

Sofia University "St. Kliment Ohridski"  
Faculty of Mathematics and Informatics  
Department of Mathematical Logic and Its Applications




Master Thesis

# Hyper Separation Logic

**Trayan Tanchev Gospodinov**

M.Sc. Logic and Algorithms, Mathematics

*Faculty Number: 5MI3100006*

Supervisor:  Prof. Dr. Tinko Tinchev

Advisors: **ETH** zürich Thibault Dardinier

**ETH** zürich Prof. Dr. Peter Müller

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Preliminaries</b>	<b>5</b>
2.1	Hoare logic . . . . .	5
2.1.1	Semantics . . . . .	5
2.1.2	Syntax . . . . .	8
2.1.3	Undecidability and effective incompleteness . . . . .	13
2.1.4	Soundness and non-effective completeness . . . . .	14
2.2	Separation logic . . . . .	26
2.2.1	Semantics . . . . .	26
2.2.2	Syntax . . . . .	29
2.2.3	Syntactic and semantic logic . . . . .	30
2.2.4	Separating conjunction and the frame rule . . . . .	32
2.3	Separation Sufficient Incorrectness logic . . . . .	35
2.4	Outcome Separation logic . . . . .	38
2.5	Relational Separation logic . . . . .	40
2.5.1	Semantics . . . . .	40
2.5.2	Syntax . . . . .	42
2.6	Properties and hyperproperties . . . . .	43
2.7	Hyper Hoare logic . . . . .	46
2.7.1	Semantics . . . . .	46
2.7.2	Syntax . . . . .	48
2.7.3	Expressivity . . . . .	49
<b>3</b>	<b>Hyper Separation logic</b>	<b>52</b>
3.1	Separating conjunction . . . . .	52
3.1.1	Desired properties . . . . .	52
3.1.2	Definition . . . . .	59
3.2	Hyper-tripe validity . . . . .	66
3.3	The strongest postcondition . . . . .	68
3.4	The Frame rule . . . . .	71
3.4.1	Soundness . . . . .	71
3.4.2	Expressivity . . . . .	72

# 1 Introduction

To formally prove a property (in the general sense) about a program, one must rely on three essential components, introduced below from the most general to the most specific. First, one needs a general foundational system in which to work (e.g., set theory). Next, a state model for representing program states must be established. Finally, the semantics of the programming language for which the property is being proven must be defined. With these in place, anyone with sufficient expertise can derive results within the limits of the chosen foundational system, such as set theory.

Hoare logics provide a structured approach to reasoning when (dis-)proving certain properties (in the general sense) of programs. While they may not introduce fundamentally new concepts—since any concept developed through Hoare logics could have emerged through general means—their true value lies in making reasoning more systematic and rigorous. This added rigor facilitates automation, allowing computers to handle the reasoning process more efficiently, reducing the reliance on manual proofs.

To start, we look at the formulae in Hoare logic [Hoare 1969],  $\{\rho\}\mathcal{C}\{q\}$ , which consist of three syntactic elements— $\rho$ ,  $\mathcal{C}$ , and  $q$ —along with the delimiting symbols  $\{$  and  $\}$ . The  $\rho$  (precondition) and  $q$  (postcondition) are first-order formulae, referred to as syntactic assertions, while  $\mathcal{C}$  is a syntactic program command. A Hoare formula  $\{\rho\}\mathcal{C}\{q\}$  is a theorem of Hoare logic iff executing  $\mathcal{C}$  in a program state that satisfies  $\rho$  leads to a program state that satisfies  $q$ , or causes divergence. It is evident that this definition relies on both the satisfiability (i.e., the interpretation) and the semantics of the program command (i.e., the interpretive model), which itself depends on the satisfiability. In this thesis,  $\rho$  and  $q$  are formulae from (a slight alteration of) Peano arithmetic, while  $\mathcal{C}$  adheres to the structure of Kleene algebra with tests. We will go over well-known results, such as the soundness of Hoare logic for any interpretation, and demonstrate that completeness is unattainable, as it would otherwise contradict Gödel’s incompleteness theorem. We will establish (following [Cook 1978]), given an oracle for the standard model of Peano arithmetic, a weaker form of completeness, namely relative completeness, which concerns only a single interpretation—in this case, the standard interpretation of Peano arithmetic.

Next, we introduce the concept of Separation logics, which are specialized Hoare logics with the added capability of reasoning about the heap. That is, the assertion language is extended, and heap-sensitive base program commands are added to the existing syntax for program commands. We demonstrate that the naive approach for local reasoning through the so called constancy rule fails, because two different variables can point to the same heap location. We then present the seminal solution from [Reynolds 2002], which re-establishes local reasoning through the introduction of the separating conjunction  $*$ , replacing  $\wedge$  in the constancy rule and thereby giving rise to the so-called frame rule.

Then, we introduce the concept of semantic logics, where the formulae are not syntactic, but rather semantic, i.e. set-theoretic, objects, and the theorems

are recursively (in the sense of set theory) defined predicates and not finitistic ones. Once the semantic logic is established, obtaining a corresponding syntactic logic becomes straightforward, with the primary challenge arising only if one aims to show relative completeness, particularly in ensuring that the syntax can express all loop invariants. We argue that, when discussing Hoare logics, it is more insightful to focus on semantic logics rather than syntactic ones. This is because the main advantage of a syntactic logics is that they are finitistic, making them well-suited for automation. However, any relatively complete syntactic Hoare logic that incorporates the standard model of Peano arithmetic relies on an oracle, thus diverging it from finitism and consequently losing its main advantage.

Subsequently, we introduce Sufficient Incorrectness logic and its Separation variant [Ascari et al. 2024]. Similarly to Hoare (and Separation) logic, their formulae are of the form  $\{\mathcal{P}\}\mathcal{E}\{\mathcal{Q}\}$ . We say that  $\{\mathcal{P}\}\mathcal{E}\{\mathcal{Q}\}$  is a theorem of (Separation) Sufficient Incorrectness logic iff for any starting state that satisfies  $\mathcal{P}$ , there exists a terminating execution of  $\mathcal{E}$  that leads to a state satisfying  $\mathcal{Q}$ . Then, we introduce the concepts of over- and underapproximate logics. In short, overapproximating logics prove properties for all possible executions, ensuring the absence of bad behaviors, while underapproximating logics establish the existence of certain executions, which can be useful for demonstrating bugs. Overapproximate logics include Hoare and Separation logic, whereas underapproximate logics include Sufficient Incorrectness logic and its separation variant.

Following this, we introduce Outcome [Zilberstein et al. 2023] and Outcome Separation logic [Zilberstein et al. 2024], which, unlike the logics mentioned above, have formulae of the form  $\{\mathcal{P}\}\mathcal{E}\{\mathcal{Q}\}$ , where  $\mathcal{P}$  and  $\mathcal{Q}$  are formulae that express not a set of states, but rather a set of sets of states. Disregarding a minor detail,  $\{\mathcal{P}\}\mathcal{E}\{\mathcal{Q}\}$  is said to be a theorem of Outcome and Outcome Separation logic iff executing (in the lifted sense)  $\mathcal{E}$  in a set of program states that satisfies  $\mathcal{P}$  leads to a set of program states that satisfies  $\mathcal{Q}$ . Outcome and its separation variant support both over- and underapproximate reasoning.

After, we introduce Relational Hoare logic [Benton 2004] and Relational Separation logic [Yang 2007], which reason about relations between programs. That is, their formulae are of the form  $\{\mathcal{R}\}_{\mathcal{E}_2}^{\mathcal{E}_1}\{s\}$ , where  $\mathcal{R}$  and  $s$  are formulae that express relations of states and  $\mathcal{E}_1$  and  $\mathcal{E}_2$  are program commands. Disregarding a minor detail, theorems of Relational Hoare logic and its separation variant are defined analogously to Hoare logic, but with pairs of states instead, where  $\mathcal{E}_1$  modifies the first coordinate state and  $\mathcal{E}_2$  modifies the second. A typical use case is demonstrating that a compiler-optimized program behaves the same as the original one. However, in this thesis, we focus on the scenario where the two programs coincide, which may initially appear to lack practical applications. That said, this is not the case, as we introduce some use cases, including observational determinism and non-interference.

The final preliminary logic that we introduce is Hyper Hoare logic [Dardinier and Müller 2024], which operates similarly to Outcome logic. That is, the formulae of the assertion language express sets of states and the theorems are defined analogously. Like Outcome logic, formulae of Hyper Hoare logic take

the form  $\{\mathcal{P}\}\mathcal{C}\{\mathcal{Q}\}$  with the key distinction that the syntax of  $\mathcal{P}$  and  $\mathcal{Q}$  is more expressive, allowing expressing statements like "all states have the same value for a given program variable". This addition is essential, as it provides the missing component for achieving a Hoare-style logic with maximal expressiveness. To formalize this, we introduce program properties and program hyperproperties, and refer to [Dardinier and Müller 2024], where it is demonstrated that Hyper Hoare logic can express any program hyperproperty.

In practice, virtually all programs in practice use heap. This renders Hyper Hoare logic, while highly expressive, practically inapplicable. This issue is not specific to Hyper Hoare logic; it arises with all Hoare logics. The standard solution, applied to essentially all well-established Hoare logics, is to develop a separation-style variant. The difficulty in Hyper Hoare logic arises from the fact that the separating conjunction, which operates over sets of states, cannot be used nearly as directly as in other Hoare logics, as we are concerned with sets of sets of states. The goal of this thesis is to explore the properties that our separating conjunction should satisfy, and then define and verify that it meets these properties. Finally, we present an initial outline of what validity in Hyper Separation Logic might look like and argue that the definition is well-grounded as it gives rise to a very expressive frame rule.

## 2 Preliminaries

In this section, we lay the groundwork required for Hyper Separation logic. We begin by introducing the foundational concept of Hoare logic [Hoare 1969], followed by Separation logic - a specialized framework for local reasoning about heap programs. Next, we delve into Separation Sufficient Incorrectness logic, offering deeper insights into the intricacies of program verification, in particular - underapproximate reasoning. Subsequently, we introduce Outcome Separation logic, which can express both over- and underapproximate reasoning. Then, we discuss Relational Separation logic, which provides a framework for reasoning about two programs. Finally, we introduce a novel logic, which can reason about arbitrary hyperproperties, called Hyper Hoare logic.

### 2.1 Hoare logic

#### 2.1.1 Semantics

The foundational concept we will rest upon is Hoare triple. A Hoare triple has the form  $\{p\}C\{q\}$ , where  $p$  (precondition) and  $q$  (postcondition) are assertions and  $C$  is a program command. A Hoare triple is valid iff executing successfully  $C$ , starting in a program state satisfying  $p$ , results in a program state satisfying  $q$ .

We begin with some definitions that depend on a non-empty set,  $\text{Val}$ , and a countably infinite set of program variables,  $\text{PVars}$ . While this dependency will be omitted in some instances and explicitly mentioned in others, we trust that the reader will be able to discern the intended context.

**Definition 2.1.** A heapless program state (or stack)  $s$  is a total function from  $\text{PVars}$  to  $\text{Val}$ , i.e.  $s : \text{PVars} \rightarrow \text{Val}$ .

We denote the set of all heapless program states by  $\text{Stacks}$ . We will use the meta symbol  $s$  with potential indices to denote a heapless program state.

**Definition 2.2.** A heapless assertion  $p$  is a set of heapless program states.

We denote the set of all heapless assertions by  $\text{SAsrts}$  (S for store/stack). We will use the meta symbols  $p, q, r, f$  with potential indices to denote assertions<sup>1</sup> and heapless assertions. Which of the two is intended will be clear from the context.

**Definition 2.3.** A program expression  $e$  is a total function from  $\text{Stacks}$  to  $\text{Val}$ .

We will use the meta symbol  $e$  with potential indices to denote a program expression.

**Definition 2.4.** A program predicate  $b$  is a set of heapless program states<sup>2</sup>.

We will use the meta symbol  $b$  with potential indices to denote a program predicate. With  $\bar{b}$  we will denote the complement of  $b$ .

**Definition 2.5.** We define heapless program commands using Backus–Naur form (BNF):

$$C \Leftarrow \mathbf{skip} \mid x := e \mid x := \mathbf{nonDet}() \mid \mathbf{assume} \ b \mid (C; C) \mid (C + C) \mid C^*,$$

where  $x \in \text{PVars}$ .

We will use the meta symbol  $C$  with potential indices to denote a program command<sup>3</sup> or a heapless program command. Which of the two is intended will be clear from the context. For the remainder of this thesis, standard parentheses omitting rules hold, e.g. when precedence is clear, when associativity holds and the outer most parentheses.

The **skip**, assignment ( $x := e$ ) and sequential composition ( $C_1; C_2$ ) are standard. The **assume** command aborts the execution if  $b$  is not satisfied, otherwise it acts like **skip**. The  $+$  command is nondeterministic choice and  $*$  is nondeterministic iteration. Using these commands we can define deterministic **if  $b$  then  $C_1$  else  $C_2$  fi** and deterministic **while  $b$  do  $C$  od** as follows:

$$\begin{aligned} \mathbf{if} \ b \ \mathbf{then} \ C_1 \ \mathbf{else} \ C_2 \ \mathbf{fi} &\Leftarrow (\mathbf{assume} \ b; C_1) + (\mathbf{assume} \ \bar{b}; C_2) \\ \mathbf{while} \ b \ \mathbf{do} \ C \ \mathbf{od} &\Leftarrow (\mathbf{assume} \ b; C)^*; \mathbf{assume} \ \bar{b} \end{aligned}$$

Our program commands also include unbounded nondeterministic assignment ( $x := \mathbf{nonDet}()$ ). In combination with **assume** we can express constrained nondeterministic assignment:

$$x := \mathbf{cNonDet}(a, b) \Leftarrow x := \mathbf{nonDet}(); \mathbf{assume} \ \{s \in \text{Stacks} \mid a \leq s(y) \leq b\}$$

<sup>1</sup>Yet to be introduced.

<sup>2</sup>Note that the definition coincides with that of a heapless assertion.

<sup>3</sup>Yet to be introduced.

$$\begin{array}{c}
\frac{}{\langle x := e, s \rangle \rightarrow \langle \mathbf{skip}, s_x^{e(s)} \rangle} \qquad \frac{}{\langle x := \text{nonDet}(), s \rangle \rightarrow \langle \mathbf{skip}, s_x^n \rangle} \\
\frac{}{\langle \mathbf{skip}; C_2, s \rangle \rightarrow \langle C_2, s \rangle} \quad \frac{\langle C_1, s \rangle \rightarrow \langle C'_1, s' \rangle}{\langle C_1; C_2, s \rangle \rightarrow \langle C'_1; C_2, s' \rangle} \quad \frac{s \in b}{\langle \mathbf{assume} \ b, s \rangle \rightarrow \langle \mathbf{skip}, s \rangle} \\
\frac{}{\langle C_1 + C_2, s \rangle \rightarrow \langle C_2, s \rangle} \quad \frac{}{\langle C_1 + C_2, s \rangle \rightarrow \langle C_1, s \rangle} \quad \frac{}{\langle C^*, s \rangle \rightarrow \langle (C; C^*) + \mathbf{skip}, s \rangle}
\end{array}$$

Figure 2.1: Small-step semantics of heapless program commands, where  $s_x^n(x) = v$  and  $s_x^n(y) = s(y)$  for  $y \neq x$ .

To define formally the semantics of our heapless program commands, we introduce:

**Definition 2.6.** A *heapless program configuration*  $\langle C, s \rangle$  is an ordered pair, where  $C$  is a heapless program command and  $s$  is a heapless program state.

**Definition 2.7.** The *small-step semantics*, denoted  $\rightarrow$ , of the heapless program commands is a relation over heapless program configurations and is defined by recursion in figure 2.1.

We will denote by  $\rightarrow^*$  the reflexive and transitive closure of the relation  $\rightarrow$ . For any  $\langle C, s \rangle, \langle C', s' \rangle$ :  $\langle C, s \rangle \rightarrow^* \langle C', s' \rangle$  we say that  $C$  executes, starting in  $s$ , with a remainder  $C'$  and results in  $s'$ . If  $C' = \mathbf{skip}$ , then we say that  $C$  executes successfully (terminates successfully), starting in  $s$  and results in  $s'$ . Note that we can have  $\langle C, s \rangle \rightarrow^* \langle C', s' \rangle$  and  $\langle C, s \rangle \rightarrow^* \langle C', s'' \rangle$ , where  $s' \neq s''$ , e.g.  $\langle x := \text{nonDet}(), s \rangle \rightarrow^* \langle \mathbf{skip}, s_x^0 \rangle$  and  $\langle x := \text{nonDet}(), s \rangle \rightarrow^* \langle \mathbf{skip}, s_x^1 \rangle$ . We say that  $C$  does not terminate, starting in  $s$  iff  $\nexists s'. \langle C, s \rangle \rightarrow^* \langle \mathbf{skip}, s' \rangle$ . We denote the input-output relation of  $C$  with  $\llbracket C \rrbracket \Leftarrow \{ \langle s, s' \rangle \mid \langle C, s \rangle \rightarrow^* \langle \mathbf{skip}, s' \rangle \}$ .

**Lemma 2.8.** The following properties hold:

- (i)  $\llbracket \mathbf{skip} \rrbracket = \{ \langle s, s \rangle \mid s \in \text{Stacks} \}$ ;
- (ii)  $\llbracket x := e \rrbracket = \{ \langle s, s_x^{e(s)} \rangle \mid s \in \text{Stacks} \}$ ;
- (iii)  $\llbracket x := \text{nonDet}() \rrbracket = \{ \langle s, s_x^n \rangle \mid s \in \text{Stacks} \wedge n \in \text{Val} \}$ ;
- (iv)  $\llbracket \mathbf{assume} \ b \rrbracket = \{ \langle s, s \rangle \mid s \in b \}$ ;
- (v)  $\llbracket C_1; C_2 \rrbracket = \llbracket C_1 \rrbracket \circ \llbracket C_2 \rrbracket$ , where  $\circ$  is composition of relations;
- (vi)  $\llbracket C_1 + C_2 \rrbracket = \llbracket C_1 \rrbracket \cup \llbracket C_2 \rrbracket$ ;
- (vii)  $\llbracket C^* \rrbracket = \bigcup_{n \in \mathbb{N}} \llbracket C^n \rrbracket$ , where  $C^0 \Leftarrow \mathbf{skip}$  and  $C^{n+1} \Leftarrow C; C^n$ .

*Proof.* See theorem `dsem_properties` in `HeaplessProgramCommands.thy`.  $\square$

Now, we are ready to define formally heapless Hoare triples and their validity.

**Definition 2.9.** A *heapless Hoare triple*  $\{p\}C\{q\}$  is an ordered triple (with a special syntax  $\{\cdot\} \cdot \{\cdot\}$ ), where  $p$  and  $q$  are heapless assertions and  $C$  is a heapless program command.

**Definition 2.10.** A *heapless Hoare triple*  $\{p\}C\{q\}$  validity is defined as follows:

$$\models_{HL} \{p\}C\{q\} \stackrel{def}{\iff} \forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C \rrbracket \Rightarrow s' \in q.$$

Before we show a few examples, it's worth noting that we haven't yet introduced any syntax for our heapless assertions. What is more, we haven't introduced any syntax neither for our program expressions, nor our program predicates. The syntax we will work with will be introduced in the syntactic part of this subsection. To distinguish the yet not introduced syntax from its intended semantics, we shall use the symbols  $\llbracket \cdot \rrbracket$  and wherever possible different notation (e.g. semantic 1 vs syntactic  $1$ ). For example,  $\llbracket x \doteq 1 \rrbracket = \{s \in \mathbf{Stacks} \mid s(x) = 1\}$  and  $\llbracket x + 1 \rrbracket = \lambda s. s(x) + 1$ <sup>4</sup>. Now, we are ready to show a few examples of valid and invalid heapless Hoare triples:

$$\begin{aligned} \models_{HL} \{\llbracket x \doteq 0 \rrbracket\}x := \llbracket x + 1 \rrbracket\{\llbracket x \doteq 1 \rrbracket\} & \not\models_{HL} \{\llbracket x \doteq 0 \rrbracket\}x := \llbracket x + 1 \rrbracket\{\llbracket x \doteq 0 \rrbracket\} \\ \models_{HL} \{\llbracket x \doteq 0 \rrbracket\}y := \llbracket x + 1 \rrbracket\{\llbracket y \doteq 1 \rrbracket\} & \not\models_{HL} \{\llbracket x \doteq 0 \rrbracket\}y := \llbracket x + 1 \rrbracket\{\llbracket y \doteq 0 \rrbracket\} \end{aligned}$$

Of course, we could've written them without involving any syntax:

$$\models_{HL} \{\{s \in \mathbf{Stacks} \mid s(x) = 0\}\}x := (\lambda s. s(x) + 1)\{\{s \in \mathbf{Stacks} \mid s(x) = 1\}\}$$

However, if we choose to opt out of such syntax, we risk missing the core idea in technicalities. To ease the readability even further<sup>5</sup>, later on we will write  $\models_{HL} \{x = 0\}x := x + 1\{x = 1\}$  instead of  $\models_{HL} \{\llbracket x \doteq 0 \rrbracket\}x := \llbracket x + 1 \rrbracket\{\llbracket x \doteq 1 \rrbracket\}$ . A heapless Hoare triple is also called partial correctness specification. Partial, because for  $\{p\}C\{q\}$  to be valid, it is not necessary for the execution of  $C$  to terminate, when started in a state satisfying  $p$ . For example the following heapless Hoare triple  $\{p\}\mathbf{while} \llbracket 0 \doteq 0 \rrbracket \mathbf{do} x := \llbracket x + 1 \rrbracket \mathbf{od}\{q\}$  is a valid for any  $p$  and  $q$ . Having said that, what we are usually interested in is total correctness. To obtain total correctness, we usually prove partial correctness and termination separately.

### 2.1.2 Syntax

We begin this technical part with the syntax of program expressions and program predicates. As a rule of thumb, the syntactic equivalents of the semantic objects will use the same meta symbols, but in caligraphic font. The syntax we are to introduce has a standard model with universe  $\mathbf{Val} = \mathbb{N}$ .

**Definition 2.11.** We define  $\mathbb{N}$ -program expressions using BNF:

$$e \Leftarrow 0 \mid 1 \mid x \mid (e + e) \mid (e \cdot e)$$

where  $x \in \mathbf{PVars}$ .

<sup>4</sup>We use lambda notation, but, formally, a function to us is defined set-theoretically.

<sup>5</sup>Most sources omit these details, usually writing  $\models_{HL} \{x = 0\}x := x + 1\{x = 1\}$ .



We denote the set of all  $\mathbb{N}$ -program expressions by  $\mathbf{AExp}_{\mathbb{N}}$ . We introduce numerals in the standard way, e.g. the numeral  $4$  stands for  $((1 + 1) + 1) + 1$ . The interpretation<sup>6</sup> (of the functional symbols  $0, 1, +, \cdot$ ) of interest is also the standard interpretation:

$$\begin{aligned} \llbracket 0 \rrbracket(s) &\Leftarrow 0 \\ \llbracket 1 \rrbracket(s) &\Leftarrow 1 \\ \llbracket x \rrbracket(s) &\Leftarrow s(x) \\ \llbracket e_1 + e_2 \rrbracket(s) &\Leftarrow \llbracket e_1 \rrbracket(s) + \llbracket e_2 \rrbracket(s) \\ \llbracket e_1 \cdot e_2 \rrbracket(s) &\Leftarrow \llbracket e_1 \rrbracket(s) \cdot \llbracket e_2 \rrbracket(s) \end{aligned}$$

We say that  $\llbracket e \rrbracket(s)$  is the value of the ( $\mathbb{N}$ -)program expression  $e$  in the considered interpretation  $\llbracket \cdot \rrbracket$ . Thus,  $\llbracket e \rrbracket$  is a function from  $\mathbf{Stacks}$  to  $\mathbb{N}$ , i.e.  $\llbracket e \rrbracket$  is a program expression (with  $\mathbf{Val} = \mathbb{N}$ ).

Note that we use the same symbols for syntactic and semantic addition and multiplication. This syntax is given for concreteness. We could've added other standard operations. However, to ensure totality (see definition 2.3), if we were to introduce division, for instance, we would need to determine how to handle division by zero. One possibility is to define it as always equal to some integer, say 0. Alternatively, we could introduce a new symbol that we interpret as "undefined." Another important point to note is that we require only totality and not computability (see definition 2.3). In real-world scenarios, computability might always be assumed, even though many results don't rely on it.

**Definition 2.12.** *We define  $\mathbb{N}$ -program predicates using BNF:*

$$\mathcal{E} \Leftarrow \top \mid \perp \mid (e < e) \mid (e \doteq e) \mid (\mathcal{E} \wedge \mathcal{E}) \mid (\mathcal{E} \vee \mathcal{E}) \mid (\mathcal{E} \Rightarrow \mathcal{E}) \mid (\mathcal{E} \Leftrightarrow \mathcal{E}) \mid \neg \mathcal{E}$$

We denote the set of all  $\mathbb{N}$ -program expressions by  $\mathbf{BExp}_{\mathbb{N}}$ . The interpretation (of the predicate symbols  $\top, \perp, <, \doteq$  and  $\Rightarrow$ ) of interest is the standard interpretation<sup>7</sup>:

$$\begin{aligned} \llbracket \top \rrbracket &\Leftarrow \mathbf{Stacks} \\ \llbracket \perp \rrbracket &\Leftarrow \emptyset \\ \llbracket e_1 < e_2 \rrbracket &\Leftarrow \{s \in \mathbf{Stacks} \mid \llbracket e_1 \rrbracket(s) < \llbracket e_2 \rrbracket(s)\} \\ \llbracket e_1 \doteq e_2 \rrbracket &\Leftarrow \{s \in \mathbf{Stacks} \mid \llbracket e_1 \rrbracket(s) = \llbracket e_2 \rrbracket(s)\} \\ \llbracket \mathcal{E}_1 \wedge \mathcal{E}_2 \rrbracket &\Leftarrow \{s \in \mathbf{Stacks} \mid s \in \llbracket \mathcal{E}_1 \rrbracket \wedge s \in \llbracket \mathcal{E}_2 \rrbracket\} = \llbracket \mathcal{E}_1 \rrbracket \cap \llbracket \mathcal{E}_2 \rrbracket \\ \llbracket \mathcal{E}_1 \vee \mathcal{E}_2 \rrbracket &\Leftarrow \{s \in \mathbf{Stacks} \mid s \in \llbracket \mathcal{E}_1 \rrbracket \vee s \in \llbracket \mathcal{E}_2 \rrbracket\} = \llbracket \mathcal{E}_1 \rrbracket \cup \llbracket \mathcal{E}_2 \rrbracket \\ \llbracket \mathcal{E}_1 \Rightarrow \mathcal{E}_2 \rrbracket &\Leftarrow \{s \in \mathbf{Stacks} \mid s \in \llbracket \mathcal{E}_1 \rrbracket \Rightarrow s \in \llbracket \mathcal{E}_2 \rrbracket\} = \overline{\llbracket \mathcal{E}_1 \rrbracket} \cup \llbracket \mathcal{E}_2 \rrbracket \\ \llbracket \mathcal{E}_1 \Leftrightarrow \mathcal{E}_2 \rrbracket &\Leftarrow \{s \in \mathbf{Stacks} \mid s \in \llbracket \mathcal{E}_1 \rrbracket \Leftrightarrow s \in \llbracket \mathcal{E}_2 \rrbracket\} = (\llbracket \mathcal{E}_1 \rrbracket \cap \llbracket \mathcal{E}_2 \rrbracket) \cup (\overline{\llbracket \mathcal{E}_1 \rrbracket} \cap \overline{\llbracket \mathcal{E}_2 \rrbracket}) \\ \llbracket \neg \mathcal{E} \rrbracket &\Leftarrow \{s \in \mathbf{Stacks} \mid s \notin \llbracket \mathcal{E} \rrbracket\} = \overline{\llbracket \mathcal{E} \rrbracket} \end{aligned}$$

<sup>6</sup>In the sense of Tarski.

<sup>7</sup>Note that we present not only the interpretation of the predicate symbols, but also remind Tarski's recursive definition of truth of non-quantified first-order logic formulae. Moreover, we did the same for the evaluation of the terms in definition 2.11.

Whenever  $s \in \llbracket \ell \rrbracket$ , we say that  $s$  satisfies  $\ell$  in the considered interpretation  $\llbracket \cdot \rrbracket$ . The relation  $\{\langle s, \llbracket \ell \rrbracket \rangle \mid s \in \llbracket \ell \rrbracket\}$  is the standard Tarski's satisfaction relation. So,  $\llbracket \ell \rrbracket$  represents the satisfaction relation in the sense that it is the set of all states  $s$  satisfying  $\ell$ . Remark that  $\llbracket \ell \rrbracket$  is a program predicate (with  $\text{Val} = \mathbb{N}$ ).

Note that we use the same symbols for syntactic and semantic logical operators. This syntax is given for concreteness. Similar to program expressions, we don't require program predicates to be computable. In real-world scenarios, computability might always be assumed, even though many results don't rely on it.

**Definition 2.13.** *We define heapless  $\mathbb{N}$ -assertions using BNF:*

$$\rho \Leftarrow \top \mid \perp \mid (e \leq e) \mid (e \doteq e) \mid (\rho \wedge \rho) \mid (\rho \vee \rho) \mid (\rho \Rightarrow \rho) \mid (\rho \Leftrightarrow \rho) \mid \neg \rho \mid \exists x. \rho \mid \forall x. \rho,$$

where  $x \in \text{PVars}$ .

We denote the set of all heapless  $\mathbb{N}$ -assertions by  $\text{SynSArts}_{\mathbb{N}}$ . The interpretation of interest is the standard interpretation (as in definition 2.12). We remind that:

$$\begin{aligned} \llbracket \exists x. \rho \rrbracket &\Leftarrow \{s \in \text{Stacks} \mid \exists n. s_x^n \in \llbracket \rho \rrbracket\} \\ \llbracket \forall x. \rho \rrbracket &\Leftarrow \{s \in \text{Stacks} \mid \forall n. s_x^n \in \llbracket \rho \rrbracket\} \end{aligned}$$

Whenever  $s \in \llbracket \rho \rrbracket$ , we say that  $s$  satisfies  $\rho$  in the considered interpretation  $\llbracket \cdot \rrbracket$ . The relation  $\{\langle s, \llbracket \rho \rrbracket \rangle \mid s \in \llbracket \rho \rrbracket\}$  is the standard Tarski's satisfaction relation. So,  $\llbracket \rho \rrbracket$  represents the satisfaction relation in the sense that it is the set of all states  $s$  satisfying  $\rho$ . Remark that  $\llbracket \rho \rrbracket$  is a heapless assertion (with  $\text{Val} = \mathbb{N}$ ).

Note that we use the same symbols for syntactic and semantic logical operators and quantifiers. This syntax is given for concreteness. Moreover, recall that the notions of heapless assertions (definition 2.2) and program predicate (definition 2.4) coincided, whereas heapless  $\mathbb{N}$ -assertion (definition 2.13) generalizes  $\mathbb{N}$ -program predicate (definition 2.12).

As previously mentioned, this syntax is provided for concreteness. In general (see [Cook 1978]), we consider two first-order languages,  $\mathcal{L}_1$  and  $\mathcal{L}_2$ , where  $\mathcal{L}_2$  is an extension of  $\mathcal{L}_1$ ,  $\text{AExps}$  are the terms of  $\mathcal{L}_1$ ,  $\text{BExps}$  are the quantifier-free formulae of  $\mathcal{L}_1$  and  $\text{SynSArts}$  are the formulae of  $\mathcal{L}_2$ . That is, the first language,  $\mathcal{L}_1$ , is for the programming language itself, while the second language,  $\mathcal{L}_2$ , is for expressing program specifications.

**Definition 2.14.** *We define heapless (AExps, BExps)-program commands using BNF:*

$$\mathcal{C} \Leftarrow \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } \ell \mid (\mathcal{C}; \mathcal{C}) \mid (\mathcal{C} + \mathcal{C}) \mid \mathcal{C}^*,$$

where  $x \in \text{PVars}$ ,  $e \in \text{AExps}$ ,  $\ell \in \text{BExps}$ .

We denote the set of all heapless (AExps, BExps)-program commands with  $\text{SCom}(\text{AExps}, \text{BExps})$ . We define  $\text{SCom}_{\mathbb{N}}$  as  $\text{SCom}(\text{AExps}_{\mathbb{N}}, \text{BExps}_{\mathbb{N}})$ . A distinction between heapless program commands and heapless (AExps, BExps)-program

commands is done, since the small-step semantic relation (fig. 2.1) doesn't depend on the language (and its interpretation of interest) used for program expressions and predicates. To apply the small-step semantics relation for a concrete interpretation (and hence language)  $\llbracket \cdot \rrbracket$  we introduce a translation function  $T_{\llbracket \cdot \rrbracket}$ , where  $T_{\llbracket \cdot \rrbracket}(\mathcal{C})$  is the translation of a heapless (AExps, BExps)-program command  $\mathcal{C}$  to a heapless program command  $T_{\llbracket \cdot \rrbracket}(\mathcal{C})$ :

$$\begin{aligned}
T_{\llbracket \cdot \rrbracket}(\mathbf{skip}) &\Leftarrow \mathbf{skip} \\
T_{\llbracket \cdot \rrbracket}(x := e) &\Leftarrow x := \llbracket e \rrbracket \\
T_{\llbracket \cdot \rrbracket}(x := \mathbf{nonDet}()) &\Leftarrow x := \mathbf{nonDet}() \\
T_{\llbracket \cdot \rrbracket}(\mathbf{assume } \mathcal{C}) &\Leftarrow \mathbf{assume } \llbracket \mathcal{C} \rrbracket \\
T_{\llbracket \cdot \rrbracket}(\mathcal{C}_1; \mathcal{C}_2) &\Leftarrow T_{\llbracket \cdot \rrbracket}(\mathcal{C}_1); T_{\llbracket \cdot \rrbracket}(\mathcal{C}_2) \\
T_{\llbracket \cdot \rrbracket}(\mathcal{C}_1 + \mathcal{C}_2) &\Leftarrow T_{\llbracket \cdot \rrbracket}(\mathcal{C}_1) + T_{\llbracket \cdot \rrbracket}(\mathcal{C}_2) \\
T_{\llbracket \cdot \rrbracket}(\mathcal{C}^*) &\Leftarrow T_{\llbracket \cdot \rrbracket}(\mathcal{C})^*
\end{aligned}$$

We denote  $\llbracket \mathcal{C} \rrbracket \Leftarrow \llbracket T_{\llbracket \cdot \rrbracket}(\mathcal{C}) \rrbracket$ <sup>8</sup> and will now summarize the usage of the notation  $\llbracket \cdot \rrbracket$ :

1.  $\llbracket C \rrbracket$ : the input-output relation of  $C$ ;
2.  $\llbracket e \rrbracket$ : the program expression, corresponding to  $e$  in the considered interpretation  $\llbracket \cdot \rrbracket$ ;
3.  $\llbracket \mathcal{C} \rrbracket$ : the program predicate, corresponding to  $\mathcal{C}$  in the considered interpretation  $\llbracket \cdot \rrbracket$ ;
4.  $\llbracket \mathcal{P} \rrbracket$ : the heapless assertion, corresponding to  $\mathcal{P}$  in the considered interpretation  $\llbracket \cdot \rrbracket$ ;
5.  $\llbracket \mathcal{C} \rrbracket$ : the input-output relation of  $\mathcal{C}$  for interpreted in  $\llbracket \cdot \rrbracket$  symbols from AExps and BExps.

