top}_2(\rho(j_k+1)) = \mathsf{pop}_1(w)$, $\rho\restriction_{[j_k,i_{k+1}]}$ is either a loop or a return, and $\rho\restriction_{[j_k,i_{k+1}]}$ does not visit the stack of $\rho(j_k)$ between its initial configuration and its final configuration for all $0 \leq k < m$.

Proof. If $s \trianglelefteq \rho$, then we set $m := 0$ and we are done. Otherwise, we proceed by induction on the length of ρ.

There is a minimal position $j_0 + 1$ such that $s \not\trianglelefteq \rho(j_0 + 1)$. By assumption on s, $\rho(j_0+1) \neq \mathsf{pop}_2(s)$. Thus, $\mathsf{top}_2(\rho(j_0)) = w$ and $\mathsf{top}_2(\rho(j_0+1)) = \mathsf{pop}_1(w)$. Now, let $i_1 > j_0$ be minimal such that $s \trianglelefteq \rho(i_1)$. Concerning the stack at i_1 there are the following possibilities.

1. If $\rho(i_1) = \mathsf{pop}_2(\rho(j_0))$ then $s \trianglelefteq \rho(i_1)$ (cf. Lemma 2.3.37). Furthermore, $\rho\restriction_{[j_0,i_1]}$ is a return.
2. Otherwise, the stacks of $\rho(j_0)$ and $\rho(i_1)$ coincide whence $\rho\restriction_{[j_0,i_1]}$ is a loop (note that between j_0 and i_1 the stack $\mathsf{pop}_2(\rho(j_0))$ is never visited due to the minimality of i_1 and due to assumption 3).

$\rho\restriction_{[i_1,n]}$ is shorter than ρ. Thus, it decomposes by induction hypothesis and the lemma follows immediately. □

This lemma gives rise to the following extension of the prefix replacement.

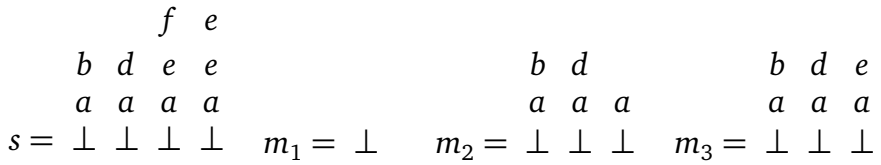


Figure 3.18.: Illustration for the construction in the proof of Lemma 3.3.31.

Definition 3.3.41. Let s be some stack and ρ be a run of some pushdown system $\mathcal{N}$ of level 2 such that $s \trianglelefteq \rho(0)$, $s \trianglelefteq \rho(\ln(\rho))$ and $|s| \leq |\rho(i)|$ for all $i \in \mathrm{dom}(\rho)$. Let u be some stack such that $\mathrm{top}_1(u) = \mathrm{top}_1(s)$, $\#\mathrm{Loop}^1(u) = \#\mathrm{Loop}^1(s)$ and $\#\mathrm{Ret}^1(u) = \#\mathrm{Ret}^1(s)$.

Let $0 = i_0 \leq j_0 < i_1 \leq j_1 < \cdots < i_{m-1} \leq j_{m-1} < i_m \leq j_m = \ln(\rho)$ be the sequence corresponding to ρ in the sense of the previous lemma. We set $(q_k, s_k) := \rho(j_k)$ and $(q'_k, s'_k) := \rho(i_{k+1})$. By definition, $\rho\restriction_{[j_k, i_{k+1}]}$ is a loop or a return from (q_k, s_k) to (q'_k, s'_k) and $\mathrm{top}_2(s_k) = \mathrm{top}_2(s)$ and $s \trianglelefteq s_k$. Thus, $\mathrm{top}_2(s_k[s/u]) = \mathrm{top}_2(u)$. Since $\#\mathrm{Ret}^1(u) = \#\mathrm{Ret}^1(s)$ and $\#\mathrm{Loop}^1(u) = \#\mathrm{Loop}^1(s)$, there is a run from $(q_k, s_k[s/u])$ to $(q'_{k+1}, s'_{k+1}[s/u])$. We set ρ_k to be the lexicographically shortest run from $(q_k, s_k[s/u])$ to $(q'_{k+1}, s'_{k+1}[s/u])$.

Then we define the run

$$\rho[s/u] := \rho\restriction_{[i_0, j_0]}[s/u] \circ \rho_0 \circ \rho\restriction_{[i_1, j_1]}[s/u] \circ \rho_1 \circ \cdots \circ \rho_{m-1} \circ \rho\restriction_{[i_m, j_m]}[s/u].$$

Note that $\rho[s/u]$ is a well-defined run from $\rho(0)[s/u]$ to $\rho(\ln(\rho))[s/u]$.

Proof of Lemma 3.3.31

Recall that Lemma 3.3.31 asserts for every run ρ the existence of a run ρ' that ends in a stack with small topmost word that is equivalent to the topmost word of the last stack of ρ. The proof of this lemma is as follows.

Proof of Lemma 3.3.31. Let ρ be some run with $|\mathrm{top}_2(\rho)| > \mathrm{BTW}(n, z)$. $(q, s) := \rho(\ln(\rho))$ denotes the final configuration of ρ. For each $k \leq \mathrm{BTW}(n, z)$, there is a maximal milestone $m_k \in \mathrm{MS}(s)$ with $|\mathrm{top}_2(m_k)| = k$. Figure 3.18 illustrates this definition. Let $w_k := \mathrm{top}_2(m_k)$ and let $\rho_k \preceq \rho$ be the largest initial segment of ρ that ends in m_k. Note that $m_k \trianglelefteq m_{k'} \trianglelefteq s$ for all $k \leq k' \leq \mathrm{BTW}(n, z)$ by the maximality of m_k and $m_{k'}$.

Then there are $i < j \leq \mathrm{BTW}(n, z)$ such that $\mathrm{top}_2(\rho_i) \equiv_n^z \mathrm{top}_2(\rho_j)$ and the final states of ρ_i and ρ_j agree.

Due to the maximality of ρ_j, no substack of $\mathrm{pop}_2(m_j)$ is visited by ρ after $k := \ln(\rho_j)$. Thus, the run $\pi := (\rho\restriction_{[k, \ln(\rho)]})[m_j/m_i]$ is well-defined (cf. Definition 3.3.41). Note that π starts by definition in (q', m_i) for $q' \in Q$ the final state of ρ_i. Thus, we can set $\hat{\rho} := \rho_i \circ \pi$. Due to $w_i \equiv_n^z w_k$ and the right congruence of $\equiv_n^z$ (cf. Lemma 3.3.25.), it is clear that $\mathrm{top}_2(\hat{\rho}) \equiv_n^z \mathrm{top}_2(\rho)$. Since $0 < |w_j| - |w_i| < \mathrm{BTW}(n, z)$, it also follows directly that

$$|\mathrm{top}_2(\rho)| - \mathrm{BTW}(n, z) \leq |\mathrm{top}_2(\hat{\rho})| < |\mathrm{top}_2(\rho)|.$$

□

Proof of Lemma 3.3.33

Recall that Lemma 3.3.33 asserts that for each run ρ there is a run ρ' such that $\mathrm{top}_2(\rho) = \mathrm{top}_2(\rho')$ and such that the height of the last stack of ρ' is bounded in terms of $|\mathrm{top}_2(\rho)|$.

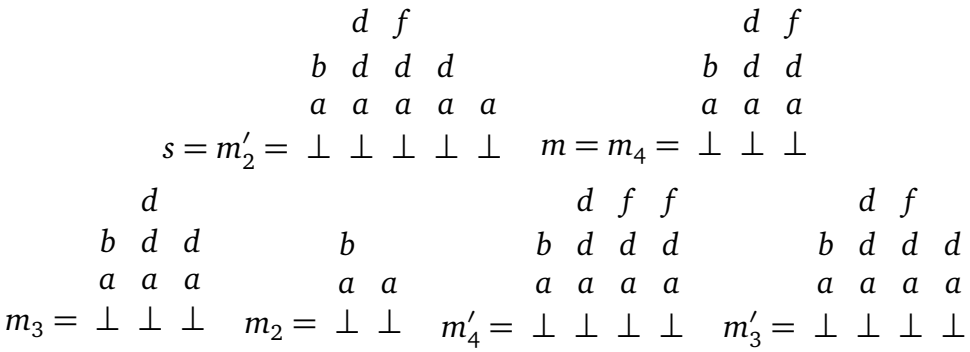


Figure 3.19.: Illustration for the construction in the proof of Lemma 3.3.33.

Proof of Lemma 3.3.33. The proof is by induction on the number of words in the last stack of ρ that have length $h := \mathrm{hgt}(\rho)$. Assume that ρ is some run such that

$$\mathrm{hgt}(\rho) > |\mathrm{top}_2(\rho)| + \mathrm{B}_{\mathrm{hgt}}.$$

In the following, we define several generalised milestones of the final stack s of ρ. An illustration of these definitions can be found in Figure 3.19.

Let $m \in \mathrm{MS}(s)$ be a milestone of the last stack of ρ such that $|\mathrm{top}_2(m)| = h$. For each $|\mathrm{top}_2(\rho)| \leq i \leq h$ let $m_i \in \mathrm{MS}(m)$ be the maximal milestone of m with $|\mathrm{top}_2(m_i)| = i$. Let n_i be maximal such that $\rho(n_i) = (q', m_i)$ for some $q' \in Q$. Let $m'_i \in \mathrm{GMS}(s) \setminus \mathrm{GMS}(m)$ be the minimal generalised milestone after m such that $\mathrm{top}_2(m'_i) = \mathrm{top}_2(m_i)$. Let n'_i be maximal with $\rho(n'_i) = (q', m'_i)$ for some $q' \in Q$.

There are $|\mathrm{top}_2(\rho)| \leq k < l \leq \mathrm{hgt}(\rho)$ satisfying the following conditions.

1. There is a $q \in Q$ such that $\rho(n_k) = (q, m_k)$ and $\rho(n_l) = (q, m_l)$.
2. There is a $q' \in Q$ such that $\rho(n'_k) = (q', m'_k)$ and $\rho(n'_l) = (q', m'_l)$.
3. $\mathrm{top}_2(m_k) \equiv^2_0 \mathrm{top}_2(m_l)$ (this assumption implies that $\#\mathrm{Loop}^1(m_k) = \#\mathrm{Loop}^1(m_l)$ and $\#\mathrm{Ret}^1(m_k) = \#\mathrm{Ret}^1(m_l)$).

By definition, we have $m_l \trianglelefteq m'_l$. Thus, the run $\pi_1 := (\rho \restriction_{[n_l, n'_l]})[m_l/m_k]$ is well defined (cf. Definition 3.3.41). Note that π_1 starts in (q, m_k) and ends in $(q', \hat{s})$ for $\hat{s} := m'_l[m_l/m_k]$). Moreover, $\mathrm{top}_2(\pi_1) = \mathrm{top}_2(m'_k) = \mathrm{top}_2(\rho(n'_k))$. Furthermore, $\rho \restriction_{[n'_k, \mathrm{ln}(\rho)]}$ never looks below the topmost word of m'_k because n'_k is the maximal node where the generalised milestone m'_k is visited. Thus, $\mathrm{pop}_2(m'_k) : \bot \trianglelefteq \rho \restriction_{[n'_k, \mathrm{ln}(\rho)]}$. Due to Lemma 2.3.38,

$$\pi_2 := \rho \restriction_{[n'_k, \mathrm{ln}(\rho)]}[\mathrm{pop}_2(m'_k) : \bot / \mathrm{pop}_2(\hat{s}) : \bot]$$

is well defined. It starts in the last stack of π_1.

Now, we define the run

$$\hat{\rho} := \rho \restriction_{[0, n_k]} \circ \pi_1 \circ \pi_2.$$

Either $\mathrm{hgt}(\hat{\rho}) < \mathrm{hgt}(\rho)$ and we are done or there are less words of height $\mathrm{hgt}(\rho)$ in the last stack of $\hat{\rho}$ than in the last stack of ρ and we conclude by induction. □

Proof of Lemma 3.3.36

Recall that Lemma 3.3.36 asserts that for any run ρ there is another run ρ' such that ρ and ρ' end in stacks with equal topmost word but the width of the final stack of ρ' is bounded in terms of its height.

Proof of Lemma 3.3.36. Assume that ρ is a run with $\ln(\rho) > \mathsf{BWW}(\mathsf{hgt}(\rho))$. We denote by n_i the maximal position in ρ such that the stack at $\rho(n_i)$ is $\mathrm{pop}_2^i(\rho)$ for each $0 \leq i \leq \mathsf{wdt}(\rho)$. There are less than $\frac{\mathsf{BWW}(\mathsf{hgt}(\rho))}{|Q|}$ many words of length up to $\mathsf{hgt}(\rho)$. Thus, there are $i < j$ such that

1. $\mathrm{top}_2\left(\mathrm{pop}_2^i(\rho)\right) = \mathrm{top}_2\left(\mathrm{pop}_2^j(\rho)\right)$, and
2. $\rho(n_i) = \left(q, \mathrm{pop}_2^i(\rho)\right)$ and $\rho(n_j) = \left(q, \mathrm{pop}_2^j(\rho)\right)$ for some $q \in Q$.

Now, let $s_i := \mathrm{pop}_2^{i+1}(\rho)$ and $s_j := \mathrm{pop}_2^{j+1}(\rho)$. There is a unique stack s such that $\rho(\ln(\rho)) = (\hat{q}, s_i : s)$. $\rho\restriction_{[n_i,\ln(\rho)]}$ is a run from $\mathrm{pop}_2^i(\rho)$ to $s_i : s$ that never visits s_i. Thus,

$$\hat{\rho}_1 := \rho\restriction_{[n_i,\ln(\rho)]}[s_i : \bot / s_j : \bot]$$

is a well defined run. The composition $\hat{\rho} := \rho\restriction_{[0,n_j]} \circ \hat{\rho}_1$ satisfies the claim. □

Proof of Proposition 3.3.39

We decompose the proof of Proposition 3.3.39 in several lemmas. Recall that this Proposition is about a run ρ from some stack $s : w$ to some stack $s : w : w'$ that does not visit substacks of $s : w$. Such a run decomposes into three parts. First, it performs a clone_2 operation. Then there is a run from $w : w$ to $w : (w \sqcap w')$, i.e., a run that removes letters from w until it reaches the greatest common prefix of w and w'. Finally, the run constructs w' from the prefix $w \sqcap w'$. In the following we first treat the second and the third part separately. We prove lemmas that allow to transfer each of these parts from one starting stack to another one. Afterwards, we compose these arguments in order to obtain the proof of Proposition 3.3.39.

The first lemma is concerned with a transfer of several runs starting with the same stack $s : w$ and ending in the same stack $s : w'$ for some prefix w' of w.

Lemma 3.3.42. Let $z, m, n \in \mathbb{N}$ such that $z \geq 2$ and $z > m$. Let v, w, w' be words with $v \leq w$, and $q, \hat{q} \in Q$ states. Let there be pairwise distinct runs $\rho_1, \ldots, \rho_m$ from (q, w) to $(\hat{q}, v)$ such that each ρ_i does not visit v before $\ln(\rho_i)$. If $w \equiv_{n+1}^z w'$, then there exist a word $v' \leq w'$ and pairwise distinct runs $\rho'_1, \ldots, \rho'_m$ from (q, w') to $(\hat{q}, v')$ such that

$$\begin{aligned}
&v \equiv_n^z v',\ \mathrm{top}_1(v) = \mathrm{top}_1(v'),\\
&v = w \text{ iff } v' = w' \text{and}\\
&\rho'_i \text{ does not visit } v' \text{ before } \ln(\rho'_i).
\end{aligned}$$

Proof. Let $i \in \mathbb{N}$ be such that $v = w_{-i}$. Then i is labelled by $S^m_{q,\hat{q}}$, $P_{\mathrm{top}_1(v)}$, and $\mathsf{Type}_n^{z;z}(v)$ in $\mathfrak{Lin}_{n+1}^{z;z}(w)$. Since $\mathfrak{Lin}_{n+1}^{z;z}(w) \simeq_z \mathfrak{Lin}_{n+1}^{z;z}(w')$, there is some $i' \in \mathbb{N}$ such that i' in $\mathfrak{Lin}_{n+1}^{z;z}(w')$ is labelled by the same relations as i in $\mathfrak{Lin}_{n+1}^{z;z}(w)$. Due to $z \geq 2$, we can choose this i' such

that $i' \neq 0$ if and only if $i \neq 0$. Note that for $v' := w'_{-i'}$, $\mathsf{top}_1(v') = \mathsf{top}_1(v)$ because i' and i agree on the label $P_{\mathsf{top}_1(v)}$. Since $i' \in S^m_{q,\hat{q}}$, there are m pairwise distinct runs from (q, w') to $(\hat{q}, v')$ that visit v' only in the final configuration. Finally, $v \equiv^z_n v'$ due to the fact that i' and i agree on the labels characterising $\mathsf{Type}^{z;z}_n(v')$ and $\mathsf{Type}^{z;z}_n(v)$, respectively. □

The next lemma is in some sense the "otherwise" to the previous one. This lemma allows to transfer runs starting in the same word w but ending in different prefixes of w.

Lemma 3.3.43. Let $z, m, n \in \mathbb{N}$ such that $z \geq 2$ and $z > m$. Let w, w' be words, $v_1, \ldots, v_m \leq w$ pairwise distinct prefixes. Let there be pairwise distinct runs $\rho_1, \ldots, \rho_m$ such that ρ_i is from (q, w) to (q_i, v_i) and ρ_i does not visit v_i before $\mathsf{ln}(\rho_i)$. If $w \equiv^z_{n+1} w'$, then there are pairwise distinct prefixes $v'_1, \ldots, v'_m \leq w'$ and pairwise distinct runs $\rho'_1, \ldots, \rho'_m$ such that each ρ'_i starts in (q, w') and ends in (q_i, v'_i), $v'_i \leq w'$, and $v'_i \equiv^z_n v_i$ such that ρ'_i does not visit v'_i before $\mathsf{ln}(\rho'_i)$.

Proof. Any winning strategy in the z-round Ehrenfeucht-Fraïssé game on $\mathfrak{Lin}^{z;z}_{n+1}(w)$ and $\mathfrak{Lin}^{z;z}_{n+1}(w')$ chooses responses $v'_1, \ldots, v'_m$ for $v_1, \ldots, v_m$ such that the labels of the nodes associated to v'_i in $\mathfrak{Lin}^{z;z}_{n+1}(w')$ and the nodes associated to v_i in $\mathfrak{Lin}^{z;z}_{n+1}(w)$ agree and the v'_i are pairwise distinct. Hence, $v_i \equiv^z_n v'_i$. Now, there are runs ρ'_i as desired due to the fact that the node associated with v'_i in $\mathfrak{Lin}^{z;z}_{n+1}(w')$ is labelled by S^1_{q,q_i}. □

Remark 3.3.44. Since $z > m$, the strategy preserves also the labels of the left and right neighbour. Thus, if $v_i = w_{-k}$ and $v'_i = w'_{-k'}$, then $k = 0$ if and only if $k' = 0$, and if $k \neq 0$, then $\mathsf{top}_1(w_{-k+1}) = \mathsf{top}_1(w'_{-k'+1})$.

The combination of the previous two lemmas yields the following corollary.

Corollary 3.3.45. Let $z, m, n \in \mathbb{N}$ such that $z \geq 2$ and $z > m$ and let w, w' be words. Let there be pairwise distinct runs $\rho_1, \ldots, \rho_m$ such that ρ_i is a run from (q_i, w) to $(\hat{q}_i, v_i)$ for $v_i \leq w$ and ρ_i does not visit v_i before $\mathsf{ln}(\rho_i)$. If $w \equiv^z_{n+1} w'$, then there are prefixes $v'_1, \ldots, v'_m \leq w'$ and pairwise distinct runs $\rho'_1, \ldots, \rho'_m$ such that ρ_i starts in (q_i, w') and ends in $(\hat{q}_i, v'_i)$, $v'_i \equiv^z_n v_i$, and ρ'_i does not visit v'_i before $\mathsf{ln}(\rho'_i)$.

This corollary provides the transfer of runs from some stack $s : w$ to stacks $s : w_i$ with $w_i \leq w$ to another starting stack $s' : w'$ if w and w' are equivalent words.

Now, we start the investigation of the other direction. We analyse runs from some word w to some extension wv. If w' is equivalent to w, then we first transfer the run from w to wv to a run from w' to $w'v$. Afterwards, we even provide a lemma that allows to shrink v during this transfer process.

Lemma 3.3.46. Let $n, m, z \in \mathbb{N}$ with $z \geq 2$ and $z > m$. Let $\rho_0, \ldots, \rho_m$ be pairwise distinct runs such that ρ_i starts in (q, w) and ends in $(\hat{q}, ww_i)$. If w' is a word such that $w \equiv^z_n w'$, then there are pairwise distinct runs $\rho'_0, \ldots, \rho'_m$ such that ρ'_i starts in (q, w') and ends in $(\hat{q}, w'w_i)$.

Proof. This follows directly from the fact that

$$\#\mathsf{Ret}^z(w) = \#\mathsf{Ret}^z(w') \text{ and } \#\mathsf{Loop}^z(w) = \#\mathsf{Loop}^z(w').$$

Each run from w to ww_i is a sequence of loops and push operations. But the existence of the corresponding loops when starting in w' are guaranteed by the inductive computability of the number of loops and returns (cf. Proposition 2.4.47). □

As already indicated, we will now improve this lemma in the sense that we shrink the word w_i to a short word w_i'.

We fix some pushdown system $\mathcal{N}$ of level 2 with state set Q and stack alphabet Σ. Let

$$\begin{aligned} &\mathrm{f} : \mathbb{N}^2 \to \mathbb{N} \\ &(n, z) \mapsto 1 + |Q| \cdot \left| \Sigma^* / \equiv_n^z \right| \end{aligned}$$

for $\left| \Sigma^* / \equiv_n^z \right|$ the index of $\equiv_n^z$.

Lemma 3.3.47. For $q, \bar{q} \in Q$, w and v words, let ρ be a run from (q, w) to $(\bar{q}, wv)$. If $|v| > \mathrm{f}(n, z)$ then there is a run ρ' from (q, w) to $(\bar{q}, wv')$ such that

$$\begin{aligned} &|v| - \mathrm{f}(n, z) \leq |v'| < |v|, \\ &wv \equiv_n^z wv' \text{ and} \\ &\text{the first letters of } v \text{ and } v' \text{ coincide.} \end{aligned}$$

Proof. Assume that $|v| > \mathrm{f}(n, z)$. Then there are two distinct prefixes $\varepsilon < u_1 < u_2 \leq v$ such that

1. ρ passes wu_1 and wu_2 in the same state $\hat{q} \in Q$,
2. $wu_1 \equiv_n^z wu_2$, and
3. $1 \leq |u_1| < |u_2| \leq f(n, z) + 1$.

Set v_i to be the unique word such that $v = u_i v_i$ for $i \in \{1, 2\}$. Since $\equiv_n^z$ is a right congruence, $wu_1 v_2 \equiv_n^z wu_2 v_2 = wv$. $wu_1 \equiv_n^z wu_2$ implies that $\#\mathrm{Ret}^z(wu_1) = \#\mathrm{Ret}^z(wu_2)$ and $\#\mathrm{Loop}^z(wu_1) = \#\mathrm{Loop}^z(wu_2)$. Since there is a run from $(\hat{q}, wu_2)$ to $(\bar{q}, wu_2 v_2) = (\bar{q}, wv)$, we can use the prefix replacement $[wu_2/wu_1]$ we obtain a run $\hat{\rho}$ from $(\hat{q}, wu_1)$ to $(\bar{q}, wu_1 v_2)$. Composition of the initial part of ρ up to $(\hat{q}, wu_1)$ with $\hat{\rho}$ yields a run ρ'. By construction ρ' satisfies the claim. □

The following corollary uses the previous lemma in such a way that we can transfer some run to a new run that does not coincide with certain given runs

Corollary 3.3.48. Let ρ,v, and w be as in the previous lemma and let $\rho_1, \ldots, \rho_m$ be runs distinct from ρ, then we can find a run ρ' distinct from all ρ_i for $1 \leq i \leq m$ from (q, w) to $(\bar{q}, wv')$ such that $|v'| \leq m \cdot \mathrm{f}(n, z)$, $wv \equiv_n^z wv'$, and v and v' start with the same letter.

Proof. If v is long enough, we find $m+2$ words that are visited in the same state and which are of the same type. There is one pair among these words which can be used as u_1 and u_2 in the previous lemma such that the final configuration does not agree with that of any of the other runs $\rho_1, \ldots, \rho_m$. □

For the proof of Proposition 3.3.39, we now compose the previous lemmas. Recall that the proposition says the following: given m runs $\rho_1, \ldots, \rho_m$ that only add one word to a given stack and given m words that are equivalent to the words on top of the initial stacks of the ρ_i, we can transfer the runs $\rho_1, \ldots, \rho_m$ to runs $\rho_1', \ldots, \rho_m'$ that start at the given m words and extend these by one word each such that the resulting new words are equivalent to the words originally created by $\rho_1, \ldots, \rho_m$.

Proof of Proposition 3.3.39. Let $\rho_1, \rho_2, \ldots, \rho_m$ and $\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_{m-1}$ be runs as required in the proposition.

First, assume that $w_i \equiv^z_{n-1} w_j$ and that all runs ρ_i end in the same state, i.e., $q_i = q_j$, for all $1 \leq i \leq j \leq m$. At the end of the proof we discuss the case that this assumption is not true.

Each run ρ_i decomposes as $\rho_i = \rho_i^0 \circ \rho_i^1 \circ \rho_i^2$ where ρ_i^0 performs only one clone operation, and ρ_i^1 is the run from $s : w : w$ to the first occurrence of $s : w : (w \sqcap w_i)$.

Due to $\mathsf{top}_1(w) = \mathsf{top}_1(\hat{w})$, there are runs $\hat{\rho}_i^0$ from $\hat{c}$ to $\hat{s} : \hat{w} : \hat{w}$ performing only one clone operation and ending in the same state as ρ_i^0.

By Corollary 3.3.45, we can transfer the ρ_i^1 to runs $\hat{\rho}_i^1$ starting at $(q, \hat{s} : \hat{w} : \hat{w})$ and ending at $\hat{s} : \hat{w} : \hat{u}_i$ with $\hat{u}_i \leq \hat{w}$ and with $w \sqcap w_i \equiv^z_{n-1} \hat{u}_i$. The lemma guarantees that $\hat{\rho}_i^1 = \hat{\rho}_j^1$ iff ${\rho^1}_i = {\rho^1}_j$.

For v_i the word such that $w_i = (w \sqcap w_i) \circ v_i$, we use Lemma 3.3.46 to construct runs from $\hat{s} : \hat{w} : \hat{u}_i$ to $(q_i, \hat{s} : \hat{w} : \hat{u}_i v_i)$ such that $\hat{u}_i v_i \equiv^z_{n-1} w_i$. Applying Corollary 3.3.48, we find words $\hat{v}_1, \ldots, \hat{v}_m$, and runs $\hat{\rho}_1^2, \ldots, \hat{\rho}_m^2$ such that $\hat{\rho}_i^2$ is a run from $\hat{s} : \hat{w} : \hat{u}_i$ to $(q_i, \hat{s} : \hat{w} : \hat{u}_i \hat{v}_i)$ such that $\hat{u}_i \hat{v}_i \equiv^z_{n-1} w_i$ and such that $\hat{u}_i \hat{v}_i$ has length bounded by

$$\mathrm{BH}_1(m, |\hat{w}|, n, z) = |\hat{w}| + m \cdot \mathrm{f}(n, z).$$

Furthermore, Corollary 3.3.48 assures that $\hat{\rho}_i^2$ and $\hat{\rho}_j^2$ coincide if and only if ρ_i^2 and ρ_j^2 coincide.

We conclude that the runs

$$\hat{\rho}_1^0 \circ \hat{\rho}_1^1 \circ \hat{\rho}_1^2, \hat{\rho}_2^0 \circ \hat{\rho}_2^1 \circ \hat{\rho}_2^2, \ldots, \hat{\rho}_m^0 \circ \hat{\rho}_m^1 \circ \hat{\rho}_m^2$$

are pairwise distinct as follows. Heading for a contradiction assume that there are $i \neq j$ such that $\hat{\rho}_i^0 \circ \hat{\rho}_i^1 \circ \hat{\rho}_i^2 = \hat{\rho}_j^0 \circ \hat{\rho}_j^1 \circ \hat{\rho}_j^2$. Since $\mathsf{ln}(\hat{\rho}_i^0) = 1 = \mathsf{ln}(\hat{\rho}_j^0)$, it follows that $\hat{\rho}_i^0 = \hat{\rho}_j^0$ and $\hat{\rho}_i^1 \circ \hat{\rho}_i^2 = \hat{\rho}_j^1 \circ \hat{\rho}_j^2$. Since the runs coincide, we have $\hat{u}_i \hat{v}_i = \hat{u}_j \hat{v}_j$. Using Remark 3.3.44 and the fact that the first letters of v_i and $\hat{v}_i$ agree, one concludes that $\hat{u}_i = \hat{w} \sqcap \hat{u}_i \hat{v}_i = \hat{w} \sqcap \hat{u}_j \hat{v}_j = \hat{u}_j$. But by definition, $\mathsf{ln}(\hat{\rho}_i^1)$ and $\mathsf{ln}(\hat{\rho}_j^1)$, respectively, is the first occurrence of $\hat{u}_i = \hat{u}_j$ in $\hat{\rho}_i^1 \circ \hat{\rho}_i^2$ and $\hat{\rho}_j^1 \circ \hat{\rho}_j^2$, respectively. It follows that $\hat{\rho}_i^1 = \hat{\rho}_j^1$ and $\hat{\rho}_i^2 = \hat{\rho}_j^2$. By definition, this implies that $\rho_i^0 = \rho_j^0$, $\rho_i^1 = \rho_j^1$, and $\rho_i^2 = \rho_j^2$. Thus, we have $\rho_i = \rho_j$ contradicting the assumptions of the proposition.

Thus, the runs are pairwise distinct and one of them does not coincide with any of the $\hat{\rho}_i$ for $1 \leq i \leq m-1$. Without loss of generality, assume that

$$\hat{\rho}_m := \hat{\rho}_m^0 \circ \hat{\rho}_m^1 \circ \hat{\rho}_m^2 \neq \hat{\rho}_i \text{ for all } 1 \leq i < m.$$

Note that $\hat{\rho}_m$ satisfies the claim of the proposition by construction.

In the case that the runs $\rho_1, \ldots, \rho_m$ end in configurations with different states or different $\equiv^z_{n-1}$-types of their topmost words. In this case, we just concentrate on those ρ_i which end in the same state as ρ_0 and with a topmost word of the same type as w_0. This is sufficient because some run ρ can only coincide with $\hat{\rho}_i$ if both runs end up in stacks whose topmost words have the same type. □

3.3.6 Small-Witness Property via Isomorphisms of Relevant Ancestors

In this section, we want to define a family of equivalence relations on tuples of runs of a level 2 nested pushdown tree. The equivalence class of a tuple $\rho_1, \dots, \rho_m$ with respect to one of these relations is the isomorphism type of the substructure induced by the relevant l-ancestors of $\rho_1, \dots, \rho_m$ extended by some information for preserving this isomorphism during an Ehrenfeucht-Fraïssé game (while decreasing l in every round of the game). Recall that such a game ends in a winning position for Duplicator if the relevant 1-ancestors of the elements that were chosen in the two structures are isomorphic (cf. Lemma 3.3.9).

An important property of these equivalence relations is that they have finite index because the sets of l-ancestors are finite and the information we add to the structure can be encoded by a bounded number of unary predicates.

Finally, we show how to construct small representatives for each equivalence class. As explained in Section 2.1.1, this property can be turned into an FO model checking algorithm on the class of 2-NPT.

Definition 3.3.49. Let $\bar{\rho} = (\rho_1, \rho_2, \dots, \rho_m)$ be runs of a level 2 pushdown system $\mathcal{N}$ and let $\mathfrak{N} := \mathrm{NPT}(\mathcal{N})$ be the 2-NPT generated by $\mathcal{N}$. Let $l, n_1, n_2, z \in \mathbb{N}$. We define the following relations on $\mathrm{RA}_l(\bar{\rho})$.

