Automated reasoning in separation logic with inductive definitions

Cristina Serban

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTE UNIVERSITE GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Cristina SERBAN

Thèse dirigée par **Radu IOSIF, CNRS, Verimag**

préparée au sein du **Laboratoire VERIMAG**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Raisonnement automatisé pour la Logique de Séparation avec des définitions inductives

Automated Reasoning in Separation Logic with Inductive Definitions

Thèse soutenue publiquement le **31 mai 2018**,
devant le jury composé de :

Monsieur Radu IOSIF
Chargé de recherche, CNRS délégation Alpes, Directeur de thèse
Monsieur Didier GALMICHE
Professeur, Université de Lorraine, Président du jury
Monsieur Nikos GKOROGIANNIS
Professeur assistant, Université du Middlesex, Rapporteur
Monsieur Etienne LOZES
Professeur, Université Nice Sophia Antipolis, Rapporteur
Monsieur Wei-Ngan CHIN
Professeur associé, Université Nationale de Singapour, Rapporteur
Madame Mihaela SIGHIREANU
Maitre de conférences, Université Paris 7, Examinateur

Abstract

The motivation behind this thesis stems from verification tasks that check whether a given piece of code conforms to its specification. Properties of a certain state of the program are described through formulae pertaining to a chosen underlying logic, and it is often needed to test whether they hold and, more importantly, whether they entail the target specification. Satisfiability modulo theory (SMT) solvers are commonly used in practice to answer such queries. They are powerful tools due to their ability to combine decision procedures for several different theories.

We want to provide proof systems for entailments encountered when verifying programs that work with recursive data structures. This adds a layer of complexity to the entailment problem, as the formulae describing program states will need to make use of inductively defined predicates characterizing the data structures. Moreover, the programs will use dynamic allocation to create as many instances of the data as needed. Thus, we are interested in using separation logic to express properties of these programs, as it is a framework that addresses many of the difficulties posed by reasoning about dynamically allocated heaps.

The main contribution of this thesis is a sound and complete proof system for entailments between inductively defined predicates. We give a generalized cyclic proof system for first-order logic, which uses the principle of infinite descent to close recurring branches of a proof, and then adapt it to separation logic. In order to ensure soundness and completeness, four semantic restrictions are introduced, and we analyse their decidability and complexity. We also propose a proof-search semi-algorithm that becomes a decision procedure for the entailment problem when the semantic restrictions hold.

This higher-order reasoning about entailments requires first-order decision procedures for the underlying logic when applying some inference rules and during proof search. To this end, we introduce two decision procedures for separation logic, considering the quantifier-free and the $\exists^*\forall^*$-quantified fragments. We study the decidability and complexity of these fragments and show evaluation results of their respective decision procedures, which were integrated in the open-source, DPLL(T)-based SMT solver CVC4.

Finally, we also present an implementation of our proof system for separation logic, which makes use of the above decision procedures in CVC4. Given inductive predicate definitions and an entailment query as input, a warning is issued when one or more semantic restrictions are violated. If the entailment is found to be valid, the output is a proof. Otherwise, one or more counterexamples are provided.

Résumé

Cette thèse est motivée par les tâches de verification formelle qui établissent si un morceau de code donné est conforme à sa spécification. Les propriétés des états du programme sont décrites par des formules appartenant à la logique sous-jacente choisie, et il est souvent nécessaire de tester si elles sont correctes et, plus important encore, si elles impliquent la spécification cible. Les solveurs pour la satisfiabilité modulo des théories (SMT) sont utilisés dans la pratique pour répondre à de telles requêtes. Ce sont des outils puissants grâce à leur capacité de combiner des procédures de décision pour plusieurs théories.

Nous voulons fournir des systèmes de preuve pour les problèmes rencontrés lors de la vérification des programmes qui utilisent des structures de données récursives. Cela ajoute une couche de complexité au problème d'implication, car les formules décrivant les états du programme devront se servir de prédicats définis de façon inductive, qui caractérisent les structures de données. De plus, les programmes utiliseront l'allocation dynamique pour créer autant d'instances de données que nécessaire. D'où l'intérêt d'utiliser la logique de séparation pour exprimer les propriétés de ces programmes, car c'est un cadre qui répond aux plusieurs difficultés posées par le raisonnement sur les tas alloués dynamiquement.

La contribution principale de cette thèse est un système de preuve correct et complet pour les implications entre les prédicats définis de façon inductive. Nous donnons un système de preuve cyclique généralisé pour la logique du premier ordre, qui utilise le principe de la descente infinie pour fermer les branches récurrentes d'une preuve, puis nous l'adaptons à la logique de séparation. Afin d'assurer la correction et la complétude, quatre restrictions sémantiques sont introduites et nous analysons leur décidabilité et leur complexité. Nous fournissons également un semi-algorithme de recherche de preuve qui devient une procédure de décision pour le problème d'implication lorsque les restrictions sémantiques sont respectés.

Ce raisonnement d'ordre supérieur sur les implications nécessite des procédures de décision de premier ordre pour la logique sous-jacente lors de l'application de certaines règles d'inférence et lors de la recherche des preuves. À fin, nous introduisons deux procédures de décision pour la logique de séparation, en considérant le fragment sans quantificateurs et le fragment quantifié $\exists^*\forall^*$. Nous étudions la décidabilité et la complexité de ces fragments et montrons les résultats d'évaluation de leurs procédures de décision, qui ont été intégrées dans le solveur SMT open source CVC4.

Enfin, nous fournissons une implémentation de notre système de preuve pour la logique de séparation, qui utilise les procédures de décision ci-dessus. Étant donné un ensemble de definitions de prédicats inductifs et une requête d'implication, un avertissement est émis lorsqu'une ou plusieurs restrictions sémantiques sont violées. Nous obtenons une preuve si l'implication est valide. Sinon, un ou plusieurs contre-exemples sont fournis.

Contents

Introduction

Program verification is the use of formal and mathematical methods to provide proof that a program meets its specification. The importance of specifying and verifying code lies in the numerous advantages that it brings. The formal specification of a program, usually a formula written in a suitable logic, can serve both as documentation and a guideline for implementation. It also dictates that a lot more care and scrutiny be put on software requirements, thus contributing to the elimination of inaccuracies during early development stages, when the cost of changes is much lower compared to later stages. Moreover, verification can help reduce the time and cost of testing, debugging and maintaining big systems, since code that is specified and verified is much easier to understand, reuse and maintain. In the case of safety-critical systems, the proof that the code meets its specification serves as certification for correct performance at all times.

Early approaches to program verification used manual annotation with assertions describing the state of the execution. A rigorous formalization of this process is given by Hoare logic, introduced by C. A. R. Hoare [22] and based on previous work by Robert W. Floyd, who proposed a similar system for flowcharts [17]. Hoare logic is a formal system used to reason about Hoare triples $\{P\}C\{Q\}$, where C is a piece of code, P is a formula called the *precondition*, describing the state before C executes, and Q is a formula called a *postcondition*, describing the state after C executes.

Example 1. Consider a function `swap(x, y)`, which swaps the values of the variables `x` and `y`. We would like such a function to satisfy the specification given by the triple $\{x = x_0 \wedge y = y_0\}$ `swap(x, y)` $\{x = y_0 \wedge y = x_0\}$, where x_0 and y_0 are constants denoting the initial values of `x` and `y`. ◀

However, manually annotating programs required a substantial effort from the user and was prone to human mistakes. As software complexity increased and verification techniques advanced, automated methods gained in popularity. It has become very easy and cheap to automatically compute the postcondition for an entire block of code starting from the precondition (or vice versa) using deduction rules. The focus has shifted from manual annotation to automated verification of entire functions or components given very little user input.

Decision procedures are building blocks that lie at the basis of automated verification. They are algorithms that, given a decision problem (i.e. a problem that can be posed as a *yes* or *no* question of the input, usually formulae limited to a particular theory), terminate with a correct answer [30]. Given a problem, we are interested in its decidability and we want a clear definition for the decidable fragment of the underlying theory. Given a decision procedure for a problem, we want it to be sound (i.e. every instance proven by the procedure is valid) and complete (i.e. every valid instance can be proven by the procedure).

In practice, however, the decidable theory fragment that guarantees completeness can be quite restrictive and lacking in expressiveness. In consequence, we want a decision procedure to be general enough that it can be run on any instance of a problem. Then, soundness is enough to give us a semi-algorithm that might not terminate on some inputs, but can still provide a correct answer on others. Additionally, a warning can be issued to the user when termination is not guaranteed for the given input.

From a practical standpoint, the usefulness of decision procedures is highlighted by satisfiability modulo theory (SMT) solvers. They are very powerful reasoning tools, able to combine decision procedures for different theories, while preserving their soundness and completeness. It is common practice for software verifiers to translate any kind of assertion (e.g. preconditions, postconditions or loop conditions) into SMT formulae and use an SMT solver to determine whether the properties can hold.

Proving entailments is a problem that regularly arises during these verification tasks due to the *rule of consequence*: if a piece of code satisfies precondition P and postcondition Q, then it also satisfies precondition P' and postcondition Q' if $P' \models P$ and $Q \models Q'$. Variations of this rule take into account only one of these entailments and either strengthen P with P' or weaken Q with Q'. In practice, proving that a program conforms to a given specification can be done by starting with the given precondition, computing the strongest postcondition of the code and then proving that it entails the postcondition from the specification. Conversely, we can start with the given postcondition, compute the weakest precondition of the code and then prove that it is entailed by the precondition from the specification.

Example 2. Consider the following in-place implementation of the `swap(x, y)` function from Example 1.

`swap(x, y)`

$\{x = x_0 \land y = y_0\}$	Precondition
`x := x + y`	
$\{x = x_0 + y_0 \land y = y_0\}$	
`y := x - y`	
$\{x = x_0 + y_0 \land y = x_0 + y_0 - y_0\}$	
`x := x - y`	
$\{x = x_0 + y_0 - (x_0 + y_0 - y_0) \land y = x_0 + y_0 - y_0\}$	Strongest postcondition

We try to verify that it satisfies the required specification by starting with the precondition and computing its strongest postcondition. Because $x = x_0 + y_0 - (x_0 + y_0 - y_0) \land y = x_0 + y_0 - y_0 \models x = y_0 \land y = x_0$, we can conclude that this implementation of `swap(x, y)` meets its specification. ◀

Function calls also introduce the necessity of proving entailments, as we need to check that the formula describing the state before the function call entails the precondition from the function specification. This ensures the safety of the function call and that the postcondition given by the function specification will hold afterwards.

Example 3. If the state before a call of `swap(x, y)` from Example 1 is described by the formula $x = x_0 \land y = y_0 \land x_0 < y_0$, then, because $x = x_0 \land y = y_0 \land x_0 < y_0 \models x = x_0 \land y = y_0$, we can safely call the function and the triple $\{x = x_0 \land y = y_0 \land x_0 < y_0\}$ `swap(x, y)` $\{x = y_0 \land y = x_0\}$ is true. ◀

Modern software is required to handle data in a systematic fashion and to adapt to any amount of information. This reflects onto the formulae that describe program states

during verification – they must be able to capture complex data properties. Thus, the ability to arrange data into suitable data structures is essential. As defined by Cormen, *data structures* represent a way to organize data in order to facilitate access and modifications [14]. Of particular interest are recursive data structures, due to their ubiquitous presence in real-life software.

We can distinguish two tiers when it comes to recursive data structures. Firstly, there is a higher tier of inductive data types, which are explicitly supported by functional languages. They represent an abstraction for the lower, concrete tier of more classic recursive structures used by imperative programming languages, such as linked lists, stacks, queues, and several types of trees that may be balanced or partially balanced such as binary search trees, AVL trees, 2-3 trees or red-black trees [14].

Inductively defined predicates can be used to describe recursive data types and, in the case of programs that use dynamic allocation, they can also specify the shape of the memory where the structures are stored. Thus, the ability to automatically reason about inductive predicates becomes an important tool for program verification. Furthermore, decision procedures that incorporate inductive predicates are essential for reasoning at both the abstract and concrete data structure tiers.

Example 4. Perhaps the simplest example of an inductive predicate is the one for natural numbers that uses the successor function *succ* from Peano arithmetic:

$$\mathsf{Natural}(x) ::= x = 0 \mid \exists y \,.\, x = \mathit{succ}(y) \wedge \mathsf{Natural}(y)$$

Similarly, we can define predicates that describe even and odd natural numbers:

$$\begin{aligned} \mathsf{Even}(x) &::= x = 0 \mid \exists y \,.\, x = \mathit{succ}(y) \wedge \mathsf{Odd}(y) \\ \mathsf{Odd}(x) &::= \exists y \,.\, x = \mathit{succ}(y) \wedge \mathsf{Even}(y) \end{aligned}$$

◀

Abstract data structures, as proposed by Hoare [23, 38], have an associated constructor function c and k selector functions $s_1, \ldots, s_k$, which satisfy abstract structural properties referring to:

(i) Construction: $c(s_1(x), \ldots, s_k(x)) = x$;

(ii) Selection: $s_i(c(x_1, \ldots, x_k)) = x_i$ for $1 \leq i \leq k$;

(iii) Acyclicity: $s_i(x) \neq x$ for $1 \leq i \leq k$, $s_i(s_j(x)) \neq x$ for $1 \leq i, j \leq k$, ...

These types can be easily defined in first-order logic under the Herbrand interpretation. Alternatively, extensions of first-order logic with recursive definitions, such as Dryad [34], have been introduced.

Example 5. A well-known implementation of an abstract data structure is the list in LISP, with constructor *cons* and selectors *car* and *cdr*. A list is either nil or constructed using *cons* by prepending an element to another list. The selector *car* returns the first element of a list and the selector *cdr* returns the rest of the list. An inductively defined predicate for this list would be

$$\mathsf{list}(x) ::= x = \mathsf{nil} \mid \exists y, z \,.\, x = \mathit{cons}(y, z) \wedge \mathsf{list}(z)$$

Note how inductive definitions can only cover the construction axioms, but not selection and acyclicity. ◀

Since the amount of data a program has to work with is hardly ever predetermined, dynamic allocation is used to create as many instances as needed. Reasoning about dynamically allocated heaps poses several difficulties. Expressing the state of the memory within a formula is hard, as it needs to cover the properties of the unbounded heap, while also taking care of other aspects such as aliasing and reachability. Furthermore, it has been observed that, in practice, pre- and postconditions can become very large and complex, describing parts of the heap that a function, for instance, does not touch, but whose properties have to be carried through the body of the function during verification.

Separation logic is a framework introduced by Ishtiaq, O'Hearn, Reynolds and Yang [29, 37, 45] that emerged to address these issues. Its separating operators together with the *frame rule* enable local reasoning by breaking down the heap into disjoint parts and allowing only those manipulated by a certain portion of the code to be taken into consideration while performing verification tasks.

Example 6. Consider `swap`$^{\text{SL}}$, a variant of the `swap` function from Example 1 that works with variables representing memory locations and swaps their content. Its specification is given by the triple $\{x \mapsto x_0 * y \mapsto y_0\}$ `swap`$^{\text{SL}}$`(x, y)` $\{x \mapsto y_0 * y \mapsto x_0\}$. The $\mapsto$ predicate indicates a single allocated cell and the $*$ connective indicates two disjoint heaps (i.e. heaps in which two disjoint sets of locations are allocated). The implementation and verification of `swap`$^{\text{SL}}$ are done in a similar fashion as in Example 2, the difference being that, instead of variable assignment, we use mutation to change the values stored at the addresses indicated by the variables `x` and `y`. ◀

Moreover, switching from first-order logic under the canonical interpretation to separation logic allows for more expressiveness when defining recursive data structures. Instead of being limited to types that can only be represented as trees, separation logic allows us to define more complex and realistic data structures, such as doubly linked lists and trees with parent pointers or linked leaves.

Example 7. Structurally, a separation logic definition for a doubly-linked list is analogous to the LISP definition of a list from Example 5 – it is either empty or obtained by prepending an element to another list. The difference lies in how the focus is shifted towards the heap in which the list is allocated. For the former case, the head of the list x is nil and nothing is allocated in the heap. For the latter case, the head of the list points to the next and previous locations y and p, and, disjointly, there is a doubly-linked list starting at y with previous location x.

$$\mathsf{dll}(x, p) ::= x = \mathsf{nil} \wedge \mathsf{emp} \mid \exists y \,.\, x \mapsto (y, p) * \mathsf{dll}(y, x)$$

Defining a binary tree is similar. Either the root is nil and the heap is empty, or the root is allocated and points to the left and right subtrees.

$$\mathsf{tree}(x) ::= x = \mathsf{nil} \wedge \mathsf{emp} \mid \exists y, z \,.\, x \mapsto (y, z) * \mathsf{tree}(y) * \mathsf{tree}(z)$$

This definition of a tree can be expanded to obtain a binary tree with at least one allocated node and linked leaves (i.e. there is a linked list from its leftmost leaf to its rightmost leaf).

$$\begin{aligned} \mathsf{tree}^{ll}(x, ll, lr) ::= {} & x = ll \wedge x \mapsto (\mathsf{nil}, \mathsf{nil}, lr) \\ \mid {} & \exists y, z, u \,.\, x \mapsto (y, z, \mathsf{nil}) * \mathsf{tree}^{ll}(y, ll, u) * \mathsf{tree}^{ll}(z, u, lr) \end{aligned}$$

◀

Contributions

The main contribution of this thesis targets proof systems for entailments involving inductively defined predicates. Influenced by an antichain-based method designed for checking language inclusion of nondeterministic finite tree automata [24], we build a generic cyclic proof system for first-order logic, which we later adapt to separation logic. The actions performed by our proof system are generalized to fit a wider class of entailment problems, but certain successions of steps can be translated back to those of the above language inclusion check. A key aspect is the reliance on the principle of Infinite Descent [10], initially formalized by Fermat, which is ubiquitous in cyclic proof systems [9]. In short, this principle allows us to close recurring branches of a proof on the premise that, if a counterexample were to be discovered on that branch, then it would lead to an infinite sequence of strictly decreasing counterexamples in a well-founded domain, which constitutes a contradiction.

Since language inclusion is decidable for nondeterministic finite tree automata [12, Corollary 1.7.9], our proof system is sure to be complete for entailment problems equivalent to those of this restricted class. In consequence, we tried to establish some boundaries for the inductive predicates within which the proof system remains sound and complete for more general problems. As a result, we obtained one semantic condition that ensures soundness, and three additional other restrictions that warrant completeness. All four constraints can be checked using existing decision procedures, with varying complexities dictated by the chosen underlying logic. This contrasts with other similar approaches, in which the inductive predicate definitions are bounded by syntactic constraints.

Multiple inference rules from our proof system rely on existing decision procedures for the non-inductive fragment of the underlying logic. In this sense, there are extensive, well-established results for first order logic [13, 39], but this is not also the case for separation logic. Therefore, we provide two decision procedures geared towards the quantifier free and the $\exists^*\forall^*$-quantified fragments of separation sogic, respectively. We study the decidability and complexity of the satisfiability problem in both of these fragments. Furthermore, these procedures were integrated in the DPLL(T)-based SMT solver CVC4 and we show the results of their performance evaluation.

Finally, we present an entailment checker tool, `Inductor`, which implements our proof search method for the specialized separation logic proof system. `Inductor` is written in C++ and utilizes our dedicated decision procedures for separation logic from CVC4. Given a proof search strategy, along with an SMT-LIB script containing inductive predicate definitions and entailments that need to be checked, `Inductor` uses a compact tree structure to explore all the possible derivations enabled by the strategy, in a breadth-first fashion. The search stops whenever a proof or a counterexample is discovered and the output is either a successful one, accompanied by the proof, or an unsuccessful one, supported by the counterexample. The result of the search may also be inconclusive, when all possibilities are explored without yielding either a proof or a counterexample. The search strategy and whether the inductive system satisfies the constraints ensuring soundness and completeness are key factors that can lead to such situations. Appropriate warnings are displayed whenever one or more of the constraints are violated.

Organization

This thesis studies complete proof systems for entailments involving inductive predicates, in both first-order logic and separation logic. Chapter 1 introduces the syntax and semantics of

inductive systems of predicates, while Chapter 2 presents proof systems for the entailment of predicates defined by an inductive system, together with the necessary restrictions that an inductive system needs to satisfy in order to allow for the soundness and completeness of the proof system. Adjacently, Chapter 3 focuses on decision procedures for separation logic in SMT, for both the quantifier-free and the $\exists^*\forall^*$-quantified fragments. Chapter 4 describes an implementation for our proof system targeting inductive entailments in separation logic, which employs the decision procedures from Chapter 3, and then goes on to analyse some insightful case studies.

Notations

The following notations will be frequently used throughout this work:

- $[i, j]$, $[i]$ – For two integers $0 \leq i \leq j$, $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$ and $[i]$ is shorthand for $[1, i]$, with $[0]$ being the empty set;
- $\|S\|$– Given a finite set S, $\|S\|$ denotes its cardinality;
- $\mathcal{P}(S)$, $\mathcal{P}_{\mathit{fin}}(S)$ – Given a set S, $\mathcal{P}(S)$ denotes its powerset and $\mathcal{P}_{\mathit{fin}}(S)$ the set of finite subsets of S;
- $\mathrm{dom}(f), \mathrm{img}(f)$ – For a (total or partial) mapping $f : A \to B$, $\mathrm{dom}(f) = A$ denotes the set of values on which f is defined and $\mathrm{img}(f) = f(\mathrm{dom}(f)) \subseteq B$ is the set of values in the range of f;
- B^A – For two sets A and B, B^A denotes the set of all functions $f : A \to B$;
- f^n – For a function $f : A \to B$ and $n \in \mathbb{N}$, f^n denotes the result of composing f with itself $n-1$ times. By convention, $f^0 = id_A$ (i.e. the identity function on the domain of f) and $f^1 = f$;
- $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ – We use $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ to denote the sets of natural, integer and real numbers, respectively;
- $\{E_i\}_{i=1}^n$ – For conciseness, we use this notation to mean the set $\{E_1, \ldots, E_n\}$ (or, equivalently, $\{E_i \mid i \in [n]\}$), where each E_i is an element indexed in some way by $i \in [n]$;
- $f{\downarrow_D}$ – Given a function f, we use this notation to refer to its restriction to the domain $D \subseteq \mathrm{dom}(f)$.
- $\mathbf{s}$, $\bar{\mathbf{s}}$ – We use $\mathbf{s} = \{s_1, \ldots, s_n\}$ and $\bar{\mathbf{s}} = \langle s_1, \ldots, s_n \rangle$ for sets and ordered tuples, respectively. Given a tuple $\bar{\mathbf{s}}$, we denote its length by $\|\bar{\mathbf{s}}\|$, the element at position $i \in [\|\bar{\mathbf{s}}\|]$ by $\mathsf{pos}_i(\bar{\mathbf{s}})$, and the set of all variables in $\bar{\mathbf{s}}$ by $\mathsf{set}(\bar{\mathbf{s}}) = \{x \mid x = \mathsf{pos}_i(\bar{\mathbf{s}}) \text{ for some } i \in [\|\bar{\mathbf{s}}\|]\}$. We call a tuple $\bar{\mathbf{s}}$ *unique* when each of its elements occurs only once (i.e. $\|\bar{\mathbf{s}}\| = \|\mathsf{set}(\bar{\mathbf{s}})\|$). Given two tuples $\bar{\mathbf{s}} = \langle s_1, \ldots, s_n \rangle$ and $\bar{\mathbf{s}}' = \langle s'_1, \ldots, s'_m \rangle$, we use $\bar{\mathbf{s}} \cdot \bar{\mathbf{s}}'$ to denote their concatenation, i.e. $\bar{\mathbf{s}} \cdot \bar{\mathbf{s}}' = \langle s_1, \ldots, s_n, s'_1, \ldots, s'_m \rangle$.

Chapter 1

Inductive Systems

In this chapter, we first introduce basic syntax and semantics for both first-order logic (Section 1.1) and separation logic (Section 1.2). We consider general first-order theories and show how separation logic formulae can be built on top of them. We then define systems of inductive definitions, together with their (least) solutions (Section 1.3), for a fixed first-order theory and a more restricted class of separation logic formulae. Lastly, we describe the entailment problem for inductive systems and show how it is undecidabile under several interpretations (Section 1.4).

1.1 First-Order Logic (FOL)

1.1.1 Syntax

We introduce the syntax of first-order logic (FOL) in the context of a pair $\Sigma = (\Sigma^{\mathsf{s}}, \Sigma^{\mathsf{f}})$, which we call a *signature*, such that:

- $\Sigma^{\mathsf{s}} = \{\sigma_1, \sigma_2, \ldots\}$ is a set of *sort symbols*. We assume the existence of a *location* sort $\mathsf{Loc} \in \Sigma^{\mathsf{s}}$ and a *boolean* sort $\mathsf{Bool} = \{\top, \bot\} \in \Sigma^{\mathsf{s}}$, where we write $\top$ and $\bot$ for the constants *true* and *false*, respectively;

- $\Sigma^{\mathsf{f}} = \{f, g, h, ...\}$ is a set of *function symbols*. For a function symbol $f^{\sigma_1 \cdots \sigma_n \sigma}$, $n \geq 0$ is its *arity* and $\sigma_1 \ldots \sigma_n \sigma$ is its *signature*, where $\sigma_1, \ldots, \sigma_n \in \Sigma^{\mathsf{s}}$ are the sorts of its arguments and $\sigma \in \Sigma^{\mathsf{s}}$ is the sort of its result. A function symbol with arity 0 is a *constant symbol* c^{σ} of sort $\sigma \in \Sigma^{\mathsf{s}}$. We omit specifying the signature of a function symbol when it is not important.

Let $\mathsf{Var} = \{x, y, z, ...\}$ be a countable set of *first-order variables*. Each variable $x^{\sigma} \in \mathsf{Var}$ has an associated sort $\sigma \in \Sigma^{\mathsf{s}}$. As we do with function symbols, we omit the sort of a variable if it is not necessary. We write $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $\ldots \subseteq \mathsf{Var}$ and $\overline{\mathbf{x}}$, $\overline{\mathbf{y}}$, $\overline{\mathbf{z}}$, $\ldots \in \bigcup_{n \in \mathbb{N}} \mathsf{Var}^n$ for sets of first-order variables and tuples of first-order variables, respectively.

Definition 1.1.1 (Term)**.** A *term* t of sort $\sigma \in \Sigma^{\mathsf{s}}$, denoted as t^{σ}, over a signature Σ (also called a Σ-*term*) is defined recursively by the grammar:

$$
\begin{array}{lll}
t^\sigma ::= x, & x^\sigma \in \mathsf{Var} & \text{(variable)} \\
\mid\ c, & c^\sigma \in \Sigma^{\mathsf{f}} & \text{(constant)} \\
\mid\ \langle t_1, \ldots, t_n \rangle, & t_1^{\sigma_1}, \ldots, t_n^{\sigma_n} \text{ terms}, \ \sigma = \sigma_1 \times \ldots \times \sigma_n & \text{(tuple)} \\
\mid\ f(t_1, \ldots, t_n), & f^{\sigma_1 \ldots \sigma_n \sigma} \in \Sigma^{\mathsf{f}},\ t_1^{\sigma_1}, \ldots, t_n^{\sigma_n} \text{ terms} & \text{(function application)}
\end{array}
$$

Any variable or constant symbol of sort σ is a term of sort σ. If $t_1, \ldots, t_n$ are terms of sorts $\sigma_1, \ldots, \sigma_n$ and $f^{\sigma_1 \ldots \sigma_n \sigma} \in \Sigma^{\mathsf{f}}$, then $\langle t_1, \ldots, t_n \rangle$ is a term of sort $\sigma_1 \times \ldots \times \sigma_n$ and $f(t_1, \ldots, t_n)$ is a term of sort σ.

We denote by $\mathcal{T}_\Sigma(\mathbf{x})$ the set of all terms constructed using function symbols in Σ^{f} and variables in the set $\mathbf{x}$ and extend this notation to tuples $\overline{\mathbf{x}}$ such that $\mathcal{T}_\Sigma(\overline{\mathbf{x}}) = \mathcal{T}_\Sigma(\mathsf{set}(\overline{\mathbf{x}}))$. We write $\mathcal{T}_\Sigma$ for the set $\mathcal{T}_\Sigma(\emptyset)$ of *ground terms*, which contain no variable occurrences.

Definition 1.1.2 (First-order formula)**.** A *first-order formula* over a signature Σ (also called a Σ-*formula*) is defined recursively by the grammar:

$$
\begin{array}{lll}
\phi^{\mathsf{FOL}} ::= \top & & \text{(true)} \\
\mid\ \bot & & \text{(false)} \\
\mid\ t, & t^{\mathsf{Bool}} \text{ term} & \text{(boolean term)} \\
\mid\ t_1 \approx t_2, & t_1^\sigma,\ t_2^\sigma \text{ terms} & \text{(equality)} \\
\mid\ \neg\psi^{\mathsf{FOL}}, & & \text{(negation)} \\
\mid\ \psi_1^{\mathsf{FOL}} \wedge \psi_2^{\mathsf{FOL}}, & & \text{(conjunction)} \\
\mid\ \psi_1^{\mathsf{FOL}} \vee \psi_2^{\mathsf{FOL}}, & & \text{(disjunction)} \\
\mid\ \exists x \,.\, \psi^{\mathsf{FOL}}, & x \in \mathrm{FV}(\psi^{\mathsf{FOL}}) & \text{(existential quantification)} \\
\mid\ \forall x \,.\, \psi^{\mathsf{FOL}}, & x \in \mathrm{FV}(\psi^{\mathsf{FOL}}) & \text{(universal quantification)}
\end{array}
$$

The constants $\top$ and $\bot$, a boolean term, and the equality between two terms of the same sort are FOL formulae. The negation, conjunction, disjunction, existential and universal quantification of FOL formulae are also FOL formulae.

For a formula ϕ, we denote by $\mathrm{FV}(\phi)$ the set of variables not occurring under the scope of a quantifier in ϕ. Writing $\phi(\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ means that $\bigcup_{i=1}^n \mathsf{set}(\overline{\mathbf{x}}_i) \subseteq \mathrm{FV}(\phi)$, where $n \geq 0$. These notations are lifted to sets of formulae F such that $\mathrm{FV}(F) = \bigcup_{\phi \in F} \mathrm{FV}(\phi)$ and $F(\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ means that $\bigcup_{i=1}^n \mathsf{set}(\overline{\mathbf{x}}_i) \subseteq \mathrm{FV}(F)$, where $n \geq 0$.

Given a formula ϕ and a tuple of variables $\overline{\mathbf{x}} = \langle x_1, \ldots, x_n \rangle$ (a set $\mathbf{x} = \{x_1, \ldots, x_n\}$) we use $\exists \overline{\mathbf{x}} \,.\, \phi$ ($\exists \mathbf{x} \,.\, \phi$) as a shorthand for $\exists x_1 \ldots \exists x_n \,.\, \phi$. We do the same for $\forall \overline{\mathbf{x}} \,.\, \phi$ ($\forall \mathbf{x} \,.\, \phi$). For formulae ϕ, ψ and ψ', we denote by $\phi[\psi]$ the fact that ψ is a subformula of ϕ and by $\phi[\psi'/\psi]$ the result of replacing ψ with ψ' in ϕ. We also write $\phi \Rightarrow \psi$ for $\neg\phi \vee \psi$.

We also introduce variable substitutions, which will be useful in defining certain properties of inductive systems.

Definition 1.1.3 (Substitution and flat substitution)**.** Given sets of variables $\mathbf{x}$ and $\mathbf{y}$, a *substitution* $\theta : \mathbf{x} \to \mathcal{T}_\Sigma(\mathbf{y})$ maps each variable in $\mathbf{x}$ to a term in $\mathcal{T}_\Sigma(\mathbf{y})$. We denote the image of $\mathbf{x}$ under the substitution θ by $\mathbf{x}\theta = \{\theta(x) \mid x \in \mathbf{x}\}$. A substitution θ is *flat* if $\mathsf{Var}\theta \subseteq \mathsf{Var}$, i.e. each variable is mapped to a variable. A flat substitution $\theta : \mathbf{x} \to \mathbf{y}$ is *injective* if, for all $x_1, x_2 \in \mathbf{x}$, $\theta(x_1) = \theta(x_2)$ implies $x_1 = x_2$. A flat substitution is *surjective* if for any $y \in \mathbf{y}$ there exists $x \in \mathbf{x}$ such that $\theta(x) = y$ or, in other words, $\mathbf{x}\theta = \mathbf{y}$.

For a formula $\phi(\mathbf{x})$, we denote by $\phi\theta$ the result obtained by replacing each occurrence of $x \in \mathbf{x}$ in ϕ with the term $\theta(x)$. This notation is lifted to sets of formulae such that $F\theta = \{\phi\theta \mid \phi \in F\}$.

1.1.2 Semantics

The semantics of FOL formulae are defined using interpretations of the sorts and functions in the signature Σ, and valuations of the variables in Var.

Definition 1.1.4 (Interpretation)**.** An *interpretation* $\mathcal{I}$ for Σ (also called a Σ-*interpretation*) maps each sort symbol $\sigma \in \Sigma^{\mathsf{s}}$ to a non-empty set $\sigma^{\mathcal{I}}$, each function symbol $f^{\sigma_1 \dots \sigma_n \sigma} \in \Sigma^{\mathsf{f}}$ with $n > 0$ to a total function $f^{\mathcal{I}} : \sigma_1^{\mathcal{I}} \times \ldots \times \sigma_n^{\mathcal{I}} \to \sigma^{\mathcal{I}}$, and each constant symbol $c^{\sigma} \in \Sigma^{\mathsf{f}}$ to an element of $\sigma^{\mathcal{I}}$.

Let $\mathcal{I}$ be an interpretation, $f^{\sigma_1 \dots \sigma_n \sigma}$ a function symbol and $\alpha : \sigma_1^{\mathcal{I}} \times \ldots \times \sigma_n^{\mathcal{I}} \to \sigma^{\mathcal{I}}$ a function. We write $\mathcal{I}[f \leftarrow \alpha]$ for an interpretation such that: (i) $\mathcal{I}[f \leftarrow \alpha](\sigma) = \mathcal{I}(\sigma)$ for any sort $\sigma \in \Sigma^{\mathsf{s}}$, (ii) $\mathcal{I}[f \leftarrow \alpha](f) = \alpha$, and (iii) $\mathcal{I}[f \leftarrow \alpha](g) = g^{\mathcal{I}}$ for any $g \in \Sigma^{\mathsf{f}}$ with $g \neq f$. We extend this notation to tuples $\overline{\mathbf{f}} = \langle f_1, \dots, f_n \rangle$ of function symbols and $\overline{\alpha} = \langle \alpha_1, \dots, \alpha_n \rangle$ of functions and write $\mathcal{I}[\overline{\mathbf{f}} \leftarrow \overline{\alpha}]$ for the interpretation $\mathcal{I}[f_1 \leftarrow \alpha_1] \dots [f_n \leftarrow \alpha_n]$.

Definition 1.1.5 (Valuation)**.** Given an interpretation $\mathcal{I}$, a *valuation* ν maps each variable $x^{\sigma} \in \mathsf{Var}$ to an element of $\sigma^{\mathcal{I}}$.

We use $\mathsf{Val} = \bigcup_{\sigma \in \Sigma^{\mathsf{s}}} \sigma^{\mathcal{I}}$ to refer to the set of all possible sort values under the interpretation $\mathcal{I}$. Also, we denote by $\mathcal{V}_{\mathcal{I}}$ the set of all possible valuations under $\mathcal{I}$. Given a valuation ν, a tuple of variables $\overline{\mathbf{x}} = \langle x_1, \dots, x_n \rangle$, and a set of variables $\mathbf{x}$, we write $\nu(\overline{\mathbf{x}})$ for the tuple $\langle \nu(x_1), \dots, \nu(x_n) \rangle$ and $\nu(\mathbf{x})$ for the set $\{\nu(x) \mid x \in \mathbf{x}\}$.

Let $\mathcal{I}$ be an interpretation, $\nu \in \mathcal{V}_{\mathcal{I}}$ a valuation, $x^{\sigma} \in \mathsf{Var}$ a variable, and $\alpha \in \sigma^{\mathcal{I}}$ a value. We write $\nu[x \leftarrow \alpha]$ for a valuation such that: (i) $\nu[x \leftarrow \alpha](x) = \alpha$, and (ii) $\nu[x \leftarrow \alpha](y) = \nu(y)$ for any $y \in \mathsf{Var}$ with $y \neq x$. We extend this notation to tuples $\overline{\mathbf{x}} = \langle x_1, \dots, x_n \rangle$ of variables and $\overline{\alpha} = \langle \alpha_1, \dots, \alpha_n \rangle$ of values, writing $\nu[\overline{\mathbf{x}} \leftarrow \overline{\alpha}]$ for the valuation $\nu[x_1 \leftarrow \alpha_1] \dots [x_n \leftarrow \alpha_n]$.

Definition 1.1.6 (Interpretation of a term)**.** Given an interpretation $\mathcal{I}$ and a valuation $\nu \in \mathcal{V}_{\mathcal{I}}$, we denote by $t_{\nu}^{\mathcal{I}}$ the *interpretation of* t, defined inductively on the structure of t:

$$
\begin{aligned}
x_{\nu}^{\mathcal{I}} &= \nu(x), && x^{\sigma} \in \mathsf{Var} \\
c_{\nu}^{\mathcal{I}} &= c^{\mathcal{I}}, && c^{\sigma} \in \Sigma^{\mathsf{f}} \\
\langle t_1, \dots, t_n \rangle_{\nu}^{\mathcal{I}} &= \langle {t_1}_{\nu}^{\mathcal{I}}, \dots, {t_n}_{\nu}^{\mathcal{I}} \rangle, && t_1^{\sigma_1}, \dots, t_n^{\sigma_n} \text{ terms} \\
(f(t_1, \dots, t_n))_{\nu}^{\mathcal{I}} &= f^{\mathcal{I}}({t_1}_{\nu}^{\mathcal{I}}, \dots, {t_n}_{\nu}^{\mathcal{I}}), && f^{\sigma_1 \dots \sigma_n \sigma} \in \Sigma^{\mathsf{f}},\ t_1^{\sigma_1}, \dots, t_n^{\sigma_n} \text{ terms}
\end{aligned}
$$

In other words, the interpretation of t relative to $\mathcal{I}$ and ν is obtained by replacing each function symbol f occurring in t by its interpretation $f^{\mathcal{I}}$ and each variable x occurring in t by its valuation $\nu(x)$.

Knowing how to interpret terms when given a valuation, as per Definition 1.1.6, we can further extend the notion of interpretation to first-order formulae.

Definition 1.1.7 (Semantics of first-order formulae)**.** Given an interpretation $\mathcal{I}$ and a valuation $\nu \in \mathcal{V}_{\mathcal{I}}$, we write $\mathcal{I}, \nu \models \phi$ if the first-order formula ϕ *is interpreted to true under* $\mathcal{I}$ *and* ν. This relation is defined inductively on the structure of ϕ:

$$
\begin{array}{ll}
\mathcal{I}, \nu \models \top & \text{always holds} \\
\mathcal{I}, \nu \models \bot & \text{never holds} \\
\mathcal{I}, \nu \models t & \text{iff } t_\nu^{\mathcal{I}} = \top,\ t^{\mathsf{Bool}} \text{ term} \\
\mathcal{I}, \nu \models t_1 \approx t_2 & \text{iff } {t_1}_\nu^{\mathcal{I}} = {t_2}_\nu^{\mathcal{I}},\ t_1^\sigma \text{ and } t_2^\sigma \text{ terms} \\
\mathcal{I}, \nu \models \neg\psi & \text{iff } \mathcal{I}, \nu \models \psi \text{ does not hold} \\
\mathcal{I}, \nu \models \psi_1 \wedge \psi_1 & \text{iff } \mathcal{I}, \nu \models \psi_1 \text{ and } \mathcal{I}, \nu \models \psi_2 \\
\mathcal{I}, \nu \models \psi_1 \vee \psi_1 & \text{iff } \mathcal{I}, \nu \models \psi_1 \text{ or } \mathcal{I}, \nu \models \psi_2 \\
\mathcal{I}, \nu \models \exists x \,.\, \psi & \text{iff } \mathcal{I}, \nu[x \leftarrow \alpha] \models \psi,\ x^\sigma \in \mathrm{FV}(\psi), \text{ for some } \alpha \in \sigma^{\mathcal{I}} \\
\mathcal{I}, \nu \models \forall x \,.\, \psi & \text{iff } \mathcal{I}, \nu[x \leftarrow \alpha] \models \psi,\ x^\sigma \in \mathrm{FV}(\psi), \text{ for any } \alpha \in \sigma^{\mathcal{I}}
\end{array}
$$

Using these semantics, we define satisfiability and entailment for FOL formulae under a given interpretation $\mathcal{I}$.

Definition 1.1.8 (Satisfiability and validity)**.** An FOL formula ϕ is *satisfiable* in the interpretation $\mathcal{I}$ if there exists a valuation ν such that $\mathcal{I}, \nu \models \phi$ and *unsatisfiable* otherwise. If $\mathcal{I}, \nu \models \phi$ for any ν, then ϕ is *valid* and $\neg\phi$ is unsatisfiable.

Definition 1.1.9 (Entailment and equivalence)**.** Given FOL formulae ϕ and ψ, we write $\phi \models^{\mathcal{I}} \psi$ and say that ϕ *entails* ψ *in the interpretation* $\mathcal{I}$ if and only if $\mathcal{I}, \nu \models \phi$ implies $\mathcal{I}, \nu \models \psi$, for any valuation ν. We call ϕ and ψ *equivalent* whenever $\phi \models^{\mathcal{I}} \psi$ and $\psi \models^{\mathcal{I}} \phi$.

We encapsulate all the notions pertaining to FOL into a first-order theory.

Definition 1.1.10 (First-order theory)**.** A *first-order theory* is a pair $T = (\Sigma, \mathbf{M})$ such that Σ is a signature and $\mathbf{M}$ is a non-empty set of $(\mathcal{I}, \nu)$ pairs, called the *models* of T, where $\mathcal{I}$ is a Σ-interpretation and $\nu \in \mathcal{V}_{\mathcal{I}}$ is a valuation.

Given a first-order theory $T = (\Sigma, \mathbf{M})$, any Σ-term t can also be called a T-term and, similarly, any Σ-formula ϕ can also be called a T-*formula*. A pair $(\mathcal{I}, \nu) \in \mathbf{M}$ such that $\mathcal{I}, \nu \models \phi$ is a T-*model* of ϕ. We denote the set of all T-models of ϕ by $[\![\phi]\!]_T = \{(\mathcal{I}, \nu) \in \mathbf{M} \mid \mathcal{I}, \nu \models \phi\}$.

Definition 1.1.11 (T-satisfiability and T-validity)**.** Given a first-order theory $T = (\Sigma, \mathbf{M})$, a T-formula ϕ is T-*satisfiable* if $[\![\phi]\!]_T \neq \emptyset$, and T-*unsatisfiable* otherwise. If ϕ is T-satisfiable if and only if ψ is T-satisfiable, then ϕ and ψ are *equisatisfiable in* T. If $[\![\phi]\!]_T = \mathbf{M}$, then ϕ is T-*valid* and $\neg\phi$ is T-unsatisfiable.

Definition 1.1.12 (T-entailment and T-equivalence)**.** Given a first-order theory $T = (\Sigma, \mathbf{M})$ and two T-formulae ϕ and ψ, we write $\phi \models_T \psi$ and say that ϕ T-*entails* ψ if and only if $[\![\phi]\!]_T \subseteq [\![\psi]\!]_T$. We call ϕ and ψ T-*equivalent* whenever $\phi \models_T \psi$ and $\psi \models_T \phi$.

Throughout the rest of Chapter 1 and also in Chapter 2, we only refer to satisfiability and entailment under a specific interpretation (Definitions 1.1.8 and 1.1.9). In Chapter 3, when discussing the satisfiability of separation logic formulae built on top a first-order theory, we refer to the more general Definitions 1.1.11 and 1.1.12, where multiple interpretations can be considered.

1.2 Separation Logic (SL)

1.2.1 Syntax

The syntax for separation logic (SL) is built upon the syntax for FOL, defined in Section 1.1.1. We consider a first-order theory $T = (\Sigma, \mathbf{M})$ such that Σ^{s} contains the sorts Loc and Data, while Σ^{f} contains a special constant $\mathsf{nil}^{\mathsf{Loc}}$.

Definition 1.2.1 (Separation logic formula)**.** A *separation logic formula* over the first-order theory $T = (\Sigma, \mathbf{M})$, also called an $\mathsf{SL}(T)$-formula, is defined recursively by the grammar:

$$
\begin{array}{rlll}
\phi^{\mathsf{SL}} ::= & \top & & \text{(true)} \\
\mid & \bot & & \text{(false)} \\
\mid & t, & t^{\mathsf{Bool}} \text{ is a } T\text{-term} & \text{(boolean term)} \\
\mid & t_1 \approx t_2, & t_1^{\sigma}, t_2^{\sigma} \text{ are } T\text{-terms} & \text{(equality)} \\
\mid & \mathsf{emp} & & \text{(empty heap)} \\
\mid & t \mapsto u, & t^{\mathsf{Loc}}, u^{\mathsf{Data}} \text{ are } T\text{-terms} & \text{(singleton heap)} \\
\mid & \neg\psi^{\mathsf{SL}}, & & \text{(negation)} \\
\mid & \psi_1^{\mathsf{SL}} \wedge \psi_2^{\mathsf{SL}}, & & \text{(conjunction)} \\
\mid & \psi_1^{\mathsf{SL}} \vee \psi_2^{\mathsf{SL}}, & & \text{(disjunction)} \\
\mid & \psi_1^{\mathsf{SL}} * \psi_2^{\mathsf{SL}}, & & \text{(separating conjunction)} \\
\mid & \psi_1^{\mathsf{SL}} \mathrel{-\!\!*} \psi_2^{\mathsf{SL}}, & & \text{(separating implication)} \\
\mid & \exists x \,.\, \psi^{\mathsf{SL}}, & x \in \mathrm{FV}(\psi^{\mathsf{SL}}) & \text{(existential quantification)} \\
\mid & \forall x \,.\, \psi^{\mathsf{SL}}, & x \in \mathrm{FV}(\psi^{\mathsf{SL}}) & \text{(universal quantification)}
\end{array}
$$

The true and false constants, a boolean T-term, and the equality between two T-terms are $\mathsf{SL}(T)$ formulae. The two new atoms describing empty and singleton heaps are also $\mathsf{SL}(T)$ formulae. The negation, conjunction, disjunction, separating conjunction and implication, existential and universal quantification of $\mathsf{SL}(T)$ formulae are $\mathsf{SL}(T)$ formulae as well.

If an $\mathsf{SL}(T)$-formula contains at least one occurrence of emp, $\mapsto$, $*$ or $\mathrel{-\!\!*}$, it is called a *spatial* formula. Otherwise, it is a *pure* formula. We extend the $*$ operator to an iterated version $\circledast$ that can be applied on a set of formulae such that $\circledast\,\emptyset = \mathsf{emp}$ and $\circledast\{\phi_1, \ldots, \phi_n\} = \phi_1 * \ldots * \phi_n$ if $n \geq 1$.

We retain the notations $\mathrm{FV}(\phi)$ ($\mathrm{FV}(F)$) for the free variables of the $\mathsf{SL}(T)$-formula ϕ (set of $\mathsf{SL}(T)$-formulae F), $\exists \overline{\mathbf{x}} \,.\, \phi$ and $\exists \mathbf{x} \,.\, \phi$ ($\forall \overline{\mathbf{x}} \,.\, \phi$ and $\forall \mathbf{x} \,.\, \phi$) for $\exists x_1 \ldots \exists x_n \,.\, \phi$ ($\forall x_1 \ldots \exists x_n \,.\, \phi$) when $\overline{\mathbf{x}} = \langle x_1, \ldots, x_n \rangle$ and $\mathbf{x} = \{x_1, \ldots, x_n\}$, $\phi[\psi]$ for when ψ is a subformula of ϕ, $\phi[\psi'/\psi]$ for the result of replacing ψ with ψ' in ϕ, and also $\phi \Rightarrow \psi$ for $\neg\phi \vee \psi$.

Most definitions of common recursive data structures, such as lists or trees, use a restricted fragment of quantifier-free $\mathsf{SL}(T)$ called *symbolic heaps*, which has a simpler syntax.

Definition 1.2.2 (Symbolic heap formula)**.** A *symbolic heap formula* is a conjunction $\Pi \wedge \Theta$ between a pure (Π) and a spatial (Θ) part, defined as:

$$
\begin{array}{lll}
\Pi ::= \top & & \text{(true)} \\
\quad \mid \bot & & \text{(false)} \\
\quad \mid t, & t^{\mathsf{Bool}} \text{ is a } T\text{-term} & \text{(boolean variable)} \\
\quad \mid t_1 \approx t_2, & t_1^{\mathsf{Loc}}, t_2^{\mathsf{Loc}} \text{ are } T\text{-terms} & \text{(equality)} \\
\quad \mid \neg(t_1 \approx t_2), & t_1^{\mathsf{Loc}}, t_2^{\mathsf{Loc}} \text{ are } T\text{-terms} & \text{(disequality)} \\
\quad \mid \Pi_1 \wedge \Pi_2, & & \text{(conjunction)} \\
\\
\Theta ::= \mathsf{emp} & & \text{(empty heap)} \\
\quad \mid t \mapsto u, & t^{\mathsf{Loc}}, u^{\mathsf{Data}} \text{ are } T\text{-terms} & \text{(singleton heap)} \\
\quad \mid \Theta_1 * \Theta_2, & & \text{(separating conjunction)}
\end{array}
$$

The pure part may consist of true and false constants, a boolean variable, the equality and disequality between two T-terms, and the conjunction of pure parts. The spatial part, on the other hand, may consist of the empty and singleton heaps, as well as the separating conjunction of two spatial parts.

1.2.2 Semantics

In order to define the semantics of $\mathsf{SL}(T)$, we use the same notion of interpretation and valuation described in Definition 1.1.5, but, additionally, we introduce heaps.

Definition 1.2.3 (Heap)**.** Given an interpretation $\mathcal{I}$, a *heap* is a finite partial mapping $h : \mathsf{Loc}^{\mathcal{I}} \rightharpoonup_{\mathsf{fin}} \mathsf{Data}^{\mathcal{I}}$ associating locations with data. We use $\mathsf{Heaps}^{\mathcal{I}}$ to denote the set of all heaps under the interpretation $\mathcal{I}$.

Two heaps h_1 and h_2 are *disjoint* if $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$ and we write $h_1 \# h_2$. In this case, $h_1 \uplus h_2$ denotes their *disjoint union*, which is undefined if h_1 and h_2 are not disjoint. We write $\biguplus H$ for the disjoint union of the heaps in the set $H \subseteq \mathsf{Heaps}$.

Definition 1.2.4 (Semantics of separation logic formulae)**.** Given an interpretation $\mathcal{I}$, a valuation $\nu \in \mathcal{V}_{\mathcal{I}}$ and a heap $h \in \mathsf{Heaps}$, we write $\mathcal{I}, \nu, h \models^{\mathsf{SL}} \phi$ if the SL formula ϕ *is interpreted to true under* $\mathcal{I}$*,* ν *and* h*.* This relation is defined inductively on the structure of ϕ:

$$
\begin{array}{ll}
\mathcal{I}, \nu, h \models^{\mathsf{SL}} \bot & \text{never holds} \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} \top & \text{always holds} \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} t & \text{iff } t_\nu^{\mathcal{I}} = \top,\ t^{\mathsf{Bool}} \text{ is a } T\text{-term} \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} t_1 \approx t_2 & \text{iff } {t_1}_\nu^{\mathcal{I}} = {t_2}_\nu^{\mathcal{I}},\ t_1^\sigma, t_2^\sigma \text{ are } T\text{-terms} \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} \mathsf{emp} & \text{iff } \mathrm{dom}(h) = \emptyset \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} t \mapsto u & \text{iff } t_\nu^{\mathcal{I}} \neq \mathsf{nil}^{\mathcal{I}} \text{ and } h = \left\{(t_\nu^{\mathcal{I}}, u_\nu^{\mathcal{I}})\right\},\ t^{\mathsf{Loc}}, u^{\mathsf{Loc}} \text{ are } T\text{-terms} \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} \neg\psi & \text{iff } \mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi \text{ does not hold} \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi_1 \wedge \psi_1 & \text{iff } \mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi_1 \text{ and } \mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi_2 \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi_1 \vee \psi_1 & \text{iff } \mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi_1 \text{ or } \mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi_2 \\
\mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi_1 * \psi_2 & \text{iff } \exists h_1 \exists h_2 \,.\, h = h_1 \uplus h_2 \text{ and } \mathcal{I}, \nu, h_1 \models^{\mathsf{SL}} \psi_1 \text{ and } \mathcal{I}, \nu, h_2 \models^{\mathsf{SL}} \psi_2
\end{array}
$$

$$\mathcal{I}, \nu, h \models^{\mathsf{SL}} \psi_1 \mathrel{-\!\!*} \psi_2 \quad \text{iff } \forall h_0 \,.\, h \# h_0 \text{ and } \mathcal{I}, \nu, h_0 \models^{\mathsf{SL}} \psi_1 \text{ imply that } \mathcal{I}, \nu, h \uplus h_0 \models^{\mathsf{SL}} \psi_2$$
$$\mathcal{I}, \nu, h \models^{\mathsf{SL}} \exists x \,.\, \psi \quad \text{iff } \mathcal{I}, \nu[x \leftarrow v], h \models^{\mathsf{SL}} \psi,\ x^\sigma \in \mathrm{FV}(\psi), \text{ for some } v \in \sigma^{\mathcal{I}}$$
$$\mathcal{I}, \nu, h \models^{\mathsf{SL}} \forall x \,.\, \psi \quad \text{iff } \mathcal{I}, \nu[x \leftarrow v], h \models^{\mathsf{SL}} \psi,\ x^\sigma \in \mathrm{FV}(\psi), \text{ for any } v \in \sigma^{\mathcal{I}}$$

A triple $(\mathcal{I}, \nu, h)$ such that $(\mathcal{I}, \nu) \in \mathbf{M}$ and $\mathcal{I}, \nu, h \models^{\mathsf{SL}} \phi$ is an $\mathsf{SL}(T)$-*model* for the $\mathsf{SL}(T)$-formula ϕ. We denote the set of all $\mathsf{SL}(T)$-models of ϕ by $[\![\phi]\!]_{\mathsf{SL}(T)} = \{(\mathcal{I}, \nu, h) \mid (\mathcal{I}, \nu) \in \mathbf{M}, h \in \mathsf{Heaps}^{\mathcal{I}} \text{ and } \mathcal{I}, \nu, h \models^{\mathsf{SL}} \phi\}$. Using these notations and the above semantics, we define satisfiability and entailment for $\mathsf{SL}(T)$-formulae.

Definition 1.2.5 ($\mathsf{SL}(T)$-satisfiability and $\mathsf{SL}(T)$-validity)**.** Given a first-order theory $T = (\Sigma, \mathbf{M})$, an $\mathsf{SL}(T)$-formula ϕ is $\mathsf{SL}(T)$-*satisfiable* if $[\![\phi]\!]_{\mathsf{SL}(T)} \neq \emptyset$, and $\mathsf{SL}(T)$-*unsatisfiable* otherwise. If ϕ is $\mathsf{SL}(T)$-satisfiable if and only if ψ is $\mathsf{SL}(T)$-satisfiable, then ϕ and ψ are *equisatisfiable in* $\mathsf{SL}(T)$. If $[\![\phi]\!]_{\mathsf{SL}(T)} = \mathbf{M}$, then ϕ is $\mathsf{SL}(T)$-*valid* and $\neg\phi$ is $\mathsf{SL}(T)$-unsatisfiable.

Definition 1.2.6 ($\mathsf{SL}(T)$-entailment and $\mathsf{SL}(T)$-equivalence)**.** Given a first-order theory $T = (\Sigma, \mathbf{M})$ and two $\mathsf{SL}(T)$-formulae ϕ and ψ, we write $\phi \models^{\mathsf{SL}}_T \psi$ and say that ϕ $\mathsf{SL}(T)$-*entails* ψ if and only if $[\![\phi]\!]_{\mathsf{SL}(T)} \subseteq [\![\psi]\!]_{\mathsf{SL}(T)}$. We call ϕ and ψ $\mathsf{SL}(T)$-*equivalent* whenever $\phi \models^{\mathsf{SL}}_T \psi$ and $\psi \models^{\mathsf{SL}}_T \phi$.

For the notions discussed throughout the rest of Chapter 1 and also in Chapter 2, we consider a restricted fragment of $\mathsf{SL}(T)$. The first-order theory $T = (\Sigma, \mathbf{M})$ is fixed such that $\Sigma^{\mathsf{s}} = \{\mathsf{Loc}, \mathsf{Data}, \mathsf{Bool}\}$, $\Sigma^{\mathsf{f}} = \{\mathsf{nil}^{\mathsf{Loc}}\}$ and $\mathsf{Data} = \mathsf{Loc}^k$ with $k \geq 1$ also fixed. Terms are reduced to just nil and variables, while $\mathcal{T}_\Sigma(\mathbf{x}) = \mathbf{x}$ for any $\mathbf{x} \subseteq \mathsf{Var}$. Moreover, we also fix an interpretation $\mathcal{I}$, where $\mathcal{I}(\mathsf{Loc}) = \mathsf{L}$ is a countably infinite set containing a value ℓ_{nil} such that $\mathsf{nil}^{\mathcal{I}} = \ell_{\mathsf{nil}}$. Then $\mathbf{M} = \{(\mathcal{I}, \nu) \mid \nu \in \mathcal{V}_{\mathcal{I}}\}$. In this context, we omit to specify T or $\mathcal{I}$ any further and we use simpler definitions for satisfiability and equivalence, closer to their FOL equivalents described by Definitions 1.1.8 and 1.1.9.

Definition 1.2.7 (Satisfiability and validity)**.** An SL formula ϕ is *satisfiable* if there exists a valuation ν and a heap h such that $\nu, h \models^{\mathsf{SL}} \phi$ and *unsatisfiable* otherwise. If $\nu, h \models^{\mathsf{SL}} \phi$ for any ν and h, then ϕ is *valid* and $\neg\phi$ is unsatisfiable.

Definition 1.2.8 (Entailment and equivalence)**.** Given SL formulae ϕ and ψ, we write $\phi \models^{\mathsf{SL}} \psi$ and say that ϕ *entails* ψ if and only if $\nu, h \models^{\mathsf{SL}} \phi$ implies $\nu, h \models^{\mathsf{SL}} \psi$, for any valuation ν and heap h. We call ϕ and ψ *equivalent* whenever $\phi \models^{\mathsf{SL}} \psi$ and $\psi \models^{\mathsf{SL}} \phi$.

1.3 Systems of Inductive Definitions

Throughout this work we will use the phrases "system of inductive (predicate) definitions" and "inductive system (of predicates)" interchangeably. To prevent confusion with the proof systems introduced in Chapter 2, we will avoid using the word "system" by itself.

We can define inductive systems using both first-order and separation logic. The only syntactical difference lies in the types of formulae accepted as constraints for the predicate rules of an inductive system, which are specific to the underlying logic. As a result, we will describe the syntax (Section 1.3.1) in the context of any signature $\Sigma = (\Sigma^{\mathsf{s}}, \Sigma^{\mathsf{f}})$ and refer to formulae without specifying their type (FOL or SL). The semantics of inductive systems, however, are highly reliant on formula semantics and on the valuation of predicate rules (e.g. using either classic or separating conjunction). As such, we independently describe the semantics for inductive systems in FOL (Section 1.3.2) and in SL (Section 1.3.3).

1.3.1 Syntax

Consider a signature $\Sigma = (\Sigma^{\mathsf{s}}, \Sigma^{\mathsf{f}})$ and let Pred be a countable set of *predicate symbols*. For a predicate symbol $p^{\sigma_1 \dots \sigma_n} \in \mathsf{Pred}$, $n \geq 1$ is its arity and $\sigma_1, \dots, \sigma_n \in \Sigma^{\mathsf{s}}$ are the sorts of its arguments. Given a tuple of terms $\langle t_1^{\sigma_1}, \dots, t_n^{\sigma_n} \rangle$, we call $p(t_1, \dots, t_n)$ a *predicate atom*. We denote by $\mathsf{Atom} = \{p(t_1, \dots, t_n) \mid p^{\sigma_1, \dots, \sigma_n} \in \mathsf{Pred}, t_i^{\sigma_i} \in \mathcal{T}_\Sigma(\mathsf{Var}) \text{ for each } i \in [n]\}$ the set of all possible predicate atoms.

Definition 1.3.1 (Predicate rule)**.** A *predicate rule* is a pair

$$\langle p(\overline{\mathbf{x}}), \{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_n), q_1(\overline{\mathbf{x}}_1), \dots, q_n(\overline{\mathbf{x}}_n)\} \rangle, n \geq 0$$

where $\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_n$ are unique tuples of variables such that $\mathsf{set}(\overline{\mathbf{x}}), \mathsf{set}(\overline{\mathbf{x}}_1), \dots, \mathsf{set}(\overline{\mathbf{x}}_n)$ are pairwise disjoint sets. Then ϕ is a formula called the *constraint*, $p(\overline{\mathbf{x}})$ is a predicate atom called the *goal*, and $q_1(\overline{\mathbf{x}}_1), \dots, q_n(\overline{\mathbf{x}}_n)$ are predicate atoms called *subgoals*. Consequently, the variables in $\overline{\mathbf{x}}$ are called *goal variables*, whereas the ones in $\bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i)$ are *subgoal variables*. Equalities between the variables in $\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_n$ may be captured by the constraint. We refer to $\{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_n), q_1(\overline{\mathbf{x}}_1), \dots, q_n(\overline{\mathbf{x}}_n)\}$ as the *body* of the predicate rule.

Whenever we consider a predicate rule $\langle p(\overline{\mathbf{x}}), \{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_n), q_1(\overline{\mathbf{x}}_1), \dots, q_n(\overline{\mathbf{x}}_n)\} \rangle$, it is implied that $\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_n$ respect the conditions from Definition 1.3.1. We often refer to a predicate rule body $R = \{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_n), q_1(\overline{\mathbf{x}}_1), \dots, q_n(\overline{\mathbf{x}}_n)\}$ by fixing specific tuples of goal and subgoal variables and writing $R(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_n)$ or, more compactly, $R(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, where $\overline{\mathbf{y}} = \overline{\mathbf{x}}_1 \cdot \ldots \cdot \overline{\mathbf{x}}_n$. Regardless of the variables with which R was initially specified, by $R(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ we mean the variant of R in which $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ replace the goal and subgoal variables, respectively. By writing only $R(\overline{\mathbf{x}})$ or R we consider all variants of R with fixed goal variables $\overline{\mathbf{x}}$ or without any fixed variables, respectively.

Definition 1.3.2 (Inductive system of predicates)**.** An *inductive system of predicates* $\mathcal{S}$ is a finite set of predicate rules.

Without loss of generality, we assume there are no goals with the same predicate and different goal variables. We use the condensed notation $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \dots \mid R_m(\overline{\mathbf{x}})$ when $\{\langle p(\overline{\mathbf{x}}), R_1(\overline{\mathbf{x}}) \rangle, \dots, \langle p(\overline{\mathbf{x}}), R_m(\overline{\mathbf{x}}) \rangle\}$ is the set of all predicate rules with goal $p(\overline{\mathbf{x}})$ in $\mathcal{S}$. We also write $\mathcal{S}^{\mathsf{p}}$ and $\mathcal{S}^{\mathsf{c}}$ for the sets of predicate symbols and of constraints, respectively, that occur in the rules of $\mathcal{S}$.

Definition 1.3.3 (Size of an inductive system)**.** The *size of an inductive system* $\mathcal{S}$ is the sum of the sizes of all constraints occurring in its predicate rules:

$$|\mathcal{S}| = \sum_{\phi \in \mathcal{S}^{\mathsf{c}}} |\phi|$$

1.3.2 Solution of an Inductive System in FOL

Let $\mathcal{S}$ be an inductive system in FOL and $\mathcal{I}$ an interpretation for the sorts and function symbols in $\mathcal{S}$. An *assignment* $\mathcal{X}$ maps each predicate $p^{\sigma_1 \dots \sigma_n} \in \mathcal{S}^{\mathsf{p}}$ to a set $\mathcal{X}(p) \subseteq \sigma_1^{\mathcal{I}} \times \dots \times \sigma_n^{\mathcal{I}}$. We extend the application of $\mathcal{X}$ to a predicate atom $p(t_1, \dots, t_n)$ such that

$$\mathcal{X}(p(t_1, \dots, t_n)) = \{\nu \mid (t_{1\nu}^{\mathcal{I}}, \dots, t_{n\nu}^{\mathcal{I}}) \in \mathcal{X}(p)\}$$

to a set $F = \{\phi_1, \dots \phi_k, q_1(\overline{\mathbf{x}}_1), \dots, q_n(\overline{\mathbf{x}}_n)\}$ of formulae and predicate atoms such that

$$\mathcal{X}(\bigwedge F) = \{\nu \mid \mathcal{I}, \nu \models \phi_1 \wedge \dots \wedge \phi_k \text{ and } \nu \in \mathcal{X}(q_i(\overline{\mathbf{x}}_i)), \forall i \in [n]\}$$

and to a predicate rule $\langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle$ such that

$$\mathcal{X}(\bigwedge R(\overline{\mathbf{x}})) = \{\nu \mid \nu(\overline{\mathbf{x}}) = \gamma(\overline{\mathbf{x}}) \text{ and } \gamma \in \mathcal{X}(\bigwedge R(\overline{\mathbf{x}}, \overline{\mathbf{y}}))\}$$

Given two sets F_1 and F_2 of formulae and predicate atoms, the following holds:

$$\mathcal{X}(\bigwedge F_1 \wedge \bigwedge F_2) = \mathcal{X}(\bigwedge F_1) \cap \mathcal{X}(\bigwedge F_2)$$

and we also define the application of $\mathcal{X}$ on a disjunction of such sets

$$\mathcal{X}(\bigwedge F_1 \vee \bigwedge F_2) = \mathcal{X}(\bigwedge F_1) \cup \mathcal{X}(\bigwedge F_2)$$

The set of all assignments on the predicates in $\mathcal{S}$ is $\mathsf{Assign}(\mathcal{S}^{\mathrm{p}})$. We identify two special elements of this set: $\mathcal{X}_\emptyset$ as the assignment that maps all predicates in $\mathcal{S}^{\mathrm{p}}$ to the empty set, and $\mathcal{X}_\sigma$ as the assignment that maps each predicate $p^{\sigma_1 \cdots \sigma_n} \in \mathcal{S}^{\mathrm{p}}$ to the set $\sigma_1^{\mathcal{I}} \times \ldots \times \sigma_n^{\mathcal{I}}$.

Given $\mathcal{X}_1, \mathcal{X}_2 \in \mathsf{Assign}(\mathcal{S}^{\mathrm{p}})$, we introduce the relation $\preccurlyeq$, together with its strict variant $\prec$, such that

$$\mathcal{X}_1 \preccurlyeq \mathcal{X}_2 \Leftrightarrow \forall p \in \mathcal{S}^{\mathrm{p}} \,.\, \mathcal{X}_1(p) \subseteq \mathcal{X}_2(p)$$
$$\mathcal{X}_1 \prec \mathcal{X}_2 \Leftrightarrow \mathcal{X}_1 \preccurlyeq \mathcal{X}_2 \wedge \mathcal{X}_1 \neq \mathcal{X}_2$$

We define the union of two assignments $\mathcal{X}_1, \mathcal{X}_2 \in \mathsf{Assign}(\mathcal{S}^{\mathrm{p}})$ as

$$(\mathcal{X}_1 \curlyvee \mathcal{X}_2)(p) = \mathcal{X}_1(p) \cup \mathcal{X}_2(p), \forall p \in \mathcal{S}^{\mathrm{p}}$$

and their intersection as

$$(\mathcal{X}_1 \curlywedge \mathcal{X}_2)(p) = \mathcal{X}_1(p) \cap \mathcal{X}_2(p), \forall p \in \mathcal{S}^{\mathrm{p}}$$

We write $\bigvee X$ and $\bigwedge X$ for the application of $\curlyvee$ and $\curlywedge$, respectively, among the elements of the set $X \subseteq \mathsf{Assign}(\mathcal{S}^{\mathrm{p}})$.

Partial orderings and complete lattices. We briefly go over some notions about partial orderings and complete lattices that are relevant for assignments, as they are described in [36, Appendix A]. A *partial ordering* on a set L is a relation $\leq_{\mathrm{L}} \subseteq L \times L$ that is *reflexive* (i.e. $\forall l \in L \,.\, l \leq_{\mathrm{L}} l$), *transitive* (i.e. $\forall l_1, l_2, l_3 \in L \,.\, l_1 \leq_{\mathrm{L}} l_2 \wedge l_2 \leq_{\mathrm{L}} l_3 \Rightarrow l_1 \leq_{\mathrm{L}} l_3$) and *anti-symmetric* (i.e. $\forall l_1, l_2 \in L \,.\, l_1 \leq_{\mathrm{L}} l_2 \wedge l_2 \leq_{\mathrm{L}} l_1 \Rightarrow l_1 = l_2$). A *partially ordered set* $(L, \leq_{\mathrm{L}})$ is a set L equipped with a partial ordering $\leq_{\mathrm{L}}$. A set $Y \subseteq L$ has a *lower bound* l if $\forall l' \in Y \,.\, l \leq_{\mathrm{L}} l'$. The *greatest lower bound* of Y is a lower bound l_0 such that $l \leq_{\mathrm{L}} l_0$ for any other lower bound l of Y. Conversely, $Y \subseteq L$ has an *upper bound* l if $\forall l' \in Y \,.\, l' \leq_{\mathrm{L}} l$, and the *least upper bound* of Y is an upper bound l_0 such that $l_0 \leq_{\mathrm{L}} l$ for any other upper bound l of Y. Not all subsets of a partially ordered set L need to have an greatest lower bound or a least upper bound, but, due to $\leq_{\mathrm{L}}$ being anti-symmetric, they are unique when they do exist and are denoted $\bigsqcap Y$ and $\bigsqcup Y$, where $\bigsqcap$ is called the *meet* operator and $\bigsqcup$ the *join* operator. A *complete lattice* $L = (L, \leq_{\mathrm{L}}) = (L, \leq_{\mathrm{L}}, \bigsqcup, \bigsqcap, \bot_{\mathrm{L}}, \top_{\mathrm{L}})$ is a partially ordered set such that any of its subsets has a least upper bound and greatest lower bound. Furthermore, $\bot_{\mathrm{L}} = \bigsqcup \emptyset = \bigsqcap L$ is the least element of L and $\top_{\mathrm{L}} = \bigsqcap \emptyset = \bigsqcup L$ is the greatest element of L.

A classic example of complete lattice is the powerset of any set S together with the set inclusion relation – $(\mathcal{P}(S), \subseteq, \bigcup, \bigcap, \emptyset, S)$. The join and meet operators are set union and intersection, respectively. The least element of $\mathcal{P}(S)$ is $\emptyset$ and its greatest element is S.

Since $\preccurlyeq$, $\bigvee$ and $\bigwedge$ are defined pointwise based on set inclusion, union and intersection, respectively, it is easy to show that $\preccurlyeq$ is a partial ordering and that $\mathsf{Assign}(\mathcal{S}^{\mathsf{P}}) = (\mathsf{Assign}(\mathcal{S}^{\mathsf{P}}), \preccurlyeq) = (\mathsf{Assign}(\mathcal{S}^{\mathsf{P}}), \preccurlyeq, \bigvee, \bigwedge, \mathcal{X}_\emptyset, \mathcal{X}_\sigma)$ is a complete lattice.