In general ([Cook 1978]), the middle three usages of  $\llbracket \cdot \rrbracket$  are expressed solely through an interpretation  $\mathcal{I}$  for  $\mathcal{L}_2$  (which is also an interpretation for  $\mathcal{L}_1$ ). The last usage gives us the value of  $\mathcal{C}$ ,  $\llbracket \mathcal{C} \rrbracket$ , in the so-called ([Cook 1978]) interpretive model  $\mathcal{M}$ , which is induced by the interpretation  $\mathcal{I}$ , i.e.  $\mathcal{M} = \mathcal{M}[\mathcal{I}]$ . The first usage gives us the "blueprint" for obtaining  $\mathcal{M}$  from  $\mathcal{I}$ . Remark that when not explicitly stated, one cannot differentiate neither  $\llbracket \mathbf{skip} \rrbracket$  (the first case) from  $\llbracket \mathbf{skip} \rrbracket$  (the fifth case), nor  $\llbracket x := \mathbf{nonDet}() \rrbracket$  (the first case) from  $\llbracket x := \mathbf{nonDet}() \rrbracket$  (the fifth case). However, this is not an issue, as the input-output relations for the first and fifth cases coincide in both instances of ambiguity.

**Definition 2.15.** A *heapless Hoare formula*  $\{\mathcal{P}\}\mathcal{C}\{\mathcal{Q}\}$  is a syntactic object composed of  $\mathcal{P}, \mathcal{Q} \in \text{SynSArts}_{\mathbb{N}}$ , each surrounded by the symbols  $\{ \text{ and } \}$  and  $\mathcal{C} \in \text{SCom}_{\mathbb{N}}$  between them.

<sup>8</sup>To eliminate ambiguity, consider  $\llbracket \mathcal{C} \rrbracket' = \llbracket T_{\llbracket \cdot \rrbracket}(\mathcal{C}) \rrbracket$ .

$$\begin{array}{c}
\frac{}{\vdash_{HL, \mathcal{D}} \{\rho\} \mathbf{skip} \{\rho\}} \text{(Skip)} \quad \frac{\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C} \{\rho\}}{\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}^* \{\rho\}} \text{(Iter)} \quad \frac{\text{sc}_q(x) \cap \text{fv}(e) = \emptyset}{\vdash_{HL, \mathcal{D}} \{q[e/x]\} x := e \{q\}} \text{(Assign)} \\
\\
\frac{}{\vdash_{HL, \mathcal{D}} \{\rho\} x := \text{nonDet}() \{\exists x. \rho\}} \text{(Havoc)} \quad \frac{\vdash_{\mathcal{D}} \rho \Rightarrow \rho' \quad \vdash_{\mathcal{D}} q' \Rightarrow q \quad \vdash_{HL, \mathcal{D}} \{\rho'\} \mathcal{C} \{q'\}}{\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C} \{q\}} \text{(Cons)} \\
\\
\frac{\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1 \{q\} \quad \vdash_{HL, \mathcal{D}} \{q\} \mathcal{C}_2 \{\rho\}}{\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1; \mathcal{C}_2 \{\rho\}} \text{(Seq)} \quad \frac{\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1 \{q\} \quad \vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_2 \{q\}}{\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1 + \mathcal{C}_2 \{q\}} \text{(Choice)} \\
\\
\frac{}{\vdash_{HL, \mathcal{D}} \{\rho\} \mathbf{assume} \ \ell \{\rho \wedge \ell\}} \text{(Assume)}
\end{array}$$

Figure 2.2: Axioms and rules of inference of Hoare logic relative to  $\mathcal{D}$ , where  $q[e/x]$  is  $q$  with  $x$  syntactically replaced by  $e$ ,  $\text{fv}(e)$  is the set of free variables in  $e$  and  $\text{sc}_q(x)$  is the set of quantified variables in  $q$  whose scope contain  $x$  as a free variable, e.g.  $\text{sc}_{\exists t. \exists x. x \dot{=} t}(x) = \emptyset$  and  $\text{sc}_{(\exists y. x \dot{=} z) \wedge (\forall z. x \dot{=} z) \Rightarrow (\exists y. \forall t. y < t)}(x) = \{y, z\}$ .

**Definition 2.16.** *Let  $\mathcal{D}$  be a (not necessarily effective) deductive system for  $\text{SynSAsrts}_{\mathbb{N}}$ . We define in figure 2.2 Hoare logic relative to  $\mathcal{D}$ .*

We will use the meta symbol  $\mathcal{D}$  with potential indices to denote a (not necessarily effective) deductive system. Remark that, in general (see [Cook 1978]) Hoare logic (relative to some  $\mathcal{D}$ ) depends on  $\mathcal{L}_1$  and  $\mathcal{L}_2$  i.e. depends on  $\text{AExps}$ ,  $\text{BExps}$  and  $\text{SynSAsrts}$ . For simplicity, we consider only the case when  $\text{AExps}$ ,  $\text{BExps}$  and  $\text{SynSAsrts}$  are  $\text{AExps}_{\mathbb{N}}$ ,  $\text{BExps}_{\mathbb{N}}$  and  $\text{SynSAsrts}_{\mathbb{N}}$ , respectively.

In [Cook 1978], a formal proof of a formula  $\{\rho\} \mathcal{C} \{q\}$  in HL (relative to some  $\mathcal{D}$ ) is a finite sequence of formulae, each either an axiom of HL, a formula deducible in  $\mathcal{D}$ , or follows from earlier formulae in the sequence by one of the rules of HL, where the last element of the sequence, which we will call a theorem of HL (relative to  $\mathcal{D}$ ), is  $\{\rho\} \mathcal{C} \{q\}$ . They introduce notation  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C} \{q\}$  to indicate that  $\{\rho\} \mathcal{C} \{q\}$  is provable in that sense. Our approach, however, differs in that we define  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C} \{q\}$  as a recursively defined predicate rather than a sequence-based proof<sup>9</sup>. Nevertheless, we will use both perspectives throughout the thesis and trust that the reader will understand the context in which each is applied.

To the best of our knowledge, every source discussing Hoare logics presents the assignment axiom without the restriction we consider. While we have not examined the soundness of the unrestricted assignment axiom, we demonstrate the soundness of the restricted version. We argue that this restricted form is more natural, as its semantic equivalent precisely represents the weakest liberal precondition, as defined in [Clarke et al. 1984]<sup>10</sup>. Furthermore, we establish

<sup>9</sup>Note that, like Cook, we refer to actual finiteness, not non-standard notions. That is, only if  $\mathcal{D}$  is effective.

<sup>10</sup>In this thesis, however, we will focus on its dual concept, known as strongest postcondition (see definition 2.20).

that this restricted assignment axiom is sufficient to demonstrate (non-effective) completeness. It is worth noting that completeness can also be achieved with an even more restricted assignment axiom, which imposes the condition that  $\text{bd}(\varphi) \cap \text{fv}(e) = \emptyset$ , where  $\text{bd}(\varphi)$  is the set of all bounded variables in  $\varphi$ .

### 2.1.3 Undecidability and effective incompleteness

Consider the heapless Hoare triple  $\{\llbracket \top \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C})\{\llbracket \perp \rrbracket\}$ ,  $\mathcal{C} \in \text{SCom}_{\mathbb{N}}$ .

$$\begin{aligned} \models_{HL} \{\llbracket \top \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C})\{\llbracket \perp \rrbracket\} &\iff \forall s \in \llbracket \top \rrbracket. \forall s'. \langle s, s' \rangle \in \llbracket T_{\llbracket \cdot \rrbracket}(\mathcal{C}) \rrbracket \Rightarrow s' \in \llbracket \perp \rrbracket \\ &\iff \forall s \in \text{Stacks}. \forall s'. \langle s, s' \rangle \in \llbracket \mathcal{C} \rrbracket \Rightarrow s' \in \emptyset \\ &\iff \forall s. \nexists s'. \langle s, s' \rangle \in \llbracket \mathcal{C} \rrbracket \end{aligned}$$

Suppose we could algorithmically decide whether  $\models_{HL} \{p\}C\{q\}$  for any  $p, q$  and  $C$ . In particular we could algorithmically decide whether  $\models_{HL} \{\llbracket \top \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C})\{\llbracket \perp \rrbracket\}$ . Then by the equivalence, we would have an algorithmic way to decide whether  $\forall s. \nexists s'. \langle s, s' \rangle \in \llbracket \mathcal{C} \rrbracket$ . But  $\mathcal{C} \in \text{SCom}_{\mathbb{N}}$  was arbitrary. Thus for any  $\mathcal{C} \in \text{SCom}_{\mathbb{N}}$  we can algorithmically decide whether  $\forall s. \nexists s'. \langle s, s' \rangle \in \llbracket \mathcal{C} \rrbracket$ . But the expressivity of  $\text{SCom}_{\mathbb{N}}$  (for the considered interpretive model) is obviously containing the computable functions, therefore we can algorithmically decide the complement of the halting problem (which isn't even recursively enumerable) - contradiction. Therefore the problem of validity of heapless Hoare triples is undecidable. Moreover, the set of all non-terminating programs

$$\{\mathcal{C} \in \text{SCom}_{\mathbb{N}} \mid \models_{HL} \{\llbracket \top \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C})\{\llbracket \perp \rrbracket\}\}$$

is not r.e. Hence, assuming soundness of HL<sup>11</sup> relative to some effective  $\mathcal{D}$ , we cannot have semantic completeness (relative to  $\llbracket \cdot \rrbracket$ ). Indeed, all theorems of such proof system are r.e. and hence

$$\{\mathcal{C} \in \text{SCom}_{\mathbb{N}} \mid \vdash_{HL, \mathcal{D}} \{\top\} \mathcal{C} \{\perp\}\}$$

is r.e. However,

$$\forall \mathcal{C} \in \text{SCom}_{\mathbb{N}}. \vdash_{HL, \mathcal{D}} \{\top\} \mathcal{C} \{\perp\} \Leftrightarrow \models_{HL} \{\llbracket \top \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C})\{\llbracket \perp \rrbracket\}$$

by soundness and completeness and therefore

$$\{\mathcal{C} \in \text{SCom}_{\mathbb{N}} \mid \vdash_{HL, \mathcal{D}} \{\top\} \mathcal{C} \{\perp\}\} = \{\mathcal{C} \in \text{SCom}_{\mathbb{N}} \mid \models_{HL} \{\llbracket \top \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C})\{\llbracket \perp \rrbracket\}\},$$

which is not r.e. - contradiction. Therefore, for any effective  $\mathcal{D}$ , sound relative to  $\llbracket \cdot \rrbracket$ , HL relative to  $\mathcal{D}$  is semantically incomplete (relative to  $\llbracket \cdot \rrbracket$ ) [Cook 1978, Theorem 2, where  $\mathcal{S} = \llbracket \cdot \rrbracket$ ].

<sup>11</sup>In fact, it is sound for every (not necessary effective)  $\mathcal{D}$ , which is sound relative to  $\llbracket \cdot \rrbracket$  (see corollary 2.18).

### 2.1.4 Soundness and non-effective completeness

We say a formula  $\rho$  is true in an interpretation  $\llbracket \cdot \rrbracket$  iff for all valuations  $s$  of the free variables, we have  $s \in \llbracket \rho \rrbracket$ . That is, in our case iff  $\llbracket \rho \rrbracket = \mathbf{Stacks}$ . Equivalently, since every interpretation induces exactly one interpretive model and every interpretive model is induced by an interpretation, we say a formula  $\rho$  is true in an interpretive model, induced by  $\llbracket \cdot \rrbracket$ , iff  $\llbracket \rho \rrbracket = \mathbf{Stacks}$ . A formula  $\{\rho\}\mathcal{E}\{q\}$  is true in interpretive model, induced by  $\llbracket \cdot \rrbracket$ , iff  $\models_{HL} \{\llbracket \rho \rrbracket\}T_{\llbracket \cdot \rrbracket}(\mathcal{E})\{\llbracket q \rrbracket\}$ . A formula  $\{\rho\}\mathcal{E}\{q\}$  is valid (sound) iff it is true for any interpretive model. A rule  $\frac{\alpha_1 \dots \alpha_n}{\beta}$  is valid (sound) iff  $\beta$  is true for any interpretive model in which  $\alpha_1, \dots, \alpha_n$  are true.<sup>12</sup>

**Theorem 2.17.** *The axioms and rules of HL are valid:*

- (i)  $\models_{HL} \{p\}\mathbf{skip}\{p\}$ ;
- (ii)  $\models_{HL} \{\{s \mid s_x^{e(s)} \in q\}\}x := e\{q\}$ ;
- (iii)  $\models_{HL} \{p\}x := \mathbf{nonDet}(\{\{s_x^n \mid s \in p \wedge n \in \mathbf{Val}\}\})$ ;
- (iv)  $p \subseteq p' \Rightarrow q' \subseteq q \Rightarrow \models_{HL} \{p'\}C\{q'\} \Rightarrow \models_{HL} \{p\}C\{q\}$ <sup>13</sup>;
- (v)  $\models_{HL} \{p\}\mathbf{assume} b\{p \cap b\}$ ;
- (vi)  $\models_{HL} \{p\}C_1\{q\} \Rightarrow \models_{HL} \{q\}C_2\{r\} \Rightarrow \models_{HL} \{p\}C_1; C_2\{r\}$ ;
- (vii)  $\models_{HL} \{p\}C_1\{q\} \Rightarrow \models_{HL} \{p\}C_2\{q\} \Rightarrow \models_{HL} \{p\}C_1 + C_2\{q\}$ ;
- (viii)  $\models_{HL} \{p\}C\{p\} \Rightarrow \models_{HL} \{p\}C^*\{p\}$ .

The precondition in (ii) is  $\{s \mid s_x^{e(s)} \in q\}$ , whereas in the axiom it is  $q[e/x]$ . That's because for any interpretation  $\llbracket \cdot \rrbracket$  we've  $\llbracket q[e/x] \rrbracket = \{s \mid s_x^{\llbracket e \rrbracket(s)} \in \llbracket q \rrbracket\}$ , where  $\mathbf{sc}_q(x) \cap \mathbf{fv}(e) = \emptyset$ <sup>14</sup>. In (iii) we require  $\{s_x^n \mid s \in p \wedge n \in \mathbf{Val}\}$ , whereas in the axiom we require  $\exists x. \rho$ . That's because for any interpretation  $\llbracket \cdot \rrbracket$ , we've  $\llbracket \exists x. \rho \rrbracket = \{s_x^n \mid s \in \llbracket \rho \rrbracket \wedge n \in \mathbf{Val}\}$ . In (iv) we require  $p \subseteq p'$  and  $q' \subseteq q$ , whereas in the axiom we require  $\vdash_{\mathcal{L}} \rho \Rightarrow \rho'$  and  $\vdash_{\mathcal{L}} q' \Rightarrow q$ . That's because by the soundness theorem (for FOL) from  $\vdash_{\mathcal{L}} \rho \Rightarrow \rho'$  and  $\vdash_{\mathcal{L}} q' \Rightarrow q$  we know that for any interpretation  $\llbracket \cdot \rrbracket$  of  $\mathcal{L}_2$  we've  $\llbracket \rho \Rightarrow \rho' \rrbracket = \mathbf{Stacks}$  and  $\llbracket q' \Rightarrow q \rrbracket = \mathbf{Stacks}$ . That is  $\llbracket \rho \rrbracket \cup \llbracket \rho' \rrbracket = \mathbf{Stacks}$  and  $\llbracket q' \rrbracket \cup \llbracket q \rrbracket = \mathbf{Stacks}$ , i.e.  $\llbracket \rho \rrbracket \subseteq \llbracket \rho' \rrbracket$  and  $\llbracket q' \rrbracket \subseteq \llbracket q \rrbracket$ . In (v) we use that for any interpretation  $\llbracket \cdot \rrbracket$  of  $\mathcal{L}_2$  we've  $\llbracket \rho \wedge q \rrbracket = \llbracket \rho \rrbracket \cap \llbracket q \rrbracket$  and that  $\mathcal{L}_2$  extends  $\mathcal{L}_1$ .

<sup>12</sup>Recall the two perspectives discussed at the end of the Syntax subsection.

<sup>13</sup>We consider implication to be right-associative.

<sup>14</sup>Consider  $q \Leftarrow \exists y. x = y$  and  $e \Leftarrow y + 1$ . We've that  $\mathbf{sc}_{\exists y. x = y}(x) \cap \mathbf{fv}(y + 1) = \{y\} \neq \emptyset$  and hence  $\llbracket q[e/x] \rrbracket = \{s \mid s_x^{\llbracket e \rrbracket(s)} \in \llbracket q \rrbracket\}$  is not guaranteed. Indeed,  $\emptyset = \llbracket (\exists y. x = y)[y + 1/x] \rrbracket \neq \{s \mid s_x^{\llbracket y + 1 \rrbracket(s)} \in \llbracket \exists y. x = y \rrbracket\} = \mathbf{Stacks}$ .

*Proof of theorem.*

- (i) (i)  $\xLeftrightarrow{def} \forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket \mathbf{skip} \rrbracket \Rightarrow s' \in p$   
 $\xLeftrightarrow{lem} \forall s \in p. \forall s'. \langle s, s' \rangle \in \{\langle s, s \rangle \mid s \in \mathbf{Stacks}\} \Rightarrow s' \in p$   
 $\iff \forall s \in p. \forall s'. s' = s \Rightarrow s' \in p$
- (ii) (ii)  $\xLeftrightarrow{def} \forall s \in \{s \mid s_x^{e(s)} \in q\}. \forall s'. \langle s, s' \rangle \in \llbracket x := e \rrbracket \Rightarrow s' \in q$   
 $\xLeftrightarrow{lem} \forall s \in \{s \mid s_x^{e(s)} \in q\}. \forall s'. \langle s, s' \rangle \in \{\langle s, s_x^{e(s)} \rangle \mid s \in \mathbf{Stacks}\} \Rightarrow s' \in q$   
 $\iff \forall s. s_x^{e(s)} \in q \Rightarrow \forall s'. s' = s_x^{e(s)} \Rightarrow s' \in q$
- (iii) (iii)  $\xLeftrightarrow{def} \forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket x := \mathbf{nonDet}() \rrbracket \Rightarrow s' \in \{s_x^n \mid s \in p \wedge n \in \mathbf{Val}\}$   
 $\xLeftrightarrow{lem} \forall s \in p. \forall s'. \langle s, s' \rangle \in \{\langle s, s_x^n \rangle \mid s \in \mathbf{Stacks} \wedge n \in \mathbf{Val}\} \Rightarrow s' \in \{s_x^n \mid s \in p \wedge n \in \mathbf{Val}\}$   
 $\iff \forall s \in p. \forall s'. \exists n \in \mathbf{Val}. s' = s_x^n \Rightarrow s' \in \{s_x^n \mid s \in p \wedge n \in \mathbf{Val}\}$
- (iv) We've  $\models_{HL} \{p'\}C\{q'\}$ , i.e.  $\forall s \in p'. \forall s'. \langle s, s' \rangle \in \llbracket C \rrbracket \Rightarrow s' \in q'$ . Now, since  $p \subseteq p'$  and  $q' \subseteq q$ , it immediately follows  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C \rrbracket \Rightarrow s' \in q$ , i.e.  $\models_{HL} \{p\}C\{q\}$ .
- (v) (v)  $\xLeftrightarrow{def} \forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket \mathbf{assumes } b \rrbracket \Rightarrow s' \in p \cap b$   
 $\xLeftrightarrow{lem} \forall s \in p. \forall s'. \langle s, s' \rangle \in \{\langle s, s' \rangle \mid s \in b\} \Rightarrow s' \in p \cap b$   
 $\iff \forall s \in p. \forall s'. s' = s \wedge s' \in b \Rightarrow s' \in p \cap b$
- (vi) We've  $\models_{HL} \{p\}C_1\{q\}$  and  $\models_{HL} \{q\}C_2\{r\}$ . That is, by definition, we've  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C_1 \rrbracket \Rightarrow s' \in q$  and  $\forall s' \in q. \forall s''. \langle s', s'' \rangle \in \llbracket C_2 \rrbracket \Rightarrow s'' \in r$ . We claim that  $\forall s \in p. \forall s''. \langle s, s'' \rangle \in \llbracket C_1; C_2 \rrbracket \Rightarrow s'' \in r$ . That is, by lemma 2.38,  $\forall s \in p. \forall s''. \langle s, s'' \rangle \in \llbracket C_1 \rrbracket \circ \llbracket C_2 \rrbracket \Rightarrow s'' \in r$ . Hence, we claim that  $\forall s \in p. \forall s''. \exists s'. \langle s, s' \rangle \in \llbracket C_1 \rrbracket \wedge \langle s', s'' \rangle \in \llbracket C_2 \rrbracket \Rightarrow s'' \in r$  by the definition of composition. Let  $s, s''$  be such that  $s \in p$  and obtain  $s'$  such that  $\langle s, s' \rangle \in \llbracket C_1 \rrbracket \wedge \langle s', s'' \rangle \in \llbracket C_2 \rrbracket$ . Then we obtain that  $s' \in q$  from  $s \in p, \langle s, s' \rangle \in \llbracket C_1 \rrbracket$  and  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C_1 \rrbracket \Rightarrow s' \in q$ . We conclude that  $s'' \in r$  using  $\forall s' \in q. \forall s''. \langle s', s'' \rangle \in \llbracket C_2 \rrbracket \Rightarrow s'' \in r$  and  $s' \in q, \langle s', s'' \rangle \in \llbracket C_2 \rrbracket$ .
- (vii) We've  $\models_{HL} \{p\}C_1\{q\}$  and  $\models_{HL} \{p\}C_2\{q\}$ . That is, by definition, we've  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C_1 \rrbracket \Rightarrow s' \in q$  and  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C_2 \rrbracket \Rightarrow s' \in q$ . We claim that  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C_1 + C_2 \rrbracket \Rightarrow s' \in q$ . That is, by lemma 2.38,  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C_1 \rrbracket \cup \llbracket C_2 \rrbracket \Rightarrow s' \in q$ . We easily obtain it, using  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C_1 \rrbracket \Rightarrow s' \in q$  and  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C_2 \rrbracket \Rightarrow s' \in q$ .
- (viii) First, we prove by induction on  $n \in \mathbb{N}$  that  $\models_{HL} \{p\}C^n\{p\}$ , using the assumption, (i) and (vi). We claim that  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C^* \rrbracket \Rightarrow s' \in p$ . That is, by lemma 2.38,  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \bigcup_{n \in \mathbb{N}} \llbracket C^n \rrbracket \Rightarrow s' \in p$ . We easily obtain it, using  $\forall n \in \mathbb{N}. \models_{HL} \{p\}C^n\{p\}$ , or, equivalently,  $\forall n \in \mathbb{N}. \forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C^n \rrbracket \Rightarrow s' \in p$ .

□

We say that  $\mathcal{D}$  is sound relative to an interpretation  $\llbracket \cdot \rrbracket$  iff every deducible formula in  $\mathcal{D}$  is true in  $\llbracket \cdot \rrbracket$ . That is, if for every  $\rho$  such that  $\vdash_{\mathcal{D}} \rho$ , we have  $\llbracket \rho \rrbracket = \text{Stacks}$ .

**Corollary 2.18.** *Let  $\mathcal{D}$  be sound relative to fixed  $\llbracket \cdot \rrbracket$ . Then  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}\{q\} \Rightarrow \models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C})\{\llbracket q \rrbracket\}$ .*

*Proof.* Induction on the recursive definition of the predicate  $\vdash_{HL, \mathcal{D}} \{\rho\} C\{q\}$ :

(i)  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathbf{skip}\{\rho\}$

By theorem 2.17 we have  $\models_{HL} \{\llbracket \rho \rrbracket\} \mathbf{skip}\{\llbracket \rho \rrbracket\}$ . Moreover, by definition  $T_{[\cdot]}(\mathbf{skip}) = \mathbf{skip}$ . Therefore  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathbf{skip})\{\llbracket \rho \rrbracket\}$ ;

(ii)  $\vdash_{HL, \mathcal{D}} \{q[e/x]\} x := e\{q\}$ , where  $\text{sc}_q(x) \cap \text{fv}(e) = \emptyset$

By theorem 2.17 we have  $\models_{HL} \{\{s \mid s_x^{\llbracket e \rrbracket}(s)} \in \llbracket q \rrbracket\}\} x := \llbracket e \rrbracket\{\llbracket q \rrbracket\}$ . Since  $\text{sc}_q(x) \cap \text{fv}(e) = \emptyset$ , we have that  $\llbracket q[e/x] \rrbracket = \{s \mid s_x^{\llbracket e \rrbracket}(s)} \in \llbracket q \rrbracket\}$ . Thus, we've  $\models_{HL} \{\llbracket q[e/x] \rrbracket\} x := \llbracket e \rrbracket\{\llbracket q \rrbracket\}$ . Now, by definition  $T_{[\cdot]}(x := e) = x := \llbracket e \rrbracket$ . Hence,  $\models_{HL} \{\llbracket q[e/x] \rrbracket\} T_{[\cdot]}(x := e)\{\llbracket q \rrbracket\}$ ;

(iii)  $\vdash_{HL, \mathcal{D}} \{\rho\} x := \text{nonDet}()\{\exists x. \rho\}$

By theorem 2.17 we've  $\models_{HL} \{\llbracket \rho \rrbracket\} x := \text{nonDet}()\{\{s_x^n \mid s \in \llbracket \rho \rrbracket \wedge n \in \mathbb{N}\}\}$ . Moreover, we've  $\llbracket \exists x. \rho \rrbracket = \{s_x^n \mid s \in \llbracket \rho \rrbracket \wedge n \in \mathbb{N}\}$ . Therefore, we have  $\models_{HL} \{\llbracket \rho \rrbracket\} x := \text{nonDet}()\{\llbracket \exists x. \rho \rrbracket\}$ . Now, by definition, we've that  $T_{[\cdot]}(x := \text{nonDet}()) = x := \text{nonDet}()$ . Finally, we conclude that the axiom is true in  $\llbracket \cdot \rrbracket$ , i.e.  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(x := \text{nonDet}())\{\llbracket \exists x. \rho \rrbracket\}$  holds;

(iv)  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}\{q\}$ , where i.h. holds for  $\vdash_{HL, \mathcal{D}} \{\rho'\} \mathcal{C}\{q'\}$ ,  $\vdash_{\mathcal{D}} \rho \Rightarrow \rho'$  and  $\vdash_{\mathcal{D}} q' \Rightarrow q$

We've  $\llbracket \rho \rrbracket \subseteq \llbracket \rho' \rrbracket \Rightarrow \llbracket q' \rrbracket \subseteq \llbracket q \rrbracket \Rightarrow \models_{HL} \{\llbracket \rho' \rrbracket\} T_{[\cdot]}(\mathcal{C})\{\llbracket q' \rrbracket\} \Rightarrow \models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C})\{\llbracket q \rrbracket\}$  by theorem 2.17. Moreover, by  $\vdash_{\mathcal{D}} \rho \Rightarrow \rho'$ ,  $\vdash_{\mathcal{D}} q' \Rightarrow q$  and soundness of  $\mathcal{D}$  relative to  $\llbracket \cdot \rrbracket$ , we've  $\llbracket \rho \Rightarrow \rho' \rrbracket = \text{Stacks}$  and  $\llbracket q' \Rightarrow q \rrbracket = \text{Stacks}$ . That is  $\llbracket \rho \rrbracket \subseteq \llbracket \rho' \rrbracket$  and  $\llbracket q' \rrbracket \subseteq \llbracket q \rrbracket$ . Moreover, by the i.h. for  $\vdash_{HL, \mathcal{D}} \{\rho'\} \mathcal{C}\{q'\}$  we've  $\models_{HL} \{\llbracket \rho' \rrbracket\} T_{[\cdot]}(\mathcal{C})\{\llbracket q' \rrbracket\}$ . Therefore, we conclude that  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C})\{\llbracket q \rrbracket\}$ ;

(v)  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathbf{assume} \ell\{\rho \wedge \ell\}$

By theorem 2.17 we've  $\models_{HL} \{\llbracket \rho \rrbracket\} \mathbf{assume} \llbracket \ell \rrbracket\{\llbracket \rho \rrbracket \cap \llbracket \ell \rrbracket\}$ . Moreover, we have that  $\llbracket \rho \wedge \ell \rrbracket = \llbracket \rho \rrbracket \cap \llbracket \ell \rrbracket$ . Hence, we've  $\models_{HL} \{\llbracket \rho \rrbracket\} \mathbf{assume} \llbracket \ell \rrbracket\{\llbracket \rho \wedge \ell \rrbracket\}$ . Now, by definition  $T_{[\cdot]}(\mathbf{assume} \ell) = \mathbf{assume} \llbracket \ell \rrbracket$ . Thus, we conclude that  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathbf{assume} \ell)\{\llbracket \rho \wedge \ell \rrbracket\}$ ;

(vi)  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1; \mathcal{C}_2\{\rho\}$  and let  $q$  be such that i.h. holds for  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1\{q\}$  and  $\vdash_{HL, \mathcal{D}} \{q\} \mathcal{C}_2\{\rho\}$

We've  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1)\{\llbracket q \rrbracket\} \Rightarrow \models_{HL} \{\llbracket q \rrbracket\} T_{[\cdot]}(\mathcal{C}_2)\{\llbracket \rho \rrbracket\} \Rightarrow \models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1); T_{[\cdot]}(\mathcal{C}_2)\{\llbracket \rho \rrbracket\}$  by theorem 2.17. Moreover, by i.h. for  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1\{q\}$  we have that



$\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1) \{\llbracket \varphi \rrbracket\}$  and by i.h. for  $\vdash_{HL, \mathcal{D}} \{\varphi\} \mathcal{C}_2 \{\rho\}$  we've that  $\models_{HL} \{\llbracket \varphi \rrbracket\} T_{[\cdot]}(\mathcal{C}_2) \{\llbracket \rho \rrbracket\}$ . Therefore  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1); T_{[\cdot]}(\mathcal{C}_2) \{\llbracket \rho \rrbracket\}$ . Now, by definition  $T_{[\cdot]}(\mathcal{C}_1; \mathcal{C}_2) = T_{[\cdot]}(\mathcal{C}_1); T_{[\cdot]}(\mathcal{C}_2)$ . Hence, we conclude that  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1; \mathcal{C}_2) \{\llbracket \rho \rrbracket\}$ ;

(vii)  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1 + \mathcal{C}_2 \{\varphi\}$  and i.h. for  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1 \{\varphi\}$  and  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_2 \{\varphi\}$

We've  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1) \{\llbracket \varphi \rrbracket\} \Rightarrow \models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_2) \{\llbracket \varphi \rrbracket\} \Rightarrow \models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1) + T_{[\cdot]}(\mathcal{C}_2) \{\llbracket \varphi \rrbracket\}$  by theorem 2.17. Moreover, by i.h. for  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_1 \{\varphi\}$  we have that  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1) \{\llbracket \varphi \rrbracket\}$  and by i.h. for  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}_2 \{\varphi\}$  we've that  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_2) \{\llbracket \varphi \rrbracket\}$ . Therefore  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1) + T_{[\cdot]}(\mathcal{C}_2) \{\llbracket \varphi \rrbracket\}$ . Now, by definition  $T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2) = T_{[\cdot]}(\mathcal{C}_1) + T_{[\cdot]}(\mathcal{C}_2)$ . Hence, we conclude that  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2) \{\llbracket \varphi \rrbracket\}$ ;

(viii)  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C}^* \{\rho\}$  and i.h. holds for  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C} \{\rho\}$

By theorem 2.17 we have  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}) \{\llbracket \rho \rrbracket\} \Rightarrow \models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C})^* \{\llbracket \rho \rrbracket\}$ . Moreover, by i.h. for  $\vdash_{HL, \mathcal{D}} \{\rho\} \mathcal{C} \{\rho\}$  we've  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}) \{\llbracket \rho \rrbracket\}$ . Hence  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C})^* \{\llbracket \rho \rrbracket\}$ . Now, by definition  $T_{[\cdot]}(\mathcal{C}^*) = T_{[\cdot]}(\mathcal{C})^*$ . Therefore, we conclude that  $\models_{HL} \{\llbracket \rho \rrbracket\} T_{[\cdot]}(\mathcal{C}^*) \{\llbracket \rho \rrbracket\}$ .

□

In the end of the previous subsection, we showed that HL relative to any effective  $\mathcal{D}$  is semantically incomplete (relative to  $\llbracket \cdot \rrbracket$ ). However, there is another way in which the system can fail to be complete. It could be the case that SynSAsrts is not powerful enough to express the necessary invariants of the loops.

**Definition 2.19.** Let  $\mathcal{L}$  be a FOL and let  $\llbracket \cdot \rrbracket$  be an interpretation for it. Let  $\rho \in \mathcal{L}$  and  $p$  be a set of valuations (in our case, heapless assertion). We say that  $\rho$  expresses  $p$  in  $\llbracket \cdot \rrbracket$  iff  $\llbracket \rho \rrbracket = p$ .

Let  $\llbracket \rho \rrbracket^{-1}$  be the formula (from  $\mathcal{L}$ ) with the lowest Gödel number that expresses  $p$  in  $\llbracket \cdot \rrbracket$  if such formula exists and be undefined otherwise. That is, if  $\llbracket \rho \rrbracket^{-1}$  is defined, then  $\llbracket \rho \rrbracket^{-1}$  expresses  $p$  in  $\llbracket \cdot \rrbracket$ , i.e.  $\llbracket \llbracket \rho \rrbracket^{-1} \rrbracket = p$ . If  $\llbracket \rho \rrbracket^{-1}$  is defined, we say that  $p$  is expressible in  $\llbracket \cdot \rrbracket$ .

**Definition 2.20.** We define strongest postcondition corresponding to  $p$  and  $C$   $\text{sp}(p, C) \Leftarrow \{s' \mid \exists s \in p. \langle s, s' \rangle \in \llbracket C \rrbracket\}$ .