1. For $k \leq l$ and $\rho \in \bar{\rho}$, let $P^k_\rho := \{\pi \in \mathrm{RA}_l(\bar{\rho}) : \pi \in \mathrm{RA}_k(\rho)\}$.
2. Let ${}_{n_1}\equiv^z_{n_2}$-Type denote the function that maps a run π to the equivalence class of the last stack of π with respect to ${}_{n_1}\equiv^z_{n_2}$.

We write $\mathfrak{rA}_{l,n_1,n_2,z}(\bar{\rho})$ for the following expansion of the relevant ancestors of $\bar{\rho}$:

$$\mathfrak{rA}_{l,n_1,n_2,z}(\bar{\rho}) := (\mathfrak{N}\restriction_{\mathrm{RA}_l(\bar{\rho})}, \vdash, \hookrightarrow, \overset{+1}{\hookrightarrow}, {}_{n_1}\equiv^z_{n_2}\text{-Type}, (P^k_{\rho_j})_{k\leq l, 1\leq j\leq m}).$$

For tuples of runs $\bar{\rho} = (\rho_1, \dots, \rho_m)$ and $\bar{\rho}' = (\rho'_1, \dots, \rho'_m)$ we set $\bar{\rho} \; {}^{l}_{n_1}\equiv^z_{n_2} \bar{\rho}'$ if

$$\mathfrak{rA}_{l,n_1,n_2,z}(\bar{\rho}) \simeq \mathfrak{rA}_{l,n_1,n_2,z}(\bar{\rho}').$$

Remark 3.3.50.

- If $\bar{\rho} \; {}^{l}_{n_1}\equiv^z_{n_2} \bar{\rho}'$ then there is a unique isomorphism $\varphi : \mathfrak{rA}_{l,n_1,n_2,z}(\bar{\rho}) \simeq \mathfrak{rA}_{l,n_1,n_2,z}(\bar{\rho}')$ witnessing this equivalence. Note that due to the predicate P^0_j, ρ_j is mapped to ρ'_j for all $1 \leq j \leq m$. Due to the predicate P^l_j, the relevant ancestors of ρ_j are mapped to the relevant ancestors of ρ'_j. Finally, φ must preserve the order of the relevant ancestors of ρ_j because they form a chain with respect to $\vdash \cup \overset{+1}{\hookrightarrow}$ (cf. Proposition 3.3.17).
- Due to Lemma 3.3.9, it is clear that $\bar{\rho} \; {}^{l}_{n_1}\equiv^z_{n_2} \bar{\rho}'$ implies that there is a partial isomorphism mapping $\rho_i \mapsto \rho'_i$ for all $1 \leq i \leq m$.

Since equivalent relevant ancestors induce partial isomorphisms, a strategy that preserves the equivalence between relevant ancestors is winning for Duplicator in the Ehrenfeucht-Fraïssé-game.

Given a level 2 pushdown system $\mathcal{N}$ we are going to show that there is a strategy in the Ehrenfeucht-Fraïssé game on $\mathsf{NPT}(\mathcal{N}) =: \mathfrak{N}, \bar{\rho}$ and $\mathfrak{N}, \bar{\rho}'$ in which Duplicator can always choose small elements compared to the size of the elements chosen so far in the structure where he has to choose. Furthermore, this strategy will preserve equivalence of the relevant ancestors in the following sense. Let $\bar{\rho}, \bar{\rho}' \subseteq \mathfrak{N}$ be the n-tuples chosen in the previous rounds of the game. Assume that Duplicator managed to maintain the relevant ancestors of these tuples equivalent, i.e., it holds that $\bar{\rho} \overset{l}{k}\equiv^z_n \bar{\rho}'$. Now, Duplicators strategy enforces that these tuples are extended by runs π and π' satisfying the following. There are numbers k_i, l_i, n_i such that $\bar{\rho}, \pi \overset{l_i}{k_i}\equiv^z_{n_i} \bar{\rho}', \pi'$ and furthermore, the size of the run chosen by Duplicator is small compared to the elements chosen so far.

The exact claim is given in the following proposition.

Proposition 3.3.51. Let $\mathcal{N}$ be a level 2 pushdown system defining the higher order nested pushdown tree $\mathfrak{N} := \mathsf{NPT}(\mathcal{N})$. Given $\mathcal{N}$, we can compute functions

$$\begin{aligned}&\mathsf{BH} : \mathbb{N}^5 \to \mathbb{N},\\ &\mathsf{BW} : \mathbb{N}^5 \to \mathbb{N}, \text{and}\\ &\mathsf{BL} : \mathbb{N}^5 \to \mathbb{N}\end{aligned}$$

with the following property.

Let $n, z, n_1', n_2', l' \in \mathbb{N}$, $l := 4l' + 5, n_1 := n_1' + 2(l'+2) + 1$, and $n_2 := n_2' + 4^{l'+1} + 1$ such that $z \geq 2$ and $z > n \cdot 4^l$. Furthermore, let $\bar{\rho}$ and $\bar{\rho}'$ be n-tuples of runs of $\mathfrak{N}$ such that

1. $\bar{\rho} \overset{l}{n_1}\equiv^z_{n_2} \bar{\rho}'$, and
2. $\mathsf{ln}(\pi) \leq \mathsf{BL}(n, l, n_1, n_2, z)$ for all $\pi \in \mathsf{RA}_l(\bar{\rho}')$,
3. $\mathsf{hgt}(\pi) \leq \mathsf{BH}(n, z, l, n_1, n_2)$ for all $\pi \in \mathsf{RA}_l(\bar{\rho}')$, and
4. $\mathsf{wdt}(\pi) \leq \mathsf{BW}(n, z, l, n_1, n_2)$ for all $\pi \in \mathsf{RA}_l(\bar{\rho}')$.

For each $\rho \in \mathfrak{N}$ there is some $\rho' \in \mathfrak{N}$ such that

1. $\bar{\rho}, \rho \overset{l'}{n_1'}\equiv^z_{n_2'} \bar{\rho}', \rho'$,
2. $\mathsf{ln}(\pi) \leq \mathsf{BL}(n+1, l', n_1', n_2', z)$ for all $\pi \in \mathsf{RA}_{l'}(\bar{\rho}', \rho')$,
3. $\mathsf{hgt}(\pi) \leq \mathsf{BH}(n+1, z, l', n_1', n_2')$ for all $\pi \in \mathsf{RA}_l(\bar{\rho}', \rho')$, and
4. $\mathsf{wdt}(\pi) \leq \mathsf{BW}(n+1, z, l', n_1', n_2')$ for all $\pi \in \mathsf{RA}_l(\bar{\rho}', \rho')$.

In the next section we show how this proposition can be used to define an FO model checking algorithm on nested pushdown trees of level 2. The rest of this section proves the main proposition. For this purpose we split the claim into several pieces. The proposition asserts bounds on the length of the runs and on the sizes of the final stacks of the relevant ancestors. As the first step we prove that Duplicator has a strategy that chooses runs with small final stacks. This result relies mainly on the Proposition 3.3.29 and Proposition 3.3.39. These results allow to construct equivalent relevant ancestor sets that contain runs ending

in small stacks.[8] Afterwards, we apply the general bounds on short loops (cf. Proposition 2.4.48) in order to shrink the length of the runs involved.

The reader who is not interested in the details of the proof of Proposition 3.3.51, may skip this part and continue reading Section 3.3.6.

Construction of Isomorphic Relevant Ancestors

Before we prove that Duplicator can choose short runs, we state some auxiliary lemmas concerning the construction of isomorphic relevant ancestors. The following lemma gives a sufficient criterion that allows to check that the relevant ancestors of two runs are equivalent. Afterwards, we show that for each run ρ we can construct a second run ρ' satisfying this criterion.

Lemma 3.3.52. Let $\rho_0 \prec \rho_1 \prec \cdots \prec \rho_m = \rho$ be runs such that $\mathsf{RA}_l(\rho) = \{\rho_i : 0 \leq i \leq m\}$. If $\hat{\rho}_0 \prec \hat{\rho}_1 \prec \cdots \prec \hat{\rho}_m$ are runs such that

- the final states of ρ_i and $\hat{\rho}_i$ coincide,
- $\rho_0 = \mathsf{pop}_2^l(\rho_m)$ or $|\rho_0| = |\hat{\rho}_0| = 1$,
- $\rho_0 \; {}_{n_1}\!\equiv^z_{n_2} \hat{\rho}_0$, and
- $\rho_i * \rho_{i+1}$ iff $\hat{\rho}_i * \hat{\rho}_{i+1}$ for all $1 \leq i < m$ and $* \in \{\overset{+1}{\hookrightarrow}\} \cup \{\vdash^\gamma : \gamma \in \Gamma\}$,

then

$$\mathsf{RA}_l(\hat{\rho}_m) = \{\hat{\rho}_i : 0 \leq i \leq m\}.$$

If additionally $\mathsf{top}_2(\rho_i) \equiv^z_{n_2'} \mathsf{top}_2(\hat{\rho}_i)$, then

$$\hat{\rho}_m \; {}^{l}_{n_1}\!\equiv^z_{n_2'} \rho_m$$

for $n_2' := n_2 - 4^l$.

Proof. First, we show that for all $0 \leq i < j \leq m$, the following statements are true:

$$\rho_i \vdash^\gamma \rho_j \text{ iff } \hat{\rho}_i \vdash^\gamma \hat{\rho}_j, \tag{3.2}$$

$$\rho_i \hookrightarrow \rho_j \text{ iff } \hat{\rho}_i \hookrightarrow \hat{\rho}_j, \text{ and} \tag{3.3}$$

$$\rho_i \overset{+1}{\hookrightarrow} \rho_j \text{ iff } \hat{\rho}_i \overset{+1}{\hookrightarrow} \hat{\rho}_j. \tag{3.4}$$

Note that $\rho_i \vdash^\gamma \rho_j$ implies $j = i + 1$. Analogously, $\hat{\rho}_i \vdash^\gamma \hat{\rho}_j$ implies $j = i + 1$. Thus, 3.2 is true by definition of the sequences.

For the other parts, it is straightforward to see that $|\rho_k| - |\rho_j| = |\hat{\rho}_k| - |\hat{\rho}_j|$ for all $0 \leq j \leq k \leq m$: for $k = j$ the claim holds trivially. For the induction step from j to $j + 1$, the claim follows from the assumption that $\rho_j * \rho_{j+1}$ if and only if $\hat{\rho}_j * \hat{\rho}_{j+1}$ for all $* \in \{\overset{+1}{\hookrightarrow}\} \cup \{\vdash^\gamma : \gamma \in \Gamma\}$.

[8] In the following we sometimes say "Duplicator can choose small stacks". This expression always means that "Duplicator can choose a run such that all its relevant ancestors end in small stacks".

Furthermore, assume that there is some $\hat{\pi}$ such that $\hat{\rho}_k \prec \hat{\pi} \prec \hat{\rho}_{k+1}$. Then it cannot be the case that $\hat{\rho}_k \vdash^{\gamma} \hat{\rho}_{k+1}$. This implies that $\rho_k \overset{+1}{\hookrightarrow} \rho_{k+1}$. Due to our assumptions, it follows that $\hat{\rho}_k \overset{+1}{\hookrightarrow} \hat{\rho}_{k+1}$. We conclude directly that $|\hat{\pi}| \geq |\hat{\rho}_{k+1}| > |\hat{\rho}_k|$. Thus,

$$\begin{aligned}
&\rho_j \hookrightarrow \rho_k \text{ iff}\\
&|\rho_j| = |\rho_k| \text{ and } |\pi| > |\rho_j| \text{ for all } \rho_j \prec \pi \prec \rho_k \text{ iff}\\
&|\hat{\rho}_j| = |\hat{\rho}_k| \text{ and } |\hat{\pi}| > |\hat{\rho}_j| \text{ for all } \hat{\rho}_j \prec \hat{\pi} \prec \hat{\rho}_k \text{ iff}\\
&\hat{\rho}_j \hookrightarrow \hat{\rho}_k.
\end{aligned}$$

Analogously, one concludes that 3.4 holds.

We now show that $\mathrm{RA}_l(\hat{\rho}_m) = \{\hat{\rho}_i : 0 \leq i \leq m\}$. Note that

$$\mathrm{RA}_l(\hat{\rho}_m) \cap \{\pi : \hat{\rho}_m \preceq \pi\} = \{\hat{\rho}_m\}.$$

Now assume that there is some $0 \leq m_0 \leq m$ such that

$$\begin{aligned}
&\mathrm{RA}_l(\hat{\rho}_m) \cap \{\pi : \hat{\rho}_{m_0} \preceq \pi\} = \{\hat{\rho}_i : m_0 \leq i \leq m\} \text{ and}\\
&\rho_i \in \mathrm{RA}_k(\rho) \text{ iff } \hat{\rho}_i \in \mathrm{RA}_k(\hat{\rho}_m) \text{ for all } k \leq l \text{ and } i \geq m_0.
\end{aligned}$$

Now, we distinguish the following cases.

- If $\rho_{m_0-1} \vdash^{\mathrm{op}} \rho_{m_0}$ for some stack-operation op then $\hat{\rho}_{m_0-1} \vdash^{\mathrm{op}} \hat{\rho}_{m_0}$ due to 3.2. Thus, there are no runs $\rho_{m_0-1} \prec \pi \prec \rho_{m_0}$. Hence, we only have to show that $\rho_{m_0-1} \in \mathrm{RA}_k(\rho_m)$ if and only if $\hat{\rho}_{m_0-1} \in \mathrm{RA}_k(\hat{\rho}_m)$ for all $k \leq l$.

  If $\rho_{m_0-1} \in \mathrm{RA}_k(\rho_m)$, then there is some $j \geq m_0$ such that $\rho_j \in \mathrm{RA}_{k-1}(\rho_m)$ and ρ_{m_0-1} is connected to ρ_j via some edge. But then $\hat{\rho}_j \in \mathrm{RA}_{k-1}(\hat{\rho}_m)$ and $\hat{\rho}_{m_0-1}$ is connected with $\hat{\rho}_j$ via the same sort of edge. Thus, $\hat{\rho}_{m_0-1} \in \mathrm{RA}_k(\hat{\rho}_m)$.

  The other direction is completely analogous.

- Now, consider the case that there is some $\rho_{m_0-1} \prec \pi \prec \rho_{m_0}$. Since its direct predecessor is not in $\mathrm{RA}_l(\rho_m)$, $\rho_{m_0} \notin \mathrm{RA}_{l-1}(\rho)$. Thus, $\hat{\rho}_{m_0} \notin \mathrm{RA}_{l-1}(\hat{\rho})$. By construction of the $\hat{\rho}_i$, $\hat{\rho}_{m_0-1} \overset{+1}{\hookrightarrow} \hat{\rho}_{m_0}$. Thus, $|\hat{\pi}| \geq |\hat{\rho}_{m_0}|$ for all $\hat{\rho}_{m_0-1} \prec \hat{\pi} \prec \hat{\rho}_{m_0}$. This implies that $\pi \not\hookrightarrow \hat{\rho}_i$ and $\pi \overset{+1}{\not\hookrightarrow} \hat{\rho}_i$ for all $m_0 < i \leq m$. This shows that $\pi \notin \mathrm{RA}_l(\hat{\rho}_m)$.

  Now, for all $k \leq l$ we conclude completely analogous to the previous case that $\hat{\rho}_{m_0-1} \in \mathrm{RA}_k(\hat{\rho}_m)$ iff $\rho_{m_0-1} \in \mathrm{RA}_k(\rho_m)$.

Up to now, we have shown that $\mathrm{RA}_l(\hat{\rho}_m) \cap \{\pi : \hat{\rho}_0 \preceq \pi\} = \{\hat{\rho}_i : 0 \leq i \leq m\}$. In order to prove $\mathrm{RA}_l(\hat{\rho}_m) = \{\hat{\rho}_i : 0 \leq i \leq m\}$, we have to show that $\hat{\rho}_0$ is the minimal element of $\mathrm{RA}_l(\hat{\rho}_m)$.

There are the following cases

1. $\rho_0 = \mathrm{pop}_2^l(\rho_m)$. In this case, we conclude that $\hat{\rho}_0 = \mathrm{pop}_2^l(\hat{\rho}_m)$ by construction. But Lemma 3.3.12 then implies that $\hat{\rho}_0$ is the minimal element of $\mathrm{RA}_l(\hat{\rho}_m)$.

2. $|\rho_0| = |\hat{\rho}_0| = 1$. Note that $\rho_0 \notin \mathrm{RA}_{l-1}(\rho_m)$ because ρ_0 is minimal in $\mathrm{RA}_l(\rho_m)$. Thus, we know that $\hat{\rho}_0 \notin \mathrm{RA}_{l-1}(\hat{\rho}_m)$.

   Heading for a contradiction, assume that there is some $\hat{\pi} \in \mathrm{RA}_l(\hat{\rho}_m)$ with $\hat{\pi} \prec \hat{\rho}_0$. We conclude that $\hat{\pi} \hookrightarrow \hat{\rho}_k$ or $\hat{\pi} \overset{+1}{\hookrightarrow} \hat{\rho}_k$ for some $\hat{\rho}_k \in \mathrm{RA}_{l-1}(\hat{\rho}_m)$. But this implies that $|\hat{\pi}| < |\hat{\rho}_0| = 1$. Since there are no stacks of width 0, this is a contradiction.

   Thus, there is no $\hat{\pi} \in \mathrm{RA}_l(\hat{\rho}_m)$ that is a proper prefix of $\hat{\rho}_0$.

We conclude that $\mathrm{RA}_l(\hat{\rho}_m) = \{\hat{\rho}_i : 0 \leq i \leq m\}$.

Now, we prove the second part of the lemma. Assume that $\mathsf{top}_2(\rho_i) \equiv^z_{n_2'} \mathsf{top}_2(\hat{\rho}_i)$ for all $0 \leq i \leq m$. Since $\hat{\rho}_i$ and $\hat{\rho}_{i+1}$ differ in at most one word, a straightforward induction shows that $\rho_i \; {}_{n_1-|\rho_0|+|\rho_i|}\!\equiv^z_{n_2'} \hat{\rho}_i$ (cf. Proposition 3.3.29). But this implies $\hat{\rho}_m \; {}^{l}_{n_1}\!\equiv^z_{n_2'} \rho_m$ because $|\rho_0| \leq |\rho_i|$ as we have seen in Lemma 3.3.12. □

The previous lemma gives us a sufficient condition for the equivalence of relevant ancestors of two elements. Now, we show how to construct such a chain of relevant ancestors.

Lemma 3.3.53. Let $l, n_1, n_2, m, z \in \mathbb{N}$ such that $n_2 \geq 4^l$ and $z \geq 2$. Let

$$\begin{aligned}&\rho_0 \prec \rho_1 \prec \cdots \prec \rho_m = \rho \text{ be runs such that}\\ &\mathrm{RA}_l(\rho) \cap \{\pi : \rho_0 \preceq \pi \preceq \rho\} = \{\rho_i : 0 \leq i \leq m\}.\end{aligned}$$

Let $\hat{\rho}_0$ be a run such that $\rho_0 \; {}_{n_1}\!\equiv^z_{n_2} \hat{\rho}_0$. Then we can effectively construct runs

$$\hat{\rho}_0 \prec \hat{\rho}_1 \prec \cdots \prec \hat{\rho}_m =: \hat{\rho}$$

such that

- the final states of ρ_i and $\hat{\rho}_i$ coincide for all $0 \leq i \leq m$,
- $\rho_i \vdash^\gamma \rho_{i+1}$ iff $\hat{\rho}_i \vdash^\gamma \hat{\rho}_{i+1}$ and $\rho_i \overset{+1}{\hookrightarrow} \rho_{i+1}$ iff $\hat{\rho}_i \overset{+1}{\hookrightarrow} \hat{\rho}_{i+1}$ for all $0 \leq i < m$, and
- $\mathsf{top}_2(\rho_i) \equiv^z_{n_2-4^l} \mathsf{top}_2(\hat{\rho}_i)$ for all $0 \leq i \leq m$.

Proof. Assume we have constructed

$$\hat{\rho}_0 \prec \hat{\rho}_1 \prec \cdots \prec \hat{\rho}_{m_0},$$

for some $m_0 < m$ such that for all $0 \leq i \leq m_0$

1. the final states of ρ_i and $\hat{\rho}_i$ coincide,
2. $\rho_i \vdash^\gamma \rho_{i+1}$ iff $\hat{\rho}_i \vdash^\gamma \hat{\rho}_{i+1}$ and $\rho_i \overset{+1}{\hookrightarrow} \rho_{i+1}$ iff $\hat{\rho}_i \overset{+1}{\hookrightarrow} \hat{\rho}_{i+1}$ (note that either $\rho_i \vdash \rho_{i+1}$ or $\rho_i \overset{+1}{\hookrightarrow} \rho_{i+1}$ hold due to Proposition 3.3.17), and
3. $\mathsf{top}_2(\rho_i) \equiv^z_{n_2-i} \hat{\rho}_i$.

We extend this chain by a new element ρ'_{m_0+1} such that all these conditions are again satisfied. We distinguish two cases.

First, assume that $\rho_{m_0} \vdash^\gamma \rho_{m_0+1}$. Since $\rho_{m_0} \equiv^z_{n_2-m_0} \hat{\rho}_{m_0}$, $\mathsf{top}_1(\rho_{m_0}) = \mathsf{top}_1(\hat{\rho}_{m_0})$. Due to Condition 1, their final states also coincide. Hence, we can define $\hat{\rho}_{m_0+1}$ such that $\hat{\rho}_{m_0} \vdash^\gamma \hat{\rho}_{m_0+1}$. Due to Proposition 3.3.29, $\hat{\rho}_{m_0+1}$ satisfies Condition 3.

Now, consider the case $\rho_{m_0} \overset{+1}{\hookrightarrow} \rho_{m_0+1}$. The run from ρ_{m_0} to ρ_{m_0+1} starts from some stack s and ends in some stack $s : w$ for w some word, the first operation is a clone and then s is never reached again. Hence, we can use Proposition 3.3.39 in order to find some appropriate $\hat{\rho}_{m_0+1}$ that satisfies Condition 3. □

The previous lemmas give us the possibility to construct an isomorphic copy of the relevant ancestors of a single run ρ. In our proofs, we want to construct such a copy while avoiding relevant ancestors of certain other runs. Using the full power of Proposition 3.3.39 we obtain the following stronger version of the lemma.

Corollary 3.3.54. Let $l, n_1, n_2, m, z \in \mathbb{N}$ be numbers such that $z > m \cdot 4^l$ and $n_2 \geq 4^l$. As before, let $\mathrm{RA}_l(\rho_m) = \{\rho_i : 0 \leq i \leq m\}$ and $\hat{\rho}_0$ some run such that $\rho_0 \,{}_{n_1}\!\equiv^z_{n_2} \hat{\rho}_0$. Let $\bar{\rho}$ and $\bar{\rho}'$ be m-tuples such that $\bar{\rho} \,{}^{l}_{n_1}\!\equiv^z_{n_2} \bar{\rho}'$ and φ_l is an isomorphism witnessing this equivalence.

If $\rho_0 \in \mathrm{RA}_l(\bar{\rho})$, $\varphi_l(\rho_0) = \hat{\rho}_0$, and if $\rho_1 \notin \mathrm{RA}_l(\bar{\rho})$ then we can construct $\hat{\rho}_1, \hat{\rho}_2, \ldots, \hat{\rho}_m$ satisfying the conditions from the previous lemma but additionally with the property that $\hat{\rho}_1 \notin \mathrm{RA}_l(\bar{\rho}')$.

Proof. We distinguish two cases.

1. Assume that $\rho_0 \vdash \rho_1$. Due to the equivalence of ρ_0 and $\hat{\rho}_0$, we can apply the transition connecting ρ_0 with ρ_1 to $\hat{\rho}_0$ and obtain a run $\hat{\rho}_1$. We have to prove that $\hat{\rho}_1 \notin \mathrm{RA}_l(\bar{\rho}')$.

   Heading for a contradiction assume that $\hat{\rho}_1 \in \mathrm{RA}_l(\bar{\rho}')$. Then φ_l^{-1} preserves the edge between $\hat{\rho}_0$ and $\hat{\rho}_1$, i.e., $\rho_0 = \varphi_l^{-1}(\hat{\rho}_0) \vdash \varphi_l^{-1}(\hat{\rho}_1)$. But this implies that $\varphi_l^{-1}(\hat{\rho}_1) = \rho_1$ which contradicts the assumption that $\rho_1 \notin \mathrm{RA}_l(\bar{\rho})$.

2. Assume that $\rho_0 \overset{+1}{\hookrightarrow} \rho_1$. Up to threshold z, for each $\hat{\pi}$ such that $\hat{\rho}_0 \overset{+1}{\hookrightarrow} \hat{\pi}$ and $\hat{\pi} \in \mathrm{RA}_l(\bar{\rho}')$ there is a run $\rho_0 \overset{+1}{\hookrightarrow} \varphi_l^{-1}(\hat{\pi})$. Since $\rho_1 \notin \mathrm{RA}_l(\bar{\rho})$, we find another run $\hat{\rho}_1$ that satisfies the conditions of the previous lemma and $\hat{\rho}_1 \notin \mathrm{RA}_l(\bar{\rho}')$. This is due to the fact that Proposition 3.3.39 allows to transfer up to $z > |\mathrm{RA}_l(\bar{\rho}')|$ many runs simultaneously. □

Strategy for Choosing Small Stacks

By now we are prepared to prove that Duplicator has a strategy that preserves the isomorphism type of the relevant ancestors but chooses short runs. First, we prove the existence of a strategy choosing runs with small final stacks. Afterwards, we show how to bound the length of such runs. The analysis of this strategy decomposes into the local and the global case. We say Spoiler makes a local move if he chooses a new element such that the relevant ancestors of this element intersect with the relevant ancestors of elements chosen so far. In this case Duplicator has to extend the other tuple by an element whose relevant ancestors intersect with the relevant ancestors of this tuple.

We say Spoiler makes a global move if he chooses an element such that the relevant ancestors of this new element do not intersect with the relevant ancestors of the elements

chosen so far. In this case Duplicator has to extend the other tuple by an element whose relevant ancestors do not intersect with the relevant ancestors of this tuple.

We first head for the result that Duplicator can manage the local case in such a way that he chooses an element such that all its relevant ancestors end in small stacks. Then we show that Duplicator can manage the global case analogously. Finally, we show that Duplicator can choose a short run ending in small stacks.

Lemma 3.3.55. Let $n, z, l, l', n_1, n_1', n_2, n_2' \in \mathbb{N}$ be numbers such that $l = 4l' + 4$, $z > n \cdot 4^l$, $z \geq 2$, $n_1 = n_1' + 2(l'+1) + 1$, $n_1' > 0$, $n_2 = n_2' + 4^{l'+1} + 1$, and $n_2' > 0$.

Let $\bar{\rho}, \bar{\rho}'$ be n-tuples of runs such that $\varphi_l : \mathfrak{rA}_{l,n_1,n_2,z}(\bar{\rho}) \simeq \mathfrak{rA}_{l,n_1,n_2,z}(\bar{\rho}')$ witnesses $\bar{\rho} \overset{l}{\underset{n_1}{\equiv}}{}^{z}_{n_2} \bar{\rho}'$. Furthermore, let ρ be some run such that $\mathrm{RA}_{l'+1}(\rho) \cap \mathrm{RA}_{l'+1}(\bar{\rho}) \neq \emptyset$. Then there is some run ρ' such that $(\bar{\rho}, \rho) \overset{l'}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} (\bar{\rho}', \rho')$.

Proof. Let $\rho_0 \in \mathrm{RA}_{l'+1}(\rho)$ be maximal such that

$$\mathrm{RA}_{l'+1}(\rho) \cap \{\pi : \pi \preceq \rho_0\} \subseteq \mathrm{RA}_{4l'+3}(\bar{\rho}) \subseteq \mathrm{RA}_l(\bar{\rho}).$$

We choose numbers $m_0 \leq 0 \leq m_1$ and runs

$$\rho_{m_0} \prec \rho_{m_0+1} \prec \cdots \prec \rho_0 \prec \rho_1 \prec \cdots \prec \rho_{m_1}$$

such that $\mathrm{RA}_{l'+1}(\rho) = \{\rho_i : m_0 \leq i \leq m_1\}$. We set $\rho_i' := \varphi_l(\rho_i)$ for all $m_0 \leq i \leq 0$. Next, we construct $\rho_1', \ldots, \rho_{m_1}'$ such that $\rho' := \rho_{m_1}'$ has relevant ancestors isomorphic to those of ρ.

Analogously to the previous corollary, we can construct runs $\rho_1', \rho_2', \ldots, \rho_{m_1}'$ such that

1. $\rho_1' \notin \mathrm{RA}_{4l'+3}(\bar{\rho}')$,
2. the final states of ρ_i and ρ_i' coincide for all $0 \leq i \leq m_1$, and
3. $\rho_i \vdash^\gamma \rho_{i+1}$ iff $\rho_i' \vdash^\gamma \rho_{i+1}'$ and $\rho_i \overset{+1}{\hookrightarrow} \rho_{i+1}$ iff $\rho_i' \overset{+1}{\hookrightarrow} \rho_{i+1}'$ for all $0 \leq i < m_1$.

By definition, it is clear that condition 2 and 3 hold also for all $m_0 \leq i < 0$. Using Lemma 3.3.52, we obtain that $\rho \overset{l'+1}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \rho'$.

As a next step, we have to show that the isomorphism between $\mathrm{RA}_l(\bar{\rho})$ and $\mathrm{RA}_l(\bar{\rho}')$ and the isomorphism between $\mathrm{RA}_{l'}(\rho)$ and $\mathrm{RA}_{l'}(\rho')$ are compatible in the sense that they may be composed to an isomorphism between $\mathrm{RA}_{l'}(\bar{\rho}, \rho)$ and $\mathrm{RA}_{l'}(\bar{\rho}', \rho')$.

The only possible candidate for such a combined isomorphism is of the form

$$\varphi_{l'} : \mathrm{RA}_{l'}(\bar{\rho}, \rho) \to \mathrm{RA}_{l'}(\bar{\rho}', \rho')$$
$$\pi \mapsto \begin{cases} \rho_i' & \text{for } \pi = \rho_i, m_0 \leq i \leq m_1 \\ \varphi_l(\pi) & \text{for } \pi \in \mathrm{RA}_{l'+1}(\bar{\rho}). \end{cases}$$

In order to see that this is a well-defined function, we have to show that if $\rho_i \in \mathrm{RA}_{l'+1}(\bar{\rho})$ then $\rho_i' = \varphi_l(\rho_i)$ for each $m_0 \leq i \leq m_1$. Note that $\rho_i \in \mathrm{RA}_{l'+1}(\bar{\rho}) \cap \mathrm{RA}_{l'+1}(\rho)$ implies (using Corollary 3.3.16) that $\pi \in \mathrm{RA}_{3l'+3}(\bar{\rho})$ for all $\pi \in \mathrm{RA}_{l'+1}(\rho)$ with $\pi \preceq \rho_i$. But then by definition $i \leq 0$ and by construction $\rho_i' = \varphi_l(\rho_i)$.

We claim that $\varphi_{l'}$ is an isomorphism. Since we composed $\varphi_{l'}$ of existing isomorphisms $\mathrm{RA}_{l'}(\bar{\rho}) \simeq \mathrm{RA}_{l'}(\bar{\rho}')$ and $\mathrm{RA}_{l'}(\rho) \simeq \mathrm{RA}_{l'}(\rho')$, respectively, we only have to consider the following question: let $\pi \in \mathrm{RA}_{l'}(\bar{\rho})$ and $\hat{\pi} \in \mathrm{RA}_{l'}(\rho)$. Does $\varphi_{l'}$ preserve the existence and nonexistence of edges between π and $\hat{\pi}$?

In other words, we have to show that for each

$$* \in \{\hookrightarrow, \hookleftarrow, \overset{+1}{\hookrightarrow}, \overset{+1}{\hookleftarrow}\} \cup \{\vdash^\gamma \colon \gamma \in \Gamma\} \cup \{\dashv^\gamma \colon \gamma \in \Gamma\},$$
$$\pi * \hat{\pi} \text{ iff } \varphi_{l'}(\pi) * \varphi_{l'}(\hat{\pi}).$$

The following case distinction treats all these cases.

- Assume that there is some $* \in \{\hookrightarrow, \overset{+1}{\hookrightarrow}\} \cup \{\vdash^\gamma \colon \gamma \in \Gamma\}$ such that $\pi * \hat{\pi}$. Then $\pi \in \mathrm{RA}_{l'+1}(\rho)$. Thus, there are $m_0 \leq i < j \leq m_1$ such that $\pi = \rho_i$ and $\hat{\pi} = \rho_j$. We have already seen that then $\varphi_{l'}(\pi) = \rho'_i$ and $\varphi_{l'}(\hat{\pi}) = \rho'_j$ and these elements are connected by an edge of the same type due to the construction of ρ'_i and ρ'_j.

