Reasoning with Separation Logics: Complexity, Expressive Power, Proof Systems

Alessio Mansutti

Dissertation abstract

Program verification and separation logic

Developing verification techniques that can be applied to real-life software is very hard:

- programmers cannot help with the verification process, as software is built collectively and each programmer only knows a fraction of it,
- the software is huge (millions of lines of code) and evolves daily,
- the data structures used in industrial-scale languages (e.g. `Java`, `C`, `Python`) are complex.

Because of these properties of industrial software, their verification can only be completely *automatic* and *modular*. Automaticity means that we cannot hope to show that a program is semantically correct (we do not know its semantics in the first place), and we should rather focus on verifying the absence of low-level implementation bugs that can cause severe security issues. In this sense, a fundamental contribution is given by the MITRE's community project *Common Weakness Enumeration* [1], which helps categorize software and hardware flaws, and estimates their severity. A quick survey of the enumeration show that memory mismanagement bugs are among the most common and severe types of vulnerabilities. This led to a quest for automatic verification of memory properties of programs, that became feasible with the introduction of separation logics.

Separation logics [19] are a family of logics developed to reason on heap-manipulating programs (both sequential and concurrent ones), in a modular way. This modularity allows the logics to scale to millions of lines of code, which led to several tools based on separation logics to be applied in industrial-scale projects. For instance, both the verification tools INFER (Facebook) and SLAyer (Microsoft Research) are based on separation logics. Most of the type system of the `Rust` programming language is verified using separation logics (thanks to the Iris framework).

To achieve its modularity, separation logics applies in a fundamental way the developments on resource reasoning given by the *logic of Bunched Implication* (BI) introduced by Pym and O'Hearn [18]. The essential features of BI are given by its *multiplicative connectives*, such as the *separating conjunction* $*$. This operator roughly stands for "and, separately", and is essentially the multiplicative conjunction of linear logic induced from the standard (additive) conjunction by removing the structural proof rules of weakening and contraction. A formula $\varphi * \psi$ of BI is satisfied by a resource r if r can be partitioned into two pieces, one satisfying φ and the other satisfying ψ. Separation logic specialises BI by considering an interpretation on *memory states*, i.e. abstractions of the stack and heap/RAM memory model used as a backbone for the semantics of many programming languages (e.g. `Java` and `C`). Thanks to this specific heap/RAM model and the multiplicative connectives of BI, separation logic makes scalable reasoning on heap-manipulating programs possible.

Several computational and proof theoretical issues arise from the interpretation on memory states and the presence of the multiplicative connectives. For instance, the model-checking and satisfiability problem of the two-variable fragment of first-order separation logic are both non-recursively enumerable [8], instead of being decidable respectively in PSpace and NExpTime, as in the case of standard first-order logic [13]. Because of this, industrial tools that use separation logic only handle a very weak fragment of the full logic, known as the *symbolic-heap fragment*.

This thesis

The thesis is divided in three parts, and while each part has its own goals, the overall objective of the thesis is to go beyond the symbolic heap fragment, and design new algorithms and proof systems for classical decision problems (e.g. satisfiability and validity) of very expressive separation logics. In working towards our main goal, we significantly advance our understanding of separation logic on a theoretical level, and draws interesting connections between separation logic and other logics, most notably modal and temporal logics. As such, the work presented in the thesis should interest a wide range of researchers and students working in program verification, proof theory, mathematical and computational logic and finite model theory.

The first part focuses on separation logics featuring reachability predicates. A location ℓ_1 (a.k.a. a memory address) is said to be reachable from a second location ℓ_2 whenever subsequent dereferentiations of ℓ_2 yield ℓ_1. Together with a single first-order variable, reachability predicates allow to check for several *robustness properties* of the memory that are essential in the context of heap-manipulating programs, such as acyclicity (i.e. the absence of cycles) and garbage freedom (i.e. the absence of dangling pointers). Our goal is to devise a separation logic featuring reachability predicates that is able to check for these properties, while keeping the complexity of its satisfiability and validity/entailment problems in check. This is easier said than done, as we show that several quantifier-free separation logics are already largely intractable (undecidable/non-recursively enumerable). Nonetheless, after studying the sources of intractability we manage to design a PSPACE fragment of first-order separation logic that is able to express the robustness properties. At the time of writing, this logic is the most expressive fragment of separation logic whose classical decision problems can be solved in PSPACE and, in particular, no other elementary-decidable separation logic is able to express the garbage freedom and acyclicity properties. The PSPACE membership is proved through the *core formulae technique*, a model theoretic approach that is reminiscent of Gaifman's locality theorem for first-order logic [10], and that is here made modular thanks to a methodology that we call *game hopping*, as it is inspired by the homonymous technique for indistinguishability games in computational security [3].

The second part of the thesis tackles the open problem of the existence of Hilbert-style proof systems for separation logics. An axiomatisation is said to be Hilbert-style (or internal) whenever its axioms and inference rules are built solely from formulae of the logic, without any external machinery such as labels or nominals. While Hilbert-style proof systems for BI exists, none of them is complete with respect to the interpretation on memory states. In fact, the main technical challenge in deriving an Hilbert-style proof system for separation logic *is* the axiomatisation of the memory model, and its intricate interplay with the multiplicative connectives. The thesis introduces the first sound and complete internal proof system for the quantifier-free fragment of separation logic. The key insight facilitating the axiomatisation and leading to the completeness of the system is again given by the core formulae technique introduced in the first part of the thesis. The thesis goes one step further and show that this technique can be reused on other logics by designing an axiomatisation for a modal logic enriched with the parallel composition operator $\mid$ from ambient logic, a logic to verify distributed systems specified in the ambient calculus [6]. The two proof systems presented in the thesis reveal interesting connections between separation logics and ambient logics.