**Lemma 2.21.**  $\models_{HL} \{p\} C \{q\} \Rightarrow \text{sp}(p, C) \subseteq q$

*Proof.* Assume  $\models_{HL} \{p\} C \{q\}$ , i.e.  $\forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C \rrbracket \Rightarrow s' \in q$ . Let  $s'$  be arbitrary such that  $s' \in \text{sp}(p, C)$ . Let  $s$  be witness such that  $s \in p$  and  $\langle s, s' \rangle \in \llbracket C \rrbracket$ . Thus  $s' \in q$ . But  $s'$  was arbitrary, therefore  $\text{sp}(p, C) \subseteq q$ . □

**Lemma 2.22.**  $\models_{HL} \{p\} C \{\text{sp}(p, C)\}$

*Proof.* Assume  $s \in p$  and  $s'$  be such that  $\langle s, s' \rangle \in \llbracket C \rrbracket$ . Then  $s$  is witness for  $\exists s \in p. \langle s, s' \rangle \in \llbracket C \rrbracket$ . That is,  $s' \in \text{sp}(p, C)$ . Therefore  $\models_{HL} \{p\} C \{\text{sp}(p, C)\}$ . □

**Lemma 2.23.** *The following properties hold:*

- (i)  $\text{sp}(p, \text{skip}) = p$ ;
- (ii)  $\text{sp}(p, x := e) = \{s_x^{e(s)} \mid s \in p\}$ ;
- (iii)  $\text{sp}(p, x := \text{nonDet}()) = \{s_x^n \mid s \in p \wedge n \in \text{Val}\}$ ;
- (iv)  $\text{sp}(p, \text{assume } b) = p \cap b$ ;
- (v)  $\text{sp}(p, C_1; C_2) = \text{sp}(\text{sp}(p, C_1), C_2)$ ;
- (vi)  $\text{sp}(p, C_1 + C_2) = \text{sp}(p, C_1) \cup \text{sp}(p, C_2)$ ;
- (vii)  $\text{sp}(p, C^*) = p \cup \text{sp}(\text{sp}(p, C^*), C) = \bigcup_{n \in \mathbb{N}} \text{sp}(p, C^n)$ .

*Proof.* See lemma `sp_properties` in `HeaplessStrongestPostcondition.thy`.  $\square$

**Lemma 2.24.** *For every  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$  and  $\mathcal{C} \in \text{SCom}_{\mathbb{N}}$ , we can (algorithmically) construct  $q \in \text{SynSAsrts}_{\mathbb{N}}$ , which expresses  $\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}))$  in  $[\cdot]$ , i.e.  $\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) = \llbracket q \rrbracket$ .*

*Proof sketch.* Induction on  $\mathcal{C}$  (and arbitrary  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ ):

- $\mathcal{C} \equiv \text{skip}$   
Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . Then  $\text{sp}(\llbracket \rho \rrbracket, \text{skip}) = \llbracket \rho \rrbracket$ .
- $\mathcal{C} \equiv x := e$   
Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . Then  $\text{sp}(\llbracket \rho \rrbracket, x := \llbracket e \rrbracket) = \llbracket \exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x] \rrbracket$ .
- $\mathcal{C} \equiv x := \text{nonDet}()$   
Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . Then  $\text{sp}(\llbracket \rho \rrbracket, x := \text{nonDet}()) = \llbracket \exists x. \rho \rrbracket$ .
- $\mathcal{C} \equiv \text{assume } \mathcal{C}$   
Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . Then  $\text{sp}(\llbracket \rho \rrbracket, \text{assume } \llbracket \mathcal{C} \rrbracket) = \llbracket \rho \wedge \mathcal{C} \rrbracket$ .
- $\mathcal{C} \equiv \mathcal{C}_1; \mathcal{C}_2$   
Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . We've  $\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1; \mathcal{C}_2)) = \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)), T_{[\cdot]}(\mathcal{C}_2))$ .  
By i.h. let  $q \in \text{SynSAsrts}_{\mathbb{N}}$  be such that  $\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) = \llbracket q \rrbracket$ . Again,  
by i.h. let  $r \in \text{SynSAsrts}_{\mathbb{N}}$  be such that  $\text{sp}(\llbracket q \rrbracket, T_{[\cdot]}(\mathcal{C}_2)) = \llbracket r \rrbracket$ . Therefore  
 $\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1; \mathcal{C}_2)) = \llbracket r \rrbracket$ .
- $\mathcal{C} \equiv \mathcal{C}_1 + \mathcal{C}_2$   
Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . We've  $\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) = \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \cup \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_2))$ .  
By i.h. let  $q_1, q_2 \in \text{SynSAsrts}_{\mathbb{N}}$  be such that  $\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_i)) = \llbracket q_i \rrbracket, i \in \{1, 2\}$ . Therefore  $\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) = \llbracket q_1 \vee q_2 \rrbracket$ .

- $\mathcal{C} \equiv \mathcal{C}_0$

Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . We define by recursion  $q_n \in \text{SynSAsrts}_{\mathbb{N}}$ :

$$\left| \begin{array}{l} q_0 \Leftarrow \rho \\ q_{n+1} \Leftarrow q, \text{ where } q \text{ is from the i.h. such that } \text{sp}(\llbracket q_n \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C}_0)) = \llbracket q \rrbracket \end{array} \right.$$

Using  $\forall p. \forall C. \text{sp}(p, C^{n+1}) = \text{sp}(\text{sp}(p, C^n), C)$ <sup>15</sup> and the i.h., it is easily seen that  $\forall n. \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C}_0^n)) = \llbracket q_n \rrbracket$  holds. Moreover, since by i.h. we obtain (algorithmically) constructible  $q$ -s, then  $Q$  s.t.  $Q(n) \Leftarrow q_n$  is recursive. Therefore, by a careful modification of the Representability theorem (see [Shoenfield 1967, Chapter 6.7]), which applies to a slight modification of Robinson arithmetic  $\mathcal{R}$  (where the functional symbol  $1$  is present and the functional symbol  $s$ , corresponding to the function successor, is absent), and since  $\llbracket \cdot \rrbracket$  is the standard interpretation (and hence being a model of this slight modification of Robinson arithmetic), we can construct a formula  $q \in \text{SynSAsrts}_{\mathbb{N}}$ , containing a brand new free variable  $z$ , such that  $\llbracket q[n/z] \rrbracket = \llbracket q_n \rrbracket$  for any  $n \in \mathbb{N}$  ( $n$  is the numeral corresponding to  $n$ ). Now, since  $\text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C}_0^*)) = \bigcup_{n \in \mathbb{N}} \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C}_0^n))$ , it follows that  $\exists z. q \in \text{SynSAsrts}_{\mathbb{N}}$  expresses  $\text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C}_0^*))$  in  $\llbracket \cdot \rrbracket$ , i.e.  $\text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C}_0^*)) = \llbracket \exists z. q \rrbracket$ .

□

For a more general proof, see [Harel 1979, Theorem 3.2].

**Definition 2.25.** We say that  $\text{SynSAsrts}$  is expressive for  $\text{SCom}(\text{AExps}, \text{BExps})$  in  $\llbracket \cdot \rrbracket$  iff for every  $\rho \in \text{SynSAsrts}$  and  $\mathcal{C} \in \text{SCom}(\text{AExps}, \text{BExps})$ ,  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C})) \rrbracket^{-1} \in \text{SynSAsrts}$  is defined.

**Corollary 2.26.**  $\text{SynSAsrts}_{\mathbb{N}}$  is expressive for  $\text{SCom}_{\mathbb{N}}$  in  $\llbracket \cdot \rrbracket$ .

*Proof.* Follows directly from lemma 2.24. □

We remark that  $\text{SynSAsrts}_+$  is not expressive for  $\text{SCom}_+ \Leftarrow \text{SCom}(\text{AExps}_+, \text{BExps}_+)$  in  $\llbracket \cdot \rrbracket_+$ <sup>16</sup>, where  $\text{SynSAsrts}_+$  is  $\text{SynSAsrts}_{\mathbb{N}}$  without multiplication, i.e. the language of the Presburger's arithmetics. Similarly,  $\text{AExps}_+$  and  $\text{BExps}_+$  are  $\text{AExps}_{\mathbb{N}}$  and  $\text{BExps}_{\mathbb{N}}$  without multiplication, respectively.

**Example 2.27.** Consider the following  $(\text{AExps}_+, \text{BExps}_+)$ -program command (note that  $\text{SCom}_+ \subseteq \text{SCom}_{\mathbb{N}}$ )

$$\mathcal{C} \Leftarrow r := 0; \text{ while } 0 \leq x \text{ do } r := r + y; x := x - 1 \text{ od}$$

and the precondition

$$\rho \Leftarrow 0 \leq x \wedge 0 \leq y \wedge x_{in} \doteq x \in \text{SynSAsrts}_+.$$

<sup>15</sup>See lemma `sp_iter` in `HeaplessStrongestPostcondition.thy`.

<sup>16</sup>The standard interpretation.

Then  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C})) \rrbracket^{-1} \in \text{SynSArts}_{\mathbb{N}}$  is defined and equivalent to

$$x \doteq 0 \wedge 0 \leq y \wedge 0 \leq x_{in} \wedge r \doteq x_{in} \cdot y,$$

but  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C})) \rrbracket_+^{-1}$  is not defined. Thus,  $\text{SynSArts}_+$  is not expressive for  $\text{SCom}_+$  in  $\llbracket \cdot \rrbracket_+$ .

Note that [Cook 1978] also requires  $\doteq$  to be in  $\text{BExps}$  and that it receives its standard interpretation. It is required for the deterministic case in the while case, whereas it isn't for the nondeterministic case<sup>17</sup>. In fact, it is sufficient (for the deterministic case) that  $\doteq$  belongs to  $\text{BExps}$  and is interpreted by a predicate that satisfies the congruence axioms for the predicates and functions of  $\text{BExps}$ . However, while this standard interpretation is not strictly necessary for FOL-expressible properties, it becomes essential when considering properties expressible in SOL. If one opts out of true equality, then unexpected behavior, such as non-computation, occurs even for the simplest syntax. For example, consider the standard interpretation of the natural numbers, but with  $\mathbb{N}$  copies of 0 ( $0_0, 0_1, \dots$ ) and two copies of 1 ( $1_0, 1_1$ ), along with the interpretation of  $+$  such that  $0_n + 1_m = 1_k$ , where  $k = 0$  if the  $n$ -th program terminates with its own code as input, and  $k = 1$  otherwise. Of course, the non-computation is with respect to the true equality and not  $\doteq$ .

Let  $\mathcal{F}$  be a deductive system. We say that  $\mathcal{F}$  is complete relative to an interpretation  $\llbracket \cdot \rrbracket$  iff every true in  $\llbracket \cdot \rrbracket$  formula is deducible in  $\mathcal{F}$ . That is, for every  $\rho$  such that  $\llbracket \rho \rrbracket = \text{Stacks}$ , we have  $\vdash_{\mathcal{F}} \rho$ . We will use the meta symbol  $\mathcal{F}$  with potential indices to denote a complete deductive system.

Before proving completeness (relative to  $\llbracket \cdot \rrbracket$ ), we must first examine the following alternative to the Hoare assignment axiom (see figure 2.2).

**Lemma 2.28.** *Let  $\mathcal{D}$  be a deductive, satisfying the logical axioms, system for some interpretation  $\llbracket \cdot \rrbracket$  for  $\text{SynSArts}_{\mathbb{N}}$ . Then the following axiom*

$$\frac{x_{in} \notin \text{fv}(\rho) \cup \text{sc}_{\rho}(x) \cup \text{fv}(e) \cup \{x\}}{\vdash_{HL, \mathcal{D}} \{\rho\}x := e\{\exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x]\}} \text{ (Assign-Floyd)}$$

is equivalent to the Hoare assignment axiom within the context of Hoare logic (relative to  $\mathcal{D}$ ).

*Proof.*

- (Assign)  $\Rightarrow$  (Assign-Floyd)

Assume that for any  $\rho, x, e$  such that  $\text{sc}_{\rho}(x) \cap \text{fv}(e) = \emptyset$ , we have that  $\vdash_{HL, \mathcal{D}} \{\rho[e/x]\}x := e\{\rho\}$ . Let  $\rho, x, e$  and  $x_{in}$  be arbitrary such that  $x_{in} \notin \text{fv}(\rho) \cup \text{sc}_{\rho}(x) \cup \text{fv}(e) \cup \{x\}$ . Then  $\text{sc}_{\exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x]}(x) = \{x_{in}\}$  and  $x_{in} \notin \text{fv}(e)$  and hence  $\text{sc}_{\exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x]}(x) \cap \text{fv}(e) = \emptyset$ . Thus, we obtain

$$\vdash_{HL, \mathcal{D}} \{(\exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x])[e/x]\}x := e\{\exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x]\}$$

<sup>17</sup>Not required when applying corollary 2.26, but used in the encoding within the proof itself.

by the assumption. The precondition coincides (syntactically) with the formula  $\exists x_{in}. \mathcal{P}[x_{in}/x] \wedge e \doteq e[x_{in}/x]$ . Finally, using the consequence rule with  $\vdash_{\mathcal{D}} \mathcal{P} \Rightarrow \exists x_{in}. \mathcal{P}[x_{in}/x] \wedge e \doteq e[x_{in}/x]$ , we conclude that (Assign-Floyd) is a theorem, i.e.  $\vdash_{HL, \mathcal{D}} \{\mathcal{P}\}x := e\{\exists x_{in}. \mathcal{P}[x_{in}/x] \wedge e \doteq e[x_{in}/x]\}$ .

- (Assign-Floyd)  $\Rightarrow$  (Assign)

Assume that for any  $\mathcal{P}, x, e, x_{in}$  such that  $x_{in} \notin \text{fv}(\mathcal{P}) \cup \text{sc}_{\mathcal{P}}(x) \cup \text{fv}(e) \cup \{x\}$ , we have  $\vdash_{HL, \mathcal{D}} \{\mathcal{P}\}x := e\{\exists x_{in}. \mathcal{P}[x_{in}/x] \wedge x \doteq e[x_{in}/x]\}$ . Let  $\mathcal{Q}, x, e$  and  $x_{in}$  be such that  $\text{sc}_{\mathcal{Q}}(x) \cap \text{fv}(e) = \emptyset$  and  $x_{in} \notin \text{fv}(\mathcal{Q}[e/x]) \cup \text{sc}_{\mathcal{Q}[e/x]}(x) \cup \text{fv}(e) \cup \{x\}$ . By the assumption,

$$\vdash_{HL, \mathcal{D}} \{\mathcal{Q}[e/x]\}x := e\{\exists x_{in}. \mathcal{Q}[e/x][x_{in}/x] \wedge x \doteq e[x_{in}/x]\}.$$

The postcondition coincides with  $\exists x_{in}. \mathcal{Q}[e[x_{in}/x]/x] \wedge x \doteq e[x_{in}/x]$  syntactically. Now, we closely examine that it's equivalent with the formula  $\exists x_{in}. \mathcal{Q} \wedge x \doteq e[x_{in}/x]$ . Consider  $\llbracket \exists x_{in}. \mathcal{Q}[e[x_{in}/x]/x] \wedge x \doteq e[x_{in}/x] \rrbracket$ . i.e.,

$$\{s \mid \exists n. s_{x_{in}}^n \in \llbracket \mathcal{Q}[e[x_{in}/x]/x] \rrbracket \wedge s_{x_{in}}^n \in \llbracket x \doteq e[x_{in}/x] \rrbracket\}.$$

Unfolding the second conjunct gives us

$$\{s \mid \exists n. s_{x_{in}}^n \in \llbracket \mathcal{Q}[e[x_{in}/x]/x] \rrbracket \wedge s_{x_{in}}^n(x) = \llbracket e[x_{in}/x] \rrbracket(s_{x_{in}}^n)\}.$$

To unfold the first conjunct to  $s_{x_{in}, x}^{n, \llbracket e[x_{in}/x] \rrbracket(s_{x_{in}}^n)} \in \llbracket \mathcal{Q} \rrbracket$  we require that  $\text{sc}_{\mathcal{Q}}(x) \cap \text{fv}(e[x_{in}/x]) = \emptyset$ <sup>18</sup>, which we obtain using  $\text{sc}_{\mathcal{Q}}(x) \cap \text{fv}(e) = \emptyset$ ,  $x_{in} \notin \text{sc}_{\mathcal{Q}[e/x]}(x)$  and considering the two cases  $x \in \text{fv}(e)$  and  $x \notin \text{fv}(e)$ . Now, using the unfolded second conjunct, we have that

$$s_{x_{in}, x}^{n, \llbracket e[x_{in}/x] \rrbracket(s_{x_{in}}^n)} = s_{x_{in}, x}^{n, s_{x_{in}}^n(x)} = s_{x_{in}}^n.$$

Finally, we obtain  $\{s \mid \exists n. s_{x_{in}}^n \in \llbracket \mathcal{Q} \rrbracket \wedge s_{x_{in}}^n \in \llbracket x \doteq e[x_{in}/x] \rrbracket\}$  by reverting the unfolding of the second conjunct, equivalent to  $\llbracket \exists x_{in}. \mathcal{Q} \wedge x \doteq e[x_{in}/x] \rrbracket$ . Since the equivalence holds for every interpretation  $\llbracket \cdot \rrbracket$ , by Gödel's completeness theorem, we obtain that

$$\vdash_{\mathcal{D}} (\exists x_{in}. \mathcal{Q}[e[x_{in}/x]/x] \wedge x \doteq e[x_{in}/x]) \Leftrightarrow (\exists x_{in}. \mathcal{Q} \wedge x \doteq e[x_{in}/x]).$$

Moreover,  $x_{in} \notin \text{fv}(\mathcal{Q})$  since  $x_{in} \notin \text{fv}(\mathcal{Q}[e/x]) \cup \{x\}$ , and hence

$$\vdash_{\mathcal{D}} (\exists x_{in}. \mathcal{Q} \wedge x \doteq e[x_{in}/x]) \Leftrightarrow (\mathcal{Q} \wedge \exists x_{in}. x \doteq e[x_{in}/x]).$$

Therefore  $\vdash_{\mathcal{D}} (\exists x_{in}. \mathcal{Q}[e[x_{in}/x]/x] \wedge x \doteq e[x_{in}/x]) \Rightarrow \mathcal{Q}$  and by the consequence rule, we conclude  $\vdash_{HL, \mathcal{D}} \{\mathcal{Q}[e/x]\}x := e\{\mathcal{Q}\}$ . □

---

<sup>18</sup>Using that if  $\text{sc}_{\mathcal{Q}}(x) \cap \text{fv}(e) = \emptyset$ , then  $s \in \llbracket \mathcal{Q}[e/x] \rrbracket \Leftrightarrow s_x^{\llbracket e \rrbracket(s)} \in \llbracket \mathcal{Q} \rrbracket$ . Particular case, where  $\llbracket \cdot \rrbracket$  is the standard interpretation, proven in Isabelle (see lemma Hoare\_assign\_axiom\_semantically in HeaplessSyntax.thy).

**Theorem 2.29.** *Let  $\mathcal{T}$  be a complete (non-effective) proof system for  $\text{SynSAsrts}_{\mathbb{N}}$  (relative to  $\llbracket \cdot \rrbracket$ ). Then  $\vdash_{HL, \mathcal{T}} \{\rho\} \mathcal{C} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\mathcal{C})) \rrbracket^{-1} \}$  for any  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$  and  $\mathcal{C} \in \text{SCom}_{\mathbb{N}}$ .*

*Proof.* Induction on  $\mathcal{C}$  (and arbitrary  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ ):

(i)  $\mathcal{C} \equiv \text{skip}$

Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . We've  $\vdash_{HL, \mathcal{T}} \{\rho\} \text{skip} \{\rho\}$  as an axiom. Moreover,  $\text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\text{skip})) = \text{sp}(\llbracket \rho \rrbracket, \text{skip}) = \llbracket \rho \rrbracket$  by lemma 2.23. We know that  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\text{skip})) \rrbracket^{-1}$  is defined, since  $\rho$  expresses it in  $\llbracket \cdot \rrbracket$ . Then  $\llbracket \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\text{skip})) \rrbracket^{-1} \rrbracket = \llbracket \rho \rrbracket$  and hence  $\vdash_{\mathcal{T}} \rho \Leftrightarrow \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\text{skip})) \rrbracket^{-1}$  by completeness of  $\mathcal{T}$  relative to  $\llbracket \cdot \rrbracket$ <sup>19</sup>. Now by consequence rule and  $\vdash_{HL, \mathcal{T}} \{\rho\} \text{skip} \{\rho\}$ , we conclude  $\vdash_{HL, \mathcal{T}} \{\rho\} \text{skip} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\text{skip})) \rrbracket^{-1} \}$ . Thus, for arbitrary  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ .

(ii)  $\mathcal{C} \equiv x := e$

Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . We've

$$\vdash_{HL, \mathcal{T}} \{\rho\} x := e \{ \exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x] \},$$

$x_{in} \notin \text{fv}(\rho) \cup \text{sc}_{\rho}(x) \cup \text{fv}(e) \cup \{x\}$  by lemma 2.28. Moreover, by lemma 2.23, we've  $\text{sp}(\llbracket \rho \rrbracket, x := e) = \{s_x^{\llbracket e \rrbracket(s)} \mid s \in \llbracket \rho \rrbracket\}$  and hence

$$\text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(x := e)) = \{s_x^{\llbracket e \rrbracket(s)} \mid s \in \llbracket \rho \rrbracket\}.$$

Now, since  $\llbracket \exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x] \rrbracket = \{s_x^{\llbracket e \rrbracket(s)} \mid s \in \llbracket \rho \rrbracket\}$ <sup>20</sup>, it follows that  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(x := e)) \rrbracket^{-1}$  is defined and by completeness of  $\mathcal{T}$  relative to  $\llbracket \cdot \rrbracket$ , we obtain that

$$\vdash_{\mathcal{T}} (\exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x]) \Leftrightarrow \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(x := e)) \rrbracket^{-1}.$$

Now, by consequence rule and

$$\vdash_{HL, \mathcal{T}} \{\rho\} x := e \{ \exists x_{in}. \rho[x_{in}/x] \wedge x \doteq e[x_{in}/x] \},$$

we conclude that  $\vdash_{HL, \mathcal{T}} \{\rho\} x := e \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(x := e)) \rrbracket^{-1} \}$ . Thus, for arbitrary  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ .

(iii)  $\mathcal{C} \equiv x := \text{nonDet}()$

Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . We've  $\vdash_{HL, \mathcal{T}} \{\rho\} x := \text{nonDet}() \{ \exists x. \rho \}$  as an axiom. Moreover, we've  $\text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(x := \text{nonDet}())) = \text{sp}(\llbracket \rho \rrbracket, x := \text{nonDet}()) = \{s_x^n \mid s \in \llbracket \rho \rrbracket \wedge n \in \mathbb{N}\}$  by lemma 2.23. Now, since

$$\llbracket \exists x. \rho \rrbracket = \{s_x^n \mid s \in \llbracket \rho \rrbracket \wedge n \in \mathbb{N}\},$$

<sup>19</sup>Note that we cannot apply Gödel's completeness theorem here, since  $\llbracket \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{\llbracket \cdot \rrbracket}(\text{skip})) \rrbracket^{-1} \rrbracket' = \llbracket \rho \rrbracket'$  doesn't hold for all  $\rho$  and interpretations  $\llbracket \cdot \rrbracket'$ .

<sup>20</sup>See lemma Floyd\_assign\_axiom\_semantically in HeaplessSyntax.thy.

it follows that  $\llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(x := \text{nonDet}())) \rrbracket^{-1}$  is defined and by completeness of  $\mathcal{F}$  relative to  $\llbracket \cdot \rrbracket$ , we obtain

$$\vdash_{\mathcal{F}} (\exists x. \mathcal{P}) \Leftrightarrow \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(x := \text{nonDet}())) \rrbracket^{-1}.$$

Now, by consequence rule and  $\vdash_{HL, \mathcal{F}} \{\mathcal{P}\}x := \text{nonDet}() \{ \exists x. \mathcal{P} \}$ , we conclude that the desired

$$\vdash_{HL, \mathcal{F}} \{\mathcal{P}\}x := \text{nonDet}() \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(x := \text{nonDet}())) \rrbracket^{-1} \}$$

holds. Thus, for arbitrary  $\mathcal{P} \in \text{SynSAsrts}_{\mathbb{N}}$ .

(iv)  $\mathcal{C} \equiv \text{assume } \mathcal{C}$

Let  $\mathcal{P} \in \text{SynSAsrts}_{\mathbb{N}}$ . We've  $\vdash_{HL, \mathcal{F}} \{\mathcal{P}\} \text{assume } \mathcal{C} \{ \mathcal{P} \wedge \mathcal{C} \}$  as an axiom. Moreover, we've

$$\text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\text{assume } \mathcal{C})) = \text{sp}(\llbracket \mathcal{P} \rrbracket, \text{assume } \llbracket \mathcal{C} \rrbracket) = \llbracket \mathcal{P} \rrbracket \cap \llbracket \mathcal{C} \rrbracket$$

by lemma 2.23. Hence, we've  $\text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\text{assume } \mathcal{C})) = \llbracket \mathcal{P} \wedge \mathcal{C} \rrbracket$ . Thus  $\llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\text{assume } \mathcal{C})) \rrbracket^{-1}$  is defined and by completeness of  $\mathcal{F}$  relative to  $\llbracket \cdot \rrbracket$ , we obtain that  $\vdash_{\mathcal{F}} \mathcal{P} \wedge \mathcal{C} \Leftrightarrow \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\text{assume } \mathcal{C})) \rrbracket^{-1}$ . Now, by consequence rule and  $\vdash_{HL, \mathcal{F}} \{\mathcal{P}\} \text{assume } \mathcal{C} \{ \mathcal{P} \wedge \mathcal{C} \}$ , we conclude that the desired  $\vdash_{HL, \mathcal{F}} \{\mathcal{P}\} \text{assume } \mathcal{C} \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\text{assume } \mathcal{C})) \rrbracket^{-1} \}$ . Thus, for arbitrary  $\mathcal{P} \in \text{SynSAsrts}_{\mathbb{N}}$ .

(v)  $\mathcal{C} \equiv \mathcal{C}_1; \mathcal{C}_2$

Assume for  $\mathcal{C}_1$  and  $\mathcal{C}_2$ . Let  $\mathcal{P} \in \text{SynSAsrts}_{\mathbb{N}}$ . By i.h. we've

$$\vdash_{HL, \mathcal{F}} \{\mathcal{P}\} \mathcal{C}_1 \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \}.$$

In particular,  $\llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \in \text{SynSAsrts}_{\mathbb{N}}$  is defined. Therefore, we can apply the i.h. for it and  $\mathcal{C}_2$ . That is, we've

$$\vdash_{HL, \mathcal{F}} \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \} \mathcal{C}_2 \{ \llbracket \text{sp}(\llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1}, T_{[\cdot]}(\mathcal{C}_2)) \rrbracket^{-1} \},$$

i.e.

$$\vdash_{HL, \mathcal{F}} \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \} \mathcal{C}_2 \{ \llbracket \text{sp}(\text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)), T_{[\cdot]}(\mathcal{C}_2)) \rrbracket^{-1} \}.$$

Now, by lemma 2.23, we've

$$\text{sp}(\text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)), T_{[\cdot]}(\mathcal{C}_2)) = \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1); T_{[\cdot]}(\mathcal{C}_2)) = \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1; \mathcal{C}_2)).$$

Therefore, we have

$$\vdash_{HL, \mathcal{F}} \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \} \mathcal{C}_2 \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1; \mathcal{C}_2)) \rrbracket^{-1} \}.$$

Now, by sequence rule and that we've  $\vdash_{HL, \mathcal{F}} \{\mathcal{P}\} \mathcal{C}_1 \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \}$ , we conclude that  $\vdash_{HL, \mathcal{F}} \{\mathcal{P}\} \mathcal{C}_2 \{ \llbracket \text{sp}(\llbracket \mathcal{P} \rrbracket, T_{[\cdot]}(\mathcal{C}_1; \mathcal{C}_2)) \rrbracket^{-1} \}$ . Thus, for arbitrary  $\mathcal{P} \in \text{SynSAsrts}_{\mathbb{N}}$ .

(vi)  $\mathcal{C} \equiv \mathcal{C}_1 + \mathcal{C}_2$

Assume for  $\mathcal{C}_1$  and  $\mathcal{C}_2$ . Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . By i.h. we've

$$\vdash_{HL, \mathcal{F}} \{\rho\} \mathcal{C}_1 \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \}$$

and

$$\vdash_{HL, \mathcal{F}} \{\rho\} \mathcal{C}_2 \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_2)) \rrbracket^{-1} \}.$$

By lemma 2.23, we've that

$$\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) \cup \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) = \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1) + T_{[\cdot]}(\mathcal{C}_2)) = \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)).$$

Now, since  $\text{SynSAsrts}_{\mathbb{N}}$  is expressive for  $\text{SCom}_{\mathbb{N}}$  in  $\llbracket \cdot \rrbracket$  (see lemma 2.25) it follows that  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) \rrbracket^{-1}$  is defined<sup>21</sup>, whereas  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1}$  and  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_2)) \rrbracket^{-1}$  are defined by the i.h. Now, by the completeness of  $\mathcal{F}$  relative to  $\llbracket \cdot \rrbracket$ ,  $\llbracket \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \rrbracket \subseteq \llbracket \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) \rrbracket^{-1} \rrbracket$  and  $\llbracket \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_2)) \rrbracket^{-1} \rrbracket \subseteq \llbracket \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) \rrbracket^{-1} \rrbracket$ , we obtain that the following  $\vdash_{\mathcal{F}} \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \Rightarrow \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) \rrbracket^{-1}$  and  $\vdash_{\mathcal{F}} \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_2)) \rrbracket^{-1} \Rightarrow \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) \rrbracket^{-1}$  hold. Applying the consequence rule with the above,  $\vdash_{HL, \mathcal{F}} \{\rho\} \mathcal{C}_1 \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \}$  and  $\vdash_{HL, \mathcal{F}} \{\rho\} \mathcal{C}_2 \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_2)) \rrbracket^{-1} \}$ , we obtain that

$$\vdash_{HL, \mathcal{F}} \{\rho\} \mathcal{C}_1 \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) \rrbracket^{-1} \}$$

and

$$\vdash_{HL, \mathcal{F}} \{\rho\} \mathcal{C}_2 \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) \rrbracket^{-1} \}.$$

Finally, applying the choice rule, we conclude that

$$\vdash_{HL, \mathcal{F}} \{\rho\} \mathcal{C}_1 + \mathcal{C}_2 \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1 + \mathcal{C}_2)) \rrbracket^{-1} \}.$$

Thus, for arbitrary  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ .

(vii)  $\mathcal{C} \equiv \mathcal{C}_0^*$

Assume for  $\mathcal{C}_0$ . Let  $\rho \in \text{SynSAsrts}_{\mathbb{N}}$ . Since  $\text{SynSAsrts}_{\mathbb{N}}$  is expressive for  $\text{SCom}_{\mathbb{N}}$  in  $\llbracket \cdot \rrbracket$  (see lemma 2.25) it follows that  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \in \text{SynSAsrts}_{\mathbb{N}}$  is defined, thus we can apply i.h. with  $\mathcal{C}_0$  and obtain

$$\vdash_{HL, \mathcal{F}} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \} \mathcal{C}_0 \{ \llbracket \text{sp}(\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1}, T_{[\cdot]}(\mathcal{C}_0)) \rrbracket^{-1} \}.$$

That is,

$$\vdash_{HL, \mathcal{F}} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \} \mathcal{C}_0 \{ \llbracket \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)), T_{[\cdot]}(\mathcal{C}_0)) \rrbracket^{-1} \}.$$

Moreover,

$$\begin{aligned} \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)), T_{[\cdot]}(\mathcal{C}_0)) &= \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)), T_{[\cdot]}(\mathcal{C}_0)) \\ &\subseteq \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \\ &= \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \end{aligned}$$

<sup>21</sup>We don't necessary need to use the expressiveness:  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_1)) \rrbracket^{-1} \vee \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_2)) \rrbracket^{-1}$ .



by lemma 2.23 and hence

$$\llbracket \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)), T_{[\cdot]}(\mathcal{C}_0)) \rrbracket^{-1} \subseteq \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1},$$

where  $\llbracket \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)), T_{[\cdot]}(\mathcal{C}_0)) \rrbracket^{-1}$  is defined by the i.h. Now, by completeness of  $\mathcal{T}$  relative to  $\llbracket \cdot \rrbracket$ , we obtain

$$\vdash_{\mathcal{T}} \llbracket \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)), T_{[\cdot]}(\mathcal{C}_0)) \rrbracket^{-1} \Rightarrow \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1}.$$

Applying the consequence rule with

$$\vdash_{HL, \mathcal{T}} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \} \mathcal{C}_0 \{ \llbracket \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)), T_{[\cdot]}(\mathcal{C}_0)) \rrbracket^{-1} \}$$

and the above, we obtain

$$\vdash_{HL, \mathcal{T}} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \} \mathcal{C}_0 \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \}.$$

Now, by the iteration rule, we obtain

$$\vdash_{HL, \mathcal{T}} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \} \mathcal{C}_0^* \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \}.$$

Analogously to  $\vdash_{\mathcal{T}} \llbracket \text{sp}(\text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)), T_{[\cdot]}(\mathcal{C}_0)) \rrbracket^{-1} \Rightarrow \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1}$ , we obtain  $\vdash_{\mathcal{T}} \rho \Rightarrow \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1}$ . Applying the consequence rule with this and the above, we conclude that

$$\vdash_{HL, \mathcal{T}} \{ \rho \} \mathcal{C}_0^* \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C}_0^*)) \rrbracket^{-1} \}.$$

Thus, for arbitrary  $\rho \in \text{SynSArts}_{\mathbb{N}}$ .

□

**Corollary 2.30.** *Let  $\mathcal{T}$  be a complete (non-effective) proof system for  $\text{SynSArts}_{\mathbb{N}}$  (relative to  $\llbracket \cdot \rrbracket$ ). Then  $\models_{HL} \{ \llbracket \rho \rrbracket \} T_{[\cdot]}(\mathcal{C}) \{ \llbracket \rho \rrbracket \} \Rightarrow \vdash_{HL, \mathcal{T}} \{ \rho \} \mathcal{C} \{ \rho \}$ .*

*Proof.* Assume  $\models_{HL} \{ \llbracket \rho \rrbracket \} T_{[\cdot]}(\mathcal{C}) \{ \llbracket \rho \rrbracket \}$ . By theorem 2.29 we've

$$\vdash_{HL, \mathcal{T}} \{ \rho \} \mathcal{C} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) \rrbracket^{-1} \}.$$

Moreover, by the assumption, lemma 2.21 and the definedness of  $\llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) \rrbracket^{-1}$ , we obtain that  $\llbracket \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) \rrbracket^{-1} \rrbracket = \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) \subseteq \llbracket \rho \rrbracket$  and hence by the completeness of  $\mathcal{T}$  relative to  $\llbracket \cdot \rrbracket$ , we obtain  $\vdash_{\mathcal{T}} \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) \rrbracket^{-1} \Rightarrow \rho$  and hence by consequence rule and  $\vdash_{HL, \mathcal{T}} \{ \rho \} \mathcal{C} \{ \llbracket \text{sp}(\llbracket \rho \rrbracket, T_{[\cdot]}(\mathcal{C})) \rrbracket^{-1} \}$ , we conclude that  $\vdash_{HL, \mathcal{T}} \{ \rho \} \mathcal{C} \{ \rho \}$ . □

Relative completeness is often established using the weakest liberal precondition (the dual notion of the strongest postcondition) due to its simplicity. However, we opted for the strongest postcondition approach, though without a specific justification. The overall reasoning remains unchanged, but a new axiom is required for the nondeterministic assignment:

$$\frac{}{\vdash_{HL, \mathcal{D}} \{ \forall x. \rho \} x := \text{nonDet}() \{ \rho \}} \text{ (Havoc-backward) }.$$

Moreover, we use the Hoare assignment axiom (rather than the Floyd assignment axiom) for the assignment case.