- Assume that there is some $* \in \{\hookleftarrow, \overset{+1}{\hookleftarrow}\} \cup \{\dashv^\gamma \colon \gamma \in \Gamma\}$ such that $\pi * \hat{\pi}$. Then $\hat{\pi} \in \mathrm{RA}_{l'+1}(\bar{\rho})$ whence $\varphi_{l'}$ coincides with the isomorphism φ_l on π and $\hat{\pi}$. But φ_l preserves edges whence $\pi * \hat{\pi}$ implies $\varphi_{l'}(\pi) * \varphi_{l'}(\hat{\pi})$.

- Assume that there is some $* \in \{\hookrightarrow, \overset{+1}{\hookrightarrow}\} \cup \{\vdash^\gamma \colon \gamma \in \Gamma\}$ such that $\varphi_{l'}(\pi) * \varphi_{l'}(\hat{\pi})$. By definition, $\varphi_{l'}(\hat{\pi}) \in \mathrm{RA}_{l'}(\rho')$ whence $\varphi_{l'}(\hat{\pi}) = \rho'_j$ for some $m_0 \leq j \leq m_1$. Thus, $\varphi_{l'}(\pi) \in \mathrm{RA}_{l'+1}(\rho')$ whence $\varphi_{l'}(\pi) = \rho'_i$ for some $m_0 \leq i < j$. We claim that $\pi = \rho_i$. Note that due to Corollary 3.3.16 for all $m_0 \leq k \leq i$ we have $\rho'_k \in \mathrm{RA}_{3l'+3}(\varphi_{l'}(\pi))$. Since $\varphi_{l'}(\pi) = \varphi_l(\pi) \in \mathrm{RA}_{l'}(\bar{\rho}')$, we conclude that $\rho'_k \in \mathrm{RA}_{4l'+3}(\bar{\rho}')$ for all $m_0 \leq k \leq i$. By construction, this implies $i \leq 0$ and $\varphi_{l'}(\pi) = \rho'_i = \varphi_l(\rho_i)$. Furthermore, since $\pi \in \mathrm{RA}_{l'}(\bar{\rho})$, $\varphi_{l'}(\pi) = \varphi_l(\pi)$. Since φ_l is an isomorphism, it follows that $\pi = \rho_i$. But this implies that there is an edge from $\pi = \rho_i$ to $\hat{\pi} = \rho_j$.

- Assume that there is some $* \in \{\hookleftarrow, \overset{+1}{\hookleftarrow}\} \cup \{\dashv^\gamma \colon \gamma \in \Gamma\}$ such that $\varphi_{l'}(\pi) * \varphi_{l'}(\hat{\pi})$. This implies

$$\varphi_{l'}(\hat{\pi}) \in \mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\rho). \tag{3.5}$$

  By definition, $\hat{\pi} = \rho_j$ and $\varphi_{l'}(\hat{\pi}) = \rho'_j$ for some $m_0 \leq j \leq m_1$. Due to 3.5, $\rho'_i \in \mathrm{RA}_{4l'+3}(\bar{\rho}')$ for all $m_0 \leq i \leq j$. Since $\rho'_1 \notin \mathrm{RA}_{4l'+3}(\bar{\rho}')$, $j \leq 0$. Thus, $\rho_j \in \mathrm{RA}_{4l'+3}(\bar{\rho})$ and $\varphi_{l'}(\hat{\pi}) = \varphi_l(\hat{\pi})$. Since φ_l preserves the relevant ancestors of $\bar{\rho}$ level by level, we obtain that $\hat{\pi} \in \mathrm{RA}_{l'+1}(\bar{\rho})$. Since $\pi \in \mathrm{RA}_{l'+1}(\bar{\rho})$, we obtain that $\varphi_{l'}(\pi) = \varphi_l(\pi)$ and $\varphi_{l'}(\hat{\pi}) = \varphi_l(\hat{\pi})$. Since φ_l is an isomorphism, we conclude that $\pi * \hat{\pi}$

Thus, we have shown that $\varphi_{l'}$ is an isomorphism witnessing $\bar{\rho}, \rho \overset{l'}{\underset{n'_1}{\equiv}}{}^{z}_{n'_2} \bar{\rho}', \rho'$. □

Due to the iterated use of Proposition 3.3.39 in the construction of ρ' we can require the following further properties for ρ':

Corollary 3.3.56. Let $n, z, l, n_1, n_2, l', n_1', n_2', \bar{\rho}, \bar{\rho}', \rho$ be as in the previous lemma. Let

$$H := \max\{\mathsf{hgt}(\pi) : \pi \in \mathrm{RA}_l(\bar{\rho}')\} \text{ and}$$
$$W := \max\{\mathsf{wdt}(\pi) : \pi \in \mathrm{RA}_l(\bar{\rho}')\}.$$

Set

$$g^{n,z}_{l',n_2}(x) := \mathrm{BH}_1(n \cdot 4^{l'+1}, x, n_2, z)$$

for BH_1 the monotone function defined in Proposition 3.3.39.

We can construct a run ρ' such that

$$\mathsf{hgt}(\rho') \leq (g^{n,z}_{l',n_2})^{(4^{l'+1})}(H),$$
$$\mathsf{wdt}(\rho') \leq W + 2l' + 2 \text{ and}$$
$$(\bar{\rho}, \rho) \,{}^{l'}_{n_1'}\!\equiv^{z}_{n_2'} (\bar{\rho}', \rho').$$

Proof. By construction of ρ_0', it is a relevant ancestor of $\bar{\rho}'$. It follows that $\mathsf{hgt}(\rho_0') \leq H$. Then an easy induction shows that $\mathsf{hgt}(\rho_i') \leq (g^{n,z}_{l',n_2})^i(H)$ for all $i \leq m_1$: in each step, we either apply Proposition 3.3.39 or ρ_{i+1}' is generated from ρ_i' by applying a single stack operation op. In the latter case we conclude by noting that $\mathsf{hgt}(\rho_{i+1}') \leq \mathsf{hgt}(\rho_i') + 1$. Note that g is a monotone function. Since $m_1 \leq 4^{l'+1} \leq |\mathrm{RA}_{l'+1}(\rho)|$, we conclude that $\mathsf{hgt}(\rho') \leq (g^{n,z}_{l',n_2})^{(4^{l'+1})}(H)$.

Now, consider the width of the ρ_i'. By assumption we know that $\mathsf{wdt}(\rho_i') \leq W$ for $m_0 \leq i \leq 0$. Furthermore, as all ρ_i' are relevant $l'+1$-ancestors of ρ', their width differ in at most $2l'+2$. Therefore, $\mathsf{wdt}(\rho_i') \leq W + 2l' + 2$ for all $m_0 \leq i \leq m_1$. □

Remark 3.3.57. Note that the monotonicity of BH_1 carries over to the monotonicity of g (in all parameters, i.e., in n, z, l', n_2, and x).

The previous lemmas showed that Duplicator can respond to local moves in such a way that she preserves isomorphisms of relevant ancestors while choosing small stacks.

Now, we deal with global moves of Spoiler. We present a strategy for Duplicator that allows to answer a global move by choosing a run with the following property. Duplicator chooses a run such that the isomorphism of relevant ancestors is preserved and such that all relevant ancestors of Duplicator's choice end in small stacks. We split this proof into several lemmas. First, we address the problem that Spoiler may choose an element far away from $\bar{\rho}$ but close to $\bar{\rho}'$. Then Duplicator has to find a run that has isomorphic relevant ancestors but that is far away from $\bar{\rho}'$.

Lemma 3.3.58. Let $l, l', n, z, n_1, n_1', n_2, n_2' \in \mathbb{N}$ be numbers such that $l > 3l' + 3$, $z > n \cdot 4^l$, $z \geq 2$, $n_1 > n_1' + 2(l'+1)$, and $n_2 > n_2' + 4^{l'+1}$. Let $\bar{\rho}$ and $\bar{\rho}'$ be n-tuples of runs such that $\bar{\rho} \,{}^{l}_{n_1}\!\equiv^{z}_{n_2} \bar{\rho}'$. Furthermore, let ρ be a run such that $\mathrm{RA}_{l'+1}(\bar{\rho}) \cap \mathrm{RA}_{l'+1}(\rho) = \emptyset$. Then there is some run ρ' such that $\bar{\rho}, \rho \,{}^{l'}_{n_1'}\!\equiv^{z}_{n_2'} \bar{\rho}', \rho'$.

Proof. We write φ_l for the isomorphism witnessing $\mathrm{RA}_l(\bar{\rho}) \overset{l}{\underset{n_1}{\equiv}}{}^{z}_{n_2} \mathrm{RA}_l(\bar{\rho}')$. If

$$\mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\rho) = \emptyset,$$

we can set $\rho' := \rho$ and we are done.

Otherwise, let $\pi_0^0 \prec \pi_1^0 \prec \cdots \prec \pi_{n_0}^0$ be an enumeration of all elements of $\mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\rho)$. Due to Corollary 3.3.19, $\mathrm{RA}_{l'+1}(\rho) \cap \{\pi : \pi \preceq \pi_{n_0}^0\} \subseteq \mathrm{RA}_{3l'+3}(\bar{\rho}')$. Since $l > 3l'+3$, we can set $\pi_i^1 := \varphi_l^{-1}(\pi_i^0)$ for all $0 \leq i \leq n_0$. Due to Lemma 3.3.53, there is an extension $\pi_{n_0}^1 \prec \rho^1$ such that $\mathrm{RA}_{l'+1}(\rho) \overset{l'+1}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \mathrm{RA}_{l'+1}(\rho^1)$ and $\pi_i^1 \in \mathrm{RA}_{l'+1}(\rho^1)$ for all $0 \leq i \leq n_0$. If $\mathrm{RA}_{l'+1}(\rho^1) \cap \mathrm{RA}_{l'+1}(\bar{\rho}') = \emptyset$ we set $\rho' := \rho^1$ and we are done.

Otherwise we can repeat this process, defining $\pi_i^2 := \varphi_l^{-1}(\pi_i^1)$ for the maximal $n_1 \leq n_0$ such that $\pi_i^1 \in \mathrm{RA}_{3l'+3}(\bar{\rho}')$ for all $0 \leq i \leq n_0$. Then we extend this run to some run ρ^2. If this process terminates with the construction of some run ρ^i such that $\mathrm{RA}_{l'+1}(\rho^i) \cap \mathrm{RA}_{l'+1}(\bar{\rho}') = \emptyset$, we set $\rho' := \rho^i$ and we are done. If this is not the case, recall that $\mathrm{RA}_{3l'+3}(\bar{\rho}')$ is finite. Thus, we eventually reach the step were we have defined $\pi_0^0, \pi_0^1, \ldots, \pi_0^m$ for some $m \in \mathbb{N}$ such that for the first time $\pi_0^m = \pi_0^i$ for some $i < m$. But if $i > 0$, then

$$\pi_0^{m-1} = \varphi_l(\pi_0^m) = \varphi_l(\pi_0^i) = \pi_0^{i-1}.$$

But this contradicts the minimality of m. We conclude that $\pi_0^m = \pi_0^0$ which implies that $\pi_0^0 \in \mathrm{RA}_{3l'+3}(\bar{\rho})$. Furthermore, by definition we have $\pi_0^0 \in \mathrm{RA}_{l'+1}(\rho)$ and there is a maximal i such that $\pi_i^0 \in \mathrm{RA}_{3l'+3}(\bar{\rho})$. Since $z > |\mathrm{RA}_l(\bar{\rho})|$, we can apply Lemma 3.3.53 and construct a chain $\varphi_l(\pi_i^0) \prec \rho_{i+1}' \prec \rho_{i+2}' \prec \cdots \prec \rho'$ such that $\bar{\rho}, \rho \overset{l'}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \bar{\rho}', \rho'$. □

The previous lemma showed that there is an answer to every global challenge of Spoiler. In the following, we use the pumping constructions from Lemmas 3.3.31 - 3.3.36 in order to show that Duplicator may answer global moves with a ρ' such that

$$\mathrm{RA}_l(\bar{\rho}, \rho) \overset{l'+1}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \mathrm{RA}_l(\bar{\rho}', \rho')$$

and such that $\mathrm{RA}_l(\bar{\rho}', \rho')$ only contains runs that end in small stacks.

Before we state this lemma, we have to give a precise notion of small stacks. For this purpose, we introduce the following functions.

Definition 3.3.59. Let $l, l', n, n_1, n_2, z \in \mathbb{N}$ such that $l \geq 3l'+3$. Set

$$\beta : \{-n_1, -n_1+1, \ldots, 4^{(l'+1)}\} \times \mathbb{N} \to \mathbb{N}$$

$$\beta(i, H) := \begin{cases} H + \mathrm{B}_{\mathrm{hgt}} + \mathrm{BTW}(n_1 + n_2 + 4^l, k, z) & \text{for } i = -n_1 \\ \beta(i-1, H) + \mathrm{BH}_1(0, \beta(i-1, H), n_2 + 4^{l'+1} - i, k, z) & \text{otherwise,} \end{cases}$$

and

$$\alpha(n_1, H, W, l') := \max\left\{W, \mathrm{BWW}(\beta(-n_1, H)) + n_1 + 2(l'+1)\right\}.$$

where $\mathrm{BH}_1, \mathrm{BTW}, \mathrm{BWW}$ the monotone functions from Proposition 3.3.39 and Lemmas 3.3.31 and 3.3.36, and $\mathrm{B}_{\mathrm{hgt}}$ the constant from Lemma 3.3.33.

Lemma 3.3.60. Let $l, l', n, n_1, n_2, z \in \mathbb{N}$ such that $l \geq 3l' + 3$. Furthermore, let $\bar{\rho}$ be an n-tuple of runs and ρ a run such that $\mathrm{RA}_{l'+1}(\bar{\rho}) \cap \mathrm{RA}_{l'+1}(\rho) = \emptyset$. Let $H, W \in \mathbb{N}$ be bounds such that $\mathsf{hgt}(\pi) \leq H$ and $\mathsf{wdt}(\pi) \leq W$ for all $\pi \in \mathrm{RA}_l(\bar{\rho})$. There is some run ρ' such that

$$\begin{aligned}
&\bar{\rho}, \rho \overset{l'}{}_{n_1}\!\equiv^z_{n_2} \bar{\rho}, \rho', \\
&\mathsf{hgt}(\pi) \leq \beta(4^{l'+1}, H), \text{ and} \\
&\mathsf{wdt}(\pi) \leq \alpha(n_1, H, W, l')
\end{aligned}$$

for all $\pi \in \mathrm{RA}_{l'}(\bar{\rho}, \rho')$.

Proof. Let $\rho_0 \prec \rho_1 \prec \cdots \prec \rho_m := \rho$ be runs such that $\mathrm{RA}_{l'+1}(\rho) = \{\rho_i : 0 \leq i \leq m\}$. We have to find an isomorphic copy of $\mathrm{RA}_{l'+1}(\rho)$ consisting of small words but not intersecting with $\mathrm{RA}_{l'+1}(\bar{\rho})$. Using Lemmas 3.3.52, 3.3.53 and 3.3.39, we can construct such an isomorphic copy as soon as we find some small ρ'_0 with $\rho'_0 \overset{}{}_{n_1}\!\equiv^z_{n_2+4^l} \rho_0$. Thus, as a first step we construct such a run ρ'_0.

Let $m_0 \geq -n_1$ be minimal such that there are runs $\rho_{m_0} \overset{+1}{\hookrightarrow} \rho_{m_0+1} \overset{+1}{\hookrightarrow} \ldots \overset{+1}{\hookrightarrow} \rho_0$. Note that by Lemma 3.3.12 either $m_0 = -n_1$ or $\mathsf{wdt}(\rho_0) \leq n_1$.

If $\mathsf{hgt}(\rho_{m_0}) \leq H$ and $\mathsf{wdt}(\rho_{m_0}) \leq W$, then we choose m_1 maximal such that $\mathsf{hgt}(\rho_i) \leq \beta(i, H)$ and $\mathsf{wdt}(\rho_i) \leq W$ for all $m_0 \leq i \leq m_1$. In this case, we set $\rho'_{m_1} := \rho_{m_1}$. Otherwise, we set $m_1 := m_0$ and by Lemmas 3.3.31,3.3.33, and 3.3.36, there is a run ρ'_{m_1} such that

$$\begin{aligned}
&\mathsf{hgt}(\rho'_{m_1}) \leq \beta(-n_1, H) \leq \beta(m_1, H), \\
&\mathsf{wdt}(\rho'_{m_1}) \leq \mathsf{BWW}(\beta(-n_1, H)), \text{ and} \\
&\rho_{m_1} \overset{}{}_{1}\!\equiv^z_{n_2+n_1+4^l} \rho'_{m_1}.
\end{aligned}$$

The last condition just says that $\mathsf{top}_2(\rho_{m_1}) \equiv^z_{n_2+n_1+4^l} \mathsf{top}_2(\rho'_{m_1})$. Furthermore, we construct ρ'_{m_1} in such a way that either $\mathsf{hgt}(\rho'_{m_1}) > H$ or $\mathsf{wdt}(\rho'_{m_1}) > W$.

Having constructed ρ'_{m_1} according to one of the two cases, in both cases we continue with the following construction. Note that $\rho'_i = \rho_i$ for all $m_0 \leq i \leq m_1$ or $H < \mathsf{hgt}(\rho'_{m_1})$ or $W < \mathsf{wdt}(\rho'_{m_1})$.

By Proposition 3.3.39, we can construct $\rho'_{m_1} \prec \rho'_{m_1+1} \prec \cdots \prec \rho'_m =: \rho'$ such that the following holds.

1. For $* \in \{\overset{+1}{\hookrightarrow}, (\vdash^{\mathrm{op}})_{\mathrm{op} \in \mathrm{OP}}\}$ and for all $m_1 \leq i < m$, $\rho'_i * \rho'_{i+1}$ iff $\rho_i * \rho_{i+1}$.
2. $\mathsf{top}_2(\rho_i) \equiv^z_{n_2} \mathsf{top}_2(\rho'_i)$.
3. $\mathsf{hgt}(\rho'_i) \leq \beta(i, H)$ and $\mathsf{wdt}(\rho'_i) \leq \mathsf{wdt}(\rho'_{m_1}) + n_1 + 2l' + 2$ for all $m_1 \leq i \leq m$.
4. $\rho'_i = \rho_i$ iff for all $m_1 \leq j \leq i$ we have $\mathsf{hgt}(\rho_j) \leq H$ and $\mathsf{wdt}(\rho_j) \leq W$ (this just requires to construct ρ'_{m_0+1} such that $\mathsf{hgt}(\rho'_{m_0+1}) > H$ or $\mathsf{wdt}(\rho'_{m_0+1}) > W$.

From Lemma 3.3.52, it follows that $\rho' \overset{l'+1}{}_{n_1}\!\equiv^z_{n_2} \rho$.

Furthermore, $\mathrm{RA}_{l'+1}(\rho') \cap \mathrm{RA}_{l'+1}(\bar{\rho}) = \emptyset$: heading for a contradiction, assume that

$$\rho'_i \in \mathrm{RA}_{l'+1}(\rho') \cap \mathrm{RA}_{l'+1}(\bar{\rho})$$

for some $0 \leq i \leq m$. Then $\rho'_j \in \mathrm{RA}_{3l'+3}(\bar{\rho}) \subseteq \mathrm{RA}_l(\bar{\rho})$ for all $0 \leq j \leq i$. Thus,

$$\begin{aligned} &\mathrm{hgt}(\rho'_j) \leq H \leq \beta(j, H) \text{ and} \\ &\mathrm{wdt}(\rho'_j) \leq W. \end{aligned}$$

This implies that $\rho'_j = \rho_j$ for all $m_0 \leq i \leq j$. But then $\rho_j = \rho'_j \in \mathrm{RA}_{l'+1}(\rho) \cap \mathrm{RA}_{l'+1}(\bar{\rho})$ which contradicts our assumptions on $\bar{\rho}$ and ρ.

Hence, $\mathrm{RA}_{l'}(\bar{\rho})$ and $\mathrm{RA}_{l'}(\rho')$ do not touch whence $\bar{\rho}, \rho \overset{l'}{}_{n_1}\!\!\equiv^z_{n_2} \bar{\rho}, \rho'$. □

Combining the previous lemmas, we obtain a proof that for each n-tuple in $\mathfrak{N}(\mathscr{S})$ there is an FO_k-equivalent one such that the relevant ancestors of the second tuple only contain runs that end in small stacks. This result is summarised in the following corollary.

Corollary 3.3.61. Let $\mathfrak{N}$ be a 2-NPT. There are monotone functions

$$\begin{aligned} &\mathrm{BH} : \mathbb{N}^5 \to \mathbb{N} \text{ and} \\ &\mathrm{BW} : \mathbb{N}^5 \to \mathbb{N} \end{aligned}$$

such that the following holds. Let $n, n'_1, n'_2, l' \in \mathbb{N}$. We set

$$\begin{aligned} l &:= 4l' + 5, \\ n_1 &:= n'_1 + 2(l'+2) + 1 \text{ and} \\ n_2 &:= n'_2 + 4^{l'+1} + 1. \end{aligned}$$

Let $z \in \mathbb{N}$ such that $z \geq 2$ and $z > n \cdot 4^l$.

For all pairs of n-tuples $\bar{\rho} = \rho_1, \ldots, \rho_n \in \mathfrak{N}$, $\bar{\rho}' = \rho'_1, \ldots, \rho'_n \in \mathfrak{N}$ such that

$$\begin{aligned} &\mathrm{hgt}(\rho'_i) \leq \mathrm{BH}(n, z, l, n_1, n_2), \\ &\mathrm{wdt}(\rho'_i) \leq \mathrm{BW}(n, z, l, n_1, n_2), \text{ and} \\ &\bar{\rho} \overset{l}{}_{n_1}\!\!\equiv^z_{n_2} \bar{\rho}', \end{aligned}$$

and for all runs $\rho \in \mathfrak{N}(\mathscr{S})$, there is a run ρ' such that

$$\begin{aligned} &\bar{\rho}, \rho \overset{l'}{}_{n'_1}\!\!\equiv^z_{n'_2} \bar{\rho}', \rho', \\ &\mathrm{hgt}(\rho') \leq \mathrm{BH}(n+1, z, l', n'_1, n'_2), \text{ and} \\ &\mathrm{wdt}(\rho') \leq \mathrm{BW}(n+1, z, l', n'_1, n'_2). \end{aligned}$$

Proof. The proof is by induction on n. Assume that we have defined $\mathrm{BH}(x_1, x_2, x_3, x_4, x_5)$ and $\mathrm{BW}(x_1, x_2, x_3, x_4, x_5)$ for all $x_2, \ldots, x_5 \in \mathbb{N}$ and $x_1 \leq n$ such that for all tuples where $x_1 \leq n$ the claim holds. For $\bar{x}_n := (n, z, l, n_1, n_2)$ and $\bar{x}_{n+1} := (n+1, z, l', n'_1, n'_2)$, we set

$$\begin{aligned} \mathrm{BH}(\bar{x}_{n+1}) &:= \max\left\{\beta(4^{l'+1}, \mathrm{BH}(\bar{x}_n)), (g^{n,z}_{l',n_2})^{4^{l'+1}}(\mathrm{BH}(\bar{x}_n))\right\} \text{ and} \\ \mathrm{BW}(\bar{x}_{n+1}) &:= \max\left\{\mathrm{BWW}\left(\beta(4^{(l'+1)}, \mathrm{BH}(\bar{x})) + n'_1 + 2l' + 2\right), \mathrm{BW}(\bar{x}_n) + 2l' + 2\right\} \end{aligned}$$

where $g^{n,z}_{l',n_2}$ is the function from Corollary 3.3.56 and β the function from Lemma 3.3.60. The following case distinction proves that this definition satisfies the claim.

1. First assume that $\mathrm{RA}_{l'+1}(\rho) \cap \mathrm{RA}_{l'+1}(\bar{\rho}) \neq \emptyset$. Then we can apply Lemma 3.3.55 and obtain an element $\rho' \in \mathfrak{N}(\mathscr{S})$ such that $\bar{\rho}, \rho \overset{l'}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \bar{\rho}', \rho'$. Furthermore, by Corollary 3.3.56, ρ' can be chosen such that

$$\begin{aligned}&\mathrm{hgt}(\rho') \leq (g^{n,z}_{l',n_2})^{4^{l'+1}}(\mathrm{BH}(n,z,l,n_1,n_2)) = (g^{n,z}_{l',n_2})^{4^{l'+1}}(\mathrm{BH}(\bar{x}_n)) \text{ and}\\ &\mathrm{wdt}(\rho') \leq \mathrm{BW}(\bar{x}_n) + 2l' + 2.\end{aligned}$$

2. Otherwise, $\mathrm{RA}_{l'+1}(\rho) \cap \mathrm{RA}_{l'+1}(\bar{\rho}) = \emptyset$. Since $l > 4l' + 4$, we can apply Lemmas 3.3.58 and 3.3.60 and obtain $\rho' \in \mathfrak{N}(\mathscr{S})$ such that

$$\begin{aligned}&\bar{\rho}, \rho \overset{l'}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \bar{\rho}', \rho',\\ &\mathrm{hgt}(\rho') \leq \beta\left(4^{l'+1}, \mathrm{BH}(n,z,l,n_1,n_2)\right) = \beta\left(4^{l'+1}, \mathrm{BH}(\bar{x}_n)\right), \text{ and}\\ &\mathrm{wdt}(\rho') \leq \mathrm{BWW}\left(\beta(4^{(l'+1)}, \mathrm{BH}(\bar{x}_n)) + n_1' + 2(l'+2)\right).\end{aligned}$$

By induction, our definition satisfies the claim. Note that the monotonicity of BH and BW follows from the monotonicity of all the components involved in the definition. □

Strategy for Bounding the Length of Runs

For each relevant ancestor set, there is an equivalent one which only contains runs that end in small stacks. But the runs leading to these stacks can still be arbitrary long. In the next lemmas, we show that we can also bound the length. For this proof, the Corollaries 2.4.49 and 2.4.50 are important tools because they allow to replace long runs by shorter ones.

Lemma 3.3.62. Let $\mathscr{N}$ be a level 2 pushdown system defining the higher order nested pushdown tree $\mathfrak{N} := \mathrm{NPT}(\mathscr{N})$. We can compute a function $\mathrm{BL} : \mathbb{N}^5 \to \mathbb{N}$ such that the following hold:

Let $n, z, n_1', n_2', l' \in \mathbb{N}$, $l := 4l' + 5$, $n_1 := n_1' + 2(l'+2) + 1$, and $n_2 := n_2' + 4^{l'+1} + 1$ such that $z \geq 2$ and $z > n \cdot 4^l$. Furthermore, let $\bar{\rho}$ and $\bar{\rho}'$ be n-tuples of runs of $\mathfrak{N}$ such that

1. $\bar{\rho} \overset{l}{\underset{n_1}{\equiv}}{}^{z}_{n_2} \bar{\rho}'$, and
2. $\mathrm{ln}(\pi) \leq \mathrm{BL}(n,l,n_1,n_2,z)$ for all $\pi \in \mathrm{RA}_l(\bar{\rho}')$,
3. $\mathrm{hgt}(\pi) \leq \mathrm{BH}(n,z,l,n_1,n_2)$ for all $\pi \in \mathrm{RA}_l(\bar{\rho}')$, and
4. $\mathrm{wdt}(\pi) \leq \mathrm{BW}(n,z,l,n_1,n_2)$ for all $\pi \in \mathrm{RA}_l(\bar{\rho}')$.

For each $\rho \in \mathfrak{N}$ there is some $\rho' \in \mathfrak{N}$ such that

1. $\bar{\rho}, \rho \overset{l'}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \bar{\rho}', \rho'$,
2. $\mathrm{ln}(\pi) \leq \mathrm{BL}(n+1,l',n_1',n_2',z)$ for all $\pi \in \mathrm{RA}_{l'}(\bar{\rho}', \rho')$,
3. $\mathrm{hgt}(\pi) \leq \mathrm{BH}(n+1,z,l',n_1',n_2')$ for all $\pi \in \mathrm{RA}_l(\bar{\rho}', \rho')$, and
4. $\mathrm{wdt}(\pi) \leq \mathrm{BW}(n+1,z,l',n_1',n_2')$ for all $\pi \in \mathrm{RA}_l(\bar{\rho}', \rho')$.

Proof. Using the Lemmas 3.3.55 – 3.3.60, we find some candidate ρ' such that

$$\bar{\rho}, \rho \overset{l'}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \bar{\rho}', \hat{\rho}'$$

and the height and width of the last stacks of all $\pi \in \mathrm{RA}_{l'+1}(\hat{\rho}')$ are bounded by $\mathrm{BH}(n+1, z, l', n_1', n_2')$ and $\mathrm{BW}(n+1, z, l', n_1', n_2')$, respectively.

Recall that there is a chain $\hat{\rho}_0' \prec \hat{\rho}_1' \prec \cdots \prec \hat{\rho}_m' = \hat{\rho}'$ for some $0 \leq m \leq 4^{(l'+1)}$ with $\hat{\rho}_i' \vdash^{\gamma} \hat{\rho}_{i+1}'$ or $\hat{\rho}_i' \overset{+1}{\hookrightarrow} \hat{\rho}_{i+1}'$ for all $0 \leq i < m$ such that $\mathrm{RA}_{l'+1}(\hat{\rho}') = \{\hat{\rho}_i' : 0 \leq i \leq m\}$.

If $\hat{\rho}_0' \notin \mathrm{RA}_{3l'+3}(\bar{\rho}')$, then we can use Corollary 2.4.49 and choose some ρ_0' that ends in the same configuration as $\hat{\rho}_0'$ such that $\rho_0' \notin \mathrm{RA}_{3l'+3}(\bar{\rho}')$ and

$$\begin{aligned}\ln(\rho_0') \leq 1 + 2 \cdot \mathrm{BH}(n+1, z, l', n_1', n_2') \cdot \mathrm{BW}(n+1, z, l', n_1', n_2')) \\ \cdot (1 + \mathrm{BLL}_z^{\mathcal{N}}(\mathrm{BH}(n+1, z, l', n_1', n_2'))).\end{aligned}$$

If $\hat{\rho}_0' \in \mathrm{RA}_{3l'+3}(\bar{\rho}')$ let $0 \leq i \leq m$ be maximal such that $\hat{\rho}_i' \in \mathrm{RA}_l(\bar{\rho}')$. In this case let $\rho_j' := \hat{\rho}_j'$ for all $0 \leq j \leq i$.

By now, we have obtained a chain $\rho_0' \prec \rho_1' \prec \cdots \prec \rho_i'$ for some $0 \leq i \leq m$. Using the previous lemma, we can extend this chain to a chain $\{\rho_i' : 0 \leq i \leq m\}$ such that

1. $\rho_i'(\ln(\rho_i')) = \hat{\rho}_i(\ln(\hat{\rho}_i))$,
2. $\rho_i' \vdash^{\gamma} \rho_{i+1}'$ iff $\hat{\rho}_i' \vdash^{\gamma} \hat{\rho}_{i+1}'$ for all $0 \leq i < m$,
3. $\rho_i' \overset{+1}{\hookrightarrow} \rho_{i+1}'$ iff $\hat{\rho}_i' \overset{+1}{\hookrightarrow} \hat{\rho}_{i+1}'$ for all $0 \leq i < m$,
4. $\ln(\rho_{i+1}') \leq \ln(\rho_i') + 2 \cdot \mathrm{BH}(n+1, z, l', n_1', n_2') \cdot (1 + \mathrm{BLL}_z^{\mathcal{N}}(\mathrm{BH}(n+1, z, l', n_1', n_2')))$, and
5. $\hat{\rho}_j' \in \mathrm{RA}_{3l'+3}(\bar{\rho}')$ for all $0 \leq j \leq i$ implies $\rho_i' = \hat{\rho}_i'$.

Assume that we have constructed the chain up to $\rho_0' \prec \cdots \prec \rho_{m_0}'$ for some $m_0 < m$. Note that $\hat{\rho}_{m_0+1} \notin \mathrm{RA}_{3l'+3}(\bar{\rho}')$ by definition of the initial segment of the ρ_i'. We can use Corollary 2.4.50 in order to construct ρ_{m_0+1}' as required. In this construction, we can enforce $\rho_{m_0+1}' \notin \mathrm{RA}_{3l'+3}(\bar{\rho}')$ if $\rho_{m_0}' = \hat{\rho}_{m_0}'$.