The relation $\preccurlyeq$ is reflexive because, given any $\mathcal{X} \in \mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$,

$$\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \mathcal{X}(p) \subseteq \mathcal{X}(p) \Leftrightarrow \mathcal{X} \preccurlyeq \mathcal{X}$$

The relation $\preccurlyeq$ is transitive because, for any $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3 \in \mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$,

$$\begin{aligned}
&\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \big(\mathcal{X}_1(p) \subseteq \mathcal{X}_2(p) \wedge \mathcal{X}_2(p) \subseteq \mathcal{X}_3(p) \Rightarrow \mathcal{X}_1(p) \subseteq \mathcal{X}_3(p)\big) \Rightarrow \\
&\big(\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \mathcal{X}_1(p) \subseteq \mathcal{X}_2(p)\big) \wedge \big(\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \mathcal{X}_2(p) \subseteq \mathcal{X}_3(p)\big) \Rightarrow \\
&\big(\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \mathcal{X}_1(p) \subseteq \mathcal{X}_3(p)\big) \Leftrightarrow \mathcal{X}_1 \preccurlyeq \mathcal{X}_2 \wedge \mathcal{X}_2 \preccurlyeq \mathcal{X}_3 \Rightarrow \mathcal{X}_1 \preccurlyeq \mathcal{X}_3
\end{aligned}$$

The relation $\preccurlyeq$ is anti-symmetric because, for any $\mathcal{X}_1, \mathcal{X}_2 \in \mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$,

$$\begin{aligned}
&\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \big(\mathcal{X}_1(p) \subseteq \mathcal{X}_2(p) \wedge \mathcal{X}_2(p) \subseteq \mathcal{X}_1(p) \Rightarrow \mathcal{X}_1(p) = \mathcal{X}_2(p)\big) \Rightarrow \\
&\big(\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \mathcal{X}_1(p) \subseteq \mathcal{X}_2(p)\big) \wedge \big(\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \mathcal{X}_2(p) \subseteq \mathcal{X}_1(p)\big) \Rightarrow \\
&\big(\forall p \in \mathcal{S}^{\mathsf{P}} \,.\, \mathcal{X}_1(p) = \mathcal{X}_2(p)\big) \Leftrightarrow \mathcal{X}_1 \preccurlyeq \mathcal{X}_2 \wedge \mathcal{X}_2 \preccurlyeq \mathcal{X}_1 \Rightarrow \mathcal{X}_1 = \mathcal{X}_2
\end{aligned}$$

Thus, $(\mathsf{Assign}(\mathcal{S}^{\mathsf{P}}), \preccurlyeq)$ is partially ordered set. For any $X \subseteq \mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$, its greatest lower bound is $\bigwedge X$ and its least greater bound is $\bigvee X$. It follows that $(\mathsf{Assign}(\mathcal{S}^{\mathsf{P}}), \preccurlyeq)$ is a complete lattice with $\mathcal{X}_\emptyset = \bigvee \emptyset = \bigwedge \mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$ and $\mathcal{X}_\sigma = \bigvee \mathsf{Assign}(\mathcal{S}^{\mathsf{P}}) = \bigwedge \emptyset$ as its least and greatest elements, respectively.

The inductive system $\mathcal{S}$ and the interpretation $\mathcal{I}$ induce a function on assignments, $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}} : \mathsf{Assign}(\mathcal{S}^{\mathsf{P}}) \to \mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$, such that

$$\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X})(p) = \bigcup_{i=1}^{m} \{\nu(\overline{\mathbf{x}}) \mid \nu \in \mathcal{X}(\bigwedge R_i(\overline{\mathbf{x}}))\}$$

where $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_m(\overline{\mathbf{x}})$.

Definition 1.3.4 (Solution of an FOL inductive system)**.** A *solution* of $\mathcal{S}$ is an assignment $\mathcal{X} \in \mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$ such that $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}) \preccurlyeq \mathcal{X}$. The set of all solutions of $\mathcal{S}$ is $\mathsf{Sol}^{\mathcal{I}}_{\mathcal{S}} = \{\mathcal{X} \mid \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}) \preccurlyeq \mathcal{X}\}$. A *least solution* of $\mathcal{S}$ is $\mu\mathcal{S}^{\mathcal{I}} \in \mathsf{Sol}^{\mathcal{I}}_{\mathcal{S}}$ such that, for any assignment $\mathcal{X} \prec \mu\mathcal{S}^{\mathcal{I}}$, $\mathcal{X} \notin \mathsf{Sol}^{\mathcal{I}}_{\mathcal{S}}$.

Lemma 1.3.1. The extension of an assignment is monotonically increasing, i.e. if $\mathcal{X}_1, \mathcal{X}_2 \in \mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$ such that $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$, then

$$\mathcal{X}_1(p(t_1, \ldots, t_n)) \subseteq \mathcal{X}_2(p(t_1, \ldots, t_n)) \text{ for any } p(t_1, \ldots, t_n) \in \mathsf{Atom} \text{ and } p \in \mathcal{S}^{\mathsf{P}} \tag{1.1}$$

$$\mathcal{X}_1\left(\bigwedge R(\overline{\mathbf{x}})\right) \subseteq \mathcal{X}_2\left(\bigwedge R(\overline{\mathbf{x}})\right) \text{ for any } \langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S} \tag{1.2}$$

Proof. Since $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$, it follows that $\mathcal{X}_1(p) \subseteq \mathcal{X}_2(p), \forall p \in \mathcal{S}^{\mathsf{P}}$. Therefore,

$$\begin{aligned}
\mathcal{X}_1(p(t_1, \ldots, t_n)) &= \{\nu \mid (t_1{}^{\mathcal{I}}_{\nu}, \ldots, t_n{}^{\mathcal{I}}_{\nu}) \in \mathcal{X}_1(p)\} \\
&\subseteq \{\nu \mid (t_1{}^{\mathcal{I}}_{\nu}, \ldots, t_n{}^{\mathcal{I}}_{\nu}) \in \mathcal{X}_2(p)\} = \mathcal{X}_2(p(t_1, \ldots, t_n))
\end{aligned}$$

for any $p(t_1, \ldots, t_n) \in \mathsf{Atom}$ with $p \in \mathcal{S}^{\mathsf{P}}$. We have successfully proven (1.1).

Let $R(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n), q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}$ such that $\langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S}$ and $\overline{\mathbf{y}} = \overline{\mathbf{x}}_1 \cdot \ldots \cdot \overline{\mathbf{x}}_n$. By the extension of assignments, for any $i \in [2]$ we obtain

$$\begin{aligned}\mathcal{X}_i\left(\bigwedge R(\overline{\mathbf{x}}, \overline{\mathbf{y}})\right) &= \{\nu \mid \mathcal{I}, \nu \models \phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}) \text{ and } \nu \in \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)), \forall j \in [n]\} \\ &= \{\nu \mid \mathcal{I}, \nu \models \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})\} \cap \{\nu \mid \nu \in \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)), \forall j \in [n]\} \\ &= \{\nu \mid \mathcal{I}, \nu \models \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})\} \cap \bigcap_{j=1}^{n} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)) \end{aligned} \tag{1.3}$$

Since $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$, it follows from (1.1) that $\mathcal{X}_1(q_j(\overline{\mathbf{x}}_j)) \subseteq \mathcal{X}_2(q_j(\overline{\mathbf{x}}_j)), \forall j \in [n]$. Therefore,

$$\bigcap_{j=1}^{n} \mathcal{X}_1(q_j(\overline{\mathbf{x}}_j)) \subseteq \bigcap_{j=1}^{n} \mathcal{X}_2(q_j(\overline{\mathbf{x}}_j)) \tag{1.4}$$

Because $\{\nu \mid \mathcal{I}, \nu \models \phi\} \subseteq \{\nu \mid \mathcal{I}, \nu \models \phi\}$ holds trivially, it easily follows from (1.3) and (1.4) that

$$\{\nu \mid \mathcal{I}, \nu \models \phi\} \cap \bigcap_{j=1}^{n} \mathcal{X}_1(q_j(\overline{\mathbf{x}}_j)) \subseteq \{\nu \mid \mathcal{I}, \nu \models \phi\} \cap \bigcap_{j=1}^{n} \mathcal{X}_2(q_j(\overline{\mathbf{x}}_j))$$

and, thus, that $\mathcal{X}_1(\bigwedge R(\overline{\mathbf{x}}, \overline{\mathbf{y}})) \subseteq \mathcal{X}_2(\bigwedge R(\overline{\mathbf{x}}, \overline{\mathbf{y}}))$. We also obtain $\mathcal{X}_1(\bigwedge R(\overline{\mathbf{x}})) \subseteq \mathcal{X}_2(\bigwedge R(\overline{\mathbf{x}}))$. As R was chosen arbitrarily, we conclude that (1.2) was successfully proven. □

Theorem 1.3.2 (Monotonicity of $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$)**.** The function $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}} : \mathsf{Assign}(\mathcal{S}^{\mathsf{p}}) \to \mathsf{Assign}(\mathcal{S}^{\mathsf{p}})$, induced by the inductive system $\mathcal{S}$ and the FOL interpretation $\mathcal{I}$, is monotonically increasing, i.e $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_1) \preccurlyeq \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_2)$ if $\mathcal{X}_1, \mathcal{X}_2 \in \mathsf{Assign}(\mathcal{S}^{\mathsf{p}})$ such that $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$.

Proof. Let $p \in \mathcal{S}^{\mathsf{p}}$ be any predicate symbol defined by the inductive system $\mathcal{S}$ such that $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_m(\overline{\mathbf{x}})$. Then, as per the definition of $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$,

$$\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_i)(p) = \bigcup_{j=1}^{m} \{\nu(\overline{\mathbf{x}}) \mid \nu \in \mathcal{X}_i\left(\bigwedge R_j(\overline{\mathbf{x}})\right)\} = \{\nu(\overline{\mathbf{x}}) \mid \nu \in \bigcup_{j=1}^{m} \mathcal{X}_i\left(\bigwedge R_j(\overline{\mathbf{x}})\right)\}, \forall i \in [2] \tag{1.5}$$

Because $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$, it follows from Lemma 1.3.1 that

$$\mathcal{X}_1\left(\bigwedge R_j(\overline{\mathbf{x}})\right) \subseteq \mathcal{X}_2\left(\bigwedge R_j(\overline{\mathbf{x}})\right), \forall j \in [m] \Rightarrow \bigcup_{j=1}^{m} \mathcal{X}_1\left(\bigwedge R_j(\overline{\mathbf{x}})\right) \subseteq \bigcup_{j=1}^{m} \mathcal{X}_2\left(\bigwedge R_j(\overline{\mathbf{x}})\right) \tag{1.6}$$

From (1.5) and (1.6) we can quickly gather that

$$\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_1)(p) = \{\nu(\overline{\mathbf{x}}) \mid \nu \in \bigcup_{j=1}^{m} \mathcal{X}_1\left(\bigwedge R_j(\overline{\mathbf{x}})\right)\} \subseteq \{\nu(\overline{\mathbf{x}}) \mid \nu \in \bigcup_{j=1}^{m} \mathcal{X}_2\left(\bigwedge R_j(\overline{\mathbf{x}})\right)\} = \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_2)(p)$$

As p was chosen arbitrarily, we can conclude that $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_1) \preccurlyeq \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_2)$. □

Ascending chains. To make a case for the continuity of $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$, we introduce ascending chains, as they are defined in [36, Appendix A]. Given a partially ordered set $(L, \leq_{\mathrm{L}})$, a subset $Y \subseteq L$ is a chain if $\forall l_1, l_2 \in Y \,.\, l_1 \leq_{\mathrm{L}} l_2$ or $l_2 \leq_{\mathrm{L}} l_1$, i.e. a chain is a (possibly empty) subset of L that is totally ordered. A sequence $(l_i)_{i\in\mathbb{N}}$ of elements from L is an *ascending chain* if $\forall i, i' \in \mathbb{N} \,.\, i \leq i' \Rightarrow l_i \leq_{\mathrm{L}} l_{i'}$.

In the following lemma and theorem, we use ascending chains of assignments from $\mathsf{Assign}(\mathcal{S}^{\mathsf{p}})$, denoted $(\mathcal{X}_i)_{i\in\mathbb{N}}$, where, as per the above definition, $\forall i, i' \in \mathbb{N} \,.\, i \leq i' \Rightarrow \mathcal{X}_i \preccurlyeq \mathcal{X}_{i'}$.

Lemma 1.3.3. The extension of an assignment is continuous, i.e. if $(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain in $\mathsf{Assign}(\mathcal{S}^{\mathsf{p}})$, then

$$\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(p(t_1,\ldots,t_n)) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(p(t_1,\ldots,t_n)), \text{ for any } p(t_1,\ldots,t_n)\in\mathsf{Atom}, p\in\mathcal{S}^{\mathsf{p}} \quad (1.7)$$

$$\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)\left(\bigwedge R(\overline{\mathbf{x}})\right) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i\left(\bigwedge R(\overline{\mathbf{x}})\right), \text{ for any } \langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S} \quad (1.8)$$

Proof. By the extension of assignments and the definition of assignment union,

$$\begin{aligned}\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(p(t_1,\ldots,t_n)) &= \{\nu \mid (t_1{}^{\mathcal{I}}_{\nu},\ldots,t_n{}^{\mathcal{I}}_{\nu}) \in \left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(p)\}\\ &= \{\nu \mid (t_1{}^{\mathcal{I}}_{\nu},\ldots,t_n{}^{\mathcal{I}}_{\nu}) \in \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(p)\}\\ &= \bigcup_{i\in\mathbb{N}}\{\nu \mid (t_1{}^{\mathcal{I}}_{\nu},\ldots,t_n{}^{\mathcal{I}}_{\nu}) \in \mathcal{X}_i(p)\} = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(p(t_1,\ldots,t_n))\end{aligned}$$

for any $p(t_1,\ldots,t_n)\in\mathsf{Atom}$ with $p\in\mathcal{S}^{\mathsf{p}}$. We have successfully proven (1.1).

Let $R(\overline{\mathbf{x}},\overline{\mathbf{y}}) = \{\phi(\overline{\mathbf{x}},\overline{\mathbf{x}}_1,\ldots,\overline{\mathbf{x}}_n), q_1(\overline{\mathbf{x}}_1),\ldots,q_n(\overline{\mathbf{x}}_n)\}$ such that $\langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S}$ and $\overline{\mathbf{y}} = \overline{\mathbf{x}}_1\cdot\ldots\cdot\overline{\mathbf{x}}_n$. By the extension of assignments,

$$\begin{aligned}\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)\left(\bigwedge R(\overline{\mathbf{x}},\overline{\mathbf{y}})\right) &= \{\nu \mid \mathcal{I},\nu\models\phi(\overline{\mathbf{x}},\overline{\mathbf{y}}) \text{ and } \nu\in\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(q_j(\overline{\mathbf{x}}_j)), \forall j\in[n]\}\\ &= \{\nu \mid \mathcal{I},\nu\models\phi(\overline{\mathbf{x}},\overline{\mathbf{y}})\} \cap \{\nu \mid \nu\in\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(q_j(\overline{\mathbf{x}}_j)), \forall j\in[n]\}\\ &= \{\nu \mid \mathcal{I},\nu\models\phi(\overline{\mathbf{x}},\overline{\mathbf{y}})\} \cap \bigcap_{j=1}^{n}\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(q_j(\overline{\mathbf{x}}_j)) \quad (1.9)\end{aligned}$$

$(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain, so it follows from (1.7) that $\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(q_j(\overline{\mathbf{x}}_j)) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))$ and we can rewrite (1.9) as

$$\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)\left(\bigwedge R(\overline{\mathbf{x}},\overline{\mathbf{y}})\right) = \{\nu \mid \mathcal{I},\nu\models\phi(\overline{\mathbf{x}},\overline{\mathbf{y}})\} \cap \bigcap_{j=1}^{n}\bigcup_{i\in\mathbb{N}} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)) \quad (1.10)$$

Moreover, since $(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain and the extension of assignments is monotonically increasing, as per Lemma 1.3.1, then $\forall i,i'\in\mathbb{N}\,.\,i\leq i'\Rightarrow \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))\subseteq\mathcal{X}_{i'}(q_j(\overline{\mathbf{x}}_j))$ and $\bigcap_{j=1}^{n}\bigcup_{i\in\mathbb{N}} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)) = \bigcup_{i\in\mathbb{N}}\bigcap_{j=1}^{n} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)), \forall j\in[n]$. In consequence, (1.10) becomes

$$\begin{aligned}\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)\left(\bigwedge R(\overline{\mathbf{x}},\overline{\mathbf{y}})\right) &= \{\nu \mid \mathcal{I},\nu\models\phi\} \cap \bigcup_{i\in\mathbb{N}}\bigcap_{j=1}^{n} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))\\ &= \bigcup_{i\in\mathbb{N}}\left(\{\nu \mid \mathcal{I},\nu\models\phi\} \cap \bigcap_{j=1}^{n} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))\right) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i\left(\bigwedge R(\overline{\mathbf{x}},\overline{\mathbf{y}})\right)\end{aligned}$$

It easily follows that also $\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)\left(\bigwedge R(\overline{\mathbf{x}})\right) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(\bigwedge R(\overline{\mathbf{x}}))$ and, as R was chosen arbitrarily, we can conclude that (1.2) was successfully proven. □

Theorem 1.3.4 (Continuity of $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$)**.** The function $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}} : \mathsf{Assign}(\mathcal{S}^{\mathsf{p}}) \to \mathsf{Assign}(\mathcal{S}^{\mathsf{p}})$, induced by the FOL inductive system $\mathcal{S}$ and the interpretation $\mathcal{I}$, is continuous, i.e $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}\big(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\big) = \bigvee_{i\in\mathbb{N}} \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_i)$ if $(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain in $\mathsf{Assign}(\mathcal{S}^{\mathsf{p}})$.

Proof. Let $p \in \mathcal{S}^{\mathsf{p}}$ be any predicate symbol defined by the inductive system $\mathcal{S}$ such that $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_m(\overline{\mathbf{x}})$. Then, as per the definition of $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$,

$$\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(p) = \bigcup_{j=1}^{m} \{\nu(\overline{\mathbf{x}}) \mid \nu \in \left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)\Big(\bigwedge R_j(\overline{\mathbf{x}})\Big)\} \tag{1.11}$$

Since $(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain, it follows by Lemma 1.3.3 that $\big(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\big)(\bigwedge R_j(\overline{\mathbf{x}})) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(\bigwedge R_j(\overline{\mathbf{x}})), \forall j \in [m]$, so we can rewrite (1.11) as

$$\begin{aligned}
\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(p) &= \bigcup_{j=1}^{m} \{\nu(\overline{\mathbf{x}}) \mid \nu \in \bigcup_{i\in\mathbb{N}} \mathcal{X}_i\Big(\bigwedge R_j(\overline{\mathbf{x}})\Big)\} \\
&= \bigcup_{j=1}^{m} \bigcup_{i\in\mathbb{N}} \{\nu(\overline{\mathbf{x}}) \mid \nu \in \mathcal{X}_i\Big(\bigwedge R_j(\overline{\mathbf{x}})\Big)\} \\
&= \bigcup_{i\in\mathbb{N}} \bigcup_{j=1}^{m} \{\nu(\overline{\mathbf{x}}) \mid \nu \in \mathcal{X}_i\Big(\bigwedge R_j(\overline{\mathbf{x}})\Big)\} \\
&= \bigcup_{i\in\mathbb{N}} \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_i)(p) = \left(\bigvee_{i\in\mathbb{N}} \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_i)\right)(p)
\end{aligned}$$

As p was chosen arbitrarily, we can conclude that $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}\big(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\big) = \bigvee_{i\in\mathbb{N}} \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_i)$. □

Theorem 1.3.5 (FOL least solution)**.** An FOL inductive system $\mathcal{S}$ has a unique least solution equal to the least fixed point of $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$, i.e. $\mu\mathcal{S}^{\mathcal{I}} = \mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})$. Moreover, $\mu\mathcal{S}^{\mathcal{I}} = {\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^{n}(\mathcal{X}_\emptyset)$, where $n \in \mathbb{N}$ is the smallest value for which ${\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^{n+1}(\mathcal{X}_\emptyset) = {\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^{n}(\mathcal{X}_\emptyset)$.

Proof. Let $\mathsf{Fp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}) = \big\{\mathcal{X} \mid \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}) = \mathcal{X}\big\}$ be the set of all fixed points of $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$. Since $(\mathsf{Assign}(\mathcal{S}^{\mathsf{p}}), \preccurlyeq)$ is a complete lattice and $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$ is monotonically increasing, as shown by Theorem 1.3.2, it follows from Tarski's fixed point theorem [47] that $(\mathsf{Fp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}), \preccurlyeq)$ is also a complete lattice.

This ensures the existence of a unique least fixed point for $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$ and, furthermore, [47] gives us a way to compute this least fixed point as

$$\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}) = \bigwedge\{\mathcal{X} \mid \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}) \preccurlyeq \mathcal{X}\} = \bigwedge \mathsf{Sol}^{\mathcal{I}}_{\mathcal{S}}$$

Thus, $\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})$ is the greatest lower bound of $\mathsf{Sol}^{\mathcal{I}}_{\mathcal{S}} \subseteq \mathsf{Assign}(\mathcal{S}^{\mathsf{p}})$ and it is unique because any subset of a complete lattice has a unique greatest lower bound. Also, due to the reflexiveness of $\preccurlyeq$,

$$\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})) = \mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}) \preccurlyeq \mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}) \;\Rightarrow\; \mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}) \in \mathsf{Sol}^{\mathcal{I}}_{\mathcal{S}}$$

So $\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})$ is also a solution of $\mathcal{S}$. Suppose $\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})$ is not a least solution of $\mathcal{S}$. Then, by definition 1.3.4, there must exist some $\mathcal{X} \prec \mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})$ such that $\mathcal{X} \in \mathsf{Sol}^{\mathcal{I}}_{\mathcal{S}}$. But, since $\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})$ is the greatest lower bound of $\mathsf{Sol}^{\mathcal{I}}_{\mathcal{S}}$, it must be the case that $\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}) \preccurlyeq \mathcal{X}$, which leads to a contradiction. Consequently, $\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})$ is the unique least solution of $\mathcal{S}$.

Additionally, because $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$ is also continuous, it follows from Kleene's fixed point theorem that we can compute $\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}})$ as the least upper bound of the chain

$$\mathcal{X}_\emptyset \preccurlyeq \mathbb{F}^{\mathcal{I}}_{\mathcal{S}}(\mathcal{X}_\emptyset) \preccurlyeq {\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^2(\mathcal{X}_\emptyset) \preccurlyeq \ldots \preccurlyeq {\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^i(\mathcal{X}_\emptyset) \preccurlyeq \ldots, i \in \mathbb{N}$$

obtained by iterating $\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}$ on the least element $\mathcal{X}_\emptyset$ of $\mathsf{Assign}(\mathcal{S}^{\mathsf{P}})$. Thus,

$$\mathsf{lfp}(\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}) = \mu\mathcal{S}^{\mathcal{I}} = \bigvee_{i \in \mathbb{N}} {\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^i(\mathcal{X}_\emptyset) = {\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^n(\mathcal{X}_\emptyset)$$

where $n \in \mathbb{N}$ is the smallest value for which ${\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^{n+1}(\mathcal{X}_\emptyset) = {\mathbb{F}^{\mathcal{I}}_{\mathcal{S}}}^n(\mathcal{X}_\emptyset)$. □

1.3.3 Solution of an Inductive System in SL

Let $\mathcal{S}$ be an inductive system in SL. An *assignment* $\mathcal{X}$ maps each predicate $p \in \mathcal{S}^{\mathsf{P}}$ of arity n to a set $\mathcal{X}(p) \subseteq \mathsf{L}^n \times \mathsf{Heaps}$. We extend $\mathcal{X}$ to a predicate atom $p(\overline{\mathbf{x}})$ such that

$$\mathcal{X}(p(\overline{\mathbf{x}})) = \{(\nu, h) \mid (\nu(\overline{\mathbf{x}}), h) \in \mathcal{X}(p)\}$$

to a set $F = \{\phi_1, \ldots \phi_k, q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}$ of formulae and predicate atoms such that

$$\mathcal{X}(\ast F) = \{(\nu, h_0 \uplus \biguplus_{j=1}^{n} h_j) \mid \nu, h_0 \models^{\mathsf{SL}} \phi_1 \ast \ldots \ast \phi_k \text{ and } (\nu, h_i) \in \mathcal{X}(q_i(\overline{\mathbf{x}}_i)), \forall i \in [n]\}$$

and to a predicate rule $\langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle$ such that

$$\mathcal{X}(\ast R(\overline{\mathbf{x}})) = \{(\nu, h) \mid \nu(\overline{\mathbf{x}}) = \gamma(\overline{\mathbf{x}}) \text{ and } (\gamma, h) \in \mathcal{X}(\ast R(\overline{\mathbf{x}}, \overline{\mathbf{y}}))\}$$

The set of all assignments on the predicates in $\mathcal{S}$ is $\mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}})$. We write $\mathcal{X}_\emptyset$ for the assignment that maps all predicates in $\mathcal{S}^{\mathsf{P}}$ to the empty set, and $\mathcal{X}_\sigma$ for the assignment that maps each predicate $p \in \mathcal{S}^{\mathsf{P}}$ of arity n to the set $\mathsf{L}^n \times \mathsf{Heaps}$.

We use the relation $\preccurlyeq$, its strict variant $\prec$, and the operators $\curlyvee$ and $\curlywedge$ with a similar meaning as in Section 1.3.2, except they are now applied to assignments in $\mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}})$. Furthermore, by a similar argument as in Section 1.3.2, $\mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}}) = (\mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}}), \preccurlyeq) = (\mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}}), \preccurlyeq, \curlyvee, \curlywedge, \mathcal{X}_\emptyset, \mathcal{X}_\sigma)$ is a complete lattice

The inductive system $\mathcal{S}$ induces a function $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}} : \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}}) \to \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}})$ such that

$$\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}(\mathcal{X})(p) = \bigcup_{i=1}^{m} \{(\nu(\overline{\mathbf{x}}), h) \mid (\nu, h) \in \mathcal{X}(\ast R_i(\overline{\mathbf{x}}))\}$$

where $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_m(\overline{\mathbf{x}})$.

Due to the semantics introduced above, the proofs for the following lemmas and theorems will handle subsets of $\mathcal{V} \times \mathsf{Heaps}$ and $\mathsf{L}^n \times \mathsf{Heaps}$. We introduce the following notation such that, when $A, B \subseteq \mathcal{V} \times \mathsf{Heaps}$,

$$A \uplus B = \{(\gamma, h \uplus h') \mid (\gamma, h) \in A, (\gamma, h') \in B \text{ and } h, h' \text{ disjoint}\}$$

and, when $A, B \subseteq \mathsf{L}^n \times \mathsf{Heaps}$,

$$A \uplus B = \{(\overline{\mathbf{v}} \cdot \overline{\mathbf{v}}', h \uplus h') \mid (\overline{\mathbf{v}}, h) \in A, (\overline{\mathbf{v}}', h') \in B \text{ and } h, h' \text{ disjoint}\}$$

Consider two other sets A', B' such that A, A', B, B' are all subsets of either $\mathcal{V} \times \mathsf{Heaps}$ or $\mathsf{L}^n \times \mathsf{Heaps}$. Then it easily follows that

$$(A \cup A') \uplus B = (A \uplus B) \cup (A' \uplus B)$$

because, when $A, B \subseteq \mathcal{V} \times \mathsf{Heaps}$,

$$\begin{aligned}(A \cup A') \uplus B =& \{(\gamma, h \uplus h') \mid (\gamma, h) \in A \cup A', (\gamma, h') \in B, \text{ and } h, h' \text{ disjoint}\}\\ =& \{(\gamma, h \uplus h') \mid (\gamma, h) \in A, (\gamma, h') \in B, \text{ and } h, h' \text{ disjoint}\}\\ & \cup \{(\gamma, h \uplus h') \mid (\gamma, h) \in A', (\gamma, h') \in B, \text{ and } h, h' \text{ disjoint}\}\\ =& (A \uplus B) \cup (A' \uplus B)\end{aligned}$$

and, when $A, B \subseteq \mathsf{L}^n \times \mathsf{Heaps}$,

$$\begin{aligned}(A \cup A') \uplus B =& \{(\overline{\mathbf{v}} \cdot \overline{\mathbf{v}}', h \uplus h') \mid (\overline{\mathbf{v}}, h) \in A \cup A', (\overline{\mathbf{v}}', h') \in B, \text{ and } h, h' \text{ disjoint}\}\\ =& \{(\overline{\mathbf{v}} \cdot \overline{\mathbf{v}}', h \uplus h') \mid (\overline{\mathbf{v}}, h) \in A, (\overline{\mathbf{v}}', h') \in B, \text{ and } h, h' \text{ disjoint}\}\\ & \cup \{(\overline{\mathbf{v}} \cdot \overline{\mathbf{v}}', h \uplus h') \mid (\overline{\mathbf{v}}, h) \in A', (\overline{\mathbf{v}}', h') \in B, \text{ and } h, h' \text{ disjoint}\}\\ =& (A \uplus B) \cup (A' \uplus B)\end{aligned}$$

This enables the following property relative to set inclusion:

$$A \uplus B \subseteq A' \uplus B \text{ when } A \subseteq A'$$

because $A' \uplus B = (A \uplus B) \cup ((A' \setminus A) \uplus B)$, which is clearly a superset of $A \uplus B$.

Based on these properties, it is also true that

$$(A \cup A') \uplus (B \cup B') = (A \uplus B) \cup (A' \uplus B') \text{ when } A \subseteq A' \text{ and } B \subseteq B'$$

because $(A \cup A') \uplus (B \cup B') = (A \uplus (B \cup B')) \cup (A' \uplus (B \cup B')) = (A \uplus (B \cup B')) \cup A' \uplus B' = (A \uplus B) \cup (A \uplus B') \cup (A' \uplus B') = (A \uplus B) \cup (A' \uplus B')$

Given now two sets F_1 and F_2 of formulae and predicate atoms, the following holds:

$$\mathcal{X}(* F_1 * * F_2) = \mathcal{X}(* F_1) \uplus \mathcal{X}(* F_2)$$

and, as we did for the FOL assignments, we also define

$$\mathcal{X}(* F_1 \vee * F_2) = \mathcal{X}(* F_1) \cup \mathcal{X}(* F_2)$$

Definition 1.3.5 (Solution of an SL inductive system)**.** A *solution* of $\mathcal{S}$ is an assignment $\mathcal{X} \in \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}})$ such that $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}(\mathcal{X}) \preccurlyeq \mathcal{X}$. The set of all solutions of $\mathcal{S}$ is $\mathsf{Sol}^{\mathsf{SL}}_{\mathcal{S}} = \{\mathcal{X} \mid \mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}(\mathcal{X}) \preccurlyeq \mathcal{X}\}$. A *least solution* of $\mathcal{S}$ is $\mu\mathcal{S}^{\mathsf{SL}} \in \mathsf{Sol}^{\mathsf{SL}}_{\mathcal{S}}$ such that, for any assignment $\mathcal{X} \prec \mu\mathcal{S}^{\mathsf{SL}}$, $\mathcal{X} \notin \mathsf{Sol}^{\mathsf{SL}}_{\mathcal{S}}$.

Lemma 1.3.6. The extension of an assignment is monotonically increasing, i.e if $\mathcal{X}_1, \mathcal{X}_2 \in \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}})$ such that $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$, then

$$\mathcal{X}_1(p(\overline{\mathbf{x}})) \subseteq \mathcal{X}_2(p(\overline{\mathbf{x}})), \text{ for any } p(\overline{\mathbf{x}}) \in \mathsf{Atom} \text{ and } p \in \mathcal{S}^{\mathsf{P}} \tag{1.12}$$

$$\mathcal{X}_1(* R(\overline{\mathbf{x}})) \subseteq \mathcal{X}_2(* R(\overline{\mathbf{x}})), \text{ for any } \langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S} \tag{1.13}$$

Proof. Since $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$, it follows that $\mathcal{X}_1(p) \subseteq \mathcal{X}_2(p), \forall p \in \mathcal{S}^{\mathsf{P}}$. Thus,

$$\begin{aligned}\mathcal{X}_1(p(\overline{\mathbf{x}})) &= \{(\nu, h) \mid (\nu(\overline{\mathbf{x}}), h) \in \mathcal{X}_1(p)\}\\ &\subseteq \{(\nu, h) \mid (\nu(\overline{\mathbf{x}}), h) \in \mathcal{X}_2(p)\} = \mathcal{X}_2(p(\overline{\mathbf{x}}))\end{aligned}$$

for any $p(\overline{\mathbf{x}}) \in \mathsf{Atom}$ with $p \in \mathcal{S}^{\mathsf{P}}$. We have successfully proven (1.12).

Let $R(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n), q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}$ such that $\langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S}$ and $\overline{\mathbf{y}} = \overline{\mathbf{x}}_1 \cdot \ldots \cdot \overline{\mathbf{x}}_n$. By the extension of assignments, for any $i \in [2]$,

$$\begin{aligned}\mathcal{X}_i(* R(\overline{\mathbf{x}})) =& \{(\nu, h_0 \uplus \biguplus_{j=1}^{n} h_j) \mid \nu, h_0 \models^{\mathsf{SL}} \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}}) \text{ and } (\nu, h_j) \in \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)), \forall j \in [n]\} \\ =& \{(\nu, h_0) \mid \nu, h_0 \models^{\mathsf{SL}} \phi\} \uplus \biguplus_{j=1}^{n} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)) \end{aligned} \tag{1.14}$$

Since $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$, it follows from (1.12) that $\mathcal{X}_1(q_j(\overline{\mathbf{x}}_j)) \subseteq \mathcal{X}_2(q_j(\overline{\mathbf{x}}_j)), \forall j \in [n]$. Thus, by chaining the property of $\uplus$ relative to subsets,

$$\biguplus_{j=1}^{n} \mathcal{X}_1(q_j(\overline{\mathbf{x}}_j)) \subseteq \biguplus_{j=1}^{n} \mathcal{X}_1(q_j(\overline{\mathbf{x}}_j)) \tag{1.15}$$

From (1.14) and (1.15) we obtain that

$$\{(\nu, h_0) \mid \nu, h_0 \models^{\mathsf{SL}} \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})\} \uplus \biguplus_{j=1}^{n} \mathcal{X}_1(q_j(\overline{\mathbf{x}}_j)) \subseteq \{(\nu, h_0) \mid \nu, h_0 \models^{\mathsf{SL}} \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})\} \uplus \biguplus_{j=1}^{n} \mathcal{X}_2(q_j(\overline{\mathbf{x}}_j))$$

and, thus, that $\mathcal{X}_1(* R(\overline{\mathbf{x}}, \overline{\mathbf{y}})) \subseteq \mathcal{X}_2(* R(\overline{\mathbf{x}}, \overline{\mathbf{y}}))$. It easily follows that also $\mathcal{X}_1(* R(\overline{\mathbf{x}})) \subseteq \mathcal{X}_2(* R(\overline{\mathbf{x}}))$. As R was chosen arbitrarily, we conclude that (1.13) was successfully proven. □

Theorem 1.3.7 (Monotonicity of $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}$)**.** The function $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}} : \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}}) \to \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}})$, induced by the SL inductive system $\mathcal{S}$, is monotonically increasing, i.e $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}(\mathcal{X}_1) \preccurlyeq \mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}(\mathcal{X}_2)$ if $\mathcal{X}_1, \mathcal{X}_2 \in \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}})$ such that $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$.

Proof. The proof is similar to the one of Theorem 1.3.2, by using Lemma 1.3.6 to justify the monotonicity of assignment extension. □

Lemma 1.3.8. The extension of an assignment is continuous, i.e if $(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain in $\mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{P}})$, then

$$\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(p(\overline{\mathbf{x}})) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(p(\overline{\mathbf{x}})), \text{ for any } p(\overline{\mathbf{x}}) \in \mathsf{Atom} \text{ and } p \in \mathcal{S}^{\mathsf{P}} \tag{1.16}$$

$$\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(* R(\overline{\mathbf{x}})) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(* R(\overline{\mathbf{x}})), \text{ for any } \langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S} \tag{1.17}$$

Proof. By the extension of assignments and the definition of assignment union,

$$\begin{aligned}\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(p(\overline{\mathbf{x}})) &= \{(\nu, h) \mid (\nu(\overline{\mathbf{x}}), h) \in \left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(p)\} \\ &= \{(\nu, h) \mid (\nu(\overline{\mathbf{x}}), h) \in \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(p)\} \\ &= \bigcup_{i\in\mathbb{N}} \{(\nu, h) \mid (\nu(\overline{\mathbf{x}}), h) \in \mathcal{X}_i(p)\} = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(p(\overline{\mathbf{x}}))\end{aligned}$$

for any $p(\overline{\mathbf{x}}) \in \mathsf{Atom}$ with $p \in \mathcal{S}^{\mathsf{p}}$. We have successfully proven (1.12).

Let $R(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = \{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n), q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}$ such that $\langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S}$ and $\overline{\mathbf{y}} = \overline{\mathbf{x}}_1 \cdot \ldots \cdot \overline{\mathbf{x}}_n$. By the extension of assignments,

$$\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(* R(\overline{\mathbf{x}}, \overline{\mathbf{y}})) = \{(\nu, h_0 \uplus \biguplus_{j=1}^{n} h_j) \mid \nu, h_0 \models \phi, (\nu, h_j) \in \left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(q_j(\overline{\mathbf{x}}_j)), \forall j \in [n]\}$$

$$= \{(\nu, h_0) \mid \nu, h_0 \models \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})\} \uplus \biguplus_{j=1}^{n} \left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(q_j(\overline{\mathbf{x}}_j)) \tag{1.18}$$

$(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain, so it follows from (1.16) that $\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(q_j(\overline{\mathbf{x}}_j)) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))$ and we can rewrite (1.18) as

$$\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(* R(\overline{\mathbf{x}}, \overline{\mathbf{y}})) = \{(\nu, h_0) \mid \nu, h_0 \models \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})\} \uplus \biguplus_{j=1}^{n} \left(\bigcup_{i\in\mathbb{N}} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))\right) \tag{1.19}$$

Due to the properties of $\uplus$, and because $(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain and the extension of assignments is monotonically increasing, as per Lemma 1.3.6, then $\biguplus_{j=1}^{n} \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j)) = \bigcup_{i\in\mathbb{N}} \biguplus_{j=1}^{n} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))$. In consequence, (1.19) becomes

$$\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(* R(\overline{\mathbf{x}}, \overline{\mathbf{y}})) = \{(\nu, h_0) \mid \nu, h_0 \models \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})\} \uplus \bigcup_{i\in\mathbb{N}} \left(\biguplus_{j=1}^{n} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))\right)$$

$$= \bigcup_{i\in\mathbb{N}} \left(\{(\nu, h_0) \mid \nu, h_0 \models \phi\} \uplus \biguplus_{j=1}^{n} \mathcal{X}_i(q_j(\overline{\mathbf{x}}_j))\right) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(* R(\overline{\mathbf{x}}, \overline{\mathbf{y}}))$$

It easily follows that also $\left(\bigvee_{i\in\mathbb{N}} \mathcal{X}_i\right)(* R(\overline{\mathbf{x}})) = \bigcup_{i\in\mathbb{N}} \mathcal{X}_i(\bigwedge R(\overline{\mathbf{x}}))$ and, as R was chosen arbitrarily, we can conclude that (1.13) was successfully proven. □

Theorem 1.3.9 (Continuity of $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}$)**.** The function $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}} : \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{p}}) \to \mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{p}})$, induced by the SL inductive system $\mathcal{S}$, is continuous, i.e $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}\left(\bigcup_{i\in\mathbb{N}} \mathcal{X}_i\right) = \bigcup_{i\in\mathbb{N}} \mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}(\mathcal{X}_i)$ if $(\mathcal{X}_i)_{i\in\mathbb{N}}$ is an ascending chain in $\mathsf{Assign}^{\mathsf{SL}}(\mathcal{S}^{\mathsf{p}})$.

Proof. The proof is similar to the one of Theorem 1.3.4, by using Lemma 1.3.8 to justify the continuity of assignment extension. □

Theorem 1.3.10 (Least solution of an SL inductive system)**.** An SL inductive system $\mathcal{S}$ has a unique least solution equal to the least fixed point of $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}$, i.e. $\mu\mathcal{S}^{\mathsf{SL}} = \mathsf{lfp}(\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}})$. Moreover, $\mu\mathcal{S}^{\mathsf{SL}} = {\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}}^{n}(\mathcal{X}_\emptyset)$, where $n \in \mathbb{N}$ is the smallest value for which ${\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}}^{n+1}(\mathcal{X}_\emptyset) = {\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}}^{n}(\mathcal{X}_\emptyset)$.

Proof. The proof is similar to the one of Theorem 1.3.5, using Theorem 1.3.7 and Theorem 1.3.9 to justify the monotonicity and continuity of $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}$, respectively. □

1.4 The Entailment Problem

In order to refer to any inductive system, regardless of its underlying logic (FOL or SL), we use $\mu\mathcal{S}$ and $\models_{\mathcal{S}}$ to mean either $\mu\mathcal{S}^{\mathcal{I}}$ and $\models^{\mathcal{I}}_{\mathcal{S}}$ or $\mu\mathcal{S}^{\mathsf{SL}}$ and $\models^{\mathsf{SL}}_{\mathcal{S}}$, depending on the context. The main concern of this thesis is the following entailment problem, which we define in the general context of any inductive system $\mathcal{S}$.

Definition 1.4.1 (Entailment problem)**.** Given an inductive system $\mathcal{S}$ and predicates $p^{\sigma_1 \cdots \sigma_m}$, $q_1^{\sigma_1 \cdots \sigma_m}, \ldots, q_n^{\sigma_1 \cdots \sigma_m} \in \mathcal{S}^{\mathsf{p}}$ with the same tuple of argument sorts, does $\mu\mathcal{S}(p) \subseteq \bigcup_{i=1}^{n} \mu\mathcal{S}(q_i)$ (denoted $p \models_{\mathcal{S}} q_1, \ldots, q_n$) hold?

We consider inductive systems for which the set of constraints $\mathcal{S}^{\mathsf{c}}$ consists of quantifier-free formulae in which no disjunction occurs positively (i.e. under an even number of $\neg$ applications) and no conjunction occurs negatively (i.e. under an odd number of $\neg$ applications). Without loss of generality, disjunctions can be eliminated from quantifier-free constraints by splitting each predicate rule $\langle p(\overline{\mathbf{x}}), \{\phi_1 \vee \ldots \vee \phi_m, q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}\rangle$ into m predicate rules of the form $\langle p(\overline{\mathbf{x}}), \{\phi_i, q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}\rangle$, one for each $i \in [m]$.

Additionally, we assume that every $p \in \mathcal{S}^{\mathsf{p}}$ is the goal of at least one predicate rule of $\mathcal{S}$. Otherwise, the least solution is empty for that predicate, i.e. $\mu\mathcal{S}^{\mathcal{I}}(p) = \emptyset$. Furthermore, from the way assignments (and, consequently, least solutions) are extended to predicate rule bodies, $\mu\mathcal{S}^{\mathcal{I}}$ will be empty for any predicate $q \in \mathcal{S}^{\mathsf{p}}$ for which p is a subgoal. Thus, all such predicates and all the predicate rules in which they occur can be safely eliminated from $\mathcal{S}$.

Example 1.4.1 (Entailment problem in FOL)**.** Consider the following inductive system, with FOL constraints:

$$
\begin{array}{ll}
p(x) \leftarrow_{\mathcal{S}} x \approx f(x_1, x_2), p_1(x_1), p_2(x_2) & q(x) \leftarrow_{\mathcal{S}} x \approx f(x_1, x_2), q_1(x_1), q_2(x_2) \\
p_1(x) \leftarrow_{\mathcal{S}} x \approx g(x_1), p_1(x_1) \mid x \approx a & \quad\quad \mid \; x \approx f(x_1, x_2), q_2(x_1), q_1(x_2) \\
p_2(x) \leftarrow_{\mathcal{S}} x \approx g(x_1), p_2(x_1) \mid x \approx b & q_1(x) \leftarrow_{\mathcal{S}} x \approx g(x_1), q_1(x_1) \mid x \approx a \\
 & q_2(x) \leftarrow_{\mathcal{S}} x \approx g(x_1), q_2(x_1) \mid x \approx b
\end{array}
$$

Intuitively, $\mathcal{S}$ models two tree automata with initial states given by the predicates p and q, where p accepts trees of the form $f(g^n(a), g^m(b))$, $n, m \in \mathbb{N}$, while q accepts both trees of the form $f(g^n(a), g^m(b))$ and $f(g^n(b), g^m(a))$, $n, m \in \mathbb{N}$. The entailment $p \models_{\mathcal{S}}^{\mathcal{I}} q$ expresses the language inclusion between the two states and it holds. On the other hand, $q \models_{\mathcal{S}}^{\mathcal{I}} p$ does not hold. ◀

Example 1.4.2 (Entailment problem in SL)**.** Consider the following inductive system, with symbolic heap constraints:

$$
\begin{array}{l}
ls^{+}(x, y) \leftarrow_{\mathcal{S}} x \mapsto y \mid y \approx y' \wedge x \mapsto x', ls^{+}(x', y') \\
ls^{e}(x, y) \leftarrow_{\mathcal{S}} x \approx y \wedge \mathsf{emp} \mid y \approx y' \wedge x \mapsto x', ls^{o}(x', y') \\
ls^{o}(x, y) \leftarrow_{\mathcal{S}} x \mapsto y \mid y \approx y' \wedge x \mapsto x', ls^{e}(x', y') \\
\widehat{ls}^{+}(x, y) \leftarrow_{\mathcal{S}} y \approx y' \wedge x \mapsto x' * ls^{e}(x', y') \mid y \approx y' \wedge x \mapsto x' * ls^{o}(x', y')
\end{array}
$$

Intuitively, $ls^{+}(x, y)$ defines the set of finite list segments of at least one element between x and y, ls^{e} and ls^{o} are list segments of even and odd length, respectively, and $\widehat{ls}^{+}(x, y)$ is the definition of a list segment consisting of one element followed by an even or an odd list segment. Entailments that hold include: (i) $ls^{+} \models^{\mathsf{SL}} \widehat{ls}^{+}$, (ii) $ls^{o} \models^{\mathsf{SL}} \widehat{ls}^{+}$, (iii) $ls^{+} \models^{\mathsf{SL}} ls^{e}, ls^{o}$ and (iv) $\widehat{ls}^{+} \models^{\mathsf{SL}} ls^{e}, ls^{o}$. Some entailments that do not hold are: (i) $ls^{o} \models^{\mathsf{SL}} ls^{e}$, (ii) $ls^{e} \models^{\mathsf{SL}} ls^{+}$, (iii) $ls^{e} \models^{\mathsf{SL}} \widehat{ls}^{+}$ and (iv) $\widehat{ls}^{+} \models^{\mathsf{SL}} ls^{o}$. ◀

1.4.1 Entailments under the Canonical Interpretation

Let $\mathbb{N}^*$ be the set of sequences of natural numbers, where $\varepsilon \in \mathbb{N}^*$ is the empty sequence and $p \cdot q$ is the concatenation of two sequences $p, q \in \mathbb{N}^*$. We call p a *prefix* of q if and only if

there exists some $r \in \mathbb{N}^*$ such that $p \cdot r = q$. A set $X \subseteq \mathbb{N}^*$ is *prefix-closed* if $p \in X$ implies that $p' \in X$ for every prefix p' of p.

A *tree* over the signature $\Sigma = (\Sigma^{\mathsf{s}}, \Sigma^{\mathsf{f}})$ is a ground term $t \in \mathcal{T}_\Sigma$, viewed as a finite partial function $t : \mathbb{N}^* \rightharpoonup_{\mathsf{fin}} \Sigma^{\mathsf{f}}$, where $\mathrm{dom}(t)$ is prefix-closed and, for all $\alpha \in \mathrm{dom}(t)$ such that $t(\alpha) = f^{\sigma_1 \cdots \sigma_n \sigma}$, we have $[n] = \{i \in \mathbb{N} \mid \alpha \cdot i \in \mathrm{dom}(t)\}$, i.e. t is defined at consecutive positions from $\alpha \cdot 1$ to $\alpha \cdot n$ for every argument of f. The tree t is of sort $\sigma \in \Sigma^{\mathsf{s}}$ (written as t^σ) if $t(\varepsilon) = f^{\sigma_1 \cdots \sigma_n \sigma}$, for some $f^{\sigma_1 \cdots \sigma_n \sigma} \in \Sigma^{\mathsf{f}}$.

The set $\mathrm{fr}(t) = \{\alpha \in \mathrm{dom}(t) \mid \alpha \cdot 1 \notin \mathrm{dom}(t)\}$ is called the *frontier* of t. Given a tree t and a position $\alpha \in \mathrm{dom}(t)$, we denote by $t_{|_\alpha}$ the *subtree* of t rooted at p, where, for each $\beta \in \mathbb{N}^*$, we have $t_{|_\alpha}(\beta) = t(\alpha \cdot \beta)$. The subtree order is defined by $u \sqsubseteq t$ if and only if $u = t_{|_\alpha}$, for some $\alpha \in \mathrm{dom}(t)$. For a function symbol $f^{\sigma_1 \cdots \sigma_n \sigma} \in \Sigma^{\mathsf{f}}$ and trees $t_1^{\sigma_1}, \ldots, t_n^{\sigma_n} \in \mathcal{T}_\Sigma$, we denote by $\tau_n(f, t_1, \ldots, t_n)$ the tree t such that $t(\varepsilon) = f$ and $t_{|_i} = t_i$, for all $i \in [n]$.

The *Herbrand (canonical) interpretation* $\mathcal{H}$ is the one in which each constant symbol is interpreted as itself and each function symbol as the function that applies it, as described, for instance, in [48, Section 3.1]. Thus, terms become both syntactical objects and values. In other words, $\mathcal{H}$ maps (i) each sort $\sigma \in \Sigma^{\mathsf{s}}$ into $\mathcal{T}_\Sigma$, (ii) each constant symbol c into the tree $c^{\mathcal{H}} = \{(\varepsilon, c)\}$ consisting of a leaf which is also the root, and (iii) each function symbol $f^{\sigma_1 \cdots \sigma_n \sigma}$ into the function $f^{\mathcal{H}}$ such that $f^{\mathcal{H}}(t_1, \ldots, t_n) = \tau_n(f, t_1, \ldots, t_n)$, for any $t_1^{\sigma_1}, \ldots, t_n^{\sigma_n} \in \mathcal{T}_\Sigma$. Under this interpretation, $t_1 \approx t_2$ if and only if $t_1(\varepsilon) = t_2(\varepsilon) = f^{\sigma_1 \cdots \sigma_n \sigma}$ and $t_{1|_i} \approx t_{2|_i}, \forall i \in [n]$.

Even in this simple case, where function symbols do not have any equational properties (e.g. commutativity, associativity, etc.) entailment problems are undecidable, as stated by the following theorem.

Theorem 1.4.1. The entailment problem is undecidable for inductive systems under the Herbrand interpretation.

Proof. This undecidability result can be shown by reduction from the inclusion problem for context-free languages, which is a known undecidable problem [25, Theorem 9.22 (e)].

Consider a context-free grammar $G = \langle \Xi, \Sigma, \Delta \rangle$, where Ξ is the set of nonterminals, Σ is the alphabet of terminals, and Δ is a set of productions $(X, w) \in \Xi \times (\Xi \cup \Sigma)^*$. For a nonterminal $X \in \Xi$, $\mathcal{L}(G, X) \subseteq \Sigma^*$ is the language produced by G starting with X as axiom. The problem "Given $X, Y \in \Xi$, does $\mathcal{L}(G, X) \subseteq \mathcal{L}(G, Y)$?" is undecidable.

To reduce this inclusion problem to the entailment problem under the Herbrand interpretation, we define an inductive system $\mathcal{S}_G$ based on a context-free grammar $G = \langle \Xi, \Sigma, \Delta \rangle$, as follows:

(i) Each nonterminal $X \in \Xi$ corresponds to a predicate $X(x^\sigma, y^\sigma)$, where σ is the only sort used in the reduction;

(ii) Each alphabet symbol $a \in \Sigma$ corresponds to a function symbol $\hat{a}^{\sigma\sigma}$, and a word $w = a_1 \ldots a_n \in \Sigma^*$ is encoded by the context (i.e. the term with a hole) $\hat{w} = \hat{a}_1(\ldots \hat{a}_n(.))$;

(iii) Each grammar rule $(X, u_1 X_1 \ldots u_n X_n u_{n+1}) \in \Delta$ corresponds to a predicate rule of the inductive system $\mathcal{S}_G$:

$$\langle X(x, y), \{\phi(x, y, x_1, y_1, \ldots, x_n, y_n), X_1(x_1, y_1), \ldots, X_n(x_n, y_n)\} \rangle$$

where $\phi \equiv x \approx \hat{u}_1(x_1) \wedge \bigwedge_{i=1}^{n-1} y_i \approx \hat{u}_{i+1}(x_{i+1}) \wedge y_n \approx \hat{u}_{n+1}(y)$. In addition, a grammar rule $(X, \epsilon) \in \Delta$ is mapped into a rule $\langle X(x, y), \{x \approx y\} \rangle \in \mathcal{S}_G$.

For any nonterminals $X, Y \in \Xi$, we must show that we have $\mathcal{L}(G, X) \subseteq \mathcal{L}(G, Y)$ if and only if $X \models^{\mathcal{H}}_{\mathcal{S}_G} Y$. This is proved using the invariant

$$\forall w \in \Sigma^* \,.\, (\forall t \in \Sigma^{\mathcal{H}} \,.\, [x \leftarrow \hat{w}(t), y \leftarrow t] \in \mu\mathcal{S}^{\mathcal{H}}_G(X)) \Leftrightarrow w \in \mathcal{L}(G, X)$$

where $[x \leftarrow \hat{w}(t), y \leftarrow t]$ denotes the valuation mapping x to $\hat{w}(t)$ and y to t. □

1.4.2 Entailments with Symbolic Heap Constraints

Considering inductive systems with symbolic heap constraints, as described by Definition 1.2.2, there already exist negative results regarding the decidability of entailment problems, such as [27, Theorem 2] and [1, Theorem 3].

Theorem 1.4.2. The entailment problem is undecidable for inductive systems with symbolic heap constraints.

Chapter 2

Proof Systems for Entailments

In this chapter, we describe cyclic proof systems suited for entailments involving inductively defined predicates. More specifically, we give a generalized one for FOL and adapt it to SL (Section 2.3). Their design was influenced by an antichain-based method for checking language inclusion of top-down nondeterministic finite tree automata (Section 2.1). We give a general proof search semi-algorithm (Section 2.2), which can become a decision procedure when soundness and completeness are assured. In order to achieve this, additional restrictions (Section 2.4) are necessary, and we show how they indeed guarantee the soundness and completeness of our proof systems (Section 2.5).

2.1 Downwards Inclusion Check for Tree Automata

Consider top-down nondeterministic finite tree automata (NFTA) over a ranked alphabet $\mathfrak{F}$ [12], where $\#f$ is the rank of the symbol $f \in \mathfrak{F}$. Then $A = \{Q_A, \mathfrak{F}, I_A, \Delta_A\}$ is an NFTA where Q_A is a set of states, $I_A \subseteq Q_A$ are the initial states of A and Δ_A is a set of transition rules $q \xrightarrow{f} (q_1, \ldots, q_n)$, where $n = \#f$ for some $f \in \mathfrak{F}$. Note that NFTA also have an equivalent bottom-up representation, where the initial states become final states and transitions are written as $(q_1, \ldots, q_n) \xrightarrow{f} q$.

In the top-down sense, an NFTA $A = \{Q_A, \mathfrak{F}, I_A, \Delta_A\}$ labels an input tree with states starting at the root and moving downwards. A transition $q \xrightarrow{f} (q_1, \ldots, q_n) \in \Delta_A$ means that, if the input is a tree with root f labelled by state q, then the automaton can move downwards and label the children of f with the states $q_1, \ldots, q_n$, in this order. This process, called a *run*, can start by using any state from I_A to label the root of the input tree, and ends successfully for each leaf a of the input labelled by state r if there exists a transition rule $r \xrightarrow{a} ()$. An input tree is accepted if there exists a successful run of A on it. Then $\mathcal{L}(A)$ – called the *language of* A – is the set of all inputs accepted by A. Also, $\mathcal{L}(A, q)$ – called the *language of* q *in* A – is the set of all inputs accepted starting from state $q \in Q_A$. Note that $\mathcal{L}(A) = \bigcup_{q \in I_A} \mathcal{L}(A, q)$.

An NFTA can be naturally viewed as an inductive system, where predicates represent states and predicate rules are obtained by translation from transition rules. For instance, $q \xrightarrow{f} (q_1, \ldots, q_n)$ is equivalent to $\langle q(x), \{x \approx f(x_1, \ldots, x_n),\ q_1(x_1), \ldots, q_n(x_n)\}\rangle$. This means that x is a tree whose root is f and, if the automaton has labelled x with state q, then

it can label its subtrees $x_1, \ldots, x_n$ with $q_1, \ldots, q_n$, respectively. The variables range over ground terms and the function symbols are interpreted in the canonical (Herbrand) sense. Note that, when describing transition rules for NFTA, the direction of the arrows denotes either the top-down or the bottom-up representation. While similar, our $q(x) \leftarrow_{\mathcal{S}} R_1, \ldots, R_n$ notation simply means that *the set of models for q is the union of the sets of models given by the predicate rules $R_1, \ldots, R_n$*, and it is not meant to indicate a certain representation. Moreover, $\mathcal{L}(A, q)$ is equivalent to $\mu\mathcal{S}^{\mathcal{H}}(q)$ for any state q and its translation into a predicate.

Given two NFTA $A = \{Q_A, \mathfrak{F}, I_A, \Delta_A\}$ and $B = \{Q_B, \mathfrak{F}, I_B, \Delta_B\}$, a common problem is language inclusion, i.e. whether $\mathcal{L}(A) \subseteq \mathcal{L}(B)$. This can be translated into a language inclusion problem for states, as it is equivalent to asking whether $\mathcal{L}(A, p) \subseteq \bigcup_{q \in I_B} \mathcal{L}(B, q)$ for every $p \in I_A$. Considering two states p and q, and an inductive system $\mathcal{S}$ representing the translation of A and B, the language inclusion problem $\mathcal{L}(A, p) \subseteq \mathcal{L}(B, q)$ is equivalent to the entailment problem $p \models^{\mathcal{H}}_{\mathcal{S}} q$, which asks whether every tree model of p (defined as an inductive predicate) is a model of q (also defined as an inductive predicate).

Example 2.1.1. The inductive system from Example 1.4.1 describes two NFTA $A = \{\{p, p_1, p_2\}, \mathfrak{F}, \{p\}, \Delta_A\}$ and $B = \{\{q, q_1, q_2\}, \mathfrak{F}, \{q\}, \Delta_B\}$, where $\mathfrak{F} = \{f(,), g(), a, b\}$ and

$$\begin{aligned}
\Delta_A = \{p \xrightarrow{f} (p_1, p_2), \quad & p_1 \xrightarrow{g} p_1, \quad p_1 \xrightarrow{a} (), \\
& p_2 \xrightarrow{g} p_2, \quad p_2 \xrightarrow{b} ()\} \\
\Delta_B = \{q \xrightarrow{f} (q_1, q_2), \quad & q_1 \xrightarrow{g} q_1, \quad q_1 \xrightarrow{a} (), \\
q \xrightarrow{f} (q_2, q_1), \quad & q_2 \xrightarrow{g} q_2, \quad q_2 \xrightarrow{b} ()\}
\end{aligned}$$

Then $\mathcal{L}(A) = \mathcal{L}(A, p) = \mu\mathcal{S}^{\mathcal{H}}(p)$ and $\mathcal{L}(B) = \mathcal{L}(B, q) = \mu\mathcal{S}^{\mathcal{H}}(q)$. The language inclusion problem $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ is then equivalent to $\mathcal{L}(A, p) \subseteq \mathcal{L}(B, q)$, which is furthermore equivalent to the entailment problem $p \models^{\mathcal{H}}_{\mathcal{S}} q$. ◀

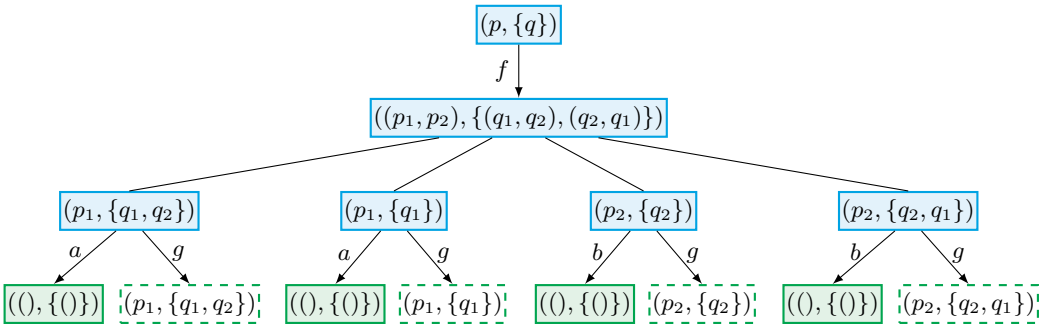
Since language inclusion is decidable for NFTA [12, Corollary 1.7.9], we leverage an existing algorithm for this problem by Holík et al. [24] to build a complete set of inference rules and derive a proof search technique. The downwards inclusion check proposed by Holík et al. searches for counterexamples of $\mathcal{L}(A, p) \subseteq \bigcup_{i=1}^{k} \mathcal{L}(B, q_i)$ by building a tree containing pairs $(r, \{s_1, \ldots, s_n\})$, where r is a state that can be reached from p via a series of transitions in Δ_A, and $\{s_1, \ldots, s_n\}$ are all the states that can be reached from $q_1, \ldots, q_k$ via a series of transitions in Δ_B with the same symbols. A counterexample is discovered when the algorithm encounters a pair $(r, \{s_1, \ldots, s_n\})$ such that there exists a transition $r \xrightarrow{f} (r'_1, \ldots r'_m)$ in Δ_A and there exists no transition $s_i \xrightarrow{f} (s'_1, \ldots s'_m)$ in Δ_B for any $i \in [n]$. In this situation, we reach a leaf of the form $((r'_1, \ldots, r'_m), \emptyset)$, where $m \geq 0$. On a different note, if a leaf $((), S)$ with $() \in S$ is reached, then the check was successful on its corresponding branch. This type of leaves indicate a transition with a symbol of rank 0 that exists for both r and some state s_i, with $i \in [n]$, of the parent pair $(r, \{s_1, \ldots, s_n\})$.

One important aspect is that, by the nature of tree automata, it is possible to have transitions such as $p \xrightarrow{f} (r_1, \ldots, r_n)$ and $q \xrightarrow{f} (s^1_1, \ldots, s^1_n), \ldots, q \xrightarrow{f} (s^k_1, \ldots, s^k_n)$. Thus, the inclusion $\mathcal{L}(A, p) \subseteq \mathcal{L}(B, q)$ holds only if $\mathcal{L}(A, (r_1, \ldots, r_n)) \subseteq \bigcup_{i=1}^{k} \mathcal{L}(B, (s^i_1, \ldots, s^i_n))$ also holds. In this case, the algorithm reaches a pair $((r_1, \ldots, r_n), \{(s^1_1, \ldots, s^1_n), \ldots, (s^i_1, \ldots, s^i_n)\})$. Because the union $\bigcup_{i=1}^{k} \mathcal{L}(B, (s^i_1, \ldots, s^i_n)) = \bigcup_{i=1}^{k} \left(\mathcal{L}(B, s^i_1) \times \ldots \times \mathcal{L}(B, s^i_n)\right)$ cannot be simply computed component-wise, there is no easy check that can be performed on such a pair. Instead, using some properties of the Cartesian product, we can perform a split action and obtain several groups of simpler inclusions $\mathcal{L}(A, r_i) \subseteq \bigcup_{s \in S_i} \mathcal{L}(B, s)$, with $S_i \subseteq$

$\{s_i^1, \ldots, s_i^k\}$ and $i \in [n]$, such that, if all the inclusions hold in at least one group, then $\mathcal{L}(A, (r_1, \ldots, r_n)) \subseteq \bigcup_{i=1}^{k} \mathcal{L}(B, (s_1^i, \ldots, s_n^i))$ also holds (see [24, Theorem 1]). Then the pair $((r_1, \ldots, r_n), \{(s_1^1, \ldots, s_n^1), \ldots, (s_1^k, \ldots, s_n^k)\})$ can also be broken down into several groups of pairs (r_i, S_i) and, if a counterexample is encountered for at least one pair in every group, then there also exists a counterexample for $((r_1, \ldots, r_n), \{(s_1^1, \ldots, s_n^1), \ldots, (s_1^k, \ldots, s_n^k)\})$.

The algorithm also keeps a workset of previously checked pairs and stops successfully whenever it encounters a pair identical to one of its ancestors. This is justified by the fact that, if the inclusion does not hold, it is not because of the sequence of pairs obtained from the ancestor to the current one. Otherwise, by infinite descent, there would exist a counterexample that is a strictly smaller tree than the one obtained from the ancestor and, by repeating the path infinitely often, we would obtain an infinite sequence of strictly smaller counterexamples, which is impossible, as we only consider finite trees.

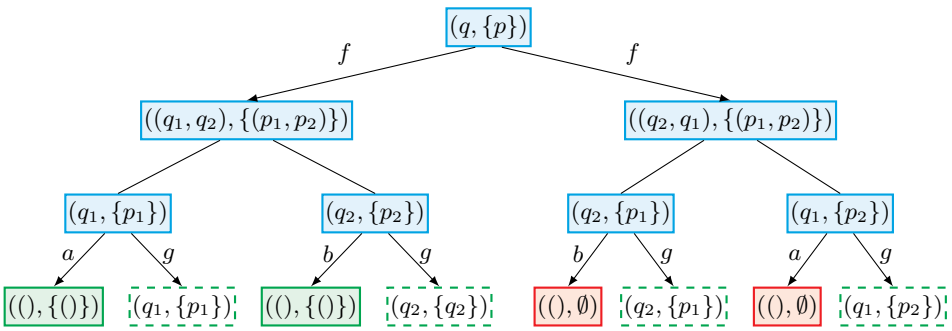
Example 2.1.2. Consider the NFTA A and B from Example 2.1.1. To check $\mathcal{L}(A, p) \subseteq \mathcal{L}(B, q)$, we start with $(p, \{q\})$. A run of the algorithm is depicted below, where the enumerated pairs are organized as a tree. The algorithm perform two types of actions: transitions (indicated by arrows labelled with symbols) and split actions (indicated by unlabelled edges).



Note that, for the pair $((p_1, p_2), \{(q_1, q_2), (q_2, q_1)\})$, this run checks the group $(p_1, \{q_1, q_2\})$, $(p_1, \{q_1\})$, $(p_2, \{q_2\})$ and $(p_2, \{q_2, q_1\})$. By [24, Theorem 1], the other possibilities are:

- $(p_1, \{q_1, q_2\})$, $(p_1, \{q_1\})$, $(p_1, \{q_2\})$ and $(p_2, \{q_2, q_1\})$;
- $(p_1, \{q_1, q_2\})$, $(p_2, \{q_1\})$, $(p_1, \{q_2\})$ and $(p_2, \{q_2, q_1\})$;
- $(p_1, \{q_1, q_2\})$, $(p_2, \{q_1\})$, $(p_2, \{q_2\})$ and $(p_2, \{q_2, q_1\})$.

On the other hand, note how $\mathcal{L}(B, q) \not\subseteq \mathcal{L}(A, p)$. There exists only one run of the algorithm starting with the pair $(q, \{p\})$, as depicted below.



Because both sets in $((q_1, q_2), \{(p_1, p_2)\})$ and $((q_2, q_1), \{(p_1, p_2)\})$ contain only one tuple, the pairs can be split in only one way, component-wise. The counterexamples in this case are $f(b, a)$, $f(b, x)$ with $x \in \mathcal{L}(B, q_1)$, and $f(x, a)$ with $x \in \mathcal{L}(B, q_2)$. ◀

2.2 A Proof Search Semi-algorithm

The main idea behind the proof systems we want to build is to view a complete, counter-example-free search tree for an inclusion problem $\mathcal{L}(A, p) \subseteq \bigcup_{i=1}^{k} \mathcal{L}(B, q_i)$ (such as the one depicted in Example 2.1.2) as a proof for the validity of the entailment $p \models_{\mathcal{S}}^{\mathcal{H}} q_1, \ldots, q_k$, where $\mathcal{S}$ contains the predicate definitions corresponding to the two NFTA in which p and $q_1, \ldots, q_k$ are states. We view the pairs $(r, \{s_1, \ldots, s_n\})$ explored by the downwards inclusion check as sequents $r(x) \vdash s_1(x), \ldots, s_n(x)$ in the proof and apply the principle of Infinite Descent to close those branches leading to an infinite sequence of strictly decreasing counterexamples.

We write $\Gamma \vdash \Delta$ to denote a *sequent*. Typically, the left-hand side Γ is a set of quantifier-free formulae and predicate atoms, while the right-hand side Δ is a set of (possibly existentially quantified) conjunctions over quantifier-free formulae and predicate atoms. In FOL, the left-hand sides are interpreted as classic conjunctions, while in SL they use the separating conjunction. In both cases, the right-hand sides are interpreted as disjunctions. When writing sequents we usually omit the braces in both Γ and Δ. A sequent of the form $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ is called *basic*.

Definition 2.2.1 (Proof system)**.** A *proof system* $\mathcal{R}$ is a set of *inference rule schemata*:

$$IR\ \frac{\Gamma_1 \vdash \Delta_1 \ \ldots\ \Gamma_n \vdash \Delta_n}{\begin{array}{c}\Gamma \vdash \Delta \\ \vdots\ \mathbf{C} \\ \vdots \\ \Gamma_p \vdash \Delta_p\end{array}}\ \text{side conditions}$$

We call $\Gamma_i \vdash \Delta_i$ the *antecedents* and $\Gamma \vdash \Delta$ the *consequent* of the inference rule. When the list of antecedents is empty (i.e. $n = 0$), an inference rule may have a *pivot* $\Gamma_p \vdash \Delta_p$ (which is always a sequent preceding the consequent in the transitive closure of the consequent-antecedent relation or, in other words, an ancestor of the consequent) and $\mathbf{C}$ is a *pivot constraint* on the path between the pivot and the consequent.

An inference rule schema (or simply inference rule, for short) allows for infinite instances that share the same structure. We refer to these as *inference rule instances* or *inference rule applications*. For any inference rule IR we assume that there are finitely many instances of IR with pivot $\Gamma_p \vdash \Delta_p$ and consequent $\Gamma \vdash \Delta$. We denote by $\#(IR)$ the number of its antecedents and write $\top$ for the antecedent list whenever $\#(IR) = 0$. It is possible to have inference rules for which $\#(IR) \geq 0$ and only a particular instance can determine the exact number of antecedents. Thus, we say that $\#(ir) = 0$ if an instance ir of IR generates an empty list of antecedents and also write $\top$ for this particular instance. Naturally, if $\#(IR) = 0$, then also $\#(ir) = 0$ for any instance ir of IR.

Definition 2.2.2 (Derivation)**.** A *derivation* using the proof system $\mathcal{R}$ is a (possibly infinite) tree $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$, where V is a set of vertices, $v_0 \in V$ is the root node, Seq is a total mapping from V to sequents, $\mathsf{Rule} : V \rightharpoonup \mathcal{R}$, $\mathsf{Par} : V \setminus \{v_0\} \to V$ and $\mathsf{Piv} : V \setminus \{v_0\} \rightharpoonup V$ such that:

(i) Each $v \in V$ is labelled by a sequent $\mathsf{Seq}(v)$ and $\mathsf{Rule}(v)$, if defined, indicates the inference rule schema whose instance is applied on $\mathsf{Seq}(v)$;

(ii) For each $v \in V \setminus \{v_0\}$, $\mathsf{Par}(v) \in V$ is the parent node of v and $\mathsf{Seq}(v)$ is an antecedent of the $\mathsf{Rule}(\mathsf{Par}(v))$ instance applied on $\mathsf{Seq}(\mathsf{Par}(v))$. If this inference rule instance generates an empty list of antecedents, then $\mathsf{Seq}(v) = \top$;

(iii) If the instance of $\mathsf{Rule}(v)$ applied on $\mathsf{Seq}(v)$ has a pivot, where $v \in V \setminus \{v_0\}$, then $\mathsf{Piv}(v) \in V$ denotes a node such that $\mathsf{Seq}(\mathsf{Piv}(v))$ is the pivot of this instance.

Definition 2.2.3 (Proof)**.** A *proof* is a finite derivation $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ such that $\mathsf{Seq}(v) = \top$ for all leaves $v \in V$ – i.e. on every branch of the derivation, the last inference rule application has an empty list of antecedents.

Having formalized derivations and proofs, we now define backlinks, traces and paths. These notions are instrumental to formulating properties of derivations and proofs, while also playing an important role in establishing soundness and completeness results later on.