## 2.2 Separation logic

### 2.2.1 Semantics

Hoare logic, despite being complete, has a significant limitation in practical applications: it lacks the ability to reason about heap-sensitive programs. We begin this subsection by defining the state model used to reason about heap programs, which sets the stage for introducing Separation logic—a specialized framework for local reasoning about heap programs.

**Definition 2.31.** A *heap*  $h$  is a finite partial function from  $\mathbb{N}$  to  $\text{Val} \cup \{\perp\}$ , i.e.  $h : \mathbb{N} \xrightarrow{\text{fin}} \text{Val} \cup \{\perp\}$ , where  $\perp \notin \text{Val}$ .

We denote the set of all heaps by **Heaps**. We will use the meta symbol  $h$  with potential indices to denote a heap. Both  $h(5) = \perp$  and  $5 \notin \text{Dom}(h)$  indicate that address 5 is not allocated. For Separation logic either of these approaches suffices. However, in order to avoid introducing separate state models—one for Separation logic, and one for Separation Sufficient Incorrectness logic—we adopt this state model for Separation logic as well. Location 0 serves as the NULL pointer, i.e.  $0 \notin \text{Dom}(h)$  for any  $h$ <sup>22</sup>.

**Definition 2.32.** A *program state*  $\sigma = \langle s, h \rangle$  is an ordered pair, consisting of a heapless program state  $s$  and a heap  $h$ , i.e.  $\sigma \in \text{Stacks} \times \text{Heaps}$ .

We denote the set of all program states by **States**. We will use the meta symbol  $\sigma$  with potential indices to denote a program state.

**Definition 2.33.** An *assertion*  $p$  is a set of program states.

We denote the set of all assertions by **Asrts**. We will use the meta symbols  $p, q, r, f$  with potential indices to denote assertion.

Note that we don't introduce program expressions and predicates that depend on the heap. Such stack-only dependent program expressions and predicates are called pure. Moreover, recall that the definitions of heapless assertion and program predicate coincided, whereas now the definitions differ in that assertions can reason about heaps, while program predicates cannot.

**Definition 2.34.** We define *program commands* using BNF:

$$C \equiv \text{skip} \mid x := e \mid x := \text{nonDet}() \mid \text{assume } b \mid x := \text{alloc}() \\ \mid [x] := e \mid y := [x] \mid \text{free}(x) \mid (C; C) \mid (C + C) \mid C^*,$$

where  $x, y \in \text{PVars}$ .

We will use the meta symbol  $C$  with potential indices to denote a program command.

The  $x := \text{alloc}()$  command allocates memory (with arbitrary initialization) and stores the address of the allocated memory in  $x$ . The  $[x] := e$  command

<sup>22</sup>We could have defined  $h : \mathbb{N}^+ \xrightarrow{\text{fin}} \text{Val} \cup \{\perp\}$ , but we chose not to for ease of use in Isabelle.

$$\begin{array}{c}
\hline
\langle x := e, \langle s, h \rangle \rangle \rightarrow \langle \mathbf{skip}, \langle s_x^{e(s)}, h \rangle \rangle \\
\hline
\frac{l \neq 0 \wedge (l \notin \text{Dom}(h) \vee h(l) = \perp)}{\langle x := \text{alloc}(), \langle s, h \rangle \rangle \rightarrow \langle \mathbf{skip}, \langle s_x^l, h_l^n \rangle \rangle} \\
\hline
\frac{s(x) \neq 0 \wedge s(x) \in \text{Dom}(h) \wedge h(s(x)) \neq \perp}{\langle y := [x], \langle s, h \rangle \rangle \rightarrow \langle \mathbf{skip}, \langle s_y^{h(s(x))}, h \rangle \rangle} \\
\hline
\end{array}
\qquad
\begin{array}{c}
\hline
\langle x := \text{nonDet}(), \langle s, h \rangle \rangle \rightarrow \langle \mathbf{skip}, \langle s_x^n, h \rangle \rangle \\
\hline
\frac{s(x) \neq 0 \wedge s(x) \in \text{Dom}(h) \wedge h(s(x)) \neq \perp}{\langle [x] := e, \langle s, h \rangle \rangle \rightarrow \langle \mathbf{skip}, \langle s, h_{s(x)}^{e(s)} \rangle \rangle} \\
\hline
\frac{s(x) \neq 0 \wedge s(x) \in \text{Dom}(h) \wedge h(s(x)) \neq \perp}{\langle \text{free}(x), \langle s, h \rangle \rangle \rightarrow \langle \mathbf{skip}, \langle s, h_{s(x)}^\perp \rangle \rangle} \\
\hline
\end{array}$$

$$\begin{array}{ccc}
\frac{}{\langle \mathbf{skip}; C_2, \sigma \rangle \rightarrow \langle C_2, \sigma \rangle} & \frac{\langle C_1, \sigma \rangle \rightarrow \langle C_1', \sigma' \rangle}{\langle C_1; C_2, \sigma \rangle \rightarrow \langle C_1'; C_2, \sigma' \rangle} & \frac{s \in b}{\langle \mathbf{assume} \ b, \langle s, h \rangle \rangle \rightarrow \langle \mathbf{skip}, \langle s, h \rangle \rangle} \\
\hline
\langle C_1 + C_2, \sigma \rangle \rightarrow \langle C_2, \sigma \rangle & \langle C_1 + C_2, \sigma \rangle \rightarrow \langle C_1, \sigma \rangle & \langle C^*, \sigma \rangle \rightarrow \langle (C; C^*) + \mathbf{skip}, \sigma \rangle
\end{array}$$

Figure 2.3: Small-step semantics of program commands.

writes the value computed by  $e$  to the memory at the address  $x$ . The  $y := [x]$  commands reads the value from the memory at address  $x$  and stores it in  $y$ . The  $\text{free}(x)$  commands frees the memory located at address  $x$ .

**Definition 2.35.** *The set of all modified variables by  $C$ , denoted  $\text{md}(C)$ , is defined as follows*

$$\begin{array}{lll}
\text{md}(\mathbf{skip}) \Leftarrow \emptyset & \text{md}(x := e) \Leftarrow \{x\} & \text{md}(x := \text{nonDet}()) \Leftarrow \{x\} \\
\text{md}(\mathbf{assume} \ b) \Leftarrow \emptyset & \text{md}(x := \text{alloc}()) \Leftarrow \{x\} & \text{md}([x] := e) = \emptyset \\
\text{md}(y := [x]) \Leftarrow \{y\} & \text{md}(\text{free}(x)) \Leftarrow \emptyset & \text{md}(C_1; C_2) = \text{md}(C_1) \cup \text{md}(C_2) \\
\text{md}(C_1 + C_2) = \text{md}(C_1) \cup \text{md}(C_2) & \text{md}(C^*) \Leftarrow \text{md}(C) &
\end{array}$$

**Definition 2.36.** *A program configuration  $\langle C, \sigma \rangle$  is an ordered pair, where  $C$  is a program command and  $\sigma$  is a program state.*

We will use the meta symbol  $c$  with potential indices to denote a program configuration.

**Definition 2.37.** *The small-step semantics, denoted  $\rightarrow$ , of the program commands is a relation over program configurations and is defined by recursion in figure 2.3.*

Similarly to the heapless program commands, we define  $\rightarrow^*$  as the reflexive and transitive closure of  $\rightarrow$ , where  $\rightarrow$  now refers to the small-step semantics of the program commands. Moreover, we adopt the same terminology for execution, termination, and related concepts as in the heapless case. The definition of  $\llbracket C \rrbracket$ ,  $\llbracket C \rrbracket \Leftarrow \{ \langle \sigma, \sigma' \rangle \mid \langle C, \sigma \rangle \rightarrow^* \langle \mathbf{skip}, \sigma' \rangle \}$ , remains the same, with the type updated.

$$\begin{array}{c}
\frac{s(x) = 0 \vee s(x) \notin \text{Dom}(h) \vee h(s(x)) = \perp}{\text{aborts } \langle [x] := e, \langle s, h \rangle \rangle} \quad \frac{s(x) = 0 \vee s(x) \notin \text{Dom}(h) \vee h(s(x)) = \perp}{\text{aborts } \langle y := [x], \langle s, h \rangle \rangle} \\
\frac{s(x) = 0 \vee s(x) \notin \text{Dom}(h) \vee h(s(x)) = \perp}{\text{aborts } \langle \text{free}(x), \langle s, h \rangle \rangle} \quad \frac{\text{aborts } \langle C_1, \sigma \rangle}{\text{aborts } \langle C_1; C_2, \sigma \rangle}
\end{array}$$

Figure 2.4: Operational erroneous semantics of program commands that immediately abort.

**Lemma 2.38.** *The following properties hold:*

- (i)  $\llbracket \text{skip} \rrbracket = \{ \langle \sigma, \sigma \rangle \mid \sigma \in \text{States} \};$
- (ii)  $\llbracket [x] := e \rrbracket = \{ \langle \langle s, h \rangle, \langle s_x^{e(s)}, h \rangle \rangle \mid \langle s, h \rangle \in \text{States} \};$
- (iii)  $\llbracket [x] := \text{nonDet}() \rrbracket = \{ \langle \langle s, h \rangle, \langle s_x^n, h \rangle \rangle \mid \langle s, h \rangle \in \text{States} \wedge n \in \text{Val} \};$
- (iv)  $\llbracket \text{assume } b \rrbracket = \{ \langle \langle s, h \rangle, \langle s, h \rangle \rangle \mid s \in b \wedge h \in \text{Heaps} \};$
- (v)  $\llbracket [x] := \text{alloc}() \rrbracket = \{ \langle \langle s, h \rangle, \langle s_x^l, h_l^n \rangle \rangle \mid (l \notin \text{Dom}(h) \vee h(l) = \perp) \wedge l \neq 0 \};$
- (vi)  $\llbracket [[x] := e] \rrbracket = \{ \langle \langle s, h \rangle, \langle s, h_{s(x)}^{e(s)} \rangle \rangle \mid \exists n. h(s(x)) = n \wedge n \neq \perp \wedge s(x) \neq 0 \};$
- (vii)  $\llbracket [y := [x]] \rrbracket = \{ \langle \langle s, h \rangle, \langle s_y^n, h \rangle \rangle \mid \exists n. h(s(x)) = n \wedge n \neq \perp \wedge s(x) \neq 0 \};$
- (viii)  $\llbracket [\text{free}(x)] \rrbracket = \{ \langle \langle s, h \rangle, \langle s, h_{s(x)}^\perp \rangle \rangle \mid \exists n. h(s(x)) = n \wedge n \neq \perp \wedge s(x) \neq 0 \};$
- (ix)  $\llbracket [C_1; C_2] \rrbracket = \llbracket [C_1] \rrbracket \circ \llbracket [C_2] \rrbracket$ , where  $\circ$  is composition of relations;
- (x)  $\llbracket [C_1 + C_2] \rrbracket = \llbracket [C_1] \rrbracket \cup \llbracket [C_2] \rrbracket$ ;
- (xi)  $\llbracket [C^*] \rrbracket = \bigcup_{n \in \mathbb{N}} \llbracket [C^n] \rrbracket$ , where  $C^0 \Leftarrow \text{skip}$  and  $C^{n+1} \Leftarrow C; C^n$ .

*Proof.* See theorem `dsem_properties` in `ProgramCommands.thy`.  $\square$

**Definition 2.39.** *The operational erroneous semantics `aborts` of program commands that immediately abort is an unary predicate over program configurations and is defined by recursion in figure 2.4.*

**Definition 2.40.** *An assertion  $p$  provides the memory required by  $C$  iff*

$$\forall \sigma \in p. \forall c'. \langle C, \sigma \rangle \rightarrow^* c' \Rightarrow \neg \text{aborts}(c')$$

Now, we are ready to define formally Hoare triples and their validity.

**Definition 2.41.** *A Hoare triple  $\{p\}C\{q\}$  is an ordered triple (with a special syntax  $\{\cdot\} \cdot \{\cdot\}$ ), where  $p$  and  $q$  are assertions and  $C$  is a program command.*

**Definition 2.42.** *A Hoare triple  $\{p\}C\{q\}$  is valid, written  $\models_{SL} \{p\}C\{q\}$ , iff*

1.  $p$  provides the memory required by  $C$ ; and
2.  $\forall \sigma \in p. \forall \sigma'. \langle \sigma, \sigma' \rangle \in \llbracket [C] \rrbracket \Rightarrow \sigma' \in q$ .

As in the heapless setting, providing intuitive examples is not feasible without introducing a clean syntax. Therefore, to aid understanding, we present a representative fragment of the syntax, along with the interpretation of interest. First, the syntax for "dereferencing" a pointer is  $x \mapsto 1$ , which is interpreted as  $\llbracket x \mapsto 1 \rrbracket = \{\langle s, h \rangle \mid h(s(x)) = 1\}$ . Second, the syntax for indicating an empty heap is  $\text{emp}$ , which is interpreted as  $\llbracket \text{emp} \rrbracket = \{\langle s, h \rangle \mid h = \emptyset\}$ . Now, it is easy to see that

$$\begin{aligned} \models_{SL} \{\llbracket \text{emp} \rrbracket\} x := \text{alloc}() \{\llbracket \exists n. x \mapsto n \rrbracket\} & \quad \models_{SL} \{\llbracket x \mapsto 1 \rrbracket\} y := [x] \{\llbracket y \doteq 1 \rrbracket\} \\ \not\models_{SL} \{\llbracket \text{emp} \rrbracket\} [x] := \llbracket 1 \rrbracket \{\llbracket x \mapsto 1 \rrbracket\} & \quad \models_{SL} \{\llbracket x \mapsto 1 \rrbracket\} \text{free}(x) \{\llbracket x \mapsto \perp \rrbracket\}, \end{aligned}$$

where the third one is not valid, because  $\llbracket \text{emp} \rrbracket$  doesn't provide the memory required by  $[x] := \llbracket 1 \rrbracket$ .

The first conjunct of the definition of Hoare triple validity serves two roles:

1. Any derivable Hoare formula ensures no NULL or "dangling" pointers are dereferenced; and
2. Ensures the soundness of the Frame rule (see below).

### 2.2.2 Syntax

We will use the same syntax for both program expressions and program predicates:  $\text{AExps}_{\mathbb{N}}$  and  $\text{BExps}_{\mathbb{N}}$ , respectively. Additionally, we will adopt the same interpretation, namely the standard model. The difference lies in the syntax for our assertions.

**Definition 2.43.** *We define  $\mathbb{N}$ -assertions using BNF:*

$$\begin{aligned} \rho \Rightarrow \top \mid \perp \mid (e \leq e) \mid (e \doteq e) \mid \text{emp} \mid x \mapsto e \mid x \mapsto \perp \mid (\rho * \rho) \\ \mid (\rho \wedge \rho) \mid (\rho \vee \rho) \mid (\rho \Rightarrow \rho) \mid (\rho \Leftrightarrow \rho) \mid \neg \rho \mid \exists x. \rho \mid \forall x. \rho, \end{aligned}$$

where  $x \in \text{PVars}$ ,  $e \in \text{AExps}_{\mathbb{N}}$ .

We denote the set of all  $\mathbb{N}$ -assertions by  $\text{SynAsrt}_{\mathbb{N}}$ . The predicates  $\text{emp}$  and  $x \mapsto \perp$  both signify that the heap is empty. Specifically,  $\text{emp}$  indicates that the entire heap is empty, while  $x \mapsto \perp$  means that the memory at location  $x$  is empty. The latter also provides the additional information that we "own" the memory at address  $x$ . The predicate  $x \mapsto e$  indicates that we "own" the memory at location  $x$  and that it is allocated, containing the value  $e$ . Lastly, the functional symbol  $*$  represents the so-called separating conjunction, which we will discuss in more detail in subsection 2.2.4. It is worth noting that  $\perp$  is used both as false and as an empty address. However, this is not a problem, since the latter case can be considered as part of the bigger unary predicate symbol " $\mapsto \perp$ ." Moreover,  $\perp$  is also used semantically in  $h(5) = \perp$ . Lastly, when we write  $x \mapsto e$ , it is important to note that  $e$  cannot be evaluated to  $\perp$  (see definition 2.11 and recall its standard interpretation).

The interpretation of interest  $\llbracket \cdot \rrbracket$  is as follows:

$$\begin{aligned}
\llbracket \top \rrbracket &\Leftarrow \text{States} \\
\llbracket \perp \rrbracket &\Leftarrow \emptyset \\
\llbracket e_1 < e_2 \rrbracket &\Leftarrow \{ \langle s, h \rangle \in \text{States} \mid \llbracket e_1 \rrbracket(s) < \llbracket e_2 \rrbracket(s) \} \\
\llbracket e_1 \doteq e_2 \rrbracket &\Leftarrow \{ \langle s, h \rangle \in \text{States} \mid \llbracket e_1 \rrbracket(s) = \llbracket e_2 \rrbracket(s) \} \\
\llbracket \text{emp} \rrbracket &\Leftarrow \{ \langle s, h \rangle \in \text{States} \mid h = \emptyset \} \\
\llbracket x \mapsto e \rrbracket &\Leftarrow \{ \langle s, h \rangle \in \text{States} \mid h(s(x)) = \llbracket e \rrbracket(s) \wedge s(x) \neq 0 \} \\
\llbracket x \mapsto \perp \rrbracket &\Leftarrow \{ \langle s, h \rangle \in \text{States} \mid h(s(x)) = \perp \wedge s(x) \neq 0 \} \\
\llbracket \mathcal{P} * \mathcal{Q} \rrbracket &\Leftarrow \{ \langle s, h_1 \cup h_2 \rangle \mid \langle s, h_1 \rangle \in \llbracket \mathcal{P} \rrbracket \wedge \langle s, h_2 \rangle \in \llbracket \mathcal{Q} \rrbracket \wedge h_1 \perp h_2 \} \\
\llbracket \exists x. \mathcal{P} \rrbracket &\Leftarrow \{ \langle s, h \rangle \in \text{States} \mid \exists n. \langle s_x^n, h \rangle \in \llbracket \mathcal{P} \rrbracket \} \\
\llbracket \forall x. \mathcal{P} \rrbracket &\Leftarrow \{ \langle s, h \rangle \in \text{States} \mid \forall n. \langle s_x^n, h \rangle \in \llbracket \mathcal{P} \rrbracket \},
\end{aligned}$$

where  $h_1 \perp h_2 \Leftarrow \text{Dom}(h_1) \cap \text{Dom}(h_2) = \emptyset$ <sup>23</sup>. The remaining logical symbols receive their standard (in the sense of Tarski) interpretation. Now that we've outlined the interpretation, it's helpful to revisit the distinction between  $\text{emp}$  and  $x \mapsto \perp$ . In simple terms,  $\text{emp}$  indicates that we have no specific information about the heap locally, in particular for the address  $x$ . On the other hand,  $x \mapsto \perp$  means that we are explicitly aware that  $x$  contains no memory. This difference explains why we can add more heap to  $x$  when  $\text{emp}$  holds—since we have no prior knowledge of its contents—whereas we cannot do so with  $x \mapsto \perp$ , as we assert that  $x$ 's memory is accounted for (and deallocated in this case). That is,  $\llbracket \text{emp} \rrbracket * \llbracket x \mapsto \perp \rrbracket = \llbracket x \mapsto \perp \rrbracket$  and  $\llbracket x \mapsto \perp \rrbracket * \llbracket x \mapsto \perp \rrbracket = \emptyset$ . Finally, we define (AExps, BExps)-program commands, denoted with  $\mathcal{C}$  with potential indices, in the obvious way.

### 2.2.3 Syntactic and semantic logic

We observed with Hoare logic that any Hoare-style logic,  $\vdash_{HL, \mathcal{D}}$ , is destined to be (effectively) incomplete<sup>24</sup>. Furthermore, it's evident that no syntax can express each assertion with a corresponding formula, as there are countably many formulae but uncountably many assertions. We argued that this isn't a problem since what we need from the syntax is just the ability to express loop invariants. Therefore, for simplicity, it's common to divide the work into two parts: first, defining the axioms and rules semantically, and then adding a syntax that can express the loop invariants. In the remainder of this thesis, we focus on the former, leaving the latter as an open question for future exploration. It's important to note that the (syntactic) axioms and rules rely solely on logical symbols—symbols with fixed interpretations—and syntactic substitution, which behaves consistently across all interpretations. This allows their semantic soundness to trivially ensure their syntactic soundness. By semantic axioms and rules, we mean precisely what is presented in theorem 2.17, which trivially ensure soundness of their syntactic counterparts (see corollary 2.18).

<sup>23</sup>Yet another usage of  $\perp$ .

<sup>24</sup>Since no complete (effective) system  $\mathcal{D}$  exists for the standard model of PA.

$$\begin{array}{c}
\frac{}{\models_{SL} \{p\}x := \text{alloc}() \{ \langle s_x^l, h_l^n \rangle \mid \langle s, h \rangle \in p \wedge (l \notin \text{Dom}(h) \vee h(s(x)) = \perp) \wedge l \neq 0 \}} \text{(Alloc)} \\
\frac{p \text{ provides the memory required by } C}{\models_{SL} \{p\}[x] := e \{ \langle s, h_{s(x)}^{e(s)} \rangle \mid \langle s, h \rangle \in p \wedge h(s(x)) = n \wedge n \neq \perp \wedge s(x) \neq 0 \}} \text{(Write)} \\
\frac{p \text{ provides the memory required by } C}{\models_{SL} \{p\}y := [x] \{ \langle s_y^n, h \rangle \mid \langle s, h \rangle \in p \wedge h(s(x)) = n \wedge n \neq \perp \wedge s(x) \neq 0 \}} \text{(Read)} \\
\frac{p \text{ provides the memory required by } C}{\models_{SL} \{p\}\text{free}(x) \{ \langle s, h_{s(x)}^\perp \rangle \mid \langle s, h \rangle \in p \wedge h(s(x)) = n \wedge n \neq \perp \wedge s(x) \neq 0 \}} \text{(Free)}
\end{array}$$

Figure 2.5: Semantic heap axioms of Separation logic.

$$\begin{array}{c}
\frac{}{\vdash_{SL, \emptyset} \{\top\}x := \text{alloc}() \{x \mapsto -\}} \text{(Alloc)} \qquad \frac{}{\vdash_{SL, \emptyset} \{x \mapsto -\}[x] := e \{x \mapsto e\}} \text{(Write)} \\
\frac{}{\vdash_{SL, \emptyset} \{x \mapsto n\}y := [x] \{x \mapsto n \wedge y \doteq n\}} \text{(Read)} \qquad \frac{}{\vdash_{SL, \emptyset} \{x \mapsto n\}\text{free}(x) \{x \mapsto \perp\}} \text{(Free)}
\end{array}$$

Figure 2.6: Syntactic heap axioms of Separation logic, where  $x \mapsto -$  is short for  $\exists n. x \mapsto n$ .

To illustrate the differences between semantic and syntactic axioms and rules, let us consider the four semantic axioms presented in figure 2.5. These axioms can be easily verified with the help of lemma 2.38. Our aim in designing semantic axioms is to ensure that the postcondition precisely corresponds to the strongest postcondition. This ensures that when formulating syntactic axioms, our only task is to demonstrate that the strongest postcondition can be expressed syntactically for each given syntactically expressible precondition and corresponding program command. It is easily verified that these axioms satisfy the desired property, i.e.  $\text{sp}(p, x := \text{alloc}()) = \{ \langle s_x^l, h_l^n \rangle \mid \langle s, h \rangle \in p \wedge (l \notin \text{Dom}(h) \vee h(s(x)) = \perp) \wedge l \neq 0 \}$ , etc. Furthermore, it is evident that the last three axioms cannot be designed for arbitrary preconditions since they need to provide the memory required by the corresponding program command. Now, using these semantic axiom, we can trivially obtain the soundness of the syntactic axioms presented in figure 2.6. The question of whether these syntactic axiom are sufficient for the completeness of Separation logic (in the sense of theorem 2.29) is, as mentioned above, beyond the scope of this thesis. That is, it is possible that there is  $\rho$  such that  $\text{sp}(\llbracket \rho \rrbracket, \mathcal{C})$  is not expressible in  $\llbracket \cdot \rrbracket$ , where  $\mathcal{C}$  is  $x := \text{alloc}()$ ,  $[x] := e$ ,  $y := [x]$  or  $\text{free}(x)$ <sup>25</sup>. For those interested, [Tat-

<sup>25</sup>An example of such  $\rho$  and  $\mathcal{C}$  was identified shortly before the finalization of this thesis:  $\text{emp}$  and  $x := \text{alloc}()$ , respectively. The solution is to remove  $\text{emp}$  from the syntax, as it adds no practical value. It, however, is retained in the thesis to better illustrate the difference between  $l \notin \text{Dom}(h)$  and  $h(l) = \perp$ .

[suta et al. 2019] demonstrates the (syntactic) completeness of Separation Logic in the context of weakest liberal preconditions, utilizing the so-called "magic wand." Moreover, [Bannister et al. 2018] shows a broader approach—covering both backward (via weakest liberal precondition) and forward (via strongest postcondition) perspectives. In our thesis, we focus on the forward direction; thus, one can adopt the approach outlined in the latter work, which introduces the so-called "septraction." It's also worth noting that neither of the two account for memory which we "own" and know to be empty, represented by  $h(l) = \perp$ , indicating that further modifications may be necessary, presumably with the allocation axiom.

#### 2.2.4 Separating conjunction and the frame rule

We now turn to the separating conjunction and frame rule, which lie at the heart of Separation logic and constitutes the primary focus of this thesis. We begin by noting that Hoare logic can reason locally:  $\vdash_{HL} \{x \doteq 0\} x := x+1 \{x \doteq 1\}$ . That is, it can focus only on the resources required/used by the program command. Moreover, it can combine multiple local specifications into a more general one using the constancy and consequence rules, where

$$\frac{\vdash_{HL, \mathcal{D}} \{\mathcal{P}\} \mathcal{C} \{ \mathcal{Q} \} \quad \text{md}(\mathcal{C}) \cap \text{fv}(\mathcal{L}) = \emptyset}{\vdash_{HL, \mathcal{D}} \{\mathcal{P} \wedge \mathcal{L}\} \mathcal{C} \{ \mathcal{Q} \wedge \mathcal{L} \}} \text{ (Constancy)}$$

and  $\text{md}(\mathcal{C})$  is the set of variables modified by  $\mathcal{C}$  (analogous to definition 2.35). Note that if the condition  $\text{md}(\mathcal{C}) \cap \text{fv}(\mathcal{L}) = \emptyset$  is not satisfied, then the constancy rule is unsound (not valid) since  $\vdash_{HL, \mathcal{D}} \{x \doteq 0\} x := 1 \{x \doteq 1\}$ , whereas  $\not\vdash_{HL, \mathcal{D}} \{x \doteq 0 \wedge x \doteq 0\} x := 1 \{x \doteq 1 \wedge x \doteq 0\}$ .

**Lemma 2.44.** *Let  $\forall s \in f. \forall x \in \text{md}(C). \forall n. s_x^n \in f$ . Then  $\text{sp}(p \cap f, C) = \text{sp}(p, C) \cap f$ .*

*Proof.* See lemma `sp_frame` in `HeaplessStrongestPostcondition.thy`.  $\square$

One can easily see that the second assumption of the constancy rule, namely  $\text{md}(\mathcal{C}) \cap \text{fv}(\mathcal{L}) = \emptyset$ , implies  $\forall s \in \llbracket \mathcal{L} \rrbracket. \forall x \in \text{md}(\mathcal{C}). \forall n. s_x^n \in \llbracket \mathcal{L} \rrbracket$  using the fact that  $\forall s \in \llbracket \mathcal{L} \rrbracket. \forall x \notin \text{fv}(\mathcal{L}). \forall n. s_x^n \in \llbracket \mathcal{L} \rrbracket$ .

**Corollary 2.45.** *Let  $\models_{HL} \{p\} C \{q\}$  and  $\forall s \in f. \forall x \in \text{md}(C). \forall n. s_x^n \in f$ . Then  $\models_{HL} \{p \cap f\} C \{q \cap f\}$ .*

*Proof.* We've that  $\text{sp}(p, C) \subseteq q$  by assumption 1 and lemma 2.21. Therefore  $\text{sp}(p, C) \cap f \subseteq q \cap f$ . Thus, by assumption 2 and lemma 2.44 it follows that  $\text{sp}(p \cap f, C) \subseteq q \cap f$ . Moreover, by lemma 2.22 we've  $\models_{HL} \{p \cap f\} C \{\text{sp}(p \cap f, C)\}$ . Therefore by theorem 2.17 we conclude that  $\models_{HL} \{p \cap f\} C \{q \cap f\}$ .  $\square$

**Corollary 2.46.** *Rule of constancy is valid.*

*Proof.* Let  $\llbracket \cdot \rrbracket$  be an interpretation such that  $\models_{HL} \{\llbracket \mathcal{P} \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C}) \{\llbracket \mathcal{Q} \rrbracket\}$  and let  $\text{md}(\mathcal{C}) \cap \text{fv}(\mathcal{L}) = \emptyset$ , i.e.  $\text{md}(T_{\llbracket \cdot \rrbracket}(\mathcal{C})) \cap \text{fv}(\mathcal{L}) = \emptyset$ . From assumption 2, we obtain that  $\forall s \in \llbracket \mathcal{L} \rrbracket. \forall x \in \text{md}(T_{\llbracket \cdot \rrbracket}(\mathcal{C})). \forall n. s_x^n \in \llbracket \mathcal{L} \rrbracket$ . Then from assumption 1 and corollary 2.45, it follows that  $\models_{HL} \{\llbracket \mathcal{P} \rrbracket \cap \llbracket \mathcal{L} \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C}) \{\llbracket \mathcal{Q} \rrbracket \cap \llbracket \mathcal{L} \rrbracket\}$ . That is,  $\models_{HL} \{\llbracket \mathcal{P} \wedge \mathcal{L} \rrbracket\} T_{\llbracket \cdot \rrbracket}(\mathcal{C}) \{\llbracket \mathcal{Q} \wedge \mathcal{L} \rrbracket\}$ .  $\square$



We showed that the rule of constancy is valid in Hoare logic. One might expect it to hold in Separation logic as well; however, it is, in fact, unsound:

$$\vdash_{HL, \mathcal{D}} \{x \mapsto 0\}[x] := 1\{x \mapsto 1\},$$

whereas

$$\not\vdash_{HL, \mathcal{D}} \{x \mapsto 0 \wedge y \mapsto 0\}[x] := 1\{x \mapsto 1 \wedge y \mapsto 0\},$$

since  $x$  and  $y$  may be equivalent. In order to resolve this problem, it is necessary to guarantee that  $x \neq y$ . To address this, we turn to the seminal solution presented in [Reynolds 2002]. Rather than using conjunction, which does not ensure that the conjuncts have disjoint memory, they propose the so-called separating conjunction  $*$ , defined as follows:

$$p * q \Leftarrow \{\langle s, h_1 \cup h_2 \rangle \mid \langle s, h_1 \rangle \in p \wedge \langle s, h_2 \rangle \in q \wedge h_1 \perp h_2\}.$$
<sup>26</sup>

This definition guarantees that states satisfying  $x \mapsto 0 * y \mapsto 0$  have  $x \neq y$  and hence we obtain that  $\vdash_{HL, \mathcal{D}} \{x \mapsto 0 * y \mapsto 0\}[x] := 1\{x \mapsto 1 * y \mapsto 0\}$  is valid.

**Remark 2.47.** Recall that a formula  $\{\mathcal{P}\}\mathcal{C}\{Q\}$  is valid iff it is true for any interpretive model. In Hoare logic, we assume that the symbols  $\doteq, \wedge, \vee, \Rightarrow, \Leftrightarrow, \neg, \exists$  and  $\forall$  have fixed interpretations. These symbols are considered logical, while  $\top, \perp$  and  $<$  are non-logical. Given this, more precisely, a formula  $\{\mathcal{P}\}\mathcal{C}\{Q\}$  is valid iff it is true for any interpretive model, induced by an interpretation  $\llbracket \cdot \rrbracket$ , where the logical symbols are assigned their fixed meanings. In Separation logic, we treat  $\text{emp}, \mapsto, \mapsto \perp$  and  $*$  as logical symbols, in addition to the logical symbols from Hoare logic, with all their interpretations fixed accordingly.

**Lemma 2.48.** *Let  $p$  provide the memory required by  $C$ . Then  $p * f$  provides the memory required by  $C$ .*

*Proof.* See lemma provides\_mono in Star.thy. □

The definition  $\text{sp}(p, C) \Leftarrow \{s' \mid \exists \sigma \in p. \langle \sigma, \sigma' \rangle \in \llbracket C \rrbracket\}$ , remains the same, with the type updated.

**Lemma 2.49.**  $\models_{SL} \{p\}C\{q\} \Rightarrow \text{sp}(p, C) \subseteq q$

*Proof.* Assume  $\models_{SL} \{p\}C\{q\}$ . Let  $\sigma'$  be arbitrary such that  $\sigma' \in \text{sp}(p, C)$ . Let  $\sigma$  be witness such that  $\sigma \in p$  and  $\langle \sigma, \sigma' \rangle \in \llbracket C \rrbracket$ . Thus, by the assumption,  $\sigma' \in q$ . But  $\sigma'$  was arbitrary, therefore  $\text{sp}(p, C) \subseteq q$ . □

**Lemma 2.50.** *Let  $p$  provide the memory required by  $C$ . Then  $\models_{SL} \{p\}C\{\text{sp}(p, C)\}$ .*

*Proof.* Assume  $s \in p$  and  $s'$  be such that  $\langle s, s' \rangle \in \llbracket C \rrbracket$ . Then  $s$  is witness for  $\exists s \in p. \langle s, s' \rangle \in \llbracket C \rrbracket$ . That is,  $s' \in \text{sp}(p, C)$ . Moreover, recall that  $p$  provides the memory required by  $C$ . Therefore  $\models_{SL} \{p\}C\{\text{sp}(p, C)\}$ . □

<sup>26</sup>Note that we use the same symbol,  $*$ , for both the syntactic and semantic instances.

**Theorem 2.51.** *Let  $\forall \langle s, h \rangle \in f. \forall x \in \text{md}(C). \forall n. \langle s_x^n, h \rangle \in f$ ,  $p$  provide the memory required by  $C$  and  $\forall \langle s, h \rangle \in f. h(l) = \perp \Rightarrow h_l^- \in f$ , where  $h_{l_0}^- \Leftarrow \{ \langle l, n \rangle \in h \mid l \neq l_0 \}$ . Then  $\text{sp}(p * f, C) = \text{sp}(p, C) * f$ .*

*Proof.* See lemma `sp_frame` in `Star.thy`.  $\square$

Note that if we didn't have a way to indicate that we own empty memory,  $\perp$ , this lemma would not hold for program commands that deallocate memory:

$$\emptyset = \text{sp}(\llbracket x \mapsto I \rrbracket * \llbracket x \mapsto I \rrbracket, \text{free}(x)) \subset \text{sp}(\llbracket x \mapsto I \rrbracket, \text{free}(x)) * \llbracket x \mapsto I \rrbracket = \llbracket x \mapsto I \rrbracket.$$

In our case, however, the equality hold, since

$$\text{sp}(\llbracket x \mapsto I \rrbracket, \text{free}(x)) * \llbracket x \mapsto I \rrbracket = \llbracket x \mapsto \perp \rrbracket * \llbracket x \mapsto I \rrbracket = \emptyset.$$

The third assumption of the lemma,  $\forall \langle s, h \rangle \in f. h(l) = \perp \Rightarrow h_l^- \in f$ , when  $f = \llbracket \ell \rrbracket$ , essentially says that " $\ell$  does not speak of  $\perp$ ", i.e.  $\perp$  does not appear (syntactically) in  $\ell$ .