Using Lemma 3.3.52, we conclude that $\rho' \overset{l'+1}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \hat{\rho}'$ for $\rho' := \rho_m'$. Furthermore, we claim that $\mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\hat{\rho}') = \mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\rho')$. By definition the inclusion from left to right is clear. For the other direction, assume that there is some element $\rho_i' \in \mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\rho')$. By Lemma 3.3.16, this implies that $\rho_j' \in \mathrm{RA}_{3l'+3}(\bar{\rho}')$ for all $0 \leq j \leq i$. Thus, $\rho_i' = \hat{\rho}_i'$, which implies that $\rho_i' \in \mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\rho')$.

We conclude that $\bar{\rho}, \rho \overset{l'}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \bar{\rho}', \hat{\rho}' \overset{l'}{\underset{n_1'}{\equiv}}{}^{z}_{n_2'} \bar{\rho}', \rho'$ because $\mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\hat{\rho}')$ is isomorphic to $\mathrm{RA}_{l'+1}(\bar{\rho}') \cap \mathrm{RA}_{l'+1}(\hat{\rho}')$. By definition, the length of ρ' is bounded by a polynomial in

$$\begin{aligned}&\mathrm{BH}(n+1, z, l', n_1', n_2'), \\ &\mathrm{BW}(n+1, z, l', n_1', n_2'), \\ &\mathrm{BLL}_z^{\mathcal{N}}(\mathrm{BH}(n+1, z, l', n_1', n_2')), \text{ and} \\ &\mathrm{BL}(n, l, n_1, n_2, z).\end{aligned}$$

This polynomial can be used to inductively define $\mathrm{BL}(n+1, l', n_1', n_2', z)$. □

Note that the previous lemma completes the proof of Proposition 3.3.51.

3.3.7 FO Model Checking Algorithm for Level 2 Nested Pushdown Trees

In the previous section, we have shown that each existential quantification on a nested pushdown tree $\mathfrak{N} := \mathrm{NPT}(\mathcal{N})$ can be witnessed by a run ρ of small length. Even when we add parameters $\rho_1, \ldots, \rho_n$ this result still holds, in the sense that there is a witness ρ of small length compared to the length of the parameters. Hence, we can decide first-order logic on level 2 nested pushdown trees with the following algorithm.

1. Given the pushdown system $\mathcal{N}$ and a first-order formula φ, the algorithm first computes the quantifier rank q of φ.
2. Then it computes numbers $z, l^1, l^2, l^3, \ldots, l^q, n_1^1, n_1^2, n_1^3 \ldots, n_1^q, n_2^1, n_2^2, n_2^3, \ldots, n_2^q \in \mathbb{N}$ such that for each $i < q$ the numbers $z, l^i, l^{i+1}, n_1^i, n_1^{i+1}, n_2^i, n_2^{i+1}$ can be used as parameters in Proposition 3.3.51.
3. These numbers define a constraint $S = (S^{\mathfrak{N}}(i))_{i \leq q}$ for Duplicator's strategy in the q-round game on $\mathfrak{N}$ and $\mathfrak{N}$ as follows. We set $(\rho_1, \rho_2, \ldots, \rho_m) \in S_m^{\mathfrak{N}}$ if for each $i \leq m$ and $\pi \in \mathrm{RA}_{l_i}(\rho_i)$

$$\begin{aligned} &\ln(\pi) \leq \mathrm{BL}(i, l^i, n_1^i, n_2^i, z), \\ &\mathrm{hgt}(\pi) \leq \mathrm{BH}(i, z, l^i, n_1^i, n_2^i), \text{ and} \\ &\mathrm{wdt}(\pi) \leq \mathrm{BW}(i, z, l^i, n_1^i, n_2^i). \end{aligned}$$

4. Due to Lemma 3.3.62, Duplicator has an S-preserving strategy in the q-round game on $\mathfrak{N}$ and $\mathfrak{N}$. Thus, applying the algorithm SModelCheck (cf. Algorithm 2 in Section 2.1.1) decides whether $\mathfrak{N} \models \varphi$.

Complexity of the Algorithm

For the case of nested pushdown trees (of level 1) our approach resulted in an 2-EXPSPACE FO model checking algorithm. In the case of level 2 nested pushdown trees, we cannot prove such a nice result. At the moment, we cannot prove an elementary complexity bound for the FO model checking on 2-NPT because we cannot determine the length of short loops. Our algorithm can only be efficient if we have a good bound on the length of the k shortest loops of any stack because we use loops as a main ingredient in the construction of equivalent relevant ancestors. But such a good bound is not known to exist. We do not know any elementary algorithm that, given a level 2 pushdown system $\mathcal{N}$ and a number k, calculates the shortest k loops from $(q_0, \bot)$ to $(q_1, \bot)$ of $\mathcal{N}$. The underlying problem is that we cannot derive an elementary bound on the length of such loops. The best bound we know can be derived as follows.

From Hayashi's pumping lemma for indexed grammars [30], we can derive that the shortest loop of $\mathcal{N}$ has size $\exp(\exp(\exp(p(|\mathcal{N}|))))$ for some polynomial p. Unfortunately, Hayashi's pumping lemma does not yield any bound on the second shortest loop. Thus, the only known way of calculating the second shortest loop is to design a copy of the pushdown system which simulates the first one but avoids this first loop. This involves increasing the number of states by the length of the shortest loop, i.e., we design a system $\mathcal{N}'$ with $|\mathcal{N}'| \approx \exp(\exp(\exp(p(|\mathcal{N}|))))$ many states. Using this system we obtain a 6-fold exponential

bound in $|\mathcal{N}|$ for the second shortest loop of $\mathcal{N}$ (which is the shortest one of $\mathcal{N}'$) the same way as we obtained the bound for the first loop. Thus, the best bound known for the k shortest loops is an exponential tower of height $3k$ in the size of the pushdown system. But it is quite clear that there are level 2 nested pushdown trees where we can define the existence of k loops from $(q_0, \bot)$ to $(q_1, \bot)$ by a first-order formula of quantifier rank linear in k. Thus, our model checking algorithm would have to choose k short loops. Given the bounds on short loops, we expect that our algorithm then needs space up to a tower of exponentials of height $3k$ in order to verify this formula. Since k is arbitrary, the algorithm has nonelementary space consumption in the quantifier rank of the formula.

It remains open to determine the exact complexity of our algorithm. We neither know whether our algorithm has elementary complexity nor do we know a good lower bound on the complexity of model checking on nested pushdown trees of level 2. These questions require further study.

3.4 Decidability of Ramsey Quantifiers on Tree-Automatic Structures

Recently, Rubin [57] proved the decidability of Ramsey quantifiers on string-automatic structures using the concept of word-combs. In this section we will lift his techniques to the tree-case, i.e., we prove the decidability of Ramsey quantifiers on (tree-)automatic structures. Actually, our proof can also be seen as an adaption of To's and Libkin's proof [60] of the decidability of the recurrent reachability problem on automatic structures. Nevertheless, our result was developed independently from To's and Libkin's work.

Let us briefly recall Rubin's ideas. His main tool is the concept of a word-comb. A word-comb is an infinite sequence of finite Σ-words such that there is a sequence of natural numbers $g_1 < g_2 < g_3 < \dots$ such that all but the shortest n words of the word-comb agree on the first g_n letters. A word-comb can be represented using infinite words as follows. Let $w_1 \in \Sigma^\omega$, $w_2 \in (\Sigma \cup \{\Box\})^\omega$ be infinite words and $G \subseteq \mathbb{N}$ an infinite set. A finite word w belongs to the word-comb represented by (w_1, w_2, G) if the following holds: w decomposes as $w = v_1 \circ v_2$ where v_1 is a prefix of w_1 and v_2 is a subword of w_2 such that

1. $|v_1| \in G$,
2. there is some $k \in \mathbb{N}$ such that $v_2 \Box^k$ is the subword of w_2 induced by the $(|v_1|+1)$-st to the $(|w|+k)$-th letter of w_2 such that $|w|+k$ is the successor of $|v_1|$ in G.

Figure 3.20 illustrates such a representation of a word-comb.

Now, we explain how the notion of a word-comb can be used to decide Ramsey quantifiers on string-automatic structures. The first important observation is that every infinite set of finite words contains a subset which is a word-comb, i.e., a subset that can be represented by some triple (w_1, w_2, G) as explained above. Secondly, ω-string-automata can be used to extract the words of the word-comb from the representation.

Recall that the Ramsey quantifier asserts the existence of an infinite subset that is homogeneous with respect to a certain formula φ, i.e., all pairwise distinct n-tuples from this set satisfy φ. Now, for each string-automatic structure $\mathfrak{A}$, this can be translated into the assertion that there is a representation of a word-comb such that each pairwise distinct n-tuple from the comb satisfies φ. This assertion can be formulated in a first-order formula φ' on a certain ω-string-automatic extension $\mathfrak{A}'$ of $\mathfrak{A}$. This extension $\mathfrak{A}'$ enriches $\mathfrak{A}$ by

those infinite strings that occur in the representation of word-combs. The classical correspondence between first-order logic on ω-string-automatic structures and ω-string-automata yields an ω-string-automaton that represents φ' on $\mathfrak{A}'$. Finally, this ω-string-automaton can be turned into a string-automaton that represents φ on $\mathfrak{A}$.

This idea carries even further. Kuske [45] introduced a logic which he calls FSO. The Formulas of FSO are formed according to the formation rules of first-order logic and the following two rules. First, one may use variables for n-ary relations, i.e., for X an n-ary relation variable and $x_1, x_2, \ldots, x_n$ element variables, $Xx_1x_2\ldots x_n$ is an atomic formula of FSO. Second, for X a relation variable that only occurs negatively in some $\varphi \in \mathsf{FSO}$, $\exists X$ inf. φ is in FSO. This formula is satisfied if there is an infinite interpretation for X that satisfies φ. FSO is a generalisation of $\mathsf{FO}((\mathsf{Ram}^n)n \in \mathbb{N})$ as follows. $\mathsf{Ram}^n\bar{x}\varphi$ is equivalent to $\exists X(\forall x_1, \ldots x_n(\bigwedge_{1\leq i\leq n} x_i \in X) \rightarrow \varphi)$. On string-automatic structures, Rubin's technique generalises to FSO: analogously to the decidability of the Ramsey quantifier, one obtains the decidability of FSO on string-automatic structures. The reason why this result extends to FSO is a closure under subsets of witnesses for FSO formulas: if an FSO formula φ asserts the existence of some infinite set X then X appears only negatively in the subformulas of φ. Without loss of generality this means that the only occurrences of X in subformulas of φ are of the form $x \notin X$. If A is an infinite set witnessing the assertion of φ, then any infinite subset $A' \subseteq A$ also witnesses the statement $x \notin A'$ if A witnesses $x \notin A$. Thus, taking an infinite subset of some witness of a formula in FSO is again a witness of this formula. Since every infinite subset of a set of words contains a word-comb, it suffices to look for witnesses of FSO formulas among the word-combs. Hence, Rubin's reduction works also for FSO.

Our goal is to lift the concept of a comb from strings to finite trees. We use three infinite trees for representing an infinite set of finite trees. Unfortunately, the correspondence we obtain is not as tight as in the string case: each infinite set of finite words contains a word-comb that is represented by some triple (w_1, w_2, G). Furthermore, there is an ω-string-automaton that decides, on input some finite word w and the representation (w_1, w_2, G) whether w is contained in the word-comb. The notion of word-combs smoothly generalises to the notion of tree-combs. Unfortunately, tree-combs do not form ω-tree-regular sets. This makes the tree case more involved.

The outline of our proof is as follows. Given an infinite set of finite trees, there is an infinite subset called a tree-comb. A tree-comb is an infinite set that allows a unique representation as a triple (T_1, T_2, G) where T_1 and T_2 are infinite trees and $G \subseteq \{0,1\}^*$. We then define an ω-automaton that extracts finite trees from the representation of a tree-comb. The set of all these trees is called the closure of the tree-comb. The connection between a tree-comb and its closure is as follows. Firstly, every tree-comb is contained in its closure. Secondly, each tree T contained in the closure is locally equal to the trees in the tree-comb: given an arbitrary infinite branch, there is a tree t' in the tree-comb such that t and t' coincide along this infinite branch.

In order to decide Ramsey quantifiers on automatic structures, we first prove that each Ramsey quantifier is witnessed by the closure of some tree-comb. In order to explain the single steps of this proof, we fix a formula $\varphi \in \mathsf{FO}$ and consider the formula

$$\mathsf{Ram}^n\bar{x}(\varphi).$$

We fix an automatic structure $\mathfrak{A}$. On $\mathfrak{A}$, φ corresponds to some automaton $\mathscr{A}_\varphi$. We prove the following.

Figure 3.20.: Word-comb (w_1, w_2, G) encoding the set $\{a^{2n}b : n \in \mathbb{N}\}$.

1. A straightforward generalisation of the string case shows that $\mathfrak{A} \models \mathsf{Ram}^n \bar{x}(\varphi)$ if and only if there is a tree-comb C witnessing this Ramsey quantifier on $\mathfrak{A}$.

2. We show that C can be chosen to be homogeneous with respect to $\mathscr{A}_\varphi$. Roughly speaking, homogeneity means that the runs of $\mathscr{A}_\varphi$ on all pairwise distinct n-tuples from C look similar.

3. For a homogeneous C, we show that $\mathscr{A}_\varphi$ accepts all n-tuples from the closure of C. The proof idea of this step is as follows. Since each n-tuple $\bar{c}$ from the closure is locally equal to n-tuples from C, we can locally copy the accepting runs of $\mathscr{A}_\varphi$ on the latter tuples and obtain a function defined on the domain of $\bar{c}$. Since all runs that we locally copy are similar, this function turns out to be a run of $\mathscr{A}_\varphi$ on $\bar{c}$. Since it is composed from accepting runs, it is also accepting.

Putting these steps together, we obtain that $\mathfrak{A}$ satisfies some Ramsey quantifier if and only if there is a closure of some tree-comb witnessing this quantifier. The proof of the decidability of Ramsey quantifiers on $\mathfrak{A}$ continues analogously to the string case. We obtain an ω-automatic extension of $\mathfrak{A}$. On this extension, the existence of the closure of a tree-comb that witnesses the Ramsey quantifier is expressible in first-order logic. The resulting first-order formula is turned into an ω-automaton using standard techniques. This ω-automaton can then be turned into an automaton corresponding to $\mathsf{Ram}^n \bar{x}(\varphi)$ on $\mathfrak{A}$. Thus, for any formula in $\mathsf{FO}(\exists^{\mathrm{mod}}, (\mathsf{Ram}^n)_{n\in\mathbb{N}})$ and any automatic structure $\mathfrak{A}$, there is an automaton $\mathscr{A}_\varphi$ that corresponds to φ on $\mathfrak{A}$. Unfortunately, this approach does not extend directly to Kuske's logic FSO. Thus, it remains an open problem whether FSO is decidable on all tree-automatic structures.

Remark 3.4.1. As already indicated, we deal with finite and infinite trees in this section. Because of this, we deviate from our notational conventions in the following way. Throughout Section 3.4, where we have to distinguish between infinite and finite trees, we write "tree" for an object that is either a finite or an infinite tree, i.e., a Σ-tree is an element of $\mathsf{Tree}_\Sigma^{\leq\omega} = \mathsf{Tree}_\Sigma \cup \mathsf{Tree}_\Sigma^\omega$. Thus, whenever we want to consider an element of Tree_Σ, we will explicitly write finite tree.

3.4.1 Tree-Combs

Recall that Ramsey quantifiers allow a restricted form of second-order quantification. In order to translate these quantifiers over an automatic structure into first-order quantifiers over an ω-automatic structure, we want to represent infinite sets of finite trees by a tuple of infinite trees.

In Definition 3.4.11, we formally introduce tree-combs. Before, we develop some machinery that allows to extract finite trees from a tuple of infinite trees. This machinery is not necessary for understanding the definition of tree-combs, but it is used to define the closure of a tree-comb. Since our interest is in the relationship of tree-combs and their closures, we postpone the definition of tree-combs.

In the following, we write $\Sigma_\square$ for $\Sigma \cup \{\square\}$ where $\square \notin \Sigma$ is some new symbol.

Recall that we defined the following notation. If t is a Σ-labelled tree, then we denote by $t^\square$ the full binary tree which consists of t padded by $\square$-labels. We define a kind of inverse to

this operation which returns the maximal Σ-labelled tree contained in a given $\Sigma_\Box$-labelled tree.

Definition 3.4.2. Let $T \in \mathrm{Tree}_{\Sigma_\Box}^{\leq\omega}$ be an arbitrary tree. Then $\mathsf{prune}(T)$ denotes the maximal initial segment of T that is in $\mathrm{Tree}_{\Sigma}^{\leq\omega}$.

Remark 3.4.3. We stress that prune yields a Σ-labelled tree from a $\Sigma_\Box$-labelled tree. This is done by extracting the initial segment up to the first occurrence of $\Box$ along each branch. In this sense, $\Box$-labelled positions in T mark undefined positions in the domain of $\mathsf{prune}(T)$.

Recall that we extract an element of a word-comb from its representation (w_1, w_2, G) by taking the prefix v_1 of w_1 of length g_1 for some $g_1 \in G$ and appending a subword v_2 of w_2. v_2 consists of the (g_1+1)-st to the g_2-th letter of w_2 where g_2 is the direct successor of g_1 in G. The function prune will be used to extract an analogue of v_1 in the tree-case. Now, we define another function, called $\mathsf{extract}$, that is the analogue to the extraction of v_2. It extracts the Σ-labelled subtree of an infinite tree from a given position up to the first occurrence of an element from G along each branch.

In the string case, we obtain an element encoded in (w_1, w_2, G) by composition of v_1 and v_2. Analogously, after defining $\mathsf{extract}$ we need a kind of composition of prune and $\mathsf{extract}$ which extracts a tree from a triple (T_1, T_2, G). This composition is a function called extree.

Definition 3.4.4. Let $T : \{0,1\}^* \to \Sigma_\Box$, $G \subseteq \{0,1\}^*$ and $e \in \{0,1\}^*$. Then $\mathsf{extract}(e, T, G)$ is the maximal initial segment of $(T)_e$ (the subtree of T rooted at e) such that the following two conditions are satisfied.

- $\mathsf{extract}(e, T, G)$ is a Σ-labelled tree, i.e., it does not contain $\Box$-labelled nodes.
- For all $d \in \mathsf{dom}(\mathsf{extract}(e, T, G))$, $ed \in G$ implies $d = \varepsilon$.

Remark 3.4.5. Note that $\Box$-labelled nodes in T mark again positions that are undefined in the domain of $\mathsf{extract}(e, T, G)$.

Note that $\mathsf{extract}(T, G, e)$ is the empty tree if and only if $T(e) = \Box$. Furthermore, it is a finite tree if every branch starting at e contains a node $e \leq e'$ with $T(e') = \Box$ or $e' \in G$. If it is a finite tree, then it is a Σ-labelled finite tree by the very definition.

Next, we define extree. In general, extree may extract infinite trees from a triple (T_1, T_2, G). But later we use it only on inputs where it extracts finite trees.

Recall that we write H^+ for the border of a tree-domain H, i.e., H^+ is the set of minimal elements of $\{0,1\}^* \setminus H$.

Definition 3.4.6. Let $T_1, T_2 : \{0,1\}^* \to \Sigma_\Box$ be trees and $G \subseteq \{0,1\}^*$ some set. Assume that $H \subseteq \{0,1\}^*$ is a finite tree-domain. Set

$$P := \mathsf{dom}(\mathsf{prune}(T_1 \restriction_H)) \text{ and}$$
$$D := P \cup \bigcup_{e \in H^+ \cap P^+} \mathsf{dom}(\mathsf{extract}(e, T_2, G)).$$

Let $t := \mathsf{extree}(H, T_1, T_2, G)$ denote the tree with domain D that is defined by

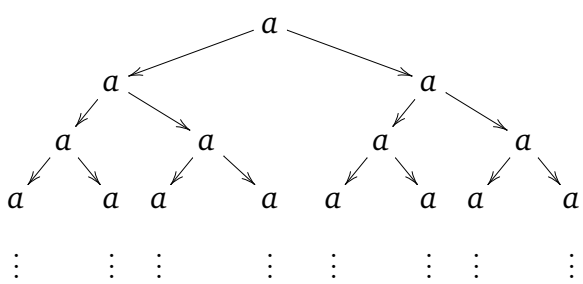
$$t(q) := \begin{cases} T_1(q) & q \in \mathsf{dom}(\mathsf{prune}(T_1 \restriction_H)), \\ T_2(q) & \text{otherwise.} \end{cases}$$

Remark 3.4.7. $\mathsf{extree}(H, T_1, T_2, G)$ extracts a tree from (T_1, T_2, G) that coincides with T_1 on domain H (where positions that are labelled by $\Box$ in T_1 count as undefined positions). For each of those branches that are defined up to the border of H, we append the corresponding subtree of T_2. That is, for $d \in H^+$ such that T_1 is defined on all ancestors of d, we append $\mathsf{extract}(d, T_2, G)$.

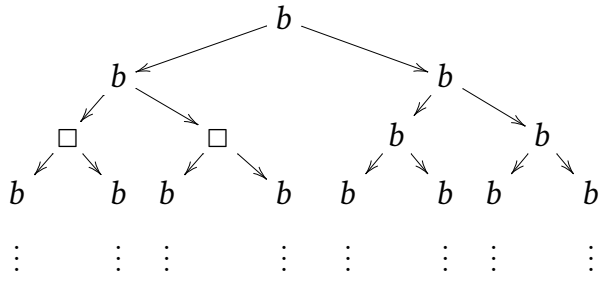
Let us illustrate these definitions in an example.

Example 3.4.8. Consider the following infinite trees T_1 and T_2:

T_1 :



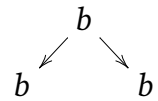
T_2 :



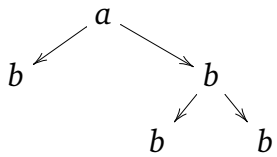
Consider $G := \{w \in \{0,1\}^* : |w| \text{ is odd}\}$ and $H := \{\varepsilon\}$. Then $H^+ = \{0,1\} \subseteq G$ and we obtain the following trees using $\mathsf{extract}$ on H^+:

$\mathsf{extract}(0, T_2, G)$:

$\mathsf{extract}(1, T_2, G)$:



Note that $\mathsf{prune}(T_1 \restriction_H) = a$. Hence, we conclude that $\mathsf{extree}(H, T_1, T_2, G)$ is the following tree:



Now, we use the function extree to define the set of infinite trees that is encoded by a triple (T_1, T_2, G).

Definition 3.4.9. Let $G \subseteq \{0,1\}^*$. A finite tree-domain H is called a G-tree if $H^+ \subseteq G$. We set

$$\mathsf{Set}(T_1, T_2, G) := \{\mathsf{extree}(H, T_1, T_2, G) : H \text{ is a } G\text{-tree}\}.$$

Remark 3.4.10. Note that $\mathsf{Set}(T_1, T_2, G)$ may contain infinite trees. Moreover, this set may be finite, e.g., if $G = \emptyset$. In our applications, we always ensure that this definition yields an infinite set of finite trees.

As the next step, we define the notions of a tree-comb and of the closure of a tree-comb. These definitions aim at the following: we look for an infinite sequence $C = (T_i)_{i \geq 1}$ of finite trees that can be encoded by a tuple (T_1^C, T_2^C, G^C) of infinite trees such that $\mathsf{Set}(T_1^C, T_2^C, G^C)$ contains C. More precisely, $T_i = \mathsf{extree}(H, T_1^C, T_2^C, G^C)$ for the G^C-tree H induced by the i-th layer of G in the following sense. Let B_1 be the set of all infinite branches b such that

$|G \cap b| \geq i$, i.e., those branches along which an element of G occurs at least i times. Along every infinite branch $b \in B_1$, $H \cap b$ is the finite branch up to the predecessor of the i-th element of $b \cap G$ (if $b \cap G$ contains at least i elements). Along every infinite branch b in the complement of B_1, $H \cap b$ is maximal in the sense that H contains all predecessors of the maximal element of $b \cap G$.

Before we state the precise definition of a tree-comb, let us explain how this notion generalises the notion of a word-comb. A word-comb is an infinite set of strings such that there is a sequence $g_1 < g_2 < g_3 < \dots$ of natural numbers such that all but the shortest n words of the word-comb agree on the first g_n letters. Furthermore, the length of the words forming the word-comb grows unbounded. We transfer this principle to the tree case as follows: we replace the notion of "length of a string" by the notion of "depth of a tree". Thus, we want a tree-comb to be an infinite sequence of trees of growing depth such that all but the first n trees coincide on a certain initial part D_n of their domain.

Before we state the definition, recall that $\mathrm{dom}(T)^{\oplus}$ denotes the union of $\mathrm{dom}(T)$ with its border $\mathrm{dom}(T)^{+}$.

Definition 3.4.11. An infinite sequence of finite trees $C = (T_i)_{i \geq 1}$ is called a tree-comb if $T_j{\restriction}_{\mathrm{dom}(T_i)^{\oplus}} = T_k{\restriction}_{\mathrm{dom}(T_i)^{\oplus}}$ for all natural numbers $1 \leq i < j < k$.

Remark 3.4.12. A tree-comb $C = (T_i)_{i \geq 1}$ is an infinite sequence of finite trees. Abusing notation, we will identify C with the set $\{T_i : i \geq 1\}$ if no confusion arises. In this sense, we write $D \subseteq C$ for the fact that D is an infinite subsequence of C. In this case, D is also a tree-comb.

We will soon see that any infinite set of trees contains a subset which forms a tree-comb. Before we come to this, let us define the notion of a representation of a tree-comb by a triple of infinite trees.

Definition 3.4.13. Let $C = (T_i)_{i \geq 1}$ be a tree-comb. We define the trees T_1^C, T_2^C, G^C as follows:

$$
\begin{aligned}
&T_1^C : \{0,1\}^* \to \Sigma_{\square} \text{ with} \\
&T_1^C(d) := \begin{cases} T_i(d) & \text{for } d \in \mathrm{dom}(T_i) \cap \bigcup_{j<i} \mathrm{dom}(T_j), \\ \square & \text{otherwise,} \end{cases} \\
&T_2^C : \{0,1\}^* \to \Sigma_{\square} \text{ with} \\
&T_2^C(d) := \begin{cases} T_i(d) & \text{for } d \in \mathrm{dom}(T_i) \setminus \bigcup_{j<i} \mathrm{dom}(T_j), \\ \square & \text{otherwise,} \end{cases} \\
&G^C := \{\varepsilon\} \cup \bigcup_{i \geq 1} \left(\mathrm{dom}(T_i)^{+} \setminus (\mathrm{dom}(T_{i-1})^{\oplus}) \right).
\end{aligned}
$$

We call the triple (T_1^C, T_2^C, G^C) the representation of C.

Remark 3.4.14. Note that T_1^C is well-defined: if there are $i' > i > j$ such that

$$d \in \mathrm{dom}(T_{i'}) \cap \mathrm{dom}(T_i) \cap \mathrm{dom}(T_j),$$

then $T_{i'}(d) = T_i(d)$ by the tree-comb property.

Furthermore, if there is some node d such that $T_1^C(d) = \square$ then $T_1^C(de) = \square$ for all $e \in \{0,1\}^*$. This is due to the fact that $T_1^C(d) = \square$ if d is in the domain of at most one of the T_i. But then all descendents of e satisfy this condition, too.

Note that $G^C = \{\varepsilon\} \cup \bigcup_{i\geq 1} \left(\mathsf{dom}(T_i)^+ \setminus \bigcup_{j<i} \mathsf{dom}(T_j)^\oplus\right)$: ($\supseteq$) is trivially true. For ($\subseteq$) assume that $\varepsilon \neq d \in \mathsf{dom}(T_i)^+ \setminus (\mathsf{dom}(T_{i-1})^\oplus)$. Heading for a contradiction, assume that $d \in \mathsf{dom}(T_j)^\oplus$ for some $j < i-1$. By definition of a tree-comb, this implies that T_i and T_{i-1} agree on d which contradicts the assumption $d \notin \mathsf{dom}(T_{i-1})^\oplus$ and $d \in \mathsf{dom}(T_i)^\oplus$. Thus, $d \notin \mathsf{dom}(T_j)^\oplus$ for all $j < i-1$ whence $d \in \bigcup_{i\geq 1} \left(\mathsf{dom}(T_i)^+ \setminus \bigcup_{j<i} \mathsf{dom}(T_j)^\oplus\right)$.

Definition 3.4.15. Let C be a tree-comb. We call $\mathsf{CL}(C) := \mathsf{Set}(T_1^C, T_2^C, G^C)$ the closure of C.

Remark 3.4.16. Calling $\mathsf{Set}(T_1^C, T_2^C.G^C)$ a closure of C requires some justification: we postpone this justification for a while. But in Lemma 3.4.20, we will see that $\mathsf{CL}(C)$ contains each element of C.

In the following, we study tree-combs, their representations and their closures. First, we show that any infinite set of trees contains a tree-comb. Then we show that every tree-comb is contained in its closure. Furthermore, we show that the closure of every tree-comb is an infinite set of finite trees. Finally, we introduce a partial order on the closure of every tree-comb. This order plays a crucial technical role in our reduction of the Ramsey quantifier. Each Ramsey quantifier that asserts a certain property of all pairwise distinct n-tuples of some infinite set will be reduced to the assertion that all pairwise comparable n-tuples of the closure of some tree-comb have this property.

We apply Ramsey's Theorem in many of the following proofs. Thus, we recall this theorem briefly.

Theorem 3.4.17 ([56]). Let S be an infinite set, C a finite set of colours. We write $P_n(S)$ for the set of n-element subsets of S. For each colouring $f : P_n(S) \to C$ of the n-element subsets of S there is an infinite subset $S' \subseteq S$ such that f is constant on $P_n(S')$.

We are now prepared to prove that every infinite set of finite trees contains a subset that induces a tree-comb.

Lemma 3.4.18. Let S be an infinite set of finite trees. Then there is a tree-comb C such that each element of C is contained in S.

Proof. We define $C = (T_i)_{i\geq 1}$ by induction.

Choose $T_1 \in S$ arbitrarily. Since $\mathsf{dom}(T_1)^\oplus$ is finite and due to Ramsey's Theorem, there is an infinite set $S^1 \subseteq S$ such that for all $T, T' \in S^1$, we have $T\restriction_{\mathsf{dom}(T_1)^\oplus} = T'\restriction_{\mathsf{dom}(T_1)^\oplus}$.

Choose $T_2 \in S^1$ arbitrarily. Again, $\mathsf{dom}(T_2)$ is finite whence there is some infinite $S^2 \subseteq S^1$ such that $T\restriction_{\mathsf{dom}(T_2)^\oplus} = T'\restriction_{\mathsf{dom}(T_2)^\oplus}$ for all $T, T' \in S^2$.

Continuing this construction, we obtain infinitely many finite trees $T_1, T_2, T_3, \ldots$ Because of the definition of these trees, $C := (T_i)_{i\geq 1}$ is a tree-comb. $\square$

Note that by definition of the Ramsey quantifier, the witnesses for Ramsey quantifiers are closed under taking infinite subsets: if S is an infinite set of finite trees witnessing some Ramsey quantifier, then every infinite subset of S also witnesses this Ramsey quantifier. From this point of view, the previous lemma says that the search space for witnesses for Ramsey quantifiers on automatic structures can be restricted to tree-combs.

The next lemma collects some technical facts about the representation (T_1^C, T_2^C, G^C) of a tree-comb C.

Lemma 3.4.19. Let $C = (T_j)_{j\geq 1}$ be a tree-comb and let $i \geq 1$ be some natural number.

1. For all $d \in \bigcup_{j<i} \mathrm{dom}(T_j)$ we have $T_1^C(d) \neq \Box$ iff $d \in \mathrm{dom}(T_i)$.
2. For all $D \subseteq \bigcup_{j<i} \mathrm{dom}(T_j)$ and for $E := \mathrm{dom}(\mathsf{prune}(T_1^C{\restriction_D}))$, we have
$$\mathsf{prune}(T_1^C{\restriction_D}) = T_i{\restriction_E}.$$
3. For all $d \in \mathrm{dom}(T_{i-1})^+ \setminus \bigcup_{j<i-1} \mathrm{dom}(T_j)^{\oplus}$, we have $\mathsf{extract}(d, T_2^C, G^C) = (T_i)_d$.