Motivated by the similarities in their proof systems, the third part of the thesis digs deep in the connections between separation logic and ambient logic. To carry out our comparison, we devise a suitable framework based on modal logic. This framework gives the common ground we need to study the two logics from the point of view of their multiplicative connectives: the separating conjunction $*$ for separation logic and the composition operator $\mid$ for ambient logic. Thanks to our framework, we are able to study the effects of having these operators as far as expressivity and complexity are concerned. Despite the similarities of the two logics in terms of their semantics, surprising differences are discovered, both in terms of expressive power and computational complexity. In particular, in our framework the composition operator $\mid$ is found to be strictly more expressive than the separating conjunction $*$, but results in logics that are easier to decide in terms of their satisfiability problem. Therefore, we provide here a nice and natural example of the fact that complexity and expressive power do not always go hand in hand.

Below, after formalising separation logic, we give a detailed view of the contributions of the thesis.

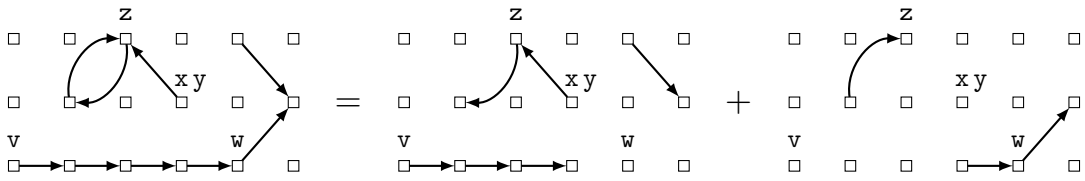

Figure 1: Union of disjoint heaps.

A brief introduction to separation logic

To better understand the content of the thesis, we consider the first-order separation logic defined in [4], which closely follow the initial presentation given by Reynolds in its seminal work [19]. Chapters 1 and 2 of the thesis provide a complete guide to this and other separation logics.

Memory states. As already stated, separation logic is interpreted on *memory states* that abstract the heap/RAM memory model. Addresses and data in the memory are abstracted with a countably infinite set of *locations*, denoted by LOC. Furthermore, a memory state contains information on the location assigned to each program variable, as well as dependencies between locations to represent pointers and their stored addresses. We write $\mathsf{VAR} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \dots\}$ for the set of *(program) variables*. A *store* $s : \mathsf{VAR} \to \mathsf{LOC}$ is a function that assigns a location to each program variable. A *heap* $h : \mathsf{LOC} \to_{\mathsf{fin}} \mathsf{LOC}$ is a partial function with finite domain of definition $\mathsf{dom}(h) = \{\ell \in \mathsf{LOC} \mid h(\ell) = \ell', \text{ for some } \ell' \in \mathsf{LOC}\}$. The heap describes the addresses in the memory that are stored in other addresses. A *memory state* is a pair (s, h) made from a store s and a heap h. For instance, the C-like instruction $^*\mathtt{x} = \mathtt{y}$ results in a memory that is described with a memory state (s, h) such that $h(s(\mathtt{x})) = s(\mathtt{y})$, as dereferencing the location stored in $\mathtt{x}$ yields the location stored in $\mathtt{y}$.

The subsequent definition of the multiplicative connectives of separation logic is based on the notion of *union of disjoint heaps*. Given two heaps h_1 and h_2 with disjoint domain of definition, their union $h_1 + h_2$ is the only heap such that $\mathsf{dom}(h_1 + h_2) = \mathsf{dom}(h_1) \cup \mathsf{dom}(h_2)$, for every $\ell \in \mathsf{dom}(h_1)$ $(h_1 + h_2)(\ell) = h_1(\ell)$, and for every $\ell \in \mathsf{dom}(h_2)$, $(h_1 + h_2)(\ell) = h_2(\ell)$. Figure 1 shows three memory states, where the heap of the leftmost one is obtained as the disjoint union of the heaps of the two memory states on the right. Locations are denoted by small boxes (□), and arrows represent the heap. The store is represented by decorating some of the locations with program variables.

Syntax and semantics. The formulae $\varphi, \psi, \dots$ of the first-order separation logic $\mathsf{SL}(\exists, *, \mathrel{-\!\!*})$ and its atomic formulae π are built from the grammars below (where $\mathtt{x}$, $\mathtt{y}$, $\mathtt{z} \in \mathsf{VAR}$):

$\pi :=$	$\top$	(true)	$\varphi, \psi :=$	π	(atomic formulae)
$\mid$	$\mathtt{emp}$	(empty predicate)	$\mid$	$\varphi \wedge \psi \mid \neg\varphi$	(Boolean connectives)
$\mid$	$\mathtt{x} = \mathtt{y}$	(equality predicate)	$\mid$	$\varphi * \psi$	(separating conjunction)
$\mid$	$\mathtt{x} \hookrightarrow \mathtt{y}$	(points-to predicate)	$\mid$	$\varphi \mathrel{-\!\!*} \psi$	(separating implication)
			$\mid$	$\exists \mathtt{z}\, \varphi$	(first-order quantification)

Let us discuss the satisfaction relation $\models$ for the formulae of $\mathsf{SL}(\exists, *, \mathrel{-\!\!*})$, given a memory state (s, h). The atomic formula $\mathtt{emp}$ tests whether the heap is empty, i.e. $(s, h) \models \mathtt{emp}$ if and only if $\mathsf{dom}(h) = \varnothing$. This formula provides a useful tool for program verification à la Hoare [15], as it allows to check whether a function called on the empty heap also returns on the empty heap. A violation of this property signals the presence of a memory leak. The formula $\mathtt{x} = \mathtt{y}$ tests whether the two variables $\mathtt{x}$ and $\mathtt{y}$ store the same location, i.e. $s(\mathtt{x}) = s(\mathtt{y})$. The formula $\mathtt{x} \hookrightarrow \mathtt{y}$ goes one step further, and states that the location assigned to the variable $\mathtt{x}$ *points to* the location assigned to $\mathtt{y}$, i.e. $h(s(\mathtt{x})) = s(\mathtt{y})$. The formula $\top$, the negation $\neg$ and conjunction $\wedge$ all have the standard semantics of propositional logic, and the existential first-order formula $\exists \mathtt{z}\, \varphi$ states that it is possible to update the location assigned to the variable $\mathtt{z}$ (via the store s) to obtain a memory state satisfying φ.