Definition 2.2.4 (Backlink)**.** Given a derivation $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$, a *backlink* is a pair (u, v) with $u, v \in V$ such that $\mathsf{Piv}(u) = v$.

Definition 2.2.5 (Trace)**.** A *trace* in a derivation $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ is a (possibly infinite) sequence of vertices $\tau = v_1, v_2, \ldots$ such that, for all $i > 1$ either $v_{i-1} = \mathsf{Par}(v_i)$ or $\mathsf{Piv}(v_{i-1}) = v_i$. We say that τ *contains a backlink* if $\mathsf{Piv}(v_{i-1}) = v_i$ for some $i > 1$.

Definition 2.2.6 (Path)**.** A *path* in a derivation $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ is a sequence of vertices $\pi = v_1, \ldots, v_n$ such that, for all $i \in [2, n]$, we have $v_{i-1} = \mathsf{Par}(v_i)$. If, furthermore, $\mathsf{Piv}(v_n) = v_1$, then π is a *direct path*.

Note that, since the branching degree of a proof is finite, by König's Lemma every path in a proof must also be finite. Traces, however, can be infinite, in both derivations and proofs. Given a path or a finite trace $\pi = v_1, \ldots, v_k$, we write $\Lambda(\pi)$ for the sequence $\mathsf{Rule}(v_1), \ldots, \mathsf{Rule}(v_k)$ of inference rule schemata applied along π. We can also decompose π into several subsequences $\rho_1, \ldots, \rho_m$ such that their concatenation is equal to π. We denote this by $\pi = \rho_1 \cdot \ldots \cdot \rho_m$ and, in this case, we also write $\Lambda(\pi) = \Lambda(\rho_1) \cdot \ldots \cdot \Lambda(\rho_m)$.

The pivot constraint $\mathbf{C}$ of an inference rule schema *IR* is a set of finite sequences of inference rule schemata, such that, for any instance *ir* of *IR*, if π is the direct path from the pivot of *ir* to its consequent, then $\Lambda(\pi) \in \mathbf{C}$. We usually specify $\mathbf{C}$ using regular expressions.

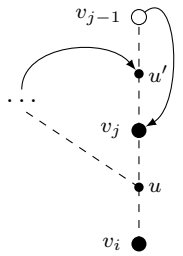
Proposition 2.2.1. Any infinite trace in a proof contains infinitely many direct paths.

Proof. Let τ be an infinite trace in the proof $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$. Since V is finite, τ must contain infinitely many backlinks. Moreover, $V \times V$ is also finite and there can only be a finite number of backlinks, thus there must exist a backlink (v_{i-1}, v_i) that repeats infinitely often in τ.

We now show that, for every finite trace $\rho = v_i, \ldots, v_{i-1}$, where (v_{i-1}, v_i) is a backlink, there exists a direct path in ρ, by induction on the number N of backlinks in ρ. If $N = 0$ the direct path is trivially ρ. For the induction step $N > 0$, we suppose that the property holds for any $N' < N$. Let $\rho = v_i, \ldots, v_{j-1}, v_j, \ldots, v_{i-1}$, where (v_{j-1}, v_j) is the last backlink on ρ. Then the suffix $v_j \ldots v_{i-1}$ of ρ is a path. Since (v_{i-1}, v_i) is a backlink, then v_i is a predecessor of v_{i-1} in D, thus v_i, v_j and v_{i-1} are on the same branch on D. We distinguish two cases:

1. If v_j is v_i or a predecessor of v_i then ρ ends in a direct path from v_i to v_{i-1} in ρ and we are done, because $\rho = v_i, \ldots, v_{j-1}, v_j, \ldots, v_i, \ldots, v_{i-1}$ and there are no more backlinks between v_j and v_{i-1}.

2. Else, if v_i is a strict predecessor of v_j, we show that there must exist another occurrence of v_j in the prefix $v_i, \ldots, v_{j-1}$ of ρ. Note that v_i, v_j and v_{j-1} occur on the same branch in D. Suppose, by contradiction, that $v_k \neq v_j$, for all $k \in [i, j-1]$. To reach v_{j-1} from v_i there must exist a strict predecessor $u \in V$ of v_j and a subsequence of ρ from u to some strict successor $u' \in V$ of v_j that goes through one or more backlinks (see the figure below). However this is not possible, since each backlink leads from a leaf of D to one of its predecessors, thus any trace starting at v_i, passing through u and then following a different branch than the one on which v_i, v_j and v_{j-1} reside, can only return to this branch at u or predecessors of u, whereas u' is a strict successor of u.



We have shown that there exists $k \in [i, j-1]$ such that $v_k = v_j$, thus we have a subsequence $\rho' = v_k, \ldots, v_{j-1}$ of ρ, where (v_{j-1}, v_k) is a backlink, containing $N' < N$ backlinks. By the induction hypothesis, ρ' contains a direct path, which concludes the induction proof.

Therefore, each finite subtrace $v_{i-1}, \ldots, v_i$ contains a direct path. Because the backlink (v_{i-1}, v_i) occurs infinitely often in τ, there are infinitely many such subtraces in τ, we can conclude that τ contains infinitely many direct paths. □

We want our proof systems to be sound and, ideally, complete. Having introduced all the above notions, we can now define soundness and completeness.

Definition 2.2.7 (Soundness and completeness)**.** Given an inductive system $\mathcal{S}$ and an interpretation $\mathcal{I}$, a proof system $\mathcal{R}$ is *sound* if, for every sequent $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ that is the root of a proof constructed with $\mathcal{R}$, the entailment $p \models^{\mathcal{I}}_{\mathcal{S}} q_1, \ldots, q_n$ holds. Moreover, $\mathcal{R}$ is *complete* if for every valid entailment $p \models^{\mathcal{I}}_{\mathcal{S}} q_1, \ldots, q_n$ there exists a proof starting with $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ that can be constructed with $\mathcal{R}$.

Trying to build a proof with no guidance on what inference rule schema to apply at any point can become quite a gruelling task. To this end, we introduce the notion of a strategy, which can be used to guide a proof-search algorithm.

Definition 2.2.8 (Strategy)**.** A *strategy* is a set $\mathbf{S}$ of inference rule schema sequences. A derivation (or proof) D is an $\mathbf{S}$-derivation (or $\mathbf{S}$-proof) if the sequences of inference rule applications along every maximal path in D belong to $\mathbf{S}$.

Definition 2.2.9 (Valid prefix)**.** Given a strategy $\mathbf{S}$, a sequence s of inference rule schemata is a *valid prefix* for $\mathbf{S}$ if there exists another, possibly empty, sequence s' such that their concatenation $s \cdot s'$ belongs to $\mathbf{S}$.

Figure 2.1 describes the data structure `Node`, which represents the nodes in any derivation built using the proof system $\mathcal{R}$. We also consider a constructor `Node(`*sequent*, *rule*, *parent*,

```
Node { sequent  : A sequent Γ ⊢ Δ,
       rule     : An inference rule schema from R
       parent   : Parent of the current node, also of type Node,
       pivot    : Pivot of rule, also of type Node,
       children : A list with elements of type Node          }
```

Figure 2.1: The data structure `Node`, representing a node in a derivation

pivot, *children*) which takes four arguments of the same types as its fields and initializes them. The proof-search semi-algorithm we propose uses this data structure, and also the notion of *applicable inference rule schema*, for which we provide the following definition.

Algorithm 1: Proof search semi-algorithm.

Input : An inductive system $\mathcal{S}$, a basic sequent $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$, a proof system $\mathcal{R}$ and a strategy $\mathbf{S}$

Output: A proof starting with $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$

```
1  Root ← Node(p(x̄) ⊢ q1(x̄), ..., qn(x̄), null, null, null, [])
2  WorkQueue ← {Root}
3  while WorkQueue ≠ [] do
4      Dequeue a node N from WorkQueue
5      Let π be the path between Root and N
6      Let R_N = {IR | IR ∈ R and IR applicable on N.sequent and π}
7      Let R^0_N = {IR | IR ∈ R_N and #(IR) = 0}
8      if Λ(π) · IR valid prefix of S for some IR ∈ R^0_N then
9          N.rule ← IR
10         if IR has a pivot then
11             Let P be the pivot for this instance of IR
12             N.pivot ← P
13         Mark N as closed
14     if N not closed and Λ(π) · IR valid prefix of S for some IR ∈ R_N then
15         N.rule ← IR
16         Let ir be an application of IR on N.sequent
17         foreach antecedent Γ' ⊢ Δ' of ir do
18             N' ← Node(Γ' ⊢ Δ', null, N, null, [])
19             Append N' to N.children
20             Enqueue N' in WorkQueue
21         if N.children is empty then
22             Mark N as closed
```

Definition 2.2.10 (Applicable inference rule schema)**.** Given an inference rule IR, a sequent $\Gamma \vdash \Delta$ and path π starting at the root of a derivation and ending at v, where $\mathsf{Seq}(v) = \Gamma \vdash \Delta$, we say that IR *is applicable on* $\Gamma \vdash \Delta$ *and* π if there exists an instance ir of IR such that:

(i) $\Gamma \vdash \Delta$ matches the consequent of ir such that the side conditions are satisfied;

(ii) There exists $v_p \neq v$ along π, where $\mathsf{Seq}(v_p)$ matches the pivot of ir such that the side

conditions are satisfied, and, if $\pi = \rho \cdot v_p \cdot \eta \cdot v$, then $\Lambda(v_p \cdot \eta) \in \mathbf{C}$, where $\mathbf{C}$ is the pivot constraint of ir.

Given an input sequent $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$, a set $\mathcal{R}$ of inference rule schemata and a strategy $\mathbf{S}$, the proof search semi-algorithm 1 iterates over a work queue of nodes to build a derivation. The root node is initialized with the input sequent, an empty list of children and *nil* for the parent pointer and the inference rule schema. The work queue initially contains only the root. When a node is removed from the work queue, the path between the root sequent and the current sequent is extracted using the parent pointers. The semi-algorithm then computes the sets $\mathcal{R}_{\mathsf{N}}$ (of inference rule schemata applicable on the current sequent and derivation path) and $\mathcal{R}_{\mathsf{N}}^0$ (the subset of $\mathcal{R}_{\mathsf{N}}$ containing only those inference rule schemata with a guaranteed empty list of antecedents). Inference rule schemata in $\mathcal{R}_{\mathsf{N}}^0$ matching the strategy are applied eagerly and the node is marked as closed, indicating a successfully proved path. Otherwise, the semi-algorithm chooses nondeterministically an application of an inference rule schema in $\mathcal{R}_{\mathsf{N}}$ that matches the strategy. Then, for every antecedent, a new child node is created, which is then added to the work queue. If the inference rule application generated an empty list of antecedents, the current node is marked as closed.

Note that, if a proof of $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ exists, then there also exists a finite execution of the semi-algorithm 1 on this sequent. Moreover, if the strategy $\mathbf{S}$ is chosen in a way that forbids infinite derivations, this turns 1 into an algorithm, which can furthermore become a decision procedure for the entailment problem if the input set of inference rules is complete for the input inductive system.

2.3 Cyclic Proof Systems for Inductive Entailments

2.3.1 The Inference Rule Set $\mathcal{R}_{\mathsf{Ind}}$ for FOL Entailments

We introduce the proof system $\mathcal{R}_{\mathsf{Ind}} = \{\mathrm{LU}, \mathrm{RU}, \mathrm{RD}, \wedge\mathrm{R}, \mathrm{SP}, \mathrm{AX}, \mathrm{ID}\}$, suited for FOL entailments. Its inference rule schemata are depicted in Figures 2.2, 2.3, 2.4 and 2.5, where long side conditions are written underneath each respective inference rule schema instead of to the right.

$$\mathrm{LU}\ \frac{\langle R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i), \Gamma \setminus p(\overline{\mathbf{x}}) \vdash \Delta \rangle_{i=1}^{n}}{\Gamma \vdash \Delta} \begin{array}{l} p(\overline{\mathbf{x}}) \in \Gamma,\ \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_n \text{ fresh} \\ p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}}) \end{array}$$

$$\mathrm{RU}\ \frac{\Gamma \vdash \{\exists \overline{\mathbf{y}}_i \,.\, \bigwedge R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i)\}_{i=1}^{n}, \Delta \setminus p(\overline{\mathbf{x}})}{\Gamma \vdash \Delta} \begin{array}{l} p(\overline{\mathbf{x}}) \in \Delta,\ \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_n \text{ fresh} \\ p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}}) \end{array}$$

Figure 2.2: Inference rule schemata for predicate unfolding.

The inference rules of type LU and RU (Figure 2.2) unfold a predicate atom $p(\overline{\mathbf{x}})$ which occurs on either the left- or right-hand side of a sequent $\Gamma \vdash \Delta$, respectively. By unfolding $p(\overline{\mathbf{x}})$, the atom is replaced with the set of predicate rules $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}})$, in which we use fresh subgoal variables. The left unfolding yields a set of sequents that must be all proved, one for each R_i, thus creating several new branches in the derivation that need to be all valid in order to obtain a proof. In contrast, the right unfolding simply replaces $p(\overline{\mathbf{x}})$ on the right-hand side of the sequent with a set of formulae obtained by applying conjunction over each $R_i, i \in [n]$ and existentially quantifying the fresh subgoal variables in them.

Example 2.3.1. Consider the inductive system from Example 1.4.1. In order to prove the entailment $p \models^{\mathcal{H}}_{\mathcal{S}} q$, we start a derivation from the sequent $p(x) \vdash q(x)$. By applying LU on this sequent to unfold $p(x)$, we obtain:

$$\text{LU}\ \frac{x \approx f(x_1, x_2), p_1(x_1), p_2(x_2) \vdash q(x)}{p(x) \vdash q(x)}$$

We can further apply RU on the resulting sequent and unfold $q(x)$:

$$\text{RU}\ \frac{\begin{array}{r} x \approx f(x_1, x_2), p_1(x_1), p_2(x_2) \vdash \exists y_1, y_2 \,.\, x \approx f(y_1, y_2) \wedge q_1(y_1) \wedge q_2(y_2), \\ \exists y_1, y_2 \,.\, x \approx f(y_1, y_2) \wedge q_2(y_1) \wedge q_1(y_2) \end{array}}{x \approx f(x_1, x_2), p_1(x_1), p_2(x_2) \vdash q(x)}$$

Note that, by unfolding, we have introduced the new subgoal variables x_1, x_2 on the left-hand side and the new, existentially quantified subgoal variables y_1, y_2 for each conjunction on the right-hand side. ◀

$$\text{RD}\ \frac{p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\mathcal{Q}_j(\overline{\mathbf{y}}_j\theta) \mid \theta \in S_j\}_{j=1}^{i}}{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n), p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\exists \overline{\mathbf{y}}_j \,.\, \psi_j(\overline{\mathbf{x}}, \overline{\mathbf{y}}_j) \wedge \mathcal{Q}_j(\overline{\mathbf{y}}_j)\}_{j=1}^{k}}$$
$$\text{where } \phi \models^{\mathcal{I}} \bigwedge_{j=1}^{i} \exists \mathbf{y}_j \,.\, \psi_j,\ \ \phi \not\models^{\mathcal{I}} \bigvee_{j=i+1}^{k} \exists \overline{\mathbf{y}}_j \,.\, \psi_j,\ \ S_j \subseteq \text{VIS}(\phi, \psi_j), j \in [i]$$

$$\wedge\text{R}\ \frac{\Gamma \vdash p(\overline{\mathbf{x}}) \wedge \mathcal{Q}, \Delta \quad \Gamma \vdash q(\overline{\mathbf{x}}) \wedge \mathcal{Q}, \Delta}{\Gamma \vdash p(\overline{\mathbf{x}}) \wedge q(\overline{\mathbf{x}}) \wedge \mathcal{Q}, \Delta}$$

Figure 2.3: Inference rule schemata for reducing constraints and for removal of predicate conjunctions over the same arguments.

The inference rules of type RD (Figure 2.3) simplify sequents by eliminating the constraints from both the left- and right-hand sides. This is done by checking the entailments $\phi \models^{\mathcal{I}} \exists \overline{\mathbf{y}}_j \,.\, \psi_j, j \in [k]$ between the left constraint and each of the constraints on the right. If the entailment does not hold, the corresponding conjunction $\exists \overline{\mathbf{y}}_j \,.\, \psi_j(\overline{\mathbf{x}}, \overline{\mathbf{y}}_j) \wedge \mathcal{Q}_j(\overline{\mathbf{y}}_j)$ from the right-hand side is ignored. If the entailment holds, however, it is witnessed by a finite set of substitutions $S_j \subseteq \text{VIS}(\phi, \psi)$ for the existentially quantified subgoal variables $\overline{\mathbf{y}}_j$, where $\text{VIS}(\phi, \psi) = \{\theta : \bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j) \to \mathcal{T}_\Sigma(\mathsf{set}(\overline{\mathbf{x}}) \cup \bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i)) \mid \phi \models^{\mathcal{I}} \psi\theta\}$. For each valid entailment, we eliminate the constraint $\psi_j(\overline{\mathbf{x}}, \overline{\mathbf{y}}_j)$ together with the existential quantification, and add every conjunction of predicates $\mathcal{Q}_j(\overline{\mathbf{y}}_j\theta)$ with $\theta \in S_j$ on the right-hand side of the antecedent. Finally, the left-hand side of the antecedent is obtained from the left-hand side of the consequent by simply eliminating the constraint.

Note that the application of RD, which substitutes the subgoal variables on the right-hand side, may generate conjunctions of predicates sharing the same tuple of arguments (if there originally were more subgoals on the right-hand side before the application of RD). These cases are eagerly eliminated using an inference rule of type $\wedge$R (Figure 2.3). We assume that every application of RD is followed by a cleanup of the right-hand side of its antecedent using sufficiently many applications of $\wedge$R to eliminate all conjunctions of predicates with the same arguments from the right-hand side.

Example 2.3.2. Continuing the derivation from Example 2.3.1, we apply RD on the antecedent of RU:

$$\text{RD}\ \frac{p_1(x_1), p_2(x_2) \vdash q_1(x_1) \wedge q_2(x_2), q_2(x_1) \wedge q_1(x_2)}{\begin{array}{r} x \approx f(x_1, x_2), p_1(x_1), p_2(x_2) \vdash \exists y_1, y_2 \,.\, x \approx f(y_1, y_2) \wedge q_1(y_1) \wedge q_2(y_2), \\ \exists y_1, y_2 \,.\, x \approx f(y_1, y_2) \wedge q_2(y_1) \wedge q_1(y_2) \end{array}}$$

This application of RD checks the entailment $x \approx f(x_1, x_2) \models^{\mathcal{I}} \exists y_1, y_2 \,.\, x \approx f(y_1, y_2)$ for both parts of the right-hand side (since they happen to have the same constraint). The entailment holds and is witnessed by the substitution $\theta = \{(y_1, x_1), (y_2, x_2)\}$. The constraints $x \approx f(x_1, x_2)$ and $x \approx f(y_1, y_2)$ together with the quantification $\exists y_1, y_2$ are removed and θ is applied on both conjunctions from the right-hand side. ◀

$$\text{SP}\ \frac{\langle p_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \vdash \{q^{\ell}_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}\rangle_{j=1}^{n^k}}{p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\mathcal{Q}_j(\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)\}_{j=1}^{k}}$$
$$\text{where } \mathsf{set}(\overline{\mathbf{x}}_i) \cap \mathsf{set}(\overline{\mathbf{x}}_j) = \emptyset, \forall i, j \in [n],$$
$$\mathcal{Q}_j = \textstyle\bigwedge_{i=1}^{n} q_i^j(\overline{\mathbf{x}}_i),\ \overline{\mathcal{Q}}_j = \langle q_1^j, \ldots, q_n^j\rangle,\ \forall j \in [k]$$
$$\mathcal{F}(\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_k) = \{f_1, \ldots, f_{n^k}\},\ \ \bar{\imath} \in [n]^{n^k}$$

Figure 2.4: Inference rule schema for splitting sequents without constraints.

The transition actions performed by the algorithm for tree automata inclusion checking detailed in Section 2.1 are equivalent to applying the LU, RU and RD inference rules all at once. This is a natural consequence of how the function symbol labelling the root of the current input controls the transition rules for tree automata. Function symbols can only be compared via equality, thus the constraints of the predicate rules corresponding to a tree automata match unambiguously. For instance, $x \approx f(x_1, x_2) \models^{\mathcal{H}} \exists y_1 \exists y_2 \,.\, x \approx g(y_1, y_2)$ if and only if f and g are the same function symbol, in which case the only substitution witnessing the validity of the entailment is $\theta(x_i) = y_i$, for $i \in [2]$.

However, when considering general constraints, matching requires the discovery of non-trivial substitutions that prove an entailment between existentially quantified formulae. Moreover, the matching step implemented by the RD rule is crucial for the completeness of the proof system. We generalize from the simple case of tree automata constraints and identify general properties for the set of constraints that allow matching to be complete. Such properties are detailed in Section 2.4.

The inference rules of type SP (Figure 2.4) split a sequent without constraints, of the from $p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\mathcal{Q}_j(\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)\}_{j=1}^{k}$, where $\mathcal{Q}_j(\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) = \bigwedge_{i=1}^{n} q_i^j(\overline{\mathbf{x}}_i)$, into basic sequents, with left-hand sides $p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n)$. This inference rule corresponds to the split action performed by the inclusion check from Section 2.1. Given a set of predicate tuples $\{\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_k\} \subseteq \mathsf{Pred}^n$, for some $n \geq 1$, a *choice function* f maps each tuple $\overline{\mathcal{Q}}_j, j \in [n]$ into an index $f(\overline{\mathcal{Q}}_j) \in [n]$ corresponding to a given position in the tuple. Let $\mathcal{F}(\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_k) = \{f_1, \ldots, f_{n^k}\}$ be the set of all such choice functions. This set has cardinality n^k, for any set of n-tuples of predicates. Consider now a tuple of length n^k containing index choices from 1 to n, each corresponding to a choice function $f_1, \ldots, f_{n^k}$. The set of all such choices is $[n]^{n^k}$. Given a choice $\bar{\imath} \in [n]^{n^k}$, let $\bar{\imath}_j = \mathsf{pos}_j(\bar{\imath})$. Then, for each $\bar{\imath}$ there exists an application of SP, generating n^k antecedents with left hand-side $p_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j}), j \in [n^k]$ and right hand-side

consisting of all predicate atoms $q^{\ell}_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j}), \ell \in [k]$ obtained from predicates at position $\bar{\imath}_j$ in the tuples $\overline{\mathcal{Q}}_\ell$ which are mapped to $\bar{\imath}_j$ by the choice function f_j. In order to obtain a proof, there must exist some application of SP – and, therefore, some $\bar{\imath} \in [n]^{n^k}$ – for which all the generated antecedents can be proven. It is important to note, however, that these tuples are meant to encode the transformation of a formula from CNF to DNF and, as such, not all possibilities are relevant. More precisely, as shown in [24, Section 3], any $\bar{\imath}$ for which there exists $j \in [n^k]$ such that $\bar{\imath}_j \notin \mathrm{img}(f_j)$ can be discarded.

To apply SP, it is necessary that all the conjunctions on the right-hand side of the consequent contain the same number of predicate atoms as the left-hand side. This might not always be the case after applying RD and as many instances of ∧R as possible. Note that the substitutions from RD which map the subgoal variables $\overline{\mathbf{y}}_j$ to the subgoal variables $\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n$ ensure that there cannot be fewer predicate atoms on the left-hand side than on the right-hand side. Otherwise at least two predicate atoms on the right-hand side would share the same tuple of subgoals $\overline{\mathbf{x}}_i$ for some $i \in [n]$ and ∧R would be enabled, which contradicts the assumption that we applied ∧R as many times as possible.

In order to enable the application of SP on sequents that have, on the right-hand side, conjunctions of fewer predicate atoms than the left-hand side, we introduce a set of universal predicate rules $\mathcal{S}_{\mathsf{u}} = \{\langle p^k_{\mathsf{u}}(x_1, \ldots, x_k), \{\top\}\rangle \mid k \geq 0\}$ and assume that any system $\mathcal{S}$ contains it by default. If there are conjunctions of predicates $\mathcal{Q}$ on the right-hand side such that there exist tuples of subgoal variables $\overline{\mathbf{x}}_i$ for which $\exists p \in \mathcal{S}^{\mathsf{p}} \,.\, p(\overline{\mathbf{x}}_i) \in \Gamma$, but $\forall q \in \mathcal{S}^{\mathsf{p}} \,.\, q(\overline{\mathbf{x}}_i)$ does not occur in $\mathcal{Q}$, then we add $p^{\|\overline{\mathbf{x}}_i\|}_{\mathsf{u}}(\overline{\mathbf{x}}_i)$ to $\mathcal{Q}$. This change does not alter the semantics of the entailment and is done implicitly when applying SP.

Example 2.3.3. Continuing the derivation from Example 2.3.1, we can immediately apply SP on the antecedent of RD. As shown in [24, Section 3], using properties of the Cartesian product, an inclusion of the form $R_1 \times R_2 \subseteq S_1 \times S_2 \cup T_1 \times T_2$ corresponding to this sequent can be rewritten as

$$(R_1 \subseteq S_1 \cup T_1) \wedge (R_1 \subseteq S_1 \vee R_2 \subseteq T_2) \wedge (R_1 \subseteq T_1 \vee R_2 \subseteq S_2) \wedge (R_2 \subseteq S_2 \cup T_2)$$

and, by translating this formula into DNF, we would obtain

$$\begin{aligned}
&(R_1 \subseteq S_1 \cup T_1) \wedge (R_1 \subseteq S_1) \wedge (R_1 \subseteq T_1) \wedge (R_2 \subseteq S_2 \cup T_2) \;\vee\\
&(R_1 \subseteq S_1 \cup T_1) \wedge (R_1 \subseteq S_1) \wedge (R_2 \subseteq S_2) \wedge (R_2 \subseteq S_2 \cup T_2) \;\vee\\
&(R_1 \subseteq S_1 \cup T_1) \wedge (R_2 \subseteq T_2) \wedge (R_1 \subseteq T_1) \wedge (R_2 \subseteq S_2 \cup T_2) \;\vee\\
&(R_1 \subseteq S_1 \cup T_1) \wedge (R_2 \subseteq T_2) \wedge (R_2 \subseteq S_2) \wedge (R_2 \subseteq S_2 \cup T_2)
\end{aligned}$$

SP encodes this transformation for $\mu\mathcal{S}^{\mathcal{H}}(p_1) \times \mu\mathcal{S}^{\mathcal{H}}(p_2) \subseteq \mu\mathcal{S}^{\mathcal{H}}(q_1) \times \mu\mathcal{S}^{\mathcal{H}}(q_2) \cup \mu\mathcal{S}^{\mathcal{H}}(q_2) \times \mu\mathcal{S}^{\mathcal{H}}(q_1)$ as follows. Let $\overline{\mathcal{Q}}_1 = (q_1, q_2)$ and $\overline{\mathcal{Q}}_2 = (q_2, q_1)$ be the tuples of predicates corresponding to the right-hand side. The set of choice functions is $\mathcal{F}(\overline{\mathcal{Q}}_1, \overline{\mathcal{Q}}_2) = \{f_1, f_2, f_3, f_4\}$:

	f_1	f_2	f_3	f_4
$\overline{\mathcal{Q}}_1$	1	1	2	2
$\overline{\mathcal{Q}}_2$	1	2	1	2

The set $\{c_1, \ldots, c_{16}\}$ of all possible index choice tuples for $\mathcal{F}(\overline{\mathcal{Q}}_1, \overline{\mathcal{Q}}_2)$ is:

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}
f_1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
f_2	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
f_3	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
f_4	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2

The antecedent of RD corresponds to a valid entailment if there exists some $i \in [16]$ such that, if we choose $\bar{\imath} = c_i$, the application of SP with $\bar{\imath}$ leads to a proof. For this example, the relevant tuples (the ones in which $\bar{\imath}_j \in \mathrm{img}(f_j)$ for every $j \in [4]$) are c_2, c_4, c_6 and c_8. We choose $\bar{\imath} = c_4 = (1, 1, 2, 2)$, generating the following application of SP:

$$\mathrm{SP}\ \frac{p_1(x_1) \vdash q_1(x_1), q_2(x_1) \qquad p_1(x_1) \vdash q_1(x_1) \qquad p_2(x_2) \vdash q_2(x_2) \qquad p_2(x_2) \vdash q_2(x_2), q_1(x_2)}{p_1(x_1), p_2(x_2) \vdash q_1(x_1) \wedge q_2(x_2), q_2(x_1) \wedge q_1(x_2)}$$

At this point, in order to prove the entailment, it is necessary to continue the derivation and obtain a proof from each branch created by this application of SP. ◀

$$\mathrm{AX}\ \frac{\top}{\Gamma \vdash \Delta}\ \bigwedge \Gamma \models^{\mathcal{I}} \bigvee \Delta \qquad \mathrm{ID}\ \frac{\top}{\begin{array}{l}\Gamma\theta \vdash \Delta'\theta \\ \vdots \\ \Gamma \vdash \Delta\end{array}}\ \begin{array}{l}\theta \text{ is a flat, injective substitution} \\ \Delta \subseteq \Delta' \\ (\mathcal{R}_{\mathsf{Ind}})^* \cdot \mathrm{LU} \cdot (\mathcal{R}_{\mathsf{Ind}})^*\end{array}$$

Figure 2.5: Inference rule schemata for closing a valid branch of the proof.

The inference rule schema AX (Figure 2.5) closes the current branch of the proof, if the sequent from its consequent can be proved using a decision procedure for the underlying constraint logic, while treating all predicate symbols as uninterpreted function symbols of boolean sort.

Example 2.3.4. Consider the sequent $p_1(x_1) \vdash q_1(x_1), q_2(x_1)$ obtained from the application of SP in Example 2.3.3. We continue the derivation by applying LU to unfold $p_1(x_1)$ and RU to unfold $q_1(x_1)$. In order to arrive at an application of AX, we only look at the branch generated by LU for the rule $\langle p_1(x_1), \{x_1 \approx a\}\rangle$. We obtain the following derivation:

$$\mathrm{RU}\ \frac{x_1 \approx a \vdash x_1 \approx a, \exists y_1 \,.\, x_1 \approx g(y_1) \wedge q_1(y_1), q_2(x_1)}{\mathrm{LU}\ \dfrac{x_1 \approx a \vdash q_1(x_1), q_2(x_1)}{p_1(x_1) \vdash q_1(x_1), q_2(x_1)}}$$

Note how $x \approx a$ appears both on the left- and right-hand sides. Any decision procedure for FOL that treats q_1 and q_2 as uninterpreted functions will indicate that $x \approx a \models^{\mathcal{U}} x \approx a \vee (\exists y_1 \,.\, x \approx g(y_1) \wedge q_1(y_1)) \vee q_2(x)$ holds. Therefore, it is possible to apply AX:

$$\mathrm{AX}\ \frac{\top}{x_1 \approx a \vdash x_1 \approx a, \exists y_1 \,.\, x_1 \approx g(y_1) \wedge q_1(y_1), q_2(x_1)}$$

Thus, this particular branch of our derivation is closed and, moreover, it represents a valid branch for the proof we are seeking. ◀

The inference rules ID (Figure 2.5), short for *Infinite Descent*, work as follows. A sequent $\Gamma\theta \vdash \Delta'\theta$ denotes a valid entailment whenever the pivot $\Gamma \vdash \Delta$, encountered earlier in the proof, is a similar sequent up to the renaming of variables by a flat, injective substitution θ. The pivot condition $(\mathcal{R}_{\mathsf{Ind}})^* \cdot \mathrm{LU} \cdot (\mathcal{R}_{\mathsf{Ind}})^*$ asks that a rule of type LU is applied somewhere on the path between the pivot $\Gamma \vdash \Delta$ and the consequent $\Gamma\theta \vdash \Delta'\theta$ in the proof. Rules of type ID are sound if the inductive system is ranked (Definition 2.4.3), by an application

of Fermat's Infinite Descent principle [10]. The soundness of ID is based on the following argument, proven in more detail in Section 2.5.1.

Consider, by contradiction, that $\Gamma \vdash \Delta$ does not denote a valid entailment. From the soundness of $\mathcal{R}_{\mathsf{Ind}} \setminus \{\text{ID}\}$, we show how there exists a counterexample $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$, which can be propagated along the path from the pivot to the consequent of ID, such that we obtain $\nu' \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma\theta) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta'\theta)$ and ν' is strictly smaller than ν in a chosen well-founded ordering. This is due to the requirement that LU must be applied along this direct path, which, in turn, requires RD to also be applied, guaranteeing a strict decrease in the multiset image of the counterexample. Since θ is flat, injective and also surjective by construction, its inverse exists and we obtain a counterexample $\nu'\theta^{-1}$ for $\Gamma \vdash \Delta$, which is strictly smaller than ν. Then, for any infinite trace along which we can propagate a counterexample, because such a trace contains infinitely many direct paths (see Proposition 2.2.1) causing a strict decrease in the counterexample, we obtain an infinite strictly decreasing sequence of multisets, contradicting the well-foundedness of the chosen order. Thus $\bigwedge \Gamma \models^{\mathcal{I}}_{\mathcal{S}} \bigvee \Delta$ must hold and the proof branch can be closed.

Example 2.3.5. Similar to how we proceeded in Example 2.3.4, we consider the sequent $p_1(x_1) \vdash q_1(x_1), q_2(x_1)$ and apply LU to unfold $p(x)$ and RU twice to unfold both $q_1(x)$ and $q_2(x)$. This time, however, we only look at the branch generated by LU for the predicate rule $\langle p_1(x_1), \{x_1 \approx g(x_{11}), p_1(x_{11})\}\rangle$.

We then apply RD, which checks the following entailments:

$$x_1 \approx g(x_{11}) \models^{\mathcal{I}} x_1 \approx a \tag{2.1}$$

$$x_1 \approx g(x_{11}) \models^{\mathcal{I}} x_1 \approx b \tag{2.2}$$

$$x_1 \approx g(x_{11}) \models^{\mathcal{I}} \exists y_{11} \,.\, x_1 \approx g(y_{11}) \tag{2.3}$$

Entailments (2.1) and (2.2) two do not hold and their respective conjunctions from the right-hand side are ignored. Entailment (2.3) holds and is witnessed by the substitution $\theta = \{(y_{11}, x_{11})\}$. This entailment corresponds to two conjunctions from the right-hand side, from which we remove the constraint and the existential quantification, we apply θ, and add the results to the right-hand side of the next sequent.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\top}{p_1(x_{11}) \vdash q_1(x_{11}), q_2(x_{11})}\,\text{ID}}{x \approx g(x_{11}), p_1(x_{11}) \vdash x_1 \approx a, \exists y_{11} \,.\, x_1 \approx g(y_{11}) \wedge q_1(y_{11}),\; x_1 \approx b, \exists y_{11} \,.\, x \approx g(y_{11}) \wedge q_2(y_{11})}\,\text{RD}}{x_1 \approx g(x_{11}), p_1(x_{11}) \vdash x_1 \approx a, \exists y_{11} \,.\, x_1 \approx g(y) \wedge q_1(y_{11}), q_2(x_1)}\,\text{RU}}{x_1 \approx g(x_{11}), p_1(x_{11}) \vdash q_1(x_1), q_2(x_1)}\,\text{RU}}{p_1(x_{11}) \vdash q_1(x_{11}), q_2(x_{11})}\,\text{LU}$$

Thus, we obtain $p_1(x_{11}) \vdash q_1(x_{11}), q_2(x_{11})$ and we can apply ID using the pivot $p_1(x_1) \vdash q_1(x_1), q_2(x_1)$ to close this valid branch of our derivation. Note that, by combining Examples 2.3.4 and 2.3.3, we obtain a proof for the sequent $p_1(x_1) \vdash q_1(x_1), q_2(x_1)$. The other three sequents obtained from SP in Example 2.3.3 can be proved in a similar manner. ◀

2.3.2 The Inference Rule Set $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ for SL Entailments

In order to prove SL entailments, we modify the $\mathcal{R}_{\mathsf{Ind}}$ proof system from Section 2.3.1 such that the sets on the left-hand sides of sequents are interpreted as separating conjunctions, and

those on the right-hand sides are interpreted as disjunctions over separating conjunctions. Thus, we obtain the proof system $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}} = \{\mathrm{LU}, \mathrm{RU}_{\mathsf{SL}}, \mathrm{RD}_{\mathsf{SL}}, \wedge\mathrm{R}, \mathrm{SP}_{\mathsf{SL}}, \mathrm{AX}_{\mathsf{SL}}, \mathrm{ID}\}$, where the inference rule schemata $\mathrm{RU}_{\mathsf{SL}}$, $\mathrm{RD}_{\mathsf{SL}}$, $\mathrm{SP}_{\mathsf{SL}}$ and $\mathrm{AX}_{\mathsf{SL}}$ have been modified from their counterparts in $\mathcal{R}_{\mathsf{Ind}}$ and are depicted in Figures 2.6, 2.7, 2.8 and 2.9.

$$\mathrm{RU}_{\mathsf{SL}}\ \frac{\Gamma \vdash \{\exists \overline{\mathbf{y}}_i \,.\, * R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i)\}_{i=1}^{n}, \Delta \setminus p(\overline{\mathbf{x}})}{\Gamma \vdash \Delta}\ \begin{array}{l} p(\overline{\mathbf{x}}) \in \Delta,\ \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_n \text{ fresh}, \\ p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}}) \end{array}$$

Figure 2.6: Inference rule schema for right-hand side predicate unfolding in SL.

Applying $\mathrm{RU}_{\mathsf{SL}}$ (Figure 2.6) to a sequent is similar to applying its RU counterpart. When $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}})$, the singleton predicate atom $p(\overline{\mathbf{x}})$ on the right-hand side of a sequent is replaced with a set of formulae obtained by applying the separating conjunction over each $R_i, i \in [n]$, and existentially quantifying the fresh subgoal variables in them.

Example 2.3.6. Consider the inductive system from Example 1.4.2. In order to prove the entailment $ls^+ \models^{\mathsf{SL}} \widehat{ls}^+$, we start a derivation from the sequent $ls^+(x,y) \vdash \widehat{ls}^+(x,y)$. We apply LU on this sequent to unfold $ls^+(x,y)$ and consider only the branch corresponding to the predicate rule $\langle ls^+(x,y), \{y \approx z_2 \wedge x \mapsto z_1, ls^+(z_1,z_2)\}\rangle$. We obtain:

$$\mathrm{LU}\ \frac{y \approx z_2 \wedge x \mapsto z_1, ls^+(z_1,z_2) \vdash \widehat{ls}^+(x,y)}{ls^+(x,y) \vdash \widehat{ls}^+(x,y)}$$

We can further apply $\mathrm{RU}_{\mathsf{SL}}$ on the resulting sequent and unfold $\widehat{ls}^+(x,y)$:

$$\mathrm{RU}_{\mathsf{SL}}\ \frac{\begin{array}{r} y \approx z_2 \wedge x \mapsto z_1, ls^+(z_1,z_2) \vdash \exists u_1, u_2 \,.\, y \approx u_2 \wedge x \mapsto u_1 * ls^e(u_1,u_2), \\ \exists u_1, u_2 \,.\, y \approx u_2 \wedge x \mapsto u_1 * ls^o(u_1,u_2) \end{array}}{y \approx z_2 \wedge x \mapsto z_1, ls^+(z_1,z_2) \vdash \widehat{ls}^+(x,y)}$$

Note that, by unfolding, we have introduced the new subgoal variables z_1, z_2 on the left-hand side and the new, existentially quantified subgoal variables u_1, u_2 for each separating conjunction on the right-hand side. ◀

$$\mathrm{RD}_{\mathsf{SL}}\ \frac{p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\mathcal{Q}_j(\overline{\mathbf{y}}_j\theta) \mid \theta \in S_j\}_{j=1}^{i}}{\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n), p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\exists \overline{\mathbf{y}}_j \,.\, \psi_j(\overline{\mathbf{x}}, \overline{\mathbf{y}}_j) * \mathcal{Q}_j(\overline{\mathbf{y}}_j)\}_{j=1}^{k}}$$
$$\text{where } \phi \models^{\mathsf{SL}} \bigwedge_{j=1}^{i} \exists \overline{\mathbf{y}}_j \,.\, \psi_j,\ \ \phi \not\models^{\mathsf{SL}} \bigvee_{j=i+1}^{k} \exists \overline{\mathbf{y}}_j \,.\, \psi_j,\ \ S_j \subseteq \mathrm{VIS}(\phi, \psi_j), j \in [i]$$

Figure 2.7: Inference rule schema for reducing constraints in SL.

The inference rules of type $\mathrm{RD}_{\mathsf{SL}}$ (Figure 2.7) operate similarly to their RD counterparts, but check the entailments $\phi \models^{\mathsf{SL}} \exists \overline{\mathbf{y}}_j \,.\, \psi_j, j \in [k]$ between the left constraint and each of the right constraints. Here, $\mathrm{VIS}(\phi, \psi) = \{\theta : \bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j) \rightarrow \mathsf{set}(\overline{\mathbf{x}}) \cup \bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i) \mid \phi \models^{\mathsf{SL}} \psi\theta\}$. Furthermore, the form of the right-hand sides is changed to accommodate the new interpretation of the sequents and each $\mathcal{Q}_j, j \in [n]$, is a separating conjunction of predicate atoms.

Example 2.3.7. Continuing the derivation from Example 2.3.6, we apply $\mathrm{RD}_{\mathsf{SL}}$ on the antecedent of $\mathrm{RU}_{\mathsf{SL}}$:

$$\mathrm{RD}_{\mathsf{SL}}\ \frac{ls^+(z_1,z_2) \vdash ls^e(z_1,z_2), ls^o(z_1,z_2)}{y \approx z_2 \wedge x \mapsto z_1, ls^+(z_1,z_2) \vdash \exists u_1,u_2\,.\, y \approx u_2 \wedge x \mapsto u_1 * ls^e(u_1,u_2), \exists u_1,u_2\,.\, y \approx u_2 \wedge x \mapsto u_1 * ls^o(u_1,u_2)}$$

This application of $\mathrm{RD}_{\mathsf{SL}}$ checks the entailment $y \approx z_2 \wedge x \mapsto z_1 \models^{\mathsf{SL}} \exists u_1 \exists u_2\,.\, y \approx u_2 \wedge x \mapsto u_1$ for both parts of the right-hand side (since they happen to have the same constraint). The entailment holds and is witnessed by the substitution $\theta = (u_1, z_1), (u_2, z_2)$. The constraints are removed and θ is applied on both separating conjunctions from the right-hand side. ◀

$$\mathrm{SP}_{\mathsf{SL}}\ \frac{\langle p_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \vdash \{q^{\ell}_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}\rangle_{j=1}^{n^k}}{p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\mathcal{Q}_j(\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)\}_{j=1}^{k}}$$
$$\text{where } \mathsf{set}(\overline{\mathbf{x}}_i) \cap \mathsf{set}(\overline{\mathbf{x}}_j) = \emptyset, \forall i,j \in [n],$$
$$\mathcal{Q}_j = *_{i=1}^{n} q_i^j(\overline{\mathbf{x}}_j),\ \ \overline{\mathcal{Q}}_j = \langle q_1^j, \ldots, q_n^j\rangle, \forall j \in [k]$$
$$\mathcal{F}(\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_k) = \{f_1, \ldots, f_{n^k}\},\ \ \bar{\imath} \in [n]^{n^k}$$

Figure 2.8: Inference rule schema for splitting sequents without constraints in SL.

The inference rules of type $\mathrm{SP}_{\mathsf{SL}}$ (Figure 2.8) and $\mathrm{AX}_{\mathsf{SL}}$ (Figure 2.9) differ from their $\mathcal{R}_{\mathsf{Ind}}$ counterparts in the form and interpretation of sequents. The right-hand side for the consequent of $\mathrm{SP}_{\mathsf{SL}}$ contains separating conjunctions of predicate atoms, while the side condition of $\mathrm{AX}_{\mathsf{SL}}$ contains a separating conjunction over the the left-hand side of its consequent.

$$\mathrm{AX}_{\mathsf{SL}}\ \frac{\top}{\Gamma \vdash \Delta}\ *\Gamma \models^{\mathsf{SL}} \bigvee \Delta$$

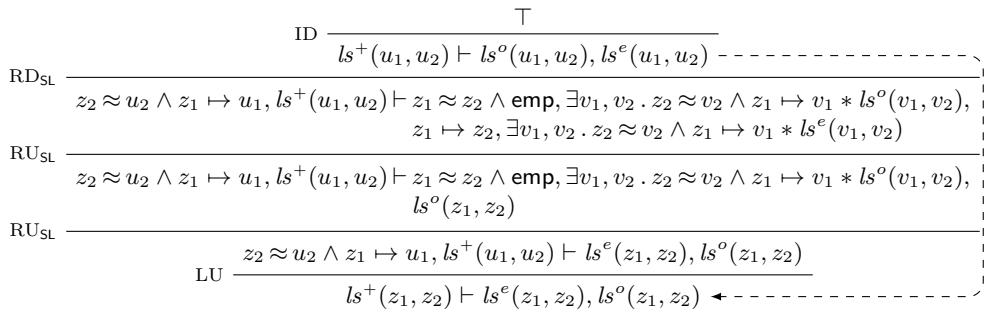
Figure 2.9: Axiom inference rule schema for SL.

Example 2.3.8. Continuing the derivation from Example 2.3.7, note that an application of $\mathrm{SP}_{\mathsf{SL}}$ is not needed. We apply LU to unfold $ls^+(z_1, z_2)$ and we separately analyse the two branches that result for each predicate rule in the definition of ls^+.

On the branch obtained by left unfolding $ls^+(z_1, z_2)$ with the predicate rule $\langle ls^+(z_1, z_2), \{z_1 \mapsto z_2\}\rangle$, we can apply $\mathrm{RU}_{\mathsf{SL}}$ to unfold $ls^o(z_1, z_2)$ on the right-hand side. Then we can close the branch by applying $\mathrm{AX}_{\mathsf{SL}}$, because any decision procedure for SL that treats ls^e as an uninterpreted function will indicate that $z_1 \mapsto z_2 \models^{\mathsf{SL}} ls^e(z_1, z_2) \vee z_1 \mapsto z_2 \vee \exists u_1, u_2\,.\, z_2 \approx u_2 \wedge z_1 \mapsto u_1 * ls^e(u_1, u_2)$ holds.

$$\mathrm{LU}\ \frac{\mathrm{RU}_{\mathsf{SL}}\ \dfrac{\mathrm{AX}_{\mathsf{SL}}\ \dfrac{\top}{z_1 \mapsto z_2 \vdash ls^e(z_1,z_2), z_1 \mapsto z_2, \exists u_1,u_2\,.\, z_2 \approx u_2 \wedge z_1 \mapsto u_1 * ls^e(u_1,u_2)}}{z_1 \mapsto z_2 \vdash ls^e(z_1,z_2), ls^o(z_1,z_2)}}{ls^+(z_1,z_2) \vdash ls^e(z_1,z_2), ls^o(z_1,z_2)}$$

On the branch obtained by left unfolding $ls^+(z_1, z_2)$ with the predicate rule $\langle ls^+(z_1, z_2), \{y \approx u_2 \wedge z_1 \mapsto u_1, ls^+(u_1, u_2)\}\rangle$, we apply $\mathrm{RU}_{\mathsf{SL}}$ twice to unfold both predicate atoms on the right-hand side, we eliminate the constraints by $\mathrm{RD}_{\mathsf{SL}}$ and then we can close the branch by ID.



2.4 Restricting the Set of Constraints

The undecidability results of Theorems 1.4.1 and 1.4.2 indicate that a number of restrictions on the constraints of an inductive system are required to ensure the soundness and completeness of our proof systems. Checking whether a given inductive system respects the restrictions that we propose is subject to the existence of a decision procedure of the $\exists^*\forall^*$-quantified fragment of the underlying logic, in which the constraints in every predicate rule are expressed. For first-order logic with the canonical interpretation, this problem is known as disunification and has been shown decidable in [13], with tighter complexity bounds in [39]. For separation logic, we provide our own decision procedures in Chapter 3. We only consider SL inductive systems whose constraints are $-\!\!*$-free, since the presence of the separating implication leads to the undecidability of the satisfiability problem for the $\exists^*\forall^*$-quantified fragment (see Section 3.3.1).

2.4.1 Well-quasi-orderings

We introduce the notion of quasi-ordering as it is defined in [31]. It is useful in defining ranked inductive systems in Section 2.4.3 and necessary for proving certain results related to derivation and proof trees in Section 2.5. Given a set D, a *quasi-ordering* or *qo* is a relation $\leq_{\mathrm{D}} \subseteq D \times D$ that is reflexive (i.e. $\forall d \in D \,.\, d \leq_{\mathrm{D}} d$) and transitive (i.e. $\forall d_1, d_2, d_3 \in D \,.\, d_1 \leq_{\mathrm{D}} d_2$ and $d_2 \leq_{\mathrm{D}} d_3 \Rightarrow d_1 \leq_{\mathrm{D}} d_3$). An infinite sequence $d_1, d_2, \ldots$ from D is *saturating* if it contains an increasing pair $d_i \leq_{\mathrm{D}} d_j$ for some $i < j$.

A quasi-ordering $\leq_{\mathrm{D}}$ on D is a *well-quasi-ordering* or *wqo* if every infinite sequence in D is saturating. A quasi-ordering $\leq_{\mathrm{D}}$ on D is *well-founded* or a *wfqo* if and only if there are no infinite decreasing sequences $d_1 >_{\mathrm{D}} d_2 >_{\mathrm{D}} \ldots$ in D, where $d_1 >_{\mathrm{D}} d_2 \Leftrightarrow d_2 \leq_{\mathrm{D}} d_1$ and $d_1 \neq d_2$. Every wqo is a wfqo, but not vice versa.

Example 2.4.1. A simple example of wqo is $(\mathbb{N}, \leq)$, the set of natural numbers together with the standard number ordering. Note that its integer counterpart, $(\mathbb{Z}, \leq)$, is not a wqo because it is not well-founded. On the other hand, $(\mathbb{N}, |)$, the set of natural numbers ordered by the divisibility relation, is a wfqo, but not a wqo, because the prime numbers form an infinite, non-saturating sequence.

A wfqo that is relevant for our work is $(\mathcal{T}_\Sigma, \sqsubseteq)$, as defined by the Herbrand interpretation in Section 1.4.1. Since $\mathcal{T}_\Sigma$ consists only of finite trees, it is impossible to build an infinite strictly decreasing sequence of subtrees, thus making it well-founded. On the other hand, infinite non-saturating sequences do exist. Consider, for instance, $\Sigma^{\mathsf{s}} = \{\sigma\}$ and $\Sigma^{\mathsf{f}} =$

$\{a^\sigma, f^{\sigma\sigma}, g^{\sigma\sigma}\}$. Then the sequence $f(a), f(g(a)), \ldots, f(g^n(a)), \ldots, n \in \mathbb{N}$ is infinite and non-saturating. Therefore $(\mathcal{T}_\Sigma, \sqsubseteq)$ is not a wqo. ◀

We extend any qo $(D, \leq_{\mathrm{D}})$ to the following order on the set of finite subsets of D. For all finite sets $Y, Z \in \mathcal{P}_{fin}(D)$, we have $Y \leq_{\mathrm{D}}^{\forall\exists} Z$ if and only if for all $d \in Y$ there exists $d' \in Z$ such that $d \leq_{\mathrm{D}} d'$. The following result is a consequence of Higman's lemma [21].

Lemma 2.4.1. Given a countable set D, if $(D, \leq_{\mathrm{D}})$ is a wqo, then $(\mathcal{P}_{fin}(D), \leq_{\mathrm{D}}^{\forall\exists})$ is also a wqo.

Proof. Let D^* be the set of finite sequences of elements from D, where u_i denotes the i-th element of $u \in D^*$ and $|u|$ is the length of u. The *subword order* $\leq_{\mathrm{D}}^{\mathsf{sw}}$ on D^* is defined as $u \leq_{\mathrm{D}}^{\mathsf{sw}} v$ if and only if there exists a strictly increasing mapping $f : [|u|] \to [|v|]$ such that $u_i = v_{f(i)}$ for all $i \in [|u|]$. The qo $\leq_{\mathrm{D}}$ on D induces the following ordering on the set D^*: for all $u, v \in D^*$, $u \leq_{\mathrm{D}}^* v$ if there exists $v' \leq_{\mathrm{D}}^{\mathsf{sw}} v$ such that $|u| = |v'|$ and $u_i \leq_{\mathrm{D}} v'_i$, for all $i \in [|u|]$. We have obtained a set of sequences D^* over a well-quasi-ordered alphabet D, with the subsequence relation $\leq_{\mathrm{D}}^*$ which allows the replacement of elements with smaller ones in the well-quasi-ordering $\leq_{\mathrm{D}}$ of D. Then, according to Higman's lemma [21], $(D^*, \leq_{\mathrm{D}}^*)$ is a wqo. Because D is countable, there is an indexing of its elements. Then we can uniquely represent each finite set $S \in \mathcal{P}_{fin}(D)$ as a finite word and $\mathcal{P}_{fin}(D)$ as subset of D^*, with the ordering $\leq_{\mathrm{D}}^{\forall\exists}$ on $\mathcal{P}_{fin}(D)$ being the equivalent of $\leq_{\mathrm{D}}^*$ on D^*. Consequently, $(\mathcal{P}_{fin}(D), \leq_{\mathrm{D}}^{\forall\exists})$ is also a wqo. □

A *multiset* over D is a mapping $M : D \to \mathbb{N}$. The multiset M is finite if $M(d) > 0$ for a finite number of elements $d \in D$. Given two sets A, B and a function $f : A \to B$, we define the multiset $f[\![A]\!] : f(A) \to \mathbb{N}$, where $f[\![A]\!](y) = \|\{x \mid x \in A, f(x) = y\}\|$ for every $y \in f(A)$.

The set of finite multisets over D is $\mathcal{M}(D)$. Given two multisets $M_1, M_2 \in \mathcal{M}(D)$, we define the membership and inclusion relations, as well as the union, intersection and difference operations:

$$d \in M_1 \Leftrightarrow \exists d \in D \,.\, M_1(d) > 0$$

$$M_1 \subseteq M_2 \Leftrightarrow \forall d \in D \,.\, M_1(d) \leq M_2(d)$$

$$(M_1 \cup M_2)(d) = \max(M_1(d), M_2(d)), \forall d \in D$$

$$(M_1 \cap M_2)(d) = \min(M_1(d), M_2(d)), \forall d \in D$$

$$(M_1 \setminus M_2)(d) = \max(0, M_1(d) - M_2(d)), \forall d \in D$$

The *multiset order* induced by $\leq_{\mathrm{D}}$ is defined as follows: $M \leq_{\mathrm{D}}^\dagger N$ if and only if either $M = N$, or there exists a non-empty finite multiset $X \subseteq N$ and a (possibly empty) multiset Y, such that $\forall y \in Y \, \exists x \in X \,.\, y <_{\mathrm{D}} x$ and $M = (N \setminus X) \cup Y$. In other words, M is obtained by replacing a non-empty subset of N with a possibly empty multiset of smaller elements. An equivalent definition is given by Huet and Oppen in [26]: $M <_{\mathrm{D}}^\dagger N$ if and only if $M \neq N$ and, for all $x \in D$, $M(x) > N(x) \Rightarrow \exists y \in D \,.\, x <_{\mathrm{D}} y$ and $M(y) < N(y)$. The following theorem was proved by Dershowittz and Manna in [15].

Theorem 2.4.2. $(\mathcal{M}(D), \leq_{\mathrm{D}}^\dagger)$ is a wfqo if and only if $(D, \leq_{\mathrm{D}})$ is a wfqo.

2.4.2 Non-filtering

The non-filtering condition for an inductive system requires that, given any models for the subgoals of a predicate rule, it be possible to find an all-encompassing model that also satisfies

the constraint of the rule. This property is important because inductive systems that are filtering lead, in general, to the undecidability of the entailment problem. Such is the case, for instance, of tree automata with equality and disequality constraints [12, Theorem 4.2.10].

The Non-filtering Property in FOL

Definition 2.4.1 (Non-filtering in FOL)**.** Given an interpretation $\mathcal{I}$ and an FOL inductive system $\mathcal{S}$, a predicate rule $\langle p(\overline{\mathbf{x}}), \{\phi, q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}\rangle \in \mathcal{S}$ is *non-filtering* if and only if, for all $i \in [n]$ and $\overline{\mathbf{v}}_i \in \mu\mathcal{S}^{\mathcal{I}}(q_i)$, there exists a valuation ν such that $\nu(\overline{\mathbf{x}}_i) = \overline{\mathbf{v}}_i$ and $\mathcal{I}, \nu \models \phi$. The inductive system $\mathcal{S}$ is *non-filtering* if and only if each rule in $\mathcal{S}$ is non-filtering.

Example 2.4.2. Consider the inductive system in Example 1.4.1. It is easy to see that it satisfies the non-filtering property. However, if we were to change, for instance, the predicate rule for p into

$$p(x) \leftarrow_{\mathcal{S}} x \approx f(x_1, x_2) \wedge x_1 \approx x_2, p_1(x_1), p_2(x_2)$$

then $\mathcal{S}$ would become filtering, as all the subgoals models for which the values of x_1 and x_2 differ would be rejected by the new predicate rule. ◀

Checking whether a given inductive system in FOL is non-filtering is undecidable, as shown by the following lemma.

Lemma 2.4.3. The problem "Given an inductive system $\mathcal{S}$ with FOL constraints, is $\mathcal{S}$ non-filtering?" is undecidable in the Herbrand interpretation.

Proof. We show the undecidability of the non-filtering problem in FOL by reduction from the disjointness problem for context-free languages, which is a known undecidable problem [25, Theorem 9.22 (a)].

As in the proof of Theorem 1.4.1, consider a context-free grammar $G = \langle \Xi, \Sigma, \Delta \rangle$, where Ξ is the set of nonterminals, Σ is the alphabet of terminals, Δ is a set of productions $(X, w) \in \Xi \times (\Xi \cup \Sigma)^*$, and $\mathcal{L}(G, X) \subseteq \Sigma^*$ denotes the language produced by G starting with the nonterminal X as axiom. The problem "Given two nonterminals $X, Y \in \Xi$, is it the case that $\mathcal{L}(G, X) \cap \mathcal{L}(G, Y) \neq \emptyset$?" is undecidable.

We can encode G as an inductive system $\mathcal{S}_G$, in the same way as done in the proof of Theorem 1.4.1, each nonterminal $Z \in \Xi$ corresponding to a predicate $Z(x, y)$. Then we add the following predicate rule:

$$\langle P(), \{x_1 \approx x_2 \wedge y_1 \approx y_2, X(x_1, y_1), Y(x_2, y_2)\}\rangle$$

Then $\mathcal{S}_G$ is non-filtering if and only if $\mathcal{L}(G, X) \cap \mathcal{L}(G, Y) \neq \emptyset$. □

This negative result prompts us to adopt a stronger sufficient condition requiring that $\forall\overline{\mathbf{x}}_1 \ldots \forall\overline{\mathbf{x}}_n \exists\overline{\mathbf{x}} \,.\, \phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ holds for each constraint $\phi \in \mathcal{S}^{\mathrm{c}}$. Checking this condition becomes decidable in the canonical Herbrand interpretation, because each constraint ϕ is a conjunction of equalities $s \approx t$ and disequalities $\neg(s \approx t)$ between terms $s, t \in \mathcal{T}_\Sigma(\mathsf{set}(\overline{\mathbf{x}}) \cup \bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i))$. Establishing the validity of $\forall\overline{\mathbf{x}}_1 \ldots \forall\overline{\mathbf{x}}_n \exists\overline{\mathbf{x}} \,.\, \phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ is equivalent to checking the unsatisfiability of the equational problem $\exists\overline{\mathbf{x}}_1 \ldots \exists\overline{\mathbf{x}}_n \forall\overline{\mathbf{x}} \,.\, \neg\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$.

Because the constraints in $\mathcal{S}^{\mathrm{c}}$ do not contain disjunctions, $\neg\phi$ is a disjunction of equalities and disequalities, thus it is trivially in conjunctive normal form. By [39, Theorem 5.2], the satisfiability of formulae $\exists\overline{\mathbf{y}}\forall\overline{\mathbf{x}} \,.\, \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ in conjunctive normal form is NP-complete. Thus, our validity problem (and, consequently, the problem of checking the non-filtering property using this sufficient condition) is in co-NP.

The Non-filtering Property in SL

Definition 2.4.2 (Non-filtering in SL)**.** Given an SL inductive system $\mathcal{S}$, a predicate rule $\langle p(\overline{\mathbf{x}}), \{\phi, q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}\rangle \in \mathcal{S}$ is *non-filtering* if and only if, for all $i \in [n]$ and $(\overline{\ell}_i, h_i) \in \mu\mathcal{S}^{\mathsf{SL}}(q_i)$, where all h_i are pairwise disjoint, there exists a valuation ν and a heap h, disjoint from $\biguplus_{i=1}^{n} h_i$, such that $\nu(\overline{\mathbf{x}}_i) = \overline{\ell}_i$ and $\nu, h \models^{\mathsf{SL}} \phi$, for all $i \in [n]$. The inductive system $\mathcal{S}$ is *non-filtering* if and only if each rule in $\mathcal{S}$ is *non-filtering*.

Example 2.4.3. The inductive system from Example 1.4.2 is non-filtering because there exists a model $\nu, h \models^{\mathsf{SL}} y \approx y' \wedge x \mapsto x'$, such that $\nu(x') = \ell_1$, $\nu(y') = \ell_2$ and $\mathrm{dom}(h) \cap \mathrm{dom}(h') = \emptyset$, for each given pair $(\langle \ell_1, \ell_2 \rangle, h')$ in $\mu\mathcal{S}^{\mathsf{SL}}(ls^+)$, $\mu\mathcal{S}^{\mathsf{SL}}(ls^e)$, or $\mu\mathcal{S}^{\mathsf{SL}}(ls^o)$. Since the set L is infinite, it is always possible to find a value $\nu(x) \notin \mathrm{dom}(h')$. ◀

The non-filtering property is decidable for inductive systems with SL constraints, unlike the case with FOL constraints under the Herbrand interpretation (Lemma 2.4.3). This is because it is possible to build an over-approximation of the least solution, that is both necessary and sufficient to characterize the satisfiability of a quantifier-free SL formula using predicate atoms [7]. The lemma below establishes the upper bound for the complexity of deciding whether a given inductive system is non-filtering.

Lemma 2.4.4. The problem "Given an inductive system $\mathcal{S}$ with $-\!\!*$-free SL constraints, is $\mathcal{S}$ non-filtering?" is in EXPSPACE.

Proof. The abstraction we need is defined as the least fixed point $\mu\mathcal{S}^\sharp$ of an operator $\mathbb{F}^\sharp_{\mathcal{S}}$. We introduce an *abstract assignment* $\mathcal{Y}$ that maps a predicate $p^{\sigma_1 \cdots \sigma_n}$ into a set of pairs (A, E), where $A \in \mathcal{P}([n])$ is a set of positions corresponding to allocated arguments and $E \subseteq [n] \times [n]$ is a set of equality constraints. For each model $(\langle \ell_1, \ldots, \ell_n \rangle, h) \in \mu\mathcal{S}^{\mathsf{SL}}(p)$ there exists a corresponding pair $(A, E) \in \mu\mathcal{S}^\sharp(p)$ in which $A = \{i \in [n] \mid \ell_i \in \mathrm{dom}(h)\}$ and $E = \{(i, j) \in [n] \times [n] \mid \ell_i = \ell_j\}$.

Given a quantifier-free SL formula φ, we define the following sets:

$$\begin{aligned}
\mathsf{alloc}^+(\varphi) &= \{x \in \mathrm{FV}(\varphi) \mid \varphi \wedge \exists z_1 \ldots \exists z_k \,.\, x \mapsto (z_1, \ldots, z_k) * \top \text{ is satisfiable}\} \\
\mathsf{alloc}^-(\varphi) &= \left\{x \in \mathrm{FV}(\varphi) \mid \varphi \models^{\mathsf{SL}} \exists z_1 \ldots \exists z_k \,.\, x \mapsto (z_1, \ldots, z_k) * \top\right\} \\
\mathsf{eq}(\varphi) &= \left\{(x, y) \in \mathrm{FV}(\varphi) \times \mathrm{FV}(\varphi) \mid \varphi \models^{\mathsf{SL}} x \approx y\right\}
\end{aligned}$$

Intuitively, $\mathsf{alloc}^+(\varphi)$ is the set of all variable that may represent allocated locations, $\mathsf{alloc}^-(\varphi)$ is the set of all variables that must represent allocated locations and $\mathsf{eq}(\varphi)$ is the equality relation induced by φ. Computing these sets can be done in polynomial space for general, $-\!\!*$-free SL formulae, using the decision procedures for quantifier-free and Bernays-Schoenfinkel-Ramsey SL formulae given in Chapter 3. For symbolic heaps, it can be done in polynomial time.

Dually, for any set $U \subseteq \mathsf{Var}$ and relation $V \subseteq \mathsf{Var} \times \mathsf{Var}$, we consider the following formulae that build the corresponding heap and equality constraints:

$$\begin{aligned}
\mathbf{A}(U) &= \mathop{\ast}_{x \in U} \exists z_1 \ldots \exists z_k \,.\, x \mapsto (z_1, \ldots, z_k) \\
\mathbf{E}(V) &= \bigwedge_{(x,y) \in V} x \approx y \wedge \bigwedge_{(x,y) \notin V} \neg(x \approx y)
\end{aligned}$$

For the remainder of this proof, consider a predicate rule R of $\mathcal{S}$, where $R = \langle p(\overline{\mathbf{x}}), \{\phi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m), q_1(\overline{\mathbf{y}}_1), \ldots, q_m(\overline{\mathbf{y}}_m)\}\rangle$, $\overline{\mathbf{x}} = \langle x_1, \ldots, x_n \rangle$ and $\overline{\mathbf{y}}_i = \langle y_1^i, \ldots, y_{n_i}^i \rangle$, $\forall i \in [m]$. For

any relation $Rel \subseteq \mathrm{FV}(\phi) \times \mathrm{FV}(\phi)$ and tuple $\overline{\mathbf{P}} = \langle (A_1, E_1), \ldots, (A_m, E_m) \rangle \in \mu\mathcal{S}^\sharp(q_1) \times \ldots \times \mu\mathcal{S}^\sharp(q_m)$, we define:

$$\omega_R(\overline{\mathbf{P}}, Rel) \equiv \mathop{\ast}_{i \in [m]} \mathbf{A}(\{x_j \mid j \in A_i\}) \wedge \mathbf{E}(\{(y_r^i, y_s^i) \mid (r,s) \in E_i, i \in [m]\}) \wedge \mathbf{E}(Rel)$$
$$\eta_R(\overline{\mathbf{P}}, Rel) \equiv \exists \overline{\mathbf{y}}_1 \ldots \exists \overline{\mathbf{y}}_m \,.\, \big(\phi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m) * \omega_R(\overline{\mathbf{P}}, Rel)\big)$$

We extend the abstract assignment $\mathcal{Y}$ to predicate rules and define $\mathcal{Y}(R)$ as the set of pairs (A, E) for which there exists a tuple $\overline{\mathbf{P}} \in \mu\mathcal{S}^\sharp(q_1) \times \ldots \times \mu\mathcal{S}^\sharp(q_m)$ and a relation $Rel \subseteq \mathrm{FV}(\phi) \times \mathrm{FV}(\phi)$ such that:

$$A = \{i \in [n] \mid x_i \in All,\ \mathsf{alloc}^-(\eta_R(\overline{\mathbf{P}}, Rel)) \subseteq All \subseteq \mathsf{alloc}^+(\eta_R(\overline{\mathbf{P}}, Rel))\}$$
$$E = \{(i,j) \in [n] \times [n] \mid (x_i, x_j) \in \mathsf{eq}(\eta_R(\overline{\mathbf{P}}, Rel))\}$$

If $\eta_R(\overline{\mathbf{P}}, Rel)$ is unsatisfiable, then $\mathsf{alloc}^-(\eta_R(\overline{\mathbf{P}}, Rel)) = \mathsf{set}(\overline{\mathbf{x}})$, $\mathsf{alloc}^+(\eta_R(\overline{\mathbf{P}}, Rel)) = \emptyset$ and there can be no choice for the set All – thus there exists no corresponding pair (A, E) either.

If $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1 \mid \ldots \mid R_m$ are all the predicate rules for p in $\mathcal{S}$, then the abstract assignment $\mathbb{F}^\sharp_{\mathcal{S}}(\mathcal{Y})$ maps the predicate p to the set $\mathbb{F}^\sharp_{\mathcal{S}}(\mathcal{Y})(p) = \bigcup_{i=1}^m \mathcal{Y}(R_i)$. Similarly to $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}$ and $\mu\mathcal{S}^{\mathsf{SL}}$ in Section 1.3.3, the operator $\mathbb{F}^\sharp_{\mathcal{S}}$ is monotone and continuous, with $\mu\mathcal{S}^\sharp$ as its least fixed point.

Considering again the arbitrary predicate rule $R \in \mathcal{S}$ described above, let $\overline{\mathbf{P}} = \langle (A_1, E_1), \ldots, (A_m, E_m) \rangle \in \mu\mathcal{S}^\sharp(q_1) \times \ldots \times \mu\mathcal{S}^\sharp(q_m)$ be a tuple of pairs and $Rel \subseteq \mathrm{FV}(\phi) \times \mathrm{FV}(\phi)$ a relation on variables, such that $\omega_R(\overline{\mathbf{P}}, Rel)$ is satisfiable. We claim that, if the formula $\phi * \omega_R(\overline{\mathbf{P}}, Rel)$ is satisfiable, then for each tuple of models $\langle (\overline{\ell}_1, h_1), \ldots, (\overline{\ell}_m, h_m) \rangle \in \mu\mathcal{S}^{\mathsf{SL}}(q_1) \times \ldots \times \mu\mathcal{S}^{\mathsf{SL}}(q_m)$, there exist a valuation ν and a heap h such that $\nu, h \models^{\mathsf{SL}} \phi$ and $\nu, \biguplus_{i=1}^m h_i \models^{\mathsf{SL}} \omega_R(\overline{\mathbf{P}}, Rel)$, where $\nu(\overline{\mathbf{y}}_i) = \overline{\ell}_i$, $\forall i \in [m]$, and $\mathrm{dom}(h) \cap (\bigcup_{i=1}^m \mathrm{dom}(h_i)) = \emptyset$.

The proof idea for this claim is that, because $\omega_R(\overline{\mathbf{P}}, Rel)$ specifies exactly those variables which are allocated, as well as those which are not, and the pairs of variables which are equal, along with the ones which are not, the truth value of $\phi * \omega_R(\overline{\mathbf{P}}, Rel)$ is invariant under the renaming of the values of $\mathsf{set}(\overline{\mathbf{x}}) \cup \bigcup_{i=1}^m \mathsf{set}(\overline{\mathbf{y}}_i)$, as long as the allocations and equalities are preserved. Moreover, each tuple of models $\langle (\overline{\ell}_1, h_1), \ldots, (\overline{\ell}_m, h_m) \rangle \in \mu\mathcal{S}^{\mathsf{SL}}(q_1) \times \ldots \times \mu\mathcal{S}^{\mathsf{SL}}(q_m)$ is a model of $\omega_R(\overline{\mathbf{P}}, Rel)$, for some $\overline{\mathbf{P}} = \langle (A_1, E_1), \ldots, (A_m, E_m) \rangle \in \mu\mathcal{S}^\sharp(q_1) \times \ldots \times \mu\mathcal{S}^\sharp(q_m)$ and $Rel \subseteq \mathrm{FV}(\phi) \times \mathrm{FV}(\phi)$. Then, for each predicate rule $R \in \mathcal{S}$ we need to check the satisfiability of $\phi * \omega_R(\overline{\mathbf{P}}, Rel)$, for each $\overline{\mathbf{P}}$ and Rel for which $\omega_R(\overline{\mathbf{P}}, Rel)$ is satisfiable.