Proof. 1. Let $d \in \bigcup_{j<i} \mathrm{dom}(T_j)$. By definition, $T_1^C(d) \neq \Box$ if and only if there is some $k \in \mathbb{N}$ such that $d \in \mathrm{dom}(T_k) \cap \bigcup_{j<k} \mathrm{dom}(T_j)$. By assumption on d, this is the case if and only if there is some k such that $d \in \mathrm{dom}(T_k) \cap \bigcup_{j<k} \mathrm{dom}(T_j) \cap \bigcup_{j<i} \mathrm{dom}(T_j)$. We have to show that this is the case if and only if $d \in \mathrm{dom}(T_i) \cap \bigcup_{j<i} \mathrm{dom}(T_j)$.

Assume that $d \in \mathrm{dom}(T_i) \cap \bigcup_{j<j} \mathrm{dom}(T_j)$. Setting $k := i$, we obtain directly that $d \in \mathrm{dom}(T_k) \cap \bigcup_{j<k} \mathrm{dom}(T_j) \cap \bigcup_{j<i} \mathrm{dom}(T_j)$.

For the other direction, assume that there is some $k \in \mathbb{N}$ such that
$$d \in \mathrm{dom}(T_k) \cap \bigcup_{j<k} \mathrm{dom}(T_j) \cap \bigcup_{j<i} \mathrm{dom}(T_j).$$

Due to the definition of a tree-comb, for all $i, k \in \mathbb{N}$ the trees T_i and T_k agree on $\bigcup_{j<\min(k,i)} \mathrm{dom}(T_j)^{\oplus}$. It follows immediately that
$$\mathrm{dom}(T_k) \cap \bigcup_{j<\min(k,i)} \mathrm{dom}(T_j) = \mathrm{dom}(T_i) \cap \bigcup_{j<\min(k,i)} \mathrm{dom}(T_j).$$

From this we derive directly that
$$\begin{aligned}
&\mathrm{dom}(T_k) \cap \bigcup_{j<k} \mathrm{dom}(T_j) \cap \bigcup_{j<i} \mathrm{dom}(T_j) \\
=&\left(\mathrm{dom}(T_k) \cap \bigcup_{j<\min(i,k)} \mathrm{dom}(T_j)\right) \cap \bigcup_{j<i} \mathrm{dom}(T_j) \\
=&\left(\mathrm{dom}(T_i) \cap \bigcup_{j<\min(i,k)} \mathrm{dom}(T_j)\right) \cap \bigcup_{j<i} \mathrm{dom}(T_j) \\
\subseteq&\mathrm{dom}(T_i).
\end{aligned}$$

Thus, we conclude that $d \in \mathrm{dom}(T_i) \cap \bigcup_{j<i} \mathrm{dom}(T_i)$.

2. Let $D \subseteq \bigcup_{j<i} \mathrm{dom}(T_j)$. The previous part of this Lemma showed that

$$D \cap \mathrm{dom}(T_i) = D \cap \{d : T_1^c(d) \neq \Box\}.$$

By definition of the function prune, it follows that

$$E := \mathrm{dom}(\mathsf{prune}(T_1^C{\restriction}_D)) = \mathrm{dom}(T_i) \cap D.$$

Together with the definition of T_C^1, this implies that $T_C^1(d) = T_i(d)$ for all $d \in E$. Thus, $\mathsf{prune}(T_1^C{\restriction}_D) = T_i{\restriction}_E$.

3. Let $d \in \mathrm{dom}(T_{i-1})^+ \setminus \bigcup_{j<i-1}(\mathrm{dom}(T_j)^\oplus)$. We have to show that

$$\mathsf{extract}(d, T_2^C, G^C) = (T_i)_d.$$

There are the following cases.

a) $d \notin \mathrm{dom}(T_k)$ for all $k \in \mathbb{N}$: by definition of T_2^C this implies $T_2^C(d) = \Box$ whence $\mathsf{extract}(d, T_2^C, G^C) = (T_i)_d = \emptyset$.

b) Otherwise, there is some $k \in \mathbb{N}$ such that $d \in \mathrm{dom}(T_k)$: in this case, $k \geq i$ because $d \notin \bigcup_{j<i-1} \mathrm{dom}(T_j)$. But then T_k and T_i agree on d because $d \in \mathrm{dom}(T_{i-1})^\oplus$ and due to the definition of a tree-comb. Hence, $d \in \mathrm{dom}(T_i)$. Furthermore, $T_2^C(e) = T_i(e)$ for all $d \leq e \in \mathrm{dom}(T_i)$ due to the definition of T_2^C. Moreover, $e \notin \mathrm{dom}(T_j)^+$ for all $d < e \in \mathrm{dom}(T_i)$ and all $j \leq i$. Remark 3.4.14 then implies that $e \notin G^C$ for all $d < e \in \mathrm{dom}(T_i)$. Finally, due to $d \in \mathrm{dom}(T_i) \setminus \mathrm{dom}(T_{i-1})$,

$$\mathrm{dom}(T_i)^+ \cap \{e : d \leq e\} \subseteq \mathrm{dom}(T_i)^+ \setminus (\mathrm{dom}(T_{i-1})^\oplus) \subseteq G^C$$

Thus, we conclude that $\mathsf{extract}(d, T_2^C, G^C) = (T_i)_d$. □

In the next lemma we show that the representation (T_1^C, T_2^C, G^C) of a tree-comb C is a correct representation in the following sense: all elements of the tree-comb can be extracted from this representation, i.e., for each tree $T \in C$, it holds that $T \in \mathrm{CL}(C)$.

Lemma 3.4.20. Let $C = (T_i)_{i\geq 1}$ be a tree-comb. For each $i \geq 1$, $T_i \in \mathrm{CL}(C)$, i.e., $T_i \in \mathrm{Set}(T_1^C, T_2^C, G^C)$.

Proof. For each $i \geq 1$, we construct a G^C-tree H such that $T_i = \mathsf{extree}(H, T_1^C, T_2^C, G^C)$. Set $H := \bigcup_{j<i} \mathrm{dom}(T_j)$. First, we show that H is a G^C-tree, then we show that $T_i = \mathsf{extree}(H, T_1^C, T_2^C, G^C)$.

1. We have to show that $H^+ \subseteq G$. Let x^- be the predecessor of some $x \in H^+$ and let j be minimal such that $x^- \in \mathrm{dom}(T_j)$. By definition $x \in \mathrm{dom}(T_j)^+ \setminus (\mathrm{dom}(T_{j-1})^\oplus)$, whence $x \in G^C$.

2. Let us first consider the restriction of this tree to H. Set

$$P := \mathrm{dom}(\mathsf{prune}(T_1^C{\restriction}_H)) = \mathrm{dom}(\mathsf{prune}(T_1^C{\restriction}_{\bigcup_{j<i} \mathrm{dom}(T_j)})).$$

By Lemma 3.4.19 T_1^C agrees with T_i on P.

Due to the definition of a G^C-tree, for each $d \in H^+$ there is some $k < i$ with $d \in \mathrm{dom}(T_k)^+ \setminus \bigcup_{j<k} \mathrm{dom}(T_j)^\oplus$.

If $k = i-1$, then the third item of Lemma 3.4.19 implies $\mathsf{extract}(d, T_2^C, G^C) = (T_i)_d$.

Otherwise, $k < i-1$. By the definition of a tree-comb, we know that $d \notin \mathrm{dom}(T_{k+1})$ iff $d \notin \mathrm{dom}(T_j)$ for all $j > k$.

By $d \in H^+$ we know that $d \notin \mathrm{dom}(T_j)$ for $j < i$. Since $k+1 < i$, $d \notin \mathrm{dom}(T_{k+1})$ whence $d \notin \mathrm{dom}(T_j)$ for all $j > k$.

Due to $k < i$, we conclude that $d \notin \mathrm{dom}(T_j)$ for all $j \geq 1$. Thus, $T_2^C(d) = \Box$ whence $\mathsf{extract}(d, T_2^C, G^C) = \emptyset = (T_i)_d$. □

The proof of the previous lemma implies the following corollary.

Corollary 3.4.21. Let C be a tree-comb and (T_1^C, T_2^C, G^C) be its representation. For each $g \in G^C$, there is some G^C-tree H such that $g \in H^+$.

Proof. By definition of G^C, there is some $i \in \mathbb{N}$ such that $g \in \mathrm{dom}(T_i)^+ \setminus (\mathrm{dom}(T_{i-1})^\oplus)$. By Remark 3.4.14, we know that $g \in \mathrm{dom}(T_i)^+ \setminus \bigcup_{j<i} \mathrm{dom}(T_j)^\oplus$. In the proof of Lemma 3.4.20 we have already seen that $H := \bigcup_{j \leq i} \mathrm{dom}(T_j)$ forms a G-tree. The claim follows from $g \in H^+$. □

The next lemma shows that for the representation (T_1^C, T_2^C, G^C) of an arbitrary tree-comb C, the set $\mathsf{Set}(T_1^C, T_2^C, G^C)$ is an infinite set of finite trees. Since we aim at representing infinite sets of finite trees, we will call any triple (T_1, T_2, G) coherent if it induces an infinite set of finite trees via the operator Set.

Lemma 3.4.22. Let $C = (T_i)_{i \geq 1}$ be an arbitrary tree-comb. Its closure $\mathsf{CL}(C)$ is coherent, i.e., $\mathsf{Set}(T_1^C, T_2^C, G^C)$ is an infinite set of finite trees.

Proof. By Lemma 3.4.20, we have already seen that all trees from C are contained in $\mathsf{CL}(C)$. Hence, $\mathsf{CL}(C)$ contains an infinite set of finite trees. Thus, it is only left to show that each G^C-tree H induces a finite tree.

Since a G^C-tree is a finite tree-domain by definition, it suffices to show the finiteness of $\mathsf{extract}(d, T_2^C, G^C)$ for all $d \in G^C$.

For this purpose, let $d \in G^C$. Then there is some $i \in \mathbb{N}$ such that $d \in T_i^+ \setminus \bigcup_{j<i} \mathrm{dom}(T_j)^\oplus$. Due to the last item of Lemma 3.4.19, $\mathsf{extract}(d, T_2^C, G^C) = (T_{i+1})_d$. Since T_{i+1} is a finite tree, its subtree rooted at d is also finite. □

In order to reduce Ramsey quantifiers on automatic structures to first-order logic on ω-automatic structures, we need to introduce one further concept concerning tree-combs: for (T_1^C, T_2^C, G^C) a representation of some tree-comb C, we define a partial order $<_{G^C}$ on $\mathsf{CL}(C)$. The purpose of this order is the following: the Ramsey quantifier asserts that there is an infinite set such that its pairwise distinct n-tuples satisfy a certain formula. This assertion will be reduced to the assertion that there is a closure of some tree-comb such that all pairwise $<_{G^C}$ comparable n-tuples satisfy the formula. We are going to define $<_{G^C}$ in such a way that the tree-comb C is ordered linearly. Thus, if there is a tree-comb C such that its closure $\mathsf{CL}(C)$ witnesses the reduced assertion, then C witnesses the original assertion: with

respect to C, the notions of "pairwise distinct" and "pairwise comparable" coincide whence C witnesses the Ramsey quantifier.

The order $<_{G^C}$ is defined on trees from $\mathsf{CL}(C)$ by comparing the underlying G^C-trees with respect to $\subsetneq$. We call a G^C-tree H the underlying tree for $T \in \mathsf{CL}(C)$, if $T = \mathsf{extree}(H, T_1^C, T_2^C, G^C)$ and H is maximal with this property. Unfortunately, for an arbitrary representation (T_1, T_2, G) this notion is not well-defined. For an extremely degenerated example, take T_1 to be the constant $\square$-labelled tree and $G = \{0,1\}^*$. Any finite tree domain H forms a G-tree and $\mathsf{extree}(H, T_1, T_2, G)$ is the empty tree for all H. Thus, there is no maximal G-tree underlying the empty tree in this representation. In order to obtain a well-defined notion of underlying G-tree, we first define the notion of a small representation. Afterwards, we show that there is an underlying G-tree for every tree T contained in a small representation. Furthermore, we prove that the representation of every tree-comb is small. Finally, we formally define the order $<_G$ for each small representation (T_1, T_2, G).

Definition 3.4.23. We call a representation (T_1, T_2, G) small if the following two conditions hold.

1. For all $g \in G$ there is some G-tree H such that $g \in H^+$.
2. If there are $d < e \in \{0,1\}^*$ with $d, e \in G$, then for all $c < d$ we have $T_1(c) \neq \square$.

Remark 3.4.24. It does not depend on T_2 whether (T_1, T_2, G) is small. Thus, we will also say (T_1, G) is small meaning that (T_1, T_2, G) is small.

Note that the representation of every tree-comb satisfies the first condition due to Corollary 3.4.21.

Lemma 3.4.25. Let T_1 and T_2 be $\Sigma_\square$-labelled infinite binary trees and $G \subseteq \{0,1\}^*$. Assume that (T_1, T_2, G) is small. For each $T \in \mathsf{Set}(T_1, T_2, G)$ there is a unique maximal G-tree H_T such that $T = \mathsf{extree}(H_T, T_1, T_2, G)$.

Proof. Fix a $T \in \mathsf{Set}(T_1, T_2, G)$. Let

$$S_T := \{H \subseteq \{0,1\}^* : H \text{ a } G\text{-tree and } T = \mathsf{extree}(H, T_1, T_2, G)\}.$$

Furthermore, let $H_T := \bigcup S_T$ be the union of all these G-trees.

First, we show that H_T is a finite tree-domain. By definition of H_T this implies that H_T is a G-tree. Afterwards, we show that it generates T.

H_T is infinite if and only if there is an infinite chain $d_0 < d_1 < d_2 < \cdots \in \{0,1\}^*$ such that for each $i \in \mathbb{N}$ there is some $H_i \in S_T$ with $d_i \in H_i^+$, i.e., the trees-domains in S_T grow unbounded along some infinite branch $d_0 < d_1 < d_2 < \cdots < b \in \{0,1\}^\omega$.

Heading for a contradiction, assume that such a chain $d_0 < d_1 < d_2 < \ldots$ exists.

Since H_i is a G-tree, each $d_i \in G$. Because of $d_1 \in G$ and $d_1 \notin \mathsf{dom}(H_0)^\oplus$, the definition of $\mathsf{extree}(H_0, T_1, T_2, G)$ implies that $d_1 \notin \mathsf{dom}(\mathsf{extree}(H_0, T_1, T_2, G))$. Due to $\mathsf{dom}(T) = \mathsf{extree}(H_0, T_1, T_2, G))$, we conclude that $d_1 \notin \mathsf{dom}(T)$.

On the other hand, (T_1, T_2, G) is small whence $d_1 < d_2 < d_3 \in G$ implies that $T_1(c) \neq \square$ for all $c \leq d_1$. Since $d_1 \in H_2$, $\mathsf{extree}(H_2, T_1, T_2, G)$ and T_1 coincide up to d_1. We conclude that $d_1 \in \mathsf{dom}(T) = \mathsf{dom}(\mathsf{extree}(H_2, T_1, T_2, G)$.

This contradicts $d_1 \notin \mathsf{dom}(T)$. Thus, the tree-domains in S_T cannot grow unbounded along any infinite branch and we conclude that H_T is a well-defined finite tree-domain.

We come to our second claim: $H_T \in S_T$, or equivalently $T = \mathsf{extree}(H_T, T_1, T_2, G)$. In order to prove this claim, let $b \in \{0,1,\}^\omega$ be an arbitrary infinite branch. There is a unique element d that is on the border of H_T in the branch b, i.e., there is a unique element $d \in b \cap H_T^+$. Let d^- be the direct predecessor of d.

By definition of H_T, $d^- \in H$ for some $H \in S_T$ and there is no $H' \in S_T$ with $d \in H'$. Thus, H_T and H agree along the branch b whence $T = \mathsf{extree}(H, T_1, T_2, G)$ and $\mathsf{extree}(H_T, T_1, T_2, G)$ coincide along b.

For each infinite branch there is such a $H \in S_T$ whence we conclude that T and $\mathsf{extree}(H_T, T_1, T_2, G)$ coincide along each infinite branch. Hence, $T = \mathsf{extree}(H_T, T_1, T_2, G)$. □

Lemma 3.4.26. Let $C = (T_i)_{i \geq 1}$ be some tree-comb and (T_1^C, T_2^C, G^C) its representation. Then (T_1^C, T_2^C, G^C) is small.

Proof. We have to show the following two claims:

1. For all $g \in G^C$ there is some G^C-tree H such that $g \in H^+$.
2. If there are $d < e \in \{0,1\}^*$ with $d, e \in G^C$, then for all $c < d$ we have $T_1^C(c) \neq \Box$.

The first claim holds due to Corollary 3.4.21. The second part is an easy consequence of the definition of T_1^C: by definition of G^C, $d, e \in G^C$ with $d \leq e$ implies that there are numbers i and j such that $i \neq j$, $d \in \mathsf{dom}(T_i)^+$ and $e \in \mathsf{dom}(T_j)^+$. But this implies that for all $c < d$, $c \in \mathsf{dom}(T_i) \cap \mathsf{dom}(T_j)$ whence by definition of T_1^C, $T_1^C(c) = T_k(c)$ for $k = \mathsf{max}(i,j)$. □

We conclude the section on tree-combs by defining the order $<_G$ for all small representations (T_1, T_2, G) and by showing that each tree-comb C is linearly ordered by the induced order $<_{G^C}$.

Definition 3.4.27. Let $G \subseteq \{0,1\}^*$. Furthermore, let H and H' be G-trees. We define $H <_G H'$ if the following two conditions hold:

1. $H \subsetneq H'$ and
2. for each infinite branch $b \in \{0,1\}^\omega$, $H \cap b = H' \cap b$ implies $(b \setminus (H^\oplus)) \cap G = \emptyset$.

This means that $H <_G H'$ holds if H' extends H properly along each branch where this is possible for a G-tree. In other words, if there is a descendent of some $d \in H^+$ which is in G, then H' must contain d. Thus, H' extends H properly along this branch.

We extend this order to $S, T \in \mathsf{Set}(T_1, T_2, G)$ for small representations (T_1, T_2, G) as follows.

Definition 3.4.28. Let (T_1, T_2, G) be a small representation. Let H_S (H_T) denote the maximal G-tree such that $S = \mathsf{extree}(H_S, T_1, T_2, G)$ ($T = \mathsf{extree}(H_T, T_1, T_2, G)$, respectively), i.e., H_S and H_T are the underlying trees for S and T, respectively. We set

$$S <_G T \text{ iff } H_S <_G H_T.$$

This order formalises the idea that the underlying G-tree H' extends H in each possible direction. Since a G-tree ends along each path just in front of a node from G, the branches where a G-tree cannot be extended are those where no further elements from G follow after $H^\oplus$.

We conclude this section by showing that any tree-comb C is linearly ordered by the induced order $<_{G^C}$.

Lemma 3.4.29. Let $C = (T_i)_{i\geq 1}$ be a tree-comb. Then $T_i <_{G^C} T_k$ for all $1 \leq i \leq k$.

Proof. Let $H_i := \bigcup_{j<i} \mathsf{dom}(T_j)$ and let $\hat{H}_i$ be the maximal G^C-tree generating T_i. From the proof of lemma 3.4.20 we know that $H_i \subseteq \hat{H}_i$ because H_i also generates T_i.

By definition of $T_i <_{G^C} T_{i+1}$, it suffices to show that $\hat{H}_i <_{G^C} \hat{H}_{i+1}$. We prove this claim in two steps. First we show that $\hat{H}_i \subseteq H_{i+1}$. This implies $\hat{H}_i \subseteq \hat{H}_{i+1}$ and furthermore, these two trees cannot coincide because they generate two different trees, namely, T_i and T_{i+1}. Afterwards, we show that for each infinite branch b the following holds. If $\hat{H}_i \cap b = \hat{H}_{i+1} \cap b$ then $(b \setminus (\hat{H}_i^{\oplus})) \cap G^C = \emptyset$.

1. Since $\hat{H}_i$ and H_{i+1} are tree-domains, $\hat{H}_i \not\subseteq H_{i+1}$ would imply that $\hat{H}_i \cap (H_{i+1}^+) \neq \emptyset$. Heading for a contradiction, assume that there is some $d \in \hat{H}_i \cap (H_{i+1}^+)$.

   By definition of H_{i+1}, $d \in \left(\bigcup_{j\leq i} \mathsf{dom}(T_i)\right)^+$. Let $D := \mathsf{dom}(\mathsf{prune}(T_1^C{\restriction}_{\hat{H}_i}))$. By definition of $\hat{H}_i$, $\mathsf{prune}(T_1^C{\restriction}_{\hat{H}_i}) = T_i{\restriction}_D$. Since $d \notin \mathsf{dom}(T_i)$, this implies $T_1^C(d) = \Box$. Thus, by definition of T_1^C it is not possible that there are two numbers $j_1 \neq j_2 \in \mathbb{N}$ such that $d \in \mathsf{dom}(T_{j_1}) \cap \mathsf{dom}(T_{j_2})$.

   We claim that then $d \notin \mathsf{dom}(T_j)$ for all $j \geq 1$.

   For $j \leq i$, $d \notin \mathsf{dom}(T_j)$ due to $d \in \left(\bigcup_{j\leq i} \mathsf{dom}(T_i)\right)^+$.

   Nevertheless, for the same reason, $d \in \mathsf{dom}(T_k)^{\oplus}$ for some $k \leq i$. Due to the tree-comb property, this implies that T_{j_1} and T_{j_2} agree at d for all $j_1 > j_2 > k$. Since we have already seen that there cannot be two different trees T_{j_1} and T_{j_2} defined at d, we conclude that there is no $j > k$ such that $d \in \mathsf{dom}(T_j)$.

   Thus, we conclude that $d \notin \mathsf{dom}(T_j)$ for all $j \geq 1$. This implies that all $d < e$ satisfy $e \notin \mathsf{dom}(T_j)^+$ for all $j \geq 1$. Due to the definition of G^C, it follows that $e \notin G^C$ for all $d < e$. Thus, d cannot be contained in any G^C-tree.

   But this contradicts the assumption that $d \in \hat{H}_i$.

   We conclude that $\hat{H}_i \cap (H_{i+1}^+) = \emptyset$ which implies $\hat{H}_i \subseteq H_{i+1} \subseteq \hat{H}_{i+1}$.

2. Fix some infinite branch b such that $\hat{H}_i \cap b = \hat{H}_{i+1} \cap b$. Due to $\hat{H}_i \subseteq H_{i+1} \subseteq \hat{H}_{i+1}$ this implies

$$\hat{H}_i \cap b = \hat{H}_{i+1} \cap b = H_{i+1} \cap b = \bigcup_{j\leq i} \mathsf{dom}(T_j) \cap b. \tag{3.6}$$

   As a direct consequence of the coincidence of $\hat{H}_i$ and $\hat{H}_{i+1}$ along b, we obtain that

$$T_i{\restriction}_b = \mathsf{extree}(\hat{H}_i, T_1^C, T_2^C, G^C){\restriction}_b = \mathsf{extree}(\hat{H}_{i+1}, T_1^C, T_2^C, G^C){\restriction}_b = T_{i+1}{\restriction}_b.$$

   Thus,

$$b \cap \bigcup_{j\leq i} \mathsf{dom}(T_j) = b \cap \bigcup_{j\leq i+1} \mathsf{dom}(T_j). \tag{3.7}$$

   Now, let d be the unique element of $b \cap (\hat{H}_{i+1}^+)$.

3.6 implies that there is some $k \leq i$ such that $d \in \mathsf{dom}(T_k)^{\oplus}$ while $d \notin \mathsf{dom}(T_j)$ for all $j \leq i$. Due to 3.7, this implies $d \notin \mathsf{dom}(T_{i+1})$. Since $i+1 > k$, it follows from the tree-comb property that $d \notin \mathsf{dom}(T_j)$ for all $j \geq i+1 > k$.

We conclude that $d \notin \mathsf{dom}(T_j)$ for all $j \geq 1$. But this implies that no proper descendant of d is contained in $\mathsf{dom}(T_j)^+$ for any $j \geq 1$. Hence, no proper descendant of d is contained in G^C. Since $d \in b \cap (\hat{H}^+_{i+1})$, it follows that $(b \setminus (\hat{H}^{\oplus}_{i+1})) \cap G = \emptyset$, which concludes the proof. □

3.4.2 Reduction of the Ramsey Quantifier

We now reduce $\mathsf{FO}((\mathsf{Ram}^n)_{n\in\mathbb{N}})$ on an automatic structure $\mathfrak{A}$ to FO on an ω-automatic structure $\mathsf{Ext}(\mathfrak{A})$.

Adding Tree-Comb Representations to an Automatic Structure

From now up to the end of Section 3.4.5, we fix an automatic structure $\mathfrak{A}$. We assume that, without loss of generality, the identity id is a tree presentation of $\mathfrak{A}$. This means that the universe of $\mathfrak{A}$ is a regular subset $A \subseteq \mathsf{Tree}_\Sigma$ and all relations of $\mathfrak{A}$ are automatic.

We next define a structure $\mathsf{Ext}(\mathfrak{A})$ corresponding to $\mathfrak{A}$ in the following sense. $\mathsf{Ext}(\mathfrak{A})$ is the disjoint union of $\mathfrak{A}$ with a structure that allows to reason about tree-combs in the following sense: each $\mathsf{FO}((\mathsf{Ram}^n)_{n\in\mathbb{N}})$ formula over $\mathfrak{A}$ can be reduced to an FO formula over $\mathsf{Ext}(\mathfrak{A})$. Furthermore, $\mathsf{Ext}(\mathfrak{A})$ turns out to be ω-automatic whence this reduction proves the decidability of $\mathsf{FO}((\mathsf{Ram}^n)_{n\in\mathbb{N}})$ over $\mathfrak{A}$. Later, we use the reduction of an $\mathsf{FO}((\mathsf{Ram}^n)_{n\in\mathbb{N}})$ formula φ on $\mathfrak{A}$ to an FO formula on $\mathsf{Ext}(\mathfrak{A})$ in order to design an ω-automaton that represents φ on $\mathsf{Ext}(\mathfrak{A})$ and that can be turned into an automaton $\mathscr{A}_\varphi$ that corresponds to φ on $\mathfrak{A}$.

Definition 3.4.30. Let $\mathsf{Ext}(\mathfrak{A})$ be the following structure.

- The universe is $A' := \mathsf{Tree}_\Sigma \cup \mathsf{B\text{-}Tree}^\omega_{\Sigma_\square} \cup \mathsf{B\text{-}Tree}^\omega_{\{0,1\}}$ where

$$\mathsf{B\text{-}Tree}^\omega_\Sigma := \{T \in \mathsf{Tree}^\omega_\Sigma : \mathsf{dom}(T) = \{0,1\}^*\}$$

  is the set of all full infinite binary Σ-trees. We identify a subset $G \subseteq \{0,1\}^*$ with its characteristic map in $\mathsf{B\text{-}Tree}^\omega_{\{0,1\}}$.

- The basic relations are those of $\mathfrak{A}$ including the unary relation A which denotes the universe of the structure $\mathfrak{A}$.

- We add the following new relations:
  1. Tree_Σ, $\mathsf{B\text{-}Tree}^\omega_{\Sigma_\square}$, and $\mathsf{B\text{-}Tree}^\omega_{\{0,1\}}$,
  2. $\mathsf{In} := \left\{(T, T_1, T_2, G) \in \mathsf{Tree}_\Sigma \times (\mathsf{B\text{-}Tree}^\omega_{\Sigma_\square})^2 \times \mathsf{B\text{-}Tree}^\omega_{\{0,1\}} : T \in \mathsf{Set}(T_1, T_2, G)\right\}$,
  3. $\mathsf{Coherent} := \left\{(T_1, T_2, G) \in (\mathsf{B\text{-}Tree}^\omega_{\Sigma_\square})^2 \times \mathsf{B\text{-}Tree}^\omega_{\{0,1\}} : \mathsf{Set}(T_1, T_2, G) \text{ is coherent}\right\}$ (recall that coherent means that $\mathsf{Set}(T_1, T_2, G)$ is an infinite set of finite trees),
  4. $\mathsf{Small} := \left\{(T_1, G) \in \mathsf{B\text{-}Tree}^\omega_{\Sigma_\square} \times \mathsf{B\text{-}Tree}^\omega_{\{0,1\}} : (T_1, G) \text{ is small}\right\}$,

5. $\mathsf{Comp} :=$

$$\{(S,T,T_1,T_2,G) \in \mathscr{T} : S,T \in \mathsf{Set}(T_1,T_2,G) \text{ and either } S <_G T \text{ or } T <_G S\}$$

for $\mathscr{T} := (\mathsf{Tree}_\Sigma)^2 \times (\mathsf{B\text{-}Tree}^\omega_{\Sigma_\square})^2 \times \mathsf{B\text{-}Tree}^\omega_{\{0,1\}}$.

Now, we construct an ω-presentation of $\mathsf{Ext}(\mathfrak{A})$ over the alphabet $\Gamma = \Sigma \cup \{\bot, \square, 0, 1\}$. Recall that we write $T^\bot$ for the lifting of a tree T to the full domain $\{0,1\}^*$ where we use $\bot$ as a padding symbol. We define the domain of the presentation to be the set

$$L := \{T^\bot : T \in \mathsf{Tree}_\Sigma\} \cup \mathsf{B\text{-}Tree}^\omega_{\Sigma_\square} \cup \mathsf{B\text{-}Tree}^\omega_{\{0,1\}}.$$

This set is obviously ω-regular. Furthermore, it is easy to describe a bijection $h : L \to A'$ by stating its inverse $h^{-1} : A' \to L$. For $T \in \mathsf{Tree}_\Sigma$, we set $h^{-1}(T) := T^\bot$, for all other elements T of A', we set $h^{-1}(T) := T$. It remains to show that the (h-preimages of the) relations of $\mathsf{Ext}(\mathfrak{A})$ are ω-automatic. This is trivial for $h^{-1}(\mathsf{Tree}_\Sigma) = \{T^\bot : T \in \mathsf{Tree}_\Sigma\}$, $h^{-1}(\mathsf{B\text{-}Tree}^\omega_{\Sigma_\square}) = \mathsf{B\text{-}Tree}^\omega_{\Sigma_\square}$ and $h^{-1}(\mathsf{B\text{-}Tree}^\omega_{\{0,1\}}) = \mathsf{B\text{-}Tree}^\omega_{\{0,1\}}$. For the relation A (the universe of $\mathfrak{A}$), we have $h^{-1}(A) = \{T^\bot : T \in A\}$. Recall that $\mathsf{id} : A \to A$ is a tree presentation for $\mathfrak{A}$, so $A \subseteq \mathsf{Tree}_\Sigma$ can be accepted by an automaton. This automaton can be transformed into an ω-automaton accepting $h^{-1}(A)$ (cf. Lemma 2.5.14). A similar argument applies to the basic relations of $\mathfrak{A}$. Thus, it remains to consider the relations In, $\mathsf{Coherent}$, Small and Comp.

Lemma 3.4.31. The relations $h^{-1}(\mathsf{In})$, $h^{-1}(\mathsf{Coherent})$, $h^{-1}(\mathsf{Small})$, and $h^{-1}(\mathsf{Comp})$ are ω-automatic.

Proof. 1. $h^{-1}(\mathsf{In})$: The property "H is a G-tree" is an MSO-definable property of the infinite tree $H \otimes G$ (where we consider H and G as characteristic maps). Similarly, "$T = \mathsf{extree}(H, T_1, T_2, G)$" is an MSO-definable property of the infinite tree $\bigotimes(T^\bot, T_1, T_2, H, G)$. Thus, also "$T \in \mathsf{Set}(T_1, T_2, G)$" is an MSO-definable property of the infinite tree $\bigotimes(T^\bot, T_1, T_2, G)$. Hence, ω-automaticity of $h^{-1}(\mathsf{In})$ follows from Theorem 2.5.13.

2. $h^{-1}(\mathsf{Coherent})$: the condition "for any G-tree H, $\mathsf{extree}(H, T_1, T_2, G)$ is actually a finite tree" is an MSO-definable property of the tree $\bigotimes(T_1, T_2, G)$ and can therefore be checked by an ω-automaton by Theorem 2.5.13. Now assume that $\mathsf{Set}(T_1, T_2, G)$ is a set of finite trees. It is infinite if and only if the union of the domains of its elements is infinite, i.e., if this union contains an infinite branch. But the property "there is an infinite branch b such that for each element d in b there is a G-tree H_d such that $d \in \mathsf{extree}(H_d, T_1, T_2, G)$" is an MSO-definable property of the tree $\bigotimes(T_1, T_2, G)$.