Lastly, $\mathsf{SL}(\exists, *, \mathrel{-\!\!*})$ features two multiplicative connectives: the separating conjunction $*$ and the separating implication $\mathrel{-\!\!*}$. For the separating conjunction, $(s, h) \models \varphi * \psi$ holds whenever it is possible

to split the heap h into two disjoint heaps h_1 and h_2 (i.e. $h = h_1 + h_2$) so that the memory state (s, h_1) satisfies φ and the memory state (s, h_2) satisfies ψ. Due to the semantics of the separating conjunction, the formula $\mathtt{x}_1 \hookrightarrow \mathtt{y}_1 * \mathtt{x}_2 \hookrightarrow \mathtt{y}_2 * \cdots * \mathtt{x}_n \hookrightarrow \mathtt{y}_n$ tacitly implies that for every two distinct $i, j \in [1, n]$, the variables $\mathtt{x}_i$ and $\mathtt{x}_j$ do not contain to the same location. We can translate this formula into one that only uses classical connectives, but unfortunately this requires a quadratic amount of inequalities: $\mathtt{x}_1 \hookrightarrow \mathtt{y}_1 \wedge \mathtt{x}_2 \hookrightarrow \mathtt{y}_2 \wedge \cdots \wedge \mathtt{x}_n \hookrightarrow \mathtt{y}_n \wedge \bigwedge_{i<j\in[1,n]} \neg(\mathtt{x}_i = \mathtt{x}_j)$. In the context of program analysis, this property of the separating conjunction comes with several benefits, that we describe in the first chapter of the thesis.

Let us now look at the separating implication. $(s, h) \models \varphi \mathrel{-\!\!*} \psi$ states that whenever we consider a heap h' disjoint from h and fulfilling $(s, h) \models \varphi$, the heap $h + h'$ fulfils $(s, h + h') \models \psi$. With this definition, the separating implication $\mathrel{-\!\!*}$ is the right adjoint of the separating conjunction $*$, exactly as the standard implication $\Rightarrow$ is the right adjoint of the conjunction $\wedge$. This means that if $\varphi * \psi \Rightarrow \chi$ is a tautology, then so is $\varphi \Rightarrow (\psi \mathrel{-\!\!*} \chi)$, and vice versa. The separating implication is computationally very powerful, and while it can be characterised in second-order logic, it cannot be defined in monadic second-order logic, as it requires a second-order quantification on a binary relation, i.e. the heap. This leads to the non-recursive enumerability of the satisfiability problem of first-order separation logic, which is otherwise Tower-complete (hence, non-elementary decidable) when the separating implication $\mathrel{-\!\!*}$ is removed form the logic [4].

Part I: reachability queries in separation logic

In program analysis, reachability predicates provide the foundation for verifying the absence of several memory mismanagements. For instance, let us consider the *reach-plus* predicate $\mathtt{x} \hookrightarrow^{+} \mathtt{y}$, that is satisfied by a memory state (s, h) whenever there is a natural number $\delta \geq 1$ for which $h^{\delta}(s(\mathtt{x})) = s(\mathtt{y})$, where h^{δ} stands for δ applications of h. That is, δ is the length of a non-empty path going from $s(\mathtt{x})$ to $s(\mathtt{y})$, in the functional graph described by h. With first-order quantification, the reach-plus predicate can be used to capture notions such as *acyclicity* and *garbage freedom*:

- *acyclicity*: A memory state is *acyclic* whenever no location can *reach* itself by traversing the heap a non-zero amount of times. In formula: $\neg\exists \mathtt{x}\, \mathtt{x} \hookrightarrow^{+} \mathtt{x}$.
- *garbage freedom*: A memory state is $\mathtt{X}$-*garbage free*, where $\mathtt{X}$ is a finite set of program variables, whenever all locations in the domain of the heap are reached by locations assigned to variables in $\mathtt{X}$. In formula: $\neg\exists \mathtt{x}\, (\mathtt{x} \hookrightarrow \_ \wedge \bigwedge_{\mathtt{y}\in\mathtt{X}} \neg(\mathtt{y} \hookrightarrow^{*} \mathtt{x}))$, where $\mathtt{x} \notin \mathtt{X}$, the *alloc* formula $\mathtt{x} \hookrightarrow \_$ is a shortcut for the formula $(\mathtt{x} \hookrightarrow \mathtt{x}) \mathrel{-\!\!*} \bot$ (stating that $s(\mathtt{x}) \in \mathsf{dom}(h)$), and $\mathtt{x} \hookrightarrow^{*} \mathtt{y}$ is $\mathtt{x} = \mathtt{y} \vee \mathtt{x} \hookrightarrow^{+} \mathtt{y}$.

As explained in [11], being able to check whether the acyclicity property holds is useful in analysing the termination of a program, as iterations on an acyclic structure are bound to terminate, whereas testing for garbage freedom allows us to show that the program does not leak memory by generating unreachable portions of the heap. Acyclicity and garbage freedom are thus key properties in the context of program verification, and logics that are able to express them are particularly desirable.