Since there are finitely many variables in $\mathcal{S}$, for a predicate $p^{\sigma_1 \cdots \sigma_n} \in \mathcal{S}^{\mathsf{p}}$ the set of pairs (A, E) is finite, of cardinality at most 2^{n+n^2}. Then $\mu\mathcal{S}^\sharp(p)$ can be computed in an exponential number of steps[1], each step requiring polynomial space. These pairs can be stored in a table that requires $2^{\mathcal{O}(n^2)}$ space, indexed by $\mathcal{O}(n^2)$ bits, where each pair occupies $\mathcal{O}(n)$ bits. Checking the satisfiability of a formula $\phi * \omega_R(\overline{\mathbf{P}}, Rel)$ is possible in polynomial space.

In conclusion, we can check if a predicate rule in $\mathcal{S}$ is non-filtering, by checking the satisfiability of an exponential number of SL formulae, where each satisfiability check can be done in polynomial space. Thus, the overall complexity of checking if an SL inductive system is non-filtering is $\mathsf{EXPSPACE}$. □

[1] See [7, Lemma 4.6] for an analogous construction for inductive systems with symbolic heap constraints.

2.4.3 Ranked

The ranked condition requires that, in any predicate rule of an inductive system, each value assigned to a subgoal variable is strictly smaller (with respect to a chosen wfqo) than the value of some goal variable. This property ensures that the principle of Infinite Descent [10] can be applied to close a branch of the proof tree.

The Ranked Property in FOL

We fix an interpretation $\mathcal{I}$ and assume that $(\sigma^{\mathcal{I}}, \leq_{\mathcal{I},\sigma})$ is a wfqo, for each $\sigma \in \Sigma^{\mathsf{s}}$. Given a valuation ν and a set of variables $\mathbf{x} \subseteq \mathsf{Var}$, we write $[\![\mathbf{x}]\!]_\nu$ for the multiset $[\![\nu(x) \mid x \in \mathbf{x}]\!]$. For two valuations ν and γ and two multisets $[\![\mathbf{x}]\!]_\nu$ and $[\![\mathbf{y}]\!]_\gamma$, we define an order $\leq^{\ddagger}_{\mathcal{I}}$ and a strict version $<^{\ddagger}_{\mathcal{I}}$ such that $[\![\mathbf{x}]\!]_\nu \leq^{\ddagger}_{\mathcal{I}} [\![\mathbf{y}]\!]_\gamma$ and, respectively, $[\![\mathbf{x}]\!]_\nu <^{\ddagger}_{\mathcal{I}} [\![\mathbf{y}]\!]_\gamma$, if and only if for each $x^\sigma \in \mathbf{x}$ there exists $y^\sigma \in \mathbf{y}$, of the same sort σ, such that $\nu(x) \leq_{\mathcal{I},\sigma} \gamma(y)$ and, respectively, $\nu(x) <_{\mathcal{I},\sigma} \gamma(y)$. Given a chain $[\![\mathbf{x}_1]\!]_{\nu_1} \sim [\![\mathbf{x}_2]\!]_{\nu_2} \sim \ldots \sim [\![\mathbf{x}_k]\!]_{\nu_k}$, where $\sim$ is either $\geq^{\ddagger}_{\mathcal{I}}$ or $>^{\ddagger}_{\mathcal{I}}$, it easily follows that we have $[\![\mathbf{x}_1]\!]_{\nu_1} >^{\ddagger}_{\mathcal{I}} [\![\mathbf{x}_k]\!]_{\nu_k}$ if and only if $[\![\mathbf{x}_i]\!]_{\nu_i} >^{\ddagger}_{\mathcal{I}} [\![\mathbf{x}_{i+1}]\!]_{\nu_{i+1}}$ for at least some $i \in [k-1]$.

Proposition 2.4.5. Given a signature Σ and an interpretation $\mathcal{I}$, $(\mathcal{M}(\mathsf{Val}), <^{\ddagger}_{\mathcal{I}})$ is a wfqo provided that, for each $\sigma \in \Sigma^{\mathsf{s}}$, $(\sigma^{\mathcal{I}}, \leq_{\mathcal{I},\sigma})$ is a wfqo.

Proof. Consider two valuations ν and γ and two multisets $[\![\mathbf{x}]\!]_\nu$ and $[\![\mathbf{y}]\!]_\gamma$. Then $[\![\mathbf{x}]\!]_\nu <^{\ddagger}_{\mathcal{I}} [\![\mathbf{y}]\!]_\gamma$ trivially gives us $[\![\mathbf{x}]\!]_\nu = ([\![\mathbf{y}]\!]_\gamma \setminus [\![\mathbf{y}]\!]_\gamma) \cup [\![\mathbf{x}]\!]_\nu$, where for all $a \in [\![\mathbf{x}]\!]_\nu$ there exists $b \in [\![\mathbf{y}]\!]_\gamma$ such that $a <_{\mathcal{I},\sigma} b$ and $a, b \in \sigma^{\mathcal{I}}$ for some $\sigma \in \Sigma^{\mathsf{s}}$. Therefore, $[\![\mathbf{x}]\!]_\nu <^{\dagger}_{\mathcal{I}} [\![\mathbf{y}]\!]_\gamma$, where $<^{\dagger}_{\mathcal{I}}$ is the Manna-Dershowitz wfqo on multisets (see Theorem 2.4.2). Then an infinite strictly decreasing sequence in $<^{\ddagger}_{\mathcal{I}}$ would imply the existence of an infinite strictly decreasing sequence in $<^{\dagger}_{\mathcal{I}}$, contradicting [15, Theorem 1]. □

Definition 2.4.3 (Ranked in FOL)**.** Given an interpretation $\mathcal{I}$ and an FOL inductive system $\mathcal{S}$, let $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$ be any constraint in $\mathcal{S}^{\mathsf{c}}$ with goal variables $\mathbf{x}$ and subgoal variables $\bigcup_{i=1}^n \mathbf{x}_i$. Then $\mathcal{S}$ is *ranked* if, given any valuation ν such that $\mathcal{I}, \nu \models \phi$, we have $[\![\bigcup_{i=1}^n \mathsf{set}(\overline{\mathbf{x}}_i)]\!]_\nu <^{\ddagger}_{\mathcal{I}} [\![\mathsf{set}(\overline{\mathbf{x}})]\!]_\nu$.

Example 2.4.4. The inductive system from Example 1.4.1 is ranked. The only constraints involving subgoal variables are (i) $x \approx f(x_1, x_2)$ and (ii) $x \approx g(x_1)$. For each valuation ν we have $\nu(x_1) \sqsubset \nu(x)$ and $\nu(x_2) \sqsubset \nu(x)$, if ν satisfies the constraint (i), and $\nu(x_1) \sqsubset \nu(x)$, if ν satisfies the constraint (ii). ◀

Lemma 2.4.6. The problem "Given an inductive system $\mathcal{S}$ with FOL constraints, is $\mathcal{S}$ ranked in the subtree order $(\mathcal{T}_\Sigma, \sqsubseteq)$?" is in co-NP.

Proof. For each constraint $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$ with goal variables $\mathbf{x}$ and subgoal variables $\mathbf{x}_1 \cup \ldots \cup \mathbf{x}_n$, we need to check if the formula

$$\phi \wedge \neg\Big(\bigwedge_{y \in \mathbf{x}_1 \cup \ldots \cup \mathbf{x}_n} \bigvee_{x \in \mathbf{x}} y \sqsubset x \Big) = \phi \wedge \Big(\bigvee_{y \in \mathbf{x}_1 \cup \ldots \cup \mathbf{x}_n} \bigwedge_{x \in \mathbf{x}} (y \approx x \vee \neg(y \sqsubseteq x)) \Big) \tag{2.4}$$

is unsatisfiable. The satisfiability of the quantifier-free fragment of FOL with a subterm relation is an NP-complete problem [49]. Since the size of the formula (2.4) is polynomially bounded by the number of variables occurring in ϕ, if follows that the problem of checking if a given inductive system is ranked is in co-NP. □

The Ranked Property in SL

Since there are no relations other than equality defined on the set of locations L, there does not exist a natural wfqo on L. Therefore, we consider the following wfqo on heaps. For any $h_1, h_2 \in \mathsf{Heaps}$, we have $h_1 \trianglelefteq h_2$ if and only if there exists $h \in \mathsf{Heaps}$ such that $h_2 = h_1 \uplus h$. If, moreover, $h \neq \emptyset$, then we write $h_1 \triangleleft h_2$.

Definition 2.4.4 (Ranked in SL)**.** Given an SL inductive system $\mathcal{S}$, let $\langle p(\overline{\mathbf{x}}), \{\phi, q_1(\overline{\mathbf{x}}_1), \ldots, q_n(\overline{\mathbf{x}}_n)\}\rangle$ be any predicate rule in $\mathcal{S}$. Then $\mathcal{S}$ is *ranked* if, for any pairs $(\overline{\ell}_i, h_i) \in \mu\mathcal{S}(q_i)$, $i \in [n]$, where $h_1, \ldots, h_n$ are disjoint, there exists $(\overline{\ell}, h) \in \mu\mathcal{S}(p)$ such that $\biguplus_{i=1}^{n} h_i \triangleleft h$.

Note that this property targets only the predicate rules in $\mathcal{S}$ that have at least one subgoal and requires that their constraints do not admit an empty heap model. The constraints of predicate rules without subgoals, however, are allowed to admit models with empty heaps.

Example 2.4.5. The inductive system from Example 1.4.2 is ranked because each predicate rule with at least one subgoal has a constraint $y \approx y' \wedge x \mapsto x'$, which does not admit an empty heap model. ◀

Lemma 2.4.7. The problem "Given an inductive system $\mathcal{S}$ with SL constraints, is $\mathcal{S}$ ranked in the subheap order $(\mathsf{Heaps}, \trianglelefteq)$?" is in PSPACE. When considering symbolic heap constraints, the problem is in P.

Proof. Since all SL constraints in an inductive system $\mathcal{S}$ are quantifier-free SL formulae, we can determine if $\mathcal{S}$ is ranked by checking the validity of $\phi \models^{\mathsf{SL}} \neg\mathsf{emp}$, which is equivalent to checking the satisfiability of $\phi \wedge \mathsf{emp}$ for each constraint ϕ belonging to predicate rules with subgoals. We know that the latter is in PSPACE [11].

The PSPACE bound drops to P for inductive systems with symbolic heap constraints because a symbolic heap $\Pi \wedge \Theta$ admits an empty heap model if and only if Θ does not contain any atoms of the form $x \mapsto (y_1, \ldots, y_k)$. This check can be performed in polynomial time, proportional to the number of atoms (or the number of $*$ occurrences) in Θ. □

2.4.4 Finite Variable Instantiation

The finite variable instantiation condition requires that, for any two constraints ϕ and ψ from an inductive system, having the same goal variables, there exist a finite number of substitutions θ mapping each group of subgoal variables in ψ to a group of subgoal variables in ϕ such that ϕ entails $\psi\theta$. This property guarantees that all constraints can be eliminated from a proof sequent by instantiating the subgoal variables on the right-hand side using finitely many substitutions that map them to the subgoal variables from the left-hand side.

The Finite Variable Instantiation Property in FOL

When the constraints $\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ and $\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m)$ occur in a sequent

$$\phi, p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \exists\overline{\mathbf{y}}_1 \ldots \exists\overline{\mathbf{y}}_m \,.\, \psi \wedge q_1(\overline{\mathbf{y}}_1) \wedge \ldots \wedge q_m(\overline{\mathbf{y}}_m)$$

and the entailment $\phi \models^{\mathcal{I}} \exists\overline{\mathbf{y}}_1 \ldots \exists\overline{\mathbf{y}}_m \,.\, \psi$ is valid, we want to continue the derivation with the sequent

$$p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{q_1(\overline{\mathbf{y}}_1\theta) \wedge \ldots \wedge q_m(\overline{\mathbf{y}}_m\theta) \mid \theta \in S\}$$

where S is a finite set of substitutions witnessing the entailment, i.e. $\phi \models^{\mathcal{I}} \psi\theta$, for each $\theta \in S$. This elimination of constraints from sequents is generally sound but incomplete. For instance, the entailment

$$\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) \models^{\mathcal{I}} \exists \overline{\mathbf{y}}_1 \ldots \exists \overline{\mathbf{y}}_m \,.\, \psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m)$$

is valid if and only if $\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) \models^{\mathcal{I}} \psi'(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$, where ψ' is obtained from ψ by replacing each $y \in \bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j)$ with a Skolem function symbol $f_y(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ not occurring in ϕ or ψ[2].

A complete proof rule based on this replacement has to consider every possible interpretation of these Skolem witnesses. However, in general, this is impossible, because the definitions of these functions are not bound to any particular form. In order to achieve completeness, we require that these functions are always flat substitutions defined on the quantified variables $\bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j)$ and ranging over the free variables of the entailment $\mathsf{set}(\overline{\mathbf{x}}) \cup \bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i)$. This condition ensures, moreover, that there are finitely many possible interpretations of these Skolem witnesses.

Definition 2.4.5 (Finite variable instantiation in FOL)**.** An FOL inductive system $\mathcal{S}$ has the *finite instantiation (fi)* property if and only if for any two constraints $\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ and $\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m)$ from $\mathcal{S}$, with goal variables $\mathsf{set}(\overline{\mathbf{x}})$ and subgoal variables $\bigcup_{i=1}^{n} \overline{\mathbf{x}}_i$ and $\bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j)$, respectively, the set $\mathrm{VIS}(\phi, \psi) = \{\theta : \bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j) \to \mathcal{T}_\Sigma(\mathsf{set}(\overline{\mathbf{x}}) \cup \bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i)) \mid \phi \models^{\mathcal{I}} \psi\theta\}$ is finite. Moreover, $\mathcal{S}$ has the *finite variable instantiation (fvi)* property if, given any $\theta \in \mathrm{VIS}(\phi, \psi)$, for all $j \in [m]$ there exists $i \in [n]$ such that $\mathsf{set}(\overline{\mathbf{y}}_j)\theta = \mathsf{set}(\overline{\mathbf{x}}_i)$.

Note that, whenever an FOL inductive system $\mathcal{S}$ has the fvi property, a constraint $\phi(\overline{\mathbf{x}})$ with no subgoal variables cannot entail a constraint $\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m)$, where $\bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j) \neq \emptyset$. If this were the case, $\phi(\overline{\mathbf{x}}) \models^{\mathcal{I}} \exists \overline{\mathbf{y}}_1 \ldots \exists \overline{\mathbf{y}}_m \,.\, \psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m)$ would imply that $\mathrm{VIS}(\phi, \psi) \neq \emptyset$. But then each flat substitution $\theta \in \mathrm{VIS}(\phi, \psi)$ would have an empty range, which is not possible.

Example 2.4.6. Consider the inductive system in Example 1.4.1. It can easily be shown that it has the fvi property. Take, for instance, the constraints $\phi \equiv x \approx f(x_1, x_2)$ and $\psi \equiv x \approx f(y_1, y_2)$. The entailment $\phi \models^{\mathcal{H}} \exists y_1 \exists y_2 \,.\, \psi$ is witnessed by a single substitution θ with $\theta(x_1) = y_1$ and $\theta(x_2) = y_2$, which means that $\mathrm{VIS}(\phi, \psi) = \{\theta\}$. ◀

The following lemma gives an upper bound for the complexity of checking whether whether a given FOL inductive system has the fvi property under the canonical Herbrand interpretation. It is unclear, for now, whether the bound can be tightened, because the exact complexity of the satisfiability of equational problems is still unknown, in general.

Lemma 2.4.8. The problem "Given an FOL inductive system $\mathcal{S}$, does $\mathcal{S}$ have the fvi property?" is in NEXPTIME under the Herbrand interpretation. If there exists a constant $K > 0$, independent of the input, such that for each constraint $\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ in $\mathcal{S}$, with goal variables $\overline{\mathbf{x}}$ and subgoal variables $\bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i)$, respectively, we have $\|\overline{\mathbf{x}}_i\| \leq K$, then the problem is in NP.

Proof. An FOL inductive system $\mathcal{S}$ has the fvi property if, for any two constraints $\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ and $\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m)$ in $\mathcal{S}^{\mathrm{c}}$, with goal variables $\mathsf{set}(\overline{\mathbf{x}})$ and subgoal variables $\bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i)$

[2] We assume w.l.o.g. that these function symbols belong to the signature, i.e. $f_y \in \Sigma^{\mathsf{f}}$.

and $\bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j)$, respectively, the following entailment is not valid:

$$\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) \models^{\mathcal{H}} \exists \overline{\mathbf{y}}_1 \ldots \exists \overline{\mathbf{y}}_m \,.\, \left(\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m) \wedge \bigvee_{j=1}^{m} \bigwedge_{i=1}^{n} \neg(\overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j) \right)$$

where $\overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j$ is shorthand for $\left(\bigwedge_{y \in \mathsf{set}(\overline{\mathbf{y}}_j)} \bigvee_{x \in \mathsf{set}(\overline{\mathbf{x}}_i)} x \approx y \right) \wedge \left(\bigwedge_{x \in \mathsf{set}(\overline{\mathbf{x}}_i)} \bigvee_{y \in \mathsf{set}(\overline{\mathbf{y}}_j)} x \approx y \right)$. This entailment is equivalent to the formula

$$\forall \overline{\mathbf{x}} \forall \overline{\mathbf{x}}_1 \ldots \forall \overline{\mathbf{x}}_n \,.\, \left(\neg\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) \vee \exists \overline{\mathbf{y}}_1 \ldots \exists \overline{\mathbf{y}}_m \,.\, \left(\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m) \wedge \bigvee_{j=1}^{m} \bigwedge_{i=1}^{n} \neg(\overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j) \right) \right)$$

and checking its validity is equivalent to checking the unsatisfiability of the formula

$$\exists \overline{\mathbf{x}} \exists \overline{\mathbf{x}}_1 \ldots \exists \overline{\mathbf{x}}_n \,.\, \left(\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) \wedge \forall \overline{\mathbf{y}}_1 \ldots \forall \overline{\mathbf{y}}_m \,.\, \left(\neg\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m) \vee \bigwedge_{j=1}^{m} \bigvee_{i=1}^{n} \overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j \right) \right)$$

which, in turn, can be rewritten as

$$\exists \overline{\mathbf{x}} \exists \overline{\mathbf{x}}_1 \ldots \exists \overline{\mathbf{x}}_n \forall \overline{\mathbf{y}}_1 \ldots \forall \overline{\mathbf{y}}_m \,.\, \left(\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) \wedge \bigwedge_{j=1}^{m} \left(\neg\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m) \vee \bigvee_{i=1}^{n} \overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j \right) \right)$$

This formula is not in conjunctive normal form (CNF) and expanding $\overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j$ to obtain the CNF causes an exponential blowup. Since checking the satisfiability of an equational problem in CNF is NP-complete, the above check can be performed in NEXPTIME. If the size of each set of subgoal variables is bound to a constant K, independent of the input, the size of each clause in the CNF expansion of the above formula is bound by a constant. Since there are at most polynomially many such constants, we can apply [39, Theorem 5.2] to obtain the NP upper bound. □

The Finite Variable Instantiation Property in SL

Definition 2.4.6 (Finite variable instantiation in SL)**.** An SL inductive system $\mathcal{S}$ has the *finite instantiation (fi)* and *finite variable instantiation (fvi)* properties under the same conditions as the ones in Definition 2.4.5, where $\mathrm{VIS}(\phi, \psi) = \{\theta : \bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j) \to \mathsf{set}(\overline{\mathbf{x}}) \cup \bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i) \mid \phi \models^{\mathsf{SL}} \psi\theta\}$ for any two constraints $\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ and $\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m)$ from $\mathcal{S}$, with goal variables $\mathsf{set}(\overline{\mathbf{x}})$ and subgoal variables $\bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i)$ and $\bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j)$, respectively.

Example 2.4.7. The system from Example 1.4.2 has the fvi property, because the entailment $y \approx y' \wedge x \mapsto x' \models^{\mathsf{SL}} \exists y'' \exists x'' \,.\, y \approx y'' \wedge x \mapsto x''$ is witnessed by a single substitution θ with $\theta(x'') = x'$ and $\theta(y'') = y'$. ◀

Lemma 2.4.9. The problem "Given an SL inductive system $\mathcal{S}$, does $\mathcal{S}$ have the fvi property?" is in PSPACE if $\mathcal{S}$ has quantifier-free and $-\!\!*$-free SL constraints, and in Σ_2^{P} if $\mathcal{S}$ has symbolic heap constraints.

Proof. Similarly to the proof for Lemma 2.4.8, an SL inductive system $\mathcal{S}$ has the fvi property if, for any two constraints $\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n)$ and $\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m)$ in $\mathcal{S}^{\mathsf{c}}$, with goal variables $\mathsf{set}(\overline{\mathbf{x}})$ and subgoal variables $\bigcup_{i=1}^{n} \mathsf{set}(\overline{\mathbf{x}}_i)$ and $\bigcup_{j=1}^{m} \mathsf{set}(\overline{\mathbf{y}}_j)$, respectively, the following entailment is not valid:

$$\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) \models^{\mathsf{SL}} \exists \overline{\mathbf{y}}_1 \ldots \exists \overline{\mathbf{y}}_m \,.\, \left(\psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m) \wedge \bigvee_{j=1}^{m} \bigwedge_{i=1}^{n} \neg(\overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j) \right)$$

where $\overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j$ is shorthand for $\left(\bigwedge_{y \in \mathsf{set}(\overline{\mathbf{y}}_j)} \bigvee_{x \in \mathsf{set}(\overline{\mathbf{x}}_i)} x \approx y\right) \wedge \left(\bigwedge_{x \in \mathsf{set}(\overline{\mathbf{x}}_i)} \bigvee_{y \in \mathsf{set}(\overline{\mathbf{y}}_j)} x \approx y\right)$. Just as in the proof for Lemma 2.4.8, this entailment is valid only if the following formula is unsatisfiable:

$$\exists \overline{\mathbf{x}} \, \exists \overline{\mathbf{x}}_1 \ldots \exists \overline{\mathbf{x}}_n \, \forall \overline{\mathbf{y}}_1 \ldots \forall \overline{\mathbf{y}}_m \, . \left(\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n) \wedge \bigwedge_{j=1}^{m} \left(\neg \psi(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m) \vee \bigvee_{i=1}^{n} \overline{\mathbf{x}}_i \cong \overline{\mathbf{y}}_j \right) \right)$$

We know that checking the satisfiability for the above formula is in PSPACE when ϕ and ψ are quantifier-free and $-\!\!*$-free SL formulae (see Section 3.3.1) and, thus, the fvi problem is in PSPACE. If, however, ϕ and ψ are symbolic heaps, the initial entailment problem is in Π_2^{P} [1, Theorem 6], thus the fvi problem is in Σ_2^{P}. □

2.4.5 Non-overlapping

The non-overlapping condition requires that, if any two constraints from an inductive system overlap (i.e. have at least one model in common), then one must entail the other. This property is required for completeness and is also related to the elimination of constraints from sequents.

The Non-overlapping Property in FOL

Definition 2.4.7 (Non-overlapping in FOL)**.** Given an interpretation $\mathcal{I}$, an FOL inductive system $\mathcal{S}$ is *non-overlapping* if, for any two constraints $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $\psi(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m)$ in $\mathcal{S}^{\mathrm{c}}$, with goal variables $\mathbf{x}$ and subgoal variables $\bigcup_{i=1}^{n} \mathbf{x}_i$ and $\bigcup_{j=1}^{m} \mathbf{y}_j$, respectively, $\phi \wedge \psi$ is satisfiable only if $\phi \models^{\mathcal{I}} \exists \mathbf{y}_1 \ldots \exists \mathbf{y}_m \, . \, \psi$.

Note that, for a non-overlapping system, if $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n) \wedge \psi(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m)$ is a satisfiable conjunction of constraints, then the formulae $\exists \mathbf{x}_1 \ldots \exists \mathbf{x}_n \, . \, \phi$ and $\exists \mathbf{y}_1 \ldots \mathbf{y}_m \, . \, \psi$ are equivalent.

Example 2.4.8. The FOL inductive system from Example 1.4.1 is non-overlapping. Take, for instance, the constraints $x \approx f(x_1, x_2)$ and $x \approx f(y_1, y_2)$. Then $x \approx f(x_1, x_2) \wedge x \approx f(y_1, y_2)$ is satisfiable and $x \approx f(x_1, x_2) \models^{\mathcal{H}} \exists y_1 \exists y_2 \, . \, x \approx f(y_1, y_2)$ holds. However, if we take the constraints $x \approx f(x_1, x_2)$ and $x \approx g(y_1)$, then $x \approx f(x_1, x_2) \wedge x \approx g(y_1)$ is unsatisfiable. ◀

Lemma 2.4.10. The problem "Given an FOL inductive system $\mathcal{S}$, does $\mathcal{S}$ have the non-overlapping property?" is in NP under the Herbrand interpretation.

Proof. Given an FOL inductive system $\mathcal{S}$, in order to determine if $\mathcal{S}$ has the non-overlappig property, we need to perform two checks for any pairs of constraints $\phi, \psi \in \mathcal{S}^{\mathrm{c}}$ with goal variables $\mathbf{x}$ and subgoal variables $\bigcup_{i=1}^{n} \mathbf{x}_i$ and $\bigcup_{j=1}^{m} \mathbf{y}_j$, respectively: (i) whether $\phi \wedge \psi$ is satisfiable and (ii) whether $\phi \models^{\mathcal{I}} \exists \mathbf{y}_1 \ldots \exists \mathbf{y}_m \, . \, \psi$ is valid. Checking the validity of the entailment in (ii) can be reduced to checking whether the formula $\exists \mathbf{x} \, \exists \mathbf{x}_1 \ldots \exists \mathbf{x}_n \, \forall \mathbf{y}_1 \ldots \forall \mathbf{y}_m \, . \, \phi \wedge \neg \psi$ is unsatisfiable.

Under the Herbrand interpretation, since ϕ and ψ are both conjunctions of literals, $\phi \wedge \psi$ and $\phi \wedge \neg \psi$ are in conjunctive normal form. Then both problems (i) and (ii) are in NP [39], and thus the non-overlapping problem under the Herbrand interpretation is in NP. □

The Non-overlapping Property in SL

Definition 2.4.8 (Non-overlapping in SL)**.** An SL inductive system $\mathcal{S}$ is *non-overlapping* under similar conditions as the ones in Definition 2.4.7, where, for any two constraints $\phi(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $\psi(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m)$ in $\mathcal{S}^{\mathsf{c}}$, with goal variables $\mathbf{x}$ and subgoal variables $\bigcup_{i=1}^{n} \mathbf{x}_i$ and $\bigcup_{j=1}^{m} \mathbf{y}_j$, respectively, $\phi \wedge \psi$ is satisfiable only if $\phi \models^{\mathsf{SL}} \exists \mathbf{y}_1 \ldots \exists \mathbf{y}_m \,.\, \psi$.

Example 2.4.9. The SL inductive system in Example 1.4.2 is non-overlapping. Take, for instance, the constraints $y \approx y' \wedge x \mapsto x'$ and $y \approx y'' \wedge x \mapsto x''$. It is easy to see that $(y \approx y' \wedge x \mapsto x') \wedge (y \approx y'' \wedge x \mapsto x'')$ is satisfiable and that the entailment $y \approx y' \wedge x \mapsto x' \models^{\mathsf{SL}} \exists x'' \exists y'' \,.\, y \approx y'' \wedge x \mapsto x''$ holds. ◀

Lemma 2.4.11. The problem "Given an SL inductive system $\mathcal{S}$, does $\mathcal{S}$ have the non-overlapping property?" is in PSPACE if $\mathcal{S}$ has quantifier-free and $-\!\!*$-free SL constraints, and in Π_2^{P} if $\mathcal{S}$ has symbolic heap constraints.

Proof. Similarly to the FOL case, as shown in the proof of Lemma 2.4.10, given an SL inductive system $\mathcal{S}$, in order to determine if $\mathcal{S}$ has the non-overlapping property, we need to perform two checks for any pairs of constraints $\phi, \psi \in \mathcal{S}^{\mathsf{c}}$ with goal variables $\mathbf{x}$ and subgoal variables $\bigcup_{i=1}^{n} \mathbf{x}_i$ and $\bigcup_{j=1}^{m} \mathbf{y}_j$, respectively: (i) $\phi \wedge \psi$ is satisfiable and (ii) $\phi \models^{\mathsf{SL}} \exists \mathbf{y}_1 \ldots \exists \mathbf{y}_m \,.\, \psi$ is valid (or, conversely, that $\exists \mathbf{x} \exists \mathbf{x}_1 \ldots \exists \mathbf{x}_n \forall \mathbf{y}_1 \ldots \forall \mathbf{y}_m \,.\, \phi \wedge \neg \psi$ is unsatisfiable).

When ϕ and ψ are quantifier-free and $-\!\!*$-free SL formulae, then (ii) is in PSPACE (see Section 3.3.1) and, thus, the non-overlapping problem is in PSPACE. If, however, ϕ and ψ are symbolic heaps, then the satisfiability problem (i) for symbolic heaps is in NP [11] and the entailment (ii) between existentially quantified symbolic heaps is Π_2^{P}-complete [1], thus the non-overlapping problem is in Π_2^{P}. □

2.5 Soundness and Completeness

In this section we address the soundness and completeness of the inference rule sets $\mathcal{R}_{\mathsf{Ind}}$ and $\mathcal{R}_{\mathsf{Ind}}^{\mathsf{SL}}$, and show that they are sound for entailments in a ranked inductive system $\mathcal{S}$. Completeness is guaranteed if, moreover, $\mathcal{S}$ is non-filtering, non-overlapping and has the fvi property. Note that both soundness and completeness are independent of any particular interpretation and rely only on the restrictions laid out in Section 2.4.

2.5.1 The Soundness and Completeness of $\mathcal{R}_{\mathsf{Ind}}$

Soundness

We develop our argument for the soundness of $\mathcal{R}_{\mathsf{Ind}}$ by first showing how every inference rule in $\mathcal{R}_{\mathsf{Ind}} \setminus \{\mathrm{ID}\}$ is locally sound. To this end, we give the following lemma, which relies on an important result from [24] and is integral to showing that SP is locally sound.

Lemma 2.5.1. Consider an FOL inductive system $\mathcal{S}$, the predicates $p_1, \ldots, p_n \in \mathcal{S}^{\mathsf{p}}$ and the tuples of predicates $\overline{\mathcal{Q}}_j = \langle q_1^j, \ldots, q_n^j \rangle \in (\mathcal{S}^{\mathsf{p}})^n, j \in [k]$. Then

$$\mu\mathcal{S}^{\mathcal{I}}(p_1) \times \ldots \times \mu\mathcal{S}^{\mathcal{I}}(p_n) \subseteq \bigcup_{j=1}^{k} \mu\mathcal{S}^{\mathcal{I}}(q_1^j) \times \ldots \times \mu\mathcal{S}^{\mathcal{I}}(q_n^j)$$

if and only if there exists a tuple $\bar{\imath} \in [n]^{n^k}$, such that

$$\forall j \in [n^k] \,.\, \mu\mathcal{S}^{\mathcal{I}}(p_{\bar{\imath}_j}) \subseteq \bigcup\{\mu\mathcal{S}^{\mathcal{I}}(q^{\ell}_{\bar{\imath}_j}) \mid \ell \in [k], f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}$$

where $\mathcal{F}(\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_k) = \{f_1, \ldots, f_{n^k}\}$ are the choice functions over $(\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_k)$.

Proof. From [24, Theorem 1], it follows that

$$\begin{aligned}
&\mu\mathcal{S}^{\mathcal{I}}(p_1) \times \ldots \times \mu\mathcal{S}^{\mathcal{I}}(p_n) \subseteq \bigcup_{j=1}^{k} \mu\mathcal{S}^{\mathcal{I}}(q^j_1) \times \ldots \times \mu\mathcal{S}^{\mathcal{I}}(q^j_n) && \Leftrightarrow \\
&\bigwedge_{j=1}^{n^k} \bigvee_{i=1}^{n} \Big(\mu\mathcal{S}^{\mathcal{I}}(p_i) \subseteq \bigcup\{\mu\mathcal{S}^{\mathcal{I}}(q^{\ell}_i) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = i\}\Big) && \Leftrightarrow \\
&\bigvee_{\bar{\imath} \in [n]^{n^k}} \bigwedge_{j=1}^{n^k} \Big(\mu\mathcal{S}^{\mathcal{I}}(p_{\bar{\imath}_j}) \subseteq \bigcup\{\mu\mathcal{S}^{\mathcal{I}}(q^{\ell}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}\Big)
\end{aligned}$$

The last step describes the translation of the formula from conjunctive normal form to disjunctive normal form. □

The local soundness of $\mathcal{R}_{\mathsf{Ind}} \setminus \{\mathrm{ID}\}$ means that, if the consequent $\Gamma \vdash \Delta$ denotes an invalid entailment, then at least one of its antecedents also denotes an invalid entailment. Moreover, their respective counterexamples can be related by a wfqo.

Lemma 2.5.2. Given an FOL inductive system $\mathcal{S}$ that is ranked in the interpretation $\mathcal{I}$, for each instance of an inference rule schema in $\mathcal{R}_{\mathsf{Ind}} \setminus \{\mathrm{ID}\}$, having the consequent $\Gamma \vdash \Delta$ and antecedents $\Gamma_i \vdash \Delta_i$ with $i \in [n]$, and each valuation $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$, there exists $\nu_i \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma_i) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta_i)$ for some $i \in [n]$ such that $[\![\mathrm{FV}(\Gamma_i)]\!]_{\nu_i} \leq^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma)]\!]_{\nu}$.

Proof. For AX, soundness follows from the side condition and its consequent admits no counterexamples, thus the lemma is trivially true in this case. We analyse LU, RU, RD, ∧R and SP individually, as follows.

(LU) Let $p(\overline{\mathbf{x}}) \in \Gamma$ be a predicate atom, where $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}})$. The antecedents of $\Gamma \vdash \Delta$ are $\Gamma_i \vdash \Delta_i \equiv R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i), \Gamma \setminus p(\overline{\mathbf{x}}) \vdash \Delta$, where $i \in [n]$ and each $\overline{\mathbf{y}}_i$ is a tuple of fresh variables. In this case, the least solution of Γ is

$$\begin{aligned}
\mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge \Gamma\right) &= \mu\mathcal{S}^{\mathcal{I}}\left(p(\overline{\mathbf{x}}) \wedge \bigwedge(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right) = \mu\mathcal{S}^{\mathcal{I}}(p(\overline{\mathbf{x}})) \cap \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \left(\bigcup_{i=1}^{n} \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge R_i(\overline{\mathbf{x}})\right)\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \bigcup_{i=1}^{n} \left(\mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge R_i(\overline{\mathbf{x}})\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right)\right) \\
&= \bigcup_{i=1}^{n} \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge R_i(\overline{\mathbf{x}}) \wedge \bigwedge(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right)
\end{aligned}$$

If there exists $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$, then $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge R_i(\overline{\mathbf{x}}) \wedge \bigwedge(\Gamma \setminus \{p(\overline{\mathbf{x}})\})) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$ for some $i \in [n]$ and there also exists $\nu_i \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i) \wedge \bigwedge(\Gamma \setminus \{p(\overline{\mathbf{x}})\})) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$ such that for every $x \in \mathrm{FV}(\Gamma)$ we have $\nu_i(x) = \nu(x)$. Furthermore, because $\mathcal{S}$ is ranked, $[\![\mathsf{set}(\overline{\mathbf{x}}_i)]\!]_{\nu_i} <^{\ddagger}_{\mathcal{I}} [\![\mathsf{set}(\overline{\mathbf{x}})]\!]_{\nu}$. Since $\mathrm{FV}(\Gamma_i) = \mathrm{FV}(\Gamma) \cup \mathsf{set}(\overline{\mathbf{x}}_i)$ and $\mathsf{set}(\overline{\mathbf{x}}) \subseteq \mathrm{FV}(\Gamma)$, it follows that $[\![\mathrm{FV}(\Gamma_i)]\!]_{\nu_i} \leq^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma)]\!]_{\nu}$.

(RU) Let $p(\overline{\mathbf{x}}) \in \Delta$ be a predicate atom, where $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}})$. Then $\Gamma \vdash \Delta$ has only one antecedent $\Gamma_1 \vdash \Delta_1 \equiv \Gamma \vdash \exists \mathbf{y}_1. \bigwedge R_1(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1), \ldots, \exists \mathbf{y}_n. \bigwedge R_n(\overline{\mathbf{x}}, \overline{\mathbf{y}}_n), \Delta \setminus p(\overline{\mathbf{x}})$, where each $\overline{\mathbf{y}}_i$ with $i \in [n]$ is a tuple of fresh variables. Note that $\mathrm{FV}(\Gamma_1) = \mathrm{FV}(\Gamma)$. In this case, the least solution of Δ is

$$\begin{aligned}
\mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta\right) &= \mu\mathcal{S}^{\mathcal{I}}\left(p(\overline{\mathbf{x}}) \vee \bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) = \mu\mathcal{S}^{\mathcal{I}}(p(\overline{\mathbf{x}})) \cup \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \left(\bigcup_{i=1}^{n} \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge R_i(\overline{\mathbf{x}})\right)\right) \cup \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \left(\bigcup_{i=1}^{n} \mu\mathcal{S}^{\mathcal{I}}\left(\exists \overline{\mathbf{y}}_i. \bigwedge R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i)\right)\right) \cup \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee_{i=1}^{n} \exists \overline{\mathbf{y}}_i. \bigwedge R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i)\right) \cup \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee_{i=1}^{n} \exists \overline{\mathbf{y}}_i. \bigwedge R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i) \vee \bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) = \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta_1\right)
\end{aligned}$$

If there exists a valuation $\nu \in \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge \Gamma\right) \setminus \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta\right)$, then it is also the case that $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma_1) \setminus \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta_1\right)$. Therefore, the counterexample for the antecedent is $\nu_1 = \nu$ and $[\![\mathrm{FV}(\Gamma_1)]\!]_{\nu_1} \leq^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma)]\!]_{\nu}$ holds trivially.

(RD) Then the sequent $\Gamma \vdash \Delta \equiv \phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n), p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\exists \overline{\mathbf{y}}_j.\psi_j(\overline{\mathbf{x}}, \overline{\mathbf{y}}_j) \wedge \mathcal{Q}_j(\overline{\mathbf{y}}_j)\}_{j=1}^{k}$ has only one antecedent $\Gamma_1 \vdash \Delta_1 \equiv p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\mathcal{Q}_j\theta \mid \theta \in S_j\}_{j=1}^{i}$. By the side condition of RD, $\phi \models^{\mathcal{I}} \bigwedge_{j=1}^{i} \exists \overline{\mathbf{y}}_j.\psi_j$. Also, by Definition 2.4.5, we have $\mu\mathcal{S}^{\mathcal{I}}(\phi) \subseteq \mu\mathcal{S}^{\mathcal{I}}(\psi_j\theta)$ for each $\theta \in \mathrm{VIS}(\phi, \psi_j)$ and $j \in [i]$. In this case, the least solution of Δ is

$$\begin{aligned}
\mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta\right) &= \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee_{j=1}^{k} \exists \overline{\mathbf{y}}_j.\psi_j \wedge \mathcal{Q}_j\right) = \bigcup_{j=1}^{k} \mu\mathcal{S}^{\mathcal{I}}\left(\exists \overline{\mathbf{y}}_j.\psi_j \wedge \mathcal{Q}_j\right) \\
&\supseteq \bigcup_{j=1}^{i} \mu\mathcal{S}^{\mathcal{I}}\left(\exists \overline{\mathbf{y}}_j.\psi_j \wedge \mathcal{Q}_j\right) \supseteq \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}\left((\psi_j \wedge \mathcal{Q}_j)\theta\right) \\
&= \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}\left(\psi_j\theta \wedge \mathcal{Q}_j\theta\right) = \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \left(\mu\mathcal{S}^{\mathcal{I}}\left(\psi_j\theta\right) \cap \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)\right) \\
&= \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}\left(\psi_j\theta\right) \cap \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)
\end{aligned}$$

Note that also $\mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) = \mu\mathcal{S}^{\mathcal{I}}\left(\phi\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(p_1(\overline{\mathbf{x}}_1) \wedge \ldots \wedge p_n(\overline{\mathbf{x}}_n)\right)$. If there exists a valuation $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta\right)$, then

$$
\begin{aligned}
\nu \in \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge \Gamma\right) \setminus & \left(\bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}\left(\psi_j\theta\right) \cap \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)\right) \\
= \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge \Gamma\right) \setminus & \left(\bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}\left(\psi_j\theta\right)\right) \cup \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \left(\bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)\right) \\
= \emptyset \cup \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge \Gamma\right) \setminus & \left(\bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)\right) \\
\subseteq \mu\mathcal{S}^{\mathcal{I}}\left(p_1(\overline{\mathbf{x}}_1) \wedge \ldots \wedge p_n(\overline{\mathbf{x}}_n)\right) \setminus & \left(\bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi,\psi_j)} \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)\right) \\
\subseteq \mu\mathcal{S}^{\mathcal{I}}\left(p_1(\overline{\mathbf{x}}_1) \wedge \ldots \wedge p_n(\overline{\mathbf{x}}_n)\right) \setminus & \left(\bigcup_{j=1}^{i} \bigcup_{\theta \in S_j} \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)\right) \\
= \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge \Gamma_1\right) \setminus & \, \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta_1\right)
\end{aligned}
$$

Therefore, the counterexample for the antecedent is $\nu_1 = \nu$. Because ϕ is introduced to the left-hand side by left unfolding and $\mathcal{S}$ is ranked, we have $[\![\mathsf{set}(\overline{\mathbf{x}}_1) \cup \ldots \cup \mathsf{set}(\overline{\mathbf{x}}_n)]\!]_{\nu} <^{\ddagger}_{\mathcal{I}} [\![\mathsf{set}(\overline{\mathbf{x}})]\!]_{\nu}$. Then $[\![\mathrm{FV}(\Gamma_1)]\!]_{\nu_1} <^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma)]\!]_{\nu}$.

($\wedge$R) Let $\Delta = \{p(\overline{\mathbf{x}}) \wedge q(\overline{\mathbf{x}}) \wedge \mathcal{Q}\} \cup \Delta'$. Then the antecedents of $\Gamma \vdash \Delta$ are $\Gamma_1 \vdash \Delta_1 \equiv \Gamma \vdash p(\overline{\mathbf{x}}) \wedge \mathcal{Q}, \Delta'$ and $\Gamma_2 \vdash \Delta_2 \equiv \Gamma \vdash q(\overline{\mathbf{x}}) \wedge \mathcal{Q}, \Delta'$. In this case, the least solution of Δ is

$$
\begin{aligned}
\mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta) &= \mu\mathcal{S}^{\mathcal{I}}\left(p(\overline{\mathbf{x}}) \wedge q(\overline{\mathbf{x}}) \wedge \mathcal{Q} \vee \bigvee \Delta'\right) = \mu\mathcal{S}^{\mathcal{I}}\left(p(\overline{\mathbf{x}}) \wedge \mathcal{Q} \wedge q(\overline{\mathbf{x}}) \wedge \mathcal{Q} \vee \bigvee \Delta'\right) \\
&= \mu\mathcal{S}^{\mathcal{I}}\left(p(\overline{\mathbf{x}}) \wedge \mathcal{Q}\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(q(\overline{\mathbf{x}}) \wedge \mathcal{Q}\right) \cup \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta'\right) \\
&= \left(\mu\mathcal{S}^{\mathcal{I}}(p(\overline{\mathbf{x}}) \wedge \mathcal{Q}) \cup \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta'\right)\right) \cap \left(\mu\mathcal{S}^{\mathcal{I}}(q(\overline{\mathbf{x}}) \wedge \mathcal{Q}) \cup \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta'\right)\right) \\
&= \mu\mathcal{S}^{\mathcal{I}}\left(p(\overline{\mathbf{x}}) \wedge \mathcal{Q} \vee \bigvee \Delta'\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(q(\overline{\mathbf{x}}) \wedge \mathcal{Q} \vee \bigvee \Delta'\right) \\
&= \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta_1\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee \Delta_2\right)
\end{aligned}
$$

If there exists a valuation $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$, then also $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus (\mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta_1) \cap \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta_2)) = (\mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta_1)) \cup (\mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta_2))$. Therefore, $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma_i) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta_i)$ for some $i \in [2]$ and the counterexample for $\Gamma_i \vdash \Delta_i$ is $\nu_i = \nu$. Because $\Gamma = \Gamma_1 = \Gamma_2$, we have $[\![\mathrm{FV}(\Gamma_i)]\!]_{\nu_i} \leq^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma)]\!]_{\nu}$.

(SP) Then $\Gamma \vdash \Delta \equiv p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\bigwedge_{i=1}^{n} q_i^j(\overline{\mathbf{x}}_i)\}_{j=1}^{k}$. For each $\bar{\imath} \in [n]^{n^k}$, the antecedents of $\Gamma \vdash \Delta$ are $\Gamma_j^{\bar{\imath}} \vdash \Delta_j^{\bar{\imath}} \equiv p_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \vdash \{q_{\bar{\imath}_j}^{\ell}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}, j \in [n^k]$.

If there exists a valuation $\nu \in \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge_{i=1}^{n} p_i(\overline{\mathbf{x}}_i)\right) \setminus \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee_{j=1}^{k} \bigwedge_{i=1}^{n} q_i^j(\overline{\mathbf{x}}_i)\right)$, then by the proof of Lemma 2.5.1, there exist $j \in [n^k]$ and counterexamples $\nu_1, \ldots, \nu_n$ such that $\nu_i \in \mu\mathcal{S}^{\mathcal{I}}(p_i(\overline{\mathbf{x}}_i)) \setminus \bigcup\{\mu\mathcal{S}^{\mathcal{I}}(q_i^{\ell}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell(\overline{\mathbf{x}}_i))) = i\}$ and $\nu(\overline{\mathbf{x}}_i) = \nu_i(\overline{\mathbf{x}}_i)$ for all $i \in [n]$.

In other words, for all tuples $\bar{\imath} \in [n]^{n^k}$ we have $\nu_{\bar{\imath}_j} \in \mu\mathcal{S}^{\mathcal{I}}(p_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j})) \setminus \bigcup\{\mu\mathcal{S}^{\mathcal{I}}(q^{\ell}_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j})) \mid \ell \in [k], f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\} = \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma^{\bar{\imath}}_j) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta^{\bar{\imath}}_j)$. Therefore, given such $j \in [n^k]$, the counterexample for each antecedent $\Gamma^{\bar{\imath}}_j \vdash \Delta^{\bar{\imath}}_j$ is $\nu_{\bar{\imath}_j}$. Since $\mathrm{FV}(\Gamma^{\bar{\imath}}_j) = \mathsf{set}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \subseteq \mathrm{FV}(\Gamma)$ and $\nu(\overline{\mathbf{x}}_{\bar{\imath}_j}) = \nu_i(\overline{\mathbf{x}}_{\bar{\imath}_j})$, we have $[\![\mathrm{FV}(\Gamma^{\bar{\imath}}_j)]\!]_{\nu_i} \leq^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma)]\!]_{\nu}$, for each $\bar{\imath} \in [n]^{n^k}$. □

This result allows us to define a reachability relation between counterexamples and write $\nu \gg \nu'$ when, given any instance of an inference rule in $\mathcal{R}_{\mathsf{Ind}} \setminus \mathrm{ID}$, ν is a counterexample of the consequent and ν' is a counterexample of one of its antecedents obtained from ν, as shown in the proof of Lemma 2.5.2.

Definition 2.5.1 (Counterexample path in FOL)**.** A path $\pi = v_1, v_2, \ldots, v_k$ in a proof $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ built with $\mathcal{R}_{\mathsf{Ind}}$ is a *counterexample path* if there exists a sequence of valuations $\nu_1, \nu_2, \ldots, \nu_k$ such that, for all $i \in [k]$: (i) $\nu_i \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma_i) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta_i)$, where $\mathsf{Seq}(v_i) = \Gamma_i \vdash \Delta_i$, and (ii) $\nu_i \gg \nu_{i+1}$ if $i < k$.

The following lemma shows that, if the given FOL inductive system is ranked, any direct counterexample path in a proof causes a strict decrease in the multiset images of the counterexamples for the pivot and consequent delimiting the path.

Lemma 2.5.3. Given an FOL inductive system $\mathcal{S}$ that is ranked in the interpretation $\mathcal{I}$, let $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ be a proof built with $\mathcal{R}_{\mathsf{Ind}}$ and $\pi = v_1, \ldots, v_k$ be a direct counterexample path in D for a backlink (v_k, v_1), with valuations $\nu_1 \gg \ldots \gg \nu_k$. Then $[\![\mathrm{FV}(\Gamma_1)]\!]_{\nu_1} >^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma_k)]\!]_{\nu_k}$, where $\mathsf{Seq}(v_i) = \Gamma_i \vdash \Delta_i$, for all $i \in [k]$.

Proof. Since π is a direct path, it follows that ID does not occur in $\Lambda(v_1, \ldots, v_{k-1})$. As shown in the proof of Lemma 2.5.2, which ensures the local soundness of $\mathcal{R}_{\mathsf{Ind}} \setminus \{\mathrm{ID}\}$, we obtain a sequence of valuations $\nu_1 \gg \ldots \gg \nu_k$ such that $[\![\mathrm{FV}(\Gamma_1)]\!]_{\nu_1} \geq^{\ddagger}_{\mathcal{I}} \ldots \geq^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma_k)]\!]_{\nu_k}$.

We know that $\mathsf{Seq}(v_1) = \Gamma_1 \vdash \Delta_1$, $\mathsf{Seq}(v_k) = \Gamma_k \vdash \Delta_k = \Gamma_1\theta \vdash \Delta'_1\theta$ with $\Delta_1 \subseteq \Delta'_1$ and LU occurs in $\Lambda(\pi)$, as required by the side condition of the ID instance applied at v_k. We show that RD is also required to occur in $\Lambda(\pi)$ by the following analysis on the form of Γ_1:

(i) If $\Gamma_1 = \{p(\overline{\mathbf{x}})\}$, then LU is the only inference rule applicable on $\Gamma_1 \vdash \Delta_1$ that changes Γ_1. This required application introduces a constraint on the left-hand side. In order to reach any sequent that has a constraint-free left-hand side, an application of RD is also required. To specifically reach $\Gamma_1\theta \vdash \Delta'_1\theta$ with $\Gamma_1\theta = \{p(\overline{\mathbf{x}}\theta)\}$, $\Lambda(\pi)$ may contain additional occurrences of LU and RD, but at least one RD is required;

(ii) If $\Gamma_1 = \{p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n)\}$ with $n > 1$, then LU is immediately applicable, but it would introduce a constraint that cannot be reduced (this requires all arguments of predicate atoms to be subgoals in the constraint) thus making it impossible to reach any sequent with only predicate atoms on the left-hand side. Therefore, SP should be applied before the required LU and, later on, RD is needed to remove the constraint introduced by LU. In order to specifically reach $\Gamma_1\theta \vdash \Delta'_1\theta$ with $\Gamma_1\theta = \{p_1(\overline{\mathbf{x}}_1\theta), \ldots, p_n(\overline{\mathbf{x}}_n\theta)\}$, $\Lambda(\pi)$ may contain additional occurrences of SP, LU and RD, but at least one RD is required;

(iii) If $\Gamma_1 = \{\phi, p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n)\}$ with $n \geq 1$, then, just as in case (ii), applying LU before another inference rule that modifies the left-hand side introduces a second constraint impossible to reduce (this requires a single constraint on the left-hand side), thus preventing the reaching of any sequent with only one constraint on the left-hand side. Therefore, RD is necessary to remove ϕ before applying LU. In order to specifically

reach $\Gamma_1\theta \vdash \Delta_1'\theta$ with $\Gamma_1\theta = \{\phi\theta, p_1(\overline{\mathbf{x}}_1\theta), \ldots, p_n(\overline{\mathbf{x}}_n\theta)\}$, $\Lambda(\pi)$ may contain additional occurrences of RD, SP and LU, but at least one RD is required;

(iv) If $\Gamma_1 = \{\phi_1, \ldots, \phi_m\}$ with $m \geq 1$, then LU is not applicable on any path starting with $\Gamma_1 \vdash \Delta_1$ because we cannot create a derivation with $\mathcal{R}_{\mathsf{Ind}}$ that introduces predicate atoms on the left-hand side;

(v) If $\Gamma_1 = \{\phi_1, \ldots, \phi_m, p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n)\}$ with $m, n > 1$, then LU is the only inference rule applicable on $\Gamma_1 \vdash \Delta_1$, and on all of the sequents derived from it, that modifies the left-hand side. Therefore, it is impossible to reach any sequent having m constraints on the left-hand side, which includes $\Gamma_1\theta \vdash \Delta_1'\theta$ with $\Gamma_1\theta = \{\phi_1\theta, \ldots, \phi_m\theta, p_1(\overline{\mathbf{x}}_1\theta), \ldots, p_n(\overline{\mathbf{x}}_n\theta)\}$, because LU would introduce extra constraints in any derived sequent that cannot be removed because RD is not applicable.

It follows from cases (i), (ii), (iii), i.e. the ones allowing the existence of the path π with the necessary LU occurrence in $\Lambda(\pi)$, that RD is required to occur in $\Lambda(\pi)$. Then $[\![\mathrm{FV}(\Gamma_i)]\!]_{\nu_i} >^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma_{i+1})]\!]_{\nu_{i+1}}$ for some $i \in [k-1]$, which leads to $[\![\mathrm{FV}(\Gamma_1)]\!]_{\nu_1} >^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma_k)]\!]_{\nu_k}$. □

Through the next lemma, we extend the reachability relation $\succcurlyeq$ to backlinks and show that any infinite trace in a proof leads to an infinite strictly decreasing sequence of multisets associated with counterexamples, contradicting the well-foundedness of the wfqo which makes the inductive system be ranked (Defintion 2.4.3). Then there cannot exist a counterexample for any sequent labelling the root of a proof built with $\mathcal{R}_{\mathsf{Ind}}$ and the associated entailment must hold.

Theorem 2.5.4. Given an FOL inductive system $\mathcal{S}$ that is ranked in the interpretation $\mathcal{I}$, if a sequent $\Gamma \vdash \Delta$ has a proof $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ built with $\mathcal{R}_{\mathsf{Ind}}$ and $\mathsf{Seq}(v_0) \equiv \Gamma \vdash \Delta$, then the entailment $\bigwedge \Gamma \models^{\mathcal{I}}_{\mathcal{S}} \bigvee \Delta$ holds.

Proof. Suppose, by contradiction, that $\bigwedge \Gamma \models^{\mathcal{I}}_{\mathcal{S}} \bigvee \Delta$ does not hold, i.e. there exists a valuation $\nu_0 \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$. Since v_0 is the root of the proof, it is also the consequent of an instance of $\mathsf{Rule}(v_0)$ and, by Lemma 2.5.2, an antecedent of this inference rule has a counterexample ν_1, such that $\nu_0 \succcurlyeq \nu_1$. Applying this argument iteratively, we build a path from v_0 to a leaf $v_k \in V$ and a sequence of valuations $\nu_0 \succcurlyeq \nu_1 \succcurlyeq \ldots \succcurlyeq \nu_k$.

Since AX inference rules, by their side condition, cannot be applied on sequents that accept counterexamples, it must be the case that $\mathsf{Rule}(v_k) \neq \mathrm{AX}$. Since v_k is a leaf, then $\mathsf{Rule}(v_k) = \mathrm{ID}$ and let $v_{k+1} = \mathsf{Piv}(v_k)$. Then $\mathsf{Seq}(v_k) = \Gamma_k \vdash \Delta_k$ and $\mathsf{Seq}(v_{k+1}) = \Gamma_{k+1} \vdash \Delta_{k+1}$ such that $\Gamma_k = \Gamma_{k+1}\theta$, $\Delta_k = \Delta_{k+1}'\theta$ and $\Delta_{k+1} \subseteq \Delta_{k+1}'$, for some injective substitution $\theta : \mathrm{FV}(\Gamma_{k+1} \cup \Delta_{k+1}) \to \mathrm{FV}(\Gamma_k \cup \Delta_k)$. We can assume w.l.o.g. that θ is surjective, by defining $\theta(x) = x$ for each $x \in \mathrm{FV}(\Gamma_k \cup \Delta_k) \setminus \theta(\mathrm{FV}(\Gamma_{k+1} \cup \Delta_{k+1}))$. Since θ is also injective, by the side condition of ID, its inverse exists and $\nu_k\theta^{-1}$ is a counterexample for $\Gamma_{k+1} \vdash \Delta_{k+1}$. Therefore, we can extend the relation $\succcurlyeq$ with the pair $(\nu_k, \nu_k\theta^{-1})$.

This argument can be continued ad infinitum and we obtain an infinite trace $\tau = v_0, v_1, \ldots$ in D together with an infinite sequence of valuations $\nu_0 \succcurlyeq \nu_1 \succcurlyeq \ldots$. If $\mathsf{Seq}(v_i) = \Gamma_i \vdash \Delta_i$, for each $i \geq 0$, by Lemma 2.5.2, we have $[\![\mathrm{FV}(\Gamma_i)]\!]_{\nu_i} \geq^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma_{i+1})]\!]_{\nu_{i+1}}$, for all $i \geq 0$. By Proposition 2.2.1, τ contains infinitely many direct paths $\pi_j = v_{k_j}, \ldots, v_{\ell_j}$, where $\{k_j\}_{j\geq 0}$ and $\{\ell_j\}_{j\geq 0}$ are infinite strictly increasing sequences of integers such that $k_j < \ell_j \leq k_{j+1} < \ell_{j+1}$, for all $j \geq 0$. By Lemma 2.5.3, we obtain that $[\![\mathrm{FV}(\Gamma_{k_j})]\!]_{\nu_{k_j}} >^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma_{\ell_j})]\!]_{\nu_{\ell_j}}$, for all $j \geq 0$. Since $[\![\mathrm{FV}(\Gamma_{\ell_j})]\!]_{\nu_{\ell_j}} \geq^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma_{k_{j+1}})]\!]_{\nu_{k_{j+1}}}$, for all $j \geq 0$, we obtain a strictly decreasing

sequence $[\![\mathrm{FV}(\Gamma_{k_0})]\!]_{\nu_{k_0}} >^{\ddagger}_{\mathcal{I}} [\![\mathrm{FV}(\Gamma_{k_1})]\!]_{\nu_{k_1}} >^{\ddagger}_{\mathcal{I}} \ldots$, which contradicts that $>^{\ddagger}_{\mathcal{I}}$ is a wfqo, by Proposition 2.4.5. We can conclude that our assumption was false, thus the entailment $\bigwedge \Gamma \models^{\mathcal{I}}_{\mathcal{S}} \bigvee \Delta$ holds. □

Completeness

We will show that $\mathcal{R}_{\mathsf{Ind}}$ is complete for entailments between predicates defined by FOL inductive systems that are ranked, non-filtering, non-overlapping and have the fvi property. To facilitate this proof, we first show some properties of the derivations built using the inference rules in $\mathcal{R}_{\mathsf{Ind}}$.

We introduce maximal, irreducible and structured derivations, and use $\mathfrak{D}(\Gamma \vdash \Delta)$ for the set of all derivations built with $\mathcal{R}_{\mathsf{Ind}}$ whose roots are labelled with $\Gamma \vdash \Delta$ and that are, at the same time, maximal, irreducible and structured.

Definition 2.5.2 (Maximal derivation)**.** A derivation using the $\mathcal{R}_{\mathsf{Ind}}$ proof system is *maximal* if it cannot be extended by an application of any inference rule from $\mathcal{R}_{\mathsf{Ind}}$.

Definition 2.5.3 (Irreducible derivation)**.** A derivation D using the $\mathcal{R}_{\mathsf{Ind}}$ proof system is *irreducible* if there exists no path in D on which ID is applicable. Otherwise, D is *reducible*.

Note that the proof-search semi-algorithm 1 from Section 2.2 only produces irreducible derivations, as it will eagerly try to apply AX and ID before the other inference rules.

Definition 2.5.4 (Structured derivation)**.** A derivation D is *structured* if, on every path of D, there exists an application of RD between any two consecutive applications of LU. Otherwise, D is *unstructured*.

We are interested in structured derivations because, intuitively, unstructured derivations constitute poor candidates for proofs. It will always be possible, for instance, to create a derivation by repeatedly applying LU and no other inference rule. However, this kind of derivation will only grow the left-hand side without making significant progress towards $\top$ or a counterexample. Note how any subtree of a structured derivation is also structured.

Lemma 2.5.5. If the FOL inductive system $\mathcal{S}$ has the fvi property, then the following properties of derivations built with $\mathcal{R}_{\mathsf{Ind}}$ hold:

(1) Any irreducible and structured derivation is finite;

(2) For any sequent $\Gamma \vdash \Delta$, the set $\mathfrak{D}(\Gamma \vdash \Delta)$ is finite.

Proof. Given an FOL inductive system $\mathcal{S}$, we introduce the following constants relative to $\mathcal{S}$:

- $\mathsf{r}^{\#} = \|\mathcal{S}\|$ is the number of predicate rules in $\mathcal{S}$;
- $\mathsf{p}^{\#} = \|\mathcal{S}^{\mathsf{p}}\|$ is the number of predicates in the inductive system $\mathcal{S}$;
- $\mathsf{s}^{\#}$ is the maximum number of subgoals occurring in each predicate rule of $\mathcal{S}$.

Since $\mathsf{p}^{\#}$ is finite for any inductive system $\mathcal{S}$, there is also a finite number $\mathsf{b}^{\#}$ of basic sequents that can be constructed using the predicates in $\mathcal{S}^{\mathsf{p}}$. As there are $\mathsf{p}^{\#}$ predicates that can occur on the left-hand side and $2^{\mathsf{p}^{\#}} - 1$ possible non-empty subsets of predicates that can occur on the right-hand side, it follows that $\mathsf{b}^{\#} = \mathsf{p}^{\#} * (2^{\mathsf{p}^{\#}} - 1)$.

Let D be a structured derivation starting from the basic sequent $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ and π be a path in D from the root to another basic sequent $r(\overline{\mathbf{y}}) \vdash s_1(\overline{\mathbf{y}}), \ldots, s_m(\overline{\mathbf{y}})$, without containing any other basic sequents. Only LU, RU, $\wedge$R, RD and SP can be applied along π, otherwise AX and ID would not allow us to reach the second basic sequent. The left-hand side of the sequents along π allows for

- at most one application of LU, on $p(\overline{\mathbf{x}})$;
- at most one application of RD, after LU, on the predicate rule that replaces $p(\overline{\mathbf{x}})$;
- at most one application of SP, after RD and $\wedge$R, if we are left with multiple predicate atoms on the left-hand side.

The right-hand side of the sequents along π allows for

- n applications of RU, where n is at most $\mathsf{p}^{\#}$;
- $\mathsf{r}^{\#} * (\mathsf{s}^{\#} - 1)$ applications of $\wedge$R, because there can be at most $\mathsf{s}^{\#} - 1$ for every sequent resulting from RU and reduced by RD, and there can be at most $\mathsf{r}^{\#}$ predicate rules that can be used to apply RU.

Thus, any path between two consecutive basic sequents is of length at most $\mathsf{b}^{\#}_{\pi} = 3 + \mathsf{p}^{\#} + \mathsf{r}^{\#} * (\mathsf{s}^{\#} - 1)$, which is a constant determined by the inductive system $\mathcal{S}$.

Case (1) Let D be an irreducible and structured derivation. Suppose, by contradiction, that D is not finite. By König's Lemma, D must have an infinite path. Let π be an arbitrary infinite path in D. Let ρ be any subsequence of π on which no LU rule has been applied. The maximum possible length of ρ is reached when the sequence starts with an antedecedent of LU applied on a basic sequent and it extends until the consequent of the next possible application of LU, while encountering the next basic sequent and covering the applications of all possible inference rules before the second LU. The only inference rule applicable on a basic sequent before LU is RU and it can occur a maximum of $\mathsf{p}^{\#}$ times. Thus, ρ has a maximum length of $\mathsf{b}^{\#}_{\pi} - 1 + \mathsf{p}^{\#}$ and is finite.

Then, for π to be infinite, LU must be applied infinitely often along π. Since D is structured, RD must also be applied infinitely often. The antecedent of each application of RD contains no constraints and each corresponding consequent is of the form $\phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n), p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \Delta$, where the left hand side of such sequents must have been produced by an application of LU on a sequent of the form $p(\overline{\mathbf{x}}) \vdash \Delta$. Such sequents can only be the antecedents of RD, $\wedge$R or SP, the first two having a single predicate atom on the left-hand side of their consequent. If $p(\overline{\mathbf{x}}) \vdash \Delta$ is the antecedent of RD or $\wedge$R, then Δ consists of singleton predicate atoms and the sequent is basic, because $\mathcal{S}$ has the fvi property and $\wedge$R is used eagerly after an application of RD to rule out conjunctions of predicate atoms with the same argument list. This is also the case for SP due to the form of its antecedents and, moreover, the consequent of SP must have been obtained from an application of RD, optionally followed by $\wedge$R.

Therefore, between every two consecutive applications of RD there exists a basic sequent and, for π to be infinite, basic sequents must occur infinitely often along it. Since the number of basic sequents is bounded by $\mathsf{b}^{\#}$, some basic sequent must occur at least twice along π, which makes ID applicable and contradicts the assumption that D is irreducible. Hence, π must be finite, and, since it was chosen arbitrarily, D cannot have an infinite path and is a finite derivation.

Case (2) Suppose, by contradiction, that $\mathfrak{D}(\Gamma \vdash \Delta)$ is infinite. We know from (1) that each derivation in $\mathfrak{D}(\Gamma \vdash \Delta)$ is finite. Let $D_1, D_2, \ldots$ be an infinite sequence of distinct, maximal derivations from $\mathfrak{D}(\Gamma \vdash \Delta)$ obtained by applying more than one inference rule. Such a sequence exists because there is only a finite number of derivations obtained by applying only one inference rule. Since there can be at most $\mathsf{b}^{\#}$ distinct basic sequents and any path between two consecutive basic sequents is of finite length at most $\mathsf{b}^{\#}_{\pi}$, then there must exist a derivation D_i in the infinite sequence chosen above that contains a path in which the same basic sequent appears at least twice. But then this means that D_i is reducible and, thus, that $D_i \notin \mathfrak{D}(\Gamma \vdash \Delta)$, which contradicts our assumption. □

Definition 2.5.5 (Tree-shaped set)**.** A set $F = \{\phi_1, \ldots, \phi_m, p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n)\}$, where $\phi_1, \ldots, \phi_m$ are constraints and $p_1(\overline{\mathbf{x}}_1), \ldots p_n(\overline{\mathbf{x}}_n)$ are predicate atoms, is *tree-shaped* if there exist trees $t_1, \ldots, t_k$ such that:

1. Each node labelled with a constraint $\phi_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_h)$ in some tree t_ℓ, $\ell \in [k]$ has exactly h children and, for all $j \in [h]$, the j-th child is labelled either with (i) a constraint whose goal variables are $\overline{\mathbf{y}}_j$, or (ii) a predicate atom $q(\overline{\mathbf{y}}_j)$, $q \in \{p_1, \ldots, p_n\}$.
2. A predicate atom $p_i(\overline{\mathbf{x}}_i), i \in [n]$ may occur only on the frontier of a tree t_j, $j \in [k]$.

If $k = 1$, then F is called *singly-tree shaped*.

Because tree-shaped sets can be uniquely represented as trees labelled with formulae, we use sets of trees and sets of formulae interchangeably. We write $\Gamma \vdash \Delta \rightsquigarrow \Gamma' \vdash \Delta'$ whenever $\Gamma' \vdash \Delta'$ labels a node in a derivation from $\mathfrak{D}(\Gamma \vdash \Delta)$. The following lemma shows an invariant on the shape of the sequents in any derivation that starts with a basic sequent.

Lemma 2.5.6. Given an FOL inductive system $\mathcal{S}$ and predicates $p, q_1, \ldots, q_n \in \mathcal{S}^{\mathsf{P}}$, in every sequent $\Gamma \vdash \Delta$ such that $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow \Gamma \vdash \Delta$, Γ is a tree-shaped set and Δ consists of finite conjunctions over tree-shaped sets, with all subgoal variables existentially quantified.

Proof. We prove this lemma by induction on the length of the path $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) = \Gamma_1 \vdash \Delta_1, \ldots, \Gamma_N \vdash \Delta_N = \Gamma \vdash \Delta$ from the derivation in which $\Gamma \vdash \Delta$ occurs. Let $T(N)$ mean "Γ_N is a tree-shaped set and Δ_N consists of finite conjunctions over tree-shaped sets, with all subgoal variables existentially quantified".

Base case. We consider the case where $N = 1$. Then $\Gamma_1 = \{p(\overline{\mathbf{x}})\}$ is trivially tree-shaped and $\Delta_1 = \{q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})\}$ trivially consists of singleton conjunctions over the tree-shaped sets $\{q_1(\overline{\mathbf{x}})\}, \ldots, \{q_n(\overline{\mathbf{x}})\}$, therefore $T(1)$ holds.

Inductive case. Assuming that $T(N-1)$ holds, we prove that $T(N)$ also holds by doing a case split on the type of the last inference rule applied on the path.

– (LU) In this case, $\Delta_N = \Delta_{N-1}$. Based the induction hypothesis, Γ_{N-1} is tree-shaped and there exists a tree t associated with Γ_{N-1} such that $t(\alpha) = r(\overline{\mathbf{y}})$, for some frontier position $\alpha \in \mathrm{fr}(t)$ and predicate atom $r(\overline{\mathbf{y}})$. Then there exists a predicate rule $\langle r(\overline{\mathbf{y}}), R(\overline{\mathbf{y}})\rangle \in \mathcal{S}$ such that $\Gamma_N = R(\overline{\mathbf{y}}, \overline{\mathbf{y}}') \cup \Gamma_{N-1} \setminus \{r(\overline{\mathbf{y}})\}$, where $R(\overline{\mathbf{y}}, \overline{\mathbf{y}}') = \phi(\overline{\mathbf{y}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m), s_1(\overline{\mathbf{y}}_1), \ldots, s_m(\overline{\mathbf{y}}_m)$, and $t_{[\alpha]} \circ \tau_m(R(\overline{\mathbf{y}}, \overline{\mathbf{y}}'))$ replaces t in the set of trees that represents Γ_N. Therefore, $T(N)$ holds.

– (RU) In this case, $\Gamma_N = \Gamma_{N-1}$. Based the induction hypothesis, there exists $r(\overline{\mathbf{y}}) \in \Delta_{N-1}$ and a tree consisting of a single node labelled with $r(\overline{\mathbf{y}})$. This tree is replaced in Δ_N by trees $t_1, \ldots, t_m$ corresponding to conjunctions over the tree-shaped sets $R_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1), \ldots, R_m(\overline{\mathbf{y}}, \overline{\mathbf{z}}_m)$ with existentially quantified subgoal variables, where $r(\overline{\mathbf{y}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{y}}) \mid \ldots \mid R_m(\overline{\mathbf{y}})$. Thus, $T(N)$ holds.

– (RD) Any antecedent of an RD application is of the form

$$\Gamma_N \vdash \Delta_N \equiv r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m) \vdash \{s_1^j(\overline{\mathbf{z}}_1^j) \wedge \ldots \wedge s_{\ell_j}^j(\overline{\mathbf{z}}_{\ell_j}^j)\}_{j=1}^k$$

Therefore, $T(N)$ trivially holds.

– ($\wedge$R) In this case, $\Gamma_N = \Gamma_{N-1}$. Based on the inductive hypothesis, Δ_N contains all the elements of Δ_{N-1}, except for one whose corresponding tree has lost a leaf consisting of a predicate atom. Therefore, $T(N)$ holds.

– (SP) Any antecedent of an SP application is of the form

$$\Gamma_N \vdash \Delta_N \equiv r(\overline{\mathbf{y}}) \vdash s_1(\overline{\mathbf{y}}), \ldots, s_m(\overline{\mathbf{y}})$$

Therefore, $T(N)$ trivially holds.

Since $T(1)$ holds and $T(N-1)$ implies $T(N)$ for any $N > 1$, we can conclude that $T(N)$ holds for any $N \geq 1$. □

The following lemma is crucial for establishing the completeness of $\mathcal{R}_{\mathsf{Ind}}$. It provides an invariant on the set of derivations for a sequent that denotes an invalid entailment and proves that they indicate the existence of a counterexample.