**Corollary 2.52.** *Let  $\models_{SL} \{p\}C\{q\}$ ,  $\forall \langle s, h \rangle \in f. \forall x \in \text{md}(C). \forall n. \langle s_x^n, h \rangle \in f$  and  $\forall \langle s, h \rangle \in f. h(l) = \perp \Rightarrow h_l^- \in f$ . Then  $\models_{SL} \{p * f\}C\{q * f\}$ .*

*Proof.* We've that  $p$  provides the memory required by  $C$  by assumption 1 and the definition of validity. Thus, by lemma 2.48 it follows that  $p * f$  provides the memory required by  $C$  and hence by lemma 2.50 we obtain  $\models_{SL} \{p * f\}C\{\text{sp}(p * f, C)\}$ . Moreover, using that  $p * f$  provides the memory required by  $C$ , assumptions 2 and 3 and theorem 2.51, we obtain that  $\text{sp}(p * f, C) = \text{sp}(p, C) * f$ . Therefore  $\models_{SL} \{p * f\}C\{\text{sp}(p, C) * f\}$ . Now, by lemma 2.49 and assumption 1, we obtain that  $\text{sp}(p, C) \subseteq q$  and hence by the monotonicity of  $*$ , which is easily verified, we obtain  $\text{sp}(p, C) * f \subseteq q * f$ . Finally, using the fact that we can weaken the postcondition, which is easily provable, and  $\models_{SL} \{p * f\}C\{\text{sp}(p, C) * f\}$ , we conclude that  $\models_{SL} \{p * f\}C\{q * f\}$ .  $\square$

Now, the soundness of the syntactic frame rule

$$\frac{\vdash_{SL, \emptyset} \{\mathcal{P}\}\mathcal{C}\{q\} \quad \text{fv}(\mathcal{P}) \cap \text{md}(\mathcal{C}) = \emptyset \quad \text{no } \perp \text{ in } \mathcal{P}}{\vdash_{SL, \emptyset} \{\mathcal{P} * \mathcal{P}'\}\mathcal{C}\{q * \mathcal{P}'\}} \text{ (Frame)}$$

can easily be obtained by corollary 2.52 and since we consider  $*$  as a logical symbol with fixed interpretation ( $\llbracket \mathcal{P} * \mathcal{Q} \rrbracket = \llbracket \mathcal{P} \rrbracket * \llbracket \mathcal{Q} \rrbracket$ ). In this thesis, we focus on the semantic frame rule due to its generality and simplicity and since the syntactic one following directly from it.

Note that if we didn't require the precondition to provide the memory required by the program command, the frame rule would be unsound:

$$\frac{\models_{SL} \{\llbracket \text{emp} \rrbracket\}[x] := \llbracket I \rrbracket \{\llbracket x \mapsto I \rrbracket\}}{\not\models_{SL} \{\llbracket \text{emp} \rrbracket * \llbracket x \mapsto I \rrbracket\}[x] := \llbracket I \rrbracket \{\llbracket x \mapsto I \rrbracket\} * \llbracket x \mapsto I \rrbracket\}}.$$

Moreover, this requirement is strictly stronger than the requirement to have a sound frame rule. That is, for  $\models_{SL'} \{p\}C\{q\}$  defined to hold whenever

1.  $\models_{SL'}$  supports a sound semantic frame rule; and
2.  $\forall \sigma \in p. \forall \sigma'. \langle \sigma, \sigma' \rangle \in \llbracket C \rrbracket \Rightarrow \sigma' \in q$ ; and
3.  $\models_{SL'}$  is the biggest such predicate,

it follows that  $\models_{SL} \{p\}C\{q\} \Rightarrow \models_{SL'} \{p\}C\{q\}$  and not  $\Leftrightarrow$ . The difference comes in the read command:

$$\models_{SL'} \{\llbracket \top \rrbracket\}x := [y]\{\llbracket \top \rrbracket\},$$

whereas

$$\not\models_{SL} \{\llbracket \top \rrbracket\}x := [y]\llbracket \top \rrbracket.$$

The goal of SL is not having as general as possible Frame rule, but rather to be useful in practice, hence the choice of definition. That is, proven (via the sound logic) specifications, guarantee that the program doesn't dereference NULL or "dangling" pointers.

### 2.3 Separation Sufficient Incorrectness logic

Hoare logic and its separation variant, Separation logic, are considered overapproximating logics because the postconditions of valid triples are overapproximations (supersets) of the set of reachable states. Formally, let  $\mathbf{post}(p, C) \Leftarrow \llbracket C \rrbracket[p]$  and  $\mathbf{pre}(q, C) \Leftarrow \llbracket C \rrbracket^{-1}[q]$ , where  $R[X]$  is the image of  $X$  under the relation  $R$ , i.e.  $R[X] \Leftarrow \{y \mid \exists x \in X. \langle x, y \rangle \in R\}$  and  $R^{-1}$  is the inverse relation of  $R$ , i.e.  $R^{-1} \Leftarrow \{\langle y, x \rangle \mid \langle x, y \rangle \in R\}$ . Then we have

$$\models_{HL} \{p\}C\{q\} \Leftrightarrow \mathbf{post}(p, C) \subseteq q$$

and

$$\models_{SL} \{p\}C\{q\} \Leftrightarrow \mathbf{post}(p, C) \subseteq q, \text{ for } p \text{ providing the memory required by } C.$$

In a similar manner, Incorrectness logic (abbreviated as IL; see [O'Hearn 2019]) and its separation variant, Incorrectness Separation logic (abbreviated as ISL; see [Raad et al. 2020]), defined as

$$\models_{I(S)L} \{p\}C\{q\} \stackrel{def}{\Leftrightarrow} \forall \sigma' \in q. \exists \sigma \in p. \langle \sigma, \sigma' \rangle \in \llbracket C \rrbracket,$$

are underapproximate, since the postcondition,  $q$ , of a valid triple is an underapproximation (subset) of the set of reachable states, i.e.

$$\models_{I(S)L} \{p\}C\{q\} \Leftrightarrow q \subseteq \mathbf{post}(p, C).$$

Note that  $\mathbf{sp}$  coincide with  $\mathbf{post}$ . Additionally, the consequence rule in Incorrectness (Separation) logic is "reversed." Consequently, the weakest postcondition, denoted  $\mathbf{wpo}$ , is introduced. It is straightforward to recognize that  $\mathbf{wpo}$  coincides with  $\mathbf{post}$ . As a further observation, the validity of all 4 logics introduced so

far can be expressed through the `post`, but not via `pre`. In this thesis, we focus on backward underapproximation, rather than (forward) underapproximation, where the precondition is a subset of the backward reachable states. Specifically, we consider Sufficient Incorrectness logic (abbreviated SIL; see [Ascari et al. 2024]) and especially its separation variant, Separation Sufficient Incorrectness logic (abbreviated SSIL; see [Ascari et al. 2024]), defined as

$$\models_{(S)SIL} \{p\}C\{q\} \stackrel{def}{\iff} \forall \sigma \in p. \exists \sigma' \in q. \langle \sigma, \sigma' \rangle \in \llbracket C \rrbracket.$$

Indeed, it is evident that every precondition,  $p$ , of a valid triple for these two logics is an underapproximation of the set of backward reachable states, i.e.

$$\models_{(S)SIL} \{p\}C\{q\} \iff p \subseteq \text{pre}(q, C).$$

While we do not delve into the details of (S)SIL, we present the following representative example:

$$\vdash_{(S)SIL, \emptyset} \{x \doteq 0\}x := \text{nonDet}()\{x \doteq 17\},$$

which says that, starting in a state satisfying  $x \doteq 0$ , there is an execution path, which terminates successfully in a state satisfying  $x \doteq 17$ .

We will now focus on one of the key aspects of SSIL that is central to this thesis: its frame rule. Immediately, it is clear that their semantic frame rule is unsound, as seen in the following example

$$\frac{\models_{SSIL} \{\llbracket \text{emp} \rrbracket\}x := \text{alloc}()\{\llbracket x \doteq 1 \rrbracket\}}{\not\models_{SSIL} \{\llbracket \text{emp} \rrbracket * \llbracket 1 \mapsto 1 \rrbracket\}x := \text{alloc}()\{\llbracket x \doteq 1 \rrbracket * \llbracket 1 \mapsto 1 \rrbracket\}}.$$

This issue is common among underapproximate logics, where in order to maintain a sound frame rule, one cannot guarantee that the allocated address is part of a fixed finite set. The essence of the problem lies in the fact that we cannot determine the allocated address locally, as this requires global knowledge of which addresses are available. Without this global context, we cannot guarantee the exact address that will be assigned during allocation, leading to potential inconsistencies when applying the frame rule. If, however, we know locally that an address is free, i.e.  $1 \mapsto \perp$ , then we have a guarantee that location 1 can be returned by `alloc()`

$$\frac{\models_{SSIL} \{\llbracket 1 \mapsto \perp \rrbracket\}x := \text{alloc}()\{\llbracket x \doteq 1 \rrbracket\}}{\models_{SSIL} \{\llbracket 1 \mapsto \perp \rrbracket * \llbracket 1 \mapsto 1 \rrbracket\}x := \text{alloc}()\{\llbracket x \doteq 1 \rrbracket * \llbracket 1 \mapsto 1 \rrbracket\}}.$$

Regardless, the semantic frame rule is unsound for SSIL. They bypass this problem by proving a syntactic frame rule sound, where the syntactic alloc axiom they use is

$$\vdash_{SSIL, \emptyset} \{\text{emp}\}x := \text{alloc}()\{x \mapsto n\}.$$

That is, they cannot obtain the problematic assumption of the frame rule, namely

$$\not\vdash_{SSIL, \emptyset} \{\text{emp}\}x := \text{alloc}()\{x \doteq n\}.$$

Using this approach, however, they encounter another issue: the alloc axiom is not complete in the sense that  $\vdash_{SSIL, \emptyset} \{\text{emp}\}x := \text{alloc}()\{x \doteq n\}$  is valid, but is not a theorem. Their proposed solution is less than ideal, as it alters the semantics of the allocation command to only produce addresses that are locally known to be free. This approach is problematic because it effectively makes the allocation command local, whereas, in reality, it is inherently (and uniquely) non-local.

A notable observation, though not directly relevant to this thesis, is that loop invariants are not complete for underapproximation. That is, the rule

$$\frac{\models_{HL} \{p\}C\{p\}}{\models_{HL} \{p\}C^*\{p\}} \text{ (Iter)}$$

won't suffice for the completeness of either I(S)L or (S)SIL. Indeed, it is clear that

$$\models_{I(S)L} \{\llbracket x \doteq 0 \rrbracket\}(x := \llbracket x + 1 \rrbracket)^* \{\llbracket x \geq 0 \rrbracket\},$$

where  $\geq$  is interpreted in the obvious way. However, there is no invariant  $p$  such that

$$\models_{I(S)L} \{p\}x := \llbracket x + 1 \rrbracket\{p\}.$$

The "best" we can do is  $p \Leftrightarrow \llbracket x \geq 0 \rrbracket$ , which still fails, since we cannot reach any state  $s$  such that  $s(x) = 0$ . If we consider the integers rather than the naturals, then we can find an invariant:

$$\models_{I(S)L} \{\llbracket \top \rrbracket_{\mathbb{Z}}\}x := \llbracket x + 1 \rrbracket_{\mathbb{Z}}\{\llbracket \top \rrbracket_{\mathbb{Z}}\}.$$

But this doesn't help either, as I(S)L's reversed consequence rule allows only precondition weakening, hence we cannot obtain the precondition  $\llbracket x \doteq 0 \rrbracket_{\mathbb{Z}}$  from  $\llbracket \top \rrbracket_{\mathbb{Z}}$ . Similarly, it is clear that

$$\models_{(S)SIL} \{\llbracket 0 < x \rrbracket\}(\mathbf{assume} \llbracket 0 < x \rrbracket; x := x - 1)^* \{\llbracket x \doteq 0 \rrbracket\}.$$

The "best" we can do for an invariant of  $\mathbf{assume} \llbracket 0 < x \rrbracket; x := x - 1$  is either  $\llbracket 0 < x \rrbracket$  or  $\llbracket x \geq 0 \rrbracket$ . The former fails because for a state  $s$  with  $s(x) = 1$ , the resulting state  $s'$  with  $s'(x) = 0$  does not satisfy  $0 < x$ . The latter fails because for a state  $s$  with  $s(x) = 0$ , no resulting state exists. To obtain completeness for the underapproximate logics, most sources (e.g. [Ascari et al. 2024]) require a rule that involves infinite union (or, in terms of the syntactic rules, infinite disjunction)

$$\frac{\text{for all } n \in \mathbb{N}. \models_{(S)SIL} \{q_{n+1}\}C\{q_n\}}{\models_{(S)SIL} \left\{ \bigcup_{n \in \mathbb{N}} q_n \right\} C^* \{q_0\}} \text{ (iter)}$$

We, however, are led to believe that, using

$$\frac{}{\models_{(S)SIL} \{q\}C^*\{q\}} \text{ (iter0)} \quad \frac{\models_{(S)SIL} \{p\}C^*; C\{q\}}{\models_{(S)SIL} \{p\}C^*\{q\}} \text{ (iterN)}$$

$$\frac{\vDash_{(S)SIL} \{p\}C\{q\}}{\vDash_{(S)SIL} \{\{\langle s_x^n, h \rangle \mid \langle s, h \rangle \in p\}\}C\{\{\langle s_x^n, h \rangle \mid \langle s, h \rangle \in q\}\}} \text{ (Exists) },$$

we obtain completeness as well, but now the syntactic variant of (Exists) requires only an existential quantifier instead of an infinite disjunction. Using (iter0) and (iterN), we obtain

$$\frac{\text{for all } n \in \mathbb{N}. \vDash_{(S)SIL} \{q_{n+1}\}C\{q_n\}}{\vDash_{(S)SIL} \{q_n\}C^*\{q_0\}} .$$

Now, if it is sufficient for the completeness to apply this with only recursive  $p_n$  and  $q_n$ , then we can integrate the idea from the proof of lemma 2.24 and finally apply the (Exists) rule. This, however, lies beyond the scope of this thesis.

## 2.4 Outcome Separation logic

All the logics discussed so far express either over- or underapproximate reasoning. We will now explore Outcome Logic (abbreviated OL; see [Zilberstein et al. 2023]) and, more relevant to this thesis, its separation variant, Outcome Separation Logic (abbreviated OSL; see [Zilberstein et al. 2024]), which support both types of reasoning. While the command semantics are parametric on an *execution model* for OL and on an *outcome algebra* and an allocator function for OSL, we consider the instantiation corresponding to nondeterministic and non-probabilistic programs. That is, the program semantics (for the considered instantiation) of OL essentially coincide with the semantics presented in lemma 2.8, whereas the program semantics (for the considered instantiation) of OSL essentially coincide with the semantics presented in lemma 2.38 with the difference that they are parametric on an allocator function<sup>27</sup>  $\text{alloc} : \text{States} \rightarrow \mathcal{P}(\mathbb{N}^+) \setminus \{\emptyset\}$

$$\llbracket x := \text{alloc}() \rrbracket_{\text{alloc}} = \{\{\langle s, h \rangle, \langle s_x^l, h_l^n \rangle \mid (l \notin \text{Dom}(h) \vee h(l) = \perp) \wedge l \neq 0 \wedge l \in \text{alloc}(\langle s, h \rangle)\}\}.$$

We say that an allocator  $\text{alloc}$  is adequate iff

$$\forall \langle s, h \rangle \in \text{States}. \exists l \in \text{alloc}(\langle s, h \rangle). l \notin \text{Dom}(h) \vee h(l) = \perp.$$

The validity of the OSL triples will quantify over all possible adequate allocators, hence solving the issue discussed with SSIL (and underapproximation in general), i.e. since we quantify over all (adequate) allocators, we cannot have guarantee for the value of the allocated address. That is,

$$\not\vDash_{OSL} \{\llbracket \text{emp} \rrbracket\}x := \text{alloc}()\{\llbracket x \doteq 1 \rrbracket\},$$

which will ensure the soundness of the (semantic) frame rule. Formally, the validity (for the considered instantiation) is defined as

$$\vDash_{OSL} \{P\}C\{Q\} \stackrel{\text{def}}{\iff} \forall S \in P. \forall \text{alloc} \in \mathcal{A}. \llbracket C \rrbracket_{\text{alloc}}[S] \in Q,$$

where  $\mathcal{A}$  is the set of adequate allocators. Note that  $P$  and  $Q$  are sets of assertions, which we will refer to as hyper-assertions (or assertions for short).

<sup>27</sup>The definition is slightly altered, but the essence remains intact. We denote the powerset of  $X$  with  $\mathcal{P}(X)$ .

**Definition 2.53.** A hyper-assertion (or hassertion)  $P$  is a set of assertions.

We denote the set of all hassertions by  $\mathbf{HAsrts}$ . We will use the meta symbols  $P, Q, R, F$  with potential indices to denote hassertion.

Now, we return to the (considered instantiation of the) validity of OSL with greater precision. We noted that their validity quantifies over all (adequate) allocators, which enables a sound semantic frame rule. To speak of the frame rule in the first place, they would require a star operator over hassertions. However, they have avoided defining such a star, missing the significant advantage of their semantics—its ability to support a general sound semantic frame rule. Instead, they have achieved a sound semantic frame rule, but limited to a specific type of hassertions—these that can be expressed by their syntax. The syntax of hassertions includes "power sets (without the empty set) of assertions", union of hassertions (syntactically  $\vee$ ), and join of hassertions (syntactically  $\otimes$ )<sup>28</sup>, where their syntax for assertions is a slight alteration of our  $\mathbb{N}$ -assertions (see definition 2.43). We use  $\cup$  as a semantic equivalent of  $\vee$  (as usual) and the same symbol  $\otimes$  for both the semantic and syntactic join. The definition of join is as follows:

$$P \otimes Q \Leftarrow \{S_P \cup S_Q \mid S_P \in P \wedge S_Q \in Q\}.$$

They define a new separating conjunction  $\circledast$  as a transformations on the syntactic hassertions as follows (illustrated semantically):

- $\mathcal{P}^+(p) \circledast f \Leftarrow \mathcal{P}^+(p * f)$ , where  $\mathcal{P}^+(X) \Leftarrow \mathcal{P}(X) \setminus \{\emptyset\}$ ;
- $(P \cup Q) \circledast f \Leftarrow (P \circledast f) \cup (Q \circledast f)$ ;
- $(P \otimes Q) \circledast f \Leftarrow (P \circledast f) \otimes (Q \circledast f)$ .

Note that the left and right arguments of  $\circledast$  have different types: the left argument is a hassertion, while the right argument is an assertion. For this star then, they prove sound semantic frame rule

$$\frac{\models_{OSL} \{P\}C\{Q\} \quad \text{"md}(C) \cap \text{fv}(f) = \emptyset"}{\models_{OSL} \{P \circledast f\}C\{Q \circledast f\}} \text{ (Frame)},$$

where "md( $C$ )  $\cap$  fv( $f$ ) =  $\emptyset$ " is  $\forall \langle s, h \rangle \in f. \forall x \in \text{md}(C). \forall n. \langle s_x^n, h \rangle \in f$  and  $P$  and  $Q$  are obtained via (finite applications of) unions and joins of non-empty powersets of assertions. It turns out, however, that there's a small mistake<sup>29</sup>: they have to require that the frame does not contain (syntactically)  $\perp$ , otherwise the frame rule becomes unsound:

$$\frac{\models_{OSL} \{\mathcal{P}^+(\llbracket \text{emp} \rrbracket)\}x := \text{alloc}()\{\mathcal{P}^+(\llbracket x \mapsto - \rrbracket)\}}{\not\models_{OSL} \{\mathcal{P}^+(\llbracket \text{emp} \rrbracket) \circledast \llbracket 1 \mapsto \perp \rrbracket\}x := \text{alloc}()\{\mathcal{P}^+(\llbracket x \mapsto - \rrbracket) \circledast \llbracket 1 \mapsto \perp \rrbracket\}},$$

since otherwise the conclusion says that location 1 cannot be returned by the allocation command, even though we know that it is free. This mistake seems to be common among sources, as it occurs also in [Ascari et al. 2024].

<sup>28</sup>In the OSL paper it is denoted with the symbol  $\oplus$  and called the outcome conjunction.

<sup>29</sup>It is unclear to us whether this condition is tacitly implied at some point in their work.

The main goal of this thesis is to design a star over assertions, which supports a sound frame rule. Inspired by the definition of OSL’s star, we aim to create a star that distributes over both union and join. It is worth noting that OSL’s assertion language includes  $\top$ , which is satisfied by anything, including erroneous states. This inclusion is necessary to “drop” *outcomes*, allowing them to reason underapproximatingly. As will be demonstrated later, this feature is also essential for enabling Hyper Separation logic to reason in an underapproximate manner. However, this aspect falls outside the scope of the current thesis, and therefore, we focus on the overapproximate part. Formally,  $\top \otimes f \Leftrightarrow \top$  plays a crucial role in enabling underapproximate reasoning in OSL. While, ideally, our star operator should serve a similar function, it falls beyond the scope of this thesis.

## 2.5 Relational Separation logic

### 2.5.1 Semantics

So far, the logics we’ve considered have focused on reasoning about a single program’s behavior. Relational Hoare logics (abbreviated RHL; see [Benton 2004]) and its separation variant Relational Separation logic (abbreviated RSL; see [Yang 2007]) take a step further by enabling us to reason about the relationship between two programs. Rather than describing properties of a single program command, relational logics allow us to analyze how one program might relate to another. That is, we now consider quadruplets  $\{R\}_{C_2}^{C_1}\{S\}$ , where  $R$  and  $S$  are relations, called (heapless) assertions, and  $C_1$  and  $C_2$  are (heapless) program commands. Such relational logic can be used to demonstrate the equivalence between a compiler-optimized program and its original, non-optimized counterpart.

**Definition 2.54.** A *relational-assertion* (or *rassertion*)  $R$  is a set of pairs of program states, i.e.  $R \subseteq \text{States}^2$ .

We denote the set of all rassertions by  $\text{RASrts}$ . We will use the meta symbols  $R, S, T, F$  with potential indices to denote rassertion. We will use interchangeably  $\left[ \begin{smallmatrix} \langle s_1, h_1 \rangle \\ \langle s_2, h_2 \rangle \end{smallmatrix} \right]$  with  $\langle \langle s_1, h_1 \rangle, \langle s_2, h_2 \rangle \rangle \in \text{States}^2$ .

**Definition 2.55.** A *Hoare quadruple*  $\{R\}_{C_2}^{C_1}\{S\}$  is an ordered quadruple (with a special syntax  $\{\cdot\};\{\cdot\}$ ), where  $R$  and  $S$  are rassertions and  $C_1$  and  $C_2$  are program commands.

**Definition 2.56.** A Hoare quadruple  $\{R\}_{C_2}^{C_1}\{S\}$  is *valid*, written  $\models_{RSL} \{R\}_{C_2}^{C_1}\{S\}$ , iff

1.  $\text{Dom}(R)$  provides the memory required by  $C_1$ ; and
2.  $\text{Rng}(R)$  provides the memory required by  $C_2$ ; and
3.  $\forall \left[ \begin{smallmatrix} \langle s_1, h_1 \rangle \\ \langle s_2, h_2 \rangle \end{smallmatrix} \right] \in R. \forall H_1 \supseteq h_1. \forall H_2 \supseteq h_2. \langle C_1, \langle s_1, H_1 \rangle \rangle$  can diverge  $\Leftrightarrow \langle C_2, \langle s_2, H_2 \rangle \rangle$  can diverge; and
4.  $\forall \left[ \begin{smallmatrix} \sigma_1 \\ \sigma_2 \end{smallmatrix} \right] \in R. \forall \sigma'_1 \in \llbracket C_1 \rrbracket[\{\sigma_1\}]. \forall \sigma'_2 \in \llbracket C_2 \rrbracket[\{\sigma_2\}]. \left[ \begin{smallmatrix} \sigma'_1 \\ \sigma'_2 \end{smallmatrix} \right] \in S.$



The first two conditions, just as in SL, guarantee the soundness of the frame rule (assuming simultaneous divergence is not a concern). Following this, when designing a logic for comparing two programs, it makes sense to check if they have diverging execution paths simultaneously, hence the third condition. However, there is more to the third condition: if we only checked for simultaneous divergence in the current heap rather than in all heap extensions, we'd end up with an unsound frame rule (using the syntax introduced in definition 2.57):

$$\frac{\models_{RSL} \{ \llbracket \text{emp} \rrbracket \}_{x:=\text{alloc()}; \text{while } \llbracket x \doteq 1 \rrbracket \text{ do skip od } \{ \llbracket x \mapsto - \rrbracket \}}{\not\models_{RSL} \{ \llbracket \text{emp} \rrbracket \boxtimes \llbracket 1 \mapsto 1 \rrbracket \}_{x:=\text{alloc()}; \text{while } \llbracket x \doteq 1 \rrbracket \text{ do skip od } \{ \llbracket x \mapsto - \rrbracket \boxtimes \llbracket 1 \mapsto 1 \rrbracket \}} .$$

Indeed, in the first case, both may diverge if allocation returns address 1. In contrast, in the second case, address 1 cannot be returned, as it is already allocated. Since no (easily identifiable) natural and practically useful condition guarantees soundness of the frame rule for divergence (whereas the first two conditions already give us soundness of the frame rule for non-diverging states), the approach taken is to embed the frame rule into the semantics. This way, it constitutes the most general predicate that ensures a sound frame rule for simultaneous divergence. The fourth condition is standard.

In the RSL paper, they approach assertions differently. They assume that the two programs operate with disjoint sets of variables, which allows them to use just one store. Therefore, they represent the assertions as sets of  $\langle s, h_1, h_2 \rangle$  (rather than  $\langle \langle s_1, h_1 \rangle, \langle s_2, h_2 \rangle \rangle$ ). Note that using a single heap to represent the two heaps is not possible (e.g., where even addresses are reserved for the first program and odd addresses for the second) without changing the semantics of the heap-sensitive commands. For instance, `alloc()` would allocate only at even addresses in the first program, and `[4]` would actually refer to address 8 (the fourth positive even number) in the first program, while it would refer to address 7 (the fourth positive odd number) in the second program. They define separating conjunction as

$$R * S \Leftrightarrow \{ \langle s, h_1^R \cup h_1^S, h_2^R \cup h_2^S \rangle \mid \langle s, h_1^R, h_2^R \rangle \in R \wedge \langle s, h_1^S, h_2^S \rangle \in S \wedge h_1^R \perp h_1^S \wedge h_2^R \perp h_2^S \}$$

and prove the soundness of the frame rule. The modified star, as per our model, is defined as

$$R \boxtimes S \Leftrightarrow \{ \langle \langle s_1, h_1^R \cup h_1^S \rangle, \langle s_2, h_2^R \cup h_2^S \rangle \rangle \mid \langle \langle s_1, h_1^R \rangle, \langle s_2, h_2^R \rangle \rangle \in R \wedge \langle \langle s_1, h_1^S \rangle, \langle s_2, h_2^S \rangle \rangle \in S \wedge h_1^R \perp h_1^S \wedge h_2^R \perp h_2^S \}.$$

The choice of model is evidently cosmetic. However, this thesis focuses exclusively on the scenario where the two programs coincide, making the use of their model not directly applicable. To apply their approach, we would assign a prime to all variables in the second coordinate. For instance, a variable  $x$  would be written as  $x'$  in the second coordinate. To avoid such additional modifications, we instead adopt the approach using  $\langle \langle s_1, h_1 \rangle, \langle s_2, h_2 \rangle \rangle$ . Examples of where this restricted scenario is useful are

- **Monotonicity**  
 $\models_{RSL} \{\llbracket \exists k. \frac{x \dot{=} k}{k \geq x} \rrbracket\}^C \{\llbracket \exists k. \frac{y \dot{=} k}{k \geq y} \rrbracket\}$  - bigger input  $x$  results in a bigger output  $y$ ;
- **Injectivity**  
 $\models_{RSL} \{\llbracket \exists k. \frac{x \dot{=} k}{x \neq k} \rrbracket\}^C \{\llbracket \exists k. \frac{y \dot{=} k}{y \neq k} \rrbracket\}$  - different input  $x$  results in a different output  $y$ ;
- **Observational determinism**<sup>30</sup>  
 $\models_{RSL} \{\llbracket \exists k. \frac{x_1 \dot{=} k}{x_1 \dot{=} k} \rrbracket \cap \dots \cap \llbracket \exists k. \frac{x_n \dot{=} k}{x_n \dot{=} k} \rrbracket\}^C \{\llbracket \exists k. \frac{x_1 \dot{=} k}{x_1 \dot{=} k} \rrbracket \cap \dots \cap \llbracket \exists k. \frac{x_n \dot{=} k}{x_n \dot{=} k} \rrbracket\}$ ,  
where  $x_1, \dots, x_n$  are the program variables appearing in  $C$ . Note that  $C$  could be the program  $x := \text{nonDet}(); x := \llbracket I \rrbracket$ , which makes it nondeterministic, but observably deterministic. To express true determinism, we require a state model based on program traces (countable sequences of states) rather than individual states;
- **Non-interference**  
 $\models_{RSL} \{\llbracket \exists k. \frac{x_1 \dot{=} k}{x_1 \dot{=} k} \rrbracket \cap \dots \cap \llbracket \exists k. \frac{x_n \dot{=} k}{x_n \dot{=} k} \rrbracket\}^C \{\llbracket \exists k. \frac{x_1 \dot{=} k}{x_1 \dot{=} k} \rrbracket \cap \dots \cap \llbracket \exists k. \frac{x_n \dot{=} k}{x_n \dot{=} k} \rrbracket\}$ ,  
meaning variables  $\{x_1, \dots, x_n\}$  are not influenced by variables  $\text{PVars} \setminus \{x_1, \dots, x_n\}$  in  $C$ . On a high level, non-interference says that there is no "leakage" of information from  $\text{PVars} \setminus \{x_1, \dots, x_n\}$  to  $\{x_1, \dots, x_n\}$ . In the literature, first introduced in [Goguen and Meseguer 1982], they are often called *high-* and *low-sensitivity* variables, respectively.

## 2.5.2 Syntax

In this subsection, we formalize the syntax and its interpretation used above.

**Definition 2.57.** We define  $\mathbb{N}$ -rassertions using BNF:

$$\mathcal{R} \Leftarrow \text{Same} \mid \llbracket \mathcal{P} \rrbracket \mid (\mathcal{R} \boxtimes \mathcal{R}) \mid (\mathcal{R} \wedge \mathcal{R}) \mid (\mathcal{R} \vee \mathcal{R}) \mid (\mathcal{R} \Rightarrow \mathcal{R}) \mid (\mathcal{R} \Leftrightarrow \mathcal{R}) \mid \neg \mathcal{R} \mid \exists x. \mathcal{R} \mid \forall x. \mathcal{R},$$

where  $\mathcal{P}, \mathcal{Q} \in \mathbb{N}$ -assertions and  $x \in \text{PVars}$ .

The interpretation of interest  $\llbracket \cdot \rrbracket$  is as follows:

$$\begin{aligned} \llbracket \text{Same} \rrbracket &\Leftarrow \{ \llbracket \frac{\langle s_1, h_1 \rangle}{\langle s_2, h_2 \rangle} \rrbracket \in \text{States}^2 \mid h_1 = h_2 \} \\ \llbracket \llbracket \mathcal{P} \rrbracket \rrbracket &\Leftarrow \{ \llbracket \frac{\langle s_1, h_1 \rangle}{\langle s_2, h_2 \rangle} \rrbracket \in \text{States}^2 \mid \langle s_1, h_1 \rangle \in \llbracket \mathcal{P} \rrbracket \wedge \langle s_2, h_2 \rangle \in \llbracket \mathcal{Q} \rrbracket \} \\ \llbracket \mathcal{R} \boxtimes \mathcal{S} \rrbracket &\Leftarrow \llbracket \mathcal{R} \rrbracket \boxtimes \llbracket \mathcal{S} \rrbracket \\ \llbracket \exists x. \mathcal{R} \rrbracket &\Leftarrow \{ \llbracket \frac{\langle s_1, h_1 \rangle}{\langle s_2, h_2 \rangle} \rrbracket \in \text{States}^2 \mid \exists n. \llbracket \frac{\langle (s_1)_x^n, h_1 \rangle}{\langle (s_2)_x^n, h_2 \rangle} \rrbracket \in \llbracket \mathcal{R} \rrbracket \} \\ \llbracket \forall x. \mathcal{R} \rrbracket &\Leftarrow \{ \llbracket \frac{\langle s_1, h_1 \rangle}{\langle s_2, h_2 \rangle} \rrbracket \in \text{States}^2 \mid \forall n. \llbracket \frac{\langle (s_1)_x^n, h_1 \rangle}{\langle (s_2)_x^n, h_2 \rangle} \rrbracket \in \llbracket \mathcal{R} \rrbracket \} \end{aligned}$$

with the remaining logical symbols receiving their standard (in the sense of Tarski) interpretation.

<sup>30</sup>The definition used here differs from the one commonly found in the literature.

An important aspect of the syntax is the ability to express connections/relations between the two coordinates, say  $\{\llbracket \langle \begin{smallmatrix} s_1, h_1 \\ s_2, h_2 \end{smallmatrix} \rangle \rrbracket \mid 4s_1(x) + s_1(y) < s_2(x)s_2(y)\}$ . We can express it using the existential quantifier:

$$\llbracket \exists x_1. \exists y_1. \llbracket \begin{smallmatrix} x \doteq x_1 \wedge y \doteq y_1 \\ 4 \cdot x_1 + y_1 < x \cdot y \end{smallmatrix} \rrbracket \rrbracket = \{\llbracket \langle \begin{smallmatrix} s_1, h_1 \\ s_2, h_2 \end{smallmatrix} \rangle \rrbracket \mid 4s_1(x) + s_1(y) < s_2(x)s_2(y)\}.$$

Roughly speaking, what we do is "copy" the  $x$  and  $y$  from the first coordinate into  $x_1$  and  $y_1$ , respectively, and then use the copies in the second coordinate. Analogously, we can express connections between the heaps as well:

$$\llbracket \exists n. \llbracket \begin{smallmatrix} x \mapsto n \\ y \mapsto n \end{smallmatrix} \rrbracket \rrbracket = \{\llbracket \langle \begin{smallmatrix} s_1, h_1 \\ s_2, h_2 \end{smallmatrix} \rangle \rrbracket \mid h_1(s_1(x)) = h_2(s_2(y))\}.$$

## 2.6 Properties and hyperproperties

In this subsection, we introduce the notion of heapless program properties and, more importantly, heapless program hyperproperties, which are essential for fully grasping the capabilities of the (later introduced) Hyper Hoare logic. The definition of the heap variants—program property and program hyperproperty—are defined analogously, but are not considered in this thesis.