Motivations. Concerning separation logic, the use of reachability predicates is often restricted to the so-called *symbolic-heap fragment*, which is a quantifier-free fragment of $\mathsf{SL}(\exists, *, \mathrel{-\!\!*})$ that, among other restrictions, does not feature negation and the separating implication $\mathrel{-\!\!*}$. Despite being very appealing on a computational level (the satisfiability and entailment problems of the symbolic heap fragment are decidable in PTime [7]), these restrictions severely limits the range of properties we are able to check. Both the properties of acyclicity and garbage freedom cannot be expressed in the symbolic heap fragment and, even worse, no known separation logic with an elementary decidability status allows to express these types of properties. This leads us to journey through various fragments of first-order separation logic featuring reachability predicates, with the goal of designing a separation logic featuring reachability predicates that is expressive enough to capture notions such as acyclicity and garbage freedom, while still having a nice computational status. More precisely, we aim for its satisfiability (and validity and entailment) problem to be decidable in PSpace, exactly as it is the case of $\mathsf{SL}(*, \mathrel{-\!\!*})$, i.e. the quantifier-free fragment of $\mathsf{SL}(\exists, *, \mathrel{-\!\!*})$. This goal, which we eventually

reach in Chapter 5, reveals to be quite ambitious, as in both Chapters 3 and 4 we show how small extensions of $\mathsf{SL}(*, -\!\!*)$ lead to negative results in terms of computational complexity.

Chapter 3: extensionality and reachability leads to non-enumerability. Our journey starts by considering $\mathsf{SL}(*, -\!\!*)$ augmented with reachability predicates (e.g. $\mathtt{x} \hookrightarrow^{+} \mathtt{y}$) as well as the *bounded reachability predicates* $\mathtt{x} \hookrightarrow^{\delta} \mathtt{y}$ ($\delta \in \mathbb{N}$) that are satisfied by a memory state if the location assigned to $\mathtt{y}$ can be obtained by dereferencing $\mathtt{x}$ exactly δ times, i.e. $h^{\delta}(s(\mathtt{x})) = s(\mathtt{y})$. Surprisingly, we show that as soon as the bounded reachability predicates $\mathtt{x} \hookrightarrow^{2} \mathtt{y}$ and $\mathtt{x} \hookrightarrow^{3} \mathtt{y}$ are added to $\mathsf{SL}(*, -\!\!*)$, the satisfiability problem jumps from PSPACE-complete to non recursively enumerable. The main cause of this computational blow-up is traced back to the interactions between reachability predicates and the separating implication $-\!\!*$, which allow us to encode first-order quantifications by means of heap updates, hence obtaining a non-recursive lower bound by reduction from the satisfiability problem of $\mathsf{SL}(\exists, *, -\!\!*)$. This result extends to several separation logics featuring reachability predicates (e.g. $\mathsf{SL}(*, -\!\!*)$ enriched with $\mathtt{x} \hookrightarrow^{*} \mathtt{y}$), as well as a modal separation logic.

Modal separation logics are a family of modal logics introduced in [9] to emphasise the similarities between separation logics and modal logics. Briefly, modal separation logics are interpreted on Kripke structures where the accessibility relation is finite and functional, exactly as a heap. The standard modal logic lingua (featuring Boolean connectives, atomic propositions p and the alethic modality of possibility $\Diamond$) is extended with the two multiplicative connectives $*$ and $-\!\!*$ that act on the accessibility relation exactly as they do on a heap. These logics might also include other standard modalities, such as the universal modality $\langle \mathrm{U} \rangle$ of Goranko and Passy [12]. Chapter 3 shows that the satisfiability problem of the modal separation logic with just the modality $\Diamond$ and the operators $*$ and $-\!\!*$ is already non-recursively enumerable.

The content of this chapter has been published in:

▷ S. Demri, E. Lozes, and A. Mansutti, "The effects of adding reachability predicates in propositional separation logic." *Foundations of Software Science and Computational Structures*. 2018.

A long version of this paper has been accepted to *ACM Transactions on Computational Logic*.

Chapter 4: intensionality and reachability leads to non-elementary logics. In view of the results in Chapter 3, we remove for the time being the separating implication and consider instead a first-order quantification over a single variable name (renaming is not allowed). In particular, we focus on the separation logic $\mathsf{SL}([\exists]_1, *, \mathtt{x} \hookrightarrow \_, \hookrightarrow^{+})$ featuring one quantified variable name, the separating conjunction $*$, the reach-plus predicate $\mathtt{x} \hookrightarrow^{+} \mathtt{y}$, and the predicate alloc $\mathtt{x} \hookrightarrow \_$ stating that the location assigned to $\mathtt{x}$ is allocated in the memory. We show that this logic is already TOWER-complete. In the chapter, this result is proved in a more general settings: we discuss a set of features centred around reachability and submodel reasoning which causes logics interpreted on trees to be TOWER-hard. These features are formally described through a new modal logic which we call an Auxiliary Logic on Trees (ALT).

Briefly, ALT is interpreted on finite forests that are encoded with a functional relation, so that edges of the forest encode the *parent* relation. As in every modal logic, the semantics of a formula is given with respect to a node in the forest, called the *current node*. Differently from any other modal logic, ALT also features a *target node* that is fixed throughout the evaluation of the formula. Moreover, instead on considering standard atomic propositions, ALT only features two interpreted propositions `Hit` and `Miss`. The proposition `Hit` states that the current node is a descendant of the target node, i.e. there is a non-empty path going from the current node to the target note. The proposition `Miss` states that the current node has a parent in the forest, but it is not a descendant of the target node. Lastly, the logic features the universal modality $\langle \mathrm{U} \rangle$ that acts as a first-order quantification on the current node, the *sabotage modality* $\blacklozenge$ of van Benthem [2], that removes exactly one edge from the forest, and its Kleene closure $\blacklozenge^{*}$ that removes an arbitrary number of edges.