Lemma 2.5.7. Given an interpretation $\mathcal{I}$, a non-filtering and non-overlapping FOL inductive system $\mathcal{S}$ with the fvi property, the predicates $p, q_1, \ldots, q_n \in \mathcal{S}^{\mathrm{p}}$, and a sequent $\Gamma \vdash \Delta$ such that $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow \Gamma \vdash \Delta$, if every derivation $D \in \mathfrak{D}(\Gamma \vdash \Delta)$ contains a leaf $\Gamma' \vdash \emptyset$ then there exists a valuation $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$.

Proof. Let $\mathfrak{D}^*(\Gamma \vdash \Delta) = \{d \mid d \in \mathfrak{D}(\Gamma \vdash \Delta) \text{ or } \exists d' \in \mathfrak{D}^*(\Gamma \vdash \Delta) \,.\, d \sqsubset d'\}$. Because, by Lemma 2.5.5, any derivation in $\mathfrak{D}(\Gamma \vdash \Delta)$ is finite and $\mathfrak{D}(\Gamma \vdash \Delta)$ itself is finite, then $\mathfrak{D}^*(\Gamma \vdash \Delta)$ is also finite. Since $(\mathfrak{D}^*(\Gamma \vdash \Delta), \sqsubseteq)$ is a wqo, by Lemma 2.4.1 $(\mathcal{P}_{fin}(\mathfrak{D}^*(\Gamma \vdash \Delta)), \sqsubseteq^{\forall\exists})$ is also a wqo and we can prove this lemma by induction on it.

Let $\mathfrak{D}(\Gamma \vdash \Delta) = \bigcup_{IR \in \mathcal{R}_{\mathsf{Ind}}} \mathfrak{D}_{IR}(\Gamma \vdash \Delta)$, where $\mathfrak{D}_{IR}(\Gamma \vdash \Delta)$ denotes the subset of $\mathfrak{D}(\Gamma \vdash \Delta)$ consisting of derivations starting with an instance of the inference rule IR. We do not consider the cases $IR \in \{\mathrm{AX}, \mathrm{ID}\}$ because these inference rules do not lead to a leaf of the required form. Given $D \in \mathfrak{D}_{IR}(\Gamma \vdash \Delta)$ with $IR \in \mathcal{R}_{\mathsf{Ind}} \setminus \{\mathrm{AX}, \mathrm{ID}\}$, we assume that, for each antecedent $\Gamma' \vdash \Delta'$ of $\Gamma \vdash \Delta$, the lemma holds for any $D' \in \bigcup_{IR' \in \mathcal{R}_{\mathsf{Ind}}} \mathfrak{D}_{IR'}(\Gamma' \vdash \Delta')$ and show that it extends to D.

(LU) Let $r(\overline{\mathbf{y}}) \in \Gamma$ be the predicate atom chosen for replacement and $r(\overline{\mathbf{y}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{y}}) \mid \ldots \mid R_m(\overline{\mathbf{y}})$. Then $R_1(\overline{\mathbf{y}}, \overline{\mathbf{y}}_1), \Gamma \setminus r(\overline{\mathbf{y}}) \vdash \Delta, \ldots, R_m(\overline{\mathbf{y}}, \overline{\mathbf{y}}_m), \Gamma \setminus r(\overline{\mathbf{y}}) \vdash \Delta$ are the antecedents for the application of $IR = \mathrm{LU}$ on $\Gamma \vdash \Delta$, for some fresh tuples of variables $\overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m$.

The right-hand side of any antecedent is $\emptyset$ only if Δ was already $\emptyset$. Because $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow \Gamma \vdash \Delta$, $\mathcal{S}$ is non-filtering, every constraint in $\mathcal{S}^{\mathrm{c}}$ is satisfiable, and there are no predicates with empty least solutions, it follows that $\mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma)$ is not empty. Therefore, there exists a counterexample $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta) = \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma)$.

If $\Delta \neq \emptyset$, suppose that, for each $i \in [m]$, there exists $D_i \in \mathfrak{D}(R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i), \Gamma \setminus r(\overline{\mathbf{y}}) \vdash \Delta)$ not containing any leaf $\Gamma' \vdash \emptyset$. Then we can choose these derivations and create one for $\Gamma \vdash \Delta$ with the same property, which contradicts the hypothesis of the lemma. Therefore, for some $i \in [m]$ every derivation in $\mathfrak{D}(R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i), \Gamma \setminus r(\overline{\mathbf{y}}) \vdash \Delta)$ must contain a leaf $\Gamma' \vdash \emptyset$.

By the induction hypothesis, there exists $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge(R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i) \cup \Gamma \setminus \{r(\overline{\mathbf{y}})\})) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$. Then it is also the case that $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$, because

$$\begin{aligned}
\mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i) \cup \Gamma \setminus \{r(\overline{\mathbf{y}})\})\right) &= \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i)\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(\Gamma \setminus \{r(\overline{\mathbf{y}})\})\right) \\
&\subseteq \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(r(\overline{\mathbf{y}})\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(\Gamma \setminus \{r(\overline{\mathbf{y}})\})\right) \\
&\subseteq \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(r(\overline{\mathbf{y}}) \wedge \bigwedge(\Gamma \setminus \{r(\overline{\mathbf{y}})\})\right) \\
&\subseteq \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge(r(\overline{\mathbf{y}}) \cup \Gamma \setminus \{r(\overline{\mathbf{y}})\})\right) = \mu\mathcal{S}^{\mathcal{I}}\left(\bigwedge \Gamma\right)
\end{aligned}$$

(RU) Let $r(\overline{\mathbf{y}}) \in \Delta$ be the predicate atom chosen for replacement and $r(\overline{\mathbf{y}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{y}}) \mid \ldots \mid R_m(\overline{\mathbf{y}})$. Then $\Gamma \vdash \exists \overline{\mathbf{z}}_1. \bigwedge R_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1), \ldots, \exists \overline{\mathbf{z}}_m. \bigwedge R_m(\overline{\mathbf{y}}, \overline{\mathbf{z}}_m), \Delta \setminus r(\overline{\mathbf{y}})$ is the antecedent for the application of $IR = \text{RU}$ on $\Gamma \vdash \Delta$. Note that, because RU is applicable, $\Delta \neq \emptyset$. Also, the right-hand side of the antecedent cannot be $\emptyset$ because all predicates in $\mathcal{S}^{\text{p}}$ must be goals for at least one predicate rule of $\mathcal{S}$.

Suppose that there exists $D \in \mathfrak{D}(\Gamma \vdash \exists \overline{\mathbf{z}}_1. \bigwedge R_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1), \ldots, \exists \overline{\mathbf{z}}_m. \bigwedge R_m(\overline{\mathbf{y}}, \overline{\mathbf{z}}_m), \Delta \setminus r(\overline{\mathbf{y}}))$ not containing any leaf $\Gamma' \vdash \emptyset$. Then we can choose D and create a derivation for $\Gamma \vdash \Delta$ with the same property, which contradicts the hypothesis of the lemma. Therefore, it must be the case that every derivation in $\mathfrak{D}(\Gamma \vdash \exists \overline{\mathbf{z}}_1. \bigwedge R_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1), \ldots, \exists \overline{\mathbf{z}}_m. \bigwedge R_m(\overline{\mathbf{y}}, \overline{\mathbf{z}}_m), \Delta \setminus r(\overline{\mathbf{y}}))$ contains a leaf $\Gamma' \vdash \emptyset$.

By the induction hypothesis, there exists $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee_{i=1}^{m} \exists \overline{\mathbf{z}}_i. \bigwedge R_i \vee \bigvee(\Delta \setminus \{r(\overline{\mathbf{y}})\}))$. Then it is also the case that $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$, because $\mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta) = \mu\mathcal{S}^{\mathcal{I}}(\bigvee_{i=1}^{m} \exists \overline{\mathbf{z}}_i. \bigwedge R_i(\overline{\mathbf{y}}, \overline{\mathbf{z}}_i) \vee \bigvee(\Delta \setminus \{r(\overline{\mathbf{y}})\}))$, as previously shown in the (RU) case of the proof for Theorem 2.5.2.

(RD) Let $\Gamma = \{\phi(\overline{\mathbf{y}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m), r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m)\}$ and $\Delta = \{\exists \overline{\mathbf{z}}_1.\psi_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1) \wedge \mathcal{Q}_1(\overline{\mathbf{z}}_1), \ldots, \exists \overline{\mathbf{z}}_k.\psi_k(\overline{\mathbf{y}}, \overline{\mathbf{z}}_k) \wedge \mathcal{Q}_k(\overline{\mathbf{z}}_k)\}$, where $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ are conjunctions of predicate atoms. Also, let $S_j = \text{VIS}(\phi, \psi_j)$, for all $j \in [k]$. Each S_j is finite because $\mathcal{S}$ has the fvi property.

If $m = 0$, then Γ contains no predicate atoms. Since $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow \Gamma \vdash \Delta$, then, by Lemma 2.5.6, Γ is tree-shaped, meaning that $\Gamma = \{\phi(\overline{\mathbf{y}})\}$ and ϕ has no subgoal variables. We distinguish the following cases:

- If $k = 0$, then $\Delta = \emptyset$. Since $\phi \in \mathcal{S}^{\text{c}}$, it must be satisfiable. Thus any valuation ν such that $\mathcal{I}, \nu \models \phi$ contradicts the entailment $\phi(\overline{\mathbf{y}}) \models^{\mathcal{I}} \bot$. Such a valuation exists because all constraints in $\mathcal{S}^{\text{c}}$ are satisfiable.

- If $k > 0$, then $\phi(\overline{\mathbf{y}}) \not\models^{\mathcal{I}} \exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{y}}, \overline{\mathbf{z}}_j)$ for each $j \in [k]$, because $\mathcal{S}$ has the fvi property. As in the previous case, any model of ϕ contradicts the entailment and such a model exists because ϕ is satisfiable.

If $m > 0$, then $r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m) \vdash \{\mathcal{Q}_j\theta \mid \theta \in S_j\}_{j=1}^{i}$ is the antecedent for the application of $IR = \text{RD}$ on $\Gamma \vdash \Delta$, where, by the side condition, $\phi \models^{\mathcal{I}} \bigwedge_{j=1}^{i} \exists \overline{\mathbf{z}}_j.\psi_j$ and $\phi \not\models^{\mathcal{I}} \bigvee_{j=i+1}^{k} \exists \overline{\mathbf{z}}_j.\psi_j$ (with a possible reordering of Δ). If $k = 0$, because $\mathcal{S}$ is non-

filtering, $\overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m$ are distinct and there are no predicates in $\mathcal{S}^{\mathrm{p}}$ with empty least solutions, it follows that $\mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma)$ is not empty. Therefore there exists a counterexample $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta) = \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma)$. Similarly, if $i = 0$, we have that $\mu\mathcal{S}^{\mathcal{I}}(r_1(\overline{\mathbf{y}}_1) \wedge \ldots \wedge r_m(\overline{\mathbf{y}}_m))$ is not empty and it contains counterexamples for the antecedent. Given $\nu \in \mu\mathcal{S}^{\mathcal{I}}(r_1(\overline{\mathbf{y}}_1) \wedge \ldots \wedge r_m(\overline{\mathbf{y}}_m))$, because the system is non-filtering, there exists $\overline{\mathbf{v}} \in \mathsf{Val}^{\|\overline{\mathbf{y}}\|}$ such that $\mathcal{I}, \nu' \models \phi$ and $\nu' = \nu[\overline{\mathbf{y}} \leftarrow \overline{\mathbf{v}}]$. Because, by the side condition of RD and the fact that $\mathcal{S}$ is non-overlapping, we have that $\mathcal{I}, \nu' \not\models \exists \overline{\mathbf{z}}_j.\psi_j$ for any $j \in [k]$, we obtain $\nu' \notin \mu\mathcal{S}^{\mathcal{I}}(\exists \overline{\mathbf{z}}_j.\psi_j \wedge \mathcal{Q}_j(\overline{\mathbf{z}}_j)), \forall j \in [k]$, thus $\nu' \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$.

If both $k > 0$ and $i > 0$, suppose that there exists $D \in \mathfrak{D}(r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m) \vdash \{\mathcal{Q}_j\theta \mid \theta \in S_j\}_{j=1}^{i})$ not containing any leaf $\Gamma' \vdash \emptyset$. Then we can choose D and create a derivation for $\Gamma \vdash \Delta$ with the same property, which contradicts the hypothesis of the lemma. Therefore, it must be the case that every derivation in $\mathfrak{D}(r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m) \vdash \{\mathcal{Q}_j\theta \mid \theta \in S_j\}_{j=1}^{i}$ contains a leaf $\Gamma' \vdash \emptyset$.

By the induction hypothesis, there exists a valuation $\nu \in \mu\mathcal{S}^{\mathcal{I}}\left(r_1(\overline{\mathbf{y}}_1) \wedge \ldots \wedge r_m(\overline{\mathbf{y}}_m)\right) \setminus \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee_{j=1}^{i}\{\mathcal{Q}_j\theta \mid \theta \in S_j\}\right)$, and also $\nu \in \mu\mathcal{S}^{\mathcal{I}}\left(r_1(\overline{\mathbf{y}}_1) \wedge \ldots \wedge r_m(\overline{\mathbf{y}}_m)\right) \setminus \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)$ for every $j \in [i]$ and $\theta \in S_j$. Because $\mathcal{S}$ is non-filtering, there exists $\overline{\mathbf{v}} \in \mathsf{Val}^{\|\overline{\mathbf{y}}\|}$ such that $\mathcal{I}, \nu' \models \phi$ and $\nu' = \nu[\overline{\mathbf{y}} \leftarrow \overline{\mathbf{v}}]$. This means that $\nu' \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma)$. Note that the only free variables occurring in the antecedent are $\bigcup_{j=1}^{m} \overline{\mathbf{y}}_j$. Then it is also true that $\nu' \in \mu\mathcal{S}^{\mathcal{I}}\left(r_1(\overline{\mathbf{y}}_1) \wedge \ldots \wedge r_m(\overline{\mathbf{y}}_m)\right) \setminus \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)$, for every $j \in [i]$ and $\theta \in S_j$. Furthermore, because $\mathcal{S}$ is non-overlapping and $\phi \not\models^{\mathcal{I}} \exists \overline{\mathbf{z}}_j.\psi_j$ for all $j \in [i+1, k]$, it follows that $\phi \wedge \exists \overline{\mathbf{z}}_j.\psi_j$ is unsatisfiable, hence ν' is also a counterexample for each entailment $\phi \models^{\mathcal{I}} \exists \overline{\mathbf{z}}_j.\psi_j$ with $j \in [i+1, k]$, and thus for the entailment $\phi \models^{\mathcal{I}} \bigvee_{j=i+1}^{k} \exists \overline{\mathbf{z}}_j.\psi_j$.

Suppose that ν' is a model of $\exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{y}}, \overline{\mathbf{z}}_j) \wedge \mathcal{Q}_j(\overline{\mathbf{z}}_j)$ for some $j \in [k]$. Then we get $\mathcal{I}, \nu' \models \exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{y}}, \overline{\mathbf{z}}_j)$ and, because $\mathcal{S}$ is non-overlapping, we have that $\phi \models^{\mathcal{I}} \exists \overline{\mathbf{z}}_j.\psi_j$. It follows that $i \notin [i+1, k]$, otherwise it would contradict the side condition $\phi \not\models^{\mathcal{I}} \exists \overline{\mathbf{z}}_{i+1}.\psi_{i+1} \vee \ldots \vee \exists \overline{\mathbf{z}}_k.\psi_k$. Therefore, $j \in [i]$ and, since $\mathcal{S}$ has the fvi property, $\mathcal{I}, \nu' \models \psi_j\theta$ for all $\theta \in S_j$. Moreover, there is no other Skolem function that witnesses this, besides the ones in S_j. Then also $\nu' \in \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)$ for all $\theta \in S_j$ and only for these substitutions. Because the range of each $\theta \in S_j$ is $\overline{\mathbf{y}}_1 \cup \ldots \cup \overline{\mathbf{y}}_m$, it must be the case that $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\mathcal{Q}_j\theta)$ for all $\theta \in S_j$, which contradicts the fact that ν is a counterexample of the antecedent. It follows that ν' cannot be a model of $\exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{y}}, \overline{\mathbf{z}}_j) \wedge \mathcal{Q}_j(\overline{\mathbf{z}}_j)$ for any $j \in [k]$ and $\nu' \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$, as required.

($\wedge$R) Let $\Delta = r(\overline{\mathbf{y}}) \wedge s(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta'$. Then $\Gamma \vdash r(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta'$ and $\Gamma \vdash s(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta'$ are the antecedents for the application of $IR = \wedge$R on $\Gamma \vdash \Delta$. Note that, because $\wedge$R is applicable, $\Delta \neq \emptyset$. Also, the right-hand side of the antecedents cannot be $\emptyset$ either.

Suppose that there exist $D_1 \in \mathfrak{D}(\Gamma \vdash r(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta')$ and $D_2 \in \mathfrak{D}(\Gamma \vdash s(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta')$ not containing any leaf $\Gamma' \vdash \emptyset$. Then we can choose D_1 and D_2 and create a derivation for $\Gamma \vdash \Delta$ with the same property, which contradicts the hypothesis of the lemma. Therefore, it must be the case that every derivation in $\mathfrak{D}(\Gamma \vdash r(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta')$ or $\mathfrak{D}(\Gamma \vdash s(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta')$ contains a leaf $\Gamma' \vdash \emptyset$.

$$\begin{aligned}
\mu\mathcal{S}^{\mathcal{I}}\left(r(\overline{\mathbf{y}}) \wedge \mathcal{Q} \vee \bigvee \Delta'\right) &= \mu\mathcal{S}^{\mathcal{I}}\left(r(\overline{\mathbf{y}})\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(\mathcal{Q} \vee \bigvee \Delta'\right) \\
&\supseteq \mu\mathcal{S}^{\mathcal{I}}\left(r(\overline{\mathbf{y}})\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(s(\overline{\mathbf{y}})\right) \cap \mu\mathcal{S}^{\mathcal{I}}\left(\mathcal{Q} \vee \bigvee \Delta'\right) \\
&= \mu\mathcal{S}^{\mathcal{I}}\left(r(\overline{\mathbf{y}}) \wedge s(\overline{\mathbf{y}}) \wedge \mathcal{Q} \vee \bigvee \Delta'\right) = \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)
\end{aligned}$$

If every derivation in $\mathfrak{D}(\Gamma \vdash r(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta')$ contains a leaf $\Gamma' \vdash \emptyset$, by the induction hypothesis there exists $\nu_1 \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(r(\overline{\mathbf{y}}) \wedge \mathcal{Q} \vee \bigvee \Delta')$. Then it is also the case that $\nu_1 \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$, because of the inclusion above.

Similarly, if every derivation in $\mathfrak{D}(\Gamma \vdash s(\overline{\mathbf{y}}) \wedge \mathcal{Q}, \Delta')$ contains a leaf $\Gamma' \vdash \emptyset$, by the induction hypothesis there exists $\nu_2 \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(s(\overline{\mathbf{y}}) \wedge \mathcal{Q} \vee \bigvee \Delta')$ and also $\nu_2 \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$.

(SP) Let $\Gamma = \{r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m)\}$ and $\Delta = \{\bigwedge_{i=1}^{m} s_i^j(\overline{\mathbf{y}}_i)\}_{j=1}^{k}$. Then, for a given $\bar{\imath} \in [m]^{m^k}$, the antecedents for the application of $IR = \text{SP}$ are $r_{\bar{\imath}_j}(\overline{\mathbf{y}}_{\bar{\imath}_j}) \vdash \{s_{\bar{\imath}_j}^{\ell}(\overline{\mathbf{y}}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}$, for every $j \in [m^k]$.

The right-hand side of any antecedent is $\emptyset$ only if Δ was already $\emptyset$, meaning $k = 0$. Because $\overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m$ are distinct and there are no predicates in $\mathcal{S}^{\text{P}}$ with empty least solutions, it follows that $\mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma)$ is not empty. Therefore, there exists a counterexample $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta) = \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma)$.

If $\Delta \neq \emptyset$, suppose that there exists $\bar{\imath} \in [m]^{m^k}$ and $D_j \in \mathfrak{D}(r_{\bar{\imath}_j}(\overline{\mathbf{y}}_{\bar{\imath}_j}) \vdash \{s_{\bar{\imath}_j}^{\ell}(\overline{\mathbf{y}}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\})$ not containing any leaf $\Gamma' \vdash \emptyset$, for every $j \in [m^k]$. Then we can choose these derivations and create one for $\Gamma \vdash \Delta$ with the same property, which contradicts the hypothesis of the lemma. Therefore, it must be the case that, for any $\bar{\imath} \in [m]^{m^k}$ there exists $j \in [m^k]$ such that every derivation in $\mathfrak{D}(r_{\bar{\imath}_j}(\overline{\mathbf{y}}_{\bar{\imath}_j}) \vdash \{s_{\bar{\imath}_j}^{\ell}(\overline{\mathbf{y}}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\})$ contains a leaf $\Gamma' \vdash \emptyset$.

By the induction hypothesis, for any $\bar{\imath} \in [n]^{n^k}$ there exist $j \in [n^k]$ and a valuation $\nu_j \in \mu\mathcal{S}^{\mathcal{I}}(r_{\bar{\imath}_j}(\overline{\mathbf{y}}_{\bar{\imath}_j})) \setminus \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee\{s_{\bar{\imath}_j}^{\ell}(\overline{\mathbf{y}}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}\right)$. Because this is equivalent to $\mu\mathcal{S}^{\mathcal{I}}(r_{\bar{\imath}_j}) \not\subseteq \mu\mathcal{S}^{\mathcal{I}}\left(\bigvee\{s_{\bar{\imath}_j}^{\ell} \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}\right)$, it follows by Lemma 2.5.1 that $\mu\mathcal{S}^{\mathcal{I}}(r_1) \times \ldots \times \mu\mathcal{S}^{\mathcal{I}}(r_m) \not\subseteq \bigcup_{j=1}^{k} \mu\mathcal{S}^{\mathcal{I}}(q_1^j) \times \ldots \times \mu\mathcal{S}^{\mathcal{I}}(q_m^j)$. This is equivalent to $\mu\mathcal{S}^{\mathcal{I}}(r_1(\overline{\mathbf{y}}_1) \wedge \ldots \wedge r_m(\overline{\mathbf{y}}_m)) \not\subseteq \mu\mathcal{S}^{\mathcal{I}}(\bigvee_{j=1}^{k} \bigwedge_{i=1}^{n} q_i^j(\overline{\mathbf{y}}_i))$, as previously shown in the (SP) case of the proof for Theorem 2.5.4. Then, there must exist $\nu \in \mu\mathcal{S}^{\mathcal{I}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta)$. □

The following theorem establishes the completeness of the inference rule set $\mathcal{R}_{\mathsf{Ind}}$, while also providing a proof search strategy **S**.

Theorem 2.5.8. Given an non-filtering, non-overlapping FOL inductive system $\mathcal{S}$ with the fvi property, which is also ranked in the interpretation $\mathcal{I}$, let $p, q_1, \ldots, q_n \in \mathcal{S}^{\text{P}}$. Then the entailment $p \models_{\mathcal{S}}^{\mathcal{I}} q_1, \ldots, q_n$ holds only if the sequent $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ has an **S**-proof with the inference rule schemata $\mathcal{R}_{\mathsf{Ind}}$, where **S** is defined by the regular expression $(\text{LU} \cdot \text{RU}^* \cdot \text{RD} \cdot \wedge\text{R}^* \cdot \text{SP?})^* \cdot \text{LU?} \cdot \text{RU}^* \cdot (\text{AX} \mid \text{ID})$.

Proof. Since $p \models_{\mathcal{S}}^{\mathcal{I}} q_1, \ldots, q_n$ and $\mathcal{S}$ is non-filtering, non-overlapping, and has the fvi property, it follows by Lemma 2.5.7 that there exists a finite maximal, structured and irreducible derivation $D \in \mathfrak{D}(p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}))$ which does not contain any leaf of the form $\Gamma \vdash \emptyset$. Furthermore, no other node in D is of the form $\Gamma \vdash \emptyset$, because all descendants of such a node (including leaves of the derivation) would have empty right-hand sides as well.

Step 1. We first show that this derivation is actually a proof (i.e. all its leaves are $\top$). Suppose, by contradiction, that there exists a leaf which is not $\top$. Then this leaf must be a sequent $\Gamma \vdash \Delta$, where $\Delta \neq \emptyset$. Let π be the path in D from the root to $\Gamma \vdash \Delta$. Since D is a maximal derivation, π cannot be extended any further by the application of an inference

rule. Let IR be the last inference rule schema applied on π and let $\Gamma' \vdash \Delta'$ be its consequent. Since $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow \Gamma' \vdash \Delta'$, it follows by Lemma 2.5.6 that Γ' is a tree-shaped set and Δ' consists of existentially quantified finite conjunctions of tree-shaped sets. Also, IR cannot be AX or ID, otherwise the leaf would be $\top$. We do a case split on IR.

(LU) Then $\Gamma = R(\overline{\mathbf{y}}, \overline{\mathbf{y}}') \cup \Gamma' \setminus \{r(\overline{\mathbf{y}})\}$ and $\Delta = \Delta'$, where $\langle r(\overline{\mathbf{y}}), R(\overline{\mathbf{y}})\rangle \in \mathcal{S}$. If Δ contains at least one singleton predicate atom, we can still apply RU, which contradicts the fact that D is maximal. Otherwise, because $\Delta \neq \emptyset$ consists of existentially quantified finite conjunctions of tree-shaped sets and it does not contain any singleton predicate atoms, then Δ contains only existentially quantified conjunctions over predicate rules from $\mathcal{S}$, obtained from previous applications of RU, or predicate atom conjunctions that are not singleton, obtained from previous applications of RD. We distinguish the following cases:

- $\Gamma' \setminus \{r(\overline{\mathbf{y}})\} = \emptyset$. Then we can apply RD to the sequent $\Gamma \vdash \Delta$ and extend D, which leads to a contradiction.
- $\Gamma' \setminus \{r(\overline{\mathbf{y}})\} \neq \emptyset$. In this case, IR cannot be the first occurrence of LU along π, otherwise we would have $\Gamma' \setminus \{r(\overline{\mathbf{y}})\} = \emptyset$. Then there must have been a previous application of LU on π, and, because D is structured, RD must have been applied between them. Therefore, since Γ' is tree-shaped, $\Gamma' \setminus \{r(\overline{\mathbf{y}})\}$ can only contain predicate atoms, because the constraints introduced by LU are always eliminated by RD. Then we can apply LU and extend D, which leads to a contradiction.

(RU) Then $\Gamma = \Gamma'$ and $\Delta = \{\exists \overline{\mathbf{z}}_i. \bigwedge R_i(\overline{\mathbf{y}}, \overline{\mathbf{z}}_i)\}_{i=1}^m \cup \Delta' \setminus \{r(\overline{\mathbf{y}})\}$, where $r(\overline{\mathbf{y}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{y}}) \mid \ldots \mid R_m(\overline{\mathbf{y}})$. If $\Delta' \setminus \{p(\overline{\mathbf{x}})\}$ contains at least one singleton predicate atom, we can continue to apply RU and extend D, which leads to a contradiction. Otherwise, because $\Delta' \neq \emptyset$ consists of existentially quantified finite conjunctions of tree-shaped sets and it does not contain any singleton predicate atoms, then it must be the case that Δ' contains only existentially quantified conjunctions over predicate rules from $\mathcal{S}$, obtained from previous applications of RU, or conjunctions of predicate atoms that are not singleton, obtained from previous applications of RD. We distinguish the following cases for Γ:

- Γ contains a predicate atom. Then we can apply LU and extend D, which leads to a contradiction.
- Γ does not contain predicate atoms. Then it can only contain a constraint with no subgoal variables, because Γ is tree-shaped and D is structured. Therefore, we can apply RD and extend D, which leads to a contradiction.

(RD) Then $\Gamma = \{r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m)\}$ and $\Delta = \{\mathcal{Q}_1, \ldots, \mathcal{Q}_i\}$, where each $\mathcal{Q}_j, j \in [i]$ is a conjunction of predicate atoms. It is possible to apply LU – or even $\wedge$R, RU or SP if their side conditions are satisfied. This means that π can still be extended, which leads to a contradiction.

($\wedge$R) Then $\Gamma = \Gamma'$ and $\Delta = \{r(\overline{\mathbf{y}}) \wedge \mathcal{Q}\} \cup \Delta''$, where $\mathcal{Q}$ is a conjunction of predicate atoms and Δ'' only contains conjunctions of predicate atoms. Since we only apply ($\wedge$R) as cleanup after RD, Γ only contains predicate atoms. If $r(\overline{\mathbf{y}}) \wedge \mathcal{Q}$ or any member of Δ'' contains some conjunction $s_1(\overline{\mathbf{z}}) \wedge s_2(\overline{\mathbf{z}})$ we can again apply $\wedge$R. Otherwise, we can apply LU – or even RU or SP if their side conditions are satisfied. This means that π can still be extended, which leads to a contradiction.

(SP) Then $\Gamma = r(\overline{\mathbf{y}})$ and $\Delta = \{s_1(\overline{\mathbf{y}}), \ldots, s_m(\overline{\mathbf{y}})\}$. We can apply LU or RU to $\Gamma \vdash \Delta$, which means that π can still be extended and leads to a contradiction.

Step 2. We now show that the sequences of inference rules applied along each branch in D are captured by the strategy $\mathbf{S}$. Let π be an arbitrary branch in D, starting at the root. Since D is maximal, π cannot be extended by the application of any inference rule. We assume that the first instance of LU along π does not immediately follow an application of RU — otherwise, the same sequent can be obtained by changing the order of the inference rules such that LU is applied first. We do a proof by induction on the number $N \geq 1$ of basic sequents that occur on π.

($N = 1$) The only basic sequent is the root $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ and it occurs on the first position of π. In this case, SP is never applied along π, otherwise we would obtain more than one basic sequent. Note that the rules applicable on a basic sequent are AX, LU and RU. We distinguish the following cases:

- If LU is not applied on π, then the only possibility to obtain a maximal π is to directly apply AX on the root. Since we exclude LU, the only other applicable inference rule would be RU, but, since it preserves the left-hand side, π would not be maximal after all possible applications of RU. Therefore, $\Lambda(\pi) = \mathrm{AX} \in \mathbf{S}$.
- If LU is applied on π, then there are multiple possibilities for $\Lambda(\pi)$, which we obtain from analysing the sequents created by each potential choice of inference rules.
  - Since AX would close the path and we assumed that RU does not immediately precede LU, it follows that LU must be the first inference rule in $\Lambda(\pi)$. Let $\langle p(\overline{\mathbf{x}}), R(\overline{\mathbf{x}})\rangle \in \mathcal{S}$ be the predicate rule chosen for applying LU on the root;
  - We obtain $R(\overline{\mathbf{x}}, \overline{\mathbf{x}}') \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ as the second sequent in π. We can continue by LU (if R contains at least one predicate atom) or RU. However, LU is not allowed at this point because the derivation is structured, so the only choice is to apply RU for one of the predicate atoms on the right-hand side.
  - If the next sequent satisfies the side condition of AX, then we can close π. Otherwise we can continue with at most $n-1$ instances of RU. If, at any point during the right unfolding, AX becomes applicable, then π can be closed. We obtain $\Lambda(\pi) = \mathrm{LU} \cdot \mathrm{RU}^* \cdot \mathrm{AX} \in \mathbf{S}$.
  - After $n-1$ more right unfoldings, we reach a sequent $R(\overline{\mathbf{x}}, \overline{\mathbf{x}}') \vdash \exists \overline{\mathbf{y}}_1 \,.\, R_1(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1), \ldots, \exists \overline{\mathbf{y}}_m \,.\, R_m(\overline{\mathbf{x}}, \overline{\mathbf{y}}_m)$, where the right-hand side is obtained from all the predicate rules for $q_1, \ldots, q_n$ in $\mathcal{S}$. If AX is not applicable, we can only continue by RD and, immediately, as many $\wedge$R applications as necessary.
  - We reach a sequent $r_1(\overline{\mathbf{x}}_1), \ldots, r_h(\overline{\mathbf{x}}_h) \vdash \mathcal{Q}_1, \ldots, \mathcal{Q}_k$, where each $\mathcal{Q}_i$ with $i \in [k]$ is a conjunction of predicate atoms over the tuples $\{\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_h\}$, such that no two predicate atoms in the same $\mathcal{Q}_i$ have the same arguments. Then $h \neq 1$, otherwise this sequent would be basic, contradicting the assumption that $N = 1$. Also, $h \neq 0$, otherwise it would lead to $k = 0$ after RD, which, by Lemma 2.5.7, indicates the existence of a counterexample and contradicts the fact that D is a proof. Since SP cannot occur along π, the only possibilities for continuation are AX, ID, LU and RU. However, LU is applicable at most once, because RD is required between two consecutive left unfoldings and RD is only applicable on sequents with a singly tree-shaped left-hand side, which is impossible here due

to $h > 1$. Also, RU can only be applied a finite number of times, equal to the number of singleton predicate atoms on the right hand side. In both cases, π is not maximal, because LU is enabled by $h > 1$. Then the only possibility is to end π by AX or ID, obtaining $\Lambda(\pi) \in \mathrm{LU} \cdot \mathrm{RU}^* \cdot \mathrm{RD} \cdot \wedge\mathrm{R}^* \cdot \mathrm{LU?} \cdot \mathrm{RU}^* \cdot (\mathrm{AX} \mid \mathrm{ID}) \subseteq \mathbf{S}$.

($N > 1$) Let $\pi = \tau \cdot \rho$, where ρ starts with the second occurrence of a basic sequent in π. As before, the first occurrence is the initial sequent $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$. Then the head of ρ is obtained after a sequence $\mathrm{RD} \cdot \wedge\mathrm{R}^* \cdot \mathrm{SP?}$, where SP is only required if the left-hand side for the consequent of RD contains more than one predicate atom. The left-hand side of this RD consequent is a predicate rule in $\mathcal{S}$ with goal $p(\overline{\mathbf{x}})$, introduced by a previous application of LU. It follows that only RU can be applied between LU and RD and, also, RU is the only other rule applicable before LU. However, the order of the LU and RU applications is not important and we consider that the left unfolding is always performed first. Assuming that $\Lambda(\rho) \in \mathbf{S}$, we obtain that $\Lambda(\pi) \in (\mathrm{LU} \cdot \mathrm{RU}^* \cdot \mathrm{RD} \cdot \wedge\mathrm{R}^* \cdot \mathrm{SP?}) \cdot \mathbf{S} \subseteq \mathbf{S}$. □

As previously discussed, the proof-search semi-algorithm 1 only produces irreducible derivations. If it is executed with $\mathbf{S}$, then these derivations will also be structured. This guarantees termination because, by point (1) of Lemma 2.5.5, each irreducible and structured derivation is finite. If the input inductive system $\mathcal{S}$ is ranked, non-filtering, non-overlapping and has the fvi property, then $\mathcal{R}_{\mathsf{Ind}}$ is complete, thus algorithm 1 becomes a decision procedure for this class of entailment problems.

2.5.2 The Soundness and Completeness of $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$

Soundness

We develop our argument for the soundness of $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ in a similar fashion as the one for $\mathcal{R}_{\mathsf{Ind}}$. The following lemma is the counterpart of Lemma 2.5.1 and its proof adapts the result of [24, Theorem 1] to support the local soundness of $\mathrm{SP}_{\mathsf{SL}}$.

Lemma 2.5.9. Consider an SL inductive system $\mathcal{S}$, the predicates $p_1, \ldots, p_n \in \mathcal{S}^{\mathsf{P}}$, and the tuples of predicates $\overline{\mathcal{Q}}_j = \langle q_1^j, \ldots, q_n^j \rangle \in (\mathcal{S}^{\mathsf{P}})^n$, $j \in [k]$. Then

$$\mu\mathcal{S}^{\mathsf{SL}}(p_1(\overline{\mathbf{x}}_1) * \ldots * p_n(\overline{\mathbf{x}}_n)) \subseteq \bigcup_{j=1}^{k} \mu\mathcal{S}^{\mathsf{SL}}(q_1^j(\overline{\mathbf{x}}_1) * \ldots * q_n^j(\overline{\mathbf{x}}_n))$$

only if there exists a tuple $\overline{\imath} \in [n]^{n^k}$, such that:

$$\mu\mathcal{S}^{\mathsf{SL}}(p_{\overline{\imath}_j}) \subseteq \{\mu\mathcal{S}^{\mathsf{SL}}(q^{\ell}_{\overline{\imath}_j}) \mid \ell \in [k], f_j(\overline{\mathcal{Q}}_\ell) = \overline{\imath}_j\}$$

for all $j \in [n^k]$, where $\mathcal{F}(\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_k) = \{f_1, \ldots, f_{n^k}\}$.

Proof. Let $\mathcal{U}_k = \mathsf{L}^k \times \mathsf{Heaps}$. Then, given some predicates $r_1, \ldots, r_n \in \mathcal{S}^{\mathsf{P}}$, the following property holds:

$$\biguplus_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}(r_i) \subseteq \bigcap_{i=1}^{n} \left(\biguplus_{j=1}^{i-1} \mathcal{U}_{\|\overline{\mathbf{x}}_j\|} \uplus \mu\mathcal{S}^{\mathsf{SL}}(r_i) \uplus \biguplus_{j=i+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_j\|} \right) \tag{2.5}$$

This proof closely follows the outline of the proof for Theorem 1 in [24]. Using property (2.5), the inclusion we need to prove can be rewritten as

$$\begin{aligned}
&\mu\mathcal{S}^{\mathsf{SL}}(p_1(\overline{\mathbf{x}}_1) * \ldots * p_n(\overline{\mathbf{x}}_n)) \subseteq \bigcup_{j=1}^{k} \mu\mathcal{S}^{\mathsf{SL}}(q_1^j(\overline{\mathbf{x}}_1) * \ldots * q_n^j(\overline{\mathbf{x}}_n)) && \Leftrightarrow \\
&\biguplus_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}(p_i) \subseteq \bigcup_{j=1}^{k} \biguplus_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}(q_i^j) && \Leftrightarrow \\
&\biguplus_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}(p_i) \subseteq \bigcup_{j=1}^{k} \bigcap_{i=1}^{n} \left(\biguplus_{\ell=1}^{i-1} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \uplus \mu\mathcal{S}^{\mathsf{SL}}(q_i^j) \uplus \biguplus_{\ell=i+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \right) && (2.6)
\end{aligned}$$

As in the proof of [24, Theorem 1], because the power set lattice $(2^V, \subseteq)$ of any set V is a completely distributive lattice, for any doubly indexed family

$$\{S_{j,k} \in 2^V \mid j \in J, k \in K_j\}$$

it holds that

$$\bigcup_{j \in J} \bigcap_{k \in K_j} S_{j,k} = \bigcap_{f \in F} \bigcup_{j \in J} S_{j,f(j)}$$

where F is the set of choice functions f choosing for each index $j \in J$ some index $f(j) \in K_j$. In our case, let $F = \mathcal{F}(\overline{\mathcal{Q}}_1, \ldots, \overline{\mathcal{Q}}_k)$. Then, we can rewrite (2.6) as

$$\begin{aligned}
&\biguplus_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}(p_i) \subseteq \bigcup_{j=1}^{k} \bigcap_{i=1}^{n} \left(\biguplus_{\ell=1}^{i-1} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \uplus \mu\mathcal{S}^{\mathsf{SL}}(q_i^j) \uplus \biguplus_{\ell=i+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \right) && \Leftrightarrow \\
&\biguplus_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}(p_i) \subseteq \bigcap_{f \in F} \bigcup_{j=1}^{k} \left(\biguplus_{\ell=1}^{f(\overline{\mathcal{Q}}_j)-1} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \uplus \mu\mathcal{S}^{\mathsf{SL}}(q^j_{f(\overline{\mathcal{Q}}_j)}) \uplus \biguplus_{\ell=f(\overline{\mathcal{Q}}_j)+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \right) && \Leftrightarrow \\
&\forall f \in F. \biguplus_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}(p_i) \subseteq \bigcup_{j=1}^{k} \left(\biguplus_{\ell=1}^{f(\overline{\mathcal{Q}}_j)-1} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \uplus \mu\mathcal{S}^{\mathsf{SL}}(q^i_{f(\overline{\mathcal{Q}}_j)}) \uplus \biguplus_{\ell=f(\overline{\mathcal{Q}}_j)+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \right) && (2.7)
\end{aligned}$$

For a fixed f, we can rewrite the right hand-side of (2.7) as

$$\begin{aligned}
&\bigcup_{j=1}^{k} \left(\biguplus_{\ell=1}^{f(\overline{\mathcal{Q}}_j)-1} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \uplus \mu\mathcal{S}^{\mathsf{SL}}(q^j_{f(\overline{\mathcal{Q}}_j)}) \uplus \biguplus_{\ell=f(\overline{\mathcal{Q}}_j)+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \right) && = \\
&\bigcup_{i=1}^{n} \bigcup_{j \in [k], f(\overline{\mathcal{Q}}_j)=i} \left(\biguplus_{\ell=1}^{i-1} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \uplus \mu\mathcal{S}^{\mathsf{SL}}(q_i^j) \uplus \biguplus_{\ell=i+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \right) && = \\
&\bigcup_{i=1}^{n} \left(\biguplus_{\ell=1}^{i-1} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \uplus \left(\bigcup_{j \in [k], f(\overline{\mathcal{Q}}_j)=i} \mu\mathcal{S}^{\mathsf{SL}}(q_i^j) \right) \uplus \biguplus_{\ell=i+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \right) && (2.8)
\end{aligned}$$

Using (2.8), the inclusion query (2.7) becomes

$$\forall f \in F.\, \biguplus_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}(p_i) \subseteq \bigcup_{i=1}^{n} \left(\biguplus_{\ell=1}^{i-1} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \uplus \left(\bigcup_{j\in[k], f(\overline{\mathcal{Q}}_j)=i} \mu\mathcal{S}^{\mathsf{SL}}(q_i^j) \right) \uplus \biguplus_{\ell=i+1}^{n} \mathcal{U}_{\|\overline{\mathbf{x}}_\ell\|} \right) \quad \Leftrightarrow$$

$$\forall f \in F\, \exists i \in [n]\,.\, \mu\mathcal{S}^{\mathsf{SL}}(p_i) \subseteq \bigcup_{j\in[k], f(\overline{\mathcal{Q}}_j)=i} \mu\mathcal{S}^{\mathsf{SL}}(q_i^j) \quad \Leftrightarrow$$

$$\bigwedge_{j=1}^{n^k} \bigvee_{i=1}^{n} \left(\mu\mathcal{S}^{\mathsf{SL}}(p_i) \subseteq \{\mu\mathcal{S}^{\mathsf{SL}}(q_i^\ell) \mid \ell \in [k], f_j(\overline{\mathcal{Q}}_\ell) = i\} \right) \quad \Leftrightarrow$$

$$\bigvee_{\bar{\imath}\in[n]^{n^k}} \bigwedge_{j=1}^{n^k} \left(\mu\mathcal{S}^{\mathsf{SL}}(p_{\bar{\imath}_j}) \subseteq \{\mu\mathcal{S}^{\mathsf{SL}}(q_{\bar{\imath}_j}^\ell) \mid \ell \in [k], f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\} \right) \tag{2.9}$$

Inclusion (2.9) is true only when there exists a tuple $\bar{\imath} \in [n]^{n^k}$, for all $j \in [n^k]$, such that $\mu\mathcal{S}^{\mathsf{SL}}(p_{\bar{\imath}_j}) \subseteq \{\mu\mathcal{S}^{\mathsf{SL}}(q_{\bar{\imath}_j}^\ell) \mid \ell \in [k], f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}$. □

The local soundness of $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}} \setminus \{\mathrm{ID}\}$ means that, if the consequent $\Gamma \vdash \Delta$ denotes an invalid entailment, then at least one of its antecedents also denotes an invalid entailment. Moreover, their respective counterexamples can be related by $\trianglelefteq$.

Lemma 2.5.10. Given a ranked SL inductive system $\mathcal{S}$, for each instance of an inference rule schema in $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}} \setminus \{\mathrm{ID}\}$, having the consequent $\Gamma \vdash \Delta$ and antecedents $\Gamma_i \vdash \Delta_i$ with $i \in [n]$, and each $(\nu, h) \in \mu\mathcal{S}^{\mathsf{SL}}(\bigwedge \Gamma) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta)$, there exists $(\nu_i, h_i) \in \mu\mathcal{S}^{\mathsf{SL}}(\bigwedge \Gamma_i) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta_i)$ for some $i \in [n]$ such that $h_i \trianglelefteq h$.

Proof. For $\mathrm{AX}_{\mathsf{SL}}$, soundness follows from the side condition and its consequent admits no counterexamples, thus the lemma is trivially true in this case. We analyse LU, $\mathrm{RU}_{\mathsf{SL}}$, $\mathrm{RD}_{\mathsf{SL}}$, $\wedge$R and $\mathrm{SP}_{\mathsf{SL}}$ individually, in a similar fashion as we did for the local soundness of $\mathcal{R}_{\mathsf{Ind}}$ (Lemma 2.5.2).

(LU) Let $p(\overline{\mathbf{x}}) \in \Gamma$ be a predicate atom, where $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}})$. The antecedents of $\Gamma \vdash \Delta$ are $\Gamma_i \vdash \Delta_i \equiv R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i), \Gamma \setminus p(\overline{\mathbf{x}}) \vdash \Delta$, where $i \in [n]$ and each $\overline{\mathbf{y}}_i$ is a tuple of fresh variables. In this case, the least solution of Γ is

$$\begin{aligned}
\mu\mathcal{S}^{\mathsf{SL}}(*\,\Gamma) &= \mu\mathcal{S}^{\mathsf{SL}}\left(p(\overline{\mathbf{x}}) * *\,(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right) = \mu\mathcal{S}^{\mathsf{SL}}(p(\overline{\mathbf{x}})) \uplus \mu\mathcal{S}^{\mathsf{SL}}\left(*\,(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \left(\bigcup_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}\left(*\, R_i(\overline{\mathbf{x}})\right) \right) \uplus \mu\mathcal{S}^{\mathsf{SL}}\left(*\,(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \bigcup_{i=1}^{n} \left(\mu\mathcal{S}^{\mathsf{SL}}\left(*\, R_i(\overline{\mathbf{x}})\right) \uplus \mu\mathcal{S}^{\mathsf{SL}}\left(*\,(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right)\right) \\
&= \bigcup_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}\left(*\, R_i(\overline{\mathbf{x}}) * *\,(\Gamma \setminus \{p(\overline{\mathbf{x}})\})\right)
\end{aligned}$$

If there exists $(\nu, h) \in \mu\mathcal{S}^{\mathsf{SL}}(*\,\Gamma) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta)$, then $(\nu, h) \in \mu\mathcal{S}^{\mathsf{SL}}(*\, R_i(\overline{\mathbf{x}}) \wedge *(\Gamma \setminus \{p(\overline{\mathbf{x}})\})) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta)$ for some $i \in [n]$. In consequence, there also exists $(\nu_i, h_i) \in \mu\mathcal{S}^{\mathsf{SL}}(*\, R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i) * *(\Gamma \setminus \{p(\overline{\mathbf{x}})\})) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta)$ such that $h_i = h$ and for every $x \in \mathrm{FV}(\Gamma)$ we have $\nu_i(x) = \nu(x)$. Then $h_i \trianglelefteq h$ holds trivially.

($\mathrm{RU}_{\mathsf{SL}}$) Let $p(\overline{\mathbf{x}}) \in \Delta$ be a predicate atom, where $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_n(\overline{\mathbf{x}})$. Then $\Gamma \vdash \Delta$ has only one antecedent $\Gamma_1 \vdash \Delta_1 \equiv \Gamma \vdash \exists \overline{\mathbf{y}}_1 . * R_1(\overline{\mathbf{x}}, \overline{\mathbf{y}}_1), \ldots, \exists \overline{\mathbf{y}}_n . * R_n(\overline{\mathbf{x}}, \overline{\mathbf{y}}_n), \Delta \setminus p(\overline{\mathbf{x}})$, where each $\overline{\mathbf{y}}_i$ with $i \in [n]$ is a tuple of fresh variables. In this case, the least solution of Δ is

$$\begin{aligned}
\mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee \Delta\right) &= \mu\mathcal{S}^{\mathsf{SL}}\left(p(\overline{\mathbf{x}}) \vee \bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) = \mu\mathcal{S}^{\mathsf{SL}}(p(\overline{\mathbf{x}})) \cup \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \left(\bigcup_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}\left(* R_i(\overline{\mathbf{x}})\right)\right) \cup \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \left(\bigcup_{i=1}^{n} \mu\mathcal{S}^{\mathsf{SL}}\left(\exists \overline{\mathbf{y}}_i . * R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i)\right)\right) \cup \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee_{i=1}^{n} \exists \overline{\mathbf{y}}_i . * R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i)\right) \cup \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) \\
&= \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee_{i=1}^{n} \exists \overline{\mathbf{y}}_i . * R_i(\overline{\mathbf{x}}, \overline{\mathbf{y}}_i) \vee \bigvee (\Delta \setminus \{p(\overline{\mathbf{x}})\})\right) = \mu\mathcal{S}^{\mathcal{I}}(\bigvee \Delta_1)
\end{aligned}$$

If there exists $(\nu, h) \in \mu\mathcal{S}^{\mathsf{SL}}\left(* \Gamma\right) \setminus \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee \Delta\right)$, then it is also the case that $(\nu, h) \in \mu\mathcal{S}^{\mathsf{SL}}(* \Gamma_1) \setminus \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee \Delta_1\right)$. Therefore, the counterexample for the antecedent is $(\nu_1, h_1) = (\nu, h)$ and $h_1 \trianglelefteq h$ holds trivially.

($\mathrm{RD}_{\mathsf{SL}}$) Then the sequent $\Gamma \vdash \Delta \equiv \phi(\overline{\mathbf{x}}, \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_n), p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\exists \overline{\mathbf{y}}_j . \psi_j(\overline{\mathbf{x}}, \overline{\mathbf{y}}_j) * \mathcal{Q}_j(\overline{\mathbf{y}}_j)\}_{j=1}^{k}$ has only one antecedent $\Gamma_1 \vdash \Delta_1 \equiv p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\mathcal{Q}_j\theta \mid \theta \in S_j\}_{j=1}^{i}$. By the side condition of RD, $\phi \models^{\mathsf{SL}} *_{j=1}^{i} \exists \overline{\mathbf{y}}_j . \psi_j$. Also, by Definition 2.4.6, we have $\mu\mathcal{S}^{\mathsf{SL}}(\phi) \subseteq \mu\mathcal{S}^{\mathsf{SL}}(\psi_j\theta)$ for each $\theta \in \mathrm{VIS}(\phi, \psi_j)$ and $j \in [i]$. In this case, the least solution of Δ is

$$\begin{aligned}
\mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee \Delta\right) &= \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee_{j=1}^{k} \exists \overline{\mathbf{y}}_j . \psi_j * \mathcal{Q}_j\right) = \bigcup_{j=1}^{k} \mu\mathcal{S}^{\mathsf{SL}}\left(\exists \overline{\mathbf{y}}_j . \psi_j * \mathcal{Q}_j\right) \\
&\supseteq \bigcup_{j=1}^{i} \mu\mathcal{S}^{\mathsf{SL}}\left(\exists \overline{\mathbf{y}}_j . \psi_j * \mathcal{Q}_j\right) \supseteq \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \mu\mathcal{S}^{\mathsf{SL}}\left((\psi_j * \mathcal{Q}_j)\theta\right) \\
&= \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \mu\mathcal{S}^{\mathsf{SL}}\left(\psi_j\theta * \mathcal{Q}_j\theta\right) = \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \left(\mu\mathcal{S}^{\mathsf{SL}}\left(\psi_j\theta\right) \uplus \mu\mathcal{S}^{\mathsf{SL}}(\mathcal{Q}_j\theta)\right) \\
&= \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \mu\mathcal{S}^{\mathsf{SL}}\left(\psi_j\theta\right) \uplus \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \mu\mathcal{S}^{\mathsf{SL}}(\mathcal{Q}_j\theta)
\end{aligned}$$

Note that also $\mu\mathcal{S}^{\mathsf{SL}}(* \Gamma) = \mu\mathcal{S}^{\mathsf{SL}}\left(\phi\right) \uplus \mu\mathcal{S}^{\mathsf{SL}}\left(p_1(\overline{\mathbf{x}}_1) * \ldots * p_n(\overline{\mathbf{x}}_n)\right)$. If there exists $(\nu, h) \in \mu\mathcal{S}^{\mathsf{SL}}(* \Gamma) \setminus \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee \Delta\right)$, then $h = h' \uplus h''$ such that $(\nu, h') \in \mu\mathcal{S}^{\mathsf{SL}}\left(\phi\right)$ and $(\nu, h'') \in \mu\mathcal{S}^{\mathsf{SL}}(p_1(\overline{\mathbf{x}}_1) * \ldots * p_n(\overline{\mathbf{x}}_n))$. It follows that we have $(\nu, h' \uplus h'') \in \left(\mu\mathcal{S}^{\mathsf{SL}}\left(\phi\right) \uplus \mu\mathcal{S}^{\mathsf{SL}}\left(p_1(\overline{\mathbf{x}}_1) * \ldots * p_n(\overline{\mathbf{x}}_n)\right)\right) \setminus \left(\bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \mu\mathcal{S}^{\mathsf{SL}}\left(\psi_j\theta\right) \uplus \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \mu\mathcal{S}^{\mathsf{SL}}(\mathcal{Q}_j\theta)\right)$ and, since as previously stated $\mu\mathcal{S}^{\mathsf{SL}}(\phi) \subseteq \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \mu\mathcal{S}^{\mathsf{SL}}\left(\psi_j\theta\right)$, we obtain $(\nu, h'') \in \mu\mathcal{S}^{\mathsf{SL}}\left(p_1(\overline{\mathbf{x}}_1) * \ldots * p_n(\overline{\mathbf{x}}_n)\right) \setminus \bigcup_{j=1}^{i} \bigcup_{\theta \in \mathrm{VIS}(\phi, \psi_j)} \mu\mathcal{S}^{\mathsf{SL}}(\mathcal{Q}_j\theta) = \mu\mathcal{S}^{\mathsf{SL}}(* \Gamma_1) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta_1)$. Therefore, the counterexample for the antecedent is $(\nu_1, h_1) = (\nu, h'')$. Because ϕ is introduced to the left-hand side by left unfolding and $\mathcal{S}$ is ranked, we have that $h' \neq \emptyset$ and, thus, $h_1 \trianglelefteq h$.

($\wedge$R) As in the $\wedge$R case from the proof of Theorem 2.5.4, we have that, for any counterexample $(\nu, h) \in \mu\mathcal{S}^{\mathsf{SL}}(* \Gamma) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta)$, it is also the case that $(\nu, h) \in (\mu\mathcal{S}^{\mathsf{SL}}(* \Gamma_1) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta_1)) \cup (\mu\mathcal{S}^{\mathsf{SL}}(* \Gamma_2) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta_2))$. Therefore, $\nu \in \mu\mathcal{S}^{\mathsf{SL}}(* \Gamma_i) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta_i)$ for some $i \in [2]$ and the counterexample for $\Gamma_i \vdash \Delta_i$ is $(\nu_i, h_i) = (\nu, h)$. Then $h_i \trianglelefteq h$ holds trivially.

($\mathrm{SP}_{\mathsf{SL}}$) Then $\Gamma \vdash \Delta \equiv p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{*_{i=1}^{n} q_i^j(\overline{\mathbf{x}}_i)\}_{j=1}^{k}$. For each $\bar{\imath} \in [n]^{n^k}$, the antecedents of $\Gamma \vdash \Delta$ are $\Gamma_j^{\bar{\imath}} \vdash \Delta_j^{\bar{\imath}} \equiv p_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \vdash \{q_{\bar{\imath}_j}^{\ell}(\overline{\mathbf{x}}_{\bar{\imath}_j}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\}, j \in [n^k]$.

If there exists $(\nu, h) \in \mu\mathcal{S}^{\mathsf{SL}}\left(*_{i=1}^{n} p_i(\overline{\mathbf{x}}_i)\right) \setminus \mu\mathcal{S}^{\mathsf{SL}}\left(\bigvee_{j=1}^{k} *_{i=1}^{n} q_i^j(\overline{\mathbf{x}}_i)\right)$, then by the proof of Lemma 2.5.9, there exists $j \in [n^k]$ and pairs $(\nu_1, h_1), \ldots, (\nu_n, h_n)$ such that $(\nu_i, h_i) \in \mu\mathcal{S}^{\mathsf{SL}}(p_i(\overline{\mathbf{x}}_i)) \setminus \bigcup\{\mu\mathcal{S}^{\mathsf{SL}}(q_i^{\ell}) \mid \ell \in [k],\ f_j(\overline{\mathcal{Q}}_\ell(\overline{\mathbf{x}}_i))) = i\}$ for all $i \in [n]$, where $h = h_1 \uplus \ldots \uplus h_n$ and $\nu(\overline{\mathbf{x}}_i) = \nu_i(\overline{\mathbf{x}}_i)$ for each $i \in [n]$. In other words, for all tuples $\bar{\imath} \in [n]^{n^k}$ we have $(\nu_{\bar{\imath}_j}, h_{\bar{\imath}_j}) \in \mu\mathcal{S}^{\mathsf{SL}}(p_{\bar{\imath}_j}(\overline{\mathbf{x}}_{\bar{\imath}_j})) \setminus \bigcup\{\mu\mathcal{S}^{\mathsf{SL}}(q_{\bar{\imath}_j}^{\ell}(\overline{\mathbf{x}}_{\bar{\imath}_j})) \mid \ell \in [k], f_j(\overline{\mathcal{Q}}_\ell) = \bar{\imath}_j\} = \mu\mathcal{S}^{\mathsf{SL}}(* \Gamma_j^{\bar{\imath}}) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta_j^{\bar{\imath}})$. Therefore, given such $j \in [n^k]$, the counterexample for each antecedent $\Gamma_j^{\bar{\imath}} \vdash \Delta_j^{\bar{\imath}}$ is $(\nu_{\bar{\imath}_j}, h_{\bar{\imath}_j})$. Since $h = h_1 \uplus \ldots \uplus h_n$, we have $h_{\bar{\imath}_j} \trianglelefteq h$, for each $\bar{\imath} \in [n]^{n^k}$. □

Based on this result, we again consider a reachability relation between counterexamples and write $(\nu, h) \succcurlyeq (\nu', h')$ when, given any instance of an inference rule in $\mathcal{R}_{\mathsf{Ind}}^{\mathsf{SL}} \setminus \mathrm{ID}$, (ν, h) is a counterexample of the consequent and (ν', h') is a counterexample of one of its antecedents obtained from (ν, h), as shown in the proof of Lemma 2.5.10. With this in mind, we revisit the definition of a counterexample path and adapt it to SL.

Definition 2.5.6 (Counterexample path in SL)**.** A path $\pi = v_1, v_2, \ldots, v_k$ in a proof $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ built with $\mathcal{R}_{\mathsf{Ind}}^{\mathsf{SL}}$ is a *counterexample path* if there exists a sequence of pairs $(\nu_1, h_1), (\nu_2, h_2), \ldots, (\nu_k, h_k)$ such that, for all $i \in [k]$ we have: (i) $(\nu_i, h_i) \in \mu\mathcal{S}^{\mathsf{SL}}(* \Gamma_i) \setminus \mu\mathcal{S}^{\mathsf{SL}}(\bigvee \Delta_i)$, where $\mathsf{Seq}(v_i) = \Gamma_i \vdash \Delta_i$, and (ii) $(\nu_i, h_i) \succcurlyeq (\nu_{i+1}, h_{i+1})$ if $i < k$.

The following lemma is the counterpart of Lemma 2.5.3 and shows that, if the given SL inductive system is ranked, any direct counterexample path in a proof causes a strict heap decrease in the counterexamples for the pivot and consequent delimiting the path.

Lemma 2.5.11. Given a ranked SL inductive system $\mathcal{S}$, let $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ be a proof built with $\mathcal{R}_{\mathsf{Ind}}^{\mathsf{SL}}$ and $\pi = v_1, \ldots, v_k$ be a direct counterexample path in D for a backlink (v_k, v_1), with counterexamples $(\nu_1, h_1) \succcurlyeq \ldots \succcurlyeq (\nu_k, h_k)$. Then $h_1 \rhd h_k$.

Proof. Through a similar reasoning as the one in the proof of Lemma 2.5.3 and using Lemma 2.5.10 to support the fact that $h_1 \trianglerighteq \ldots \trianglerighteq h_k$, we obtain that $\mathrm{RD}_{\mathsf{SL}}$ is required to occur in $\Lambda(\pi)$, leading to $h_i \rhd h_{i+1}$ for some $i \in [k-1]$. Then $h_1 \rhd h_k$. □

Finally, the following theorem extends the reachability relation $\succcurlyeq$ to backlinks and shows that any infinite trace in a proof leads to an infinite strictly decreasing sequence of heaps, contradicting the well-foundedness of the wfqo which makes the inductive system be ranked (Definition 2.4.4). Then there cannot exist a counterexample for any sequent labelling the root of a proof built with $\mathcal{R}_{\mathsf{Ind}}^{\mathsf{SL}}$ and the associated entailment must hold.

Theorem 2.5.12. Given a ranked SL inductive system $\mathcal{S}$, if a sequent $\Gamma \vdash \Delta$ has a proof $D = (V, v_0, \mathsf{Seq}, \mathsf{Rule}, \mathsf{Par}, \mathsf{Piv})$ built with $\mathcal{R}_{\mathsf{Ind}}^{\mathsf{SL}}$ and $\mathsf{Seq}(v_0) \equiv \Gamma \vdash \Delta$, then the entailment $* \Gamma \models_{\mathcal{S}}^{\mathsf{SL}} \bigvee \Delta$ holds.

Proof. This proof by contradiction closely follows the one of Theorem 2.5.4. We suppose that $* \Gamma \models^{\mathsf{SL}}_{\mathcal{S}} \bigvee \Delta$ does not hold and, by Lemma 2.5.10, obtain a path from v_0 to a leaf $v_k \in V$ and an associated sequence of counterexamples $(\nu_1, h_1) \succcurlyeq \ldots \succcurlyeq (\nu_k, h_k)$.

Clearly, $\mathsf{Rule}(v_k) = \text{ID}$. Let $v_{k+1} = \mathsf{Piv}(v_k)$, $\mathsf{Seq}(v_k) = \Gamma_k \vdash \Delta_k$ and $\mathsf{Seq}(v_{k+1}) = \Gamma_{k+1} \vdash \Delta_{k+1}$ such that $\Gamma_k = \Gamma_{k+1}\theta$, $\Delta_k = \Delta'_{k+1}\theta$ and $\Delta_{k+1} \subseteq \Delta'_{k+1}$, for some injective substitution $\theta : \mathrm{FV}(\Gamma_{k+1} \cup \Delta_{k+1}) \to \mathrm{FV}(\Gamma_k \cup \Delta_k)$. Again, we can assume w.l.o.g. that θ is surjective, by defining $\theta(x) = x$ for each $x \in \mathrm{FV}(\Gamma_k \cup \Delta_k) \setminus \theta(\mathrm{FV}(\Gamma_{k+1} \cup \Delta_{k+1}))$. Since θ is also injective, by the side condition of ID, its inverse exists and $(\nu_k\theta^{-1}, h_k)$ is a counterexample for $\Gamma_{k+1} \vdash \Delta_{k+1}$. Therefore, we can extend the relation $\succcurlyeq$ with the pair $((\nu_k, h_k), (\nu_k\theta^{-1}, h_k))$.

We obtain an infinite trace $\tau = v_0, v_1, \ldots$ in D together with an infinite sequence $(\nu_0, h_0) \succcurlyeq (\nu_1, h_1) \succcurlyeq \ldots$. By Lemma 2.5.10, we have $h_i \trianglerighteq h_{i+1}$, for all $i \geq 0$. By Proposition 2.2.1, τ contains infinitely many direct paths $\pi_j = v_{k_j}, \ldots, v_{\ell_j}$, where $\{k_j\}_{j \geq 0}$ and $\{\ell_j\}_{j \geq 0}$ are infinite strictly increasing sequences of integers such that $k_j < \ell_j \leq k_{j+1} < \ell_{j+1}$, for all $j \geq 0$. By Lemma 2.5.11, we obtain that $h_{k_j} \rhd h_{\ell_j}$, for all $j \geq 0$. Since $h_{\ell_j} \trianglerighteq h_{k_{j+1}}$ for all $j \geq 0$, this leads to a strictly decreasing sequence $h_{k_0} \rhd h_{k_1} \rhd \ldots$, contradicting the fact that $\trianglelefteq$ is a wfqo. We can conclude that our initial assumption was false, thus the entailment $* \Gamma \models^{\mathsf{SL}}_{\mathcal{S}} \bigvee \Delta$ holds. □

Completeness

The $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ proof system is not complete for SL entailments, even if the predicates involved in the entailment are defined by inductive systems that satisfy all the restrictions introduced in Section 2.4. Instead, we will show completeness for a more restricted class of entailment problems. Whether a complete set of inference rules for the general entailment problem in SL exists is, to our knowledge, still an open question.

The restricted semantics that we consider for SL entailments take into consideration the way a predicate definition unfolds a heap. The same heap can be built in different ways, but $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ is complete only for entailments that involve predicates whose definitions unfold the heap in a similar manner.

Definition 2.5.7 (Coverage trees)**.** A coverage tree for a heap $h \in \mathsf{Heaps}$ is a tree $t : \mathbb{N}^* \rightharpoonup_{\mathsf{fin}} \mathsf{Heaps}$ such that $\mathrm{dom}(t(\alpha_1)) \cap \mathrm{dom}(t(\alpha_2)) = \emptyset$ for all $\alpha_1, \alpha_2 \in \mathrm{dom}(t)$ with $\alpha_1 \neq \alpha_2$ and also $h = \biguplus_{\alpha \in \mathrm{dom}(t)} t(\alpha)$. In other words, the labels of the coverage tree form a partition of the heap. We write Cover to mean the set of all coverage trees.

We restrict the class of SL entailments by defining a new type of assignments that map a predicate of arity n into a subset of $\mathsf{L}^n \times \mathsf{Heaps} \times \mathsf{Cover}$. Given an SL inductive system $\mathcal{S}$, we refer to the set of all assignments on the predicates of $\mathcal{S}$ as $\mathsf{Assign}^{\cup}(\mathcal{S}^{\mathsf{P}})$. We extend any assignment $\mathcal{X} \in \mathsf{Assign}^{\cup}(\mathcal{S}^{\mathsf{P}})$ to a singly tree-shaped set represented as a tree T and define $\mathcal{X}(* T)$ as the set of tuples (ν, h, t), where $\nu : \bigcup_{\alpha \in \mathrm{dom}(T)} \mathrm{FV}(T(\alpha)) \to \mathsf{Loc}$ is a valuation, h is a heap and t is a coverage tree for h such that:

- For each $\alpha \in \mathrm{dom}(T) \setminus \mathrm{fr}(T)$ we have that $\nu, t(\alpha) \models^{\mathsf{SL}} T(\alpha)$;
- For each $\alpha \in \mathrm{fr}(T)$ such that $T(\alpha) = p(\overline{\mathbf{x}})$ there exists $(\nu(\overline{\mathbf{x}}), t(\alpha), t_{|_\alpha}) \in \mathcal{X}(p)$.

We can further extend the above definition to a tree-shaped set $T = T_1 \cup \ldots \cup T_k$, which consists of multiple singly tree-shaped sets $T_1, \ldots, T_k$. Then $\mathcal{X}(* T) = \mathcal{X}(* T_1 * \ldots * * T_k) = \{(\nu, h_1 \uplus \ldots \uplus h_k, \{t_1, \ldots, t_k\}) \mid (\nu, h_i, t_i) \in \mathcal{X}(* T_i),\ i \in [k]\}$. As it was the case for the

previous types of assignments, $\mathcal{X}(*T \wedge *T') = \mathcal{X}(*T) \cap \mathcal{X}(*T')$, $\mathcal{X}(*T \vee *T') = \mathcal{X}(*T) \cup \mathcal{X}(*T')$ and $\mathcal{X}(\exists \mathbf{x}.\ *T) = \mathcal{X}(*T)$ for any two tree-shaped sets T and T'.

The SL inductive system $\mathcal{S}$ now induces a function on assignments, $\mathbb{F}^{\mathsf{U}}_{\mathcal{S}} : \mathsf{Assign}^{\mathsf{U}}(\mathcal{S}^{\mathsf{P}}) \to \mathsf{Assign}^{\mathsf{U}}(\mathcal{S}^{\mathsf{P}})$, such that $\mathbb{F}^{\mathsf{U}}_{\mathcal{S}}(\mathcal{X})(p) = \bigcup_{i=1}^{m} \{(\nu(\overline{\mathbf{x}}), h, t) \mid (\nu, h, t) \in \mathcal{X}(R_i)\}$, where $p(\overline{\mathbf{x}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{x}}) \mid \ldots \mid R_m(\overline{\mathbf{x}})$. It can easily be shown that $\mathbb{F}^{\mathsf{U}}_{\mathcal{S}}$ is monotone and continuous, just as we previously have shown for $\mathbb{F}^{\mathsf{SL}}_{\mathcal{S}}$ in Theorems 1.3.7 and 1.3.9. These proofs are enabled by introducing an operator $\mathcal{T}_k$ with $k \in \mathbb{N}$ such that, given sets $A \in \mathsf{L}^n \times \mathsf{Heaps}, B_1 \in \mathsf{L}^{n_1} \times \mathsf{Heaps} \times \mathsf{Cover}, \ldots, B_k \in \mathsf{L}^{n_k} \times \mathsf{Heaps} \times \mathsf{Cover}$, we have $\mathcal{T}_k(A, B_1, \ldots, B_k) = \{(\overline{\ell} \cdot \overline{\ell}_1 \cdot \ldots \cdot \overline{\ell}_k, h \uplus h_1 \uplus \ldots \uplus h_k, \tau_k(h, t_1, \ldots, t_k)) \mid (\overline{\ell}, h) \in A, (\overline{\ell}_i, h_i, t_i) \in B_i, i \in [k]\}$. We similarly use it for sets $A \in \mathcal{V} \times \mathsf{Heaps}, B_1 \in \mathcal{V} \times \mathsf{Heaps} \times \mathsf{Cover}, \ldots, B_k \in \mathcal{V} \times \mathsf{Heaps} \times \mathsf{Cover}$.

A solution of the SL inductive system $\mathcal{S}$ is now an assignment $\mathcal{X} \in \mathsf{Assign}^{\mathsf{U}}(\mathcal{S}^{\mathsf{P}})$ such that $\mathbb{F}^{\mathsf{U}}_{\mathcal{S}}(\mathcal{X}) \preccurlyeq \mathcal{X}$. The set of all solutions of $\mathcal{S}$ is $\mathsf{Sol}^{\mathsf{U}}_{\mathcal{S}} = \{\mathcal{X} \mid \mathbb{F}^{\mathsf{U}}_{\mathcal{S}}(\mathcal{X}) \preccurlyeq \mathcal{X}\}$. A least solution of $\mathcal{S}$ is $\mu\mathcal{S}^{\mathsf{U}} \in \mathsf{Sol}^{\mathsf{U}}_{\mathcal{S}}$ such that, for any assignment $\mathcal{X} \prec \mu\mathcal{S}^{\mathsf{U}}$, $\mathcal{X} \notin \mathsf{Sol}^{\mathsf{U}}_{\mathcal{S}}$. Similar to Theorem 1.3.10, it can be shown that $\mu\mathcal{S}^{\mathsf{U}}$ is unique and $\mu\mathcal{S}^{\mathsf{U}} = {\mathbb{F}^{\mathsf{U}}_{\mathcal{S}}}^{n}(\mathcal{X}_\emptyset)$, where $n \in \mathbb{N}$ is the smallest value for which ${\mathbb{F}^{\mathsf{U}}_{\mathcal{S}}}^{n+1}(\mathcal{X}_\emptyset) = {\mathbb{F}^{\mathsf{U}}_{\mathcal{S}}}^{n}(\mathcal{X}_\emptyset)$.