**Definition 2.58.** A heapless program property  $\mathfrak{p}$  is a set of pairs of heapless program states, i.e.  $\mathfrak{p} \in \mathcal{P}(\text{Stacks}^2)$ .

We denote the set of all heapless program properties by  $\text{SProps}$ . We will use the meta symbol  $\mathfrak{p}$  with potential indices to denote a heapless program property.

**Definition 2.59.** We say that a heapless command  $C$  satisfies a heapless program property  $\mathfrak{p}$  iff  $\llbracket C \rrbracket \subseteq \mathfrak{p}$ .

For example, there exists a function  $\mathfrak{p} : \text{SArts}^2 \rightarrow \text{SProps}$  such that for all  $p, C$  and  $q$ ,

$$\models_{HL} \{p\}C\{q\} \Leftrightarrow \llbracket C \rrbracket \subseteq \mathfrak{p}(p, q).$$

Indeed,  $\mathfrak{p}(p, q) \Leftarrow \{\langle s, s' \rangle \mid s \in p \Rightarrow s' \in q\}$  is a witness:

$$\begin{aligned} \models_{HL} \{p\}C\{q\} &\stackrel{def}{\Leftrightarrow} \forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C \rrbracket \Rightarrow s' \in q \\ &\Leftrightarrow \forall \langle s, s' \rangle \in \llbracket C \rrbracket. s \in p \Rightarrow s' \in p \\ &\Leftrightarrow \forall \langle s, s' \rangle \in \llbracket C \rrbracket. \langle s, s' \rangle \in \{\langle s, s' \rangle \mid s \in p \Rightarrow s' \in q\} \\ &\Leftrightarrow \llbracket C \rrbracket \subseteq \{\langle s, s' \rangle \mid s \in p \Rightarrow s' \in q\} = \mathfrak{p}(p, q) \end{aligned}$$

However, for any  $p$  and  $q$  such that  $p \neq \emptyset$ , it is impossible to equivalently state  $\models_{SIL} \{p\}C\{q\}$  via satisfaction of some property. That is, for any property  $\mathfrak{p} \in \text{SProps}$ , there exists  $C$  such that  $\models_{SIL} \{p\}C\{q\} \not\Leftrightarrow \llbracket C \rrbracket \subseteq \mathfrak{p}$ . Indeed,  $C \Leftarrow \mathbf{while} \llbracket x \doteq x \rrbracket \mathbf{do skip od}$  is a (uniform) witness:

$$\models_{SIL} \{p\}\mathbf{while} \llbracket x \doteq x \rrbracket \mathbf{do skip od}\{q\} \not\Leftrightarrow \llbracket C \rrbracket \subseteq \mathfrak{p},$$

since the left side is false, whereas the right side is trivially true. Therefore, there does not exist  $\mathbf{p} : \mathbf{SAsrts}^2 \rightarrow \mathbf{SProps}$  such that for all  $p, C$  and  $q$ ,  $\models_{SIL} \{p\}C\{q\} \Leftrightarrow \llbracket C \rrbracket \subseteq \mathbf{p}(p, q)$ .

In order to show further examples, we first need to generalize our state model to reason about non-termination, i.e.  $\mathbf{Stacks}^\uparrow \Leftarrow \mathbf{Stacks} \cup \{\uparrow\}$ , where  $\uparrow \notin \mathbf{Stacks}$  is denoting divergence. We generalize  $\llbracket \cdot \rrbracket$  to  $\llbracket \cdot \rrbracket^\uparrow$  in the obvious way, e.g.  $\llbracket \mathbf{while} \llbracket x \doteq x \rrbracket \mathbf{do skip od} \rrbracket^\uparrow = \{\langle s, \uparrow \rangle \mid s \in \mathbf{Stacks}^\uparrow\}$ . We also generalize heapless program property to be any  $\mathbf{p} \subseteq \mathbf{Stacks}^\uparrow \times \mathbf{Stacks}^\uparrow$  (and its satisfaction analogously) and trust that the context will clear any ambiguities<sup>31</sup>. Then the heapless property "termination on  $p$ " is

$$\mathbf{p} \Leftarrow \{\langle s, s' \rangle \in \mathbf{Stacks} \times \mathbf{Stacks}^\uparrow \mid s \in p \Rightarrow s' \neq \uparrow\},$$

i.e. any program  $C$  satisfying  $\mathbf{p}$ ,  $\llbracket C \rrbracket^\uparrow \subseteq \mathbf{p}$ , terminates for any execution path, starting with input state  $s \in p$ . Formally, let

$$\mathbf{div}(s, C) \Leftarrow s \neq \uparrow \Rightarrow \exists f : \mathbb{N} \rightarrow \mathbf{SConfs}. f(0) = s \wedge \forall n \in \mathbb{N}. f(n) \rightarrow f(n+1),$$

where  $\mathbf{SConfs}$  is the set of all heapless program configurations, and

$$\llbracket C \rrbracket^\uparrow \Leftarrow \llbracket C \rrbracket \cup \{\langle s, \uparrow \rangle \mid s \in \mathbf{Stacks}^\uparrow \wedge \mathbf{div}(s, C)\}.$$

Then we have

$$\forall s \in p. \neg \mathbf{div}(s, C) \iff \llbracket C \rrbracket^\uparrow \subseteq \{\langle s, s' \rangle \in \mathbf{Stacks} \times \mathbf{Stacks}^\uparrow \mid s \in p \Rightarrow s' \neq \uparrow\}.$$

Now that we have a way to reason about non-termination, one might expect that we can now equivalently state SIL's validity, but it remains impossible. To show this, we define validity over this generalized state model<sup>32</sup>:

$$\models_{HL}^\uparrow \{p\}C\{q\} \stackrel{def}{\iff} \forall s \in p. \forall s' \neq \uparrow. \langle s, s' \rangle \in \llbracket C \rrbracket^\uparrow \Rightarrow s' \in q$$

and

$$\models_{SIL}^\uparrow \{p\}C\{q\} \stackrel{def}{\iff} \forall s \in p. \exists s' \in q \setminus \{\uparrow\}. \langle s, s' \rangle \in \llbracket C \rrbracket^\uparrow.$$

The initial HL example can be equivalently stated as follows

$$\models_{HL}^\uparrow \{p\}C\{q\} \Leftrightarrow \llbracket C \rrbracket^\uparrow \subseteq \{\langle s, s' \rangle \mid s \in p \Rightarrow s' \in q \vee s' = \uparrow\}.$$

We claim that there isn't  $\mathbf{p}$  such that for all  $C$ ,

$$\models_{SIL}^\uparrow \{\llbracket \top \rrbracket\}C\{\llbracket \top \rrbracket\} \iff \llbracket C \rrbracket^\uparrow \subseteq \mathbf{p},$$

Indeed, suppose that such property exists. Then, since

$$\models_{SIL}^\uparrow \{\llbracket \top \rrbracket\}C\{\llbracket \top \rrbracket\} \Rightarrow \llbracket C \rrbracket^\uparrow \subseteq \mathbf{p},$$

<sup>31</sup>In general, (heapless) program properties are defined using infinite traces, where the final state repeats upon termination, rather than input-output pairs—an approach that is essential for parallel programs.

<sup>32</sup>We also show the generalization of HL's validity to further clarify the concept.

it follows that  $\mathfrak{p}$  must contain all pairs  $\langle s, \uparrow \rangle$ ,  $s \in \text{Stacks}^\uparrow$ , since for

$$C \Leftarrow \text{skip} + \text{while } \llbracket x \doteq x \rrbracket \text{ do skip od},$$

we have that  $\models_{\text{SIL}}^\uparrow \{\llbracket \top \rrbracket\}C\{\llbracket \top \rrbracket\}$  and hence

$$\{\langle s, s \rangle, \langle s, \uparrow \rangle \mid s \in \text{Stacks}^\uparrow\} = \llbracket C \rrbracket^\uparrow \subseteq \mathfrak{p}.$$

But then the heapless program command  $C' \Leftarrow \text{while } \llbracket x \doteq x \rrbracket \text{ do skip od}$  with  $\llbracket C' \rrbracket^\uparrow = \{\langle s, \uparrow \rangle \mid s \in \text{Stacks}^\uparrow\}$  satisfies  $\mathfrak{p}$ , i.e.  $\llbracket C' \rrbracket^\uparrow \subseteq \mathfrak{p}$ , whereas  $\not\models_{\text{SIL}}^\uparrow \{\llbracket \top \rrbracket\}C'\{\llbracket \top \rrbracket\}$  - contradiction. Therefore,  $\models_{\text{SIL}}^\uparrow \{p\}C\{q\}$  cannot be equivalently stated via satisfiability of (heapless) program properties.

In order to equivalently state  $\models_{\text{SIL}} \{p\}C\{q\}$ ,  $\models_{\text{OL}} \{p\}C\{q\}$  and  $\models_{\text{RHL}} \{p\}C\{q\}$ , we introduce the so called heapless program hyperproperties.

**Definition 2.60.** *A heapless program hyperproperty  $\mathfrak{P}$  is a set of sets of pairs of heapless program states, i.e.  $\mathfrak{P} \subseteq \mathcal{P}(\text{Stacks}^2)$ .*

We denote the set of all heapless program hyperproperties by  $\text{SHProps}$ . We will use the meta symbol  $\mathfrak{P}$  with potential indices to denote a heapless program hyperproperty.

**Definition 2.61.** *We say that a heapless command  $C$  satisfies a heapless program hyperproperty  $\mathfrak{P}$  iff  $\llbracket C \rrbracket \in \mathfrak{P}$ .*

We can equivalently state  $\models_{\text{HL}} \{p\}C\{q\}$ ,  $\models_{\text{SIL}} \{p\}C\{q\}$ ,  $\models_{\text{OL}} \{p\}C\{q\}$  and  $\models_{\text{RHL}} \{p\}C\{q\}$ , using heapless hyperproperties:

$$\begin{aligned} \models_{\text{HL}} \{p\}C\{q\} &\Leftrightarrow \llbracket C \rrbracket \in \{X \subseteq \text{Stacks}^2 \mid \forall s \in p. \forall s'. \langle s, s' \rangle \in X \Rightarrow s' \in q\} \\ \models_{\text{SIL}} \{p\}C\{q\} &\Leftrightarrow \llbracket C \rrbracket \in \{X \subseteq \text{Stacks}^2 \mid \forall s \in p. \exists s' \in q. \langle s, s' \rangle \in X\} \\ \models_{\text{OL}} \{P\}C\{Q\} &\Leftrightarrow \llbracket C \rrbracket \in \{X \subseteq \text{Stacks}^2 \mid \forall S \in P. X[S] \in Q\} \\ \models_{\text{RHL}} \{R\}_C^{\mathcal{C}}\{S\} &\Leftrightarrow \llbracket C \rrbracket \in \{X \subseteq \text{Stacks}^2 \mid \forall [s_1] \in R. \forall s'_1 \in X[\{s_1\}]. \forall s'_2 \in X[\{s_2\}]. [s'_2] \in S\}, \end{aligned}$$

where

$$\models_{\text{OL}} \{P\}C\{Q\} \stackrel{\text{def}}{\Leftrightarrow} \forall S \in P. \llbracket C \rrbracket[S] \in Q$$

and

$$\models_{\text{RHL}} \{R\}_{C_2}^{C_1}\{S\} \stackrel{\text{def}}{\Leftrightarrow} \forall [s_1] \in R. \forall s'_1 \in \llbracket C_1 \rrbracket[\{s_1\}]. \forall s'_2 \in \llbracket C_2 \rrbracket[\{s_2\}]. [s'_2] \in S.$$

Note that, unlike RSL, simultaneous divergence is not taken into account in RHL (see [Benton 2004]), which we believe to be purely a design choice in the sense that both alternatives are viable, each with its differences. Had it been considered, we would need the generalized state model—the one that accounts for non-termination—to state it equivalently with heapless program hyperproperties.

In general, (see [Clarkson and Schneider 2008]) heapless program hyperproperties are defined via program traces—infinite sequences of states—to facilitate reasoning about concurrency. This, however, lies outside the scope of this thesis.

## 2.7 Hyper Hoare logic

### 2.7.1 Semantics

The final preliminary Hoare-style logic we need to introduce is Hyper Hoare logic (abbreviated HHL; see [Dardinier and Müller 2024]), which is, in fact, the logic whose separation variant we aim to design in this thesis and more precisely, its frame rule. Similarly to OL, HHL’s pre- and postconditions are not merely sets of (heapless) states, but rather sets of sets of (heapless) states, called analogously heapless hyper-assertions (or heapless hassertions). However, a key difference is that in OL, they consider only specific type of heapless hassertions (see [Zilberstein et al. 2023]), whereas HHL considers arbitrary heapless hassertions. In [Dardinier and Müller 2024], they show that they can prove or disprove arbitrary heapless program hyperproperties. Therefore, using their validity, they can capture HL, SIL, OL, RHL and beyond. Note that they use logical variables in addition to the program variables, whereas we won’t for simplicity. However, we will still be able to capture the essence of these 4 logics with this simplification in place.

**Definition 2.62.** *A heapless hyper-assertion (or heapless hassertion)  $P$  is a set of heapless assertions.*

We denote the set of all heapless hassertions by SHAsrts. We will use the meta symbols  $P, Q, R, F$  with potential indices to denote heapless hassertion.

**Definition 2.63.** *A heapless hyper-triple  $\{P\}C\{Q\}$  is an ordered triple (with a special syntax  $\{\cdot\} \cdot \{\cdot\}$ ), where  $P$  and  $Q$  are heapless hassertions and  $C$  is a heapless program command.*

**Definition 2.64.** *A heapless hyper-triple  $\{P\}C\{Q\}$  validity is defined as follows:*

$$\models_{HHL} \{P\}C\{Q\} \stackrel{def}{\iff} \forall S \in P. \llbracket C \rrbracket [S] \in Q.$$

Intuitively, an HHL triple  $\{P\}C\{Q\}$  is valid iff the strongest postcondition of every  $p \in P$  and  $C$  belongs to  $Q$ , i.e.  $\forall p \in P. \text{sp}(p, C) \in Q$ . A rule that sheds light on the nature of HHL is

$$\frac{\models_{HHL} \{P\}C_1\{Q_1\} \quad \models_{HHL} \{P\}C_2\{Q_2\}}{\models_{HHL} \{P\}C_1 + C_2\{Q_1 \otimes Q_2\}} \text{ (Choice)}.$$

Note that the resulting postcondition is  $Q_1 \otimes Q_2 = \{S_1 \cup S_2 \mid S_1 \in Q_1 \wedge S_2 \in Q_2\}$  and not  $Q_1 \cup Q_2$ . That’s because  $\text{sp}(p, C_1 + C_2) = \text{sp}(p, C_1) \cup \text{sp}(p, C_2)$  and by the assumption we know that  $\text{sp}(p, C_1) \in Q_1$  and  $\text{sp}(p, C_2) \in Q_2$ . Therefore,  $\text{sp}(p, C_1 + C_2) \in Q_1 \otimes Q_2$ . If we were to show that  $\text{sp}(p, C_1 + C_2) \in Q_1 \cup Q_2$ , then it would’ve been the case that  $\text{sp}(p, C_1 + C_2) \in Q_1$  or  $\text{sp}(p, C_1 + C_2) \in Q_2$ , which is clearly not the case (in general). An example, illustrating the above:

$$\frac{\begin{array}{c} \overbrace{\models_{HHL} \{\{\{s\}\}\}x := \llbracket 0 \rrbracket \{\{\{s_x^0\}\}\}}^P \quad \overbrace{\models_{HHL} \{\{\{s\}\}\}x := \llbracket 1 \rrbracket \{\{\{s_x^1\}\}\}}^{Q_2} \\ \overbrace{\models_{HHL} \{\{\{s\}\}\}x := \llbracket 0 \rrbracket + x := \llbracket 1 \rrbracket \{\{\{s_x^0, s_x^1\}\}\}}^P \end{array}}{\overbrace{\models_{HHL} \{\{\{s\}\}\}x := \llbracket 0 \rrbracket + x := \llbracket 1 \rrbracket \{\{\{s_x^0, s_x^1\}\}\}}^{Q_1 \otimes Q_2}}.$$

It is clear that  $\{\mathfrak{s}_x^0, \mathfrak{s}_x^1\} \notin Q_1 \cup Q_2$ .

In a similar manner to  $\text{sp}(p, C)$ , we define  $\text{SP}(P, C) \Leftarrow \{\llbracket C \rrbracket[S] \mid S \in P\}$  and show that for any  $P, Q$  and  $C$ :

1.  $\models_{\text{HHL}} \{P\}C\{\text{SP}(P, C)\}$ ; and
2.  $\models_{\text{HHL}} \{P\}C\{Q\} \Rightarrow \text{SP}(P, C) \subseteq Q$ .

[Dardinier and Müller 2024] establish the (semantic) completeness of HHL, with the only non-trivial aspect being the inductive steps for  $C \equiv C_1 + C_2$  and  $C \equiv C_0^*$ . Notably, only one direction holds:  $\text{SP}(P, C_1 + C_2) \subseteq \text{SP}(P, C_1) \otimes \text{SP}(P, C_2)$  for arbitrary  $P$  and  $C$ .

Let  $\mathfrak{s} \in \text{Stacks}$  be such that  $\forall x. \mathfrak{s}(x) = 0$ . Now, the choice rule

$$\frac{\begin{array}{c} \overbrace{\models_{\text{HHL}} \{\{\mathfrak{s}\}, \{\mathfrak{s}_x^2\}\} \text{skip}}^P \overbrace{\{\{\mathfrak{s}\}, \{\mathfrak{s}_x^2\}\}}^{Q_1} \\ \overbrace{\models_{\text{HHL}} \{\{\mathfrak{s}\}, \{\mathfrak{s}_x^2\}\} \text{skip}}^P \overbrace{\{x := \llbracket x + 1 \rrbracket \{\{\mathfrak{s}_x^1\}, \{\mathfrak{s}_x^3\}\}\}}^{Q_2} \end{array}}{\models_{\text{HHL}} \underbrace{\{\{\mathfrak{s}\}, \{\mathfrak{s}_x^2\}\} \text{skip}}_P + x := \llbracket x + 1 \rrbracket \underbrace{\{\{\mathfrak{s}, \mathfrak{s}_x^1\}, \{\mathfrak{s}, \mathfrak{s}_x^3\}, \{\mathfrak{s}_x^2, \mathfrak{s}_x^1\}, \{\mathfrak{s}_x^2, \mathfrak{s}_x^3\}\}}_{Q_1 \otimes Q_2}} \text{ (Choice)}$$

does not yield the strongest postcondition, despite the premises having the strongest postcondition, i.e.  $\text{SP}(P, \text{skip}) = Q_1$  and  $\text{SP}(P, x := \llbracket x + 1 \rrbracket) = Q_2$ , but  $\text{SP}(P, \text{skip} + x := \llbracket x + 1 \rrbracket) \supset Q_1 \otimes Q_2$ . Note that  $|\text{SP}(P, C)| \leq |P|$ , since every assertion  $p$  from the hassertion  $P$  ends up as  $\text{sp}(p, C)$  in the postcondition  $Q$  and since  $\text{sp}(p_1, C)$  and  $\text{sp}(p_2, C)$  could coincide for different  $p_1, p_2 \in P$ . To overcome the problem above, we work with hassertions  $P$  with cardinality 1. Then it is guaranteed that the strongest postcondition  $Q = \text{SP}(P, C)$  has cardinality 1, i.e.  $|P| = 1 \Rightarrow |\text{SP}(P, C)| = 1$ . That way, applying the choice rule won't join  $\text{sp}(\{\mathfrak{s}\}, \text{skip}) = \{\mathfrak{s}\}$  with  $\text{sp}(\{\mathfrak{s}_x^2\}, x := \llbracket x + 1 \rrbracket) = \{\mathfrak{s}_x^3\}$  and  $\text{sp}(\{\mathfrak{s}_x^2\}, \text{skip}) = \{\mathfrak{s}_x^2\}$  with  $\text{sp}(\{\mathfrak{s}\}, x := \llbracket x + 1 \rrbracket) = \{\mathfrak{s}_x^1\}$ , since the precondition cannot have cardinality 2, i.e.  $P$  cannot be  $\{\{\mathfrak{s}\}, \{\mathfrak{s}_x^2\}\}$ . After obtaining the strongest postcondition, by applying the choice rule over premises with preconditions with cardinality 1, we apply the union rule

$$\frac{\models_{\text{HHL}} \{P_1\}C\{Q_1\} \quad \models_{\text{HHL}} \{P_2\}C\{Q_2\}}{\models_{\text{HHL}} \{P_1 \cup P_2\}C\{Q_1 \cup Q_2\}} \text{ (Union)},$$

which preserves the strongest postcondition:

$$\frac{\begin{array}{c} \overbrace{\models_{\text{HHL}} \{\{\mathfrak{s}\}\} \text{skip}}^{P_1} \overbrace{\{\{\mathfrak{s}\}\}}^{Q_{11}} \\ \overbrace{\models_{\text{HHL}} \{\{\mathfrak{s}\}\} \text{skip}}^{P_1} \overbrace{\{x := \llbracket x + 1 \rrbracket \{\{\mathfrak{s}_x^1\}\}\}}^{Q_{12}} \end{array}}{\models_{\text{HHL}} \underbrace{\{\{\mathfrak{s}\}\} \text{skip}}_{P_1} + x := \llbracket x + 1 \rrbracket \underbrace{\{\{\mathfrak{s}, \mathfrak{s}_x^1\}\}}_{Q_{11} \otimes Q_{12}}} \text{ (Choice)} \quad \frac{\begin{array}{c} \overbrace{\models_{\text{HHL}} \{\{\mathfrak{s}_x^2\}\} \text{skip}}^{P_2} \overbrace{\{\{\mathfrak{s}_x^2\}\}}^{Q_{21}} \\ \overbrace{\models_{\text{HHL}} \{\{\mathfrak{s}_x^2\}\} \text{skip}}^{P_2} \overbrace{\{x := \llbracket x + 1 \rrbracket \{\{\mathfrak{s}_x^3\}\}\}}^{Q_{22}} \end{array}}{\models_{\text{HHL}} \underbrace{\{\{\mathfrak{s}_x^2\}\} \text{skip}}_{P_2} + x := \llbracket x + 1 \rrbracket \underbrace{\{\{\mathfrak{s}_x^2, \mathfrak{s}_x^3\}\}}_{Q_{21} \otimes Q_{22}}} \text{ (Choice)}}{\models_{\text{HHL}} \underbrace{\{\{\mathfrak{s}\}, \{\mathfrak{s}_x^2\}\} \text{skip}}_{P_1 \cup P_2} + x := \llbracket x + 1 \rrbracket \underbrace{\{\{\mathfrak{s}, \mathfrak{s}_x^1\}, \{\mathfrak{s}_x^2, \mathfrak{s}_x^3\}\}}_{(Q_{11} \otimes Q_{12}) \cup (Q_{21} \otimes Q_{22})}} \text{ (Union)}$$

For infinite preconditions, the same approach is applied, but with the infinite union rule

$$\frac{\text{for all } i \in I. \models_{\text{HHL}} \{P_i\}C\{Q_i\}}{\models_{\text{HHL}} \left\{ \bigcup_{i \in I} P_i \right\} C \left\{ \bigcup_{i \in I} Q_i \right\}} \text{ (Idx-Union)}$$

The iteration case of the induction hypothesis follows analogously, using the rule

$$\frac{\text{for all } n \in \mathbb{N}. \models_{HHL} \{I_n\} C \{I_{n+1}\}}{\models_{HHL} \{I_0\} C^* \{\bigotimes_{n \in \mathbb{N}} I_n\}} \text{ (Iter)},$$

where  $\bigotimes_{i \in I} X_i \Leftrightarrow \{\bigcup_{i \in I} f(i) \mid \forall i \in I. f(i) \in X_i\}$ <sup>33</sup>. More precisely, we break down the precondition  $P = \bigcup_{p \in P} X_0^{(p)}$ , where  $X_0^{(p)} = \{p\}$  and define  $X_{n+1}^{(p)} \Leftrightarrow \text{SP}(X_n^{(p)}, C)$ , each with cardinality of 1. Then by the iteration rule, we obtain  $\models_{HHL} \{X_0^{(p)}\} C^* \{\bigotimes_{n \in \mathbb{N}} X_n^{(p)}\}$ , where  $|\bigotimes_{n \in \mathbb{N}} X_n^{(p)}| = 1$  and hence is the strongest postcondition. Therefore, by the infinite union rule, we obtain that  $\models_{HHL} \{\bigcup_{p \in P} X_0^{(p)}\} C^* \{\bigcup_{p \in P} (\bigotimes_{n \in \mathbb{N}} X_n^{(p)})\}$ . Finally, since infinite union preserves the strongest postcondition (easily verifiable), we conclude that  $\text{SP}(P, C^*) = \bigcup_{p \in P} (\bigotimes_{n \in \mathbb{N}} X_n^{(p)})$ .

### 2.7.2 Syntax

Let  $\text{SVars}$  be a countably infinite set of stack variables such that  $\text{SVars} \cap \text{PVars} = \emptyset$ . We introduce a slight alteration of the syntax presented in [Dardinier and Müller 2024].

**Definition 2.65.** *We define heapless  $\mathbb{N}$ -hassertions using BNF:*

$$\mathcal{P} \Leftrightarrow \top \mid \perp \mid \mathcal{J}(\mathcal{P}) \mid [\mathcal{J}_1^{\mathcal{A}}]_{\mathcal{J}_2}(\mathcal{A}) \mid \exists x. \mathcal{P} \mid \forall x. \mathcal{P} \mid \exists \langle \mathcal{J} \rangle. \mathcal{P} \mid \forall \langle \mathcal{J} \rangle. \mathcal{P} \mid (\mathcal{P} \wedge \mathcal{P}) \mid (\mathcal{P} \vee \mathcal{P}),$$

where  $\mathcal{P}$  is a heapless  $\mathbb{N}$ -assertion,  $\mathcal{A}$  is a heapless  $\mathbb{N}$ -rassertion<sup>34</sup>,  $x \in \text{PVars}$  and  $\mathcal{J}, \mathcal{J}_1, \mathcal{J}_2 \in \text{SVars}$ .

Note that the notation  $\langle \cdot \rangle$  is used purely for readability. That is, since  $\text{PVars} \cap \text{SVars} = \emptyset$ , then the distinction between  $\mathcal{J} \in \text{SVars}$  and  $x \in \text{PVars}$  is clear regardless. The interpretation of interest  $\llbracket \cdot \rrbracket_{\Sigma}$ , where  $\Sigma : \text{SVars} \rightarrow \text{Stacks}$ ,

<sup>33</sup>Informally, if  $I = \{i_1, i_2, \dots\}$  is countable, then  $\bigotimes_{i \in I} X_i = \{S_{i_1} \cup S_{i_2} \cup \dots \mid S_{i_1} \in X_{i_1} \wedge S_{i_2} \in X_{i_2} \wedge \dots\}$ .

<sup>34</sup>We trust that the reader can define such a notion.



is as follows:

$$\begin{aligned}
\llbracket \top \rrbracket_\Sigma &\Leftarrow \text{SAsrts} \\
\llbracket \perp \rrbracket_\Sigma &\Leftarrow \emptyset \\
\llbracket \lambda j. \mathcal{P} \rrbracket_\Sigma &\Leftarrow \{S \in \text{SAsrts} \mid \Sigma(j) \in \llbracket \mathcal{P} \rrbracket\} \\
\llbracket [\lambda_{j_1}^{j_2}]. \mathcal{R} \rrbracket_\Sigma &\Leftarrow \{S \in \text{SAsrts} \mid \llbracket \frac{\Sigma(j_1)}{\Sigma(j_2)} \rrbracket \in \llbracket \mathcal{R} \rrbracket\} \\
\llbracket \exists x. \mathcal{P} \rrbracket_\Sigma &\Leftarrow \{S \in \text{SAsrts} \mid \exists n. \{s_x^n \mid s \in S\} \in \llbracket \mathcal{P} \rrbracket_{(\lambda j. \Sigma(j)_x^n)}\} \\
\llbracket \forall x. \mathcal{P} \rrbracket_\Sigma &\Leftarrow \{S \in \text{SAsrts} \mid \forall n. \{s_x^n \mid s \in S\} \in \llbracket \mathcal{P} \rrbracket_{(\lambda j. \Sigma(j)_x^n)}\} \\
\llbracket \exists \langle j \rangle. \mathcal{P} \rrbracket_\Sigma &\Leftarrow \{S \in \text{SAsrts} \mid \exists s \in S. S \in \llbracket \mathcal{P} \rrbracket_{\Sigma_s^s}\} \\
\llbracket \forall \langle j \rangle. \mathcal{P} \rrbracket_\Sigma &\Leftarrow \{S \in \text{SAsrts} \mid \forall s \in S. S \in \llbracket \mathcal{P} \rrbracket_{\Sigma_s^s}\} \\
\llbracket \mathcal{P} \wedge \mathcal{Q} \rrbracket_\Sigma &\Leftarrow \llbracket \mathcal{P} \rrbracket_\Sigma \cap \llbracket \mathcal{Q} \rrbracket_\Sigma \\
\llbracket \mathcal{P} \vee \mathcal{Q} \rrbracket_\Sigma &\Leftarrow \llbracket \mathcal{P} \rrbracket_\Sigma \cup \llbracket \mathcal{Q} \rrbracket_\Sigma.
\end{aligned}$$

The formulae  $\mathcal{P}$  of interest are those with no free variables from  $\text{SVars}$ , i.e.  $\text{fv}(\mathcal{P}) \cap \text{SVars} = \emptyset$ .

**Lemma 2.66.** *Let  $P, \Sigma$  and  $\Sigma'$  be such that  $\forall j \in \text{fv}(P) \cap \text{SVars}. \Sigma(j) = \Sigma'(j)$ . Then  $\llbracket P \rrbracket_\Sigma = \llbracket P \rrbracket_{\Sigma'}$ .*

*Proof.* See lemma `inter_depends_on_free2` in `HeaplessSyntax.thy`.  $\square$

**Corollary 2.67.** *Let  $P$  be such that  $\text{fv}(P) \cap \text{SVars} = \emptyset$ . Then for every  $\Sigma$  and  $\Sigma'$ ,  $\llbracket P \rrbracket_\Sigma = \llbracket P \rrbracket_{\Sigma'}$ .*

*Proof.* Follows directly by lemma 2.66.  $\square$

Since we will consider only such closed formulae (w.r.t.  $\text{SVars}$ ) and they are not dependent on  $\Sigma$ , then we introduce the abbreviation  $\llbracket P \rrbracket \Leftarrow \llbracket P \rrbracket_{\Sigma_0}$ , for some fixed  $\Sigma_0$ .

### 2.7.3 Expressivity

Using this syntax, HHL can capture HL, SIL, OL and RHL:

**Lemma 2.68.** *The following properties hold:*

1.  $\models_{HL} \{\llbracket \mathcal{P} \rrbracket\}C\{\llbracket \mathcal{Q} \rrbracket\} \iff \models_{HHL} \{\llbracket \forall \langle j \rangle. \lambda j. \mathcal{P} \rrbracket\}C\{\llbracket \forall \langle j \rangle. \lambda j. \mathcal{Q} \rrbracket\}$
2.  $\models_{SIL} \{\llbracket \mathcal{P} \rrbracket\}C\{\llbracket \mathcal{Q} \rrbracket\} \iff \models_{HHL} \{\llbracket \exists \langle j \rangle. \lambda j. \mathcal{P} \rrbracket\}C\{\llbracket \exists \langle j \rangle. \lambda j. \mathcal{Q} \rrbracket\}$
3.  $\models_{OL} \{\llbracket \mathcal{P} \rrbracket\}C\{\llbracket \mathcal{Q} \rrbracket\} \iff \models_{HHL} \{\llbracket \text{enc}_{OL}(\mathcal{P}) \rrbracket\}C\{\llbracket \text{enc}_{OL}(\mathcal{Q}) \rrbracket\}$ , where

- Let `toDNF` be an algorithm, which takes as an input OL formula and returns an OL formula such that for any OL formula  $\mathcal{P}$ , we have  $\llbracket \text{toDNF}(\mathcal{P}) \rrbracket = \llbracket \mathcal{P} \rrbracket$  and

$$\text{toDNF}(\mathcal{P}) = (\mathcal{P}^+(\mathcal{P}_1^{(1)}) \otimes \dots \otimes \mathcal{P}^+(\mathcal{P}_{n_1}^{(1)})) \vee \dots \vee (\mathcal{P}^+(\mathcal{P}_1^{(k)}) \otimes \dots \otimes \mathcal{P}^+(\mathcal{P}_{n_k}^{(k)})),$$

where the syntax of OL is  $\mathcal{P} \Leftarrow \mathcal{P}^+(\mathcal{p}) \mid \mathcal{P} \vee \mathcal{P} \mid \mathcal{P} \otimes \mathcal{P}, \mathcal{p} \in \text{SynSAsrts}^{35}$ . Such an algorithm exists—analogueous to how every propositional formula can be algorithmically transformed into disjunctive normal form—since  $\otimes$  distributes over  $\vee$ .

- Now,  $\text{enc}_{OL}(\mathcal{P})$  is defined as follows

$$\text{enc}_{OL}(\mathcal{P}) \Leftarrow \mathcal{P}_1 \vee \dots \vee \mathcal{P}_k,$$

where

$$\mathcal{P}_i \Leftarrow (\forall \langle \mathcal{s} \rangle. \mathcal{s}(\mathcal{p}_1^{(i)} \vee \dots \vee \mathcal{p}_{n_i}^{(i)})) \wedge (\exists \langle \mathcal{s} \rangle. \mathcal{s}(\mathcal{p}_1^{(i)})) \wedge \dots \wedge (\exists \langle \mathcal{s} \rangle. \mathcal{s}(\mathcal{p}_{n_i}^{(i)})), i \in \{1, \dots, k\}$$

and

$$\text{toDNF}(\mathcal{P}) = (\mathcal{P}^+(\mathcal{p}_1^{(1)}) \otimes \dots \otimes \mathcal{P}^+(\mathcal{p}_{n_1}^{(1)})) \vee \dots \vee (\mathcal{P}^+(\mathcal{p}_1^{(k)}) \otimes \dots \otimes \mathcal{P}^+(\mathcal{p}_{n_k}^{(k)})).$$

Note that replacing all base assertions  $\mathcal{P}^+(\mathcal{p})$  with  $\forall \langle \mathcal{s} \rangle. \mathcal{s}(\mathcal{p})$  would have provided a sufficient encoding. However, we presented this alternative encoding to give a clearer intuition for the later introduced properties (OSL<sub>1</sub>) and (OSL<sub>k</sub>).