Chapter 4 first looks at the expressive power of ALT by defining a suitable notion of Ehrenfeucht-Fraïssé games. Afterwards, the satisfiability problem of ALT is shown to be TOWER-complete. The TOWER-hardness is proved by reduction from the satisfiability problem of Moszkowski's *Propositional interval temporal logic* (PITL) under locality principle [17]. Because of inexpressibility results

derived from the Ehrenfeucht-Fraïssé games, direct reductions from PITL to ALT are rather cumbersome. To alleviate this issue, in the thesis we introduce a new semantics for PITL which is proved to be equivalent and as concise as the original one, and that simplify the reduction to ALT.

The TOWER lower bound result for ALT is key, as this logic can be then used as a tool for proving other TOWER-hardness results for several logics interpreted over tree-like structures. Apart from $\mathsf{SL}([\exists]_1, *, \mathtt{x} \hookrightarrow \_, \hookrightarrow^{+})$, we show that ALT is captured by *quantified computation tree logic* (an extension of CTL featuring second-order propositional quantifiers) [16], *modal separation logics* without the separating implication $-\!\!*$ [9] and the modal logic of heaps from [8]. Thanks to ALT, new strict fragments of these logics are discovered to be non-elementary.

This chapter covers the work published in:

▷ A. Mansutti, "An auxiliary logic on trees: on the tower-hardness of logics featuring reachability and submodel reasoning". *Foundations of Software Science and Computational Structures*. 2020.

A long version of this paper was submitted to *Information and Computation* in November 2020.

Chapter 5: deciding robustness properties in PSpace. The negative results of Chapters 3 and 4 guide us to the definition of a separation logic that satisfies all our desiderata: (I) it extends $\mathsf{SL}(*, -\!\!*)$ with reachability predicates, (II) it can express the properties of acyclicity and garbage freedom, and (III) its satisfiability problem is PSPACE-complete, exactly as for $\mathsf{SL}(*, -\!\!*)$. We denote this logic with $\mathsf{SL}([\exists]_1, *, [-\!\!*, \hookrightarrow^{+}]^{\mathcal{S}}_{\mathcal{W}})$. At the time of writing, $\mathsf{SL}([\exists]_1, *, [-\!\!*, \hookrightarrow^{+}]^{\mathcal{S}}_{\mathcal{W}})$ is the only separation logic satisfying these three conditions. The logic is a syntactical fragment of first-order separation logic crafted to avoid the interactions between reachability and the connectives $*$ and $-\!\!*$ that, in the previous two chapters, were discovered causing computational blow-ups.

The entirety of Chapter 5 is devoted to the prove the PSPACE upper bound of the satisfiability problem for this logic, which reveals to be a challenging task. We rely on techniques from finite model theory, and establish a result that is reminiscent of the first-order locality theorem proved by H. Gaifman in [10]: every formula of $\mathsf{SL}([\exists]_1, *, [-\!\!*, \hookrightarrow^{+}]^{\mathcal{S}}_{\mathcal{W}})$ is shown to be equivalent to a Boolean combination of *core formulae*. Thanks to the definition of the core formulae, this equivalence yields a polynomial small model property for the logic, leading to the aforementioned PSPACE upper bound. Defining the right set of core formulae for a given logic is usually quite challenging, and even after obtaining the correct definition, proving that they capture the expressive power of the original logic is non-trivial. In our case, this is done by relying on simulation arguments that are related to a particular class of Ehrenfeucht-Fraïssé games for $\mathsf{SL}([\exists]_1, *, [-\!\!*, \hookrightarrow^{+}]^{\mathcal{S}}_{\mathcal{W}})$. In particular, we design an *indistinguishability relation* between memory states, so that two memory states are indistinguishable if they satisfy the same core formulae. With ad-hoc notions of simulations, we then show that indistinguishable memory states satisfy the same formulae of $\mathsf{SL}([\exists]_1, *, [-\!\!*, \hookrightarrow^{+}]^{\mathcal{S}}_{\mathcal{W}})$, starting from the atomic formulae and ending with the formulae headed by an operator $*$ or $-\!\!*$. Applied naïvely, these simulation arguments would lead to a very tedious combinatorial proof that exhaust all the interplay between the multiplicative connectives, the first-order quantification and the reachability predicates of the logic. To partially ease this issue, we introduce a technique that we call *game hopping*, as it is inspired by the homonymous technique for indistinguishability games in computational security [3]. Essentially, in a game hopping proof, the simulation arguments are only applied to memory states that are very similar, and for which the underlying combinatorics is amenable. The simulation argument between arbitrary memory states is then constructed by repeatedly modifying the two structures very slightly, in a way that allows to rely on the case of "similar memory states" to carry out the proof. Thanks to this novelty, the PSPACE upper bound is established in a more modular way, that avoids the monolithic proofs found in other iterations of the core formulae technique.

The content of this chapter has been published in:

▷ A. Mansutti, "Extending propositional separation logic for robustness properties". *Foundations of Software Technology and Theoretical Computer Science*. 2018.

Part II : internal calculi for spatial logics.

An Hilbert-style proof system $\mathcal{H}$ for a logic $\mathcal{L}$ is a set of formulae of $\mathcal{L}$ (the axioms), together with rules of the form $\varphi_1, \ldots, \varphi_n \vdash \psi$, which should be read as "if $\varphi_1, \ldots, \varphi_n$ are all valid formulae, then so is ψ". An Hilbert-style proof system is *internal*: all the formulae $\varphi_1, \ldots, \varphi_n, \psi$ are from $\mathcal{L}$. As usual, a system is said sound if all its axioms and rules are valid, and complete if every semantically valid formula of the logic can be proved valid in the calculus. Designing a sound and complete internal proof system is usually quite challenging, but beneficial:

- the system gives an exhaustive understanding of the expressive power of the logic, by discarding the use of any external artifact referring to semantical objects (e.g. labels or nominals),
- thanks to the insights on the expressive power, the system can improve existing decision procedures, by means of better abstractions or more refined calculi,
- the proof of completeness can be theoretically novel, thus paving the way towards Hilbert-style proof systems of other logics.