For each $(\nu, h, t) \in \mu\mathcal{S}^{\mathsf{U}}(p)$, for a predicate $p \in \mathcal{S}^{\mathsf{P}}$, we call t an *unfolding tree* of p. Given $p, q_1, \ldots, q_n \in \mathcal{S}^{\mathsf{P}}$, the entailment problem now becomes $p(\overline{\mathbf{x}}) \models^{\mathsf{U}}_{\mathcal{S}} q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ if and only if $\mu\mathcal{S}^{\mathsf{U}}(p) \subseteq \mu\mathcal{S}^{\mathsf{U}}(q_1) \cup \ldots \cup \mu\mathcal{S}^{\mathsf{U}}(q_n)$. Note that $p(\overline{\mathbf{x}}) \models^{\mathsf{U}}_{\mathcal{S}} q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ implies $p(\overline{\mathbf{x}}) \models^{\mathsf{SL}}_{\mathcal{S}} q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$, but not the other way around.

Example 2.5.1. Consider the following SL inductive system $\mathcal{S}$:

$$ls_1(x, y) \leftarrow_{\mathcal{S}} x \approx y \wedge \mathsf{emp} \mid y \approx y' \wedge x \mapsto x', ls_1(x', y')$$
$$ls_2(x, y) \leftarrow_{\mathcal{S}} x \approx y \wedge \mathsf{emp} \mid y \approx y' \wedge x \mapsto x' * x' \mapsto z', ls_2(z', y')$$

The predicates $ls_1(x, y)$ and $ls_2(x, y)$ both define list segments, but the latter only covers list segments of even length and unfolds them two elements at a time. The entailment $ls_2 \models^{\mathsf{SL}}_{\mathcal{S}} ls_1$ holds. Given $(\overline{\ell}, h) = (\langle 1, 5\rangle, \{(1, 2), (2, 3), (3, 4), (4, 5)\})$, clearly $(\overline{\ell}, h) \in \mu\mathcal{S}^{\mathsf{SL}}(ls_1)$ and $(\overline{\ell}, h) \in \mu\mathcal{S}^{\mathsf{SL}}(ls_2)$. However, the two definitions generate different coverage trees t_1 and t_2 for h, such that $(\overline{\ell}, h, t_1) \in \mu\mathcal{S}^{\mathsf{U}}(ls_1)$ and $(\overline{\ell}, h, t_2) \in \mu\mathcal{S}^{\mathsf{U}}(ls_2)$:

$$t_1 = \begin{array}{c} \{(1,2)\} \\ | \\ \{(2,3)\} \\ | \\ \{(3,4)\} \\ | \\ \{(4,5)\} \\ | \\ \emptyset \end{array} \qquad t_2 = \begin{array}{c} \{(1,2),(2,3)\} \\ | \\ \{(3,4),(4,5)\} \\ | \\ \emptyset \end{array}$$

It follows that $\mu\mathcal{S}^{\mathsf{U}}(ls_2) \not\subseteq \mu\mathcal{S}^{\mathsf{U}}(ls_1)$ and the entailment $ls_2 \models^{\mathsf{U}}_{\mathcal{S}} ls_1$ does not hold.

On a different vein, consider the SL inductive system $\mathcal{S}'$:

$$\begin{array}{rcl} dlls(hd, p, tl, n) & \leftarrow_{\mathcal{S}'} & hd \approx tl \wedge hd \mapsto (p, n) \\ & \mid & p' \approx hd \wedge tl' \approx tl \wedge n' \approx n \wedge hd \mapsto (p, hd'), dlls(hd', p', tl', n') \\ dlls^r(hd, p, tl, n) & \leftarrow_{\mathcal{S}'} & hd \approx tl \wedge hd \mapsto (p, n) \\ & \mid & hd' \approx hd \wedge p' \approx p \wedge n' \approx tl \wedge tl \mapsto (n, tl'), dlls^r(hd', p', tl', n') \end{array}$$

Both predicates define doubly-linked list segments, but $dlls$ unfolds them starting at the head, while $dlls^r$ unfolds starting at the tail. The entailments $dlls \models^{\mathsf{SL}}_{\mathcal{S}'} dlls^r$ and $dlls^r \models^{\mathsf{SL}}_{\mathcal{S}'} dlls$ hold.

Let $(\overline{\ell}', h') = (\langle 0, 1, 4, 5\rangle, \{(1, \langle 0, 2\rangle), (2, \langle 1, 3\rangle), (3, \langle 2, 4\rangle), (4, \langle 3, 5\rangle)\})$. It is easy to see that $(\overline{\ell}', h') \in \mu\mathcal{S}^{\mathsf{SL}}(dlls)$ and $(\overline{\ell}', h') \in \mu\mathcal{S}^{\mathsf{SL}}(dlls^r)$. Again, the two definitions generate different coverage trees t'_1 and t'_2 for h', such that $(\overline{\ell}', h', t'_1) \in \mu\mathcal{S}^{\mathsf{U}}(dlls)$ and $(\overline{\ell}', h', t'_2) \in \mu\mathcal{S}^{\mathsf{U}}(dlls^r)$:

$$t'_1 \;=\; \begin{array}{c}\{(1,\langle 0,2\rangle)\}\\ |\\ \{(2,\langle 1,3\rangle)\}\\ |\\ \{(3,\langle 2,4\rangle)\}\\ |\\ \{(4,\langle 3,5\rangle)\}\end{array} \qquad t'_2 \;=\; \begin{array}{c}\{(4,\langle 3,5\rangle)\}\\ |\\ \{(3,\langle 2,4\rangle)\}\\ |\\ \{(2,\langle 1,3\rangle)\}\\ |\\ \{(1,\langle 0,2\rangle)\}\end{array}$$

It follows that $\mu\mathcal{S}^{\mathsf{U}}(dlls) \not\subseteq \mu\mathcal{S}^{\mathsf{U}}(dlls^r)$ and $\mu\mathcal{S}^{\mathsf{U}}(dlls^r) \not\subseteq \mu\mathcal{S}^{\mathsf{U}}(dlls)$. The corresponding entailments $dlls \models^{\mathsf{U}}_{\mathcal{S}'} dlls^r$ and $dlls^r \models^{\mathsf{U}}_{\mathcal{S}'} dlls$ do not hold. ◀

The definitions for maximal and irreducible derivations using $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ are the same as their $\mathcal{R}_{\mathsf{Ind}}$ counterparts in Section 2.5.1. Similarly, a derivation built with $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ is structured if and only if $\mathrm{RD}_{\mathsf{SL}}$ occurs between any two consecutive applications of LU. We use $\mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta)$ for the set of irreducible, maximal and structured having $\Gamma \vdash \Delta$ as a root and using $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$. We also write $\Gamma \vdash \Delta \rightsquigarrow^{\mathsf{SL}} \Gamma' \vdash \Delta'$ if $\Gamma' \vdash \Delta'$ occurs as a node in a derivation from $\mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta)$.

The following lemmas are the SL counterparts of the ones in Section 2.5.1. Since $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ is structurally the same as $\mathcal{R}_{\mathsf{Ind}}$, we need not redo the proofs, as they closely resemble the ones for Lemma 2.5.5 and Lemma 2.5.6.

Lemma 2.5.13. If the SL inductive system $\mathcal{S}$ has the fvi property, then the following properties of derivations built with $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ hold:

(1) Any irreducible and structured derivation is finite;

(2) For any sequent $\Gamma \vdash \Delta$, the set $\mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta)$ is finite.

Lemma 2.5.14. Given an SL inductive system $\mathcal{S}$ and predicates $p, q_1, \ldots, q_n \in \mathcal{S}^{\mathsf{P}}$, in every sequent $\Gamma \vdash \Delta$ such that $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow^{\mathsf{SL}} \Gamma \vdash \Delta$, Γ is a tree-shaped set and Δ consists of finite separating conjunctions over tree-shaped sets, with all subgoal variables existentially quantified.

The following lemma is the counterpart of Lemma 2.5.7, providing a similar invariant on the set of derivations for a sequent that denotes an invalid entailment. As in the FOL case, this result is important for showing the completeness of $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ under the new interpretation of entailments, based on unfolding trees.

Lemma 2.5.15. Given a non-filtering and non-overlapping SL inductive system with the fvi property, the predicates $p, q_1, \ldots, q_n \in \mathcal{S}^{\mathsf{P}}$, and a sequent $\Gamma \vdash \Delta$ such that $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow^{\mathsf{SL}} \Gamma \vdash \Delta$, if every derivation $D \in \mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta)$ contains a leaf $\Gamma' \vdash \emptyset$ then there exists a valuation ν, a heap h and a set of unfolding trees U such that $(\nu, h, U) \in \mu\mathcal{S}^{\mathsf{U}}(\ast\,\Gamma) \setminus \mu\mathcal{S}^{\mathsf{U}}(\bigvee \Delta)$.

Proof. Let $(\mathfrak{D}^{\mathsf{SL}})^*(\Gamma \vdash \Delta) = \{d \mid d \in \mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta) \text{ or } \exists d' \in (\mathfrak{D}^{\mathsf{SL}})^*(\Gamma \vdash \Delta)\,.\, d \sqsubset d'\}$. Because, by Lemma 2.5.13, any derivation in $\mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta)$ is finite and $\mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta)$ itself is finite, then $(\mathfrak{D}^{\mathsf{SL}})^*(\Gamma \vdash \Delta)$ is also finite. Since $((\mathfrak{D}^{\mathsf{SL}})^*(\Gamma \vdash \Delta), \sqsubseteq)$ is a wqo, by Lemma 2.4.1 $(\mathcal{P}_{fin}((\mathfrak{D}^{\mathsf{SL}})^*(\Gamma \vdash \Delta)), \sqsubseteq^{\forall\exists})$ is also a wqo and we can prove this lemma by induction on it.

Let $\mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta) = \bigcup_{IR \in \mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}} \mathfrak{D}^{\mathsf{SL}}_{IR}(\Gamma \vdash \Delta)$, where $\mathfrak{D}^{\mathsf{SL}}_{IR}(\Gamma \vdash \Delta)$ denotes the subset of $\mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \Delta)$ consisting of derivations starting with an instance of the inference rule IR. We do not consider the cases $IR \in \{\mathrm{AX}_{\mathsf{SL}}, \mathrm{ID}\}$ because these inference rules do not lead to a leaf of the required form. Given $D \in \mathfrak{D}^{\mathsf{SL}}_{IR}(\Gamma \vdash \Delta)$ with $IR \in \mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}} \setminus \{\mathrm{AX}_{\mathsf{SL}}, \mathrm{ID}\}$, we assume that, for each antecedent $\Gamma' \vdash \Delta'$ of $\Gamma \vdash \Delta$, the lemma holds for any $D' \in \bigcup_{IR' \in \mathcal{R}_{\mathsf{Ind}}} \mathfrak{D}^{\mathsf{SL}}_{IR'}(\Gamma' \vdash \Delta')$ and show that it extends to D.

(LU) Let $r(\overline{\mathbf{y}}) \in \Gamma$ be the predicate atom chosen for replacement and $r(\overline{\mathbf{y}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{y}}) \mid \ldots \mid R_m(\overline{\mathbf{y}})$. Then $R_1(\overline{\mathbf{y}}, \overline{\mathbf{y}}_1), \Gamma \setminus r(\overline{\mathbf{y}}) \vdash \Delta, \ldots, R_m(\overline{\mathbf{y}}, \overline{\mathbf{y}}_m), \Gamma \setminus r(\overline{\mathbf{y}}) \vdash \Delta$ are the antecedents for the application of $IR = \mathrm{LU}$ on $\Gamma \vdash \Delta$. If $\Delta = \emptyset$, then, similarly to the LU case in Lemma 2.5.7, there exists a counterexample $\nu \in \mu\mathcal{S}^{\mathsf{U}}(* \Gamma) \setminus \mu\mathcal{S}^{\mathsf{U}}(\bigvee \Delta) = \mu\mathcal{S}^{\mathsf{U}}(* \Gamma)$.

If $\Delta \neq \emptyset$, suppose that, for each $i \in [m]$, there exists $D_i \in \mathfrak{D}^{\mathsf{SL}}(R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i), \Gamma \setminus r(\overline{\mathbf{y}}) \vdash \Delta)$ not containing any leaf $\Gamma' \vdash \emptyset$. Then we can choose these derivations and create one for $\Gamma \vdash \Delta$ with the same property, which contradicts the hypothesis of the lemma. Therefore, there must exist $i \in [m]$ such that every derivation in $\mathfrak{D}^{\mathsf{SL}}(R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i), \Gamma \setminus r(\overline{\mathbf{y}}) \vdash \Delta)$ contains a leaf $\Gamma' \vdash \emptyset$.

Because $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow^{\mathsf{SL}} \Gamma \vdash \Delta$, Γ is a tree-shaped set and let $T_1, \ldots, T_k$ be the singly-tree shaped sets, represented as trees labelled with formulae, such that $\Gamma = \bigcup_{i=1}^{k} T_i$. Then there exists a tree T_j, $j \in [k]$ and a frontier position $\alpha_r \in \mathrm{fr}(T_j)$ such that $T_j(\alpha_r) = r(\overline{\mathbf{y}})$. Then the tree-shaped set $R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i), \Gamma \setminus r(\overline{\mathbf{y}})$ is represented by the singly-tree shaped sets $T'_1, \ldots, T'_k$, where $T'_\ell = T_\ell$ for all $\ell \in [k] \setminus \{j\}$ and $T'_j = T_{j_{[\alpha_r]}} \circ R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i)$.

By the induction hypothesis, there exists a counterexample $(\nu, h_1 \uplus \ldots \uplus h_k, \{t_1, \ldots, t_k\}) \in \mu\mathcal{S}^{\mathsf{U}}(*(R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i) \cup \Gamma \setminus \{r(\overline{\mathbf{y}})\})) \setminus \mu\mathcal{S}^{\mathsf{U}}(\bigvee \Delta)$, where $(\nu, h_\ell, t_\ell) \in \mu\mathcal{S}^{\mathsf{U}}(* T'_\ell)$, for all $\ell \in [k]$. Because $T'_\ell = T_\ell$ for all $\ell \in [k] \setminus \{j\}$, we only need to show that $(\nu, h_j, t_j) \in \mu\mathcal{S}^{\mathsf{U}}(* T_j)$. Since $(\nu, h_j, t_j) \in \mu\mathcal{S}^{\mathsf{U}}(*(T_{j_{[\alpha_r]}} \circ R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i)))$, there exist some disjoint heaps h'_j and h''_j, a context $t'_{j_{[\alpha_r]}}$ and a cover t''_j such that: (i) $h_j = h'_j \uplus h''_j$ and $t_j = t'_{j_{[\alpha_r]}} \circ t''_j$, (ii) $t'_{j_{[\alpha_r]}}$ covers h'_j and $(\nu, t''_j, h''_j) \in \mu\mathcal{S}^{\mathsf{U}}(* R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i))$. Since $\langle r(\overline{\mathbf{y}}), R_i(\overline{\mathbf{y}}) \rangle \in \mathcal{S}$ is a predicate rule, we have $\mu\mathcal{S}^{\mathsf{U}}(* R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i)) \subseteq \mu\mathcal{S}^{\mathsf{U}}(r(\overline{\mathbf{y}}))$. This leads to $(\nu, t''_j, h''_j) \in \mu\mathcal{S}^{\mathsf{U}}(r(\overline{\mathbf{y}}))$ and $(\nu, h_j, t_j) \in \mu\mathcal{S}^{\mathsf{U}}(*(T_{j_{[\alpha_r]}} \circ R_i(\overline{\mathbf{y}}, \overline{\mathbf{y}}_i))) \subseteq \mu\mathcal{S}^{\mathsf{U}}(*(T_{j_{[\alpha_r]}} \circ r(\overline{\mathbf{y}}))) = \mu\mathcal{S}^{\mathsf{U}}(* T_j)$. Therefore, $(\nu, h_\ell, t_\ell) \in \mu\mathcal{S}^{\mathsf{U}}(* T_\ell)$ for all $\ell \in [k]$ and $(\nu, h_1 \uplus \ldots \uplus h_k, \{t_1, \ldots, t_k\}) \in \mu\mathcal{S}^{\mathsf{U}}(* \Gamma) \setminus \mu\mathcal{S}^{\mathsf{U}}(\bigvee \Delta)$.

($\mathrm{RU}_{\mathsf{SL}}$) Let $r(\overline{\mathbf{y}}) \in \Delta$ be the predicate atom chosen for replacement and $r(\overline{\mathbf{y}}) \leftarrow_{\mathcal{S}} R_1(\overline{\mathbf{y}}) \mid \ldots \mid R_m(\overline{\mathbf{y}})$. Then $\Gamma \vdash \exists \overline{\mathbf{z}}_1. * R_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1), \ldots, \exists \overline{\mathbf{z}}_m. * R_m(\overline{\mathbf{y}}, \overline{\mathbf{z}}_m), \Delta \setminus r(\overline{\mathbf{y}})$ is the antecedent for the application of $IR = \mathrm{RU}_{\mathsf{SL}}$ on $\Gamma \vdash \Delta$. Since RU is applicable, $\Delta \neq \emptyset$. Also, the right-hand side of the antecedent cannot be $\emptyset$ because all predicates in $\mathcal{S}^{\mathsf{p}}$ must be goals for at least one predicate rule of $\mathcal{S}$.

Suppose that there exists $D \in \mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \exists \overline{\mathbf{z}}_1. * R_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1), \ldots, \exists \overline{\mathbf{z}}_m. * R_m(\overline{\mathbf{y}}, \overline{\mathbf{z}}_m), \Delta \setminus r(\overline{\mathbf{y}}))$ not containing any leaf $\Gamma' \vdash \emptyset$. Then we can choose D and create a derivation for $\Gamma \vdash \Delta$ with the same property, which contradicts the hypothesis of the lemma. Therefore, it must be the case that every derivation in $\mathfrak{D}^{\mathsf{SL}}(\Gamma \vdash \exists \overline{\mathbf{z}}_1. * R_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1), \ldots, \exists \overline{\mathbf{z}}_m. * R_m(\overline{\mathbf{y}}, \overline{\mathbf{z}}_m), \Delta \setminus r(\overline{\mathbf{y}}))$ contains a leaf $\Gamma' \vdash \emptyset$.

By the induction hypothesis, there exists $(\nu, h, U) \in \mu\mathcal{S}^{\mathsf{U}}(* \Gamma) \setminus \mu\mathcal{S}^{\mathsf{U}}(\bigvee_{i=1}^{m} \exists \overline{\mathbf{z}}_i. * R_i(\overline{\mathbf{y}}, \overline{\mathbf{z}}_i) \vee \bigvee(\Delta \setminus \{r(\overline{\mathbf{y}})}))$. Then it is also the case that $(\nu, h, U) \in \mu\mathcal{S}^{\mathsf{U}}(* \Gamma) \setminus \mu\mathcal{S}^{\mathsf{U}}(\bigvee \Delta)$, because $\mu\mathcal{S}^{\mathsf{U}}(\bigvee \Delta) = \mu\mathcal{S}^{\mathsf{U}}(\bigvee_{i=1}^{m} \exists \overline{\mathbf{z}}_i. * R_i(\overline{\mathbf{y}}, \overline{\mathbf{z}}_i) \vee \bigvee(\Delta \setminus \{r(\overline{\mathbf{y}})}))$ by a similar argument as in the ($\mathrm{RU}_{\mathsf{SL}}$) case of the proof for Theorem 2.5.10.

($\mathrm{RD}_{\mathsf{SL}}$) Let $\Gamma = \{\phi(\overline{\mathbf{y}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_n), r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m)\}$ and $\Delta = \{\exists \overline{\mathbf{z}}_1.\psi_1(\overline{\mathbf{y}}, \overline{\mathbf{z}}_1) * \mathcal{Q}_1(\overline{\mathbf{z}}_1), \ldots, \exists \overline{\mathbf{z}}_k.\psi_k(\overline{\mathbf{y}}, \overline{\mathbf{z}}_k) * \mathcal{Q}_k(\overline{\mathbf{y}}_k)\}$, where $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ are separating conjunctions of predicate atoms. Also, let $S_j = \mathrm{VIS}(\phi, \psi_j)$, for all $j \in [k]$. Each S_j is finite because $\mathcal{S}$ has the fvi property.

If $m = 0$, the proof is similar to the one in the (RD) case of Lemma 2.5.7.

If $m > 0$, then $r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m) \vdash \{\mathcal{Q}_j\theta \mid \theta \in S_j\}_{j=1}^{i}$ is the antecedent for the application of $IR = \mathrm{RD}_{\mathsf{SL}}$ on $\Gamma \vdash \Delta$, where, by the side condition, $\phi \models^{\cup} \bigwedge_{j=1}^{i} \exists \overline{\mathbf{y}}_j.\psi_j$ and $\phi \not\models^{\cup} \bigvee_{j=i+1}^{k} \exists \overline{\mathbf{y}}_j.\psi_j$ (with a possible reordering of Δ). If $k = 0$, because $\mathcal{S}$ is non-filtering, $\overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m$ are distinct and there are no predicates in $\mathcal{S}^{\mathsf{P}}$ with empty least solutions, it follows that $\mu\mathcal{S}^{\cup}(* \Gamma)$ is not empty. Therefore there exists a counterexample $(\nu, h, t) \in \mu\mathcal{S}^{\cup}(* \Gamma) \setminus \mu\mathcal{S}^{\cup}(\bigvee \Delta) = \mu\mathcal{S}^{\cup}(* \Gamma)$. Similarly, if $i = 0$, we have that $\mu\mathcal{S}^{\cup}(r_1(\overline{\mathbf{y}}_1) * \ldots * r_m(\overline{\mathbf{y}}_m))$ is not empty and it contains counterexamples for the antecedent. Given $(\nu, h, \{t_1, \ldots, t_m\}) \in \mu\mathcal{S}^{\cup}(r_1(\overline{\mathbf{y}}_1) * \ldots * r_m(\overline{\mathbf{y}}_m))$, because the system is non-filtering, there exist a pair $(\overline{\ell}, h_0) \in \mathsf{L}^{\|\overline{\mathbf{y}}\|} \times \mathsf{Heaps}$ such that h_0 is disjoint from h and $\nu[\overline{\mathbf{y}} \leftarrow \overline{\ell}], h_0 \models^{\mathsf{SL}} \phi$. By the side condition of RD and the fact that $\mathcal{S}$ is non-overlapping, we have that $\nu[\overline{\mathbf{y}} \leftarrow \overline{\ell}], h \not\models^{\mathsf{SL}} \exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{z}}_j), \forall j \in [k]$. We obtain $(\nu[\overline{\mathbf{y}} \leftarrow \overline{\ell}], h_0 \uplus h, \tau_m(h_0, t_1, \ldots, t_m)) \notin \mu\mathcal{S}^{\cup}(\exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{y}}, \overline{\mathbf{z}}_j) * \mathcal{Q}_j(\overline{\mathbf{z}}_j)), \forall j \in [k]$, thus $(\nu[\overline{\mathbf{y}} \leftarrow \overline{\ell}], h_0 \uplus h, \tau_m(h_0, t_1, \ldots, t_m)) \in \mu\mathcal{S}^{\cup}(* \Gamma) \setminus \mu\mathcal{S}^{\cup}(\bigvee \Delta)$.

If both $k > 0$ and $i > 0$, suppose that there exists $D \in \mathfrak{D}^{\mathsf{SL}}(r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m) \vdash \{\mathcal{Q}_j\theta \mid \theta \in S_j\}_{j=1}^{i})$ not containing any leaf $\Gamma' \vdash \emptyset$. Then we can choose D and create a derivation for $\Gamma \vdash \Delta$ with the same property, which contradicts the hypothesis of the lemma. Thus, it must be the case that every derivation in $\mathfrak{D}^{\mathsf{SL}}(p_1(\overline{\mathbf{x}}_1), \ldots, p_n(\overline{\mathbf{x}}_n) \vdash \{\mathcal{Q}_j\theta \mid \theta \in S_j\}_{j=1}^{i}$ contains a leaf $\Gamma' \vdash \emptyset$.

By the induction hypothesis, there exists $(\nu, h, \{t_1, \ldots, t_m\}) \in \mu\mathcal{S}^{\cup}(r_1(\overline{\mathbf{y}}_1) * \ldots * r_m(\overline{\mathbf{y}}_m)) \setminus \mu\mathcal{S}^{\cup}\left(\bigvee_{j=1}^{i}\{\mathcal{Q}_j\theta \mid \theta \in S_j\}\right)$, and also, for every $j \in [i]$ and $\theta \in S_j$, $(\nu, h, \{t_1, \ldots, t_m\}) \in \mu\mathcal{S}^{\cup}(r_1(\overline{\mathbf{y}}_1) * \ldots * r_m(\overline{\mathbf{y}}_m)) \setminus \mu\mathcal{S}^{\cup}(\mathcal{Q}_j\theta)$. Then $h = \biguplus_{j=1}^{m} h_j$, where $(\nu(\overline{\mathbf{y}}_j), h_j, t_j) \in \mu\mathcal{S}^{\cup}(r_j)$, for all $j \in [m]$. Because $\mathcal{S}$ is non-filtering, there exists a pair $(\overline{\ell}, h_0) \in \mathsf{L}^{\|\overline{\mathbf{y}}\|} \times \mathsf{Heaps}$ such that h_0 is disjoint from h and $\nu[\overline{\mathbf{y}} \leftarrow \overline{\ell}], h_0 \models^{\mathsf{SL}} \phi$. Because $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}}) \rightsquigarrow^{\mathsf{SL}} \Gamma \vdash \Delta$, by Lemma 2.5.6 the set $\Gamma = \{\phi(\overline{\mathbf{y}}, \overline{\mathbf{y}}_1, \ldots, \overline{\mathbf{y}}_m), r_1(\overline{\mathbf{y}}_1), \ldots, r_m(\overline{\mathbf{y}}_m)\}$ is tree-shaped, and because the number of predicate atoms equals the number of tuples of subgoal variables, it must be a singly-tree shaped set. Then $(\nu[\overline{\mathbf{y}} \leftarrow \overline{\ell}], h_0 \uplus h, \tau_n(h_0, t_1, \ldots, t_n)) \in \mu\mathcal{S}^{\cup}(* \Gamma)$.

Suppose that $(\nu[\overline{\mathbf{x}} \leftarrow \overline{\ell}], h_0 \uplus h, \tau_n(h_0, t_1, \ldots, t_m))$ is a model of $\exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{x}}, \overline{\mathbf{z}}_j) * \mathcal{Q}_j(\overline{\mathbf{z}}_j)$ for some $j \in [k]$. Then $\nu[\overline{\mathbf{x}} \leftarrow \overline{\ell}], h_0 \models^{\mathsf{SL}} \exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{x}}, \overline{\mathbf{z}}_j)$ and, because $\mathcal{S}$ is non-overlapping, we have that $\phi \models^{\mathsf{SL}} \exists \overline{\mathbf{z}}_j.\psi_j$. It follows that $j \notin [i+1, k]$, otherwise it would contradict the side condition $\phi \not\models^{\mathsf{SL}} \exists \overline{\mathbf{z}}_{i+1}.\psi_{i+1} \vee \ldots \vee \exists \overline{\mathbf{z}}_k.\psi_k$. Therefore, $j \in [i]$ and, because $\mathcal{S}$ has the fvi property, $\nu[\overline{\mathbf{x}} \leftarrow \overline{\ell}], h_0 \models^{\mathsf{SL}} \psi_j\theta$, for all $\theta \in \mathrm{VIS}(\phi, \psi_j)$. Moreover, there is no other Skolem function that witnesses this entailment, besides the ones in S_j. Then also $(\nu[\overline{\mathbf{x}} \leftarrow \overline{\ell}], h, \{t_1, \ldots, t_n\}) \in \mu\mathcal{S}^{\cup}(\mathcal{Q}_j\theta)$, for all $\theta \in S_j$ and only for these substitutions. Because the range of each $\theta \in S_j$ is $\overline{\mathbf{y}}_1 \cup \ldots \cup \overline{\mathbf{y}}_m$, it must be the case that $(\nu, h, \{t_1, \ldots, t_m\}) \in \mu\mathcal{S}^{\cup}(\mathcal{Q}_j\theta)$ for all $\theta \in S_j$, which contradicts our assumption that this is a counterexample of the antecedent. It follows that $(\nu[\overline{\mathbf{x}} \leftarrow \overline{\ell}], h_0 \uplus h, \tau_n(h_0, t_1, \ldots, t_m))$ cannot be a model of $\exists \overline{\mathbf{z}}_j.\psi_j(\overline{\mathbf{x}}, \overline{\mathbf{z}}_j) \wedge \mathcal{Q}_j(\overline{\mathbf{z}}_j)$ for any $j \in [k]$ and $(\nu[\overline{\mathbf{x}} \leftarrow \overline{\ell}], h_0 \uplus h, \tau_n(h_0, t_1, \ldots, t_n)) \in \mu\mathcal{S}^{\cup}(* \Gamma) \setminus \mu\mathcal{S}^{\cup}(\bigvee \Delta)$, as required.

($\wedge$R) Similar to the ($\wedge$R) case in the proof of Lemma 2.5.7.

($\mathrm{SP}_{\mathsf{SL}}$) Similar to the (SP) case in the proof of Lemma 2.5.7, using a variation of Lemma 2.5.9 in which $\mathcal{U}_k = \mathsf{L}^k \times \mathsf{Heaps} \times \mathsf{Cover}$ and $\biguplus_{i=1}^{n} \mu\mathcal{S}^{\cup}(p_i) \stackrel{\text{def}}{=} \{(\overline{\ell}_1 \cdot \ldots \cdot \overline{\ell}_n, h_1 \uplus \ldots \uplus h_n, \{t_1, \ldots, t_n\}) \mid (\overline{\ell}_i, h_i, t_i) \in \mu\mathcal{S}^{\cup}(p_i), i \in [n]\}$, for some predicates $p_1, \ldots, p_n \in \mathcal{S}^{\mathsf{P}}$. $\square$

The following theorem establishes the completeness of $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ under the unfolding trees interpretation of entailments, while also providing a proof search strategy $\mathbf{S}$ that is similar to its FOL counterpart.

Theorem 2.5.16. Given a ranked, non-filtering, non-overlapping SL inductive system $\mathcal{S}$ with the fvi property, let $p, q_1, \ldots, q_n \in \mathcal{S}^{\mathsf{P}}$. Then the entailment $p \models^{\mathsf{U}}_{\mathcal{S}} q_1, \ldots, q_n$ holds only if $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ has an $\mathbf{S}$-proof with the inference rule schemata $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$, where $\mathbf{S}$ is defined by the regular expression $(\mathrm{LU} \cdot \mathrm{RU}^*_{\mathsf{SL}} \cdot \mathrm{RD}_{\mathsf{SL}} \cdot \wedge\mathrm{R}^* \cdot \mathrm{SP}_{\mathsf{SL}}?)^* \cdot \mathrm{LU}? \cdot \mathrm{RU}^*_{\mathsf{SL}} \cdot (\mathrm{AX}_{\mathsf{SL}} \mid \mathrm{ID})$.

Proof. Since the proof systems $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ and $\mathcal{R}_{\mathsf{Ind}}$ are structurally the same, only replacing the classical conjunction with the separating conjunction in $\mathrm{RU}_{\mathsf{SL}}$, $\mathrm{RD}_{\mathsf{SL}}$, $\mathrm{SP}_{\mathsf{SL}}$ and $\mathrm{AX}_{\mathsf{SL}}$, the completeness proof for SL inductive systems closely mirrors the proof of Theorem 2.5.8. □

As in the FOL case, the fact that each $\mathbf{S}$-derivation is irreducible and structured allows the proof search semi-algorithm 1 to terminate on all inputs, thus becoming a decision procedure for entailments $p \models^{\mathsf{U}}_{\mathcal{S}} q_1, \ldots, q_n$, where $p, q_1, \ldots, q_n \in \mathcal{S}^{\mathsf{P}}$ and the SL inductive system $\mathcal{S}$ is ranked, non-filtering, non-overlapping and has the fvi property.

Chapter 3

Decision procedures for Separation Logic in SMT

In this chapter, we provide decision procedures for the satisfiability problem of the quantifier-free (Section 3.2) and $\exists^*\forall^*$-quantified (Section 3.3) fragments of $\mathsf{SL}(T)$ parametrized by a theory of heap locations and data, chosen from the wide array of theories handled by SMT solvers, such as integer or real arithmetic, strings, sets or uninterpreted functions. This is a joint work with Andrew Reynolds (University of Iowa) and Tim King (Google Inc.) [42, 41].

Both procedures are designed as components of a DPLL(T) architecture, widely used by modern SMT solvers, and implemented as subsolvers of CVC4 [3], which, as indicated by our experimental evaluation, is able to handle non-trivial examples quite effectively. Practical uses of our procedures, which guided our choice for the evaluation benchmarks, include:

- Integration with theorem provers for SL with inductively defined predicates. Most inductive provers for SL (including our $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ from Section 2.3.2) use a higher level proof search strategy relying on separate decision procedures for non-inductive entailments to simplify proof obligations [8]. The difficulty posed by these entailments in the general fragment of SL usually leads to only using symbolic heap formulae, for which entailments are proved by syntactic substitutions and matching. It is thus possible to extend the language of inductive SL solvers and allow our specialized procedure to tackle a more expressive range of non-inductive entailments. To justify this use case, a significant portion of our experiments focuses on entailments between finite unfoldings of inductive predicates commonly used in practice.
- Back-end component in a bounded model checker for programs with pointer and data manipulations, using a complete weakest precondition calculus involving the magic wand such as the one described in [29]. This led us to test our procedures on verification conditions automatically generated by applying the weakest precondition calculus of [29] to several program fragments that manipulate lists, with or without data fields.

3.1 The $\mathsf{SL}(T)$-satisfiability problem

We tackle the $\mathsf{SL}(T)$-satisfiability problem, as defined below, under the assumption that the quantifier-free theory $T = (\Sigma, \mathbf{M})$ has a decidable satisfiability problem.

Definition 3.1.1 (T-satisfiability problem)**.** Given a theory $T = (\Sigma, \mathbf{M})$ and a T-formula ϕ, is $[\![\phi]\!]_T \neq \emptyset$ (i.e. does ϕ have a T-model)?

Definition 3.1.2 ($\mathsf{SL}(T)$-satisfiability problem)**.** Given a theory $T = (\Sigma, \mathbf{M})$ and an $\mathsf{SL}(T)$-formula ϕ, is $[\![\phi]\!]_{\mathsf{SL}(T)} \neq \emptyset$ (i.e. does ϕ have an $\mathsf{SL}(T)$-model)?

It has been proved by Calcagno et al. [11] that the satisfiability problem is PSPACE-complete for the $\mathsf{SL}(T)$ fragment in which $\mathsf{Data} = \mathsf{Loc} \times \mathsf{Loc}$. We try to generalize this result and consider any first-order theory T whose quantifier-free fragment has a satisfiability problem in PSPACE. In general, this is the case for most theories supported by SMT solvers, which are typically in NP (e.g. linear arithmetic of integers or reals, possibly with sets and uninterpreted functions).

3.2 The Quantifier-Free Fragment of $\mathsf{SL}(T)$

We show that the satisfiability problem for the quantifier-free fragment of $\mathsf{SL}(T)$ is PSPACE-complete if the satisfiability problem of the base theory T is in PSPACE. A semantics-preserving translation is used to transform $\mathsf{SL}(T)$-formulae into second-order T-formulae with quantifiers that range over a domain of sets and uninterpreted functions, where the cardinality of these sets is polynomially bound by the size of the input formula. We then introduce a method for lazy, counterexample-driven quantifier instantiation, tailored specifically for this fragment of T-formulae translated from $\mathsf{SL}(T)$, and show that it is sound, complete and terminating.

3.2.1 Reducing $\mathsf{SL}(T)$ to Multisorted Second-Order Logic

Due to the behaviour of the $*$ and $-\!\!*$ connectives, separation logic cannot be formalized as a classical (unsorted) first-order theory. The separating conjunction does not comply with the standard rules of contraction and weakening, meaning that $\phi \Rightarrow \phi * \phi$ and $\phi * \psi \Rightarrow \phi$ do not hold because $*$ requires its operands to hold on disjoint heaps. Similarly, $\phi -\!\!* \psi$ holds on a heap that, when extended by a disjoint heap satisfying ϕ, must satisfy ψ. We capitalize on the expressiveness of multi-sorted first-order theories by translating $\mathsf{SL}(T)$-formulae into quantified T-formulae, under the assumption that T subsumes a theory of sets and uninterpreted functions.

To integrate separation logic within the DPLL(T) framework [18], it is necessary to present the input logic as a multi-sorted one. We assume, without loss of generality, the existence of a fixed theory $T = (\Sigma, \mathbf{M})$ that subsumes a theory for sets of sort $\mathsf{Set}(\sigma)$ [2], for any sort $\sigma \in \Sigma^{\mathsf{s}}$ of set elements. The functions for the set theory are the union $\cup^{\mathsf{Set}(\sigma)\,\mathsf{Set}(\sigma)\,\mathsf{Set}(\sigma)}$, intersection $\cap^{\mathsf{Set}(\sigma)\,\mathsf{Set}(\sigma)\,\mathsf{Set}(\sigma)}$, singleton set $\{.\}^{\sigma\,\mathsf{Set}(\sigma)}$ and empty set $\emptyset^{\mathsf{Set}(\sigma)}$. The interpretation of these functions is the classical one. We introduce the shorthands $s \subseteq s'$ for $s \cup s' \approx s'$ and $t \in s$ for $\{t\} \subseteq s$, given any terms s and s' of sort $\mathsf{Set}(\sigma)$ and t of sort σ. Moreover, $\mathsf{Loc}, \mathsf{Data} \in \Sigma^{\mathsf{s}}$, $\mathsf{nil} \in \Sigma^{\mathsf{f}}$ and we assume that Σ^{f} contains infinitely many function symbols $\mathsf{pt}^{\mathsf{Loc}\,\mathsf{Data}}$. Additionally, we consider an if-then-else operator $\mathsf{ite}(b, t, u)$, of sort $\mathsf{Bool} \times \sigma \times \sigma \to \sigma$, for each sort $\sigma \in \Sigma^{\mathsf{s}}$, that evaluates to t if b is true, and to u, otherwise.

The aim is to express separation logic connectives in multi-sorted second-order logic through a *labelling* transformation which introduces (i) constraints over variables of sort $\mathsf{Set}(\mathsf{Loc})$ and (ii) terms over uninterpreted *points-to* functions of sort $\mathsf{Loc} \to \mathsf{Data}$.

$$\frac{(\phi * \psi) \lhd [\overline{\ell}, \overline{\mathsf{pt}}]}{\neg\forall \ell_1 \forall \ell_2 \,.\, \neg(\ell_1 \cap \ell_2 \approx \emptyset \wedge \ell_1 \cup \ell_2 \approx \bigcup \overline{\ell} \wedge \phi \lhd [\ell_1 \cap \overline{\ell}, \overline{\mathsf{pt}}] \wedge \psi \lhd [\ell_2 \cap \overline{\ell}, \overline{\mathsf{pt}}])}$$

$$\frac{(\phi \mathrel{-\!\!*} \psi) \lhd [\overline{\ell}, \overline{\mathsf{pt}}]}{\forall \ell' \forall \mathsf{pt}' \,.\, (\ell' \cap (\bigcup \overline{\ell}) \approx \emptyset \wedge \phi \lhd [\ell', \mathsf{pt}']) \Rightarrow \psi \lhd [\ell' \cdot \overline{\ell}, \mathsf{pt}' \cdot \overline{\mathsf{pt}}]}$$

$$\frac{t \mapsto u \lhd [\overline{\ell}, \overline{\mathsf{pt}}]}{\bigcup \overline{\ell} \approx \{t\} \wedge \mathsf{ite}(t \in \overline{\ell}, \overline{\mathsf{pt}}(t) \approx u) \wedge \neg(t \approx \mathsf{nil})} \qquad \frac{(\phi \wedge \psi) \lhd [\overline{\ell}, \overline{\mathsf{pt}}]}{\phi \lhd [\overline{\ell}, \overline{\mathsf{pt}}] \wedge \psi \lhd [\overline{\ell}, \overline{\mathsf{pt}}]}$$

$$\frac{(\neg\phi) \lhd [\overline{\ell}, \overline{\mathsf{pt}}]}{\neg(\phi \lhd [\overline{\ell}, \overline{\mathsf{pt}}])} \qquad \frac{\mathsf{emp} \lhd [\overline{\ell}, \overline{\mathsf{pt}}]}{\bigcup \overline{\ell} \approx \emptyset} \qquad \frac{\phi \lhd [\overline{\ell}, \overline{\mathsf{pt}}]}{\phi} \ \phi \text{ is a } T\text{-formula}$$

Figure 3.1: Labelling Rules

Given a $\mathsf{SL}(T)$-formula ϕ, we denote its translation as $\phi \lhd [\overline{\ell}, \overline{\mathsf{pt}}]$, where $\overline{\ell} = \langle \ell_1, \ldots, \ell_n \rangle$ is a tuple of variables of sort $\mathsf{Set}(\mathsf{Loc})$ and $\overline{\mathsf{pt}} = \langle \mathsf{pt}_1, \ldots, \mathsf{pt}_n \rangle$ is a tuple of universally quantified uninterpreted function symbols. To ease the notation, we write ℓ and pt for the singleton tuples $\langle \ell \rangle$ and $\langle \mathsf{pt} \rangle$, $\bigcup \overline{\ell}$ for $\ell_1 \cup \ldots \cup \ell_n$, $\ell' \cap \overline{\ell}$ for $\langle \ell' \cap \ell_1, \ldots, \ell' \cap \ell_n \rangle$, $\ell' \cdot \overline{\ell}$ for $\langle \ell', \ell_1, \ldots, \ell_n \rangle$, and $\mathsf{ite}(t \in \overline{\ell}, \overline{\mathsf{pt}}(t) \approx u)$ for $\mathsf{ite}(t \in \ell_1, \mathsf{pt}_1(t) \approx u, \mathsf{ite}(t \in \ell_2, \mathsf{pt}_2(t) \approx u, \ldots \mathsf{ite}(t \in \ell_n, \mathsf{pt}_n(t) \approx u, \top) \ldots))$.

Intuitively, a labelled formula $\phi \lhd [\overline{\ell}, \overline{\mathsf{pt}}]$ indicates that it is possible to build, from any of its models $(\mathcal{I}, \nu)$, a heap h such that $\mathcal{I}, \nu, h \models^{\mathsf{SL}} \phi$, where $\mathrm{dom}(h) = \nu(\ell_1) \cup \ldots \cup \nu(\ell_n)$ and $h = \mathsf{pt}_1^{\mathcal{I}}{\downarrow_{\nu(\ell_1)}} \cup \ldots \cup \mathsf{pt}_n^{\mathcal{I}}{\downarrow_{\nu(\ell_n)}}$. For any $i \in [n]$, ℓ_i defines a slice of the domain of the heap, while the restriction of pt_i to the valuation of ℓ_i describes the heap relation on that slice. Note how each interpretation $\mathcal{I}$ of $\overline{\mathsf{pt}}$ and valuation $\nu \in \mathcal{V}_{\mathcal{I}}$ of $\overline{\ell}$ such that $\nu(\ell_i) \cap \nu(\ell_j) = \emptyset$, for all $i, j \in [k]$ with $i \neq j$, define a unique heap.

The rewriting rules in Figure 3.1 translate an input $\mathsf{SL}(T)$ formula ϕ into a labelled second-order formula, with quantifiers over sets and uninterpreted functions. A labelling step $\phi[\psi] \Longrightarrow \phi[\psi'/\psi]$ can be applied to ϕ if ψ and ψ' match the antecedent and consequent of a rewriting rule, respectively. We write $\phi \Longrightarrow^* \phi'$ to indicate that ϕ' was obtained from ϕ after one or more labelling steps. Our rewriting system is confluent, and we use $\phi{\Downarrow}$ for the normal form of ϕ with respect to the application of labelling steps.

Example 3.2.1. Consider the $\mathsf{SL}(T)$ formula $\mathsf{emp} \wedge (x \mapsto a \mathrel{-\!\!*} y \mapsto b)$. The reduction to second-order logic is given below:

$$\begin{array}{ll}
(\mathsf{emp} \wedge (x \mapsto a \mathrel{-\!\!*} y \mapsto b)) \lhd [\ell, \mathsf{pt}] & \stackrel{\wedge}{\Longrightarrow} \\
\mathsf{emp} \lhd [\ell, \mathsf{pt}] \wedge (x \mapsto a \mathrel{-\!\!*} y \mapsto b) \lhd [\ell, \mathsf{pt}] & \stackrel{\mathsf{emp}}{\Longrightarrow} \\
\ell \approx \emptyset \wedge (x \mapsto a \mathrel{-\!\!*} y \mapsto b) \lhd [\ell, \mathsf{pt}] & \stackrel{\mathrel{-\!\!*}}{\Longrightarrow} \\
\ell \approx \emptyset \wedge (\forall \ell' \forall \mathsf{pt}' \,.\, \ell' \cap \ell \approx \emptyset \wedge x \mapsto a \lhd [\ell', \mathsf{pt}'] \Rightarrow y \mapsto b \lhd [\langle \ell', \ell \rangle, \langle \mathsf{pt}', \mathsf{pt} \rangle]) & \stackrel{\mapsto}{\Longrightarrow} \\
\ell \approx \emptyset \wedge (\forall \ell' \forall \mathsf{pt}' \,.\, \ell' \cap \ell \approx \emptyset \wedge \ell' \approx \{x\} \wedge \mathsf{ite}(x \in \ell', \mathsf{pt}'(x) \approx a, \top) \wedge \neg(x \approx \mathsf{nil}) \Rightarrow & \\
\ell' \cup \ell \approx \{y\} \wedge \mathsf{ite}(y \in \ell', \mathsf{pt}'(y) \approx b, \mathsf{ite}(y \in \ell, \mathsf{pt}(y) \approx b, \top)) \wedge \neg(y \approx \mathsf{nil})) & \blacktriangleleft
\end{array}$$

The following lemma reduces the $\mathsf{SL}(T)$-satisfiability problem to the satisfiability of a second-order fragment of the multi-sorted theory T, with quantifiers over sets and uninter-

preted functions. Given the tuple of heaps $\overline{\mathbf{h}} = \langle h_1, \ldots, h_n \rangle$, we write $\mathrm{dom}(\overline{\mathbf{h}})$ to denote the tuple $\langle \mathrm{dom}(h_1), \ldots, \mathrm{dom}(h_n) \rangle$.

Lemma 3.2.1. Given an $\mathsf{SL}(T)$-formula ϕ and the tuples $\overline{\ell} = \langle \ell_1, \ldots, \ell_n \rangle$, $\overline{\mathsf{pt}} = \langle \mathsf{pt}_1, \ldots, \mathsf{pt}_n \rangle$ with $n > 0$, then, for any interpretation $\mathcal{I}$, valuation $\nu \in \mathcal{V}_{\mathcal{I}}$ and heap $h \in \mathsf{Heaps}^{\mathcal{I}}$, we have that $\mathcal{I}, \nu, h \models^{\mathsf{SL}} \phi$ if and only if the entailment $\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}'}], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \phi \triangleleft [\overline{\ell}, \overline{\mathsf{pt}}] \Downarrow$ holds: (1) for all heaps $\overline{\mathbf{h}} = \langle h_1, \ldots, h_n \rangle$ such that $h = h_1 \uplus \ldots \uplus h_n$, and (2) for all heaps $\overline{\mathbf{h}'} = \langle h'_1, \ldots, h'_n \rangle$ such that $h_1 \subseteq h'_1, \ldots, h_n \subseteq h'_n$.

Proof. We prove this lemma by induction on the structure of ϕ.

- If ϕ is a T-formula, then the lemma holds trivially, as $\phi \triangleleft [\overline{\ell}, \overline{\mathsf{pt}}] \Downarrow \equiv \phi$ and the symbols from $\overline{\ell}$ and $\overline{\mathsf{pt}}$ do not occur in ϕ.
- If $\phi \equiv \mathsf{emp}$, then $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \mathsf{emp}$ if and only if $\mathrm{dom}(h) = \emptyset$.

  "$\Rightarrow$" For any tuples $\overline{\mathbf{h}}$ satisfying condition (1) we have $\mathrm{dom}(h_i) = \emptyset$ for all $i \in [n]$, thus $\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}}'], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \bigcup \overline{\ell} = \emptyset$ for all $\overline{\mathbf{h}}'$ satisfying condition (2).

  "$\Leftarrow$" Let $\overline{\mathbf{h}}$ and $\overline{\mathbf{h}}'$ be tuples of heaps satisfying conditions (1) and (2), such that $\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}}'], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \mathsf{emp} \triangleleft [\overline{\ell}, \overline{\mathsf{pt}}] \Downarrow$, then $\mathrm{dom}(h) = \emptyset$ and $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \mathsf{emp}$.
- If $\phi \equiv t \mapsto u$, then $\mathcal{I}, h, \nu \models^{\mathsf{SL}} t \mapsto u$ if and only if $h = \{(t^{\mathcal{I}}_\nu, u^{\mathcal{I}}_\nu)\}$ and $t^{\mathcal{I}}_\nu \neq \mathsf{nil}^{\mathcal{I}}$.

  "$\Rightarrow$" Let $\overline{\mathbf{h}}$ and $\overline{\mathbf{h}}'$ be tuples of heaps satisfying the conditions (1) and (2). Then there exists $i \in [n]$ such that $\mathrm{dom}(h_i) = \{t^{\mathcal{I}}_\nu\}$, $h'_i(t^{\mathcal{I}}_\nu) = u^{\mathcal{I}}_\nu$ and $\mathrm{dom}(h_j) = \emptyset$ for all $j \in [n] \setminus \{i\}$. In consequence, we obtain that $\mathcal{I}[\mathsf{pt} \leftarrow \overline{\mathbf{h}}'], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \bigcup \overline{\ell} = \{t\} \wedge \mathsf{ite}(t \in \overline{\ell}, \overline{\mathsf{pt}}(t) \approx u) \wedge \neg(t \approx \mathsf{nil})$.

  "$\Leftarrow$" If $\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}'}], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T t \mapsto u \triangleleft [\overline{\ell}, \overline{\mathsf{pt}}] \Downarrow$ for each $\overline{\mathbf{h}}$ and $\overline{\mathbf{h}}'$ satisfying conditions (1) and (2), we easily obtain that $\{t^{\mathcal{I}}_\nu\} = \mathrm{dom}(h_i) \subseteq \mathrm{dom}(h'_i)$ and $h'_i(t^{\mathcal{I}}_\nu) = u^{\mathcal{I}}_\nu$ for some $i \in [n]$, leading to $h = \{(t^{\mathcal{I}}_\nu, u^{\mathcal{I}}_\nu)\}$. Moreover, $\neg(t^{\mathcal{I}}_\nu \approx \mathsf{nil})$ holds, thus we obtain $\mathcal{I}, h, \nu \models^{\mathsf{SL}} t \mapsto u$.
- If $\phi \equiv \psi_1 * \psi_2$, then $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \psi_1 * \psi_2$ if and only if there exist heaps g_1, g_2 with $h = g_1 \uplus g_2$ such that $\mathcal{I}, g_1, \nu \models^{\mathsf{SL}} \psi_1$ and $\mathcal{I}, g_2, \nu \models^{\mathsf{SL}} \psi_2$.

  "$\Rightarrow$" Let $\overline{\mathbf{h}}$ and $\overline{\mathbf{h}'}$ be tuples of heaps satisfying conditions (1) and (2). By the induction hypothesis we obtain

  $$\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}'}], \nu[\ell_1 \leftarrow \mathrm{dom}(g_1)][\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_1 \triangleleft [\ell_1 \cap \overline{\ell}, \overline{\mathsf{pt}}] \Downarrow$$
  $$\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}'}], \nu[\ell_2 \leftarrow \mathrm{dom}(g_2)][\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_2 \triangleleft [\ell_2 \cap \overline{\ell}, \overline{\mathsf{pt}}] \Downarrow$$

  because $g_j \cap h_i \subseteq h'_i$ for each $j \in [2]$ and $i \in [n]$. Moreover, $\mathrm{dom}(g_1) \cap \mathrm{dom}(g_2) = \emptyset$ and $\mathrm{dom}(g_1) \cup \mathrm{dom}(g_2) = \bigcup_{i=1}^{n} \mathrm{dom}(h_i)$, thus we obtain $\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}'}], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_1 * \psi_2 \triangleleft [\ell, \mathsf{pt}] \Downarrow$.

  "$\Leftarrow$" If $\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}'}], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_1 * \psi_2 \triangleleft [\ell, \mathsf{pt}] \Downarrow$, there exist sets $L_1, L_2 \subseteq \mathsf{Loc}$ such that $L_1 \cap L_2 = \emptyset$ and $\mathrm{dom}(h) = L_1 \cup L_2$. Let $g_1 = h{\downarrow_{L_1}}$ and $g_2 = h{\downarrow_{L_2}}$. We have that $h = g_1 \uplus g_2$ and

  $$\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}'}], \nu[\ell_1 \leftarrow \mathrm{dom}(g_1)][\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_1 \triangleleft [\ell_1 \cap \overline{\ell}, \overline{\mathsf{pt}}] \Downarrow$$
  $$\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}'}], \nu[\ell_2 \leftarrow \mathrm{dom}(g_2)][\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_2 \triangleleft [\ell_2 \cap \overline{\ell}, \overline{\mathsf{pt}}] \Downarrow$$

  Moreover, $g_j = \biguplus_{i=1}^{n} g_j \cap h_i$ and $g_j \cap h_i \subseteq h'_i$, for $j \in [2]$ and $i \in [n]$, thus we can apply the induction hypothesis to obtain $\mathcal{I}, g_1, \nu \models^{\mathsf{SL}} \psi_1$ and $\mathcal{I}, g_2, \nu \models^{\mathsf{SL}} \psi_2$, which together give us $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \psi_1 * \psi_2$.

– If $\phi \equiv \psi_1 \mathrel{-\!\!*} \psi_2$, then $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \psi_1 \mathrel{-\!\!*} \psi_2$ if and only if for any heap $h_0 \# h$ such that $\mathcal{I}, h_0, \nu \models^{\mathsf{SL}} \psi_1$, we have $\mathcal{I}, h_0 \uplus h, \nu \models^{\mathsf{SL}} \psi_2$.

"$\Rightarrow$" Suppose that $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \psi_1 \mathrel{-\!\!*} \psi_2$ and let $g \subseteq g'$ be heaps such that $g \# h$ and $\mathcal{I}[\mathsf{pt}' \leftarrow g'], \nu[\ell' \leftarrow \mathrm{dom}(g)] \models_T \psi_1 \triangleleft [\ell', \mathsf{pt}']\Downarrow$. By the induction hypothesis, we obtain $\mathcal{I}, g, \nu \models^{\mathsf{SL}} \psi_1$, thus $\mathcal{I}, g \uplus h, \nu \models^{\mathsf{SL}} \psi_2$. Since, moreover, $g \cdot \overline{\mathbf{h}}$ and $g' \cdot \overline{\mathbf{h}'}$ satisfy the conditions (1) and (2), by the induction hypothesis, we obtain $\mathcal{I}[\mathsf{pt}' \leftarrow g'][\mathsf{pt} \leftarrow \overline{\mathbf{h}}'], \nu[\ell' \leftarrow \mathrm{dom}(g)][\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_2 \triangleleft [\ell' \cdot \overline{\ell}, \mathsf{pt}' \cdot \overline{\mathsf{pt}}]\Downarrow$. Since the choice of g and g' was arbitrary, it follows that $\mathcal{I}[\mathsf{pt} \leftarrow \overline{\mathbf{h}}'], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_1 \mathrel{-\!\!*} \psi_2 \triangleleft [\overline{\ell}, \overline{\mathsf{pt}}]\Downarrow$.

"$\Leftarrow$" Suppose that $\mathcal{I}[\mathsf{pt} \leftarrow \overline{\mathbf{h}}'], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_1 \mathrel{-\!\!*} \psi_2 \triangleleft [\overline{\ell}, \overline{\mathsf{pt}}]\Downarrow$ and let $g \subseteq g'$ be heaps such that $g \# h$ and $\mathcal{I}, g, \nu \models^{\mathsf{SL}} \psi_1$. By the induction hypothesis, we have that $\mathcal{I}[\mathsf{pt}' \leftarrow g'], \nu[\ell' \leftarrow \mathrm{dom}(g)] \models_T \psi_1 \triangleleft [\ell', \mathsf{pt}']\Downarrow$ and, since $\mathrm{dom}(g) \cap (\bigcup_{i=1}^{n} \mathrm{dom}(h_i)) = \emptyset$, we have $\mathcal{I}[\mathsf{pt}' \leftarrow g'][\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}}'], \nu[\ell' \leftarrow \mathrm{dom}(g)][\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \psi_2 \triangleleft [\ell' \cdot \overline{\ell}, \mathsf{pt}' \cdot \overline{\mathsf{pt}}]\Downarrow$. By the induction hypothesis, we obtain $\mathcal{I}, g \uplus h, \nu \models^{\mathsf{SL}} \psi_2$, thus $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \psi_1 \mathrel{-\!\!*} \psi_2$.

We omit the cases $\phi \equiv \psi_1 \wedge \psi_2$ and $\phi \equiv \neg\psi$, since they are straightforward and can be proved by simple applications of the inductive hypothesis. □

Although satisfiability is undecidable in the presence of quantifiers and uninterpreted functions, the following section strengthens the reduction by adapting the rewriting rules for $*$ and $\mathrel{-\!\!*}$ (Figure 3.1) to use bounded quantification over finite set domains.

3.2.2 Reducing $\mathsf{SL}(T)$ to Quantifiers over Bounded Sets

Through the translation introduced in the previous section, we have reduced any instance of the $\mathsf{SL}(T)$-satisfiability problem to an instance of the T-satisfiability problem in the second-order multi-sorted extension of the theory T, which subsumes a theory for sets and uninterpreted function symbols. A key aspect of this translation is that the quantifiers occurring in second order T-formulae range over variables of sort $\mathsf{Set}(\mathsf{Loc})$ and function symbols of sort $\mathsf{Loc} \rightarrow \mathsf{Data}$. Building upon a small model property for SL over the data domain $\mathsf{Data} = \mathsf{Loc} \times \mathsf{Loc}$ [11], we can show that it is sufficient to consider quantified variables that range over a bounded domain of sets. This enables the replacement of universal quantifiers with finite conjunctions and leads to a decidability result that only relies on the decidability of the quantifier-free theory T, with sets and uninterpreted functions. We then develop an efficient procedure for lazy quantifier instantiation, based on counterexample-driven refinement, in Section 3.2.3.

With this purpose in mind, we quote Proposition 96 from [50], emphasizing the fact that its proof does not rely on the assumption that $\mathsf{Data} = \mathsf{Loc} \times \mathsf{Loc}$. Given an $\mathsf{SL}(T)$-formula ϕ, we introduce the following measure:

$$|\phi * \psi|_{\mathsf{h}} = |\phi|_{\mathsf{h}} + |\psi|_{\mathsf{h}} \qquad |\phi \mathrel{-\!\!*} \psi|_{\mathsf{h}} = |\psi|_{\mathsf{h}} \qquad |\phi \wedge \psi|_{\mathsf{h}} = \max(|\phi|_{\mathsf{h}}, |\psi|_{\mathsf{h}}) \qquad |\neg\phi|_{\mathsf{h}} = |\phi|_{\mathsf{h}}$$
$$|t \mapsto u|_{\mathsf{h}} = 1 \qquad |\mathsf{emp}|_{\mathsf{h}} = 1 \qquad |\phi|_{\mathsf{h}} = 0 \text{ if } \phi \text{ is pure}$$

Intuitively, $|\phi|_{\mathsf{h}}$ is the the maximum number of *invisible* locations in the domain of a heap h that are distinguished by ϕ but are not in the range of any valuation $\nu \in \mathcal{V}_{\mathcal{I}}$ such that $\mathcal{I}, \nu, h \models^{\mathsf{SL}} \phi$. For instance, if $\mathcal{I}, \nu, h \models^{\mathsf{SL}} \neg\mathsf{emp} * \neg\mathsf{emp}$ and $\mathrm{dom}(h) \geq 2$, then it is possible to restrict $\mathrm{dom}(h)$ to $|\neg\mathsf{emp} * \neg\mathsf{emp}|_{\mathsf{h}} = 2$ locations only, to satisfy this formula.

Let $\mathrm{Pt}(\phi)$ denote the set of terms (of sort $\mathsf{Loc} \cup \mathsf{Data}$) that occur on the left- or right-hand side of a $\mapsto$ operator in ϕ. We define $\mathrm{Pt}(t \mapsto u) = \{t, u\}$, $\mathrm{Pt}(\phi * \psi) = \mathrm{Pt}(\phi \mathrel{-\!\!*} \psi) =$

$\mathrm{Pt}(\phi) \cup \mathrm{Pt}(\psi)$, $\mathrm{Pt}(\neg\phi) = \mathrm{Pt}(\phi)$, and $\mathrm{Pt}(\mathsf{emp}) = \mathrm{Pt}(\phi) = \emptyset$ for a pure formula ϕ. The small model property is given by the following lemma.

Lemma 3.2.2. [50, Proposition 96] Given an $\mathsf{SL}(T)$-formula ϕ, for any interpretation $\mathcal{I}$ and any valuation $\nu \in \mathcal{V}_\mathcal{I}$, let $L \subseteq \mathsf{Loc}^\mathcal{I} \setminus \mathrm{Pt}(\phi)^\mathcal{I}_\nu$ be a set of locations, such that $\|L\| = |\phi|_\mathsf{h}$ and $v \in \mathsf{Data}^\mathcal{I} \setminus \mathrm{Pt}(\phi)^\mathcal{I}_\nu$. Then for any heap $h \in \mathsf{Heaps}^\mathcal{I}$ we have $\mathcal{I}, \nu, h \stackrel{\mathsf{SL}}{\models} \phi$ if and only if $\mathcal{I}, \nu, h' \stackrel{\mathsf{SL}}{\models} \phi$ for any $h' \in \mathsf{Heaps}^\mathcal{I}$ such that: (1) $\mathrm{dom}(h') \subseteq L \cup \mathrm{Pt}(\phi)^\mathcal{I}$, and (2) for all $\ell \in \mathrm{dom}(h')$, $h'(\ell) \in \mathrm{Pt}(\phi)^\mathcal{I} \cup \{v\}$.

The decidability of separation logic when $\mathsf{Data} = \mathsf{Loc} \times \mathsf{Loc}$ is justified by the fact that that, when an interpretation $\mathcal{I}$ of the free variables is fixed, it is enough to look within a finite set of heaps h', satisfying the above conditions, in order to find a heap model h for a separation logic formula. This gives a PSPACE upper bound for the satisfiability problem in this fragment.

Since the proof of Lemma 3.2.2 [50] does not involve reasoning about data values, other than equality checking, we can refine our reduction from the previous section by bounding the quantifiers to finite sets of constants of known size. We assume the existence of a total order on the (countable) set of constants in Σ^f of sort Loc, disjoint from any terms that occur in a given formula ϕ, and define $\mathrm{Bnd}(\phi, C) = \{c_{m+1}, \ldots, c_{m+|\phi|_\mathsf{h}}\}$, where $m = \max\{i \mid c_i \in C\}$, and $m = 0$ if $C = \emptyset$. Clearly, we have $\mathrm{Pt}(\phi) \cap \mathrm{Bnd}(\phi, C) = \emptyset$ and also $C \cap \mathrm{Bnd}(\phi, C) = \emptyset$, for any C and any ϕ.

$$\frac{\phi * \psi \lhd [\overline{\ell}, \overline{\mathsf{pt}}, C]}{\begin{array}{l}\neg\forall \ell_1 \forall \ell_2 \,.\, \ell_1 \cup \ell_2 \subseteq C \cup \mathrm{Pt}(\phi * \psi) \Rightarrow \\ \qquad \neg(\ell_1 \cap \ell_2 \approx \emptyset \wedge \ell_1 \cup \ell_2 \approx \bigcup \overline{\ell} \wedge \phi \lhd [\ell_1 \cap \overline{\ell}, \overline{\mathsf{pt}}, C] \wedge \psi \lhd [\ell_2 \cap \overline{\ell}, \overline{\mathsf{pt}}, C])\end{array}}$$

$$\frac{\phi \mathrel{-\!\!*} \psi \lhd [\overline{\ell}, \overline{\mathsf{pt}}, C]}{\begin{array}{l}\forall \ell' \forall \mathsf{pt}' \,.\, \ell' \subseteq C' \cup \mathrm{Pt}(\phi \mathrel{-\!\!*} \psi) \wedge \\ \qquad \mathsf{pt}' \subseteq (C' \cup \mathrm{Pt}(\phi \mathrel{-\!\!*} \psi)) \times (\mathrm{Pt}(\phi \mathrel{-\!\!*} \psi) \cup \{d\}) \Rightarrow \\ \qquad ((\ell' \cap (\bigcup \overline{\ell}) \approx \emptyset \ \wedge \ \phi \lhd [\ell', \mathsf{pt}', C']) \Rightarrow \psi \lhd [\ell' \cdot \overline{\ell}, \mathsf{pt}' \cdot \overline{\mathsf{pt}}, C])\end{array}} \quad \begin{array}{l} C' = \mathrm{Bnd}(\phi \wedge \psi, C) \\ d \notin \mathrm{Pt}(\phi \mathrel{-\!\!*} \psi)\end{array}$$

Figure 3.2: Bounded Quantifier Labelling Rules for $*$ and $\mathrel{-\!\!*}$

We now consider labelling judgements of the form $\varphi \lhd [\overline{\ell}, \overline{\mathsf{pt}}, C]$, where C is a finite set of constants of sort Loc, and modify all the rules in Figure 3.1, besides the ones with premises $(\phi * \psi) \lhd [\overline{\ell}, \overline{\mathsf{pt}}]$ and $(\phi \mathrel{-\!\!*} \psi) \lhd [\overline{\ell}, \overline{\mathsf{pt}}]$, by replacing any judgement $\varphi \lhd [\overline{\ell}, \overline{\mathsf{pt}}]$ with $\varphi \lhd [\overline{\ell}, \overline{\mathsf{pt}}, C]$. The two rules in Figure 3.2 are the bounded-quantifier equivalents of the $(\phi * \psi) \lhd [\overline{\ell}, \overline{\mathsf{pt}}]$ and $(\phi \mathrel{-\!\!*} \psi) \lhd [\overline{\ell}, \overline{\mathsf{pt}}]$ rules in Figure 3.1. We denote by $(\phi \lhd [\overline{\ell}, \overline{\mathsf{pt}}, C])\Downarrow$ the formula obtained by exhaustively applying the new labelling rules to $\phi \lhd [\overline{\ell}, \overline{\mathsf{pt}}, C]$.

The result of the labelling process is a formula in which all quantifiers are of the form $\forall \ell_1 \ldots \forall \ell_n \forall \mathsf{pt}_1 \ldots \forall \mathsf{pt}_n \,.\, \bigwedge_{i=1}^n \ell_i \subseteq L_i \wedge \bigwedge_{i=1}^n \mathsf{pt}_i \subseteq L_i \times D_i \Rightarrow \psi(\overline{\ell}, \overline{\mathsf{pt}})$, where L_i and D_i are finite sets of terms, none of which involve quantified variables, and ψ is a formula in the theory T with sets and uninterpreted functions. Moreover, the labelling rule for $\phi \mathrel{-\!\!*} \psi \lhd [\overline{\ell}, \overline{\mathsf{pt}}, C]$ uses a fresh constant d that does not occur in ϕ or ψ.

Example 3.2.2. We revisit below the labelling of from Example 3.2.1.

$$
\begin{array}{ll}
(\mathsf{emp} \wedge (x \mapsto a \mathrel{-\!\!*} y \mapsto b)) \triangleleft [\ell, \mathsf{pt}, \{c_1\}] & \Longrightarrow^{*} \\
\ell \approx \emptyset \wedge (x \mapsto a \mathrel{-\!\!*} y \mapsto b) \triangleleft [\ell, \mathsf{pt}, \{c_1\}] & \overset{\mathrel{-\!\!*}}{\Longrightarrow} \\
\ell \approx \emptyset \wedge (\forall \ell' \forall \mathsf{pt}' \,.\, \ell' \subseteq \{x, y, a, b, c_2\} \wedge \mathsf{pt}' \subseteq \{x, y, a, b, c_2\} \times \{x, y, a, b, d\} \Rightarrow & \\
((\ell' \cap \ell \approx \emptyset \wedge x \mapsto a \triangleleft [\ell', \mathsf{pt}', \{c_2\}]) \Rightarrow y \mapsto b \triangleleft [\langle \ell', \ell \rangle, \langle \mathsf{pt}', \mathsf{pt} \rangle, \{c_1\}])) & \overset{\mapsto}{\Longrightarrow} \\
\ell \approx \emptyset \wedge (\forall \ell' \forall \mathsf{pt}' \,.\, \ell' \subseteq \{x, y, a, b, c_2\} \wedge \mathsf{pt}' \subseteq \{x, y, a, b, c_2\} \times \{x, y, a, b, d\} \Rightarrow & \\
(\ell' \cap \ell \approx \emptyset \wedge \ell' \approx \{x\} \wedge \mathsf{ite}(x \in \ell', \mathsf{pt}'(x) \approx a, \top) \wedge \neg(x \approx \mathsf{nil}) \Rightarrow & \\
\ell' \cup \ell \approx \{y\} \wedge \mathsf{ite}(y \in \ell', \mathsf{pt}'(y) \approx b, \mathsf{ite}(y \in \ell, \mathsf{pt}(y) \approx b, \top)) \wedge \neg(y \approx \mathsf{nil}))) &
\end{array}
$$

We start the labelling using a set $C = \{c_1\}$, since $|\mathsf{emp} \wedge (x \mapsto a \mathrel{-\!\!*} y \mapsto b)|_{\mathsf{h}} = 1$. The rules for $\wedge$ and emp have the same behaviour, but they additionally carry over the set C. We apply the bounded quantifier version of the rule for $\mathrel{-\!\!*}$. Note that $\mathrm{Pt}(x \mapsto a \mathrel{-\!\!*} y \mapsto b) = \{x, y, a, b\}$. Because $|x \mapsto a \wedge y \mapsto b|_{\mathsf{h}} = 1$, the bounded quantifier labelling of the term $x \mapsto a \mathrel{-\!\!*} y \mapsto b$ introduces $C' = \{c_2\}$. The rule for $\mapsto$ is applied as usual. ◀

We now establish the soundness of the translation of $\mathsf{SL}(T)$ formulae in a fragment of T that contains only bounded quantifiers, using the rules in Figure 3.2. This relies on the following technical property of the magic wand.

Proposition 3.2.3. [50, Proposition 89] Given two $\mathsf{SL}(T)$-formulae ϕ and ψ, for any interpretation $\mathcal{I}$ of T and any valuation ν under $\mathcal{I}$, let $L \subseteq \mathsf{Loc}^{\mathcal{I}} \setminus \mathrm{Pt}(\phi \mathrel{-\!\!*} \psi)^{\mathcal{I}}_{\nu}$ be a set such that $\|L\| = \max(|\phi|_{\mathsf{h}}, |\psi|_{\mathsf{h}})$ and $v \in \mathsf{Data}^{\mathcal{I}} \setminus \mathrm{Pt}(\phi \mathrel{-\!\!*} \psi)^{\mathcal{I}}_{\nu}$ be a data value. Then for any heap $h \in \mathsf{Heaps}^{\mathcal{I}}$ we have $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \phi \mathrel{-\!\!*} \psi$ if and only if $\mathcal{I}, h \uplus h', \nu \models^{\mathsf{SL}} \psi$ for any $h' \in \mathsf{Heaps}^{\mathcal{I}}$, where $h' \# h$ and $\mathcal{I}, h', \nu \models^{\mathsf{SL}} \phi$, such that: (1) $\mathrm{dom}(h') \subseteq L \cup \mathrm{Pt}(\phi \mathrel{-\!\!*} \psi)^{\mathcal{I}}_{\nu}$, and (2) $h'(\ell) \in \mathrm{Pt}(\phi \mathrel{-\!\!*} \psi)^{\mathcal{I}}_{\nu} \cup \{v\}$, for all $\ell \in \mathrm{dom}(h')$.