4. Let  $\llbracket \mathcal{A} \rrbracket \subseteq \llbracket \left[ \begin{smallmatrix} t \doteq 1 \\ t \doteq 2 \end{smallmatrix} \right] \rrbracket$ ,  $t \notin \text{md}(C)$ . Then  $\models_{RHL} \{\llbracket \mathcal{A} \rrbracket\}_C^C \{\llbracket \mathcal{S} \rrbracket\} \iff \models_{HHL} \{\llbracket \text{enc}_{RHL}(\mathcal{A}, t) \rrbracket\}_C \{\llbracket \text{enc}_{RHL}(\mathcal{S}, t) \rrbracket\}$ , where

$$\text{enc}_{RHL}(\mathcal{A}, t) \Leftarrow \forall \langle \mathcal{s}_1 \rangle. \forall \langle \mathcal{s}_2 \rangle. \mathcal{s}_1(\neg t \doteq 1) \vee \mathcal{s}_2(\neg t \doteq 2) \vee \left[ \begin{smallmatrix} \mathcal{s}_1 \\ \mathcal{s}_2 \end{smallmatrix} \right](\mathcal{A}).$$

*Proof.*

1. Since  $\llbracket \forall \langle \mathcal{s} \rangle. \mathcal{s}(\mathcal{p}) \rrbracket = \{S \mid S \subseteq \llbracket \mathcal{p} \rrbracket\}$  and

$$\begin{aligned} \models_{HHL} \{\{S \mid S \subseteq p\}\}_C \{\{S \mid S \subseteq q\}\} &\iff \forall S \subseteq p. \llbracket C \rrbracket[S] \subseteq q \\ &\iff \forall s \in p. \llbracket C \rrbracket[\{s\}] \subseteq q \\ &\iff \forall s \in p. \forall s'. \langle s, s' \rangle \in \llbracket C \rrbracket \Rightarrow s' \in q \\ &\iff \models_{HL} \{p\}_C \{q\}; \end{aligned}$$

2. Since  $\llbracket \exists \langle \mathcal{s} \rangle. \mathcal{s}(\mathcal{p}) \rrbracket = \{S \mid S \cap \llbracket \mathcal{p} \rrbracket \neq \emptyset\}$  and

$$\begin{aligned} \models_{HHL} \{\{S \mid S \cap p \neq \emptyset\}\}_C \{\{S \mid S \cap q \neq \emptyset\}\} &\iff \forall S. S \cap p \neq \emptyset \Rightarrow \llbracket C \rrbracket[S] \cap q \neq \emptyset \\ &\iff \forall s \in p. \llbracket C \rrbracket[\{s\}] \cap q \neq \emptyset \\ &\iff \forall s \in p. \exists s'. \langle s, s' \rangle \in \llbracket C \rrbracket \wedge s' \in q \\ &\iff \models_{SIL} \{p\}_C \{q\}; \end{aligned}$$

3. Since  $\llbracket \mathcal{P} \rrbracket = \llbracket \text{toDNF}(\mathcal{P}) \rrbracket = \llbracket \text{enc}_{OL}(\mathcal{P}) \rrbracket$  and  $\models_{HHL} \{P\}_C \{Q\} \iff \models_{OL} \{P\}_C \{Q\}$ ;

---

<sup>35</sup>Strictly speaking, this is the (heapless variant of the) syntax of OSL rather than the syntax of OL. However, since our goal is to demonstrate that our logic captures OSL, we present this informal blend of OL and OSL.

4. Since, by definition, we've that for any  $\mathcal{P}$ ,  $\llbracket \forall \langle \mathcal{J}_1 \rangle . \forall \langle \mathcal{J}_2 \rangle . \mathcal{J}_1(\neg t \doteq 1) \vee \mathcal{J}_2(\neg t \doteq 2) \vee \mathcal{P} \rrbracket_\Sigma = \{S \mid \forall s_1 \in S. \forall s_2 \in S. s_1(t) = 1 \Rightarrow s_2(t) = 2 \Rightarrow S \in \llbracket \mathcal{P} \rrbracket_{\Sigma_{\mathcal{J}_1, \mathcal{J}_2}^{s_1, s_2}}\}$ . Therefore,  $\llbracket \text{enc}_{RHL}(\mathcal{A}) \rrbracket = \{S \mid \forall s_1 \in S. \forall s_2 \in S. s_1(t) = 1 \Rightarrow s_2(t) = 2 \Rightarrow \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \in \llbracket \mathcal{A} \rrbracket\}$ . Let's denote the semantic encoding with  $\text{ENC}_{RHL}(\mathbf{r}, t) \Leftarrow \{S \mid \forall s_1 \in S. \forall s_2 \in S. s_1(t) = 1 \Rightarrow s_2(t) = 2 \Rightarrow \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \in \mathbf{r}\}$ . Now, the claim follows since  $\llbracket \text{enc}_{RHL}(\mathcal{A}, t) \rrbracket = \text{ENC}_{RHL}(\llbracket \mathcal{A} \rrbracket, t)$  and

$$\begin{aligned} \models_{HHL} \{\text{ENC}_{RHL}(\mathbf{r}, t)\} C \{\text{ENC}_{RHL}(\mathbf{s}, t)\} &\iff \forall S. S_1 \times S_2 \subseteq \mathbf{r} \Rightarrow (\llbracket C \rrbracket[S])_1 \times (\llbracket C \rrbracket[S])_2 \subseteq \mathbf{s} \\ &\iff \forall S. S_1 \times S_2 \subseteq \mathbf{r} \Rightarrow (\llbracket C \rrbracket[S_1])_1 \times (\llbracket C \rrbracket[S_2])_2 \subseteq \mathbf{s} \\ &\iff \forall \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \in \mathbf{r}. \forall s'_1 \in \llbracket C \rrbracket[\{s_1\}]. \forall s'_2 \in \llbracket C \rrbracket[\{s_2\}]. \begin{bmatrix} s'_1 \\ s'_2 \end{bmatrix} \in \mathbf{s} \\ &\iff \models_{RHL} \{R_p\}^C \{R_q\}, \end{aligned}$$

where  $S_i \Leftarrow \{s \in S \mid s(t) = i\}$ , i.e.  $\text{ENC}_{RHL}(\mathbf{r}) = \{S \mid S_1 \times S_2 \subseteq \mathbf{r}\}$ . □

Inspired by these encodings and by the fact that we aim to design a most general frame rule<sup>36</sup>, it would be ideal if we can obtain the frame rules of SL, SSIL, OSL and RSL directly from our frame rule. That is, ideally

$$\begin{aligned} \text{(SL)} \quad &\llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}) \rrbracket \star \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{F}) \rrbracket = \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P} \star \mathcal{F}) \rrbracket; \\ \text{(SSIL)} \quad &\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}) \rrbracket \star \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{F}) \rrbracket = \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P} \star \mathcal{F}) \rrbracket; \\ \text{(OSL}_1) \quad &\llbracket (\forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P})) \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P})) \rrbracket \star \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{F}) \rrbracket = \llbracket (\forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P} \star \mathcal{F})) \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P} \star \mathcal{F})) \rrbracket; \\ \text{(OSL}_k) \quad &\llbracket (\forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}_1 \vee \dots \vee \mathcal{P}_k)) \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}_1)) \wedge \dots \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}_k)) \rrbracket \star \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{F}) \rrbracket = \\ &\llbracket (\forall \langle \mathcal{J} \rangle . \mathcal{J}((\mathcal{P}_1 \vee \dots \vee \mathcal{P}_k) \star \mathcal{F})) \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}_1 \star \mathcal{F})) \wedge \dots \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}_k \star \mathcal{F})) \rrbracket; \\ \text{(RSL)} \quad &\llbracket \forall \langle \mathcal{J}_1 \rangle . \forall \langle \mathcal{J}_2 \rangle . \mathcal{J}_1(\neg t \doteq 1) \vee \mathcal{J}_2(\neg t \doteq 2) \vee \begin{bmatrix} \mathcal{J}_1 \\ \mathcal{J}_2 \end{bmatrix}(\mathcal{A}) \rrbracket \star \llbracket \forall \langle \mathcal{J}_1 \rangle . \forall \langle \mathcal{J}_2 \rangle . \mathcal{J}_1(\neg t \doteq 1) \vee \mathcal{J}_2(\neg t \doteq 2) \vee \begin{bmatrix} \mathcal{J}_1 \\ \mathcal{J}_2 \end{bmatrix}(\mathcal{F}) \rrbracket = \\ &\llbracket \forall \langle \mathcal{J}_1 \rangle . \forall \langle \mathcal{J}_2 \rangle . \mathcal{J}_1(\neg t \doteq 1) \vee \mathcal{J}_2(\neg t \doteq 2) \vee \begin{bmatrix} \mathcal{J}_1 \\ \mathcal{J}_2 \end{bmatrix}(\mathcal{A} \boxtimes \mathcal{F}) \rrbracket. \end{aligned}$$

Note that one might expect to require

$$\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}) \rrbracket \star \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{F}) \rrbracket = \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P} \star \mathcal{F}) \rrbracket$$

for SSIL,

$$\llbracket (\forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P})) \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P})) \rrbracket \star \llbracket (\forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{F})) \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{F})) \rrbracket = \llbracket (\forall \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P} \star \mathcal{F})) \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P} \star \mathcal{F})) \rrbracket$$

for OSL<sub>1</sub>, and an analogous condition for OSL<sub>k</sub>. However, an important property our star, i.e. separating conjunction, must satisfy is to act like conjunction, but to somehow "split the heap(s)". In particular, when we reason only about the stack, our star must act **exactly** like conjunction, i.e.

$$\begin{aligned} \llbracket (\exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0)) \star (\exists \langle \mathcal{J} \rangle . \mathcal{J}(y \doteq 1)) \rrbracket &= \llbracket (\exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0)) \wedge (\exists \langle \mathcal{J} \rangle . \mathcal{J}(y \doteq 1)) \rrbracket \\ &\supseteq \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0 \wedge y \doteq 1) \rrbracket \\ &= \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0 \star y \doteq 1) \rrbracket. \end{aligned}$$

<sup>36</sup>For non-parallel, non-probabilistic and nondeterministic programs.

So, the property we are looking for SSIL is not  $\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P}) \rrbracket \star \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{L}) \rrbracket = \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\mathcal{P} \star \mathcal{L}) \rrbracket$ . On the other hand, the "one might expect" variant of OSL(1) is reasonable and might work. However, we refrain from designing a frame rule, whose frame is not downward closed, since it does not appear particularly useful for non-concurrent programs and adding such a heap is generally unsound.<sup>37</sup>

$$\frac{\models_{HHL} \{ \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \} \mathbf{div} \{ \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \}}{\not\models_{HHL} \{ \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \} \star \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \} \mathbf{div} \{ \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \} \star \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \}}$$

and

$$\frac{\models_{HHL} \{ \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \} \mathbf{if} \llbracket x \doteq 0 \rrbracket \mathbf{then skip else div fi} \{ \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \}}{\not\models_{HHL} \{ \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \} \star \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(y \doteq 0) \rrbracket \} \mathbf{if} \llbracket x \doteq 0 \rrbracket \mathbf{then skip else div fi} \{ \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \} \star \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(y \doteq 0) \rrbracket \}, \quad (1)}$$

where  $\mathbf{div} \Leftrightarrow \mathbf{while} \llbracket x \doteq x \rrbracket \mathbf{do skip od}$ . The first examples' assumption holds trivially, since the program always diverges and the postcondition is downward closed, i.e. for any  $p$ ,  $\mathbf{sp}(p, \mathbf{div}) = \emptyset$  belongs to any downward closed set, whereas the conclusion fails, since its postcondition no longer contains the empty set. The second example demonstrates that the issue is more general, extending beyond "removal of the empty set". The conclusion of the second example fails for  $S \Leftarrow \{ \langle \mathfrak{s}_y^1, \emptyset \rangle, \langle \mathfrak{s}_x^1, \emptyset \rangle \}$ <sup>38</sup>. Indeed,  $S \in \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \star \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(y \doteq 0) \rrbracket$ , since  $\langle \mathfrak{s}_y^1, \emptyset \rangle$  is a witness for  $\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket$  and  $\langle \mathfrak{s}_x^1, \emptyset \rangle$  is a witness for  $\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(y \doteq 0) \rrbracket$ , but the strongest postcondition

$$\llbracket \mathbf{if} \llbracket x \doteq 0 \rrbracket \mathbf{then skip else div fi} \rrbracket [S] = \{ \langle \mathfrak{s}_y^1, \emptyset \rangle \} \notin \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \doteq 0) \rrbracket \star \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(y \doteq 0) \rrbracket,$$

since there is no witness for  $\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(y \doteq 0) \rrbracket$ . Resolving this issue is left for future work, specifically when developing a concurrent version of HHL.

## 3 Hyper Separation logic

### 3.1 Separating conjunction

#### 3.1.1 Desired properties

We begin by noting that we want our separating conjunction (or star), written  $\star$ , to be commutative, associative and to have an unit element. The rationale for these requirements, beyond their general desirability, is presented in subsection 3.4. Moreover, we formally define what it means for the star to behave exactly like conjunction for (h)assertions that refer only to the stack. We first demonstrate this in the context of SL before specifying it for HSL.

**Definition 3.1.** *We say that an assertion  $p$  is intuitionistic, denoted  $\mathit{intu}(p)$ , iff*

$$\forall \langle s, h_0 \rangle \in p. \forall h \supseteq h_0. \langle s, h \rangle \in p.$$

<sup>37</sup>Recall that  $\star$  has to act like  $\cap$  when reasoning about the stack only.

<sup>38</sup>Recall that  $\mathfrak{s}$  is  $(\lambda x. 0)$ .

Consider the assertions  $\llbracket x \doteq 0 \rrbracket$ ,  $\llbracket \neg x \doteq 0 \rrbracket$ ,  $\llbracket x \mapsto 0 \rrbracket$  and  $\llbracket \neg x \mapsto 0 \rrbracket$ , the first three of which are intuitionistic, i.e. states with larger heaps (and the same stores) are being preserved. This, however, is not true for the last, since  $\langle s_x^5, \emptyset \rangle \in \llbracket \neg x \mapsto 0 \rrbracket$ , but  $\langle s_x^5, \emptyset_5^1 \rangle \notin \llbracket \neg x \mapsto 0 \rrbracket$ , for any  $s$ .

**Lemma 3.2.** *Let  $p, q$  be intuitionistic. Then  $p * q$  is intuitionistic.*

*Proof.* Let  $s, h$  and  $H$  be such that  $\langle s, h \rangle \in p * q$  and  $h \subseteq H$ . By the definition of  $*$  we obtain  $h_p, h_q$  where  $h = h_p \cup h_q$ ,  $h_p \perp h_q$ ,  $\langle s, h_p \rangle \in p$  and  $\langle s, h_q \rangle \in q$ . Let  $H_p \Leftarrow H \upharpoonright \text{Dom}(h_p)$  and  $H_q \Leftarrow H \setminus H_p$ , where  $f \upharpoonright A \Leftarrow f \cap (A \times \text{Rng}(f))$  is the restriction of function  $f$  to the set  $A$ . Then  $h_p \subseteq H_p$  and  $h_q \subseteq H_q$ , hence by the assumption  $\langle s, H_p \rangle \in p$  and  $\langle s, H_q \rangle \in q$ . Moreover,  $H = H_p \cup H_q$  and  $H_p \perp H_q$ , therefore  $\langle s, H \rangle \in p * q$ .  $\square$

**Definition 3.3.** *We say that an assertion  $p$  is pure, denoted  $\text{pure}(p)$ , iff*

$$\forall \langle s, h \rangle \in p. \forall h'. \langle s, h' \rangle \in p.$$

Consider the same 4 assertions  $\llbracket x \doteq 0 \rrbracket$ ,  $\llbracket \neg x \doteq 0 \rrbracket$ ,  $\llbracket x \mapsto 0 \rrbracket$  and  $\llbracket \neg x \mapsto 0 \rrbracket$ . Now, only the first two of them are pure, i.e. they are stack-only dependent. It is evident that if  $\text{pure}(p)$ , then  $\text{intu}(p)$ .

**Lemma 3.4.** *Let  $p, q$  be pure. Then  $p * q$  is pure.*

*Proof.* Let  $s, h$  and  $h'$  be such that  $\langle s, h \rangle \in p * q$ . By the definition of  $*$  we obtain  $h_p, h_q$  where  $\langle s, h_p \rangle \in p$  and  $\langle s, h_q \rangle \in q$ . By the assumption we derive that  $\langle s, h' \rangle \in p$  and  $\langle s, \emptyset \rangle \in q$ . Since  $h' = h' \cup \emptyset$  and  $h' \perp \emptyset$ , we conclude that  $\langle s, h' \rangle \in p * q$ .  $\square$

**Lemma 3.5.** *Let  $\rho \in \text{SynAsrts}_{\mathbb{N}}$  not contain  $\text{emp}, x \mapsto e, x \mapsto \perp$  for any  $x \in \text{PVars}$  and  $e \in \text{AExps}_{\mathbb{N}}$ . Then  $\text{pure}(\llbracket \rho \rrbracket)$ .*

*Proof.* Induction on  $\rho$ :

- Base cases follow trivially;

- $\rho \equiv \rho_1 * \rho_2$

Let  $s, h$  and  $h'$  be such that  $\langle s, h \rangle \in \llbracket \rho_1 * \rho_2 \rrbracket$ . By the definition of  $*$ , we obtain heaps  $h_1, h_2$  such that  $\langle s, h_1 \rangle \in \llbracket \rho_1 \rrbracket$  and  $\langle s, h_2 \rangle \in \llbracket \rho_2 \rrbracket$ . By the i.h., we have that  $\langle s, h' \rangle \in \llbracket \rho_1 \rrbracket$  and  $\langle s, \emptyset \rangle \in \llbracket \rho_2 \rrbracket$ . Therefore, by the definition of  $*$ , we conclude that  $\langle s, h' \rangle \in \llbracket \rho_1 * \rho_2 \rrbracket$ ;

- $\rho \equiv \rho_1 \wedge \rho_2$

Let  $s, h$  and  $h'$  be such that  $\langle s, h \rangle \in \llbracket \rho_1 \wedge \rho_2 \rrbracket = \llbracket \rho_1 \rrbracket \cap \llbracket \rho_2 \rrbracket$ . By the i.h., we obtain  $\langle s, h' \rangle \in \llbracket \rho_1 \rrbracket \cap \llbracket \rho_2 \rrbracket = \llbracket \rho_1 \wedge \rho_2 \rrbracket$ ;

- $\rho \equiv \rho_1 \vee \rho_2$

Let  $s, h$  and  $h'$  be such that  $\langle s, h \rangle \in \llbracket \rho_1 \vee \rho_2 \rrbracket = \llbracket \rho_1 \rrbracket \cup \llbracket \rho_2 \rrbracket$ . By the i.h., considering the two cases  $\langle s, h \rangle \in \llbracket \rho_1 \rrbracket$  and  $\langle s, h \rangle \in \llbracket \rho_2 \rrbracket$ , we obtain  $\langle s, h' \rangle \in \llbracket \rho_1 \rrbracket \cup \llbracket \rho_2 \rrbracket = \llbracket \rho_1 \vee \rho_2 \rrbracket$ ;

- $\rho \equiv \rho_1 \Rightarrow \rho_2$   
Let  $s, h$  and  $h'$  be such that  $\langle s, h \rangle \in \llbracket \rho_1 \Rightarrow \rho_2 \rrbracket$ . Then  $\langle s, h \rangle \in \overline{\llbracket \rho_1 \rrbracket} \cup \llbracket \rho_2 \rrbracket$ . Consider the case  $\langle s, h \rangle \in \overline{\llbracket \rho_1 \rrbracket}$ . Then  $\langle s, h \rangle \notin \llbracket \rho_1 \rrbracket$  and by the i.h.,  $\langle s, h' \rangle \notin \llbracket \rho_1 \rrbracket$ , i.e.  $\langle s, h' \rangle \in \overline{\llbracket \rho_1 \rrbracket}$ . The other case follows directly by the i.h. Therefore  $\langle s, h' \rangle \in \llbracket \rho_1 \Rightarrow \rho_2 \rrbracket$ ;
- $\rho \equiv \rho_1 \Leftrightarrow \rho_2$   
Let  $s, h$  and  $h'$  be such that  $\langle s, h \rangle \in \llbracket \rho_1 \Leftrightarrow \rho_2 \rrbracket$ . Then  $\langle s, h \rangle \in (\llbracket \rho_1 \rrbracket \cap \llbracket \rho_2 \rrbracket) \cup (\overline{\llbracket \rho_1 \rrbracket} \cap \overline{\llbracket \rho_2 \rrbracket})$ . Analogously, considering two the cases  $\langle s, h \rangle \in \llbracket \rho_1 \rrbracket \cap \llbracket \rho_2 \rrbracket$  and  $\langle s, h \rangle \in \overline{\llbracket \rho_1 \rrbracket} \cap \overline{\llbracket \rho_2 \rrbracket}$ ;
- $\rho \equiv \neg \rho_0$  - Analogously;
- $\rho \equiv \exists x. \rho_0$   
Let  $s, h$  and  $h'$  be such that  $\langle s, h \rangle \in \llbracket \exists x. \rho_0 \rrbracket$ . Then let  $n$  be such that  $\langle s_x^n, h \rangle \in \llbracket \rho_0 \rrbracket$ . Now, by the i.h.,  $\langle s_x^n, h' \rangle \in \llbracket \rho_0 \rrbracket$  and hence  $\langle s, h \rangle \in \llbracket \exists x. \rho_0 \rrbracket$ ;
- $\rho \equiv \forall x. \rho_0$  - Analogously.

□

**Lemma 3.6.** *Let  $p, q$  be assertions such that  $\text{intu}(p)$  and  $\text{pure}(q)$ . Then  $p * q = p \cap q$ .*

*Proof.* We prove both inclusions individually:

- $p * q \subseteq p \cap q$   
Let  $\langle s, h \rangle \in p * q$ . By the definition of  $*$ , we obtain  $h_p, h_q$  such that  $h = h_p \cup h_q$ ,  $h_p \perp h_q$ ,  $\langle s, h_p \rangle \in p$  and  $\langle s, h_q \rangle \in q$ . Moreover,  $h_p \subseteq h$  since  $h_p \perp h_q$ . Therefore, using  $\langle s, h_p \rangle \in p$  and  $\text{intu}(p)$ , we obtain that  $\langle s, h \rangle \in p$ . Now, since  $\langle s, h_q \rangle \in q$  and by the pureness of  $q$ , we obtain that  $\langle s, h \rangle \in q$ . Therefore  $\langle s, h \rangle \in p \cap q$ .
- $p \cap q \subseteq p * q$   
Let  $\langle s, h \rangle \in p \cap q$ . Then  $\langle s, \emptyset \rangle \in q$  by the pureness of  $q$ . Therefore, by the definition of  $*$ , we conclude that  $\langle s, h \rangle \in p * q$ .

□

**Corollary 3.7.** *Let  $p, q$  be assertions such that  $\text{pure}(p)$  and  $\text{pure}(q)$ . Then  $p * q = p \cap q$ .*

*Proof.* Follows directly from lemma 3.6 and the fact that  $\forall p. \text{pure}(p) \Rightarrow \text{intu}(p)$ .

□

We will now define a notion of intuitionistic and pureness for hassertions and formally state the requirements for our star: it should preserve intuitionistic and pureness and coincide with conjunction when one argument is intuitionistic and the other is pure.

The syntax for  $\mathbb{N}$ -hassertions is not yet clear, but we aim to be something similar to that of heapless  $\mathbb{N}$ -hassertions. For example, we want to include (at least)  $\forall\langle\mathcal{J}\rangle.\mathcal{J}(\mathcal{P})$  and  $\exists\langle\mathcal{J}\rangle.\mathcal{J}(\mathcal{P})$ , where  $\mathcal{P} \in \text{SynSArts}_{\mathbb{N}}$ . Note that  $\text{pure}(\llbracket\mathcal{P}\rrbracket)$ , since  $\mathcal{P} \in \text{SynSArts}_{\mathbb{N}}$ , i.e. it depends only on the stack. We want to define a notion of pureness, such that  $\text{Pure}(\llbracket\forall\langle\mathcal{J}\rangle.\mathcal{J}(\mathcal{P})\rrbracket)$  and  $\text{Pure}(\llbracket\exists\langle\mathcal{J}\rangle.\mathcal{J}(\mathcal{P})\rrbracket)$ , where  $\text{pure}(\llbracket\mathcal{P}\rrbracket)$  and similarly for intuitionistic. Semantically, we require that if  $p$  is pure (intuitionistic), then  $\{S \mid S \subseteq p\}$  and  $\{S \mid S \cap p \neq \emptyset\}$  are also pure (intuitionistic).

**Definition 3.8.** We say that a hassertion  $P$  is pure, denoted  $\text{Pure}(P)$ , iff

$$\forall S \in P. \forall S'. ((\forall\langle s, h \rangle \in S. \exists h'. \langle s, h' \rangle \in S') \wedge (\forall\langle s, h' \rangle \in S'. \exists h. \langle s, h \rangle \in S)) \Rightarrow S' \in P.$$

**Lemma 3.9.** Let  $p$  be pure. Then  $\{S \mid S \subseteq p\}$  and  $\{S \mid S \cap p \neq \emptyset\}$  are pure.

*Proof.* We show individually:

- $\text{Pure}(\{S \mid S \subseteq p\})$

Let  $S$  and  $S'$  be such that  $S \in \{S \mid S \subseteq p\}$  and  $\forall\langle s, h' \rangle \in S'. \exists h. \langle s, h \rangle \in S$ . Then, since  $S \subseteq p$ , it follows that  $\forall\langle s, h' \rangle \in S'. \exists h. \langle s, h \rangle \in p$ . Now, since  $\text{pure}(p)$ , we obtain that  $\forall\langle s, h' \rangle \in S'. \langle s, h' \rangle \in p$ . Therefore  $S' \in \{S \mid S \subseteq p\}$ .

- $\text{Pure}(\{S \mid S \cap p \neq \emptyset\})$

Let  $S$  and  $S'$  be such that  $S \in \{S \mid S \cap p \neq \emptyset\}$  and  $\forall\langle s, h \rangle \in S. \exists h'. \langle s, h' \rangle \in S'$ . Let  $s$  and  $h$  be such that  $\langle s, h \rangle \in S$  and  $\langle s, h \rangle \in p$ . Now, let  $h'$  be such that  $\langle s, h' \rangle \in S'$ . By the pureness of  $p$  and that  $\langle s, h \rangle \in p$ , it follows that  $\langle s, h' \rangle \in p$ . Therefore  $\langle s, h' \rangle \in S' \cap p$ , hence  $S' \in \{S \mid S \cap p \neq \emptyset\}$ .

□

**Definition 3.10.** We say that a hassertion  $P$  is intuitionistic, denoted  $\text{Intu}(P)$ , iff

$$\forall S \in P. \forall S'. ((\forall\langle s, h \rangle \in S. \exists h' \supseteq h. \langle s, h' \rangle \in S') \wedge (\forall\langle s, h' \rangle \in S'. \exists h \subseteq h'. \langle s, h \rangle \in S)) \Rightarrow S' \in P.$$

**Lemma 3.11.** Let  $p$  be intuitionistic. Then  $\{S \mid S \subseteq p\}$  and  $\{S \mid S \cap p \neq \emptyset\}$  are intuitionistic.

*Proof.* We show individually:

- $\text{Intu}(\{S \mid S \subseteq p\})$

Let  $S$  and  $S'$  be such that  $S \in \{S \mid S \subseteq p\}$  and  $\forall\langle s, h' \rangle \in S'. \exists h \subseteq h'. \langle s, h \rangle \in S$ . Then, since  $S \subseteq p$ , it follows that  $\forall\langle s, h' \rangle \in S'. \exists h \subseteq h'. \langle s, h \rangle \in p$ . Now, since  $\text{intu}(p)$ , we obtain that  $\forall\langle s, h' \rangle \in S'. \langle s, h' \rangle \in p$ . Therefore  $S' \in \{S \mid S \subseteq p\}$ .

- $\text{Intu}(\{S \mid S \cap p \neq \emptyset\})$

Let  $S$  and  $S'$  be such that  $S \in \{S \mid S \cap p \neq \emptyset\}$  and  $\forall \langle s, h \rangle \in S. \exists h' \supseteq h. \langle s, h' \rangle \in S'$ . Let  $s$  and  $h$  be such that  $\langle s, h \rangle \in S$  and  $\langle s, h \rangle \in p$ . Now, let  $h'$  be such that  $h \subseteq h'$  and  $\langle s, h' \rangle \in S'$ . By the intuitionisticity of  $p$ ,  $\langle s, h \rangle \in p$  and  $h \subseteq h'$ , it follows that  $\langle s, h' \rangle \in p$ . Therefore  $\langle s, h' \rangle \in S' \cap p$ , hence  $S' \in \{S \mid S \cap p \neq \emptyset\}$ .

□

To summarize, our goal is to design a star,  $\star$ , which satisfies the following properties:

1. **(unit)**  $\exists U. \forall P. P \star U = P$
2. **(commutativity)**  $P \star Q = Q \star P$
3. **(associativity)**  $(P \star Q) \star R = P \star (Q \star R)$
4. **(monotonicity)**  $P \subseteq P' \Rightarrow Q \subseteq Q' \Rightarrow P \star Q \subseteq P' \star Q'$
5. **( $\cup$ -distributivity)**  $(P \cup Q) \star F = (P \star F) \cup (Q \star F)$
6. **( $\bigcup$ -distributivity)**  $(\bigcup_{P \in X} P) \star F = \bigcup_{P \in X} (P \star F)$
7. **( $\otimes$ -distributivity)**  $(P \otimes Q) \star F = (P \star F) \otimes (Q \star F)$
8. **( $\bigotimes$ -distributivity)**  $(\bigotimes_{P \in X} P) \star F = \bigotimes_{P \in X} (P \star F)$
9. **(pure-intersection)**  $\text{Intu}(P) \Rightarrow \text{Pure}(Q) \Rightarrow P \star Q = P \cap Q$
10. **(Intu-preserving)**  $\text{Intu}(P) \Rightarrow \text{Intu}(Q) \Rightarrow \text{Intu}(P \star Q)$
11. **(Pure-preserving)**  $\text{Pure}(P) \Rightarrow \text{Pure}(Q) \Rightarrow \text{Pure}(P \star Q)$
12. **(operational-coherence)**  $\text{SP}(P \star F, C) = \text{SP}(P, C) \star F$ ,  
for appropriate frames  $F$  and definition of  $\text{SP}$
13. **(FR-SL)**  $\{S \mid S \subseteq p\} \star \{S \mid S \subseteq f\} = \{S \mid S \subseteq p \star f\}$
14. **(FR-SSIL)**  $\{S \mid S \cap p \neq \emptyset\} \star \{S \mid S \subseteq f\} = \{S \mid S \cap (p \star f) \neq \emptyset\}$
15. **(FR-OSL-1)**  $\{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\} \star \{S \mid S \subseteq f\} = \{S \mid S \subseteq p \star f \wedge S \cap (p \star f) \neq \emptyset\}$
16. **(FR-OSL-2)**  $\{S \mid S \subseteq p_1 \cup p_2 \wedge S \cap p_1 \neq \emptyset \wedge S \cap p_2 \neq \emptyset\} \star \{S \mid S \subseteq f\} =$   
 $\{S \mid S \subseteq (p_1 \cup p_2) \star f \wedge S \cap (p_1 \star f) \neq \emptyset \wedge S \cap (p_2 \star f) \neq \emptyset\}$ <sup>39</sup>
17. **(FR-RSL)**  $\{S \mid S_1 \times S_2 \subseteq \mathbb{R}\} \star \{S \mid S_1 \times S_2 \subseteq \mathbb{F}\} = \{S \mid S_1 \times S_2 \subseteq \mathbb{R} \boxtimes \mathbb{F}\}$ ,  
where  $S_i \Leftarrow \{\langle s, h \rangle \in S \mid s(t) = i\}$ .

<sup>39</sup>Note that  $\star$  distributes over  $\cup$ .



The first three properties are used in subsection 3.4 and are generally desirable. The fourth is necessary, for instance, to establish the fifth property. Properties 5–6 are standard in separation logics, while properties 7–8 are inspired by OSL and appear essential when generalizing a separation logic for hassertions. Properties 9–11 ensure that the star operator behaves like ordinary conjunction for stack-only dependent hassertions. Property 12 is the most indicative, capturing the expected behavior of the star. Finally, properties 13–17 are needed to formulate the frame rules of SL, SSIL, OSL, and RSL solely through our frame rule<sup>40</sup>.

We regard property 9 as a fundamental design requirement—it must hold. After all, the operation is called separating conjunction, so it should behave like a conjunction for pure hassertions, or more precisely, for one pure hassertion and one intuitionistic one. This immediately implies that properties 7 (and its general form, property 8) and 14 cannot hold, since they are incompatible with property 9:

- Property 7

- $F$  has to be downward closed

Let  $P \Leftarrow \llbracket (\forall \langle s \rangle. s(x \dot{=} 0)) \wedge (\exists \langle s \rangle. s(x \dot{=} 0)) \rrbracket$ ,  $Q \Leftarrow \llbracket (\forall \langle s \rangle. s(x \dot{=} 1)) \wedge (\exists \langle s \rangle. s(x \dot{=} 1)) \rrbracket$  and  $F \Leftarrow \llbracket (\exists \langle s \rangle. s(x \dot{=} 0)) \wedge (\exists \langle s \rangle. s(x \dot{=} 1)) \rrbracket$ . Then

$$(P \otimes Q) \cap F = P \otimes Q = \llbracket (\forall \langle s \rangle. s(x \dot{=} 0 \vee x \dot{=} 1)) \wedge (\exists \langle s \rangle. s(x \dot{=} 0)) \wedge (\exists \langle s \rangle. s(x \dot{=} 1)) \rrbracket,$$

whereas

$$(P \cap F) \otimes (Q \cap F) = \emptyset \otimes \emptyset = \emptyset.$$

This issue could be resolved by ensuring that  $F$  is downward closed.

- $F$  has to be union closed

Let  $P \Leftarrow \llbracket \forall \langle s \rangle. s(x \dot{=} 0) \rrbracket$ ,  $Q \Leftarrow \llbracket \forall \langle s \rangle. s(x \dot{=} 1) \rrbracket$  and  $F \Leftarrow \llbracket \forall \langle s_1 \rangle. \langle s_2 \rangle. \left[ \begin{smallmatrix} s_1 \\ s_2 \end{smallmatrix} \right] (\exists y. \left[ \begin{smallmatrix} x \dot{=} y \\ x \dot{=} y \end{smallmatrix} \right]) \rrbracket$ . Then

$$(P \cap F) \otimes (Q \cap F) = P \otimes Q = \llbracket \forall \langle s \rangle. s(x \dot{=} 0 \vee x \dot{=} 1) \rrbracket,$$

whereas

$$(P \otimes Q) \cap F = \llbracket \forall \langle s \rangle. s(x \dot{=} 0 \vee x \dot{=} 1) \rrbracket \cap F \subset \llbracket \forall \langle s \rangle. s(x \dot{=} 0 \vee x \dot{=} 1) \rrbracket,$$

since  $\{s_x^0, s_x^1\} \in \llbracket \forall \langle s \rangle. s(x \dot{=} 0 \vee x \dot{=} 1) \rrbracket \setminus F$ , for any  $s$ . This issue could be resolved by ensuring that  $F$  is union closed, i.e.  $\forall S \in F. \forall S' \in F. S \cup S' \in F$ .