Because of these features, many logics related to program verification have been axiomatised, often requiring non-trivial completeness proofs. Unfortunately, this is not the case for separation logic, as the separating connectives $*$ and $-\!\!*$, together with the interpretation on memory states used by the logic, break standard techniques used to prove the completeness of Hilbert-style proof systems.

Motivations. Since the birth of separation logics, there has been a lot of interest in the study of decidability and computational complexity issues, and comparatively less attention to the design of proof systems, and even less with the puristic approach that consists in discarding any external feature such as nominals or labels in the calculi. In particular, an important open problem in the field is how to develop sound and complete Hilbert-style proof systems for separation logics. While sound and complete systems for BI exists, they do not account for the interpretation on memory states, and thus their completeness is lost when it comes to separation logic. In the second part of the thesis, we tackle this problem and introduce a methodology to axiomatise separation logics and similar logics, based on the core formulae technique introduced in Chapter 5. In the thesis, we use this method to axiomatise two logics: the quantifier-free separation logic $\mathsf{SL}(*, -\!\!*)$ and a modal logic enriched with the parallel composition operator $\mid$ form ambient logic [6]. To provide further evidence that the method can be reused in practice, we also applied it to derive axiomatisations of modal separation logics. While this last result is not part of the thesis, it can be found in the paper:

▷ S. Demri, R. Fervari, and A. Mansutti, "Axiomatising logics with separating conjunction and modalities". *Logics in Artificial Intelligence*. 2019.

A long version of this paper has been accepted to *Journal of Logic and Computation*.

Chapter 6: a complete axiomatisation for quantifier-free separation logic. We present the first Hilbert-style proof system for the quantifier-free separation logic $\mathsf{SL}(*, -\!\!*)$, whose formulae $\varphi, \psi, \ldots$ are built from the following grammar:

$$\varphi, \psi \coloneqq \top \mid \mathtt{emp} \mid \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \varphi \wedge \psi \mid \neg\varphi \mid \varphi * \psi \mid \varphi -\!\!* \psi.$$

As already stated, to design the proof system we rely on the core formulae technique. More precisely, we design an axiomatisation for Boolean combinations of core formulae, and add axioms and rules that allow to syntactically transform every formula of $\mathsf{SL}(*, -\!\!*)$ into such Boolean combinations. Schematically, our proof system is able to show the validity of a formula φ of $\mathsf{SL}(*, -\!\!*)$ by showing the validity of a Boolean combination of core formulae φ', together with the validity of the double implication $\varphi' \Leftrightarrow \varphi$. Our methodology leads to a modular calculus that is divided in three parts:

- the axiomatisation of Boolean combinations of core formulae, which extend a complete system for propositional calculus with axioms to deal with the interpretation on memory states,

- axioms and inference rules to simulate a bottom-up elimination of the separating conjunction $*$. Thanks to these rules, given Boolean combinations of core formulae φ and ψ, the formula $\varphi * \psi$ can be translated into an equivalent Boolean combination of core formulae,
- axioms and inference rules to simulate a bottom-up elimination of the separating implication $-\!\!*$: given Boolean combinations of core formulae φ and ψ, the formula $\varphi -\!\!* \psi$ can be translated into an equivalent Boolean combination of core formulae.

This methodology leads to a very desirable way to show the completeness of the proof system: only the completeness of the axiomatisation of Boolean combinations of core formulae has to be proven by semantical means (e.g. via a standard countermodel argument). The bottom-up elimination of the $*$ and $-\!\!*$ is showed completely syntactically, which leads to a less error-prone proof of completeness of the full calculus, and should drastically simplify its formalisation in a theorem prover.

The content of this chapter has been published in:

▷ S. Demri, E. Lozes, and A. Mansutti, "Internal calculi for separation logics". *Computer Science Logic*. 2020.

This paper has been selected for a special issue in *Logical Methods in Computer Science*.

Chapter 7: axiomatising a modal logic featuring ambient-like composition. Ambient logics [6] are modal logics introduced to verify properties of distributed systems specified in the calculus of Mobile Ambients. Their main feature of ambient logics is given by the composition operator $\varphi \mid \psi$ that, similarly to the separating conjunction, asks to spatially split the distributed process into two pieces, one satisfying the formula φ and the other satisfying the formula ψ. In Chapter 7 of the thesis, we consider the addition of the operator $\varphi \mid \psi$ to the standard modal logic K, obtaining the logic $\mathsf{ML}(\mid)$. We give an interpretation of this logic on Kripke structures that are forests, where the accessibility relation encodes the *child* relation. This interpretation makes the satisfiability problems of $\mathsf{ML}(\mid)$ and ambient logic equireducible.

Let us be more precise, and formalise the semantics of the composition operator $\mid$ in $\mathsf{ML}(\mid)$. Consider a Kripke-style forest $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$, where $\mathcal{W}$ is a set of worlds, R is a binary *accessibility* relation on worlds that represents a forest, and $\mathcal{V}$ evaluates the set of propositional symbols that are true in a given world. Given a *current world* $\mathsf{w} \in \mathcal{W}$, the formula $\varphi \mid \psi$ is satisfied by $(\mathcal{K}, \mathsf{w})$ whenever it is possible to partition the relation R into two relations R_1 and R_2 such that $((\mathcal{W}, R_1, \mathcal{V}), \mathsf{w})$ satisfies φ and $((\mathcal{W}, R_2, \mathcal{V}), \mathsf{w})$ satisfies ψ. Moreover, we constrain R_1 and R_2 so that if a world w' is a descendant of w with respect to R, i.e. there is a path going from w to w' in the forest described by R, then w' is a descendant of w with respect to either R_1 or R_2. This last condition brings $\mathsf{ML}(\mid)$ close to *graded modal logic* (GML), a well-known extension of modal logic K featuring graded modalities $\Diamond_{\geq k}\varphi$ stating that at least k children of the current world satisfy the formula φ. In fact, in $\mathsf{ML}(\mid)$ this formula is definable as $\Diamond\varphi \mid \cdots \mid \Diamond\varphi$, where $\mid$ occurs $k-1$ times.