Lemma 3.2.4. Given a formula ϕ in the language $\mathsf{SL}(T)$, for any interpretation $\mathcal{I}$ of T and valuation ν under $\mathcal{I}$, let $L \subseteq \mathsf{Loc}^{\mathcal{I}} \setminus \mathrm{Pt}(\phi)^{\mathcal{I}}_{\nu}$ be a set of locations such that $\|L\| = |\phi|_{\mathsf{h}}$ and $v \in \mathsf{Data}^{\mathcal{I}} \setminus \mathrm{Pt}(\phi)^{\mathcal{I}}$ be a data value. Then there exists a heap h such that $\mathcal{I}, \nu, h \models^{\mathsf{SL}} \phi$ if and only if there exist heaps $\overline{\mathbf{h}'} = \langle h'_1, \ldots, h'_n \rangle$ and $\overline{\mathbf{h}''} = \langle h''_1, \ldots, h''_n \rangle$ such that: (1) for all $i, j \in [n]$ with $i < j$, we have $h'_i \# h'_j$, (2) for all $i \in [n]$, we have $h'_i \subseteq h''_i$ and (3) $\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}''}][C \leftarrow L][d \leftarrow v], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}'})] \models_T \phi \triangleleft [\overline{\ell}, \overline{\mathsf{pt}}, C]\Downarrow$.

Proof. By Lemma 3.2.2, there exists h such that $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \phi$ if and only if there exists h' such that $\mathrm{dom}(h') \subseteq L \cup \mathrm{Pt}(\phi)^{\mathcal{I}}_{\nu}$ and for all $\ell \in \mathrm{dom}(h')$, $h'(\ell) \in \mathrm{Pt}(\phi)^{\mathcal{I}} \cup \{v\}$, for a value $v \in \mathsf{Data}^{\mathcal{I}} \setminus \mathrm{Pt}(\phi)^{\mathcal{I}}$. It is thus sufficient to prove the statement for these heaps only and assume that h satisfies the conditions of Lemma 3.2.2.

We show the following stronger statement. For any heap h, where $\mathrm{dom}(h) \subseteq L \cup \mathrm{Pt}(\phi)^{\mathcal{I}}_{\nu}$ and $\mathrm{img}(h) \in \mathrm{Pt}(\phi)^{\mathcal{I}}_{\nu} \cup \{v\}$, we have $\mathcal{I}, h, \nu \models^{\mathsf{SL}} \phi$ if and only if $\mathcal{I}[\overline{\mathsf{pt}} \leftarrow \overline{\mathbf{h}}'][C \leftarrow L][d \leftarrow v], \nu[\overline{\ell} \leftarrow \mathrm{dom}(\overline{\mathbf{h}})] \models_T \phi \triangleleft [\overline{\ell}, \overline{\mathsf{pt}}, C]\Downarrow$ for all tuples $\overline{\mathbf{h}} = \langle h_1, \ldots, h_n \rangle$ and $\overline{\mathbf{h}'} = \langle h'_1, \ldots, h'_n \rangle$ such that $h = h_1 \uplus \ldots \uplus h_n$ and $h_1 \subseteq h'_1, \ldots, h_n \subseteq h'_n$.

This can be proven by induction on the structure of ϕ, similarly to the proof of Lemma 3.2.1. It can be easily verified that the quantified sets variables and uninterpreted functions belong to the domains required by the rules of Figure 3.2 – for magic wand, in particular, this is ensured by Proposition 3.2.3. □

3.2.3 A Decision Procedure for Quantifier-Free $\mathsf{SL}(T)$ in SMT

This section presents our decision procedure for the $\mathsf{SL}(T)$-satisfiability of quantifier-free $\mathsf{SL}(T)$ formulae ϕ, which uses an underlying efficient decision procedure for the T-satisfiability of $\phi \triangleleft [\ell, \mathsf{pt}, C]\Downarrow$, obtained from the transformation described in Section 3.2.2. The main challenge presented by this problem is treating the universal quantification occurring in $\phi \triangleleft [\ell, \mathsf{pt}, C]\Downarrow$, which may involve arbitrary levels of quantifier alternation. As previously mentioned, the key factor for decidability is that all universally quantified symbols range over bounded sets.

More concretely, all universally quantified subformulae of $\phi \triangleleft [\ell, \mathsf{pt}, C]\Downarrow$ are of the form $\forall \overline{\mathbf{x}} \,.\, (\bigwedge \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}}) \Rightarrow \psi$, where $\overline{\mathbf{x}}$ is a tuple of ground terms and $\overline{\mathbf{s}}$ is a tuple of sets (or products of sets) containing ground terms. For brevity, we refer to such formulae using the shorthand $\forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi$ to denote a quantified formula of this form. While we can clearly reduce any such formulae to a quantifier-free conjunction of instances, constructing these instances is, in practice, very expensive. Following recent approaches for handling universal quantification [19, 40, 5, 43], we use a counterexample-guided method for choosing the instances of quantified formulae that are relevant to the satisfiability of our input. This method relies on an iterative procedure that maintains an evolving, equisatisfiable set Γ of quantifier-free formulae, which is initially obtained from ϕ by a purification step.

To this end, we extend some notions related to T-satisfiability (Definition 1.1.11) and T-entailment (Definition 1.1.12) to sets of formulae. Let F, F_1 and F_1 be sets of T-formulae. We write $[\![F]\!]_T$ for $[\![\bigwedge F]\!]_T$ and say that F is T-satisfiable whenever $[\![F]\!]_T \neq \emptyset$. Also, $F_1 \models_T F_2$ if and only if $[\![F_1]\!]_T \subseteq [\![F_2]\!]_T$. We call F_1 and F_2 equisatisfiable whenever $[\![F_1]\!]_T \neq \emptyset$ implies $[\![F_2]\!]_T \neq \emptyset$. In the case of our procedure, the sets Γ and $\{\phi\}$ are equisatisfiable, where ϕ is the input formula.

For a closed quantified (i.e. not occurring under any other quantifiers) formula $\forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \phi$, the purification step involves associating it to a boolean variable A, called the *guard* of the formula and a unique tuple of Skolem symbols $\overline{\mathbf{k}}$ of the same length and sorts as $\overline{\mathbf{x}}$. We write $(A, \overline{\mathbf{k}}) \leftrightharpoons \forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \phi$ to show that A and $\overline{\mathbf{k}}$ are associated with the formula.

Given a set of T-formulae Γ, we use $\mathrm{Q}(\Gamma)$ for the set of quantified formulae whose guard occurs within a formula in Γ. Replacing all closed quantified subformulae in ϕ with their corresponding guards is called *purifying* ϕ and we refer to the result as $\lfloor \phi \rfloor$. Conversely, replacing all guards in Γ by the quantified formulae they are associated with is called *unpurifying* and we denote the result by $\lceil \Gamma \rceil$.

We use $\lfloor \phi \rfloor^*$ for the the smallest set of formulae obtained by purifying an input formula ϕ and continuing to apply the purification step to its subformulae such that:

$$\lfloor \phi \rfloor \in \lfloor \phi \rfloor^*$$
$$(\neg A \Rightarrow \lfloor \neg \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor) \in \lfloor \phi \rfloor^* \text{ if } \forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi \in \mathrm{Q}(\lfloor \phi \rfloor^*) \text{ where } (A, \overline{\mathbf{k}}) \leftrightharpoons \forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi$$

In other words, $\lfloor \phi \rfloor^*$ contains clauses that witness the negation of each universally quantified formula occurring in ϕ. If ϕ is a T-formula possibly containing quantifiers, then $\lfloor \phi \rfloor^*$ is a set of quantifier-free T-formulae. Moreover, if all quantified formulae in ϕ are of the form $\forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi$ mentioned above, then all quantified formulae in $\mathrm{Q}(\lfloor \phi \rfloor^*)$ are also of this form.

Example 3.2.3. Consider the $\mathsf{SL}(T)$ formula $\phi \equiv \mathsf{emp} \wedge (y \mapsto 0 \mathrel{-\!\!*} y \mapsto 1) \wedge \neg(y \approx \mathsf{nil})$. Through a translation process similar to the one in Example 3.2.2, we obtain:

$$\begin{aligned}(\phi \triangleleft [\ell, \mathsf{pt}, \{c_1\}])\Downarrow \equiv\, & \ell \approx \emptyset \wedge (\forall \ell' \forall \mathsf{pt}' \,.\, \ell' \subseteq \{y, 0, 1, c_2\} \wedge \mathsf{pt}' \subseteq \{y, 0, 1, c_2\} \times \{y, 0, 1, d\} \Rightarrow \\ & (\ell' \cap \ell \approx \emptyset \wedge \ell' \approx \{y\} \wedge \mathsf{ite}(y \in \ell', \mathsf{pt}'(y) \approx 0, \top) \wedge \neg(y \approx \mathsf{nil}) \Rightarrow \ell' \cup \ell \approx \{y\} \\ & \wedge \mathsf{ite}(y \in \ell', \mathsf{pt}'(y) \approx 1, \mathsf{ite}(y \in \ell, \mathsf{pt}(y) \approx 1, \top)) \wedge \neg(y \approx \mathsf{nil}))) \wedge \neg(y \approx \mathsf{nil})\end{aligned}$$

Let $(\phi \triangleleft [\ell, \mathsf{pt}, \{c_1\}])\Downarrow = \ell \approx \emptyset \wedge (\forall \ell' \forall \mathsf{pt}' \,.\, \psi) \wedge \neg(y \approx \mathsf{nil})$. Purifying the result of the translation yields the following set, where $(A, \langle k_1, k_2 \rangle) \leftrightharpoons \forall \ell' \forall \mathsf{pt}' \,.\, \psi$:

$$\lfloor (\phi \triangleleft [\ell, \mathsf{pt}, \{c_1\}])\Downarrow \rfloor^* = \{\ell \approx \emptyset \wedge A \wedge \neg(y \approx \mathsf{nil}), \neg A \Rightarrow \neg\psi[\langle k_1, k_2 \rangle / \langle \ell', \mathsf{pt}' \rangle]\}$$

This set is obtained by first adding $\ell \approx \emptyset \wedge A \wedge \neg(y \approx \mathsf{nil}) \equiv \lfloor (\phi \triangleleft [\ell, \mathsf{pt}, \{c_1\}])\Downarrow \rfloor$. Then, because $\forall \ell' \forall \mathsf{pt}' \,.\, \psi \in \mathrm{Q}(\lfloor (\phi \triangleleft [\ell, \mathsf{pt}, \{c_1\}])\Downarrow \rfloor^*)$, we also need to add $\neg A \Rightarrow \lfloor \neg\psi[\langle k_1, k_2 \rangle / \langle \ell', \mathsf{pt}' \rangle] \rfloor$. Since ψ does not contain any other quantified subformulae, we have that $\lfloor \neg\psi[\langle k_1, k_2 \rangle / \langle \ell', \mathsf{pt}' \rangle] \rfloor \equiv \neg\psi[\langle k_1, k_2 \rangle / \langle \ell', \mathsf{pt}' \rangle]$ and the recursive purification stops here. ◀

$\mathsf{solve}_{\mathsf{SL}(T)}(\phi)$:

Let C be a set of fresh constants of sort Loc such that $\|C\| = |\phi|_{\mathsf{h}}$.

Let ℓ and pt be fresh symbols of sort $\mathsf{Set}(\mathsf{Loc})$ and $\mathsf{Loc} \rightarrow \mathsf{Data}$, respectively.

Return $\mathsf{solve}_T(\lfloor (\phi \triangleleft [\ell, \mathsf{pt}, C])\Downarrow \rfloor^*)$.

$\mathsf{solve}_T(\Gamma)$:

1. If Γ is T-unsatisfiable,

   return "unsat",

   else let $(\mathcal{I}, \nu)$ be a T-model of Γ.

2. If $\Gamma \cup \{A\} \models_T \lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$ for all $\forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi \in \mathrm{Q}(\Gamma)$, $(A, \overline{\mathbf{k}}) \leftrightharpoons \forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi$ and $A^{\mathcal{I}} = \top$,

   return "sat",

   else let $(\mathcal{J}, \gamma)$ be a T-model of $\Gamma \cup \{A, \neg\lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor\}$ for some $\prec_{\Gamma,(\mathcal{I},\nu)}$-minimal $\forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi$, where $(A, \overline{\mathbf{k}}) \leftrightharpoons \forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} . \psi$.

3. Let $\overline{\mathbf{t}}$ be a vector of terms, such that $\overline{\mathbf{t}} \subseteq \overline{\mathbf{s}}$, and $\overline{\mathbf{t}}_\gamma^{\mathcal{J}} = \overline{\mathbf{k}}_\gamma^{\mathcal{J}}$.

   Return $\mathsf{solve}_T(\, \Gamma \cup \lfloor A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}] \rfloor^*)$.

Figure 3.3: Procedure for deciding the satisfiability of the $\mathsf{SL}(T)$ formula ϕ.

Figure 3.3 illustrates our $\mathsf{solve}_{\mathsf{SL}(T)}$ procedure for determining the $\mathsf{SL}(T)$-satisfiability of the input formula ϕ. It operates by first constructing the set C of fresh constant symbols based on the value of $|\phi|_{\mathsf{h}}$, computed by traversing the structure of ϕ, and also choosing the fresh symbols ℓ and pt. By applying the rules in Figure 3.2, ϕ is translated to $(\phi \triangleleft [\ell, C])\Downarrow$, then purified to obtain $\lfloor (\phi \triangleleft [\ell, \mathsf{pt}, C])\Downarrow \rfloor^*$. The subprocedure solve_T is invoked on the set resulting from the purification.

Broadly speaking, the recursive procedure solve_T takes as input a quantifier-free set of T-formulae Γ that is T-unsatisfiable if and only if $\phi \triangleleft [\ell, \mathsf{pt}, C] \Downarrow$ is. The set Γ can be viewed as an under-approximation of $\phi \triangleleft [\ell, \mathsf{pt}, C] \Downarrow$, where the models for Γ are a superset of those for $\phi \triangleleft [\ell, \mathsf{pt}, C] \Downarrow$. On each call, solve_T either (i) terminates with "unsat", indicating that $\phi \triangleleft [\ell, \mathsf{pt}, C]\Downarrow$ is T-unsatisfiable, (ii) terminates with "sat", indicating that $\phi \triangleleft [\ell, \mathsf{pt}, C] \Downarrow$ is T-satisfiable, or (iii) adds the set resulting from the purification of the instance $\lfloor A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}] \rfloor^*$ to Γ and repeats.

More concretely, Step 1 of the procedure determines the T-satisfiability of Γ using a combination of a satisfiability solver and a decision procedure for T. Non-constant Skolem symbols k introduced by the procedure may be treated as uninterpreted functions. Constraints of the form $k \subseteq S_1 \times S_2$ are translated to $\bigwedge_{c \in S_1} k(c) \in S_2$, while the domain of k may be restricted to the set $\{c^{\mathcal{I}} \mid c \in S_1\}$ in models $(\mathcal{I}, \nu)$ found in steps 1 and 2 of the procedure. By the construction of $\phi \triangleleft [\ell, \mathsf{pt}, C]\Downarrow$, k is only applied to the terms in S_1, thus these restrictions do not cause a loss of generality. Because Γ is T-entailed by $\lceil\Gamma\rceil$, if it is established that Γ is T-unsatisfiable, then the procedure may terminate with "unsat". Otherwise, there exists a T-model $\mathcal{I}$ for Γ.

For each guard A that is mapped to true by ν, Step 2 of the procedure solve_T checks whether $\Gamma \cup \{A\}$ T-entails $\lfloor\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\rfloor$ for the fresh free constants $\overline{\mathbf{k}}$. This can be accomplished by determining whether $\Gamma \cup \{A, \neg\lfloor\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\rfloor\}$ is T-unsatisfiable. A successful check for a quantified formula $\forall\overline{\mathbf{x}}\,.\,\psi$ means that $\Gamma \models_T \{\forall\overline{\mathbf{x}}\,.\,\psi\}$. If this check succeeds for all such quantified formulae, then Γ is equivalent to $\lceil\Gamma\rceil$ and the procedure may terminate with "sat". Otherwise, let $Q^+_{(\mathcal{I},\nu)}(\Gamma)$ be the subset of $Q(\Gamma)$ for which this check did not succeed and which we call *active quantified formulae* for Γ and $(\mathcal{I}, \nu)$. We consider an active quantified formula that is minimal with respect to the relation $\prec_{\Gamma,(\mathcal{I},\nu)}$ over $\mathrm{Q}(\Gamma)$, where:

$$\psi \prec_{\Gamma,(\mathcal{I},\nu)} \phi \quad \text{if and only if } \psi \in \mathrm{Q}(\lfloor\phi\rfloor^*) \cap Q^+_{(\mathcal{I},\nu)}(\Gamma)$$

Through this ordering, we consider the innermost active quantified formulae first. Let $\forall\overline{\mathbf{x}}\,.\,\psi$ be minimal with respect to $\prec_{\Gamma,(\mathcal{I},\nu)}$, where $(A, \overline{\mathbf{k}}) \leftrightharpoons \forall\overline{\mathbf{x}}\,.\,\psi$. Since $\Gamma \cup A$ does not T-entail $\lfloor\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\rfloor$, there must exist a model $(\mathcal{J}, \gamma)$ for $\Gamma \cup \{\lfloor\neg\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\rfloor\}$ where $A^{\mathcal{J}} = \top$.

Step 3 of the procedure chooses a tuple of terms $\overline{\mathbf{t}} = \langle t_1, \ldots, t_n\rangle$ based on the model $(\mathcal{J}, \gamma)$ and adds to Γ the set of formulae obtained by purifying $A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}]$, where A is the guard of $\forall\overline{\mathbf{x}} \subseteq \overline{\mathbf{s}}\,.\,\psi$. Let $\overline{\mathbf{s}} = \langle s_1, \ldots, s_n\rangle$, where each s_i is a finite union of ground T-terms. Each tuple $\overline{\mathbf{t}}$ is chosen such that t_i is a subset of s_i, for all $i \in [n]$, with $\overline{\mathbf{t}}^{\mathcal{J}}_\gamma = \overline{\mathbf{k}}^{\mathcal{J}}_\gamma$. These two criteria are essential to the termination of our algorithm: the former ensures that only a finite number of possible instances can ever be added to Γ, and the latter ensures that the same instance is never added more than once.

We now show the correctness of our procedure, which requires some intermediate lemmas.

Lemma 3.2.5. Given any T-formula ϕ, the following hold:

(1) ϕ is T-satisfiable only if $\lfloor\phi\rfloor^*$ is T-satisfiable, and

(2) ϕ and $\lceil\lfloor\phi\rfloor^*\rceil$ are T-equivalent up to their shared variables.

Proof. (Sketch) To show (1), let $(\mathcal{I}, \nu)$ be a T-model of ϕ. Let γ be an extension of ν such that $\gamma(A) = (\forall\overline{\mathbf{x}}.\psi)^{\mathcal{I}}_\nu$, for each $\forall\overline{\mathbf{x}}.\psi \in \mathrm{Q}(\lfloor\phi\rfloor^*)$ where $(A, \overline{\mathbf{k}}) \leftrightharpoons \forall\overline{\mathbf{x}}.\psi$. Then the pair $(\mathcal{I}, \gamma)$ is a T-model for $\lfloor\phi\rfloor^*$.

To show (2), notice that $\lceil\lfloor\phi\rfloor^*\rceil$ is a set that can be constructed from an initial value $\{\phi\}$, and updated by adding formulae of the form $(\neg\forall\overline{\mathbf{x}}.\psi \Rightarrow \neg\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}])$ for fresh constants $\overline{\mathbf{k}}$. For each step of this construction, the set of models are the same when restricted to the interpretation of all variables apart from $\overline{\mathbf{k}}$. Thus, by induction, ϕ and $\lceil\lfloor\phi\rfloor^*\rceil$ are T-equivalent up to their shared variables. □

Lemma 3.2.6. For every recursive call to $\mathsf{solve}_T(\Gamma)$, if $\Gamma \cup A \models_T \lfloor\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\rfloor$, where $(A, \overline{\mathbf{k}}) \leftrightharpoons \forall\overline{\mathbf{x}}.\psi$, then $\Gamma \models_T \forall\overline{\mathbf{x}}.\psi$.

Proof. It suffices to show there exists a subset Γ' of Γ such that Γ' does not contain $\overline{\mathbf{k}}$ and $\Gamma' \models_T \lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$. Note that, if such a Γ' exists, then, since $\Gamma' \subseteq \Gamma$ and Γ' do not contain $\overline{\mathbf{k}}$, we have that $\Gamma \models_T \forall \overline{\mathbf{x}}.\psi$. By construction, Γ may be partitioned into sets Γ' and Γ'', where Γ' does not contain $\overline{\mathbf{k}}$, and Γ'' contains only (i) $\neg A \Rightarrow \lfloor \neg\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$, and (ii) constraints of the form $\neg A_1 \Rightarrow \lfloor \neg\psi_1[\overline{\mathbf{j}}/\overline{\mathbf{y}}] \rfloor$ and $\lfloor A_1 \Rightarrow \psi_1[\overline{\mathbf{t}}/\overline{\mathbf{y}}] \rfloor^*$, where $(A_1, \overline{\mathbf{j}}) \leftrightharpoons \forall \overline{\mathbf{y}}.\psi_1$ and A_1 does not occur in Γ'.

Assume that $\Gamma \setminus \Gamma'', A, \lfloor \neg\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$ has a T-model $(\mathcal{I}, \nu)$. Let γ be an extension of ν such that for each $(A_1, \overline{\mathbf{j}}) \leftrightharpoons \forall \overline{\mathbf{x}}.\psi_1$ occurring in $\mathrm{Q}(\Gamma'')$ and not in $\mathrm{Q}(\Gamma')$ we have $\gamma(A_1) = (\forall \overline{\mathbf{x}}.\psi_1)^{\mathcal{I}}_{\nu}$. The pair $(\mathcal{I}, \gamma)$ is a T-model for Γ, A and $\lfloor \neg\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$, noting that the constraint $\neg A \Rightarrow \lfloor \neg\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$ holds since $\gamma(A)$ must be $\top$. This contradicts the assumption $\Gamma \cup \{A\} \models_T \lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$, and thus $(\Gamma \setminus \Gamma'') \cup \{A\} \cup \lfloor \neg\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$ is T-unsatisfiable. In other words, $\Gamma' \cup \{A\} \models_T \lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$. Since A does not occur in $\lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$, we have that $\Gamma' \models_T \lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$, and the lemma holds. □

Lemma 3.2.7. If $\Gamma = \{\lfloor \phi \rfloor^* \mid \phi \in S\}$ for some set S, and $\forall \overline{\mathbf{x}}.\psi \in \mathrm{Q}(\Gamma)$ is $\prec_{\Gamma,(\mathcal{I},\nu)}$-minimal, then $\Gamma \cup \{(\neg)\lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor\}$ and $\Gamma \cup \{(\neg)\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\}$ are T-equivalent up to their shared variables.

Proof. (Sketch) By definition of $\prec_{\Gamma,(\mathcal{I},\nu)}$-minimal, we have that $\Gamma, A_0 \models_T \lfloor \psi_0[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$ for all $\forall \overline{\mathbf{x}}_0.\psi_0 \in Q^+_{\mathcal{I}}(\lfloor \psi[\overline{\mathbf{k}}_0/\overline{\mathbf{x}}_0] \rfloor)$ where $(A_0, \overline{\mathbf{k}}_0) \leftrightharpoons \forall \overline{\mathbf{x}}_0.\psi_0$. For each such formula, by Lemma 3.2.6, $\Gamma \models_T \forall \overline{\mathbf{x}}_0.\psi_0$. Therefore, $\Gamma \cup \{(\neg)\lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor\}$ and $\Gamma \cup \{(\neg)\lceil \lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor \rceil\}$ are T-equivalent up to their shared variables, which by Lemma 3.2.5.(2) implies that $\Gamma \cup \{(\neg)\lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor\}$ and $\Gamma \cup \{(\neg)\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\}$ are T-equivalent up to their shared variables. □

Lemma 3.2.8. Given any $\mathsf{SL}(T)$-formula ϕ, $\mathsf{solve}_T(\lfloor \psi \rfloor^*)$ where ψ is $(\phi \triangleleft [\ell, \mathsf{pt}, C])\Downarrow$:

(1) Answers "unsat" only if ψ is T-unsatisfiable.

(2) Answers "sat" only if ψ is T-satisfiable.

(3) Terminates.

Proof. Assume $\mathsf{solve}_T(\Gamma_i)$ calls $\mathsf{solve}_T(\Gamma_{i+1})$ for $i \in [n-1]$, where n is finite and $\Gamma_0 = \lfloor \psi \rfloor^*$. By the definition of solve_T and Lemma 3.2.5 (2), it can be shown that $\lceil \Gamma_i \rceil$ and $\lceil \Gamma_{i+1} \rceil$ are equisatisfiable in T, and thus, by induction, $\lceil \Gamma_j \rceil$ and $\lceil \Gamma_k \rceil$ are equisatisfiable in T for each $j, k \in [n]$.

To show (1), assume without loss of generality, that $\mathsf{solve}_T(\Gamma_n)$ answers "unsat". Then Γ_n is T-unsatisfiable, and, by Lemma 3.2.5.(1), we have that $\lceil \Gamma_n \rceil$ is T-unsatisfiable. Thus, $\lceil \Gamma_0 \rceil$ is T-unsatisfiable, and thus, by Lemma 3.2.5.(2), ψ is T-unsatisfiable.

To show (2), assume without loss of generality that $\mathsf{solve}_T(\Gamma_n)$ answers "sat". Then Γ_n is T-satisfiable with model $(\mathcal{I}, \nu)$. We argue that $(\mathcal{I}, \nu)$ is a model for $\lceil \Gamma_n \rceil$, which implies that $\lceil \Gamma_0 \rceil$ is T-satisfiable, and thus by Lemma 3.2.5 (2), ψ is T-satisfiable. Assume that $(\mathcal{I}, \nu)$ is a model for Γ_n but not for $\lceil \Gamma_n \rceil$. Thus, $\nu(A) \neq (\forall \overline{\mathbf{x}}.\psi)^{\mathcal{I}}_{\nu}$ for some $\forall \overline{\mathbf{x}}.\psi \in \mathrm{Q}(\Gamma_n)$ where $(A, \overline{\mathbf{k}}) \leftrightharpoons \forall \overline{\mathbf{x}}.\psi$. In the case that $\nu(A) = \bot^{\mathcal{I}}$ and $(\forall \overline{\mathbf{x}}.\psi)^{\mathcal{I}}_{\nu} = \top^{\mathcal{I}}$, note that $(\mathcal{I}, \nu)$ is not a model for $(\neg A \Rightarrow \lfloor \neg\phi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor) \in \Gamma_n$. In the case that $\nu(A) = \top^{\mathcal{I}}$ and $(\forall \overline{\mathbf{x}}.\psi)^{\mathcal{I}}_{\nu} = \bot^{\mathcal{I}}$, by the definition of solve_T, we have $\Gamma_n \cup \{A\} \models_T \lfloor \psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}] \rfloor$. By Lemma 3.2.6, we have that $\Gamma_n \models_T \forall \overline{\mathbf{x}}.\psi$. Since $(\mathcal{I}, \nu)$ is a model of Γ_n, it must be the case that $(\forall \overline{\mathbf{x}}.\psi)^{\mathcal{I}}_{\nu} = \top^{\mathcal{I}}$, contradicting the fact that $(\forall \overline{\mathbf{x}}.\psi)^{\mathcal{I}}_{\nu} = \bot^{\mathcal{I}}$. Therefore, our assumption that $(\mathcal{I}, \nu)$ is not a model for $\lceil \Gamma_n \rceil$ is false.

To show (3), first note that the checks for T-satisfiability and T-entailment terminate, by assumption of a decision procedure for the T-satisfiability of quantifier-free formulae. Moreover, for each quantified formula $\forall \overline{\mathbf{x}}.\psi$, only a finite number of instances $A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}]$ exist for which the algorithm will add $\lfloor A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}] \rfloor^*$ to Γ, which implies that the algorithm

will consider only a finite number of quantified formulae, and for each only a finite number of instances will be added in this way. Thus, it suffices to show that the algorithm adds to Γ only sets $\lfloor A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}]\rfloor^*$ that are not a subset of Γ. Assume this is not the case for some $\lfloor A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}]\rfloor^*$, where $\forall\overline{\mathbf{x}}.\psi \in \mathrm{Q}(\Gamma_n)$, $\forall\overline{\mathbf{x}}.\psi$ is $\prec_{\Gamma,(\mathcal{I},\nu)}$-minimal, and $(A, \overline{\mathbf{k}}) \leftrightharpoons \forall\overline{\mathbf{x}}.\psi$. The terms $\overline{\mathbf{t}}$ meet the criteria in the procedure. In particular, for some model $(\mathcal{J},\gamma)$ of $\Gamma \cup \{\neg\lfloor\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\rfloor\}$ where $\gamma(A) = \top^{\mathcal{I}}$, we have $\overline{\mathbf{t}}_\gamma^{\mathcal{J}} = \overline{\mathbf{k}}_\gamma^{\mathcal{J}}$. Since $\gamma(A) = \top^{\mathcal{I}}$ and $(A \Rightarrow \lfloor\psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}]\rfloor) \in \lfloor A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}]\rfloor^* \subseteq \Gamma$, $(\mathcal{J},\gamma)$ is a model for $\lfloor\psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}]\rfloor$ and also $\neg\lfloor\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]\rfloor$. By Lemma 3.2.7, since $\forall\overline{\mathbf{x}}.\psi$ is $\prec_{\Gamma,(\mathcal{I},\nu)}$-minimal, $(\mathcal{J},\gamma)$ satisfies $\psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}]$ and $\neg\psi[\overline{\mathbf{k}}/\overline{\mathbf{x}}]$. Therefore, we obtain that $\overline{\mathbf{k}}_\gamma^{\mathcal{J}} \neq \overline{\mathbf{t}}_\gamma^{\mathcal{J}}$, which is a contradiction, thus our initial assumption is false. □

Theorem 3.2.9. Given any $\mathsf{SL}(T)$-formula ϕ, $\mathsf{solve}_{\mathsf{SL}(T)}(\phi)$:

(1) Answers "unsat" only if ϕ is $\mathsf{SL}(T)$-unsatisfiable.

(2) Answers "sat" only if ϕ is $\mathsf{SL}(T)$-satisfiable.

(3) Terminates.

Proof. To show (1), by Lemma 3.2.8 (1), it must be the case that $(\phi \triangleleft [\ell, \mathsf{pt}, C])\Downarrow$ is T-unsatisfiable. Thus, there cannot be heaps meeting the requirements of $\overline{\mathbf{h}'}$ and $\overline{\mathbf{h}''}$ in Lemma 3.2.1, and by that lemma we get that ϕ is $\mathsf{SL}(T)$-unsatisfiable.

To show (2), by Lemma 3.2.8 (2), it must be the case that $(\phi \triangleleft [\ell, \mathsf{pt}, C])\Downarrow$ has a T-model $(\mathcal{J},\gamma)$. Let h' be the heap with domain $\gamma(\ell)$ such that $h'(u) = \mathsf{pt}^{\mathcal{J}}(u)$ for all $u \in \gamma(\ell)$, and let $h'' = \mathsf{pt}^{\mathcal{J}}$. Since $\mathsf{Loc}^{\mathcal{J}}$ has infinite cardinality, and due to the structure of $(\phi \triangleleft [\ell, \mathsf{pt}, C])\Downarrow$, we may assume that $C^{\mathcal{J}} \subseteq \mathsf{Loc}^{\mathcal{J}} \setminus \mathrm{Pt}(\phi)_\gamma^{\mathcal{J}}$ and call this set L. Let $v = d^{\mathcal{J}}$. Consider $(\mathcal{I},\nu)$ such that $(\mathcal{J},\gamma) = (\mathcal{I}[\mathsf{pt} \hookleftarrow h''][C \leftarrow L][d \leftarrow v], \nu[\ell \leftarrow \mathrm{dom}(h')])$. Since we assumed $\mathcal{J},\gamma \models_T (\phi \triangleleft [\ell, \mathsf{pt}, C])\Downarrow$, then, by Lemma 3.2.4, there exists a heap h such that $\mathcal{I},\nu,h \models^{\mathsf{SL}} \phi$, and thus ϕ is $\mathsf{SL}(T)$-satisfiable.

Lastly, (3) is an immediate consequence of Lemma 3.2.8 (3). □

Theorem 3.2.9 establishes that $\mathsf{solve}_{\mathsf{SL}(T)}$ is a decision procedure for the $\mathsf{SL}(T)$-satisfiability of quantifier-free $\mathsf{SL}(T)$-formulae. A tight complexity bound for this problem is given by the following corollary.

Corollary 3.2.10. The $\mathsf{SL}(T)$-satisfiability problem is PSPACE-complete for any theory T whose satisfiability problem for the quantifier-free fragment is in PSPACE.

Proof. PSPACE-hardness can be obtained by reduction from QSAT, which generalizes [11, Definition 7] to our case. Let $\phi \equiv \forall x_1 \exists y_1 \ldots \forall x_n \exists y_n . \psi$ be an instance of QSAT, where ψ is a boolean combination of the variables x_i and y_i of sort Bool. We encode ϕ in $\mathsf{SL}(T)$ using the translation function Tr(.), defined recursively on the structure of ϕ as follows:

$$\begin{array}{rcl@{\qquad}rcl} \mathrm{Tr}(x_i) & \equiv & (t_i \mapsto d) * \top & \mathrm{Tr}(y_i) & \equiv & (u_i \mapsto d) * \top \\ \mathrm{Tr}(\neg\psi) & \equiv & \neg\mathrm{Tr}(\psi) & \mathrm{Tr}(\psi_1 \bullet \psi_2) & \equiv & \mathrm{Tr}(\psi_1) \bullet \mathrm{Tr}(\psi_2) \\ \mathrm{Tr}(\exists y_i \,.\, \psi) & \equiv & ((u_i \mapsto d) \vee \mathsf{emp}) * \mathrm{Tr}(\psi) & \mathrm{Tr}(\forall x_i \,.\, \psi) & \equiv & \neg(((t_i \mapsto d) \vee \mathsf{emp}) * \neg\mathrm{Tr}(\psi)) \end{array}$$

where d is constant of sort Data and $\bullet \in \{\wedge, \vee\}$. It is not hard to check that ϕ is satisfiable if and only if there exists a T-model $(\mathcal{I},\nu)$ and a heap h such that $\mathcal{I},\nu,h \models^{\mathsf{SL}} \mathrm{Tr}(\phi)$. To prove that $\mathsf{SL}(T)$-satisfiability is in PSPACE, we analyse the space complexity of the $\mathsf{solve}_{\mathsf{SL}(T)}$ algorithm. First, for any $\mathsf{SL}(T)$-formula ϕ, let $\mathrm{Size}(\phi)$ be the size of the syntax tree of ϕ. Clearly $\|\lfloor\phi\rfloor^*\| \leq \mathrm{Size}(\phi)$, and moreover, $\mathrm{Size}(\psi) \leq \mathrm{Size}(\phi)$ for each $\psi \in \lfloor\phi\rfloor^*$. Then a representation

of the set $\lfloor \phi \rfloor^*$ by simply enumerating its elements will take space at most $\mathrm{Size}(\phi)^2$. For a set of formulae Γ, let $\mathrm{Size}(\Gamma) = \sum_{\phi \in \Gamma} \mathrm{Size}(\phi)$ denote the size of its enumerative representation.

Secondly, it is not difficult to see that $|\phi|_{\mathsf{h}} \leq \mathrm{Size}(\phi)$ and $\|\mathrm{Pt}(\phi)\| \leq \mathrm{Size}(\phi)$, for any $\mathsf{SL}(T)$-formula ϕ. Then, for each subformula $\forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi$ of $\phi \triangleleft [\ell, \mathsf{pt}, C]\Downarrow$, we have $\|s\| \leq \mathrm{Size}(\phi)^2$ – in fact $\|s\| \leq \mathrm{Size}(\phi)$ if x is of sort $\mathsf{Set}(\mathsf{Loc})$ and $\|s\| \leq \mathrm{Size}(\phi)^2$ if x is of sort $\mathsf{Loc} \mapsto \mathsf{Data}$. Then there are at most $\mathrm{Size}(\phi)^2$ recursive calls on line 3 of the solve_T procedure, that corresponds to an instance of the subformula $\forall \overline{\mathbf{x}} \subseteq \overline{\mathbf{s}} \,.\, \psi$ of $\phi \triangleleft [\ell, \mathsf{pt}, C]\Downarrow$. Since there are at most $\mathrm{Size}(\phi)$ such subformulae, there are at most $\mathrm{Size}(\phi)^3$ recursive calls to solve_T with arguments $\Gamma_0, \ldots, \Gamma_n$, respectively. Moreover, we have $\Gamma_0 = \{\phi \triangleleft [\ell, \mathsf{pt}, C]\Downarrow\}$, thus $\mathrm{Size}(\Gamma_0) = \mathcal{O}(\mathrm{Size}(\phi))$ and $\mathrm{Size}(\Gamma_{i+1}) \leq \mathrm{Size}(\Gamma_i) + \mathrm{Size}(\phi)^2$ for each $i \in [n-1]$, because a set of formulae $\lfloor A \Rightarrow \psi[\overline{\mathbf{t}}/\overline{\mathbf{x}}] \rfloor^*$ of size at most $\mathrm{Size}(\phi)^2$ is added to Γ_i. Because T-satisfiability is in PSPACE, by the hypothesis, the checks at lines 1 and 2 can be done within space bounded by a polynomial in $\mathrm{Size}(\phi)$, thus the space needed by $\mathsf{solve}_{\mathsf{SL}(T)}(\phi)$ is also bounded by a polynomial in $\mathrm{Size}(\phi)$. Hence the $\mathsf{SL}(T)$-satisfiability problem is in PSPACE. □

In practice, the procedure $\mathsf{solve}_{\mathsf{SL}(T)}$ terminates in much less time that the theoretical worst-case complexity given by the above corollary. Our evaluation of the prototype implementation, described in Section 3.2.5, corroborates this fact. Moreover, the following example also illustrates this aspect.

Example 3.2.4. The translation and purification shown in Example 3.2.2 cover the steps taken by our $\mathsf{solve}_{\mathsf{SL}(T)}$ procedure before the call to solve_T. We now show what happens when the procedure solve_T is invoked, having $\Gamma_0 = \{\ell \approx \emptyset \wedge A \wedge \neg(y \approx \mathsf{nil}), \neg A \Rightarrow \neg\psi[\langle k_1, k_2\rangle / \langle \ell', \mathsf{pt}'\rangle]\}$ as input.

Step 1 determines that Γ_0 is T-satisfiable with a model $\mathcal{I}$ such that $A^{\mathcal{I}} = \top^{\mathcal{I}}$. Since $\Gamma_0 \cup \{A\} \models_T \psi[\langle k_1, k_2\rangle / \langle \ell', \mathsf{pt}'\rangle]$ does not hold, Step 2 of solve_T finds a model $\mathcal{J}$ for the set $\Gamma_0 \cup \{A, \neg\psi[\langle k_1, k_2\rangle / \langle \ell', \mathsf{pt}'\rangle]\}$. Note that

$$\begin{aligned}\neg\psi \equiv\ & \ell' \subseteq \{y, 0, 1, c_2\} \wedge \mathsf{pt}' \subseteq \{y, 0, 1, c_2\} \times \{y, 0, 1, d\}) \wedge \\ & \ell' \cap \ell \approx \emptyset \wedge \ell' \approx \{y\} \wedge \mathsf{ite}(y \in \ell', \mathsf{pt}'(y) \approx 0, \top) \wedge \neg(y \approx \mathsf{nil}) \wedge \\ & \neg(\ell' \cup \ell \approx \{y\} \ \wedge \mathsf{ite}(y \in \ell', \mathsf{pt}'(y) \approx 1, \mathsf{ite}(y \in \ell, \mathsf{pt}(y) \approx 1, \top)) \wedge \neg(y \approx \mathsf{nil}))\end{aligned}$$

We can choose $\langle t_1, t_2\rangle$ such that $t_1^{\mathcal{J}} = k_1^{\mathcal{J}} = \{y\}$ and $t_2^{\mathcal{J}} = k_2^{\mathcal{J}}$. It is required that $t_2 \subseteq \{y, 0, 1, c_2\} \times \{y, 0, 1, d\}$ and $t_2(y)^{\mathcal{J}} = 0^{\mathcal{J}}$. Then Step 3 recursively invokes solve_T on the set

$$\begin{aligned}\Gamma_1 &= \Gamma_0 \cup \lfloor A \Rightarrow \psi[\langle t_1, t_2\rangle \,/\, \langle \ell', \mathsf{pt}'\rangle] \rfloor^* \\ &\equiv \Gamma_0 \cup \{A \Rightarrow (\neg(y \approx \mathsf{nil}) \Rightarrow (\{y\} \approx \{y\} \wedge \mathsf{ite}(y \in \{y\}, 0 \approx 1, \mathsf{pt}(y) \approx 1) \wedge \neg(y \approx \mathsf{nil})))\} \\ &\equiv \Gamma_0 \cup \{A \Rightarrow (\neg(y \approx \mathsf{nil}) \Rightarrow \bot)\}\end{aligned}$$

The new formula added to Γ_1 contradicts $\ell \approx \emptyset \wedge A \wedge \neg(y \approx \mathsf{nil})$, thus the next call to solve_T will determine that the input set is T-unsatisfiable. Therefore, the input formula ϕ is also $\mathsf{SL}(T)$-unsatisfiable. ◀

3.2.4 Partial Support for Quantifiers

In many practical cases it is useful to check the validity of entailments between existentially quantified $\mathsf{SL}(T)$-formulae such as $\exists \mathbf{x} \,.\, \phi(\mathbf{x})$ and $\exists \mathbf{y} \,.\, \psi(\mathbf{y})$. Typically, this problem translates into a satisfiability query for an $\mathsf{SL}(T)$-formula $\exists \mathbf{x} \forall \mathbf{y} \,.\, \phi(\mathbf{x}) \wedge \neg\psi(\mathbf{y})$, with one quantifier alternation. A partial solution to this problem is to first check the satisfiability of ϕ. If ϕ

$$\mathsf{ls}_1(x,y,a) \leftarrow x \approx y \wedge \mathsf{emp} \mid b \approx a+10 \wedge x \mapsto (a,z), \mathsf{ls}_1(z,y,b)$$
$$\mathsf{ls}_2(x,y,a) \leftarrow x \approx y \wedge \mathsf{emp} \mid a \leq b \wedge x \mapsto (a,z), \mathsf{ls}_2(z,y,b)$$

$$\mathsf{tree}_1(x,a) \leftarrow x \approx \mathsf{nil} \wedge \mathsf{emp} \mid b \approx a-10 \wedge c \approx a+10 \wedge x \mapsto (a,y,z), \mathsf{tree}_1(y,b), \mathsf{tree}_1(z,c)$$
$$\mathsf{tree}_2(x,a) \leftarrow x \approx \mathsf{nil} \wedge \mathsf{emp} \mid b \leq a \wedge a \leq c \wedge x \mapsto (a,y,z), \mathsf{tree}_2(y,b), \mathsf{tree}_2(z,c)$$

$$\mathsf{pos}_1(x,a) \leftarrow x \mapsto a \mid x \mapsto a, \mathsf{pos}_1(y,b) \qquad \mathsf{neg}_3(x,a) \leftarrow x \mapsto a \mid x \mapsto a, \neg\mathsf{neg}_3(y,b)$$
$$\mathsf{pos}_2(x,y) \leftarrow x \mapsto y \mid x \mapsto y, \mathsf{pos}_2(y,z) \qquad \mathsf{neg}_4(x,a) \leftarrow x \mapsto a \mid \neg(x \mapsto a), \neg\mathsf{neg}_4(y,b)$$
$$\mathsf{neg}_1(x,a) \leftarrow \neg(x \mapsto a) \mid x \mapsto a, \mathsf{neg}_1(y,b) \qquad \mathsf{neg}_5(x,y) \leftarrow \neg(x \mapsto y) \mid x \mapsto y, \mathsf{neg}_5(y,z)$$
$$\mathsf{neg}_2(x,a) \leftarrow x \mapsto a \mid \neg(x \mapsto a), \mathsf{neg}_2(y,b) \qquad \mathsf{neg}_6(x,y) \leftarrow x \mapsto y \mid \neg(x \mapsto y), \mathsf{neg}_6(y,z)$$

Figure 3.4: Inductive predicates whose finite unfoldings are used in the experiments

is not satisfiable, the entailment holds trivially, so let us assume that ϕ has a model. We then check the satisfiability of $\phi \wedge \psi$. If unsatisfiable, then the entailment cannot hold, because there exists a model of ϕ which is not a model of ψ. Else, if $\phi \wedge \psi$ has a model, we add an equality $x \approx y$ for each pair of variables $(x,y) \in \mathbf{x} \times \mathbf{y}$ that are mapped to the same term in this model, the result being a conjunction $E(\mathbf{x},\mathbf{y})$ of equalities. Finally, we check the satisfiability of the formula $E \wedge \phi \wedge \neg\psi$. If this formula is unsatisfiable, the entailment is valid, otherwise, the test is inconclusive. This method was applied manually for all entailments in our evaluation set, described in Section 3.2.5. A general procedure for quantifier instantiation in the context of the $\exists^*\forall^*$-quantified fragment of $\mathsf{SL}(T)$ is introduced in Section 3.3.

3.2.5 Experimental Evaluation

We tested our implementation of the $\mathsf{SL}(T)$-satisfiability procedure in CVC4 version 1.5[1]. on two kinds of benchmarks. All experiments were run on a 2.80GHz Intel® Core™ i7-4600U CPU machine with 8MB of cache.

The first part of our experiments focuses on finite unfoldings of the inductive predicates shown in Figure 3.4, mostly inspired by existing benchmarks, such as SL-COMP'14 [46]. For instance, unfolding ls_1 once results in the formula

$$x \approx y \wedge \mathsf{emp} \vee \exists z \exists b \,.\, (b = a+10 \wedge x \mapsto (a,z)) * (z \approx y \wedge \mathsf{emp})$$

while unfolding it twice gives the formula

$$x \approx y \wedge \mathsf{emp} \vee$$
$$\exists z \exists b \,.\, (b \approx a+10 \wedge x \mapsto (a,z)) * (z \approx y \wedge \mathsf{emp} \vee (c = b+10 \wedge z \mapsto (b,w)) * (w \approx y \wedge \mathsf{emp}))$$

Since our decision procedure targets the quantifier-free fragment of $\mathsf{SL}(T)$, for each entailment we unfold both predicates using the same variables. We checked the validity of the entailment between **LHS** and **RHS**, where both predicates are unfolded $n=1$, $n=2$, $n=3$, $n=4$ and $n=8$ times, by applying the manual instantiation method described in Section 3.2.4. The entailment in which **LHS** is $\mathsf{ls}_1(x,y,a)$, **RHS** is $\mathsf{ls}_2(x,y,a)$ and $n=1$ is, actually,

$$\begin{array}{rcl} \exists z \exists b \,.\, x \approx y \wedge \mathsf{emp} \vee & \models^{\mathsf{SL}} & \exists z' \exists b' \,.\, x \approx y \wedge \mathsf{emp} \vee \\ (b \approx a+10 \wedge x \mapsto (a,z)) * (z \approx y \wedge \mathsf{emp}) & & (a \leq b' \wedge x \mapsto (a,z')) * (z' \approx y \wedge \mathsf{emp}) \end{array}$$

[1]Available at `http://cvc4.cs.nyu.edu/web/`.

and the final formula being checked by the decision procedure is

$$z' \approx z \wedge b' \approx b \wedge (x \approx y \wedge \mathsf{emp} \vee (b \approx a + 10 \wedge x \mapsto (a, z)) * (z \approx y \wedge \mathsf{emp})) \wedge \\ \neg(x \approx y \wedge \mathsf{emp} \vee (a \leq b' \wedge x \mapsto (a, z')) * (z' \approx y \wedge \mathsf{emp}))$$

The second part of our experiments uses verification conditions automatically generated by applying the weakest precondition calculus of [29] several times to the program loops depicted in Figure 3.5. These verification conditions are of the forms $\phi \models^{\mathsf{SL}} \mathsf{wp}(\mathbf{l}, \phi)$ and $\phi \models^{\mathsf{SL}} \mathsf{wp}^n(\mathbf{l}, \phi)$, where $\mathsf{wp}(\mathbf{l}, \phi)$ denotes the weakest precondition of the SL formula ϕ with respect to the sequence of statements $\mathbf{l}$, and $\mathsf{wp}^n(\mathbf{l}, \phi) = \mathsf{wp}(\mathbf{l}, \ldots \mathsf{wp}(\mathbf{l}, \mathsf{wp}(\mathbf{l}, \phi)) \ldots)$ denotes the iterative application of the weakest precondition calculus n times in a row. For each loop $\mathbf{l}$ we also consider the variant $\mathbf{zl}$ as well, which, through the assertions on line 2 of each loop, tests that the data values contained within the memory cells are 0. The postconditions are specified using lists of specific lengths, defined by the predicates list^n and zlist^n with $n \geq 0$. The manual instantiation method described in Section 3.2.4 was again used to eliminate the quantified variables in these verification conditions.

```
1: while w ≠ nil do
2:     assert(w.data = 0)
3:     v := w;
4:     w := w.next;
5:     dispose(v);
6:     do
```

(z)disp

```
1: while u ≠ nil do
2:     assert(u.data = 0)
3:     w := u.next;
4:     u.next := v;
5:     v := u;
6:     u := w;
7:     do
```

(z)rev

$$\mathsf{list}^0(x) \leftarrow x \approx \mathsf{nil} \wedge \mathsf{emp} \qquad \mathsf{zlist}^0(x) \leftarrow x \approx \mathsf{nil} \wedge \mathsf{emp}$$
$$\mathsf{list}^n(x) \leftarrow \exists y \,.\, x \mapsto y * \mathsf{list}^{n-1}(y) \qquad \mathsf{zlist}^n(x) \leftarrow \exists y \,.\, x \mapsto (0, y) * \mathsf{zlist}^{n-1}(y)$$

Figure 3.5: Program loops and the list definitions on which they operate

As an example, $\mathsf{list}^2(w)$ is the formula $\exists u \exists v \,.\, w \mapsto u * u \mapsto v * (v \approx \mathsf{nil} \wedge \mathsf{emp})$ and the weakest precondition calculus $\mathsf{wp}^2(\mathbf{disp}, \mathsf{emp} \wedge w = \mathsf{nil})$ yields

$$\exists w_1 \exists w_2 \exists w_3 \exists w_4 \,.\, ((((w_2 \approx \mathsf{nil} \wedge \mathsf{emp}) * w_4 \mapsto w_1) \wedge (w_4 \mapsto w_2 * \top)) * w \mapsto w_3) \wedge (w \mapsto w_4 * \top)$$

Then the final formula checked by our decision procedure is

$$w_1 \approx v \wedge w_2 \approx v \wedge w_3 \approx u \wedge w_4 \approx u \wedge (w \mapsto u * u \mapsto v * (v \approx \mathsf{nil} \wedge \mathsf{emp})) \wedge \\ \neg((((w_2 \approx \mathsf{nil} \wedge \mathsf{emp}) * w_4 \mapsto w_1) \wedge (w_4 \mapsto w_2 * \top)) * w \mapsto w_3) \wedge (w \mapsto w_4 * \top)$$

For a majority of benchmarks, the runtime of CVC4 is quite low, with the exception of the $n = 4$ and $n = 8$ cases of the entailments between tree_1 and tree_2 formulae, which resulted in a timeout after 300 seconds due to the exponential blow-up in the size of the formulae and the number of variables used. For such cases in which CVC4 times out, the performance bottleneck resides in its ground decision procedure for finite sets, indicating efficient support for this theory is important for our approach to separation logic.

LHS	RHS	$n=1$	$n=2$	$n=3$	$n=4$	$n=8$
Unfoldings of inductive predicates						
$\mathsf{ls}_1(x,y,a)$	$\mathsf{ls}_2(x,y,a)$	unsat <0.01s	unsat <0.01s	unsat <0.01s	unsat 0.01s	unsat 0.01s
$\mathsf{tree}_1(x,a)$	$\mathsf{tree}_2(x,a)$	unsat <0.01s	unsat 0.06s	unsat 1.89s	timeout >300s	timeout >300s
$\mathsf{pos}_1(x,a)$	$\mathsf{neg}_1(x,a)$	unsat 0.02s	unsat 0.04s	unsat 0.11s	unsat 0.25 s	unsat 3.01s
$\mathsf{pos}_1(x,a)$	$\mathsf{neg}_2(x,a)$	unsat 0.01s	unsat 0.05s	unsat 0.11s	unsat 0.23s	unsat 2.10s
$\mathsf{pos}_1(x,a)$	$\mathsf{neg}_3(x,a)$	unsat 0.02s	unsat 0.07s	unsat 0.24s	unsat 0.46s	unsat 4.05s
$\mathsf{pos}_1(x,a)$	$\mathsf{neg}_4(x,a)$	unsat 0.05s	sat 0.24s	unsat 0.33s	sat 2.77s	sat 24.72s
$\mathsf{pos}_2(x,y)$	$\mathsf{neg}_5(x,y)$	unsat 0.02s	unsat 0.05s	unsat 0.14s	unsat 0.32s	unsat 3.69s
$\mathsf{pos}_2(x,y)$	$\mathsf{neg}_6(x,y)$	sat 0.02s	unsat 0.04s	unsat 0.13s	unsat 0.27s	unsat 2.22s
Verification conditions						
$\mathsf{list}^n(w)$	$\mathsf{wp}(\mathbf{disp}, \mathsf{list}^{n-1}(w))$	<0.01s	0.02s	0.05s	0.12s	1.97s
$\mathsf{list}^n(w)$	$\mathsf{wp}^n(\mathbf{disp}, \mathsf{list}^0(w)$	<0.01s	0.02s	0.12s	0.41s	22.97s
$\mathsf{zlist}^n(w)$	$\mathsf{wp}(\mathbf{zdisp}, \mathsf{zlist}^{n-1}(w))$	0.01s	0.02s	0.05s	0.11s	1.34s
$\mathsf{zlist}^n(w)$	$\mathsf{wp}^n(\mathbf{zdisp}, \mathsf{zlist}^0(w)$	0.01s	0.02s	0.11s	0.43s	24.13s
$\mathsf{list}^n(u) * \mathsf{list}^0(v)$	$\mathsf{wp}(\mathbf{rev}, \mathsf{list}^{n-1}(u) * \mathsf{list}^1(v))$	0.06s	0.08s	0.14s	0.30s	2.83s
$\mathsf{list}^n(u) * \mathsf{list}^0(v)$	$\mathsf{wp}^n(\mathbf{rev}, \mathsf{list}^0(u) * \mathsf{list}^n(v))$	0.06s	0.12s	0.56s	1.75s	27.82s
$\mathsf{zlist}^n(u) * \mathsf{zlist}^0(v)$	$\mathsf{wp}(\mathbf{zrev}, \mathsf{zlist}^{n-1}(u) * \mathsf{zlist}^1(v))$	0.22s	0.04s	0.12s	0.25s	2.16s
$\mathsf{zlist}^n(u) * \mathsf{zlist}^0(v)$	$\mathsf{wp}^n(\mathbf{zrev}, \mathsf{zlist}^0(u) * \mathsf{zlist}^n(v))$	0.04s	0.10s	0.41s	1.27s	20.26s

Table 3.1: Experimental results of our procedure for quantifier-free $\mathsf{SL}(T)$ inputs

3.3 The Bernays-Schönfinkel-Ramsey Fragment of $\mathsf{SL}(T)$

The Bernays-Schönfinkel-Ramsey class refers to the $\exists^*\forall^*$-quantified fragment of first-order logic with equality and uninterpreted predicate symbols, but without any function symbols [33]. We use this name to analogously refer to the $\exists^*\forall^*$-quantified fragment of $\mathsf{SL}(T)$ and consider formulae $\phi \equiv \exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \,.\, \psi(x_1, \ldots, x_m, y_1, \ldots, y_n)$, where ψ is any quantifier-free $\mathsf{SL}(T)$-formula and the quantified variables range over the set of memory locations. Then ϕ is satisfiable if and only if its *functional form* $\forall y_1 \ldots \forall y_n \,.\, \psi[c_1/x_1, \ldots, c_m/x_m]$ is satisfiable, where $c_1, \ldots, c_m$ are fresh (Skolem) constant symbols.

We explicitly refer to the sorts for memory locations and data by writing $\mathsf{SL}(T)_{\mathsf{Loc,Data}}$, and we denote the Bernays-Schönfinkel-Ramsey fragment by $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc,Data}}$. For programs using pointer arithmetic, as it is the case of C or C++, it is useful to consider LIA (Liniar Integer Arithmetic) for the theory of memory addresses. Otherwise, if the program only performs equality checks on the values of the pointers, as in Java, we can use E (Equality with Uninterpreted Functions) for this purpose.

3.3.1 Decidability and complexity results

We consider the satisfiability of the $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$ fragment in the following cases:

1. Loc is interpreted as the sort U of E and Data as U^k, for some $k \geq 1$. In general, the satisfiability problem for $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ with $k \geq 2$ is undecidable [16, Theorem 1]. The root cause of this undecidability result is the occurrence of universally quantified variables within the scope of a separating implication, which, moreover, appears under an even number of negations. We can recover decidability when U is finite or countably infinite by considering more restricted fragments in which we forbid universally quantified variables from appearing under the scope of any positive separating implication or, respectively, forbid any positive occurrence of separating implications whatsoever [16, Theorems 2 and 3]. In this cases, the satisfiability problem for $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ becomes PSPACE-complete. In consequence, satisfiability for the $-\!\!*$-free fragment of $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ when U is either finite or countably infinite is also PSPACE-complete.

2. Both Loc and Data are interpreted as Int, equipped with addition and total order. Then the satisfiability problem for $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$ is undecidable;

3. Loc is interpreted as the sort U of E, and Data as $U \times \mathsf{Int}$. Then the satisfiability problem for $\exists^*\forall^*\mathsf{SL}(\mathsf{ELIA})_{U,U\times\mathsf{Int}}$ is undecidable.

The results for cases 2 and 3 rely on an undecidability argument for a fragment of Presburger arithmetic with one monadic predicate symbol, interpreted over finite sets. We denote by $(\exists^*\forall^* \cap \forall^*\exists^*)$-$\mathsf{LIA}$ the set of conjunctions between two linear arithmetic formulae, one $\exists^*\forall^*$-quantified, and the other $\forall^*\exists^*$-quantified.

Theorem 3.3.1. The satisfiability problem is undecidable for the fragment $(\exists^*\forall^* \cap \forall^*\exists^*) - \mathsf{LIA}$, with one monadic predicate symbol, interpreted over finite sets of integers.

Proof. We reduce from the following variant of *Hilbert's 10th Problem*: given a multivariate Diophantine polynomial $R(x_1, \ldots, x_n)$, the problem "does $R(x_1, \ldots, x_n) = 0$ have a solution in $\mathbb{N}^n$?" is undecidable [35].

By introducing sufficiently many free variables, we encode $R(x_1, \ldots, x_n) = 0$ as an equisatisfiable Diophantine system of degree at most two, containing only equations of the form $x = yz$ (resp. $x = y^2$) and linear equations $\sum_{i=1}^{k} a_i x_i = b$, where $a_1, \ldots, a_k, b \in \mathbb{Z}$. Next, we replace each equation of the form $x = yz$, with y and z distinct variables, with the quadratic system $2x + t_y + t_z = t_{y+z} \wedge t_y = y^2 \wedge t_z = z^2 \wedge t_{y+z} = (y+z)^2$, where t_y, t_z and t_{y+z} are fresh (free) variables. In this way, we replace all multiplications between distinct variables by occurrences of the squaring function. Let $\Psi_{R(x_1,\ldots,x_n)=0}$ be the conjunction of the above equations. It is manifest that $R(x_1, \ldots, x_n) = 0$ has a solution in $\mathbb{N}^n$ iff $\Psi_{R(x_1,\ldots,x_n)=0}$ is satisfiable, with all free variables ranging over $\mathbb{N}$.

Now we introduce a monadic predicate symbol P, which is intended to denote a (possibly finite) set of consecutive perfect squares, starting with 0. To capture this definition, we require the following:

$$\begin{array}{l} P(0) \wedge P(1) \wedge \forall x \forall y \forall z \,.\, P(x) \wedge P(y) \wedge P(z) \wedge x < y < z \wedge \\ (\forall u \,.\, x < u < y \vee y < u < z \Rightarrow \neg P(u)) \Rightarrow z - y = y - x + 2 \end{array} \qquad (\mathsf{sqr})$$

Observe that this formula is a weakening of the definition of the infinite set of perfect squares given by Halpern [20], from which the conjunct $\forall x \exists y \,.\, y > x \wedge P(y)$, requiring that P is an

infinite set of natural numbers, has been dropped. Moreover, notice that sqr has quantifier prefix $\forall^3\exists$, due to the fact that $\forall u$ occurs implicitly under negation, on the left-hand side of an implication. If P is interpreted as a finite set $P^{\mathcal{I}} = \{p_0, p_1, \ldots, p_N\}$ such that (w.l.o.g.) $p_0 < p_1 < \ldots < p_N$, it is easy to show, by induction on $N > 0$, that $p_i = i^2$, for all $i = 0, 1, \ldots, N$.

The next step is encoding the squaring function using the monadic predicate P. This is done by replacing each atomic proposition $x = y^2$ in $\Psi_{R(x_1,\ldots,x_n)=0}$ by the formula $\theta_{x=y^2} \equiv P(x) \wedge P(x+2y+1) \wedge \forall z \,.\, x < z < x+2y+1 \Rightarrow \neg P(z)$.

Fact 1. For each interpretation $\mathcal{I}$ and valuation $\nu \in \mathcal{V}_{\mathcal{I}}$ mapping x and y into $\mathbb{N}$, $\mathcal{I}, \nu \models x = y^2$ if and only if $\mathcal{I}$ can be extended to an interpretation of P as a finite set of consecutive perfect squares such that $\mathcal{I}, \nu \models \theta_{x=y^2}$.

Proof of Fact 1. For the direct implication, if $\mathcal{I}, \nu \models x = y^2$ then we have $(\nu(y)+1)^2 = \nu(x) + 2\nu(y) + 1$. Let $P^{\mathcal{I}}$ be the set $\{0, 1, \ldots, (\nu(y)+1)^2\}$. Clearly $\nu(x), \nu(x) + 2\nu(y) + 1 \in P^{\mathcal{I}}$ and, since they are consecutive perfect squares, every number in between $\nu(x)$ and $\nu(x)+2\nu(y)+1$ does not belong to $P^{\mathcal{I}}$. Thus $\mathcal{I}, \nu \models \theta_{x=y^2}$. Considering the reverse implication, if $\mathcal{I}, \nu \models \theta_{x=y^2}$ and $P^{\mathcal{I}}$ is a set of consecutive perfect squares, it follows that $\nu(x)$ and $\nu(x)+2\nu(y)+1$ are consecutive perfect squares, i.e. $\nu(x) = n^2$ and $\nu(x)+2\nu(y)+1 = (n+1)^2$ for some $n \in \mathbb{N}$. Then $\nu(y) = n$, thus $\mathcal{I}, \nu \models x = y^2$.

Let $\Phi_{R(x_1,\ldots,x_n)=0}$ be the conjunction of sqr with the formula obtained by replacing each atomic proposition $x = y^2$ with $\theta_{x=y^2}$ in $\Psi_{R(x_1,\ldots,x_n)=0}$. Observe that each universally quantified variable in $\Phi_{R(x_1,\ldots,x_n)=0}$ occurs either in sqr or in some $\theta_{x=y^2}$, and moreover, each $\theta_{x=y^2}$ belongs to the $\exists^*\forall^*$ fragment of LIA. $\Phi_{R(x_1,\ldots,x_n)=0}$ belongs thus to the $\exists^*\forall^* \cap \forall^*\exists^*$ fragment of LIA, with P being the only monadic predicate symbol. Finally, we prove that $R(x_1, \ldots, x_n) = 0$ has a solution in $\mathbb{N}^n$ iff $\Phi_{R(x_1,\ldots,x_n)=0}$ is satisfiable.

"$\Rightarrow$" Let $\mathcal{I}$ be an interpretation $\mathcal{I}$ and $\nu \in \mathcal{V}_{\mathcal{I}}$ be a valuation mapping $x_1, \ldots, x_n$ into $\mathbb{N}$, such that $\mathcal{I}, \nu \models R(x_1, \ldots, x_n) = 0$. Obviously, ν can be extended to a model of $\Psi_{R(x_1,\ldots,x_n)=0}$ by assigning $\nu(t_x) = (\nu(x))^2$ for all auxiliary variables t_x occurring in $\Psi_{R(x_1,\ldots,x_n)=0}$. We extend $(\mathcal{I}, \nu)$ to a model of $\Phi_{R(x_1,\ldots,x_n)=0}$ by assigning $P^{\mathcal{I}} = \{n^2 \mid 0 \leq n \leq \sqrt{m}\}$, where $m = \max\{(\nu(x)+1)^2 \mid x \in \mathrm{FV}(\Psi_{R(x_1,\ldots,x_n)=0})\}$. Clearly $P^{\mathcal{I}}$ meets the requirements of sqr. By Fact 1, we obtain that $\mathcal{I}, \nu \models \theta_{x=y^2}$ for each subformula $\theta_{x=y^2}$ of $\Phi_{R(x_1,\ldots,x_n)=0}$, thus $\mathcal{I}, \nu \models \Phi_{R(x_1,\ldots,x_n)=0}$.

"$\Leftarrow$" If $\mathcal{I}, \nu \models \Psi_{R(x_1,\ldots,x_n)=0}$ then, by sqr, $P^{\mathcal{I}}$ is a set of consecutive perfect squares, and, by Fact 1, $\mathcal{I}, \nu \models x = y^2$ for each subformula $\theta_{x=y^2}$ of $\Phi_{R(x_1,\ldots,x_n)=0}$. Then $\mathcal{I}, \nu \models \Psi_{R(x_1,\ldots,x_n)=0}$ and, consequently, $\mathcal{I}, \nu \models R(x_1, \ldots, x_n) = 0$. □

Consider now the satisfiability problem for the fragment $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$ where both Loc and Data are the Int sort, equipped with addition and total order. In this case, the heap consists of a set of lists with possible aliases and circularities. Without loss of generality, we consider that Int is the set of positive integers – extending the interpretation of Loc to include negative integers does not make any difference for the undecidability result.

The above theorem cannot be directly used for the undecidability of $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$, by interpreting the (unique) monadic predicate as the (finite) domain of the heap. This is due to the sqr formula, which defines the interpretation of the monadic predicate as a set of consecutive perfect squares $0, 1, \ldots, n^2$, and whose quantifier prefix belongs to the $\forall^*\exists^*$

fragment. We overcome this problem by replacing the sqr formula above with a definition of such sets in $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$. Consider first the following SL properties: [6]:

$$\begin{array}{rcl} \sharp x \geq 1 & \equiv & \exists u \,.\, u \mapsto x * \top \\ \sharp x \leq 1 & \equiv & \forall u \forall t \,.\, \neg(u \mapsto x * t \mapsto x * \top) \end{array}$$

Intuitively, $\sharp x \geq 1$ states that x has at least one predecessor in the heap and $\sharp x \leq 1$ states that x has at most one predecessor. We use $\sharp x = 0$ and $\sharp x = 1$ as shorthands for $\neg(\sharp x \geq 1)$ and $\sharp x \geq 1 \wedge \sharp x \leq 1$, respectively. The formula below states that the heap can be decomposed into a list segment, starting with x and ending in y, and several disjoint cyclic lists:

$$\begin{array}{rcl} x \overset{\circlearrowleft}{\rightarrow}^+ y & \equiv & \sharp x = 0 \wedge \mathsf{alloc}(x) \wedge \sharp y = 1 \wedge \neg\mathsf{alloc}(y) \wedge \\ & & \forall z \,.\, \neg(z \approx y) \Rightarrow (\sharp z = 1 \Rightarrow \mathsf{alloc}(z)) \wedge \forall z \,.\, \sharp z \leq 1 \end{array}$$

We forbid the existence of circular lists by adding the following arithmetic constraint:

$$\forall u \forall t \,.\, u \mapsto t * \top \Rightarrow u < t \qquad (\mathsf{nocyc})$$

We also ask that the elements of the list segment starting in x are consecutive perfect squares:

$$\mathsf{consqr}(x) \equiv x = 0 \wedge x \mapsto 1 * \top \wedge \forall z \forall u \forall t \,.\, z \mapsto u * u \mapsto t * \top \Rightarrow t - u = u - z + 2 \quad (\mathsf{consqr})$$

The formula $\exists x \exists y \,.\, x \overset{\circlearrowleft}{\rightarrow}^+ y \wedge \mathsf{nocyc} \wedge \mathsf{consqr}(x)$ belongs to the $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$ fragment.

Theorem 3.3.2. The satisfiability problem for $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$ is undecidable.

Proof. We use the same reduction as in the proof of Theorem 3.3.1, but replace sqr by $\exists x \exists y \,.\, x \overset{\circlearrowleft}{\rightarrow}^+ y \wedge \mathsf{nocyc} \wedge \mathsf{consqr}(x)$, and define $\theta_{x=y^2}$ as $\mathsf{alloc}(x) \wedge \mathsf{alloc}(x + 2y + 1) \wedge \forall z \,.\, x < z < x + 2y + 1 \Rightarrow \neg\mathsf{alloc}(z)$. □

Finally, we consider a variation on the previous undecidability result, where locations are the (uninterpreted) sort U of E and the data consists of tuples of sort $U \times \mathsf{Int}$. This fragment of SL can be useful when reasoning about lists with integer data. Its undecidability can be proved in a manner similar with Theorem 3.3.2.

Theorem 3.3.3. The satisfiability problem for $\exists^*\forall^*\mathsf{SL}(\mathsf{ELIA})_{U,U\times\mathsf{Int}}$ is undecidable.

Proof. We redefine the following shorthands:

$$\begin{array}{rcl} \sharp \mathsf{x} \geq 1 & \equiv & \exists u^U \exists d^{\mathsf{Int}} \,.\, u \mapsto (d, \mathsf{x}) * \top \\ \sharp \mathsf{x} \leq 1 & \equiv & \forall u^U \forall t^U \forall d^{\mathsf{Int}} \,.\, \neg(u \mapsto (d, \mathsf{x}) * t \mapsto (d, \mathsf{x}) * \top) \\ \\ \mathsf{nocyc} & \equiv & \forall u^U \forall t^U \forall v^U \forall d^{\mathsf{Int}} \forall e^{\mathsf{Int}} \,.\, u \mapsto (d, t) * t \mapsto (e, v) * \top \Rightarrow d < e \\ \\ \mathsf{consqr}(\mathsf{x}) & \equiv & \exists \mathsf{y}^U \exists \mathsf{z}^U \,.\, \mathsf{x} \mapsto (0, \mathsf{y}) * \mathsf{y} \mapsto (1, \mathsf{z}) * \top \wedge \\ & & \forall u^U \forall t^U \forall v^U \forall w^U \forall d^{\mathsf{Int}} \forall e^{\mathsf{Int}} \forall f^{\mathsf{Int}} \,.\, u \mapsto (d, t) * t \mapsto (e, v) * v \mapsto (f, w) * \top \Rightarrow \\ & & f - e = e - d + 2 \end{array}$$

The proof is similar to the one of Theorem 3.3.1, using the above shorthands. □

3.3.2 A Semi-decision Procedure for $\exists^*\forall^*\mathsf{SL}(T)$ in SMT

We present a procedure for the satisfiability of $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ formulae implemented in the SMT solver CVC4 and building upon the one given in Section 3.2.3, designed for quantifier-free $\mathsf{SL}(T)_{\mathsf{Loc,Data}}$ inputs, where T is a first-order theory who has a decidable satisfiability problem for quantifier-free T-constraints. Similar to other existing approaches for quantified formulae in SMT [19, 40], our procedure uses incremental quantifier instantiation based on candidate models returned by a solver for quantifier-free inputs.

$\mathsf{solve}(\exists\overline{\mathbf{x}}\,\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{x}},\overline{\mathbf{y}}))$ where $\overline{\mathbf{x}} = \langle x_1,\ldots,x_m\rangle$ and $\overline{\mathbf{y}} = \langle y_1,\ldots,y_n\rangle$:

Let $\overline{\mathbf{k}} = \langle k_1,\ldots,k_m\rangle$ and $\overline{\mathbf{e}} = \langle e_1,\ldots,e_n\rangle$ be tuples of fresh constants with the same sorts as $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$, respectively

Let $L = L' \cup \mathsf{set}(\overline{\mathbf{k}})$ and L' be a set of fresh constants with $\|L'\| = |\phi(\overline{\mathbf{x}},\overline{\mathbf{y}})|_{\mathsf{h}} + n$.