3. $h^{-1}(\mathsf{Small})$: the property "for each $d \in G$, there is G-tree H with $d \in H^+$" is an MSO-definable property. Furthermore, "for all $c < d < e$ with $e \in G$ and $d \in G$, it holds that $T_1(c) \neq \square$" is first-order definable on $(G \otimes T_1, <)$ and the prefix order $<$ on $G \cap \{0,1\}^*$ is MSO definable on G.

4. $h^{-1}(\mathsf{Comp})$: the assertion "$H <_G H'$" is an MSO-definable property of the infinite tree $\bigotimes(H, H', G)$: there is a formula that checks for each branch b that either

$$H \cap b \subsetneq H' \cap b \text{ or } H \cap b = H' \cap b$$

and there is no $d \in (b \cap G) \setminus (H^{\oplus})$ with $G(d) = 1$. Furthermore, the maximal G-trees generating S and T are MSO definable in $\bigotimes(S^{\perp}, T^{\perp}, T_1, T_2, G)$: we have already seen that $T = \mathsf{extree}(H, T_1, T_2, G)$ is MSO definable. Thus, the set of G-trees generating T (or S) are definable. The maximal of these trees is the G-tree underlying T (or S). But maximality of a tree among an MSO-definable set of trees is clearly MSO definable. □

Thus, summarising these results, we obtain the following corollary.

Corollary 3.4.32. For each automatic structure $\mathfrak{A}$, the corresponding structure $\mathsf{Ext}(\mathfrak{A})$ is ω-automatic.

Reduction of the Ramsey quantifier

We now inductively translate an $\mathsf{FO}((\mathsf{Ram}^n)_{n\in\mathbb{N}})$ formula in the language of $\mathfrak{A}$ into an FO formula in the language of $\mathsf{Ext}(\mathfrak{A})$. The idea is to replace the occurrence of a Ramsey quantifier like $\mathsf{Ram}^n \bar{x}(\varphi)$ by the assertion that there is a small and coherent representation (T_1, T_2, G) such that all pairwise $<_G$-comparable n-tuples from $\mathsf{Set}(T_1, T_2, G)$ satisfy φ. Our intention is to consider (T_1, T_2, G) as the representation of some tree-comb. We will first define this reduction in detail. Then we prove its soundness. Finally, we show that this reduction is correct.

Definition 3.4.33. For each $\mathsf{FO}(\exists^{\mathrm{mod}}, (\mathsf{Ram}^n)_{n\in\mathbb{N}})$ formula φ in the language of $\mathfrak{A}$, we define its reduction $\mathsf{red}(\varphi)$ to the $\mathsf{FO}(\exists^{\mathrm{mod}})$ language of $\mathsf{Ext}(\mathfrak{A})$ by

$$\begin{aligned}
\mathsf{red}(\varphi) &:= \varphi \text{ for } \varphi \text{ an atomic formula},\\
\mathsf{red}(\varphi \vee \psi) &:= \mathsf{red}(\varphi) \vee \mathsf{red}(\psi),\\
\mathsf{red}(\neg\varphi) &:= \neg\mathsf{red}(\varphi),\\
\mathsf{red}(\exists x \varphi) &:= \exists x (x \in A \wedge \mathsf{red}(\varphi)),\\
\mathsf{red}(\exists^{k,l} x \varphi) &:= \exists^{k,l} x (x \in A \wedge \mathsf{red}(\varphi)),\\
\mathsf{red}(\mathsf{Ram}^n \bar{x}(\varphi)) &:= \exists T_1, T_2 \in \text{B-Tree}^{\omega}_{\Gamma_\square}, G \in \text{B-Tree}^{\omega}_{\{0,1\}} \psi_{\mathrm{CoSm}}(T_1, T_2, G) \wedge \psi_{\mathrm{Ram}}(T_1, T_2, G),
\end{aligned}$$

where

$$\psi_{\mathrm{CoSm}}(T_1, T_2, G) := \mathsf{Coherent}(T_1, T_2, G) \wedge \mathsf{Small}(T_1, G) \wedge \forall x \left(\mathsf{In}(x, T_1, T_2, G) \to x \in A\right)$$

and

$$\psi_{\mathrm{Ram}} := \forall x_1, \ldots, x_n \in \mathrm{Tree}_{\Sigma} \left(\left(\bigwedge_{1\le i\le n} \mathsf{In}(x_i, T_1, T_2, G) \wedge \bigwedge_{1\le i<j\le n} \mathsf{Comp}(x_i, x_j, T_1, T_2, G) \right) \to \mathsf{red}(\varphi) \right).$$

Remark 3.4.34. This reduction of a Ramsey quantifier asserts that there is a representation of an infinite set such that each pairwise comparable n-tuple from this set satisfies φ. At first, this seems to be a weaker condition than the assertion of the Ramsey quantifier because there are tuples of pairwise distinct elements that are not tuples of pairwise $<_G$ comparable elements. But it turns out that this condition is sufficient: there is an infinite linearly $<_G$-ordered subset S' for each $\mathsf{Set}(T_1, T_2, G)$ where (T_1, T_2, G) is a coherent and small representation.

In the following we first show that this translation is sound, i.e., for any formula φ, if $\mathsf{Ext}(\mathfrak{A}) \models \mathsf{red}(\varphi)$, then $\mathfrak{A} \models \varphi$. Afterwards, we prove the correctness, i.e., for any formula φ, if $\mathfrak{A} \models \varphi$, then $\mathsf{Ext}(\mathfrak{A}) \models \mathsf{red}(\varphi)$.

3.4.3 Soundness of the Reduction

In order to prove the soundness of our reduction, we start with a technical lemma. It asserts that for (T_1, T_2, G) some representation of an infinite set of finite trees, there is at least one branch with infinitely many nodes in G. We use this fact in order to prove the existence of an infinite linear $<_G$-ordered subset of every small representation.

Lemma 3.4.35. Let $T_1, T_2 : \{0,1\}^* \to \Sigma_\square$ and $G \subseteq \{0,1\}^*$ such that (T_1, T_2, G) is coherent, i.e., $S = \mathsf{Set}(T_1, T_2, G)$ is an infinite set of finite Σ-trees. If b is an infinite branch in $\bigcup_{T \in S} \mathsf{dom}(T)$, then $|b \cap G| = \infty$.

Proof. Let b be an infinite branch in $\bigcup_{T \in S} \mathsf{dom}(T)$. Assume that $|b \cap G| < \infty$. Then there are $d_1 < d_2 < \cdots < d_n \in \{0,1\}^*$ such that $G \cap b = \{d_1, d_2, \ldots, d_n\}$. Under this assumption, $b \subseteq \bigcup_{T \in S} \mathsf{dom}(T)$ implies that there is some $i \leq n$ and some G-tree H with $d_i \in H^+$ such that $\mathsf{dom}(\mathsf{extract}(d_i, T_2, G)) \cap B$ is infinite. But this implies that S contains an infinite tree which is a contradiction to the assumption that $S \subseteq \mathsf{Tree}_\Sigma$. □

Lemma 3.4.36. Let $T_1, T_2 : \{0,1\}^* \to \Sigma_\square$ and $G \subseteq \{0,1\}^*$ such that $S := \mathsf{Set}(T_1, T_2, G)$ is coherent and (T_1, G) is small. Then there is an infinite subset $S' \subseteq S$ which is linearly ordered by $<_G$.

Proof. We show that for every tree $T \in S$ there is a tree $T' \in S$ with $T <_G T'$. Let $T \in S$ and H be the maximal G-tree such that $T = \mathsf{extree}(H, T_1, T_2, G)$.

Let

$$D := \{d \in H^+ : \text{ there is an infinite branch } b \text{ such that } d < b \text{ and } (b \setminus (H^\oplus)) \cap G \neq \emptyset\}.$$

Since $|S| = \infty$, there is an infinite branch in $\bigcup_{T \in S} \mathsf{dom}(T)$. Together with the previous lemma, this implies that D is nonempty.

Since (T_1, G) is small, for each $d \in D$ there exists a G-tree H_d with $d \in H_d$. Set $H' := H \cup \bigcup_{d \in D} H_d$. We claim that H' is a G-tree with $H <_G H'$.

Since D is finite, H' is a finite tree. Furthermore for each $e \in H'^+$ either $e \in H^+$ or $e \in H_d^+$ for some $d \in D$. Thus, H' is a G-tree. We claim that $H <_G H'$. It is clear that $d \in H' \setminus H$ for all $d \in D \neq \emptyset$ and that $H \subseteq H'$ whence $H \subsetneq H'$. Now assume that b is an infinite branch such that $H \cap b = H' \cap b$. We have to show that $(b \setminus (H'^\oplus)) \cap G = \emptyset$.

Heading for a contradiction assume that there is some element e contained in this set. Let d be the unique element in $b \cap H^+$. By definition of D, $d \in D$. Thus, $d \in H' \setminus H$ which contradicts $H \cap b = H' \cap b$.

Hence, for $T' := \mathsf{extree}(H', T_1, T_2, G)$ we have $T <_G T'$. Repeating this construction ad infinitum we obtain an infinite, linearly $<_G$-ordered subset of S. □

Lemma 3.4.37. Let $\varphi \in \mathsf{FO}(\exists^{\mathsf{mod}}, (\mathsf{Ram}^n)_{n \in \mathbb{N}})$ be a sentence. If $\mathsf{Ext}(\mathfrak{A}) \models \mathsf{red}(\varphi)$, then $\mathfrak{A} \models \varphi$.

Proof. Since we want to prove the proposition by induction on the construction of φ, we also have to consider formulas with free variables. Hence, the statement we actually prove is the following:

Claim. Let $\varphi \in \mathsf{FO}(\exists^{\mathsf{mod}}, (\mathsf{Ram}^n)_{n\in\mathbb{N}})$ be a formula with free variables among $x_1, \ldots, x_n$ and let $a_1, a_2, \ldots, a_n \in A$. If $\mathsf{Ext}(\mathfrak{A}), (a_1, a_2, \ldots, a_n) \models \mathsf{red}(\varphi)$, then $\mathfrak{A}, (a_1, a_2, \ldots, a_n) \models \varphi$.

The inductive proof of this claim is rather clear except for the case $\varphi = \mathsf{Ram}^n \bar{x}(\psi)$. So let $a_1, a_2, \ldots, a_m \in \mathfrak{A}$, and let $\mathsf{Ext}(\mathfrak{A}), (a_1, a_2, \ldots, a_m) \models \mathsf{red}(\varphi)(y_1, y_2, \ldots, y_m)$. Then there are infinite trees T_1, T_2 and G with the properties given by $\mathsf{red}(\varphi)$. In particular, $S = \mathsf{Set}(T_1, T_2, G) \subseteq A$ is an infinite set of finite trees and (T_1, G) is small. By Lemma 3.4.36, there is an infinite $S' \subseteq S$ that is linearly ordered by $<_G$. Hence, from the properties of T_1, T_2, G, we obtain that

$$\mathsf{Ext}(\mathfrak{A}), (t_1, \ldots, t_n, a_1, a_2, \ldots, a_m,) \models \mathsf{red}(\psi)(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)$$

for every tuple $(t_1, \ldots, t_n) \in (S')^n$ such that the t_i are pairwise $<_G$-comparable. Since the pairwise $<_G$-comparable tuples are exactly the pairwise distinct tuples in S', $S' \subseteq A$ witnesses $\mathfrak{A}, (a_1, a_2, \ldots, a_m) \models \mathsf{Ram}^n \bar{x}(\psi)(y_1, y_2, \ldots, y_m)$. □

3.4.4 Correctness of the Reduction

The outline of the correctness proof is as follows. We fix some formula $\varphi := \mathsf{Ram}^n \bar{x}(\psi)$. We have already seen that every witness for the Ramsey quantifier in φ contains a subset that forms a tree-comb. In the following we show that this tree-comb contains a certain tree-comb $D \subseteq C$ such that the trees T_1^D, T_2^D, and G^D witness the reduction $\mathsf{red}(\varphi)$. This means that the pairwise $<_G$-comparable tuples from the closure $\mathsf{CL}(D)$ witness $\mathsf{red}(\psi)$. In order to prove this, we introduce the notion of homogeneity of some tree-comb with respect to an automaton $\mathscr{A}$. We will show that any tree-comb witnessing a Ramsey quantifier contains a homogeneous tree-comb. Furthermore, an automaton accepts all pairwise distinct n-tuples from a homogeneous tree-comb if and only if it accepts all pairwise comparable n-tuples from the closure of this tree-comb. This completes the proof because the existence of such a set is exactly what the reduction of φ asserts.

Before we give formal definitions, let us informally explain what the concept of homogeneity is. Consider a formula $\mathsf{Ram}^n \bar{x}(\psi)$ with $\psi \in \mathsf{FO}$. Assume that there is some tree-comb $C = (T_i)_{i\geq 1} \subseteq \mathfrak{A}$ witnessing this quantifier on $\mathfrak{A}$. Let $\mathscr{A}$ denote the automaton corresponding to ψ, i.e., $\mathfrak{A}, \bar{a} \models \psi$ if and only if $\mathscr{A}$ accepts $\bigotimes \bar{a}$. Thus, any pairwise distinct n-tuple from C is accepted by $\mathscr{A}$. Recall that for any finite tree-domain D most of the elements from C agree on D. More precisely, if $D = \mathsf{dom}(T_i)$ then $T_{i+1}, T_{i+2}, T_{i+3}, \ldots$ agree on D. We call the tree-comb homogeneous with respect to $\mathscr{A}$, if all n-tuples from the closure of the tree-comb that agree on some finite domain D are accepted by runs that coincide on D.

The purpose of this concept is the following: First of all, note that every tree T from the closure $\mathsf{CL}(C)$ locally coincides with a tree from C in the following sense. Let $D \subseteq \mathsf{dom}(T_i)^{\oplus} \setminus \mathsf{dom}(T_{i-1})$ be some tree-domain, i.e., there is a unique minimal element $d \in D$ and for every $d' \in D \setminus \{d\}$ the predecessor of d' is contained in D. Then, $T{\restriction_D}$ coincides with either $T_{i-1}{\restriction_D}$ or $T_i{\restriction_D}$ or $T_{i+1}{\restriction_D}$. We denote by H the tree underlying T. The three cases correspond to the following three conditions on H.

1. If there are $e_1 < e_2 \leq d$ such that $e_1 \in H^+$ and $e_2 \in G^C$, then

$$D \cap \mathsf{dom}(T) = \emptyset = D \cap \mathsf{dom}(T_{i-1}).$$

2. If there is some $e_1 \leq d$ such that $e_1 \in H^+$ and G does not contain any element between e_1 and d, then $T\restriction_D$ coincides with $T_i\restriction_D$ (cf. Lemma 3.4.19).

3. If $d \in H$, $T\restriction_D$ coincides with $T_{i+1}\restriction_D$. Moreover, the definition of a tree-comb implies that $T\restriction_D$ then coincides with $T_k\restriction_D$ for all $k > i$.

Now, given a pairwise $<_{G^C}$-comparable n-tuple $\bar{T}$ from $\mathsf{CL}(C)$, $\bar{T}$ coincides locally with pairwise distinct n-tuples from C. We can then define a function ρ on $\bar{T}$ by locally copying the accepting runs on the n-tuples from C. If C is homogeneous with respect to $\mathscr{A}$, ρ is an accepting run on $\bar{T}$ due to the following fact. Let

$$D_1 \subseteq \mathsf{dom}(T_i)^{\oplus} \setminus \mathsf{dom}(T_{i-1}) \text{ and } D_2 \subseteq \mathsf{dom}(T_{i+1})^{\oplus} \setminus \mathsf{dom}(T_i)$$

be maximal tree domains such that D_1 and D_2 are touching. Then there are tuples $\bar{C}_1, \bar{C}_2 \in C$ such that ρ coincides on D_1 with the accepting run on $\bar{C}_1$ and ρ coincides on D_2 with the accepting run on $\bar{C}_2$. Due to homogeneity, the accepting run on $\bar{C}_2$ coincides with the accepting run on $\bar{C}_1$ on the path from the minimal element of D_1 to the minimal element of D_2. Thus, ρ respects the transition relation at the border between D_1 and D_2. Since this argument applies at all borders where ρ consists of copies of different accepting runs, ρ respects the transition relation whence it is a run on $\bar{T}$. Moreover, the function copies the behaviour of an accepting run on each branch. Hence, the run is an accepting run on $\bar{T}$.

The precise definition of homogeneity is more complicated than indicated above because we have to deal with different permutations of fixed n-tuples. When we investigate pairwise $<_{G^C}$-comparable tuples, we can order these tuples in various ways. But the accepting run for each permutation of a tuple may differ from all the accepting runs on the other permutations. Thus, our definition of homogeneity asserts that n-tuples from the tree-comb share similar accepting runs if their elements are ordered by $<_{G^C}$ in the same manner. Let us first define some auxiliary notation. Afterwards, we state the exact definition of homogeneity.

Definition 3.4.38.

- Let C be some tree-comb and $D \subseteq \mathsf{CL}(C)$. Then we write $\left[\frac{D}{n}\right]$ for the set of $<_{G^C}$-increasing n-tuples from D.

- For $\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ a permutation and $\bar{x} = (x_1, x_2, \ldots, x_n)$ some n-tuple, we write $\sigma(\bar{x}) := (x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)})$.

- Let D and E be subsets of some tree-comb C such that $d <_{G^C} e$ for all $d \in D$ and $e \in E$. Furthermore, let $\mathscr{A}$ be a deterministic automaton recognising an n-ary relation of Σ-trees. We write $\rho_{\sigma(\bar{T})}$ for the run of $\mathscr{A}$ on $\bigotimes \sigma(\bar{T})$ for all n-tuples $\bar{T}$ of trees and all permutations σ.

  Set $F := \bigcup_{T \in D} \mathsf{dom}(T)^{\oplus}$. We say $\mathscr{A}$ runs homogeneously on E with respect to D, if for each permutation σ of n elements and each number $0 \leq m \leq n$ the following holds:

  For all $\bar{T} \in \left[\frac{D}{m}\right]$, all $\bar{U} \in \left[\frac{E}{n-m}\right]$ and all $\bar{V} \in \left[\frac{E}{n-m}\right]$,

  $$\rho_{\sigma(\bar{T}\bar{U})}\restriction_F = \rho_{\sigma(\bar{T}\bar{V})}\restriction_F,$$

  i.e., the runs on $\bigotimes \sigma(\bar{T}\bar{U})$ and $\bigotimes \sigma(\bar{T}\bar{V})$ coincide on the domain F.

This means that different tuples from E that are in the same order with respect to the tree-comb order $<_G$ have identical runs with each fixed tuple from D on domain F, where F may be seen as the "maximal domain" of D. Note that this assertion is symmetric in the order in which we mix the tuple from D with the tuples from E.

Definition 3.4.39. Let $\mathscr{A}$ be some deterministic automaton and $C = (T_i)_{i\geq 1}$ be a tree-comb. Set

$$D_n := \{T_i : 1 \leq i < n\} \text{ and } E_n := \{T_i : n \leq i\} \text{ for all } n \geq 1.$$

We say C is homogeneous with respect to $\mathscr{A}$, if, for all $n \in \mathbb{N}$, $\mathscr{A}$ runs homogeneously on E_n with respect to D_n.

The crucial observation for the correctness proof is the following. Any tree-comb whose pairwise distinct n-tuples are all accepted by an automaton $\mathscr{A}$ contains a subcomb that is homogeneous with respect to $\mathscr{A}$. Every set M that witnesses the Ramsey quantifier $\mathsf{Ram}^n\bar{x}(\psi)$ contains a tree-comb C which also witnesses the Ramsey quantifier. We are going to show that C contains a subcomb C' which is homogeneous with respect to $\mathscr{A}_\psi$ (where $\mathscr{A}_\psi$ corresponds to ψ). Because of this homogeneity, we can then construct an accepting run of $\mathscr{A}_\psi$ on each pairwise comparable n-tuple from $\mathsf{CL}(C')$. Since $\mathscr{A}_\psi$ corresponds to ψ, this implies that every pairwise comparable n-tuple from $\mathsf{CL}(C')$ satisfies ψ. Thus, the representation of such a closure is a witness for $\mathsf{red}(\mathsf{Ram}^n\bar{x}(\psi))$.

Lemma 3.4.40. Let C be a tree-comb and $\mathscr{A}$ some deterministic automaton such that $\mathscr{A}$ accepts $\bigotimes \sigma(\bar{T})$ for all $\bar{T} \in \left[\frac{C}{n}\right]$ and all permutations σ. Then there is a subcomb $C^{\mathscr{A}} \subseteq C$ which is homogeneous with respect to $\mathscr{A}$.

Proof. We generate $C^{\mathscr{A}}$ by the use of Ramsey's Theorem (Theorem 3.4.17).

For each $\bar{T} \in \left[\frac{C}{n}\right]$ and each permutation σ, we denote the accepting run of $\mathscr{A}$ on $\bar{T}$ by $\rho_{\sigma(\bar{T})}$.

We are going to define two infinite chains

$$D_0 \subsetneq D_1 \subsetneq D_2 \subsetneq \dots \text{ and}$$
$$C_0 \supsetneq C_1 \supsetneq C_2 \supsetneq \dots$$

such that $\mathscr{A}$ runs homogeneously on C_i with respect to D_i. D_{i+1} will extend D_i by exactly one finite tree T_{i+1}. The sequence of these trees then forms a tree-comb that is homogeneous with respect to $\mathscr{A}$.

At the beginning we set $D_0 := \emptyset$. Since $\emptyset^\oplus = \{\varepsilon\}$, we have to provide an infinite set $C_0 \subseteq C$ such that for each permutation σ all $\bar{T}, \bar{T}' \in \left[\frac{C_0}{n}\right]$ satisfy $\rho_{\sigma(\bar{T})}(\varepsilon) = \rho_{\sigma(\bar{T}')}(\varepsilon)$. This set C_0 can be obtained by applying Ramsey's Theorem as follows: For each $\bar{T} \in \left[\frac{C_0}{n}\right]$ and for each permutation σ the function $\rho_{\sigma(\bar{T})}{\restriction}_{\{\varepsilon\}}$ has finite domain and range. Let $\sigma_1, \sigma_2, \dots, \sigma_m$ be a fixed enumeration of all permutations of n elements. Assigning

$$\bar{T} \mapsto (\rho_{\sigma_1(\bar{T})}{\restriction}_{\{\varepsilon\}}, \rho_{\sigma_2(\bar{T})}{\restriction}_{\{\varepsilon\}}, \dots, \rho_{\sigma_m(\bar{T})}{\restriction}_{\{\varepsilon\}})$$

for each $\bar{T} \in \left[\frac{C}{n}\right]$ induces a finite colouring of all n-element subsets of C: since C is linearly ordered by $<_{G^C}$ (see Lemma 3.4.29), each pairwise distinct n-tuple has a unique representative among the $<_{G^C}$ increasing sequences of length n. Furthermore, the range of this map is finite.

By Ramsey's theorem, there is an infinite subset $C_0 \subseteq C$ that is homogeneous with respect to this colouring, i.e., if the $<_{C^G}$-order of two tuples from C coincides, then their accepting runs coincide on the state at the root.

We now construct D_{i+1} and C_{i+1} from D_i and C_i by generalising this process. Assume that $D_i, C_i \subseteq C$ are disjoint sets such that D_i is finite, C_i is infinite, and $T <_{G^C} T'$ for each $T \in D_i$ and $T' \in C_i$. Furthermore, assume that $\mathscr{A}$ runs homogeneously on C_i with respect to D_i.

Let T_i be the minimal element of C_i with respect to $<_{G^C}$. We set $D_{i+1} := D_i \cup \{T_i\}$. Applying Ramsey's Theorem iteratively for each $0 \leq k < n$ and each $\bar{T} \in \left[\frac{D_{i+1}}{k}\right]$, we can choose an infinite $C_{i+1} \subseteq C_i \setminus \{T_i\}$ such that $\mathscr{A}$ runs homogeneously on C_{i+1} with respect to D_{i+1}. We explain one of these applications of Ramsey's Theorem in detail:

Fix a number $k \leq n$ and some $\bar{T} \in \left[\frac{D_{i+1}}{k}\right]$. In this step we consider the colouring that maps each $\bar{U} \in \left[\frac{C_i}{n-k}\right]$ to

$$(\rho_{\sigma_1(\bar{T}\bar{U})}\restriction_F, \rho_{\sigma_2(\bar{T}\bar{U})}\restriction_F, \ldots, \rho_{\sigma_m(\bar{T}\bar{U})}\restriction_F) \text{ where}$$
$$F := \bigcup_{T \in D_{i+1}} \mathrm{dom}(T)^{\oplus}.$$

Since F is finite, this induces a colouring of finite range on the k-tuples of C_i. Applying Ramsey's Theorem, there is a homogeneous infinite subset $C' \subseteq C_i$ with respect to this colouring.

Iterating this construction for each $k \leq n$ and each $\bar{T} \in \left[\frac{D_{i+1}}{k}\right]$, we obtain a subset $C_{i+1} \subseteq C' \subseteq C_i$ such that $\mathscr{A}$ runs homogeneously on C_{i+1} with respect to D_{i+1}.

Furthermore, it is clear that $T <_{G^C} T'$ for all $T \in D_{i+1}$ and $T' \in C_{i+1}$ because of the following facts. The same claim is true for D_i and C_i. Furthermore, D_{i+1} is D_i extended by the minimal element of C_i and C_{i+1} does not contain the minimal element of C_i.

Repeating this construction, we obtain a sequence of trees $T_1, T_2, T_3, \ldots$ such that $D_i = \bigcup_{j \leq i} \{T_j\}$. The sequence $(T_i)_{i \geq 1}$ is a tree-comb because it is a subsequence of C. We set $C^{\mathscr{A}} := (T_i)_{i \geq 1}$. Note that $C^{\mathscr{A}}$ is homogeneous with respect to $\mathscr{A}$ by construction. □

Next, we show the following. Let C be some tree-comb such that all pairwise distinct n-tuples from C are accepted by some automaton $\mathscr{A}$. If C is homogeneous with respect to $\mathscr{A}$, then $\mathscr{A}$ accepts all pairwise comparable n-tuples of $\mathsf{CL}(C)$. The proof of this claim relies on the fact that every tree in $\mathsf{CL}(C)$ is locally similar to one of the trees in C and accepting runs for different trees coincide on equal prefixes.

In the next lemma we use the following notation. Let T be some tree and $D \subseteq \mathrm{dom}(T)^{\oplus}$ an initial segment. We call a map $\rho : D \to Q$ a partial run of $\mathscr{A}$ on T if ρ respects the transition relation of $\mathscr{A}$ on T. We call a partial run ρ accepting if $\rho(\varepsilon)$ is a final state. Let ρ be some partial run on some tree T. For some $d \in \{0,1\}^*$ we call ρ total and correctly initialised (tci) towards d if there is some $e \leq d$ such that $e \in \mathrm{dom}(T)^+$ and $\rho(e) = q_I$, i.e., the domain of T ends at some ancestor of d and the border of its domain along this branch is labelled by the initial state.

Remark 3.4.41. Note that a partial run ρ on some tree T is an accepting run if and only if it is accepting and tci towards all $d \in \mathrm{dom}(T)^+$.

Lemma 3.4.42. Let $\mathscr{A} = (Q, \Sigma, q_I, F, \Delta)$ be some deterministic automaton. Let C be a tree-comb such that $\mathscr{A}$ accepts all pairwise disjoint n-tuples of C. If C is homogeneous with respect to $\mathscr{A}$, then $\mathscr{A}$ accepts all pairwise comparable n-tuples from $\mathsf{CL}(C)$, i.e., $\mathscr{A}$ accepts $\sigma(T)$ for all permutations σ and all $\bar{T} = (T_1, T_2, \ldots, T_n) \in \left[\frac{\mathsf{CL}(C)}{n}\right]$.

Proof. We write (T_1^C, T_2^C, G^C) for the representation of $C = (C_i)_{i \geq 1}$. Assume that

$$\bar{T} = (T_1, T_2, \ldots, T_n) \in \left[\frac{\mathsf{CL}(C)}{n}\right].$$

Furthermore, assume that $H_1, H_2, \ldots, H_n$ are the G^C-trees that underlie the trees $T_1, T_2, \ldots, T_n$ (i.e., for each i, $T_i = \mathsf{extree}(H_i, T_1^C, T_2^C, G^C)$ and H_i is maximal with this property).

We assume that $\sigma = \mathsf{id}$ (due to the symmetric definition of homogeneity, the proof is completely analogous for any other permutation).

For each $\bar{C} \in \left[\frac{C}{n}\right]$, we write $\rho_{\bar{C}}$ for the accepting run of $\mathscr{A}$ on $\bar{C}$.

Set $F_k := \bigcup_{i<k} \mathsf{dom}(C_i)^{\oplus}$. We will define an accepting run $\rho_{\bar{T}}$ of $\mathscr{A}$ on $\bigotimes \bar{T}$ as the union of accepting partial runs $\rho_{\bar{T}} \restriction_{F_k}$.

We start with the definition of $\rho_{\bar{T}} \restriction_{F_1}$. Note that $F_1 = \{\varepsilon\}$. We set $\rho_{\bar{T}}(\varepsilon) := \rho_{\bar{C}}(\varepsilon)$ for an arbitrary $\bar{C} \in \left[\frac{C}{n}\right]$. Recall that by homogeneity of $\bar{C}$, this definition is independent of the concrete choice of $\bar{C}$. Furthermore, since $\rho_{\bar{C}}$ is accepting $\rho_{\bar{T}} \restriction_{F_1}$ is an accepting partial run.

For $d = \varepsilon, k = 1$, and $m = 1$, $\rho_{\bar{T}} \restriction_{F_1}$ satisfies the following properties.

1. $d \in \mathsf{dom}(C_{k-1})^+ \setminus \bigcup_{j<k-1}(\mathsf{dom}(C_j)^{\oplus})$ (where we define $C_0 := \emptyset$),
2. $d \in \mathsf{dom}(T_m)^{\oplus}$ (just by definition of $^{\oplus}$),
3. $d \in H_j^{\oplus}$ for $m \leq j \leq n$, and
4. $\rho_{\bar{T}}(d) = \rho_{\bar{C}}(d)$ for any $\bar{C} = C_{i_1}, C_{i_2}, \ldots, C_{i_n}$ with $k \leq i_1 < i_2 < \cdots < i_n$.

For each $k \geq 1$, we inductively extend the accepting partial run $\rho_{\bar{T}} \restriction_{F_{k-1}}$ to an accepting partial run on domain $F_k \cap \mathsf{dom}(\bigotimes \bar{T})^{\oplus}$. In each step of this construction, we preserve the property that for each maximal element $d \in F_k$, at least one of the following conditions hold.

1. $\rho_{\bar{T}}$ is a tci accepting partial run on $\bigotimes \bar{T}$ towards d.
2. There is some $1 \leq m \leq n$ such that the following conditions are satisfied:
   a) $d \in \mathsf{dom}(C_{k-1})^+ \setminus F_{k-1}$, i.e., $d \in \mathsf{dom}(C_{k-1})^+) \setminus \bigcup_{j<k-1}(\mathsf{dom}(C_j)^{\oplus})$,
   b) $d \notin \mathsf{dom}(T_j)$ for all $1 \leq j < m$,
   c) $d \notin H_j^{\oplus}$ for $1 \leq j < m$,
   d) $d \in \mathsf{dom}(T_m)^{\oplus}$,
   e) $d \in H_j^{\oplus}$ for all $m \leq j \leq n$,
   f) there are natural numbers $1 \leq i_1 < i_2 < \cdots < i_{m-1} < k \leq i_m < i_{m+1} < \cdots < i_n$ such that $\rho_{\bar{T}}(d) = \rho_{\bar{C}}(d)$ where $\bar{C} := (C_{i_1}, C_{i_2}, \ldots, C_{i_n})$. We stress that $\rho_{\bar{C}}(d)$ does not depend on the concrete choice of $i_m, i_{m+1}, \ldots, i_n$, i.e., for all $\bar{C}' := (C_{i_1}, C_{i_2}, \ldots, C_{i_{m-1}}, C_{i'_m}, C_{i'_{m+1}}, \ldots C_{i'_n})$ with $k \leq i'_m < i'_{m+1} < \cdots < i'_n$, we have $\rho_{\bar{T}}(d) = \rho_{\bar{C}}(d)$ due to the homogeneity of $\mathscr{A}$ on $(C)_{i \geq k}$ with respect to $(C)_{i<k}$.