Back to the axiomatisation of $\mathsf{ML}(\mid)$, this connection with GML proves to be essential: following the same principles as the one of $\mathsf{SL}(*, -\!\!*)$, we are able to take the formulae of GML as a set of core formulae, and add axioms to the standard proof system of GML in order to transform every formula of the form $\varphi \mid \psi$ into a formula of GML. In this way, we obtain a complete calculus for $\mathsf{ML}(\mid)$ that also shows that this logic is as expressive as GML.

Part III : mixing multiplicative connectives and modalities.

A closer look at the proof systems for $\mathsf{SL}(*, -\!\!*)$ and $\mathsf{ML}(\mid)$ introduced in the second part of the thesis reveals interesting similarities between separation logics and ambient logics. Despite the novelty of the two proof systems, connections between these two families of logics were already established in the early 2000s, when the two logics were firstly introduced. For instance, the decidability of the static ambient logic SAL considered in [5] is based on the proof technique firstly used to show the decidability of $\mathsf{SL}(*, -\!\!*)$. Nonetheless, a direct comparison between separation logics and ambient logics is missing or limited to a very superficial analysis on the different classes of models and multiplicative connectives used by the two logics.

Motivations. We aim for an in-depth comparison between separation logic and ambient logic in terms of their multiplicative connectives $\mid$ and $*$, by identifying a common ground in terms of logical languages and models. As a consequence, we are able to study the effects of having these operators as far as expressivity and complexity are concerned. To carry out our comparison, we consider $\mathsf{ML}(\mid)$ as an ambient logic and introduce the modal logic $\mathsf{ML}(*)$ which is obtained from $\mathsf{ML}(\mid)$ by replacing the composition operator $\mid$ with the separating conjunction $*$. This framework is sufficiently fundamental to give us the possibility to take advantage of model theoretical tools from modal logics, and sets a common ground for comparison that may lead to further connections with other logics. Despite the semantical similarities between $\mathsf{ML}(*)$ and $\mathsf{ML}(\mid)$, we identify surprising differences in terms of their expressiveness and complexity.

Part III covers the work published in:

▷ B. Bednarczyk, S. Demri, R. Fervari, and A. Mansutti, "Modal logics with composition on finite forests: Expressivity and complexity". *Logic in Computer Science*. 2020.

Chapter 8: the complexity of the modal logic $\mathsf{ML}(\mid)$. We continue the analysis of $\mathsf{ML}(\mid)$ started in Chapter 7, by looking at its computational complexity. We show that the satisfiability problem of $\mathsf{ML}(\mid)$ is AExp_{Pol}-complete, where AExp_{Pol} is the class of decision problems solvable by an alternating Turing machine with exponential runtime and polynomial number of alternations between existential and universal states. The AExp_{Pol} upper bound is derived from a refined translation to GML, using fundamental properties of the proof system designed in Chapter 7. To show that the satisfiability problem of $\mathsf{ML}(\mid)$ is AExp_{Pol}-hard, we propose to translate propositional team logic into a fragment of $\mathsf{ML}(\mid)$.

Propositional team logic is a semantics for propositional logic, proposed by Väänänen and based on a compositional semantics due to Hodges, where the satisfaction of a propositional formula is given with respect to a *team*, that is a set of valuations giving a truth value to propositional atoms. In team logics, the disjunction $\vee$ is not the dual of the conjunction $\wedge$, and is instead explicitly defined from the set union of teams: a team $\mathfrak{T}$ satisfies the formula $\varphi \vee \psi$ if and only if there are two teams $\mathfrak{T}_1$ and $\mathfrak{T}_2$ such that $\mathfrak{T} = \mathfrak{T}_1 \cup \mathfrak{T}_2$, $\mathfrak{T}_1$ satisfies φ and $\mathfrak{T}_2$ satisfies ψ. This makes propositional team logic an instantiation of BI, bringing it closer to $\mathsf{ML}(\mid)$, ambient logics and separation logics.

From the AExp_{Pol}-hardness of propositional team logic shown in [14] we are able to characterise the complexity of $\mathsf{ML}(\mid)$. Furthermore, we relate $\mathsf{ML}(\mid)$ to the intensional fragment of static ambient logic $\mathsf{SAL}(\mid)$ from [5] by providing polynomial-time reductions between their satisfiability problems. Consequently, we establish AExp_{Pol}-completeness for the satisfiability problem of $\mathsf{SAL}(\mid)$, refuting hints from [5, Section 6], were the problem was conjectured to be PSpace-complete.

Chapter 9: the complexity and expressive power of the modal logic $\mathsf{ML}(*)$. We introduce $\mathsf{ML}(*)$, the logic obtained from $\mathsf{ML}(\mid)$ by replacing the composition operator $\mid$ with the separating conjunction $*$. The semantics of $\varphi * \psi$ is similar to the semantics of $\varphi \mid \psi$: given a Kripke forest $\mathcal{K} = (\mathcal{W}, R, \mathcal{V})$ and $\mathsf{w} \in \mathcal{W}$, the formula $\varphi * \psi$ is satisfied by $(\mathcal{K}, \mathsf{w})$ whenever it is possible to partition the relation R into two relations R_1 and R_2 such that $((\mathcal{W}, R_1, \mathcal{V}), \mathsf{w})$ satisfies φ and $((\mathcal{W}, R_2, \mathcal{V}), \mathsf{w})$ satisfies ψ. However, differently from the operator $\mid$, the operator $*$ does not enforce additional constraints on R_1 and R_2, and in particular a descendant of w in R can become unreachable from w in both R_1 and R_2. This means that whereas every structure satisfying the formula $\varphi \mid \psi$ also satisfies $\varphi * \psi$, the opposite is not true.