Return $\mathsf{solve\_rec}(\exists\overline{\mathbf{x}}\,\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{x}},\overline{\mathbf{y}}),\emptyset,L)$.

$\mathsf{solve\_rec}(\exists\overline{\mathbf{x}}\,\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{x}},\overline{\mathbf{y}}),\Gamma,L)$:

1. If Γ is $\mathsf{SL}(\mathsf{E})$-unsat, return "unsat".
2. Assume $\exists\overline{\mathbf{x}}\,\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{x}},\overline{\mathbf{y}})$ is equivalent to $\exists\overline{\mathbf{x}}.(\forall\overline{\mathbf{y}}.\,\phi_1(\overline{\mathbf{x}},\overline{\mathbf{y}}) \wedge \ldots \wedge \forall\overline{\mathbf{y}}.\,\phi_p(\overline{\mathbf{x}},\overline{\mathbf{y}}))$.

   If $\Gamma'_j = \Gamma \cup \{\neg\phi_j(\overline{\mathbf{k}},\overline{\mathbf{e}}) \wedge \bigwedge_{i=1}^{n}\bigvee_{t\in L} e_i \approx t\}$ is $\mathsf{SL}(\mathsf{E})$-unsat for all $j \in [p]$, return "sat".
3. Otherwise, let $\mathcal{I},\nu,h \models^{\mathsf{SL}} \bigwedge\Gamma'_j$ for some $j \in [p]$.

   Let $\overline{\mathbf{t}} = \langle t_1,\ldots,t_n\rangle$ be a tuple such that $e_i^{\mathcal{I}} = t_i^{\mathcal{I}}$ and $t_i \in L$ for each $i \in [n]$.

   Return $\mathsf{solve\_rec}(\exists\overline{\mathbf{x}}\,\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{x}},\overline{\mathbf{y}}),\Gamma \cup \{\phi_j(\overline{\mathbf{k}},\overline{\mathbf{t}})\},L)$.

Figure 3.6: A counterexample-guided procedure for $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ formulae

Our counterexample-guided approach for establishing the satisfiability of $\exists\overline{\mathbf{x}}\,\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{x}},\overline{\mathbf{y}})$ is described in Figure 3.6. We first introduce the tuples $\overline{\mathbf{k}}$ and $\overline{\mathbf{e}}$, containing fresh constants of the same type as $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$, respectively. The basis of the procedure lies in finding a set of instantiations of $\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{k}},\overline{\mathbf{y}})$ that are either collectively unsatisfiable or are satisfiable and entail our input. We then construct a set L which consists of the union between the constants in $\overline{\mathbf{k}}$ and a set L' of fresh constants, whose cardinality is equal to $|\phi(\overline{\mathbf{x}},\overline{\mathbf{y}})|_{\mathsf{h}}$ plus the number n of universal variables in our input. As such, L represents a finite set of terms from which the instantiations of $\overline{\mathbf{y}}$ in $\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{k}},\overline{\mathbf{y}})$ can be built.

We then invoke the recursive subprocedure $\mathsf{solve\_rec}$ on the initially empty set of formulae Γ and the set of constants L, which will incrementally add instances of $\forall\overline{\mathbf{y}}.\,\phi(\overline{\mathbf{k}},\overline{\mathbf{y}})$ to Γ. Step 1 checks whether Γ is $\mathsf{SL}(\mathsf{E})$-unsatisfiable using the procedure from Section 3.2.3. If this is the case, then our input is also $\mathsf{SL}(\mathsf{E})$-unsatisfiable and the procedure returns "unsat".

Otherwise, Step 2 considers the *miniscoped* form $\exists\overline{\mathbf{x}}.\,(\forall\overline{\mathbf{y}}.\,\phi_1(\overline{\mathbf{x}},\overline{\mathbf{y}}) \wedge \ldots \wedge \forall\overline{\mathbf{y}}.\,\phi_p(\overline{\mathbf{x}},\overline{\mathbf{y}}))$ of our input, where the quantification over $\overline{\mathbf{x}}$ is distributed over conjunctions. Note that we may omit quantification on conjuncts ϕ_j that do not contain variables from $\overline{\mathbf{y}}$. Given this

miniscoped formula, we construct a set Γ'_j, for each $j \in [p]$, containing Γ, the negation of ϕ_j in which $\overline{\mathbf{y}}$ is replaced by the fresh constants in $\overline{\mathbf{e}}$, and a conjunction of constraints requiring each e_i to be equal to at least one term in L for $i = [n]$. If Γ'_j is $\mathsf{SL}(\mathsf{E})$-unsatisfiable for all $j \in [p]$, we can conclude that our input is $\mathsf{SL}(\mathsf{E})$-satisfiable.

Otherwise, given an interpretation $\mathcal{I}$, a valuation $\nu \in \mathcal{V}_\mathcal{I}$ and heap h satisfying $\bigwedge \Gamma'_j$, Step 3 constructs a tuple of terms $\overline{\mathbf{t}} = \langle t_1, \ldots, t_n \rangle$ that is used to instantiate $\forall \overline{\mathbf{y}}.\, \phi_j(\overline{\mathbf{k}}, \overline{\mathbf{y}})$. For each $i \in [n]$, we choose a constant t_i from L, whose interpretation under $\mathcal{I}$ coincides with the one of e_i. The existence of such a t_i is guaranteed by the equality constraints from Γ'_j introduced in Step 2 and by the fact that $\mathcal{I}$ satisfies $\bigwedge \Gamma'_j$, together with ν and h. This selection helps ensure that, in each iteration, the instantiations are unique and are chosen from a finite set of possibilities. The formula $\phi_j(\overline{\mathbf{k}}, \overline{\mathbf{t}})$ is added to Γ and solve_rec is invoked recursively. In practice, for both unsatisfiable and satisfiable inputs, the procedure terminates before considering all $\overline{\mathbf{t}}$ from L^n for each $\forall \overline{\mathbf{y}}.\, \phi_j(\overline{\mathbf{x}}, \overline{\mathbf{y}})$.

Theorem 3.3.4. Let U be an uninterpreted sort belonging to the signature of E. Given any $\exists^* \forall^* \mathsf{SL}(\mathsf{E})_{U,U^k}$ formula ψ of the form $\exists \overline{\mathbf{x}} \forall \overline{\mathbf{y}}.\, \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, $\mathsf{solve}(\psi)$:

(1) Answers "unsat" only if ψ is $\mathsf{SL}(\mathsf{E})$-unsatisfiable.

(2) Terminates.

Proof. For point (1), note that Γ contains only formulae $\phi_j(\overline{\mathbf{k}}, \overline{\mathbf{t}})$, which are consequences of our input. Therefore, when Γ is $\mathsf{SL}(\mathsf{E})$-unsatisfiable, our input is $\mathsf{SL}(\mathsf{E})$-unsatisfiable as well.

For point (2), clearly only a finite number of possible formulae can be added to Γ as a result of the procedure, since all terms $\overline{\mathbf{t}}$ belong to the finite set L and p is finite. Furthermore, on every iteration, for any j, $\mathcal{I}$ satisfies Γ and $\neg \phi_j(\overline{\mathbf{k}}, \overline{\mathbf{e}})$. Since $e_i^\mathcal{I} = t_i^\mathcal{I}$ for each $i = 1, \ldots, n$, we have that $\phi_j(\overline{\mathbf{k}}, \overline{\mathbf{t}}) \notin \Gamma$, and thus a new formula is added to Γ on every call. Only a finite number of recursive calls are made to solve_rec. Since the $\mathsf{SL}(\mathsf{E})$-satisfiability of quantifier-free is decidable, all steps in the procedure are terminating, and thus solve terminates. □

We discuss a few important details regarding our implementation of the procedure.

Miniscoping. In practice, the universal quantification in our input $\exists \overline{\mathbf{x}} \forall \overline{\mathbf{y}}.\, \phi(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ may be transformed into a formula of the form $\exists \overline{\mathbf{x}}.\, \psi_1 \wedge \ldots \wedge \psi_k$, where each $\psi_i \equiv \forall \overline{\mathbf{z}}.\, \varphi_i(\overline{\mathbf{x}}, \overline{\mathbf{z}})$ for a (possibly empty) tuple $\overline{\mathbf{z}}$ of variables that are a subset of those in $\overline{\mathbf{y}}$, and φ_i is quantifier-free. Our procedure then instantiates each of these quantified formulae independently in the manner described in Figure 3.6. In each iteration, an instance of some $\forall \overline{\mathbf{z}}.\, \varphi_i(\overline{\mathbf{x}}, \overline{\mathbf{z}})$ is added to Γ. The procedure terminates with "sat" only when the set $\Gamma \cup \{\varphi_i(\overline{\mathbf{k}}, \overline{\mathbf{z}})[\overline{\mathbf{y}}/\overline{\mathbf{e}}] \wedge \bigwedge_{i=1}^{n} \bigvee_{t \in L} e_i \approx t\}$ is $\mathsf{SL}(\mathsf{E})$-unsatisfiable for all $i \in [k]$.

Matching heuristics. When constructing the terms $\overline{\mathbf{t}}$ for instantiation, it is possible that there exists $i \in [n]$ such that $e_i^\mathcal{I} = u^\mathcal{I}$ for multiple $u \in L$. In this case, the procedure will choose one such u for instantiation. To increase the likelihood of the instantiation being relevant to the satisfiability of our input, we use heuristics for selecting the best possible u among those whose interpretation is equal to e_i in $\mathcal{I}$. In particular, if $e_i^\mathcal{I} = u_1^\mathcal{I} = u_2^\mathcal{I}$, and Γ' contains atoms of the form $e_i \mapsto v$ and $u_1 \mapsto v_1$ for some v, v_1 where $v^\mathcal{I} = v_1^\mathcal{I}$ but no atom of the form $u_2 \mapsto v_2$ for some v_2 where $v^\mathcal{I} = v_2^\mathcal{I}$, then we strictly prefer u_1 over u_2 when choosing t_i for e_i.

Finding minimal models. Efficient techniques for finding small models for uninterpreted sorts in CVC4 have been developed in previous work [44]. We have found these techniques beneficial to the performance of the procedure in Figure 3.6. We use these techniques to find interpretations $\mathcal{I}$ in solve_rec that map U to a finite set of minimal size. When combined with the matching heuristics discussed above, these techniques lead to finding useful instantiations more quickly, since more constants are constrained to be equal to e_i, for $i \in [n]$, under the interpretations $\mathcal{I}$.

Symmetry breaking. The procedure in Figure 3.6 introduces a set of fresh constants L, which in turn introduce the possibility of discovering interpretations $\mathcal{I}$ that are isomorphic – i.e. identical up to the renaming of constants in L'. We introduce additional constraints in Γ, which do not affect its satisfiability, but reduce the number of isomorphic models. We consider an ordering $\prec$ on the constants from L', and add constraints that ensure that, for all models $(\mathcal{I}, \nu, h)$ of Γ, if $\ell_1^{\mathcal{I}} \notin \mathrm{dom}(h)$ then $\ell_2^{\mathcal{I}} \notin \mathrm{dom}(h)$ for all ℓ_2 such that $\ell_1 \prec \ell_2$.

Example 3.3.1. Consider the entailment $\neg(x \approx y) \wedge x \mapsto z \models^{\mathsf{SL}} \exists u \,.\, x \mapsto u$, where x, y, z, u are of sort U of E. It is valid if and only if the formula $\exists x \exists y \exists z \forall u \,.\, \neg(x \approx y) \wedge x \mapsto z \wedge \neg x \mapsto u$ is $\mathsf{SL}(\mathsf{E})$-unsatisfiable. Since $|\neg(x \approx y) \wedge x \mapsto z \wedge \neg x \mapsto u|_{\mathsf{h}} = 1$, a run of the procedure in Figure 3.6 on this input constructs $\overline{\mathbf{k}} = \langle k_x, k_y, k_z \rangle$, $\overline{\mathbf{e}} = \langle e_u \rangle$ and $L = \{k_x, k_y, k_z, \ell_1, \ell_2\}$. We then invoke solve_rec where Γ is initially empty. By miniscoping, our input is equivalent to $\exists x \exists y \exists z \,.\, \neg(x \approx y) \wedge x \mapsto z \wedge \forall u \,.\, \neg x \mapsto u$. On the first two recursive calls to solve_rec, we may add $\neg(k_x \approx k_y)$ and $k_x \mapsto k_z$ to Γ by trivial instantiation of the first two conjuncts. On the third recursive call, Γ is $\mathsf{SL}(\mathsf{E})$-satisfiable, and we check the satisfiability of:

$$\Gamma' = \{\neg(k_x \approx k_y), k_x \mapsto k_z, k_x \mapsto e_u \wedge (e_u \approx k_x \vee e_u \approx k_y \vee e_u \approx k_z \vee e_u \approx \ell_1 \vee e_u \approx \ell_2)\}$$

Since $k_x \mapsto k_z$ and $k_x \mapsto e_u$ are in Γ', for all interpretations $\mathcal{I}$, valuations $\nu \in \mathcal{V}_{\mathcal{I}}$ and heaps h such that $\mathcal{I}, \nu, h \models^{\mathsf{SL}} \Gamma'$ we have $e_u^{\mathcal{I}} = k_z^{\mathcal{I}}$. Since $k_z \in L$, we may choose to add the instantiation $\neg k_x \mapsto k_z$ to Γ. In consequence, Γ is $\mathsf{SL}(\mathsf{E})$-unsatisfiable on the next recursive call to solve_rec. Thus, our input is $\mathsf{SL}(\mathsf{E})$-unsatisfiable and the entailment is valid. ◀

A modified version of the procedure in Figure 3.6 can be used for establishing the satisfiability of $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$ formulae for theories T beyond equality, and where Loc and Data are not restricted to uninterpreted sorts. In such cases, solve_rec cannot restrict interpretations $\mathcal{I}$ to map each e_i to the same value as a member of the finite set L. Therefore,

$$\begin{aligned}
&\mathsf{ls}_3(x,y) \leftarrow x \approx y \wedge \mathsf{emp} \mid \neg(x \approx y) \wedge x \mapsto z, \mathsf{ls}_3(z,y)\\
&\mathsf{ls}_4(x,y) \leftarrow x \approx y \wedge \mathsf{emp} \mid x \mapsto z, \mathsf{ls}_4(z,y)\\
&\mathsf{tree}_3(x) \leftarrow x \approx \mathsf{nil} \wedge \mathsf{emp} \mid \neg(l \approx r) \wedge x \mapsto (l,r), \mathsf{tree}_3(l), \mathsf{tree}_3(r)\\
&\mathsf{tree}_4(x) \leftarrow x \approx \mathsf{nil} \wedge \mathsf{emp} \mid x \mapsto (l,r), \mathsf{tree}_4(l), \mathsf{tree}_4(r)\\
&\mathsf{ts}_1(x,y) \leftarrow x \approx y \wedge \mathsf{emp} \mid \neg(x \approx y) \wedge x \mapsto (l,r), \mathsf{ts}_1(l,y), \mathsf{tree}_4(r)\\
&\qquad\qquad\qquad\qquad\quad \mid \neg(x \approx y) \wedge x \mapsto (l,r), \mathsf{tree}_4(l), \mathsf{ts}_1(r,y)\\
&\mathsf{ts}_2(x,y) \leftarrow x \approx y \wedge \mathsf{emp} \mid x \mapsto (l,r), \mathsf{ts}_2(l,y), \mathsf{tree}_4(r)\\
&\qquad\qquad\qquad\qquad\quad \mid x \mapsto (l,r), \mathsf{tree}_4(l), \mathsf{ts}_2(r,y)
\end{aligned}$$

Figure 3.7: Inductive predicates whose finite unfoldings are used in the experiments

solve_rec requires a modification that omits the equality constraint between the elements of $\overline{\mathbf{e}}$ and L in Step 2. This modified procedure is still sound, but is no longer terminating in general. Nevertheless, it can still be used as a heuristic for $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc,Data}}$-satisfiability.

3.3.3 Experimental Evaluation

The solve procedure from Figure 3.6 was implemented within the CVC4 SMT solver and tested on similar benchmarks as the ones presented in Section 3.2.5: entailments between predicate unfoldings and verification conditions obtained from the weakest precondition calculus. In the former case, we preserved the entailments involving pos_1, pos_2, neg_1, neg_2, neg_5 and neg_6, while also introducing some new predicates, whose definitions are shown in Figure 3.7. In the latter case, the benchmarks remained the same. All experiments were run on the same 2.80GHz Intel® Core™ i7 CPU machine with 8MB of cache.

LHS	**RHS**		$n=1$	$n=2$	$n=3$	$n=4$	$n=8$
Unfoldings of inductive predicates							
$\mathsf{ls}_3(x,y)$	$\mathsf{ls}_4(x,y)$	solve	< 0.01s	0.02s	0.03s	0.05s	0.21s
		manual	< 0.01s	< 0.01s	< 0.01s	< 0.01s	< 0.01s
$\mathsf{tree}_3(x)$	$\mathsf{tree}_4(x)$	solve	< 0.01s	0.04s	1.43s	23.42s	> 300s
		manual	< 0.01s	< 0.01s	< 0.01s	< 0.01s	0.09s
$\mathsf{ts}_1(x,a)$	$\mathsf{ts}_2(x,a)$	solve	< 0.01s	0.81s	> 300s	> 300s	> 300s
		manual	< 0.01s	0.03s	103.89s	> 300s	> 300s
$\mathsf{pos}_1(x,a)$	$\mathsf{neg}_1(x,a)$	solve	0.34s	0.01s	0.31s	0.76s	21.19s
		manual	0.04s	0.05s	0.08s	0.12s	0.53s
$\mathsf{pos}_1(x,a)$	$\mathsf{neg}_2(x,a)$	solve	0.03s	0.12s	0.23s	0.46s	3.60s
		manual	0.05s	0.08s	0.08s	0.12s	0.54s
$\mathsf{pos}_2(x,a)$	$\mathsf{neg}_5(x,a)$	solve	0.04s	0.13s	0.28s	0.48s	4.20s
		manual	0.01s	0.03s	0.05s	0.09s	0.45s
$\mathsf{pos}_2(x,a)$	$\mathsf{neg}_6(x,a)$	solve	—	0.08s	0.15s	0.26s	1.33s
		manual	—	0.03s	0.06s	0.09s	0.46s
Verification conditions							
$\mathsf{list}^n(w)$	$\mathsf{wp}(\mathbf{disp}, \mathsf{list}^{n-1}(w))$	solve	0.01s	0.03s	0.08s	0.19s	1.47s
		manual	< 0.01s	0.01s	0.02s	0.05s	0.26s
$\mathsf{list}^n(w)$	$\mathsf{wp}^n(\mathbf{disp}, \mathsf{list}^0(w))$	solve	0.01s	0.06s	0.17s	0.53s	7.08s
		manual	< 0.01s	0.02s	0.08s	0.14s	2.26s
$\mathsf{zlist}^n(w)$	$\mathsf{wp}(\mathbf{zdisp}, \mathsf{zlist}^{n-1}(w))$	solve	0.04s	0.05s	0.09s	0.19s	1.25s
		manual	< 0.01s	0.01s	0.02s	0.04s	0.29s
$\mathsf{zlist}^n(w)$	$\mathsf{wp}^n(\mathbf{zdisp}, \mathsf{zlist}^0(w))$	solve	0.01s	0.10s	0.32s	0.87s	11.88s
		manual	0.01s	0.02s	0.07s	0.15s	2.20s
$\mathsf{list}^n(u) * \mathsf{list}^0(v)$	$\mathsf{wp}(\mathbf{rev}, \mathsf{list}^{n-1}(u) * \mathsf{list}^1(v))$	solve	0.38s	0.06s	0.11s	0.16s	0.56s
		manual	0.07s	0.03s	0.07s	0.11s	0.43s
$\mathsf{list}^n(u) * \mathsf{list}^0(v)$	$\mathsf{wp}^n(\mathbf{rev}, \mathsf{list}^0(u) * \mathsf{list}^n(v))$	solve	0.38s	0.07s	0.30s	68.68s	> 300s
		manual	0.08s	0.06s	0.11s	0.23s	1.79s
$\mathsf{zlist}^n(u) * \mathsf{zlist}^0(v)$	$\mathsf{wp}(\mathbf{zrev}, \mathsf{zlist}^{n-1}(u) * \mathsf{zlist}^1(v))$	solve	0.22s	0.07s	0.15s	0.21s	0.75s
		manual	0.04s	0.02s	0.04s	0.06s	0.31s
$\mathsf{zlist}^n(u) * \mathsf{zlist}^0(v)$	$\mathsf{wp}^n(\mathbf{zrev}, \mathsf{zlist}^0(u) * \mathsf{zlist}^n(v))$	solve	0.23s	0.09s	0.17s	0.30s	2.06s
		manual	0.04s	0.02s	0.05s	0.09s	0.48s

Table 3.2: Experimental results of our procedure for $\exists^*\forall^*$-quantified $\mathsf{SL}(T)$ inputs

We compared the performance of this procedure with the results of applying our previous

CVC4 decision procedure for the quantifier-free $\mathsf{SL}(T)$ inputs to a variant of the benchmarks obtained by manual quantifier instantiation, as described in Section 3.2.4. For each set of benchmarks, Table 3.2 lists the results of both methods. Note that the entailment between pos_2 and neg_4 when $n = 1$ is skipped because it is not valid – since the negated formula is satisfiable, we cannot generate the manual instantiation.

Compared to checking the manual instantiation, the fully automated solver was less than 0.5 seconds slower on 72% of the test cases, and less than 1 second slower on 79% of the test cases. The automated solver experienced 3 timeouts in cases where the manual instantiation succeeds – for the entailments between tree_3 and tree_4 with $n = 8$, ts_1 and ts_2 with $n = 3$, $\mathsf{list}^n(u) * \mathsf{list}^0(v)$ and $\mathsf{wp}^n(\mathbf{rev}, \mathsf{list}^0(u) * \mathsf{list}^n(v))$ with $n = 8$. These timeouts are caused by the first call to the quantifier-free $\mathsf{SL}(T)$ decision procedure, which fails to produce a model in less than 300 seconds (time not accounted for in the manually produced instance of the problem).

Chapter 4

An Inductive Entailment Checker for Separation Logic

In this chapter we describe `Inductor`, an entailment checker tool that implements the proof-search semi-algorithm 1 from Section 2.2, using the set $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ of inference rules for inductive entailments in separation logic, introduced in Section 2.3.2. `Inductor` is written in C++ and uses the DPLL(T)-based SMT solver CVC4 [3] as a back-end that it queries in order to establish the satisfiability of $\mathsf{SL}(T)$-formulae, in which the occurrences of inductive predicates are treated as uninterpreted functions. More specifically, these queries are handled by the decision procedures provided in Chapter 3 and integrated into CVC4.

Some key implementation details are provided in Section 4.1. We also analyse several case studies in Sections 4.2–4.5, which portray the behaviour of our tool for a series of interesting inputs. These case studies demonstrate how proofs and counterexamples are obtained, while also illustrating the range of problems `Inductor` can handle.

4.1 Implementation Details

The inputs mainly handled by `Inductor` are SMT-LIB scripts, abiding by the SMT-LIB Standard: Version 2.6 [4]. Theory and logic files are loaded automatically, based on the logic set in the input script being handled. Additionally, proof strategies are specified as nondeterministic finite word automata, in a language similar to that accepted by libVATA [32]. The front-end interprets these input files using custom parsers constructed with Flex[1] and Bison[2], which generate abstract syntax trees. In the case of SMT-LIB scripts, an additional well-sortedness check can be performed to ensure their correctness. The abstract syntax trees are then transformed into hierarchies specifically tailored for inductive predicate entailments in separation logic and proof strategies, respectively.

4.1.1 Defining SL Inductive Predicates in SMT-LIB

In order to use SL connectives in the predicate definitions expressed in SMT-LIB, we declare them as function symbols in a new theory, depicted in Listing 4.1. Specifying the

[1] Flex – The Fast Lexical Analyzer, `https://github.com/westes/flex`

[2] GNU Bison – The Yacc-compatible Parser Generator, `https://www.gnu.org/software/bison`

`:left-assoc` and `:right-assoc` attributes for `sep` and `wand` allows them to be used for more than two arguments. The `pto` and `nil` function symbols are parametric with respect to the location and data sorts.

```
(theory SepLogic :funs ( (emp Bool)
                         (sep Bool Bool Bool :left-assoc)
                         (wand Bool Bool Bool :right-assoc)
                         (par (Loc Data) (pto Loc Data Bool))
                         (par (Loc) (nil Loc)) ) )
```

Listing 4.1: An SMT-LIB theory declaration for separation logic

This new theory may be combined with some already existing ones, such as `Ints` to obtain the `SEPLOGLIA` logic in Listing 4.2. Such a logic can then be set in an SMT-LIB script to indicate the theories in which its content is expressed.

```
(logic SEPLOGLIA :theories (SepLogic Ints) )
```

Listing 4.2: An SMT-LIB logic declaration for combining `SepLogic` with `Ints`

Inductive predicates can be defined as recursive functions, using two available commands: `define-fun-rec` for a single predicate, and `define-funs-rec` for (possibly) multiple mutually recursive predicates. A simple definition for a singly-linked list with integer memory locations is shown in Listing 4.3.

```
(set-logic SEPLOGLIA)

(define-fun-rec ls ((x Int) (y Int)) Bool
    (or  (and (= x y) emp)
         (exists ((z Int)) (sep (pto x z) (ls z y))) ) )
```

Listing 4.3: The definition of a list segment in SMT-LIB

Whenever memory locations need to point to tuples of values, such as in the case of a tree definition, we declare an additional datatype with constructors and selectors for each of its fields. Two commands are at our disposal for this purpose: `declare-datatype` for a single algebraic datatype, and `declare-datatypes` for (possibly) multiple mutually recursive datatypes. A simple binary tree is defined in this manner in Listing 4.4.

```
(set-logic SEPLOGLIA)

(declare-datatypes ((Node 0)) (((node (left Int) (right Int)))))

(define-fun-rec tree ((x Int)) Bool
    (or  (and (= x (as nil Int)) emp)
         (exists ((l Int) (r Int)) (sep (pto x (node l r)) (tree l) (tree r))) ) )
```

Listing 4.4: The definition of a binary tree in SMT-LIB

4.1.2 Specifying Proof Strategies as Automata

As previously mentioned, `Inductor` can also accept a regular expression representing a proof strategy as input – if no proof strategy is given, then **S** from Theorem 2.5.16 will be used as default. By Kleene's Theorem, it is known that, given a regular expression, there exists an equivalent nondeterministic finite word automaton (NFA), possibly with ε-transitions (NFA-ε). Figure 4.1 depicts a straightforward NFA-ε that is equivalent to **S**.

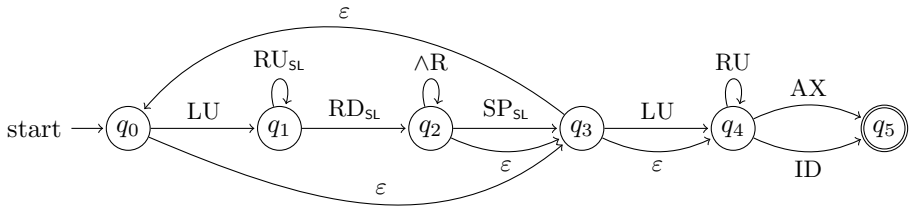


Figure 4.1: An NFA-ε equivalent to our default proof strategy **S**

We are more interested in such a representation because, after applying a certain inference rule, we want to easily check which inference rules that comply with the strategy could be applied next. However, given an NFA-ε, the ε-transitions are cumbersome and, thus, we prefer an equivalent NFA – which is guaranteed to exist, since the two classes of automata are known to be equivalent.

As such, the proof strategies that `Inductor` accepts as input are given as NFA, rather than regular expressions. The definition of such an NFA is specified in a language inspired by the simplicity of the one used by libVATA and whose grammar is depicted in Listing 4.5.

```
<file>         :  'Rules' <rule_list> <automaton>
<rule_list>    :  <rule> <rule> ...
<rule>         :  string
<automaton>    :  'Automaton' string 'States' <state_list> 'Initial State' <state>
                  'Final States' <state_list> 'Transitions' <trans_list>
<state_list>   :  <state> <state> ...
<state>        :  string
<trans_list>   :  <trans> <trans> ...
<trans>        :  '(' <state> ',' <rule> ')' '->' <state>
```

Listing 4.5: Grammar for files specifying proof strategies as NFA

Using this language, Listing 4.6 defines an NFA that is equivalent with the NFA-ε from Figure 4.1, and consequently, is also equivalent with our default proof search strategy.

```
Rules LU RU RD RI SP ID AX
Automaton Default    States q0 q1 q2 q3 q4 q5   Initial state q0    Final states q5
Transitions
(q0, LU) -> q1   (q1, RD) -> q0   (q2, RU) -> q4   (q2, SP) -> q3   (q3, AX) -> q5
(q0, LU) -> q4   (q1, RD) -> q2   (q2, RI) -> q0   (q2, SP) -> q4   (q3, ID) -> q5
(q0, RU) -> q4   (q1, RD) -> q3   (q2, RI) -> q2   (q2, AX) -> q5   (q4, RU) -> q4
(q0, AX) -> q5   (q1, RD) -> q4   (q2, RI) -> q3   (q2, ID) -> q5   (q4, AX) -> q5
(q0, ID) -> q5   (q2, LU) -> q1   (q2, RI) -> q4   (q3, LU) -> q4   (q4, ID) -> q5
(q1, RU) -> q1   (q2, LU) -> q4   (q2, SP) -> q0   (q3, RU) -> q4
```

Listing 4.6: The definition of the NFA corresponding to the default proof search strategy

4.1.3 A Breadth-First Proof Search Implementation

The proof search method sketched by semi-algorithm 1 is reliant on the choice of *IR* made at line 14. Whenever there are more than one applicable inference rules, only one is selected and the rest are discarded. Furthermore, as is the case for $\mathrm{SP}_{\mathsf{SL}}$, some inference rules can have multiple possible instances for the same sequent, where only one is required to succeed in obtaining a proof. Algorithm 1 again only chooses one of them. In our implementation, we wanted to explore all the potential derivations resulting from the inference rule instances available at any point. Moreover, since we use a queue for the nodes still needing to be explored, we generate derivations in a breadth-first fashion. Thus, any proof or counterexample we obtain is guaranteed to result from the shortest possible paths.

Consequently, we use a different tree-like structure to compactly store all the derivations explored at any given moment. This structure accepts two types of nodes, depicted in Figure 4.2, which represent sequents (`SNode`) and inference rule instances (`RNode`), respectively. The node types alternate in the tree, thus an `SNode` only has `RNode` children, and vice-versa.

```
SNode { sequent  : A sequent Γ ⊢ Δ,
        states   : A list of states in the search strategy
        parent   : RNode parent of the current node
        children : A list with children of type RNode     }

RNode { rule     : An inference rule schema
        pivot    : SNode pivot for this instance of rule,
        parent   : SNode parent of the current node,
        children : A list with children of type SNode     }
```

Figure 4.2: The data structures representing sequents and inference rule instances

With these new data structures, we say that an inference rule $IR \in \mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$ is applicable on a given `SNode` N whenever there exists an instance *ir* of *IR* for which: (i) the consequent of *ir* matches $\mathsf{N}.\mathit{sequent}$ and the pivot of *ir* (if it exists) matches $\mathsf{A}.\mathit{sequent}$, for some `SNode` ancestor A of N, such that the side conditions of *ir* are satisfied, and (ii) if $\mathsf{R}_1, \ldots, \mathsf{R}_n$ is the `RNode` sequence extracted from the path starting at A and ending at N, then $\mathsf{R}_1.\mathit{rule} \cdot \ldots \cdot \mathsf{R}_n.\mathit{rule}$ satisfies the pivot condition of *ir* (if it exists).

Both types of nodes are marked with either a *Closed*, *Failed* or *Unknown* status. By default, all nodes are initially *Unknown*. The status of an `SNode` can be changed to *Closed* whenever: (i) its sequent is $\top$, or (ii) at least one of its `RNode` children is *Closed*. An `RNode` becomes *Closed* when all of its `SNode` children are *Closed*. Conversely, an `SNode` is marked as *Failed* whenever: (i) its sequent is of the form $\Gamma \vdash \emptyset$, or (ii) all of its `RNode` children are *Failed*. An `RNode` is marked *Failed* when at least one of its `SNode` children is *Failed*. Changing the status of a node always prompts a status update for all of its ancestors.

Algorithm 2 provides a sketch of our proof search implementation for $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$, which, given an input sequent $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$ and an NFA $\mathsf{Strat} = (Q_{\mathsf{Strat}}, \mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}, \Delta_{\mathsf{Strat}}, q_0, F_{\mathsf{Strat}})$ describing the proof strategy, explores all possible derivations rooted at the input sequent. We construct a node Root, with which we initialise the work queue. While the work queue is not empty and the status of Root is *Unknown*, we dequeue an `SNode` N. We denote by Q^{IR}_{N} the set of states in Strat towards which we transition from $\mathsf{N}.\mathit{states}$ by applying *IR*, and build a set $\mathcal{R}_{\mathsf{N}}$ of applicable inference rule schema that are also accepted by the strategy.

If $\mathrm{AX}_{\mathsf{SL}}$ or ID are in $\mathcal{R}_{\mathsf{N}}$ and, moreover, their application leads Strat to transition to some

Algorithm 2: A sketch of our exhaustive proof search implementation for $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$

Input : An SL inductive system $\mathcal{S}$, a basic sequent $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$, a proof strategy $\mathsf{Strat} = (Q_{\mathsf{Strat}}, \mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}, \Delta_{\mathsf{Strat}}, q_0, F_{\mathsf{Strat}})$ represented as an NFA

Output: `VALID` and a proof starting with $p(\overline{\mathbf{x}}) \vdash q_1(\overline{\mathbf{x}}), \ldots, q_n(\overline{\mathbf{x}})$;
`INVALID` and a counterexample for $p(\overline{\mathbf{x}}) \models^{\mathsf{SL}}_{\mathcal{S}} q_1(\overline{\mathbf{x}}) \vee \ldots \vee q_n(\overline{\mathbf{x}})$;
`UNKNOWN` and the proof search tree constructed by the algorithm

```
Root ← SNode(p(x̄) ⊢ q_1(x̄), ..., q_n(x̄), [q_0], null, [])
WorkQueue ← {Root}
while WorkQueue ≠ [] and Root is Unknown do
    Dequeue a node N from WorkQueue
    Let Q_N^IR = {q' | (q, IR) → q' ∈ Δ_Strat and q ∈ N.states} for any IR ∈ R^SL_Ind
    Let R_N = {IR | Q_N^IR ≠ ∅ and IR applicable on N}
    if AX_SL ∈ R_N and Q_N^{AX_SL} ∩ F_Strat ≠ ∅ then
        R ← RNode(AX_SL, null, N, [])
        N' ← SNode(⊤, Q_N^{AX_SL}, R, [])
        Add R to N.children
        Add N' to R.children and mark it as Closed
    else if ID ∈ R_N and Q_N^ID ∩ F_Strat ≠ ∅ then
        R ← RNode(ID, A, N, []) for some ancestor A of N that is pivot for ID
        N' ← SNode(⊤, Q_N^ID, R, [])
        Add R to N.children
        Add N' to R.children and mark it as Closed
    else
        foreach instance ir of each IR ∈ R_N do
            R ← RNode(IR, null, N, [])
            Add R to N.children
            Let k be the number of antecedents generated by ir
            if k = 0 and Q_N^IR ∩ F_Strat ≠ ∅ then
                N' ← SNode(⊤, Q_N^IR, R, [])
                Add N' to R.children and mark it as Closed
            foreach antecedent Γ_i ⊢ Δ_i of ir with i ∈ [k] do
                N_i ← SNode(Γ_i ⊢ Δ_i, Q_N^IR, R, [])
                Add N_i to R.children
                if Δ_i = ∅ then
                    Mark N_i as Failed
            if R is not Failed then
                Enqueue N_1, ..., N_k in WorkQueue
if Root is Closed then
    return VALID and ExtractProof(Root)
else if Root is Failed then
    return INVALID and ExtractCounterexamples(Root)
else
    return UNKNOWN and Root
```

final states, then this branch of the derivation has been successful. We add a $\top$ leaf, which is marked as *Closed*.

Otherwise, for each inference rule $IR \in \mathcal{R}_{\mathsf{N}}$ we consider each instance ir of IR with antecedents $\Gamma_1 \vdash \Delta_1, \ldots, \Gamma_k \vdash \Delta_k$. If $k = 0$ and, by applying IR, we reach some final state in Strat, then this branch has been successful. As before, we add a $\top$ leaf that we mark as *Closed*. Otherwise, if $k > 1$, we create an `RNode` R for ir and an `SNode` N_i for each of its antecedents. If $\Delta_i = \emptyset$ for some $i \in [k]$, then N_i is marked as *Failed*. If this is not the case for any $i \in [k]$, then we add $\mathsf{N}_1, \ldots, \mathsf{N}_k$ to the work queue and continue.

When the status of Root changes to *Closed*, then a proof has been obtained. The proof is extracted from the proof search tree and offered as a certificate. Otherwise, if it changes to *Failed*, then at least one counterexample has been discovered. We extract the counterexamples from the proof search tree and give them as witnesses. If the work queue becomes empty, but the status of Root is still *Unknown*, then the proof search was inconclusive and our entire proof search tree is returned as justification.

4.2 Case Study: Binary Trees

Consider the following definitions for binary trees. The predicate $tree$ accepts any binary tree as model, $tree_1^+$ accepts binary trees with at least one node, and $tree_2^+$ accepts binary trees with at least one node in which the children of any node are either both allocated or both nil. Note that $\mathcal{S}_t$ is ranked, non-filtering, non-overlapping and has the fvi property.

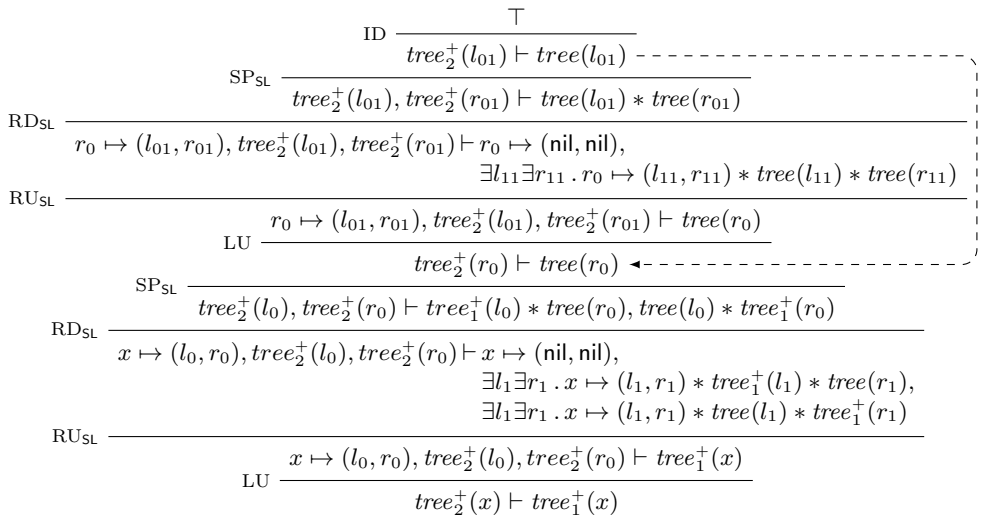
$$
\begin{aligned}
tree(x) &\leftarrow_{\mathcal{S}_t} x \approx \mathsf{nil} \wedge \mathsf{emp} \mid x \mapsto (l, r), tree(l), tree(r) \\
tree_1^+(x) &\leftarrow_{\mathcal{S}_t} x \mapsto (\mathsf{nil}, \mathsf{nil}) \mid x \mapsto (l, r), tree_1^+(l), tree(r) \\
&\qquad\qquad\qquad\qquad\quad \mid x \mapsto (l, r), tree(l), tree_1^+(r) \\
tree_2^+(x) &\leftarrow_{\mathcal{S}_t} x \mapsto (\mathsf{nil}, \mathsf{nil}) \mid x \mapsto (l, r), tree_2^+(l), tree_2^+(r)
\end{aligned}
$$

The equivalent SMT-LIB definitions for $tree$, $tree_1^+$ and $tree_2^+$ are provided in Listing 4.7.

```
(set-logic SEPLOGLIA)
(declare-datatypes ((Node 0)) (((node (left Int) (right Int)))))

(define-fun-rec tree ((x Int)) Bool
    (or  (and (= x (as nil Int)) emp)
         (exists ((l Int) (r Int)) (sep (pto x (node l r)) (tree l) (tree r))) ) )

(define-fun-rec treep1 ((x Int)) Bool
    (or  (pto x (node (as nil Int) (as nil Int)))
         (exists ((l Int) (r Int)) (sep (pto x (node l r)) (treep1 l) (tree r)))
         (exists ((l Int) (r Int)) (sep (pto x (node l r)) (tree l) (treep1 r))) ) )

(define-fun-rec treep2 ((x Int)) Bool
    (or  (pto x (node (as nil Int) (as nil Int)))
         (exists ((l Int) (r Int)) (sep (pto x (node l r)) (treep2 l) (treep2 r))) ))
```

Listing 4.7: Definitions of binary trees in SMT-LIB

The entailments $tree_1^+ \models^{\mathsf{SL}}_{\mathcal{S}_t} tree$, $tree_2^+ \models^{\mathsf{SL}}_{\mathcal{S}} tree$ and $tree_2^+ \models^{\mathsf{SL}}_{\mathcal{S}_t} tree_1^+$ hold, facts corroborated by `Inductor`. A branch of the proof for $tree_2^+(x) \vdash tree_1^+(x)$ is depicted below.



However, the reversed entailments do not hold and the following witnesses are provided:

- $x \approx \mathsf{nil} \wedge \mathsf{emp}$ for $tree(x) \vdash tree_1^+(x)$ and $tree(x) \vdash tree_2^+(x)$;
- $x \mapsto (l_0, r_0) * tree_1^+(l_0) * (r_0 \approx \mathsf{nil} \wedge \mathsf{emp})$ for $tree_2^+(x) \vdash tree_1^+(x)$. Note that predicate atoms can occur within counterexamples and indicate that they can be substituted by any model to obtain a more concrete one. In this case, an immediate substitution with the base case of $tree_1^+(l_0)$ gives us $x \mapsto (l_0, r_0) * l_0 \mapsto (\mathsf{nil}, \mathsf{nil}) * (r_0 \approx \mathsf{nil} \wedge \mathsf{emp})$.

4.3 Case Study: Acyclic List Segments

Consider the following definitions for cyclic and acyclic list segments. Note that $\mathcal{S}_l$ is ranked, non-filtering, non-overlapping and has the fvi property.

$$ls(x, y) \leftarrow_{\mathcal{S}_l} x \approx y \wedge \mathsf{emp} \mid x \mapsto z, ls(z, y)$$
$$ls^a(x, y) \leftarrow_{\mathcal{S}_l} x \approx y \wedge \mathsf{emp} \mid \neg(x \approx y) \wedge x \mapsto z, ls^a(z, y)$$

The equivalent SMT-LIB definitions for ls and ls^a are provided in Listing 4.8.

```
(set-logic SEPLOGLIA)

(define-fun-rec ls ((x Int) (y Int)) Bool
    (or  (and (= x y) emp)
         (exists ((z Int)) (sep (pto x z) (ls z y))) ) )

(define-fun-rec lsa ((x Int) (y Int)) Bool
    (or  (sep (= x y) emp)
         (exists ((z Int)) (sep (distinct x y) (pto x z) (lsa z y))) ) )
```

Listing 4.8: Definitions of possibly cyclic and acyclic list segments in SMT-LIB

Naturally, the entailment $ls^a \models^{\mathsf{SL}}_{\mathcal{S}_l} ls$ holds, while $ls \models^{\mathsf{SL}}_{\mathcal{S}_l} ls^a$ does not. In the latter case, the counterexample provided by `Inductor` for $ls(x, y) \vdash ls^a(x, y)$ is $x \approx y \wedge x \mapsto z_0 * ls(z_0, y)$,

from which we can obtain the more concrete one $x \approx y \wedge x \mapsto z_0 * (z_0 \approx y \wedge \mathsf{emp})$. The proof for the former case is shown below.

$$\dfrac{\dfrac{\dfrac{}{x \approx y \wedge \mathsf{emp} \vdash x \approx y \wedge \mathsf{emp},\ \exists z_1.x \mapsto z_1 * ls(z_1,y)}\ \mathrm{AX_{SL}}}{x \approx y \wedge \mathsf{emp} \vdash ls(x,y)}\ \mathrm{RU_{SL}} \qquad \dfrac{\dfrac{\dfrac{}{ls^a(z_0,y) \vdash ls(z_1,y)}\ \mathrm{ID}}{\neg(x \approx y) \wedge x \mapsto z_0, ls^a(z_0,y) \vdash x \approx y \wedge \mathsf{emp},\ \exists z_1.x \mapsto z_1 * ls(z_1,y)}\ \mathrm{RD_{SL}}}{\neg(x \approx y) \wedge x \mapsto z_0, ls^a(z_0,y) \vdash ls(x,y)}\ \mathrm{RU_{SL}}$$

$$\dfrac{x \approx y \wedge \mathsf{emp} \vdash ls(x,y) \qquad \neg(x \approx y) \wedge x \mapsto z_0, ls^a(z_0,y) \vdash ls(x,y)}{ls^a(x,y) \vdash ls(x,y)}\ \mathrm{LU}$$

4.4 Case Study: List Segments of Even and Odd Length

Consider the following definitions for list segments of even and odd length, together with the previously introduced definition for list segments and two alternate definitions of list segments with at least one element.

$$\begin{array}{rll}
ls^e(x,y) & \leftarrow_{\mathcal{S}_{eo}} x \approx y \wedge \mathsf{emp} & \mid x \mapsto z, ls^o(z,y) \\
ls^o(x,y) & \leftarrow_{\mathcal{S}_{eo}} x \mapsto y & \mid x \mapsto z, ls^e(z,y) \\
ls(x,y) & \leftarrow_{\mathcal{S}_{eo}} x \approx y \wedge \mathsf{emp} & \mid x \mapsto z, ls(z,y) \\
ls^+(x,y) & \leftarrow_{\mathcal{S}_{eo}} x \mapsto y & \mid x \mapsto z, ls^+(z,y) \\
\widehat{ls}^+(x,y) & \leftarrow_{\mathcal{S}_{eo}} x \mapsto z, ls^e(z,y) & \mid x \mapsto z, ls^o(z,y)
\end{array}$$

Listing 4.9 shows the equivalent SMT-LIB definitions for ls^e, ls^o, ls, ls^+ and $\widehat{ls}$.

```
(set-logic SEPLOGLIA)

(define-funs-rec ((lse ((x Int) (y Int)) Bool)
                  (lso ((x Int) (y Int)) Bool))
   ((or  (and (= x y) emp)
         (exists ((z Int)) (sep (pto x z) (lso z y))))

    (or  (pto x y)
         (exists ((z Int)) (sep (pto x z) (lse z y)))) ) )

(define-fun-rec ls ((x Int) (y Int)) Bool
    (or  (and (= x y) emp)
         (exists ((z Int)) (sep (pto x z) (ls z y))) ) )

(define-fun-rec lsp ((x Int) (y Int)) Bool
    (or  (pto x y)
         (exists ((z Int)) (sep (pto x z) (lsa z y))) ) )

(define-fun-rec lspeo ((x Int) (y Int)) Bool
    (or  (exists ((z Int)) (sep (pto x z) (lse z y)))
         (exists ((z Int)) (sep (pto x z) (lso z y))) ) )
```

Listing 4.9: Definitions for list segments of even and odd length in SMT-LIB

The entailments $ls^o \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} \widehat{ls}^+$, $ls^+ \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} \widehat{ls}^+$, $\widehat{ls}^+ \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} ls^+$, $ls^+ \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} ls^e$, ls^o and $\widehat{ls}^+ \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} ls^e, ls^o$ hold. A branch of the proof for $ls^+(x,y) \vdash \widehat{ls}^+(x,y)$ is shown below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\top}{ls^+(z_{00},y) \vdash ls^o(z_{00},y), ls^e(z_{00},y)}\,\mathrm{ID}}{z_0 \mapsto z_{00}, ls^+(z_{00},y) \vdash z_0 \approx y \wedge \mathsf{emp}, \exists z_{01}\,.\, z_0 \mapsto z_{01} * ls^o(z_{01},y), z_0 \mapsto y, \exists z_{11}\,.\, z_0 \mapsto z_{11} * ls^e(z_{11},y)}\,\mathrm{RD}}{z_0 \mapsto z_{00}, ls^+(z_{00},y) \vdash z_0 \approx y \wedge \mathsf{emp}, \exists z_{01}\,.\, z_0 \mapsto z_{01} * ls^o(z_{01},y), ls^o(z_0,y)}\,\mathrm{RU}}{z_0 \mapsto z_{00}, ls^+(z_{00},y) \vdash ls^e(z_0,y), ls^o(z_0,y)}\,\mathrm{RU}}{ls^+(z_0,y) \vdash ls^e(z_0,y), ls^o(z_0,y)}\,\mathrm{LU}}{x \mapsto z_0, ls^+(z_0,y) \vdash \exists z_1\,.\, x \mapsto z_1 * ls^e(z_1,y), \exists z_1\,.\, x \mapsto z_1 * ls^o(z_1,y)}\,\mathrm{RD}}{x \mapsto z_0, ls^+(z_0,y) \vdash \widehat{ls}^+(x,y)}\,\mathrm{RU}}{ls^+(x,y) \vdash \widehat{ls}^+(x,y)}\,\mathrm{LU}$$

On the other hand, entailments such as $ls^e \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} \widehat{ls}^+$, $ls^e \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} ls^o$, $ls^o \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} ls^e$, $ls^+ \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} ls^e$ or $\widehat{ls}^+ \models^{\mathsf{SL}}_{\mathcal{S}_{eo}} ls^o$ do not. `Inductor` gives the following counterexamples:

- $x \approx y \wedge \mathsf{emp}$ for $ls^e(x,y) \vdash \widehat{ls}^+(x,y)$ and $ls^e(x,y) \vdash ls^o(x,y)$;
- $x \mapsto y$ for $ls^o(x,y) \vdash ls^e(x,y)$ and $ls^+(x,y) \vdash ls^e(x,y)$;
- $x \mapsto z_0 * z_0 \mapsto y$ for $\widehat{ls}^+(x,y) \vdash ls^o(x,y)$.

4.5 Case Study: Lists of Acyclic List Segments

We adapt the acyclic list segments definitions from the previous section to a fragment in which each memory location points to a pair of locations, and use them to define lists whose elements point at cyclic or acyclic list segments. Note that $\mathcal{S}_{ll}$ is still ranked, non-filtering, non-overlapping and has the fvi property.

$$\begin{aligned}
ls(x,y) &\leftarrow_{\mathcal{S}_{ll}} x \approx y \wedge \mathsf{emp} \mid x \mapsto (z, \mathsf{nil}), ls(z,y)\\
ls^a(x,y) &\leftarrow_{\mathcal{S}_{ll}} x \approx y \wedge \mathsf{emp} \mid \neg(x \approx y) \wedge x \mapsto (z, \mathsf{nil}), ls^a(z,y)\\
lls(x,v) &\leftarrow_{\mathcal{S}_{ll}} x \approx \mathsf{nil} \wedge \mathsf{emp} \mid x \mapsto (z,u), ls(u,v), lls(z,w)\\
lls^a(x,v) &\leftarrow_{\mathcal{S}_{ll}} x \approx \mathsf{nil} \wedge \mathsf{emp} \mid x \mapsto (z,u), ls^a(u,v), lls^a(z,w)
\end{aligned}$$

The equivalent SMT-LIB definitions for ls, ls^a, lls and lls^a are provided in Listing 4.10.

```
(set-logic SEPLOGLIA)
(declare-datatypes ((Node 0)) (((node (next Int) (data Int)))))

(define-fun-rec ls ((x Int) (y Int)) Bool
   (or  (and (= x y) emp)
        (exists ((z Int)) (sep (pto x (node z (as nil Int))) (ls z y))) ) )

(define-fun-rec lsa ((x Int) (y Int)) Bool
   (or  (and (= x y) emp)
```

```
        (exists ((z Int))
            (sep (distinct x y) (pto x (node z (as nil Int))) (lsa z y))) ) )

(define-fun-rec lls ((x Int) (v Int)) Bool
    (or  (and (= x (as nil Int)) emp)
         (exists ((z Int) (u Int) (w Int))
             (sep (pto x (node z u)) (ls u v) (lls z w))) ) )

(define-fun-rec llsa ((x Int) (v Int)) Bool
    (or  (and (= x (as nil Int)) emp)
         (exists ((z Int) (u Int) (w Int))
             (sep (pto x (node z u)) (lsa u v) (llsa z w))) ) )
```

Listing 4.10: Definitions for lists of possibly cyclic and acyclic list segments in SMT-LIB

The entailment $lls^a \models^{\mathsf{SL}}_{\mathcal{S}_{ll}} lls$ holds, while its reverse $lls \models^{\mathsf{SL}}_{\mathcal{S}_{ll}} lls^a$ does not. In the latter case, the counterexample provided by `Inductor` is $x \mapsto (z_0, u_0) * (u_0 \approx v \wedge u_0 \mapsto u_{00} * ls(u_{00}, v)) * lls(z_0, w_0)$, from which we can obtain the more concrete one $x \mapsto (z_0, u_0) * (u_0 \approx v \wedge u_0 \mapsto u_{00} * (u_{00} \approx y \wedge \mathsf{emp})) * (z \approx \mathsf{nil} \wedge \mathsf{emp})$. The proof for the former case is partially shown below. The subproof for $ls^a(u_0, v) \vdash ls(u_0, v)$ is skipped as it is identical to the one from Section 4.3 modulo a variable renaming.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\top}{ls^a(u_{00}, v) \vdash ls(u_{00}, v)}\,\mathrm{ID}}{\vdots}}{ls^a(u_0, v) \vdash ls(u_0, v)}\,\mathrm{LU} \qquad \dfrac{\top}{lls^a(z_0, w_0) \vdash lls(u_0, w_0)}\,\mathrm{ID}}{ls^a(u_0, v), lls^a(z_0, w_0) \vdash ls(u_0, v) * lls(z_0, w_0)}\,\mathrm{SP}_{\mathsf{SL}}}{x \mapsto (z_0, u_0), ls^a(u_0, v), lls^a(z_0, w_0) \vdash x \approx \mathsf{nil} \wedge \mathsf{emp}, \exists z_1 \exists u_1 \exists w_1 \,.\, x \mapsto (z_1, u_1) * ls(u_1, v) * lls(z_1, w_1)}\,\mathrm{RD}_{\mathsf{SL}}}{\dfrac{x \mapsto (z_0, u_0), ls^a(u_0, v), lls^a(z_0, w_0) \vdash lls(x, v)}{lls^a(x, v) \vdash lls(x, v)}\,\mathrm{LU}}\,\mathrm{RU}_{\mathsf{SL}}$$

4.6 Experimental results

Table 4.1 summarizes the experimental results obtained for the entailments discussed in Sections 4.2-4.5. All experiments were run on a 2.10GHz Intel® Core™ i7-4600U CPU machine with 4MB of cache. For each case, we indicate:

- The result (column **Result**), which can be `VALID` or `INVALID`;
- The total number of sequent nodes created (column **Sequents**);
- The total number of inference rule nodes created (column **Rules**);
- The maximum number of sequent nodes along a branch (column $\mathbf{H}_{\mathrm{Seq}}$);
- The maximum number of LU applications along a branch (column $\mathbf{H}_{\mathrm{LU}}$);
- The maximum number of SP applications along a branch (column $\mathbf{H}_{\mathrm{SP}}$);

– The run time for the proof search algorithm (column **Time**);

– The total number of calls made to CVC4 (column **CVC4**).

LHS	**RHS**	**Result**	**Sequents**	**Rules**	$\mathbf{H}_{\mathrm{Seq}}$	$\mathbf{H}_{\mathrm{LU}}$	$\mathbf{H}_{\mathrm{SP}}$	**Time**	**CVC4**
$tree_1^+$	$tree$	VALID	34	22	7	2	1	0.096s	9
$tree_2^+$	$tree$	VALID	21	15	7	2	1	0.053s	7
$tree_2^+$	$tree_1^+$	VALID	1477	889	11	3	2	5.515s	37
$tree$	$tree_1^+$	INVALID	7	5	4	1	0	0.033s	7
$tree$	$tree_2^+$	INVALID	7	5	4	1	0	0.028s	5
$tree_1^+$	$tree_2^+$	INVALID	38	25	8	2	1	0.096s	14
ls^a	ls	VALID	8	6	5	1	0	0.014s	2
ls	ls^a	INVALID	7	5	4	1	0	0.015s	2
lls^a	lls	VALID	21	15	9	2	1	0.048s	4
lls	lls^a	INVALID	20	14	8	2	1	0.043s	4
ls^o	$\widehat{ls}^+$	VALID	10	8	6	1	0	0.032s	5
ls^+	$\widehat{ls}^+$	VALID	16	13	8	2	0	0.049s	9
ls^+	ls^e, ls^o	VALID	8	6	5	1	0	0.020s	4
$\widehat{ls}^+$	ls^e, ls^o	VALID	9	7	5	1	0	0.028s	8
ls^e	$\widehat{ls}^+$	INVALID	7	5	4	1	0	0.024s	4
ls^e	ls^o	INVALID	7	5	4	1	0	0.030s	5
ls^+	ls^e	INVALID	13	10	6	2	0	0.075s	8
$\widehat{ls}^+$	ls^o	INVALID	20	16	9	3	0	0.143s	12

Table 4.1: Experimental results for `Inductor`

As shown by the **Time** column, the execution times are fairly low. The size of the derivations are influenced by how elaborate the inductive definitions are. For instance, $tree_1^+$ is defined by three predicate rules, thus when encountered on the left-hand side of an entailment will generate a larger number of nodes due to left-unfolding. On the right-hand side, the number of predicate rules in a definition and the number of subgoals in each predicate rule both influence the complexity of SP, which can lead to higher execution times than expected, given the size of the derivation, since all instances of SP need to be generated and then checked. In the $tree_2^+ \overset{\mathsf{SL}}{\models}_{\mathcal{S}_t} tree_1^+$ case, the large number of sequents generated is caused by both the complexity of the inductive definitions and the number of SP applications along a branch. Moreover, the fact that we only allow ancestral pivots for ID creates a large number of redundancies, requiring sequents that only differ by a substitution of variables to be proved multiple times, in different parts of the derivation. This case alone is enough motivation to consider non-ancestral pivot nodes as future work.

Conclusions

We propose cyclic proof systems for entailments between inductively defined predicates in (multisorted) first-order logic and separation logic, based on Fermat's principle of infinite descent, which allows inductive invariants to be produced during proof search, as opposed to classical induction, where they have to be provided. We have established the boundaries within which these proof systems remain sound and complete using semantic constraints. Soundness is ensured by requiring the models generated by unfoldings to decrease in a well-founded domain. Completeness is determined by three additional conditions on the set of constraints used in the inductive definitions.

These restrictions are decidable, with computational complexities that depend on the logical fragment in which the constraints of the inductive system are written, and they can all be essentially reduced to determining the satisfiability of $\exists^*\forall^*$-quantified formulae. While there already were well-established results for first-order logic, this was not also the case for separation logic. Hence, we provide two decision procedures for the satisfiability of formulae written in separation logic and belonging either to the quantifier-free or the $\exists^*\forall^*$-quantified fragments. We also analyse decidability and complexity bounds for these fragments.

Ultimately, we describe an entailment checker tool called `Inductor`, which implements our proof system for separation logic and utilizes the aforementioned decision procedures. The tool outputs a proof whenever an entailment is found to be valid, or counterexamples when it is not. It is still possible to use `Inductor` outside of the boundaries that warrant soundness and completeness, and warnings are emitted whenever one or more semantic restrictions are violated. In this case, however, the proof search results may be inconclusive.

Future work

As mentioned in Section 2.5.2, our proof system for entailments involving inductive predicates written in separation logic is only complete when the predicates unfold the heap in a similar manner. In practice, however, it is often useful to prove entailments such as $dlls \models^{\mathsf{SL}}_{\mathcal{S}'} dlls^r$ and $dlls^r \models^{\mathsf{SL}}_{\mathcal{S}'} dlls$ from Example 2.5.1. Unfolding these predicates generates different coverage trees for the heap, but they still are related by a tree isomorphism resulting from a rotation.

This type of behaviour can be captured by a CUT inference rule, as used by other cyclic proof systems [8, 9]. In our case, if $\Gamma \vdash * \Upsilon$ is a predecessor of $\Gamma', \Gamma\theta \vdash \Delta$, CUT would allow the replacement of $\Gamma\theta$ with $\Upsilon\theta$ on the left-hand side of the latter sequent. At a semantic level, this replacement should be viewed as extracting the coverage tree corresponding to $\Gamma\theta$ from the larger one corresponding to $\{\Gamma', \Gamma\theta\}$, thus creating a hole, and then rotating the latter such that the parent of $\Gamma\theta$ becomes the new root, while the hole propagates to the node that was the old root. Then, the coverage tree corresponding to $\Upsilon\theta$ is attached at the

newly obtained hole. A proof system containing such a CUT inference rule would be sound, but would cease to be complete. However, coupled with a comprehensive proof strategy, it could still prove its usefulness in practice.

On a different note, from a practical standpoint, we are often interested to separately reason about the shape of a data structure (e.g. a singly-linked list, a doubly-linked list, a tree) and the information stored by each node (e.g. a sorted list, a binary search tree). Capturing both types of information in a single constraint for an inductive predicate rule can often lead to violating the non-overlapping restriction.

Consider the following singly-linked list definitions with integer data constraints:

$$
\begin{array}{l}
p_0(x_0, d_0) \leftarrow_{\mathcal{S}} x_0 \mapsto (x_1, d_1) \wedge 0 \leq d_1 \leq 1, p_1(x_1, d_1) \\
p_1(x_0, d_0) \leftarrow_{\mathcal{S}} x_0 \mapsto (x_1, d_1) \wedge d_1 = 1 - d_0, p_1(x_1, d_1) \mid x_0 \approx \mathsf{nil} \wedge \mathsf{emp}
\end{array}
$$

$$
\begin{array}{rl}
q_0(x_0, d_0) \leftarrow_{\mathcal{S}} & x_0 \mapsto (x_1, d_1) \wedge d_1 \leq 0, q_1(x_1, d_1) \\
\mid & x_0 \mapsto (x_1, d_1) \wedge d_1 \geq 1, q_2(x_1, d_1) \\
q_1(x_0, d_0) \leftarrow_{\mathcal{S}} & x_0 \mapsto (x_1, d_1) \wedge d_1 = d_0 + 1, q_0(x_1, d_1) \mid x_0 \approx \mathsf{nil} \wedge \mathsf{emp} \\
q_2(x_0, d_0) \leftarrow_{\mathcal{S}} & x_0 \mapsto (x_1, d_1) \wedge d_1 = d_0 - 1, q_0(x_1, d_1) \mid x_0 \approx \mathsf{nil} \wedge \mathsf{emp}
\end{array}
$$

From a structural standpoint, both p_0 and q_0 describe the same kind of singly-linked lists and, moreover, the coverage trees obtained by unfolding them can be matched directly. However, the data constraints make $\mathcal{S}$ overlapping. For instance, $\phi \equiv x_0 \mapsto (x_1, d_1) \wedge 0 \leq d_1 \leq 1$ and $\psi \equiv x_0 \mapsto (x_1, d_1) \wedge d_1 \leq 0$ have common models where $d_1 = 0$, but neither $\phi \models^{\mathsf{SL}} \psi$ nor $\psi \models^{\mathsf{SL}} \phi$ hold. The entailment $p_0 \models^{\mathsf{SL}}_{\mathcal{S}} q_0$ holds, but has no proof using $\mathcal{R}^{\mathsf{SL}}_{\mathsf{Ind}}$.

Alternating data automata, however, are better equipped for modelling and reasoning about relations between past and current data variables. Their main advantage is the fact that complementation and intersection are possible in real time, which enables a concise encoding of trace inclusions – an inclusion problem $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ is reduced to the non-emptiness of the intersection $\mathcal{L}(A) \cap \mathcal{L}(\overline{B})$, where $\overline{B}$ is the complement of B. Although the emptiness problem is undecidable in general, efficient model-checking semi-algorithms, based on abstraction refinement methods, have been provided [28]. For future work, we envision a merging of the two approaches by extracting traces from failed branches of derivations built with our inference rules, using them to construct an interpolation problem, and then using the interpolant obtained as a solution to refine the derivation such that sequents containing overlapping constraints are split into several non-overlapping ones, which together correctly capture the original entailment problem.

Bibliography

[1] Timos Antonopoulos, Nikos Gorogiannis, Christoph Haase, Max I. Kanovich, and Joël Ouaknine. Foundations for decision problems in separation logic with general inductive predicates. In *Foundations of Software Science and Computation Structures: 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, pages 411–425. Springer Berlin Heidelberg, 2014.

[2] Kshitij Bansal. *Decision Procedures for Finite Sets with Cardinality and Local Theory Extensions*. Ph.D. thesis, New York University, 2016.

[3] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, pages 171–177. Springer Berlin Heidelberg, 2011.

[4] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa, 2017. Available at `www.SMT-LIB.org`.

[5] Nikolaj Bjørner and Mikoláš Janota. Playing with quantified satisfaction. In *LPAR-20: 20th International Conferences on Logic for Programming, Artificial Intelligence and Reasoning - Short Presentations*, volume 35 of *EPiC Series in Computing*, pages 15–27. EasyChair, 2015.

[6] Rémi Brochenin, Stéphane Demri, and Etienne Lozes. On the almighty wand. *Information and Computation*, 211:106–137, 2012.

[7] James Brotherston, Carsten Fuhs, Juan A. Navarro Pérez, and Nikos Gorogiannis. A decision procedure for satisfiability in separation logic with inductive predicates. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, pages 25:1–25:10, New York, NY, USA, 2014. ACM.

[8] James Brotherston, Nikos Gorogiannis, and Rasmus L. Petersen. A generic cyclic theorem prover. In *Programming Languages and Systems: 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11-13, 2012. Proceedings*, pages 350–367. Springer Berlin Heidelberg, 2012.

[9] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

[10] W. H. Bussey. Fermat's method of infinite descent. *The American Mathematical Monthly*, 25(8):333–337, 1918.

[11] Cristiano Calcagno, Hongseok Yang, and Peter W. O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science: 21st Conference Bangalore, India, December 13–15, 2001, Proceedings*, pages 108–119, London, UK, 2001. Springer Berlin Heidelberg.

[12] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications, 2005.
URL: `http://www.grappa.univ-lille3.fr/tata`.

[13] Hubert Comon and Pierre Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, January 1989.

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[15] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, August 1979.

[16] Mnacho Echenim, Radu Iosif, and Nicolas Peltier. On the expressive completeness of Bernays-Schönfinkel-Ramsey separation logic. *CoRR*, February 2018.

[17] Robert W. Floyd. Assigning meanings to programs. In *Proceedings of the American Mathematical Society Symposia on Applied Mathematics*, volume 19 of *Proceedings of Symposia in Applied Mathematics*, pages 14–26. American Mathematical Society, 1967.

[18] Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. DPLL(T): Fast decision procedures. In *Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings*, pages 175–188. 2004.

[19] Yeting Ge and Leonardo de Moura. Complete instantiation for quantified formulas in satisfiability modulo theories. In *Computer Aided Verification: 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, pages 306–320. Springer Berlin Heidelberg, 2009.

[20] Joseph Y. Halpern. Presburger arithmetic with unary predicates is Π_1^1 complete. *The Journal of Symbolic Logic*, 56(2):637–642, 1991.

[21] Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2(1):326–336, 1952.

[22] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, October 1969.

[23] C. A. R. Hoare. Recursive data structures. *International Journal of Computer & Information Sciences*, 4(2):105–132, June 1975.

[24] Lukás Holík, Ondrej Lengál, Jirí Simácek, and Tomás Vojnar. Efficient inclusion checking on explicit and semi-symbolic tree automata. In *Automated Technology for Verification and Analysis: 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011, Proceedings*, pages 243–258. Springer Berlin Heidelberg, 2011.

[25] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2000.

[26] Gerard Huet and Derek C. Oppen. Equations and rewrite rules: A survey. Technical report, Stanford, CA, USA, January 1980.

[27] Radu Iosif, Adam Rogalewicz, and Tomás Vojnar. Deciding entailments in inductive separation logic with tree automata. In *Automated Technology for Verification and Analysis: 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings*, pages 201–218, Cham, Switzerland, 2014. Springer International Publishing.

[28] Radu Iosif and Xiao Xu. The impact of alternation. *CoRR*, abs/1705.05606, 2017.

[29] Samin S. Ishtiaq and Peter W. O'Hearn. BI as an assertion language for mutable data structures. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '01, pages 14–26, New York, NY, USA, 2001. ACM.

[30] Daniel Kroening and Ofer Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[31] Joseph B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13:297–305, 1972.

[32] Ondrej Lengál, Jirí Simácek, Tomás Vojnar, Martin Hruska, and Lukás Holík. libVATA - a C++ library for efficient manipulation with non-deterministic finite (tree) automata. URL: `https://github.com/ondrik/libvata`.

[33] Harry R. Lewis. Complexity results for classes of quantificational formulas. *Journal of Computer and System Sciences*, 21(3):317–353, December 1980.

[34] Parthasarathy Madhusudan, Xiaokang Qiu, and Andrei Stefanescu. Recursive proofs for inductive tree data-structures. In *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '12, pages 123–136, New York, NY, USA, 2012. ACM.

[35] Yuri V. Matiyasevich. Enumerable sets are diophantine. *Journal of Sovietic Mathematics*, 11(2):354–358, 1970.

[36] Flemming Nielson, Hanne R. Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1999.

[37] Peter W. O'Hearn, John C. Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. In *Proceedings of the 15th International Workshop on Computer Science Logic*, CSL '01, pages 1–19, London, UK, 2001. Springer-Verlag.

[38] Derek C. Oppen. Reasoning about recursively defined data structures. *Journal of the ACM*, 27(3):403–411, July 1980.

[39] Reinhard Pichler. On the complexity of equational problems in CNF. *Journal of Symbolic Computation*, 36:235 – 269, 2003. First Order Theorem Proving.

[40] Andrew Reynolds, Morgan Deters, Viktor Kuncak, Cesare Tinelli, and Clark Barrett. Counterexample-guided quantifier instantiation for synthesis in SMT. In *Computer Aided Verification: 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*, pages 198–216. Springer International Publishing, 2015.

[41] Andrew Reynolds, Radu Iosif, and Cristina Serban. Reasoning in the Bernays-Schönfinkel-Ramsey fragment of separation logic. In *Verification, Model Checking, and Abstract Interpretation: 18th International Conference, VMCAI 2017, Paris, France, January 15–17, 2017, Proceedings*, pages 462–482, Cham, Switzerland, 2017. Springer International Publishing.

[42] Andrew Reynolds, Radu Iosif, Cristina Serban, and Tim King. A decision procedure for separation logic in SMT. In *Automated Technology for Verification and Analysis: 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*, pages 244–261. Springer International Publishing, 2016.

[43] Andrew Reynolds, Tim King, and Viktor Kuncak. An instantiation-based approach for solving quantified linear arithmetic. *CoRR*, abs/1510.02642, 2015.

[44] Andrew Reynolds, Cesare Tinelli, Amit Goel, and Sava Krstić. Finite model finding in SMT. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044, pages 640–655. Springer Berlin Heidelberg, 2013.

[45] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, LICS '02, pages 55–74, Washington, DC, USA, 2002. IEEE Computer Society.

[46] Mihaela Sighireanu and David Cok. Report on SL-COMP 2014. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:173–186, 2014.

[47] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[48] Dirk van Dalen. *Logic and structure*. Springer-Verlag Berlin Heidelberg, 4th edition, 2004.

[49] K. N. Venkataraman. Decidability of the purely existential fragment of the theory of term algebras. *Journal of the ACM*, 34(2):492–510, April 1987.

[50] Hongseok Yang. *Local Reasoning for Stateful Programs*. Ph.D. thesis, University of Illinois at Urbana-Champaign, 2001.