Note that these conditions imply that $\rho_{\bar{T}}\restriction_{F_k}$ is defined on $F_k \cap \mathrm{dom}(\bigotimes \bar{T})$ and that $\rho_{\bar{T}}\restriction_{F_k}$ may be extendable to an accepting run of $\mathscr{A}$ on $\bar{T}$. Especially, if the first condition applies to all maximal $d \in F_k$, then $\rho_{\bar{T}}$ is an accepting run on $\bigotimes \bar{T}$.

We now extend $\rho_{\bar{T}}$ to the maximal possible segment of $F_{k+1} = \bigcup_{i<k+1} \mathrm{dom}(C_i)^{\oplus}$, i.e., to $F_{k+1} \cap (\mathrm{dom}(\bigotimes \bar{T})^{\oplus})$.

By assumption, we only have to extend $\rho_{\bar{T}}$ at the maximal positions $d \in F_k$ where the second condition holds. We will distinguish the following three cases.

1. $d \notin \mathrm{dom}(C_j)$ for all $j \geq 1$,
2. $d \in \mathrm{dom}(C_k)$ and $d \in H_m$, and
3. $d \in \mathrm{dom}(C_k)$ but $d \notin H_m$.

Let us first explain why this case distinction is complete: Assume that there is some $j \geq 1$ such that $d \in \mathrm{dom}(C_j)$. Then $j \geq k$ because $d \in \mathrm{dom}(C_{k-1})^+ \setminus F_{k-1}$. Since $d \in \mathrm{dom}(C_{k-1})^{\oplus}$ and due to the tree-comb property, $d \in \mathrm{dom}(C_j)$ for some $j \geq k$ if and only if $d \in \mathrm{dom}(C_j)$ for all $j \geq k$. Thus, we conclude that $d \in \mathrm{dom}(C_k)$.

For the case distinction, let us fix a tuple $\bar{C} = (C_{i_1}, C_{i_2}, \ldots, C_{i_n})$ witnessing condition 2f.

1. $d \notin \mathrm{dom}(C_j)$ for all $j \geq 1$: first of all, note that in this case either $d \in \mathrm{dom}(C_{i_j})^+$ for some $1 \leq j \leq n$ and $\rho_{\bar{C}}(d) = q_I$ or $d \notin \mathrm{dom}(C_{i_j})^{\oplus}$ for all $1 \leq j \leq n$ and $d \notin \mathrm{dom}(\rho_{\bar{C}})$.

   Secondly, by definition of G^C, we have $G^C \cap \{e : d < e\} = \emptyset$ whence $d \notin H$ for all G-trees H. Especially, $d \notin H_i$ for $1 \leq i \leq n$. Furthermore, by Lemma 3.4.19 $\mathsf{extract}(d, T_2^C, G^C) = \emptyset$. Hence, $d \notin \mathrm{dom}(T_i)$ for all i. Recall that $d \in \mathrm{dom}(T_m)^{\oplus}$ by assumption, whence $d \in \mathrm{dom}(\bigotimes \bar{T})^+$ and furthermore, $d \in F_k \cap (\mathrm{dom}(\bigotimes \bar{T})^{\oplus})$. Thus, $\rho_{\bar{T}}\restriction_{F_k}$ is defined at d.

   Putting these two facts together, it is only possible that $\rho_{\bar{C}}$ and $\rho_{\bar{T}}$ agree on d if $\rho_{\bar{T}}(d) = \rho_{\bar{C}}(d) = q_I$ whence $\rho_{\bar{T}}$ is an accepting partial run on $\bigotimes \bar{T}$ that is tci towards d. Thus, the first condition is satisfied for all $d' \in \{e : d \leq e\}$.

2. $d \in \mathrm{dom}(C_k)$ and $d \in H_m$: first of all, we claim that $\mathrm{dom}(C_k) \cap \{e : d \leq e\} \subseteq H_m$.

   By definition of H_m and G^C, $e \in H_m^+$ implies that $e \in \mathrm{dom}(C_l)^+ \setminus \bigcup_{l'<l} (\mathrm{dom}(C_{l'})^{\oplus})$ for some $l \in \mathbb{N}$. Due to $d \in \mathrm{dom}(C_{k-1})^+ \setminus F_{k-1}$, no proper successor e of d is contained in $\mathrm{dom}(C_l)^+$ for $l < k$. Thus, the first descendants of d that are contained in G are contained in $\mathrm{dom}(C_k)^+$. Thus, all elements between d and $\mathrm{dom}(C_k)^+$ are contained in H_m.

   Since $H_m \subsetneq H_j$, the same holds for all H_j with $j \geq m$. Thus, $T_m, T_{m+1}, \ldots, T_n$ agree with $C_{k+1}, C_{k+2}, \ldots, C_{k+n-m}$ on $D := \mathrm{dom}(C_k) \cap \{e : d \leq e\}$ (cf. Lemma 3.4.19).

   Furthermore, $D \cap \mathrm{dom}(T_j) = \emptyset$ for $j < m$ by the assumption that $d \notin \mathrm{dom}(T_j)$ for all $j < m$. Similarly, $d \notin \mathrm{dom}(C_{i_j})$ for $j < m$ due to $i_j < k$ and $d \notin \bigcup_{l<k} \mathrm{dom}(C_l)$ (recall that d is maximal in F_k.

   Thus, T_j agrees with C_{i_j} for all $j < m$ on the subtree rooted at d.

   It follows that, for $\bar{C}' := (C_{i_1}, C_{i_2}, \ldots, C_{i_{m-1}}, C_{k+1}, C_{k+2}, \ldots C_{k+n-m})$, the trees $\bigotimes \bar{C}'$ and $\bigotimes \bar{T}$ agree on D. By condition 2f, $\rho_{\bar{T}}(d) = \rho_{\bar{C}'}(d) = \rho_{\bar{C}}(d)$. Thus, setting $\rho_{\bar{T}}(e) := \rho_{\bar{C}'}(e)$ for all $e \in D^{\oplus} \cap (\mathrm{dom}(\bigotimes \bar{C}')^{\oplus})$ extends $\rho_{\bar{T}}$ in such a way that it still is

a partial run on $\bigotimes \bar{T}$. Note that the maximal elements of $D^{\oplus}$ are by definition maximal elements of F_{k+1}. We claim that for any such element d' condition 1 or condition 2 holds. There are the following cases.

For the first case, assume that $d' \notin \mathrm{dom}(\rho_{\bar{T}})$. This implies that $d' \notin \mathrm{dom}(\bigotimes \bar{T})^{\oplus}$ whence by coincidence of $\bigotimes \bar{T}$ and $\bigotimes \bar{C}'$ on D, it follows that $d' \notin \mathrm{dom}(\bigotimes \bar{C}')^{\oplus}$. Thus, $\bigotimes \bar{C}'$ and $\bigotimes \bar{T}$ agree on the path from d to d'. $\rho_{\bar{C}'}$ is tci on $\bigotimes \bar{C}'$ towards d' because $d' \notin \mathrm{dom}(\bigotimes \bar{C}')$. Since $\rho_{\bar{C}'}$ and $\rho_{\bar{T}}$ agree on this path, $\rho_{\bar{T}}$ is tci on $\bigotimes \bar{T}$ towards d'.

For the second case, assume that $d' \in \mathrm{dom}(\rho_{\bar{T}})$. In this case, we show that the second condition holds for $k+1$ and m.

a) we have to show that $d' \in \mathrm{dom}(C_k)^+ \backslash F_k$. $d' \in \mathrm{dom}(C_k)^+$ follows from its definition while $d' \notin F_k$ follows from the facts that $d < d'$, $d \notin F_{k-1}$ and $d \in \mathrm{dom}(C_{k-1})^+$: note that $F_k = F_{k-1} \cup \mathrm{dom}(C_{k-1})^{\oplus}$ and d is by definition a maximal element of this set.

b) $d' \notin \mathrm{dom}(T_j)^{\oplus}$ for all $1 \leq j < m$ because $d < d'$ and $d \notin \mathrm{dom}(T_j)$ for all $1 \leq j < m$ by assumption.

c) $d' \notin H_j^{\oplus}$ for $1 \leq j < m$ because $d < d'$ and $d \notin H_j^{\oplus}$ for $1 \leq j < m$ by assumption.

d) Since $d' \in \mathrm{dom}(\rho_{\bar{T}})$, $d' \in \bigcup_{j=1}^{n} \mathrm{dom}(T_j)^{\oplus}$. Thus, there is some $1 \leq j \leq n$ such that $d' \in \mathrm{dom}(T_j)^{\oplus}$. b) implies that $j \geq m$. Furthermore, due to

$$\mathrm{dom}(C_k) \cap \{e : d \leq e\} \subseteq H_m \subseteq H_{m+1} \subseteq \cdots \subseteq H_n,$$

the trees $T_m, T_{m+1}, \ldots, T_n$ agree on $\mathrm{dom}(C_k) \cap \{e : d \leq e\}$. But the predecessor of d' is contained in $\mathrm{dom}(C_k)$. Thus, we conclude that $d' \in \mathrm{dom}(T_j)^{\oplus}$ for all $j \geq m$.

e) $d' \in H_j^{\oplus}$ for all $j \geq m$ follows directly from $d < d'$, $d' \in \mathrm{dom}(C_k)^+$ and $\mathrm{dom}(C_k) \cap \{e : d \leq e\} \subseteq H_j$.

f) According to the definition of $\rho_{\bar{T}}$ on $D^{\oplus}$, $\rho_{\bar{T}}(d') = \rho_{\bar{C}'}(d')$ where

$$\bar{C}' = (C_{i_1}, C_{i_2}, \ldots, C_{i_{m-1}}, C_{k+1}, C_{k+2}, \ldots, C_{k+m-n}).$$

Hence, this tuple witnesses condition 2f.

3. $d \in \mathrm{dom}(C_k)$ and $d \notin H_m$: due to 2e, $d \in H_m^{\oplus}$ whence we know that $d \in H_m^+$. Furthermore, $d \notin F_{k-1} = \bigcup_{j<k-1}(\mathrm{dom}(C_j)^{\oplus})$. Since H_m is a G^C-tree, we conclude that $d \in \mathrm{dom}(C_{k-1})^+$. Due to $d \in \mathrm{dom}(C_k)$, it follows immediately that

$$\emptyset \neq \mathrm{dom}(C_k)^+ \cap \{e : d < e\} \subseteq G^C.$$

Thus, for all $j > m$, H_j extends H_m along the subtree rooted at d because $H_m <_{G^C} H_j$.

For $m < j \leq n$, this implies $d \in H_j$ whence $\mathrm{dom}(C_k) \cap \{e : d \leq e\} \subseteq H_j$. Hence, $T_{m+1}, T_{m+2}, \ldots, T_n$ agree with $C_{k+1}, C_{k+2}, \ldots, C_{k+n-m}$ on $\mathrm{dom}(C_k) \cap \{e : d \leq e\}$.

Furthermore, Lemma 3.4.19 implies that C_k and T_m agree on $\{e : d \leq e\}$.

Condition 2b implies $d \notin \mathrm{dom}(T_j)$ for $j < m$. By condition 2a, $d \notin \mathrm{dom}(C_{i_j})$ for $j < m$ (recall that the i_j are defined as in condition 2f).

We conclude that $\bar{T}$ and $\bar{C}' := (C_{i_1}, C_{i_2}, \ldots, C_{i_{m-1}}, C_k, C_{k+1}, \ldots, C_{k+n-m})$ agree on $\mathrm{dom}(C_k) \cap \{e : d \leq e\}$.

Due to condition 2f, $\rho_{\bar{T}}(d) = \rho_{\bar{C}}(d) = \rho_{\bar{C}'}(d)$. Thus, setting $\rho_{\bar{T}}(e) := \rho_{\bar{C}}(e)$ for all $e \in D := \mathrm{dom}(C_k)^{\oplus} \cap \{e : d < e\}$ extends $\rho_{\bar{T}}$ in such a way that it is still a partial run.

Note that the maximal elements of $\mathrm{dom}(C_k)^{\oplus} \cap \{e : d \leq e\}$ are the maximal elements of $F_{k+1} \cap \{e : d \leq e\}$. We claim that for each maximal element d' in $\mathrm{dom}(C_k)^{\oplus} \cap \{e : d \leq e\}$ condition 1 or condition 2 with $k+1$ and $m+1$ are satisfied. Again, we prove this claim by case distinction.

For the first case, assume that $d' \notin \mathrm{dom}(\rho_{\bar{T}})$. This implies that $d' \notin \mathrm{dom}(\bigotimes \bar{T})^{\oplus}$ whence by coincidence of $\bigotimes \bar{T}$ and $\bigotimes \bar{C}'$ on all $d \leq e < d'$, it follows that $d' \notin \mathrm{dom}(\bigotimes \bar{C}')^{\oplus}$. Thus, $\bigotimes \bar{C}'$ and $\bigotimes \bar{T}$ agree on the path from d to d' and $\rho_{\bar{C}'}$ is tci on $\bigotimes \bar{C}'$ towards d' because $d' \notin \mathrm{dom}(\bigotimes \bar{C}')$ and $\rho_{\bar{C}'}$ is an accepting run on $\bigotimes \bar{C}'$. But for the coincidence of $\rho_{\bar{C}'}$ and $\rho_{\bar{T}}$ on this path, $\rho_{\bar{T}}$ is then tci on $\bigotimes \bar{T}$ towards d'.

For the second case, assume that $d' \in \mathrm{dom}(\rho_{\bar{T}})$. We show that condition 1 or condition 2 is satisfied; if $m = n$ or $m < n$ and $d' \notin \mathrm{dom}(T_{m+1})^{\oplus}$, condition 1 is satisfied, i.e., the partial run $\rho_{\bar{T}}$ is tci towards d'. Otherwise, we show that condition 2 is satisfied.

Independent of the case we are in, we first show conditions 2a – 2c. These are also helpful when discussing the cases $m = n$ or $d' \notin \mathrm{dom}(T_{m+1})^{\oplus}$.

a) We show that $d' \in \mathrm{dom}(C_k)^{+} \setminus F_k$. $d' \in \mathrm{dom}(C_k)^{+}$ follows from its definition while $d' \notin F_k$ follows from $d < d'$, $d \notin F_{k-1}$ and $d \in \mathrm{dom}(C_{k-1})^{+}$: note that $F_k = F_{k-1} \cup \mathrm{dom}(C_{k-1})^{\oplus}$ and d is by definition a maximal element of this set.

b) $d' \notin \mathrm{dom}(T_j)^{\oplus}$ for all $1 \leq j < m$ because $d < d'$ and $d \notin \mathrm{dom}(T_j)$ for all $1 \leq j < m$ by induction hypothesis. Furthermore, $d' \in \mathrm{dom}(C_k)^{+}$ and T_m agrees with C_k on $\{e : d \leq e\}$. Thus, $d' \in \mathrm{dom}(T_m)^{+}$ whence $d' \notin \mathrm{dom}(T_m)$.

c) $d' \notin H_j^{\oplus}$ for $1 \leq j < m$ because $d \notin H_j^{\oplus}$ for $1 \leq j < m$ and $d < d'$. Since we are in the case $d \notin H_m$, we also have $d' \notin H_m^{\oplus}$ because $d < d'$.

d) In order to satisfy condition 2d, we would have to show that $m+1 \leq n$ and $d' \in \mathrm{dom}(T_{m+1})^{\oplus}$. Instead, we show the following: if this is not the case, then condition 1 is satisfied.

   First assume that $m = n$. We have seen that $d' \notin \mathrm{dom}(T_j)$ for $j \leq m = n$. Since $\bar{T}$ and $\bar{C}'$ coincide on $d \leq e$, $d' \notin \mathrm{dom}(\bigotimes \bar{C}')$ and $\rho_{\bar{C}'}$ is tci on $\bigotimes \bar{C}'$ towards d'. But then $\rho_{\bar{T}}$ is tci on $\bigotimes \bar{T}$ towards d' because $\rho_{\bar{T}}$ and $\rho_{\bar{C}'}$ agree on $d \leq e \leq d'$.

   Now assume that $m < n$ and $d' \notin \mathrm{dom}(T_{m+1})^{\oplus}$. We have already seen that $\mathrm{dom}(C_k) \cap \{e : d \leq e\} \subseteq H_{m+1}$. Hence, the predecessor of d' is in H_{m+1}. Thus, $d' \notin \mathrm{dom}(T_{m+1})^{\oplus}$ implies that either $d' \in H_{m+1}$ and $T_1^C(d') = \Box$ or $d' \notin H_{m+1}$ and $T_2^C(d') = \Box$. Recalling the definitions of T_1^C and T_2^C, we conclude that in the first case there is at most one $j \geq 1$ such that $C_j(d')$ is defined while in the second case there is no $j \geq 1$ such that $C_j(d')$ is defined.

   Heading for a contradiction, assume that there is a $j \in \mathbb{N}$ such that $C_j(d')$ is defined. By a), we know $j > k$. Due to the tree-comb property and because of $d' \in \mathrm{dom}(C_k)^{\oplus}$, we know that C_i and C_j agree on d' for all $k < i < j$. Thus,

we arrive at the contradiction that there are infinitely many j such that $C_j(d')$ is defined.

Thus, $d' \notin \mathrm{dom}(C_j)$ for all $j \geq 1$ whence $d' \notin \mathrm{dom}(T_j)$ for all $1 \leq j \leq n$. But then $\rho_{\bar{C}'}$ is tci on $\bigotimes \bar{C}'$ towards d'. For the coincidence of $\rho_{\bar{C}'}$ and $\rho_{\bar{T}}$ on the path between d and d', $\rho_{\bar{T}}$ is tci on $\bigotimes \bar{T}$ towards d'.

We conclude that either $\rho_{\bar{T}}$ is tci towards d' whence d' satisfies condition 1 or $m < n$ and $d' \in \mathrm{dom}(T_{m+1})^{\oplus}$ whence it satisfies condition 2d for $m+1$. In case that $m < n$ and $d' \in \mathrm{dom}(T_{m+1})^{\oplus}$, we continue by showing that conditions 2e and 2f are also satisfied.

e) $d' \in H_j^{\oplus}$ for $m+1 \leq j \leq n$ follows directly from $d' \in \mathrm{dom}(C_k)^+$ and $\mathrm{dom}(C_k) \cap \{e : d \leq e\} \subseteq H_j$.

f) By the very definition, $\bar{C}'$ is a witness for the claim $\rho_{\bar{T}}(d') = \rho_{\bar{C}'}(d')$ whence condition 2f is satisfied.

This completes the third case. We have shown that one of the following holds:

- $m = n$ and d' satisfies condition 1, i.e., $\rho_{\bar{T}}$ is tci towards d'.
- $m < n$, $d' \notin \mathrm{dom}(T_{m+1})^{\oplus}$ and d' satisfies condition 1.
- $m < n$, $d' \in \mathrm{dom}(T_{m+1})^{\oplus}$ and condition 2 is satisfied for k replaced by $k+1$ and m replaced by $m+1$.

Repeating this inductive definition for all $k \in \mathbb{N}$, we define a partial run $\rho_{\bar{T}}$ on $\mathrm{dom}(\bigotimes \bar{T}) \cap F_k$ for all $k \in \mathbb{N}$. Note that this inductive process terminates at some step because $\bigotimes \bar{T}$ is a finite tree with $\mathrm{dom}(\bar{T}) \subseteq \bigcup_{k \in \mathbb{N}} F_k$. Due to the finiteness of $\bigotimes \bar{T}$, there is some $i \in \mathbb{N}$ such that $\mathrm{dom}(\bar{T}) \subseteq \bigcup_{k=0}^{n} F_k$.

Note that this process stops if and only if all maximal elements of $\mathrm{dom}(\rho_{\bar{T}})$ satisfy condition 1, i.e., $\mathrm{dom}(\rho_{\bar{T}})$ is tci on $\bigotimes \bar{T}$ towards all $d \in \mathrm{dom}(\bigotimes \bar{T})^+$. This is equivalent to the fact that $\rho_{\bar{T}}$ is an run on $\bigotimes \bar{T}$. Since its root is labelled by an accepting state, we have constructed an accepting run of $\mathscr{A}$ on $\bar{T}$ as required by the lemma. □

By now, we have obtained the following result. For each tree-comb C whose pairwise distinct n-tuples are accepted by some automaton $\mathscr{A}$, there is a subcomb C' that is homogeneous with respect to $\mathscr{A}$. Due to homogeneity, all pairwise comparable n-tuples from the closure of C' are accepted by $\mathscr{A}$.

We apply this result in order to prove the correctness of our reduction of the Ramsey quantifier and to prove decidability of the $\mathrm{FO}(\exists^{\mathrm{mod}}, (\mathrm{Ram}^n)_{n \in \mathbb{N}})$-theory of automatic structures. We prove these two facts simultaneously. The correctness of the reduction relies on the fact that every formula induces an automaton that corresponds to this formula. On the other hand, the correctness of the reduction allows the construction of an automaton corresponding to a formula. Thus, we prove both facts by parallel induction. Let us start with an auxiliary lemma that allows to extend the correctness proof for one construction step.

Lemma 3.4.43. Let $\varphi \in \mathrm{FO}(\exists^{\mathrm{mod}}, (\mathrm{Ram}^n)_{n \in \mathbb{N}})$ be a formula such that for each proper subformula ψ of φ, there is an automaton $\mathscr{A}_\psi$ that corresponds to ψ on $\mathfrak{A}$. For each $\bar{a} \in \mathfrak{A}$, $\mathfrak{A}, \bar{a} \models \varphi$ implies $\mathrm{Ext}(\mathfrak{A}), \bar{a} \models \mathrm{red}(\varphi)$.

Proof. Except for the case $\varphi = \mathsf{Ram}^n \bar{x}(\psi)$, the inductive proof of this claim is straightforward (for these cases we even do not need the fact that there is a corresponding automaton for each proper subformula).

Assume that $\varphi = \mathsf{Ram}^n \bar{x}(\psi)$ and assume that there is some $\bar{a} = a_1, a_2, \ldots, a_m \in \mathfrak{A}$ such that $\mathfrak{A}, (a_1, a_2, \ldots, a_m) \models \varphi(y_1, y_2, \ldots, y_m)$. By assumption, we know that there is an automaton $\mathscr{A}_\psi$ corresponding to ψ, i.e., for all $\bar{b} = b_1, b_2, \ldots, b_n \in \mathfrak{A}$,

$$\mathfrak{A}, (b_1, b_2, \ldots, b_n, a_1, a_2, \ldots a_m) \models \psi(\bar{x}, \bar{y}) \text{ if and only if } \mathscr{A}_\psi \text{ accepts } \bigotimes \bar{b} \otimes \bigotimes \bar{a}.$$

Due to $\mathfrak{A}, \bar{a} \models \varphi$, there is an infinite set $S \subseteq \mathfrak{A}$ such that

$$\mathfrak{A}, (b_1, b_2, \ldots, b_n, a_1, a_2, \ldots a_m) \models \psi(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)$$

for all pairwise distinct n-tuples $\bar{b} = b_1, b_2, \ldots, b_n$ from S. By Lemma 3.4.18, there is a tree-comb C' contained in S. This tree-comb contains a subcomb C that is homogeneous with respect to the automata $\mathscr{A}_A$ and $\mathscr{A}_\psi$ where $\mathscr{A}_A$ recognises the domain of $\mathfrak{A}$. By the previous lemma, it follows that (T_1^C, T_2^C, G^C) witnesses

$$\mathsf{Ext}(\mathfrak{A}), (a_1, a_2, \ldots, a_m) \models \mathsf{red}(\varphi)$$

due to the following facts:

1. (T_1^C, T_2^C, G^C) is coherent due to Lemma 3.4.22,
2. (T_1^C, G^C) is small due to Lemma 3.4.25
3. Since C is homogeneous with respect to $\mathscr{A}_A$, the previous lemma shows that $x \in A$ for all $x \in \mathsf{CL}(C)$.
4. Since C is homogeneous with respect to $\mathscr{A}_\psi$, the previous lemma shows that for each pairwise comparable n-tuple $\bar{b}$ from $\mathsf{CL}(C)$,

   $$\mathfrak{A}, \bar{b}, \bar{a} \models \psi(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m).$$

   By induction hypothesis, this implies that

   $$\mathsf{Ext}(\mathfrak{A}), \bar{b}, \bar{a} \models \mathsf{red}(\psi)(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m).$$

   Thus, (T_1^C, T_2^C, G^C) witnesses $\mathsf{red}(\varphi)$ which concludes the proof. □

Using the previous lemma, we can now prove that there is an automata construction corresponding to the Ramsey quantifier.

Lemma 3.4.44. Let $\varphi(\bar{y}) \in \mathsf{FO}(\exists^{\mathsf{mod}}, (\mathsf{Ram}^n)_{n \in \mathbb{N}})$ be a formula with free variables among $\bar{y}$. Then there is an automaton $\mathscr{A}_\varphi$ such that for all $\bar{a} \in \mathfrak{A}$

$$\mathfrak{A}, \bar{a} \models \varphi(\bar{y}) \text{ iff } \mathscr{A}_\varphi \text{ accepts } \bigotimes \bar{a}.$$

Proof. Except for the case $\varphi = \mathsf{Ram}^n \bar{x}(\psi)$, the inductive proof of this claim is a straightforward adaption of the proof of Lemma 2.5.18.

Now, consider the case $\varphi(\bar{y}) = \mathsf{Ram}^n \bar{x}(\psi)$ where $\bar{y} = y_1, y_2, \ldots, y_m$. By induction hypothesis there is an automaton $\mathscr{A}_\psi$ that corresponds to ψ on $\mathfrak{A}$, i.e.,

$$\begin{aligned}
&\text{for all } \bar{a} = a_1, a_2, \ldots, a_m \in \mathfrak{A} \text{ and } \bar{b} = b_1, b_2, \ldots, b_n \in \mathfrak{A},\\
&\mathfrak{A}, \bar{a}, \bar{b} \models \psi(\bar{x}, \bar{y}) \text{ iff}\\
&\mathscr{A}_\psi \text{ accepts } \bigotimes \bar{a} \otimes \bigotimes \bar{b}.
\end{aligned}$$

Due to the soundness of the reduction and due to the previous lemma, this implies that

$$\mathfrak{A}, \bar{a} \models \varphi \text{ if and only if } \mathsf{Ext}(\mathfrak{A}), \bar{a} \models \mathsf{red}(\varphi). \tag{3.8}$$

By Lemma 2.5.14, the ω-automaton $\mathscr{A}_\psi^\infty$ corresponds to $\mathsf{red}(\psi)$ on $\mathsf{Ext}(\mathfrak{A})$ in the sense that for all $\bar{a}, \bar{b} \in \mathfrak{A}$,

$$\begin{aligned}
&\mathsf{Ext}(\mathfrak{A}), \bar{a}, \bar{b} \models \mathsf{red}(\psi)\\
\text{iff } &\mathfrak{A}, \bar{a}, \bar{b} \models \psi\\
\text{iff } &\mathscr{A}_\psi \text{ accepts } \bigotimes \bar{a} \otimes \bigotimes \bar{b}\\
\text{iff } &\mathscr{A}_\psi^\infty \text{ accepts} \bigotimes \bar{a} \otimes \bigotimes \bar{b}.
\end{aligned}$$

Recall that the construction of $\mathsf{red}(\varphi) = \mathsf{red}(\mathsf{Ram}^n \bar{x}(\psi))$ is first-order except for the construction of $\mathsf{red}(\psi)$. Thus, we can use standard techniques in order to construct an ω-automaton $\mathscr{A}_\varphi^\infty$ from $\mathscr{A}_\psi^\infty$ which corresponds to φ on $\mathsf{Ext}(\mathfrak{A})$ in the sense that for all $\bar{a} \in \mathfrak{A}$, $\mathsf{Ext}(\mathfrak{A}), \bar{a} \models \mathsf{red}(\varphi)$ if and only if $\mathscr{A}_\varphi^\infty$ accepts $\bigotimes \bar{a}$.

But now, Lemma 2.5.15 provides the automaton $\mathscr{A}_\varphi := (\mathscr{A}_\varphi^\infty)^{\mathsf{fin}}$. Due to 3.8, we obtain that for all $\bar{a} \in \mathfrak{A}$,

$$\begin{aligned}
&\mathfrak{A}, \bar{a} \models \varphi\\
\text{iff } &\mathsf{Ext}(\mathfrak{A}), \bar{a} \models \mathsf{red}(\varphi)\\
\text{iff } &\mathscr{A}_\varphi^\infty \text{ accepts } \bigotimes \bar{a}\\
\text{iff } &\mathscr{A}_\varphi \text{ accepts } \bigotimes \bar{a}.
\end{aligned}$$

Thus, $\mathscr{A}_\varphi$ corresponds to φ on $\mathfrak{A}$. This concludes our proof. □

Remark 3.4.45. Theorem 3.0.4 is a direct corollary of the previous lemma. Every $\mathsf{FO}(\exists^{\mathsf{mod}}, (\mathsf{Ram}^n)_{n\in\mathbb{N}})$ formula can be effectively translated into a corresponding finite automaton on every given automatic structure. This reduces the model checking problem to the membership problem of regular languages. The latter problem is decidable.

3.4.5 Recurrent Reachability on Automatic Structures

In this section we review To's and Libkin's result [60] on the recurrent reachability problem for automatic structures. Unaware of the concept of word- or tree-combs, they constructed an automaton for the recurrent reachability problem by hand. In fact, they designed an automaton that looks for a tree-comb witnessing recurrent reachability.

We first describe the recurrent reachability problem. Then we show how our method can be adapted to solve this problem. In fact, we only have to replace the role of pairwise comparable tuples by increasing chains. We conclude this section with an application of our results to the decision problem whether a definable partial ordering is a quasi-well-ordering. The recurrent reachability problem is defined as follows.

Definition 3.4.46. Given a starting point p, a relation R and a subset S, decide whether there is an infinite R-path starting at p and reaching S infinitely often.

To and Libkin proved that the recurrent reachability problem is decidable on automatic structures with a regular set S and a transitive, regular relation R. They solve the problem globally, i.e., they construct an automaton that accepts those starting points for which the set S is recurrently reachable.

Theorem 3.4.47 ([60]). Let $\mathfrak{A}$ be an automatic structure with an automatic transitive relation R and let S be a regular subset of its domain A. Then the recurrent reachability problem for R and S is decidable. Moreover, one can effectively construct an automaton $\mathscr{R}(R,S)$ that accepts those nodes p of $\mathfrak{A}$ such that there starts an infinite R path at p that passes S infinitely often. The size of $\mathscr{R}(R,S)$ is polynomially bounded in the size of the automata for R and S.

The proof of To and Libkin gives an explicit construction of $\mathscr{R}(R,s)$. Roughly speaking, this construction yields an automaton corresponding to an existential quantification over a tree-comb whose elements form an R-chain in S.

Our proof can be adapted to reprove the decidability of the recurrent reachability problem in To's and Libkin's setting. If we consider transitive relations, the recurrent reachability problem has solutions of two different types. Either there is an element pRq such that qRq and $q \in S$ or there is an infinite chain $pRq_1Rq_2Rq_3\ldots$ of pairwise distinct elements $q_1, q_2, q_3, \cdots \in S$. The first case is first-order definable whence it is decidable on automatic structures. Hence, we only have to provide a decidability result for the other case. In order to obtain this result, we modify our reduction of the Ramsey quantifier to a reduction of a kind of chain quantifier. Recall that the reduction of a Ramsey quantifier is of the form

$$\mathrm{red}(\mathrm{Ram}^n \bar{x}(\varphi)) := \exists T_1, T_2 \in \text{B-Tree}^{\omega}_{\Gamma_\square}, G \in \text{B-Tree}^{\omega}_{\{0,1\}} \psi_{\mathrm{CoSm}}(T_1, T_2, G) \wedge \psi_{\mathrm{Ram}}(T_1, T_2, G),$$

where

$$\psi_{\mathrm{CoSm}}(T_1, T_2, G) := \mathrm{Coherent}(T_1, T_2, G) \wedge \mathrm{Small}(T_1, G) \wedge \forall x(\mathrm{In}(x, T_1, T_2, G) \rightarrow x \in A)$$

and

$$\psi_{\mathrm{Ram}} := \forall x_1, \ldots, x_n \in \mathrm{Tree}_\Sigma \left(\left(\bigwedge_{1 \leq i \leq n} \mathrm{In}(x_i, T_1, T_2, G) \wedge \bigwedge_{1 \leq i < j \leq n} \mathrm{Comp}(x_i, x_j, T_1, T_2, G) \right) \rightarrow \mathrm{red}(\varphi) \right).$$

In order to solve the recurrent reachability problem, we propose to replace ψ_{Ram} by the formula

$$\forall x_1, x_2 \in \mathrm{Tree}_\Sigma \left(\left(\bigwedge_{i \in \{1,2\}} \mathrm{In}(x_i, T_1, T_2, G) \wedge x_1 <_G x_2 \right) \rightarrow (pRx_1 \wedge x_1 R x_2 \wedge S x_2) \right)$$

Note that $pRx_1 \wedge x_1 R x_2 \wedge S x_2$ is represented by some automaton due to the regularity of S and R. Analogously to the automaton recognising Comp, there is an ω-automaton for "$x_1 <_G x_2$" on input (x_1, x_2, T_1, T_2, G) for all small and coherent triples (T_1, T_2, G).