We study the expressiveness of $\mathsf{ML}(*)$ as well as the complexity of its satisfiability problem. First, we show that $\mathsf{ML}(*)$ is strictly less expressive than $\mathsf{ML}(\mid)$ (or equivalently GML). Interestingly, this development partially reuses the result for $\mathsf{ML}(\mid)$, hence showing that our framework allows us to transpose results between the two logics. To show that $\mathsf{ML}(\mid)$ and GML are strictly more expressive than $\mathsf{ML}(*)$, we define an ad-hoc notion of Ehrenfeucht-Fraïssé games for $\mathsf{ML}(*)$.

Very surprisingly, although $\mathsf{ML}(*)$ is strictly less expressive than $\mathsf{ML}(\mid)$, its complexity is much higher. More precisely, we show that the satisfiability problem for $\mathsf{ML}(*)$ is Tower-complete. This is done by showing that for every $k \geq 2$, there is a formula of $\mathsf{ML}(*)$ of size (only) exponential in k that can be used to solve a kNExpTime-complete variant of the tiling problem.

Conclusion

The thesis provides new results on the complexity, expressiveness and proof theoretical aspects of separation logics. The first part of the thesis provides an in-depth analysis of separation logics with reachability predicates, which leads us to a PSPACE-complete separation logic that has genuine applications in program verification, being able to express useful properties of the heap/RAM model that are out of the scope of any other known elementary-decidable separation logic. In carrying out this analysis, we find several connections between separation logics and other logical formalisms, which is best witnessed by the results about the newly designed modal logic ALT interpreted on finite forests. In the second part of the thesis establishes the first sound and complete Hilbert-style proof system for separation logic. The model theoretical technique used to design the system and prove its completeness is shown to be reusable on other types of logics. The third part of the thesis discovers surprising differences between the separating conjunction $*$ of separation logic and the composition operator $\mid$ of ambient logics, where the operator $*$ is found to be less expressive but computationally harder than the operator $\mid$.

References

[1] "Common weakness enumeration," MITRE, https://cwe.mitre.org/.

[2] G. Aucher, J. van Benthem, and D. Grossi, "Modal logics of sabotage revisited," *Journal of Logic and Computation*, vol. 28, pp. 269 – 303, 2018.

[3] B. Blanchet and D. Pointcheval, "Automated security proofs with sequences of games," in *International Cryptology Conference*, ser. LNCS, vol. 4117. Springer, 2006, pp. 537–554.

[4] R. Brochenin, S. Demri, and E. Lozes, "On the almighty wand," *Information and Computation*, vol. 211, pp. 106–137, 2012.

[5] C. Calcagno, L. Cardelli, and A. D. Gordon, "Deciding validity in a spatial logic for trees," in *International Workshop on Types in Languages Design and Implementation*. ACM, 2003, pp. 62–73.

[6] L. Cardelli and A. D. Gordon, "Anytime, anywhere: Modal logics for mobile ambients," in *ACM SIGPLAN Symposium on Principles of Programming Languages*. ACM, 2000, pp. 365–377.

[7] B. Cook, C. Haase, J. Ouaknine, M. J. Parkinson, and J. Worrell, "Tractable reasoning in a fragment of separation logic," in *Concurrency Theory*, ser. LNCS, vol. 6901. Springer, 2011, pp. 235–249.

[8] S. Demri and M. Deters, "Two-variable separation logic and its inner circle," *Transactions on Computational Logic*, vol. 16, pp. 15:1–15:36, 2015.

[9] S. Demri and R. Fervari, "On the complexity of modal separation logics," in *Advances in Modal Logic*. College Publications, 2018, pp. 179–198.

[10] H. Gaifman, "On local and non-local properties," *Studies in Logic and the Foundations of Mathematics*, vol. 107, pp. 105–135, 1982.

[11] S. Genaim and D. Zanardini, "Inference of field-sensitive reachability and cyclicity," *Transactions on Computational Logic*, vol. 15, pp. 1–41, 2014.

[12] V. Goranko and S. Passy, "Using the universal modality: Gains and questions," *Journal of Logic and Computation*, vol. 2, pp. 5–30, 1992.

[13] E. Grädel, P. G. Kolaitis, and M. Y. Vardi, "On the decision problem for two-variable first-order logic," *Bulletin of Symbolic Logic*, vol. 3, no. 1, pp. 53–69, 1997.

[14] M. Hannula, J. Kontinen, J. Virtema, and H. Vollmer, "Complexity of propositional logics in team semantic," *Transactions on Computational Logic*, vol. 19, no. 1, pp. 2:1–2:14, 2018.

[15] C. A. R. Hoare, "An axiomatic basis for computer programming," *Communications of the Association for Computing Machinery*, vol. 12, no. 10, pp. 576–580, 1969.

[16] F. Laroussinie and N. Markey, "Quantified CTL: expressiveness and complexity," *Logical Methods in Computer Science*, vol. 10, 2014.

[17] B. C. Moszkowski, "Reasoning about digital circuits," Ph.D. dissertation, 1983.

[18] P. W. O'Hearn and D. J. Pym, "The logic of bunched implications," *Bulletin of Symbolic Logic*, vol. 5, pp. 215–244, 1999.

[19] J. C. Reynolds, "Separation logic: A logic for shared mutable data structures," in *Logic in Computer Science*. IEEE, 2002, pp. 55–74.