The formula asserts that there is a closure of some coherent and small tree-comb such that each element a of this closure satisfies the following conditions:

1. a is an R successor of p,
2. a is in S and
3. if there is some a' with $a <_G a'$ then aRa' holds.

A witness (T_1, T_2, G) for this assertion induces an infinite increasing $<_G$ chain in S. Hence, the soundness of this reduction is obvious. For the completeness, we use a tree-comb that is homogeneous with respect to the automaton corresponding to $pRx_1 \wedge x_1 R x_2 \wedge S x_2$. Recall that any witness of the recurrent reachability of S is an infinite set that is linearly ordered by R. Analogously to the fact that any infinite set contains a tree-comb, one proves that any ascending infinite chain contains a tree-comb whose induced order coincides with the order of the chain. Once we have obtained this result, the decidability proof for the recurrent reachability problem is analogous to the decidability proof of the Ramsey quantifier.

Let us conclude this section with an application of our result to partial orderings.

Example 3.4.48. Consider a formula $\varphi(x, y)$ that defines a partial order $\leq$ on some automatic structure $\mathfrak{A}$. Assume that φ is represented by some automaton, e.g., assume that $\varphi \in \mathsf{FO}$. Now, the Ramsey quantifier can be used to formalise the existence of an infinite antichain. Let $\psi(x, y) := \mathrm{Ram}^2 x, y(\neg\varphi(x, y) \wedge \neg\varphi(y, x))$. ψ asserts that there is an infinite set of pairwise $\leq$-incomparable sets, i.e., an infinite antichain. Thus, there is an automaton corresponding to the assertion that $\leq$ does not contain an infinite antichain.

We can also construct effectively an automaton that decides whether $<$ contains an infinite descending chain. This is the same as deciding whether there is some point p for which $>$ satisfies the recurrent reachability problem with respect to the full domain of the structure.

If there is neither an infinite antichain nor an infinite descending chain, $\leq$ is a well-quasi-ordering. Thus, if φ defines a partial order, the statement $\mathsf{WQO}(\varphi) :=$"φ induces a well-quasi-ordering" is decidable on automatic structures. Furthermore, one can effectively construct an automaton that corresponds to $\mathsf{WQO}(\varphi)$.

4 Conclusions

In the following we summarise the main results of this thesis and relate these results to open problems.

We have shown that $\mathsf{FO}(\mathrm{Reg})$ model checking on level 2 collapsible pushdown graphs is decidable and Broadbent showed that first-order model checking on level 3 is undecidable (even with fixed formula or fixed graph). The positive result on level 2 is in fact even stronger: the extensions by regular reachability, Ramsey quantifiers and $L\mu$-definable predicates is still decidable. Hence, the structures in level 3 of the hierarchy are much more complicated than those structures in level 2. But it is still an open question what the reason for this difference is. Broadbent's results point out that even a very weak use of collapse operations already turns the first-order model checking undecidable on level 3. It would be nice to clarify which structural difference between the graphs of level 2 and those of level 3 provokes the rather big difference in the model checking results. Another direction of further research is the question for extensions of our results. What is the largest fragment of MSO that is decidable on collapsible pushdown graphs of level 2?

We introduced the new hierarchy of higher-order nested pushdown trees and provided first-order model checking algorithms for the first two levels of this hierarchy. Due to its similarity to a subclass of collapsible pushdown graphs, we also conjecture that the $L\mu$ model checking is decidable on this hierarchy. But the proof of this conjecture is still open. Another open question concerns first-order model checking on levels $3, 4, 5, \ldots$ in this new hierarchy. Our approach on level 2, i.e., the use of the analysis of strategies in the Ehrenfeucht-Fraïssé game via the notion of relevant ancestors is extendable to higher levels. But on higher levels, we miss an analysis of "higher-order loops" in analogy to our results for loops of collapsible pushdown systems of level 2. Further research is necessary in order to clarify whether FO model checking on higher-order nested pushdown trees is decidable.

Focusing on the second level of the nested pushdown tree hierarchy, we are still lacking a characterisation of the complexity of first-order model checking on nested pushdown trees of level 2. Is there another approach that yields an elementary complexity? Can we derive any reasonable lower bound for the first-order model checking on nested pushdown trees? We already know that first-order with reachability model checking has nonelementary complexity on nested pushdown trees.

Another more general question concerning first-order model checking and collapsible pushdown graphs is the classification of those graphs in the hierarchy that have decidable first-order theories. What kind of restrictions can one impose on the transition relation of a collapsible pushdown graph in order to obtain decidability of its first-order theory?

Another open question concerns the characterisation of the differences between collapsible pushdown graphs and higher-order pushdown graphs. We propose the further study of higher-order nested pushdown trees in order to approach this question. Higher-order nested pushdown trees can be seen as collapsible pushdown graphs with a rather tame application of collapse.

We now turn to a more general direction of research. In this thesis, we have focused on what is called the local model checking. Global model checking on the other hand asks for identifying all elements in a given structure that satisfy some formula. Recently,

Broadbent et al. [13] showed the global $L\mu$ model checking on collapsible pushdown graphs to be decidable. Furthermore, they showed that collapsible pushdown graphs themselves are sufficient to describe the result of the global $L\mu$ modal checking in the following sense: for each formula and each pushdown system there is another one that generates the same graph but marks each element that satisfies the given formula. The analogous questions for first-order model checking on (higher-order) nested pushdown trees or collapsible pushdown graphs have not been investigated yet and their investigation may reveal interesting insights into these classes.

We have also shown that Ramsey quantifiers on tree-automatic structures are decidable. This extends the corresponding result and proof techniques for the string-automatic case. But in fact, on string automatic structure a far stronger logic is decidable[45]. Kuske called this logic FSO. It is the extension of FO by existential quantification over infinite relations that only occur negatively, i.e., under the scope of an odd number of negations. Ramsey quantifiers can be rewritten in terms of FSO. Thus, our result shows the decidability of a fragment of FSO on tree-automatic structures. It is an open problem whether FSO is decidable on all tree-automatic structures or whether some undecidable problem may be encoded into FSO on some tree-automatic structure. For most results on string-automatic structures there has been found an analogous one for tree-automatic structures. If this were not the case for FSO model checking this may point to new insights into the difference between tree-automata and string-automata.

A Undecidability of $\mathbf{L}\mu$ on the Bidirectional Half-Grid

In this appendix, we show the undecidability of $L\mu$ on the bidirectional $\mathbb{N} \times \mathbb{N}$-grid, i.e., the grid $\mathbb{N} \times \mathbb{N}$ with modalities "left", "right", "up", and "down" (denoted by $\leftarrow$, $\rightarrow$, $\uparrow$, $\downarrow$). Note that we do not allow atomic propositions apart from True and False. The proof is by reduction to the halting problem of Turing machines. At the end, we will see how this proof generalises to the case of the bidirectional half-grid $\mathfrak{H}$ (cf. Figure 2.1). Before we start the proof, we will shortly recall our notation concerning Turing machines and recall the definition of the halting problem.

A.1 Turing Machines

In order to fix notation, we briefly recall the notion of a Turing machine.

Definition A.1.1. The tuple $\mathscr{M} = (Q, \Sigma, q_I, q_F, \Delta)$ with Q the finite sets of states, Σ the finite tape alphabet, $q_I, q_F \in Q$ the initial, respectively, final state, and

$$\Delta : Q \times \Sigma \to \Sigma \times \{l, r\} \times Q$$

a transition function is called a Turing machine. The set of configurations of $\mathscr{M}$ is

$$\mathsf{CONF} := \Sigma^\omega \times \mathbb{N} \times Q.$$

Remark A.1.2. The elements in $\{l, r\}$ are called head instructions where l denotes "move to the left" and r denotes "move to the right".

In the literature this definition is normally called a deterministic Turing machine, while in the general nondeterministic case Δ is assumed to be a relation instead of a function. We restrict ourselves to deterministic Turing machines because they have the same computational power as nondeterministic ones (cf. [33]).

The notion of the computation of a Turing machine is captured by the notion of a run of the machine. A run is a list of configurations where the $(n+1)$-st configuration evolves from the n-th by applying Δ.

Definition A.1.3. Let $M = (Q, \Sigma, q_I, q_F, \Delta)$ be a Turing machine. For $w \in \sigma^\omega$ we write $w(i)$ for the i-th letter in w. M induces a function $\vdash\colon \mathsf{CONF} \to \mathsf{CONF}$ as follows. Let (w, i, q) and (w', i', q') in CONF. Assume that $\Delta(q, w(i)) = (\sigma', o, q')$. It holds that $(w, i, q) \vdash (w', i', q')$ if

1. $w'(j) = w(j)$ for all $i \neq j \in \mathbb{N}$,
2. $w'(i) = \sigma'$, and
3. $i' = i - 1$ if $o = l$ and $i' = i + 1$ if $o = r$.

A run of M on input $w \in \Sigma^*\{0\}^\omega$ is a function $\rho : \mathbb{N} \to \mathsf{CONF}$ such that $\rho(0) = (w, 1, q_i)$ and $\rho(i) \vdash \rho(i+1)$ for all $i \in \mathbb{N}$.

We say that the computation of M on w terminates if there is some $i \in \mathbb{N}$ such that $\rho(i) = (w', i', q_F)$ for ρ the run of M on input w.

For a detailed introduction into the theory of Turing machines we recommend [33]. For the purpose of this proof, we only need one of the cornerstones of computability theory: the undecidability of the halting problem. The halting problem is the problem whether the computation of a given Turing machine terminates on input 0^ω. This is one of the classical examples of undecidable problems.

Theorem A.1.4 ([61]). The halting problem is undecidable.

A.2 Reduction to the Halting Problem

For simplicity, we only consider Turing machines with tape alphabet $\Sigma = \{0,1\}$. We assume that the state set is $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$. We will represent the run of an arbitrary fixed Turing machine on input 0^ω as an $L\mu$-definable colouring of the bidirectional grid. Then using an $L\mu$ definable reachability query for a final state of the Turing machine, the halting problem is reduced to $L\mu$ model-checking on the bidirectional grid. The idea is as follows.

Each configuration $c = (w, p, q)$ of a Turing machine M can be encoded as an infinite bitstring $v \in \{0,1\}^\omega$. We use the letters $v((|Q|+3)\cdot i), \ldots, v((|Q|+3)\cdot(i+1)-1)$ to encode the information concerning the i-th cell of c. We use the last two bits of the representation of a cell to indicate whether the corresponding cell contains the letter 0 or 1. Thus, we define $v((|Q|+3)\cdot(i+1)-1) := 1$ if and only if $w(i) = 1$ and $v((|Q|+3)\cdot(i+1)-2) := 1$ if and only if $w(i) = 0$. The other $|Q|+1$ bits are used to encode the information about the position of the head p and the state of the machine q. We set the first bit of a cell to 1 if the head is not above this cell and we set the j-th bit of a cell, if the head is above this cell and $q = q_{j-1}$. Formally, we set $v(k) := 1$ if $k = (|Q|+3)\cdot i$ for some $i \neq p$ or $k = (|Q|+3)\cdot p + j$ for $q = q_j$. All other positions in v are set to 0. We denote as $W : \mathsf{CONF} \to \{0,1\}^\omega$ the function that translates each configuration into the corresponding encoding.

Now, it is easy to encode the run ρ of M on 0^ω in the infinite $\mathbb{N} \times \mathbb{N}$-grid using a set $X_M \subseteq \mathbb{N} \times \mathbb{N}$. We write $c_n := \rho(n)$ and $v_n := W(c_n)$. We set $(i,j) \in X_M$ if and only if $v_i(j) = 1$.

As a next step, we show that X_M is definable in $L\mu$. Using this fact, we later define an $L\mu$ formula that is true on the infinite grid if and only if ρ terminates.

Lemma A.2.1. There is an effective translation from a given Turing machine M into an $L\mu$ formula φ_M such that φ_M defines X_M on the infinite grid.

Proof. As a preliminary step, we want to define those $(i,j) \in \mathbb{N} \times \mathbb{N}$ where the encoding of one of the cells start, i.e., the set $\{i, j : \exists k \in \mathbb{N}\ j = k \cdot (|Q|+3)\}$.

This is done by the formula

$$\mathsf{Cells} := \mu X.\left(([\uparrow]\mathsf{False} \wedge [l]\mathsf{False}) \vee \langle\uparrow\rangle^{|Q|+3} X \vee \langle\leftarrow\rangle X\right).$$

The first part of this formula defines the position $(0,0)$, the second part adds the position $(i, j+|q|+3)$ to Cells for each $(i,j) \in \mathsf{Cells}$ and the last part adds $(i+1, j)$ to Cells for $(i,j) \in \mathsf{Cells}$. We call a node in Cells initial position of an encoding of a cell, or simply initial position of a cell. In the following we identify the initial position of a cell with the cell itself.

In the following, we use some auxiliary formulas:

- We set $\mathsf{Pos}_i := \langle\uparrow\rangle^i \mathsf{Cells}$ for $0 \leq |Q| + 2$. Pos_i holds on (j,k) if (j,k) is the $i+1$-st bit of the encoding of one of the cells.

- For $\varphi \in L\mu$, we write $\mathsf{From}_0(\varphi)$ for the formula $\bigvee_{i=0}^{|Q|+2} \left(\mathsf{Pos}_i \wedge \langle\uparrow\rangle^i \varphi\right)$. Some node (i,j) satisfies $\mathsf{From}_0(\varphi)$ if and only if the initial position corresponding to the same cell as (i,j) satisfies φ.

- We need formulas for navigation on the cells in encoded form. We set

$$\mathsf{Left}(\varphi) := \mathsf{From}_0(\langle\uparrow^{|Q|+3}\rangle\varphi) \text{ and}$$
$$\mathsf{Right}(\varphi) := \mathsf{From}_0(\langle\downarrow\rangle^{|Q|+3}\varphi).$$

  These formulas are satisfied at (i,j) if "the cell to the left of the one corresponding to (i,j) satisfies φ", respectively "the cell to the right ...". Similarly we use $\mathsf{Before}(\varphi) := \mathsf{From}_0(\langle\leftarrow\rangle\varphi)$. This formula is satisfied by node whose cell satisfied φ in the preceding configuration.

As a next step we introduce some formulas that recover information about the configuration from its encoding in the grid. For this we assume that X is a colouring that colours each column of the grid with the encoding of some configuration.

- For $q_i \in Q$ set $\mathsf{State}_{q_i} := \mathsf{From}_0(\langle\downarrow\rangle^i X)$. This formula is satisfied at (i,j) if the head of M is above the corresponding cell and the state of M is q_i. Analogously, we write $\mathsf{State}_0 := \mathsf{From}_0(X)$ for the formula specifying the head is not at the cell corresponding to this position.

- For $\sigma \in \{0,1\}$ we set $\mathsf{Tape}_\sigma := \mathsf{From}_0(\langle\downarrow\rangle^{|Q|+1+\sigma} X)$. Tape_σ is satisfied if the corresponding cell contains the letter σ.

Since we want to generate the encoding of the run of M on input 0^ω by a fixpoint formula in $L\mu$, we have to define "update-formulas" which, given the encoding of a valid configuration in the i-th column, return the encoding of the following configuration in the $i+1$-st column.

Let us first consider the information concerning the update of the tape. There are two possibilities for each cell: either the head is not at this cell, then its value is preserved by $\vdash$, or the head is at this cell, then its value depends on $\Delta(q,\sigma)$ where q is the state of the machine and σ is the symbol of this cell. For $\sigma \in \Sigma$, set

$$Set_\sigma := \{(\tau, q_k) \in \Sigma \times Q : \Delta(q_k, \tau) = (\sigma, d, q), d \in \{l, r\}, q \in Q\}.$$

Note that Set_1 contains those combinations of a letter σ and a state q such that the head of M will write 1 onto the tape if it is in state q and reads σ. The analogous claim holds for Set_0. Thus,

$$\mathsf{Update}_\sigma := \mathsf{Pos}_{|Q|+1+\sigma} \wedge \left(\mathsf{Before}(\mathsf{State}_0 \wedge \mathsf{Tape}_\sigma) \vee \bigvee_{(i,q)\in Set_\sigma} \mathsf{Before}(\mathsf{State}_q \wedge \mathsf{Tape}_i)\right)$$

are the correct update formulas for the information concerning the tape.

The update for the information concerning the head of the Turing machine are slightly more complicated because the head can reach a certain cell either from the cell to the left or

from the cell to the right. Furthermore, we also have to update the information on those cells where the head is not positioned. There, we have to set the first bit encoding the cell which represents the absence of the head from this cell. We start by collecting those combinations of states and letters that induce the head to move to the left or to the right, respectively. For $q_j \in Q$, set

$$\mathrm{Left}_{q_j} := \{(\sigma, q) \in \Sigma \times Q : \exists \tau \in \Sigma\ \Delta(q, \sigma) = (\tau, l, q_j)\}, \text{ and}$$
$$\mathrm{Right}_{q_j} := \{(\sigma, q) \in \Sigma \times Q : \exists \tau \in \Sigma\ \Delta(q, \sigma) = (\tau, r, q_j)\}.$$

Now, we can use these sets to define the position of the head in the next configuration. We set

$$\mathsf{Update}_{q_j} := \mathrm{Pos}_j \wedge$$
$$\Big(\bigvee_{(\sigma,q)\in \mathrm{Left}_{q_j}} \mathsf{Before}(\mathrm{Right}(\mathrm{State}_q \wedge \mathrm{Tape}_\sigma)) \vee \bigvee_{(\sigma,q)\in \mathrm{Right}_{q_j}} \mathsf{Before}(\mathrm{Left}(\mathrm{State}_q \wedge \mathrm{Tape}_\sigma))\Big).$$

These formulas update correctly the information on the head of the tape, i.e., if X encodes some configuration in the i-th column of the grid, then Update_{q_k} will hold at some $(i+1, k)$ if and only if the next configuration has state q_k, $(i+1, k)$ corresponds to the encoding of state q_k in some cell, and the head is at this cell in the next configuration. Of course, we also have to propagate the "no-head" information along all other cells. This is the case if either the head is neither to the left nor to the right in the previous configuration or it is positioned one step to the left, respectively right, but will move further left, respectively right. Note that by the definition of a Turing machine, the sets $(\mathrm{Left}_q)_{q\in Q}$ and $(\mathrm{Right}_q)_{q\in Q}$ form a partition of $\Sigma \times Q$. Hence, the following formula updates the "no-head" information:

$$\begin{aligned}\mathsf{Update}_0 := \big(\mathrm{Pos}_0 \wedge \mathsf{Before}(\mathrm{Left}(\mathrm{State}_0)) \wedge \mathsf{Before}(\mathrm{Right}(\mathrm{State}_0))\big) \\ \vee \bigvee_{(\sigma,q)\in \mathrm{Left}_{q_j}} \mathsf{Before}(\mathrm{Left}(\mathrm{State}_q \wedge \mathrm{Tape}_\sigma)) \\ \vee \bigvee_{(\sigma,q)\in \mathrm{Right}_{q_j}} \mathsf{Before}(\mathrm{Right}(\mathrm{State}_q \wedge \mathrm{Tape}_\sigma)).\end{aligned}$$

The first part of this formula deals with positions where the head in the previous configuration is not one step to the left or to the right and the second and third part update the "no-head" information if the head is close but is going to move further away.

Having defined the necessary formulas for the update from one configuration to the next, we have to define the colouring of the initial configuration of the run of M on input 0^ω in order to obtain the encoding of the full run by a least fixpoint induction. For this purpose, we assume that $q_I = q_1$ and we set

$$\mathsf{Init} := \langle\leftarrow\rangle \mathrm{False} \wedge (\mathrm{Pos}_{|Q|+1} \vee (\mathrm{Left}(\mathrm{True}) \wedge \mathrm{Pos}_0) \vee (\mathrm{Left}(\mathrm{False}) \wedge \mathrm{Pos}_1)).$$

Init holds only at positions on the leftmost column of the grid which means that it only initialises the encoding of the first configuration. The first part sets in each cell the position

$|Q|+1$ which corresponds to setting all cells to 0. Secondly, we set in all but the first cell the "no-head" information. Finally, we set the state $q_I = q_1$ in the first cell. Hence, Init is the definition of the encoding of the first configuration of the run of M on input 0^ω.

We claim that the formula

$$\varphi_M := \mu X.(\mathsf{Init} \vee \bigvee_{q \in Q} \mathsf{Update}_q \vee \mathsf{Update}_0 \vee \bigvee_{\sigma \in \Sigma} \mathsf{Update}_\sigma)$$

defines the encoding of the run of M on input 0^ω on the bidirectional grid. In fact, an easy but technical induction shows that the n-th stage of the fixpoint of φ_M defines exactly the encoding of the first n configurations of the run of M on input 0^ω. □

From the translation of runs of Turing machines into $L\mu$ formulas on the grid, the undecidability of the $L\mu$ model checking on the grid follows immediately.

Lemma A.2.2. $L\mu$ model checking is undecidable on the bidirectional grid.

Proof. By reduction to the halting problem: Deciding the halting problem for M is the same as deciding whether φ_M defines some cell where State_{q_F} holds. But this is the same as deciding whether

$$\mathsf{Halting} := \mu Y.\mathsf{State}_{q_F}(\varphi_M) \vee \langle\downarrow\rangle Y \vee \langle\rightarrow\rangle Y$$

is satisfied in the position $(0,0)$ of the bidirectional grid. Thus, a model checking algorithm of $L\mu$ on the bidirectional grid would lead to a decision procedure for the halting problem. This proves the undecidability of $L\mu$ on the grid. □

Remark A.2.3. In the presence of a universal modality, the proof can be adapted to show the undecidability of the $\mathbb{N} \times \mathbb{N}$ grid only with the modalities left and up. The search for a cell with state q_F can be done by using the universal modality. Hence, we only have to remove the down modalities from the update formulas. This can be achieved by shifting the beginning of the encoding of the i-th column by $2i(|Q|+3)$.

Having obtained the undecidability of $L\mu$ on the grid, we want to refine the result such that it applies to the half-grid. But this is easy by noting that the head of the Turing machine in the i-th configuration of a run can only have visited the first i-cells of the tape.

Corollary A.2.4. $L\mu$ on the bidirectional half-grid is undecidable.

Proof. Instead of encoding the i-th configuration of the run of M in the i-th row, we can use the $i(|Q|+|\Sigma|+1)$-st row instead. Doing this, the head of the Turing machine is in all configurations at some cell which is encoded by elements in the grid of the form (i,j) where $i > j$. Thus, we can treat the missing nodes in the half-grid $\{(i,j) \in \mathbb{N} \times \mathbb{N} : i > j\}$ as the encodings of cells which contain the symbol 0 and which contain the "no-head" marker. □

Bibliography

[1] K. Aehlig, J. G. de Miranda, and C.-H. L. Ong. Safety is not a restriction at level 2 for string languages. Technical Report RR-04-23, Oxford University Computing Laboratory, October 2004.

[2] R. Alur, S. Chaudhuri, and P. Madhusudan. Languages of nested trees. In Proc. 18th International Conference on Computer-Aided Verification, volume 4144 of LNCS, pages 329–342. Springer, 2006.

[3] Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase-structure grammars. Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung, 14(2):143–172, 1961.

[4] V. Bárány, E. Grädel, and S. Rubin. Automata-based presentations of infinite structures. Submitted for publication.

[5] V. Bárány, Ł. Kaiser, and A. Rabinovich. Cardinality quantifiers in MLO over trees. In CSL 2009, volume 5771 of LNCS, pages 117–132. Springer, 2009.

[6] P. Blackburn, F. Wolter, and J. van Benthem, editors. Handbook of Modal Logic. Elsevier, 2007.

[7] A. Blumensath. Automatic structures. Diploma thesis, RWTH Aachen, 1999.

[8] A. Blumensath. On the structure of graphs in the Caucal hierarchy. Theoretical Computer Science, 400:19–45, 2008.

[9] A. Blumensath, T. Colcombet, and C. Löding. Logical theories and compatible operations. In J. Flum, E. Grädel, and T. Wilke, editors, Logic and automata: History and Perspectives, pages 72–106. Amsterdam University Press, 2007.

[10] A. Blumensath and E. Grädel. Automatic structures. In Proc. 15th IEEE Symp. on Logic in Computer Science, pages 51–62. IEEE Computer Society Press, 2000.

[11] A. Blumensath and E. Grädel. Finite presentations of infinite structures: Automata and interpretations. Theory of Computing Systems, 37:641 – 674, 2004.

[12] C. H. Broadbent. Private communication. September 2010.

[13] C. H. Broadbent, A. Carayol, C.-H. Luke Ong, and O. Serre. Recursion schemes and logical reflection. In LICS, Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, pages 120–129, 2010.

[14] J. R. Büchi. Regular canonical systems. Archiv für mathematische Logik und Grundlagenforschung, (6):91–111, 1964.

[15] A. Carayol. Regular sets of higher-order pushdown stacks. In MFCS 05, pages 168–179, 2005.

[16] A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2003, volume 2914 of LNCS, pages 112–123. Springer, 2003.

[17] D. Caucal. On infinite terms having a decidable monadic theory. In MFCS 02, pages 165–176, 2002.

[18] E. M. Clarke and E. Allen. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Logic of Programs, pages 52–71, 1981.

[19] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 2007. release October, 12th 2007.

[20] B. Courcelle. Graph rewriting: an algebraic and logic approach. In Jan van Leeuwen, editor, Handbook of theoretical computer science (vol. B), pages 193–242. MIT Press, 1990.

[21] J. Doner. Tree acceptors and some of their applications. J. Comput. Syst. Sci., 4(5):406–451, 1970.

[22] A. Ehrenfeucht. An application of games to the completeness problem for formalized theories. Fund. Math., 49:129–141, 1960/1961.

[23] J. Ferrante and C.W. Rackoff. The Computational Complexity of Logical Theories. Springer-Verlag, Berlin, 1979.

[24] R. Fraïssé. Sur quelques classifications des systèmes de relations. Publications Scientifiques de l'Université d'Alger, série A, 1:35–182, 1954.

[25] H. Gaifman. On local and nonlocal properties. In Proceedings of the Herbrand symposium (Marseilles, 1981), volume 107 of Stud. Logic Found. Math., pages 105–135. North-Holland, Amsterdam, 1982.

[26] E. Grädel. Simple Interpretations among Complicated Theories. Information Processing Letters, 35:235–238, 1990.

[27] M. Hague, A. S. Murawski, C-H. L. Ong, and O. Serre. Collapsible pushdown automata and recursion schemes. In LICS '08: Proceedings of the 2008 23rd Annual IEEE Symposium on Logic in Computer Science, pages 452–461, 2008.

[28] W. P. Hanf. Model-theoretic methods in the study of elementary logic. In J.W. Addison, L. Henkin, and A. Tarski, editors, The Theory of Models. North-Holland, Amsterdam, 1965.

[29] David Harel. Towards a theory of recursive structures. In Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors, STACS, volume 775 of LNCS, pages 633–645. Springer, 1994.

[30] Takeshi Hayashi. On derivation trees of indexed grammars. Publ. RIMS, Kyoto Univ., 9:61–92, 1973.

[31] B. R. Hodgson. On direct products of automaton decidable theories. Theor. Comput. Sci., 19:331–335, 1982.

[32] B.R. Hodgson. Décidabilité par automate fini. Ann. sc. math. Québec, 7(1):39–57, 1983.

[33] J. E. Hopcroft and J. D. Ullman. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Massachusetts, 1979.

[34] A. Kartzow. FO model checking on nested pushdown trees. In MFCS 09, volume 5734 of LNCS, pages 451–463. Springer, 2009.

[35] A. Kartzow. Collapsible pushdown graphs of level 2 are tree-automatic. In STACS 10, volume 5 of LIPIcs, pages 501–512. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

[36] B. Khoussainov and M. Minnes. Model-theoretic complexity of automatic structures. Ann. Pure Appl. Logic, 161(3):416–426, 2009.

[37] B. Khoussainov and A. Nerode. Automatic presentations of structures. In LCC, pages 367–392, 1994.

[38] B. Khoussainov, A. Nies, S. Rubin, and F. Stephan. Automatic structures: Richness and limitations. Logical Methods in Computer Science, 3(2), 2007.

[39] B. Khoussainov and S. Rubin. Graphs with automatic presentations over a unary alphabet. Journal of Automata, Languages and Combinatorics, 6(4):467–480, 2001.

[40] B. Khoussainov, S. Rubin, and F. Stephan. Definability and regularity in automatic structures. In Volker Diekert and Michel Habib, editors, STACS, volume 2996 of LNCS, pages 440–451. Springer, 2004.

[41] T. Knapik, D. Niwinski, and P. Urzyczyn. Higher-order pushdown trees are easy. In FOSSACS'02, volume 2303 of LNCS, pages 205–222. Springer, 2002.

[42] B. Knaster. Un théorème sur les fonctions d'ensembles. Annales de la Société Polonaise de Mathématiques, 6:133–134, 1928.

[43] N. Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In POPL 09, pages 416–428. ACM, 2009.

[44] D. Kuske. Is Cantor's theorem automatic? In LPAR, pages 332–345, 2003.

[45] D. Kuske. Theories of automatic structures and their complexity. In CAI'09, Third International Conference on Algebraic Informatics, volume 5725 of LNCS, pages 81–98. Springer, 2009.

[46] D. Kuske, J. Liu, and M. Lohrey. The isomorphism problem on classes of automatic structures. In LICS, Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, pages 160–169, 2010.

[47] D. Kuske and M. Lohrey. Automatic structures of bounded degree revisited. In CSL, pages 364–378, 2009.

[48] C. I. Lewis. A Survey of Symbolic Logic. University of California Press, 1918.

[49] P. Lindstrom. First order predicate logic with generalized quantifiers. Theoria 32, pages 186–195, 1966.

[50] M. Magidor and J. Malitz. Compact extensions of L(Q). Annals of Mathematical Logic, 11:217–261, 1977.

[51] R. McNaughton and S. A. Papert. Counter-Free Automata (M.I.T. research monograph no. 65). The MIT Press, 1971.

[52] A. Mostowski. On a generalization of quantifiers. Fundamenta Mathematicae, 44:12–36, 1957.

[53] D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. Theor. Comput. Sci., 37:51–75, 1985.

[54] E. L. Post. A variant of a recursively unsolvable problem. Bulletin of the American Mathematical Society, 52:264–268, 1946.

[55] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. Transaction of the American Mathematical Society, 141:1–35, 1969.

[56] F. P. Ramsey. On a problem of formal logic. Proc. London Math. Soc. Series 2, 30:264 – 286, 1930.

[57] S. Rubin. Automata presenting structures: A survey of the finite string case. Bulletin of Symbolic Logic, 14(2):169–209, 2008.

[58] L. J. Stockmeyer. The Complexity of Decision Problems in Automata Theory and Logic. PhD thesis, MIT, Cambridge, Massasuchets, USA, 1974.

[59] J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. Mathematical Systems Theory, 2(1):57–81, 1968.

[60] A. W. To and L. Libkin. Recurrent reachability analysis in regular model checking. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, LPAR, volume 5330 of LNCS, pages 198–213. Springer, 2008.

[61] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 2(42):230–265, 1936.

[62] J. A. Väänänen. Generalized quantifiers. Bulletin of the EATCS, 62, 1997.

[63] I. Walukiewicz. Pushdown processes: games and model checking. Information and Computation, 164:234–263, 2001.

Wissenschaftlicher Werdegang von Alexander Kartzow,
geboren am 12. Januar 1983 in Gießen

Schulabschluss	
Juni 2002	Abitur
Studium	
2002-2007	Studium der Mathematik mit Nebenfach Informatik an der TU Darmstadt
2005-2006	Studium der Mathematik und der Informatik an der Universidad de Salamanca, Spanien
November 2007	Diplom in Mathematik
Promotion	
2007 - 2011	Promotionsstudium an der TU Darmstadt