- Property 14

Let  $p \Leftarrow \llbracket x \dot{=} 0 \rrbracket$  and  $f \Leftarrow \llbracket x \dot{=} 0 \rrbracket$ . Then

$$\{S \mid S \cap p \neq \emptyset\} \cap \{S \mid S \subseteq f\} = \{S \mid S \subseteq \llbracket x \dot{=} 0 \rrbracket \wedge S \cap \llbracket x \dot{=} 0 \rrbracket \neq \emptyset\} \subset \{S \mid S \cap (p \cap f) \neq \emptyset\},$$

<sup>40</sup>Later, we'll see that some of these properties don't hold, and we'll explore the implications of that.

since the right side permits states with  $x \neq 0$ , while the left doesn't. In subsection 3.4 we argue that this direction is sufficient.

Recall the definition

$$p * q \Leftrightarrow \{\langle s, h_p \cup h_q \rangle \mid \langle s, h_p \rangle \in p \wedge \langle s, h_q \rangle \in q \wedge h_p \perp h_q\}.$$

Note that the definition of  $p * q$  operates in a pointwise manner, combining individual elements from  $p$  and  $q$  whenever their heaps are disjoint. It does not impose any additional structure on  $p$  and  $q$  as wholes but instead lifts the heap union operation to compatible pairs. This locality is precisely why  $*$  satisfies monotonicity (property 4), as extending either argument with additional elements naturally extends their combinations. In fact, this is the behavior we want for our separating conjunction, rather than monotonicity itself being the goal. We claim that property 17 cannot be satisfied by any such pointwise/local  $\star$ :

- Property 17

– Semantically

Let  $R \Leftrightarrow \{\llbracket \langle s_t^1, h_1 \rangle \rrbracket, \llbracket \langle s_t^1, h_2 \rangle \rrbracket\}$  and  $S \Leftrightarrow \{\llbracket \langle s_t^1, \emptyset \rangle \rrbracket, \llbracket \langle s_t^1, h_1 \rangle \rrbracket\}$  for some  $s$  and non-empty  $h_1, h_2$  such that  $h_1 \perp h_2$ . Then

$$R \boxtimes S = \{\llbracket \langle s_t^1, h_1 \rangle \rrbracket, \llbracket \langle s_t^1, h_2 \rangle \rrbracket\}.$$

Note how  $R \boxtimes S$  is "rectangular", i.e. it's a cartesian product of  $\{\langle s_t^1, h_1 \rangle, \langle s_t^1, h_2 \rangle\}$  and  $\{\langle s_t^2, h_1 \cup h_2 \rangle\}$ , whereas  $R$  is not rectangular. The failure of property 17 stems from the ability to derive a rectangular rassertion from a non-rectangular one. More precisely, let  $S \Leftrightarrow \{\langle s_t^1, h_1 \rangle, \langle s_t^1, h_2 \rangle, \langle s_t^2, h_1 \cup h_2 \rangle\} \in \{S \mid S_1 \times S_2 \subseteq R \boxtimes S\}$ . Since we want  $\star$  to be pointwise, we want  $S_R$  and  $S_S$  such that  $S_R \in \{S \mid S_1 \times S_2 \subseteq R\}$  and  $S_S \in \{S \mid S_1 \times S_2 \subseteq s\}$  and that they somehow combine up to  $S$ . Note that each element of  $\{S \mid S_1 \times S_2 \subseteq R\}$  represents a rectangular subrelation of  $R$ <sup>41</sup>. That is,  $\emptyset$  represents  $\emptyset \subseteq R$ <sup>42</sup>,  $\{\langle s_t^1, h_1 \rangle, \langle s_t^2, h_1 \rangle\}$  represents  $\{\llbracket \langle s_t^1, h_1 \rangle \rrbracket\} \subseteq R$  and  $\{\langle s_t^1, h_2 \rangle, \langle s_t^2, h_2 \rangle\}$  represents  $\{\llbracket \langle s_t^1, h_2 \rangle \rrbracket\} \subseteq R$ . There is no way to represent the whole  $R$ , since it is not rectangular. Thus, we are missing either of the two elements of  $R$ , whereas both are required to obtain  $R \boxtimes S$ , represented by  $S$ . Therefore, we can hope for a star, satisfying at most  $\{S \mid S_1 \times S_2 \subseteq R\} \star \{S \mid S_1 \times S_2 \subseteq F\} \subseteq \{S \mid S_1 \times S_2 \subseteq R \boxtimes F\}$ . In subsection 3.4 we argue that this direction is sufficient.

– Syntactically

<sup>41</sup>The elements with  $t \notin \{1, 2\}$  are irrelevant, so we consider only the cases with no such elements.

<sup>42</sup>Any set  $S$  with  $S_1 = \emptyset \vee S_2 = \emptyset$  also represents  $\emptyset \subseteq R$ .

Now we demonstrate that this issue can arise with semantic objects, expressible in our syntax. Let  $R \Leftarrow \llbracket ([\frac{z \doteq 0}{x \mapsto 0}] \vee [\frac{z \doteq 1}{y \mapsto 1}]) \wedge [\frac{t \doteq 1}{t \doteq 2}] \rrbracket$  and  $S \Leftarrow \llbracket ([\frac{z \doteq 0}{y \mapsto 1}] \vee [\frac{z \doteq 1}{x \mapsto 0}]) \wedge [\frac{t \doteq 1}{t \doteq 2}] \rrbracket$ . Then

$$\left\{ \llbracket \langle s_{z,t}^{0,1}, \emptyset \rangle \rrbracket, \llbracket \langle s_{x,y,t}^{1,2,2}, \emptyset_{1,2}^0 \rangle \rrbracket \right\} \in R \boxtimes S,$$

where the witnesses are

$$\left\{ \llbracket \langle s_{z,t}^{0,1}, \emptyset \rangle \rrbracket, \llbracket \langle s_{x,y,t}^{1,2,2}, \emptyset_1^0 \rangle \rrbracket \right\} \in R$$

and

$$\left\{ \llbracket \langle s_{z,t}^{0,1}, \emptyset \rangle \rrbracket, \llbracket \langle s_{x,y,t}^{1,2,2}, \emptyset_2^1 \rangle \rrbracket \right\} \in S.$$

Again, the result is rectangular, whereas the witnesses are non-rectangular. Moreover, there are no such rectangular witnesses.

To summarize, we weaken properties 7, 8, 14 and 17 as follows:

7. ( $\otimes$ -distributivity)  $(P \otimes Q) \star \{S \mid S \subseteq f\} = (P \star \{S \mid S \subseteq f\}) \otimes (Q \star \{S \mid S \subseteq f\})$
8. ( $\otimes$ -distributivity)  $(\bigotimes_{P \in X} P) \star \{S \mid S \subseteq f\} = \bigotimes_{P \in X} (P \star \{S \mid S \subseteq f\})$
14. (**FR-SSIL**)  $\{S \mid S \cap p \neq \emptyset\} \star \{S \mid S \subseteq f\} \subseteq \{S \mid S \cap (p * f) \neq \emptyset\}$
17. (**FR-RSL**)  $\{S \mid S_1 \times S_2 \subseteq R\} \star \{S \mid S_1 \times S_2 \subseteq F\} = \{S \mid S_1 \times S_2 \subseteq R \boxtimes F\}$

Note that  $\{S \mid S \subseteq f\}$  is both downward closed and closed under union. However, not every family  $F$  with these closure properties admits a witness  $f$  such that  $F = \{S \mid S \subseteq f\}$ . For instance, this fails for  $F = \{\emptyset, \{0\}, \{0, 1\}, \{0, 1, 2\} \dots\}$ . If, however,  $F$  is also closed under **infinite** union, then such  $f$  exists, namely  $f \Leftarrow \bigcup F$ . For property 7, finite union closure suffices (which we establish), but we present this weaker condition for simplicity. In contrast, for property 8, infinite union closure is necessary.

### 3.1.2 Definition

The naive definition of our star, denoted  $\star_{\text{naive}}$ , given by

$$P \star_{\text{naive}} Q \Leftarrow \{S \mid S \subseteq S_p * S_q \wedge S_p \in P \wedge S_q \in Q\},$$

fails, since it considers only the "universal part", i.e.

$$\{S \mid S \subseteq p\} \star_{\text{naive}} \{S \mid S \subseteq q\} = \{S \mid S \subseteq p * q\},$$

but doesn't consider the "existential part", i.e.

$$\{S \mid S \cap p \neq \emptyset\} \star_{\text{naive}} \{S \mid S \subseteq q\} \not\subseteq \{S \mid S \cap (p \cap q) \neq \emptyset\}.$$

That's because if  $\{S \mid S \cap p \neq \emptyset\} \star_{\text{naive}} \{S \mid S \subseteq q\} \neq \emptyset$ , then we have  $\emptyset \in \{S \mid S \cap p \neq \emptyset\} \star_{\text{naive}} \{S \mid S \subseteq q\}$ . That is, we "lose" the witness of  $S \cap p \neq \emptyset$ .

One might suspect that this issue arises solely because we allow all possible subsets, including those that have "lost" the witness (of  $S \cap p \neq \emptyset$ ). However, even modifying the definition to

$$P \star'_{\text{naive}} Q \Leftarrow \{S_p * S_q \mid S_p \in P \wedge S_q \in Q\},$$

does not resolve the problem. This definition, like the original, considers only the "universal part", i.e.  $\{S \mid S \subseteq p\} \star'_{\text{naive}} \{S \mid S \subseteq q\} \subseteq \{S \mid S \subseteq p * q\}$ <sup>43</sup>, but doesn't consider the "existential part", i.e.

$$\{S \mid S \cap p \neq \emptyset\} \star'_{\text{naive}} \{S \mid S \subseteq q\} \not\subseteq \{S \mid S \cap (p \cap q) \neq \emptyset\}.$$

Indeed, consider

$$\underbrace{\{\langle s_x^0, \emptyset \rangle, \langle s_{x,y}^1, \emptyset \rangle\}}_{\text{witness}} \in \{S \mid S \cap \llbracket x \doteq o \rrbracket\}$$

and

$$\{\langle s_{x,y}^1, \emptyset \rangle\} \in \{S \mid S \subseteq \llbracket y \doteq o \rrbracket\}.$$

Then

$$\{\langle s_{x,y}^1, \emptyset \rangle\} = \underbrace{\{\langle s_x^0, \emptyset \rangle, \langle s_{x,y}^1, \emptyset \rangle\}}_{\text{witness}} * \{\langle s_{x,y}^1, \emptyset \rangle\} \in \{S \mid S \cap \llbracket x \doteq o \rrbracket\} \star'_{\text{naive}} \{S \mid S \subseteq \llbracket y \doteq o \rrbracket\}.$$

Thus,  $\star'_{\text{naive}}$  once again fails to preserve the witness.

The solution we propose explicitly enforces the preservation of witnesses using the predicates

$$\text{left\_lives}(S, S_1, S_2) \stackrel{\text{def}}{\iff} \forall \sigma_1 \in S_1. \exists \sigma \in S. \exists \sigma_2 \in S_2. \text{hAdd}(\sigma, \sigma_1, \sigma_2)$$

and

$$\text{right\_lives}(S, S_1, S_2) \stackrel{\text{def}}{\iff} \text{left\_lives}(S, S_2, S_1),$$

where

$$\text{hAdd}(\langle s, h \rangle, \langle s_1, h_1 \rangle, \langle s_2, h_2 \rangle) \stackrel{\text{def}}{\iff} s = s_1 \wedge s_2 \wedge h = h_1 \cup h_2 \wedge h_1 \perp h_2.$$

Now, two natural definitions emerge:

1.  $P \star Q \stackrel{?}{\iff} \{S \mid S = S_P * S_Q \wedge S_P \in P \wedge S_Q \in Q \wedge \text{left\_lives}(S, S_P, S_Q) \wedge \text{right\_lives}(S, S_P, S_Q)\};$
2.  $P \star Q \stackrel{?}{\iff} \{S \mid S \subseteq S_P * S_Q \wedge S_P \in P \wedge S_Q \in Q \wedge \text{left\_lives}(S, S_P, S_Q) \wedge \text{right\_lives}(S, S_P, S_Q)\}.$

---

<sup>43</sup>Note that we changed  $=$  to  $\subseteq$ . Turns out, having enough of the other properties,  $\subseteq$  is actually sufficient to express the frame rule of SL, since we can generate all singletons  $\{\sigma\} \subseteq p * q$ , which can be then combined into arbitrary subset of  $p * q$  with the yet not introduced (Idx-Join) rule. This idea is explored in 3.4.2 in detail.

The first definition seems more natural, as it more closely mirrors the definition of  $*$ . However, it fails to satisfy associativity<sup>44</sup>, even though it seems to satisfy almost all other properties (or their slightly weaker variants). Specifically, due to its rectangular design (i.e., its resemblance to the Cartesian product), it satisfies at most one direction of properties 7<sup>45</sup>, 8, 13, 15, and 16:

- 7. ( **$\otimes$ -distributivity**)  $(P \otimes Q) \star F \subseteq (P \star F) \otimes (Q \star F)$ ,  $F$  - downward closed
- 8. ( **$\otimes$ -distributivity**)  $(\bigotimes_{P \in X} P) \star F \subseteq \bigotimes_{P \in X} (P \star F)$ ,  $F$  - downward closed
- 13. (**FR-SL**)  $\{S \mid S \subseteq p\} \star \{S \mid S \subseteq f\} \subseteq \{S \mid S \subseteq p \star f\}$
- 15. (**FR-OSL-1**)  $\{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\} \star \{S \mid S \subseteq f\} \subseteq \{S \mid S \subseteq p \star f \wedge S \cap (p \star f) \neq \emptyset\}$
- 16. (**FR-OSL-2**)  $\{S \mid S \subseteq p_1 \cup p_2 \wedge S \cap p_1 \neq \emptyset \wedge S \cap p_2 \neq \emptyset\} \star \{S \mid S \subseteq f\} \subseteq \{S \mid S \subseteq (p_1 \cup p_2) \star f \wedge S \cap (p_1 \star f) \neq \emptyset \wedge S \cap (p_2 \star f) \neq \emptyset\}$

We believe these weaker variants would still suffice to express the four logics and yield a reasonable logic overall—but that is clearly not the case, as associativity is missing.

Therefore, we adopt the second option:

$$P \star Q \equiv \{S \mid S \subseteq S_P \star S_Q \wedge S_P \in P \wedge S_Q \in Q \wedge \text{left\_lives}(S, S_P, S_Q) \wedge \text{right\_lives}(S, S_P, S_Q)\}.$$

**Lemma 3.12.** *The following provides an equivalent characterization of  $\star$ :*

$$P \star Q = \{S \mid \exists \tau \in \text{States}^3. (\forall \langle \sigma, \sigma_P, \sigma_Q \rangle \in \tau. \text{hAdd}(\sigma, \sigma_P, \sigma_Q)) \wedge \pi_1^3(\tau) = S \wedge \pi_2^3(\tau) \in P \wedge \pi_3^3(\tau) \in Q\},$$

where  $\pi_i^n(X) \equiv \{s_i \mid \langle \sigma_1, \dots, \sigma_n \rangle \in X\}$  is the  $i^{\text{th}}$  projection of  $X$  for  $n \in \mathbb{N}, i \in \{1, \dots, n\}$ .

*Proof.* See lemma `hstar_via_triples` in `HyperStar.thy`. □

Now, we present sketch proofs for all properties except 12, which will be addressed in subsection 3.3. For formal proofs of these properties, see theorem `hstar_properties` in `HyperStar.thy`.

1. (**unit**)  $\exists U. \forall P. P \star U = P$

Let  $U \equiv \{S \mid S \subseteq \{\langle s, \emptyset \rangle \mid s \in \text{Stacks}\}\}$ . Then for each  $P$ ,  $P \star U = P$ . The key observation is that  $\text{hAdd}(\langle s, h \rangle, \langle s, h \rangle, \langle s, \emptyset \rangle)$ .

- $P \star U \subseteq P$

Let  $S \in P \star U$ . Then obtain  $S_P \in P$  and  $S_U \in U$  such that  $S \subseteq S_P \star S_U$  and  $\text{left\_lives}(S, S_P, S_U)$ . Therefore  $S = S_P$  and hence  $S \in P$ .

<sup>44</sup>Let  $P \equiv \{\{\langle s, \emptyset_1^1 \rangle, \langle s, \emptyset_2^2 \rangle\}\}$ ,  $Q \equiv \{\{\langle s, \emptyset_1^1 \rangle, \langle s, \emptyset_3^3 \rangle\}\}$  and  $R \equiv \{\{\langle s, \emptyset_2^2 \rangle, \langle s, \emptyset_4^4 \rangle\}\}$ . Then  $P \star (Q \star R) = \emptyset$ , whereas  $(P \star Q) \star R = \{\langle s, \emptyset_{1,2,3}^{1,2,3} \rangle, \langle s, \emptyset_{1,2,4}^{1,2,4} \rangle, \langle s, \emptyset_{1,3,4}^{1,3,4} \rangle, \langle s, \emptyset_{2,3,4}^{2,3,4} \rangle\}$ .

<sup>45</sup>Let  $P \equiv \{\{\langle s, \emptyset_1^1 \rangle\}\}$ ,  $Q \equiv \{\{\langle s, \emptyset_2^2 \rangle\}\}$  and  $F \equiv \{S \mid S \subseteq \{\langle s, \emptyset_3^3 \rangle, \langle s, \emptyset_4^4 \rangle\}\}$ . Then  $(P \otimes Q) \star F \subset (P \star F) \otimes (Q \star F)$ .

- $P \subseteq P \star U$

Let  $S \in P$ . Now, define  $S_U \triangleq \{\langle s, \emptyset \rangle \mid \exists h. \langle s, h \rangle \in S\} \in U$ . Now,  $S \subseteq S \star S_U$ ,  $\text{left\_lives}(S, S, S_U)$  and  $\text{right\_lives}(S, S, S_U)$ . Therefore  $S \in P \star U$ .

2. **(commutativity)**  $P \star Q = Q \star P$

Immediately follows, since  $\star$  is commutative and the definition of  $\text{right\_lives}$ .

3. **(associativity)**  $(P \star Q) \star R = P \star (Q \star R)$

Unfolding twice the left side, using lemma 3.12, we obtain

$$\begin{aligned} (P \star Q) \star R &= \{S \mid \exists \tau, \tau_{PQ} \in \text{States}^3. \\ &\quad (\forall \langle \sigma_{PQR}, \sigma_{PQ}, \sigma_R \rangle \in \tau. \text{hAdd}(\sigma_{PQR}, \sigma_{PQ}, \sigma_R)) \wedge \\ &\quad (\forall \langle \sigma_{PQ}, \sigma_P, \sigma_Q \rangle \in \tau_{PQ}. \text{hAdd}(\sigma_{PQ}, \sigma_P, \sigma_Q)) \wedge \\ &\quad \pi_1^3(\tau) = S \wedge \pi_1^3(\tau_{PQ}) = \pi_2^3(\tau) \wedge \\ &\quad \pi_2^3(\tau_{PQ}) \in P \wedge \pi_3^3(\tau_{PQ}) \in Q \wedge \pi_3^3(\tau) \in R\}. \end{aligned}$$

Building on the connection between the triples, namely  $\pi_1^3(\tau_{PQ}) = \pi_2^3(\tau)$ , we can express it more concisely by combining the triples into a single quadruple, leading to an equivalent formulation as follows:

$$\begin{aligned} (P \star Q) \star R &= \{S \mid \exists \tau \in \text{States}^4. \\ &\quad (\forall \langle \sigma_{PQR}, \sigma_P, \sigma_Q, \sigma_R \rangle \in \tau. \exists \sigma_{PQ}. \text{hAdd}(\sigma_{PQR}, \sigma_{PQ}, \sigma_R) \wedge \text{hAdd}(\sigma_{PQ}, \sigma_P, \sigma_Q)) \wedge \\ &\quad \pi_1^4(\tau) = S \wedge \pi_2^4(\tau) \in P \wedge \pi_3^4(\tau) \in Q \wedge \pi_4^4(\tau) \in R\}. \end{aligned}$$

Analogously,

$$\begin{aligned} P \star (Q \star R) &= \{S \mid \exists \tau \in \text{States}^4. \\ &\quad (\forall \langle \sigma_{PQR}, \sigma_P, \sigma_Q, \sigma_R \rangle \in \tau. \exists \sigma_{QR}. \text{hAdd}(\sigma_{PQR}, \sigma_P, \sigma_{QR}) \wedge \text{hAdd}(\sigma_{QR}, \sigma_Q, \sigma_R)) \wedge \\ &\quad \pi_1^4(\tau) = S \wedge \pi_2^4(\tau) \in P \wedge \pi_3^4(\tau) \in Q \wedge \pi_4^4(\tau) \in R\}. \end{aligned}$$

Therefore, since  $\text{hAdd}$  is associative, which can be easily verified, it follows that  $(P \star Q) \star R = P \star (Q \star R)$ .

4. **(monotonicity)**  $P \subseteq P' \Rightarrow Q \subseteq Q' \Rightarrow P \star Q \subseteq P' \star Q'$

Follows directly from the pointwise nature of  $\star$ .

5. **( $\cup$ -distributivity)**  $(P \cup Q) \star F = (P \star F) \cup (Q \star F)$

Follows directly from the pointwise nature of  $\star$ .

6. **( $\bigcup$ -distributivity)**  $(\bigcup_{P \in X} P) \star F = \bigcup_{P \in X} (P \star F)$

Follows directly from the pointwise nature of  $\star$ .

7. **( $\otimes$ -distributivity)**  $(P \otimes Q) \star F = (P \star F) \otimes (Q \star F)$ ,  $F$  - downward and union closed

- $(P \otimes Q) \star F \subseteq (P \star F) \otimes (Q \star F)$ ,  $F$  - downward closed

Let  $S \in (P \otimes Q) \star F$ . Then obtain  $S_{PQ}$  and  $S_F$  where  $S \subseteq S_{PQ} \star S_F$ ,  $S_{PQ} \in P \star Q$ ,  $S_F \in F$  and  $\text{left\_lives}(S, S_{PQ}, S_F)$ . Now, let  $S_P \in P$  and  $S_Q \in Q$  be such that  $S_{PQ} = S_P \cup S_Q$ . Define

$$\tau_{PF} \Leftarrow \{ \langle \sigma, \sigma_P, \sigma_F \rangle \mid \sigma \in S \wedge \sigma_P \in S_P \wedge \sigma_F \in S_F \wedge \text{hAdd}(\sigma, \sigma_P, \sigma_F) \}.$$

Then  $\pi_2^3(\tau_{PF}) = S_P \in P$  and  $\pi_3^3(\tau_{PF}) \subseteq S_F \in F$ . Thus, by the downward closeness of  $F$ ,  $\pi_3^3(\tau_{PF}) \in F$ . Moreover, note that  $\pi_1^3(\tau_{PF}) \subseteq \pi_2^3(\tau_{PF}) \star \pi_3^3(\tau_{PF})$ ,  $\text{left\_lives}(\pi_1^3(\tau_{PF}), \pi_2^3(\tau_{PF}), \pi_3^3(\tau_{PF}))$  and  $\text{right\_lives}(\pi_1^3(\tau_{PF}), \pi_2^3(\tau_{PF}), \pi_3^3(\tau_{PF}))$ . Therefore  $\pi_1^3(\tau_{PF}) \in P \star F$ . Analogously, for

$$\tau_{QF} \Leftarrow \{ \langle \sigma, \sigma_Q, \sigma_F \rangle \mid \sigma \in S \wedge \sigma_Q \in S_Q \wedge \sigma_F \in S_F \wedge \text{hAdd}(\sigma, \sigma_Q, \sigma_F) \},$$

we obtain that  $\pi_1^3(\tau_{QF}) \in Q \star F$ . Finally, notice that  $S = \pi_1^3(\tau_{PF}) \cup \pi_1^3(\tau_{QF})$ . Therefore,  $S \in (P \star F) \otimes (Q \star F)$ .

- $(P \star F) \otimes (Q \star F) \subseteq (P \otimes Q) \star F$ ,  $F$  - union closed

Let  $S \in (P \star F) \otimes (Q \star F)$ . Then obtain  $S_{PF} \in P \star F$  and  $S_{QF} \in Q \star F$  such that  $S = S_{PF} \cup S_{QF}$ . Let  $S_P \in P$  and  $S_F \in F$  be witnesses for  $S_{PF} \subseteq P \star F$  and  $S_Q \in Q$  and  $S'_F \in F$  be witnesses for  $S_{QF} \subseteq Q \star F$ . Then  $S_P \cup S_Q \in P \otimes Q$  and  $S_F \cup S'_F \in F$ , since  $F$  is union closed. Moreover,  $S_{PF} \cup S_{QF} \subseteq (S_P \cup S_Q) \star (S_F \cup S'_F)$ ,  $\text{left\_lives}(S_{PF} \cup S_{QF}, S_P \cup S_Q, S_F \cup S'_F)$  and  $\text{right\_lives}(S_{PF} \cup S_{QF}, S_P \cup S_Q, S_F \cup S'_F)$ . Therefore  $S = S_{PF} \cup S_{QF} \in (P \star Q) \star F$ .

8. ( **$\otimes$ -distributivity**)  $(\otimes_{P \in X} P) \star F = \otimes_{P \in X} (P \star F)$ ,  $F$  - downward and infinite union closed

- $(\otimes_{P \in X} P) \star F \subseteq \otimes_{P \in X} (P \star F)$ ,  $F$  - downward closed

Let  $S \in (\otimes_{P \in X} P) \star F$ . Then obtain  $S_X \in (\otimes_{P \in X} P)$  and  $S_F \in F$  where  $S \subseteq S_X \star S_F$  and  $\text{left\_lives}(S, S_X, S_F)$ . Now, let  $f$  be such that  $\bigcup_{P \in X} f(P)$  and  $\forall P \in X. f(P) \in P$ . Let

$$\tau_P \Leftarrow \{ \langle \sigma, \sigma_P, \sigma_F \rangle \mid \sigma \in S \wedge \sigma_P \in f(P) \wedge \sigma_F \in S_F \wedge \text{hAdd}(\sigma, \sigma_P, \sigma_F) \}$$

for  $P \in X$ . Analogously to ( **$\otimes$ -distributivity**), we obtain that  $\pi_1^3(\tau_P) \in P \star F$ , using the downward closeness of  $F$ , and  $\bigcup_{P \in X} \pi_1^3(\tau_P) = S$ . Therefore  $S \in \otimes_{P \in X} (P \star F)$ .

- $\otimes_{P \in X} (P \star F) \subseteq (\otimes_{P \in X} P) \star F$ ,  $F$  - infinite union closed

Analogously to ( **$\otimes$ -distributivity**), i.e. the witnesses are simply the unions of the assumption witnesses.

9. (**pure-intersection**)  $\text{Intu}(P) \Rightarrow \text{Pure}(Q) \Rightarrow P \star Q = P \cap Q$

- $\text{Intu}(P) \Rightarrow \text{Intu}(Q) \Rightarrow P \star Q \subseteq P \cap Q$

Let  $S \in P \star Q$  and let  $S_P \in P$  and  $S_Q \in Q$  be witnesses for this. We have that

$$(\forall \langle s, h_P \rangle \in S_P. \exists h \supseteq h_P. \langle s, h \rangle \in S) \wedge (\forall \langle s, h \rangle \in S. \exists h_P \subseteq h. \langle s, h_P \rangle \in S_P)$$

and hence  $S \in P$  by first assumption and  $S_P \in P$ . Similarly,  $S \in Q$  by the second assumption. Therefore  $S \in P \cap Q$ .

- **Pure(Q)  $\Rightarrow$   $P \cap Q \subseteq P \star Q$**

Let  $S \in P \cap Q$ . Then  $S \in P$  and  $I \Leftarrow \{\langle s, \emptyset \rangle \mid \langle s, h \rangle \in S\} \in Q$  by the pureness of  $Q$ . Moreover,  $S = S * I$ ,  $\text{left\_lives}(S, S, I)$  and  $\text{right\_lives}(S, S, I)$ . Therefore,  $S \in P \star Q$ .

10. **(Intu-preserving)  $\text{Intu}(P) \Rightarrow \text{Intu}(Q) \Rightarrow \text{Intu}(P \star Q)$**

Let  $S \in P \star Q$  and let  $S'$  be such that

$$(\forall \langle s, h \rangle \in S. \exists h' \supseteq h. \langle s, h' \rangle \in S') \wedge (\forall \langle s, h' \rangle \in S'. \exists h \subseteq h'. \langle s, h \rangle \in S).$$

Let  $\tau$  be such that  $\forall \langle \sigma, \sigma_P, \sigma_Q \rangle \in \tau. \mathbf{hAdd}(\sigma, \sigma_P, \sigma_Q)$ ,  $\pi_1^3(\tau) = S$ ,  $\pi_2^3(\tau) \in P$  and  $\pi_3^3(\tau) \in Q$ . We define

$$\begin{aligned} \tau' \Leftarrow \{ \langle \langle s, h' \rangle, \langle s, h' \upharpoonright \overline{\text{Dom}(h_Q)} \rangle, \langle s, h' \upharpoonright \text{Dom}(h_Q) \rangle \mid \langle s, h' \rangle \in S' \wedge h \subseteq h' \\ \wedge \langle \langle s, h \rangle, \langle s, h_P \rangle, \langle s, h_Q \rangle \rangle \in \tau \}. \end{aligned}$$

Now, it is easy to verify that  $S' = \pi_1^3(\tau')$ . Moreover,  $\pi_2^3(\tau) \in P$  and  $\pi_2^3(\tau')$  satisfy the antecedent of the matrix<sup>46</sup> of the unfolded  $\text{Intu}(P)$ . Therefore,  $\pi_2^3(\tau') \in P$ . Analogously,  $\pi_3^3(\tau') \in Q$ . Furthermore, we have  $\pi_1^3(\tau') \subseteq \pi_2^3(\tau') * \pi_3^3(\tau')$ ,  $\text{left\_lives}(\pi_1^3(\tau'), \pi_2^3(\tau'), \pi_3^3(\tau'))$  and  $\text{right\_lives}(\pi_1^3(\tau'), \pi_2^3(\tau'), \pi_3^3(\tau'))$ . Therefore,  $S' \in P \star Q$ .

11. **(Pure-preserving)  $\text{Pure}(P) \Rightarrow \text{Pure}(Q) \Rightarrow \text{Pure}(P \star Q)$**

Let  $S \in P \star Q$  and let  $S'$  be such that

$$(\forall \langle s, h \rangle \in S. \exists h'. \langle s, h' \rangle \in S') \wedge (\forall \langle s, h' \rangle \in S'. \exists h. \langle s, h \rangle \in S)$$

and let  $S_P \in P$  and  $S_Q \in Q$  be witnesses for  $S \in P \star Q$ . It's easy to verify that  $S_P \in P$  and  $S'$  satisfy the antecedent of the matrix of  $\text{Pure}(P)$ . Therefore  $S' \in P$ . Analogously,  $S_Q \in Q$  and  $I \Leftarrow \{\langle s, \emptyset \rangle \mid \langle s, h' \rangle \in S'\}$  satisfy the antecedent of the matrix of  $\text{Pure}(Q)$ . Therefore  $I \in Q$ . Furthermore,  $S' \subseteq S' * I$ ,  $\text{left\_lives}(S', S', I)$  and  $\text{right\_lives}(S', S', I)$ . Therefore,  $S' \in P \star Q$ .

13. **(FR-SL)  $\{S \mid S \subseteq p\} \star \{S \mid S \subseteq f\} = \{S \mid S \subseteq p * f\}$**

- $\{S \mid S \subseteq p\} \star \{S \mid S \subseteq f\} \subseteq \{S \mid S \subseteq p * f\}$

Let  $S \in \{S \mid S \subseteq p\} \star \{S \mid S \subseteq f\}$  and let  $S_p \subseteq p$  and  $S_f \subseteq f$  be witnesses for this. In particular,  $S \subseteq S_p * S_f$  and hence, by monotonicity,  $S \subseteq p * f$ .

<sup>46</sup>The matrix of a formula in prenex normal form is its quantifier-free core, i.e., the part remaining after all quantifiers have been removed.



- $\{S \mid S \subseteq p * f\} \subseteq \{S \mid S \subseteq p\} * \{S \mid S \subseteq f\}$   
Let  $S \in \{S \mid S \subseteq p * f\}$ , i.e.  $S \subseteq p * f$ . We define

$$\tau \Leftarrow \{\langle \sigma, \sigma_p, \sigma_f \rangle \mid \sigma \in S \wedge \sigma_p \in p \wedge \sigma_f \in f \wedge \mathbf{hAdd}(\sigma, \sigma_p, \sigma_f)\}.$$

Now,  $S = \pi_1^3(\tau)$ ,  $\pi_2^3(\tau) \in \{S \mid S \subseteq p\}$  and  $\pi_3^3(\tau) \in \{S \mid S \subseteq f\}$ .  
Moreover,  $\pi_1^3(\tau) \subseteq \pi_2^3(\tau) * \pi_3^3(\tau)$ ,  $\mathbf{left\_lives}(\pi_1^3(\tau), \pi_2^3(\tau), \pi_3^3(\tau))$  and  $\mathbf{right\_lives}(\pi_1^3(\tau), \pi_2^3(\tau), \pi_3^3(\tau))$ . Therefore,  $S \in \{S \mid S \subseteq p\} * \{S \mid S \subseteq f\}$ .

14. **(FR-SSIL)**  $\{S \mid S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\} \subseteq \{S \mid S \cap (p * f) \neq \emptyset\}$

Let  $S \in \{S \mid S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\}$  and let  $S_p \in \{S \mid S \cap p \neq \emptyset\}$  and  $S_f \in \{S \mid S \subseteq f\}$  be witnesses for this. Let  $\langle s, h_p \rangle \in S_p \cap p$  and since  $\mathbf{left\_lives}(S, S_p, S_f)$ , we obtain  $h$  and  $h_f$  where  $\langle s, h \rangle \in S$ ,  $\langle s, h_f \rangle \in S_f$ ,  $\mathbf{hAdd}(\langle s, h \rangle, \langle s, h_p \rangle, \langle s, h_f \rangle)$  and  $h_p \perp h_f$ . Then, we have  $\langle s, h_f \rangle \in f$  by the definition of  $S_f$ . Therefore,  $\langle s, h \rangle \in p * f$  and hence, since  $\langle s, h \rangle \in S$ , we conclude that  $S \in \{S \mid S \cap (p * f) \neq \emptyset\}$ .

15. **(FR-OSL-1)**  $\{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\} = \{S \mid S \subseteq p * f \wedge S \cap (p * f) \neq \emptyset\}$

- $\{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\} \subseteq \{S \mid S \subseteq p * f \wedge S \cap (p * f) \neq \emptyset\}$

Let  $S \in \{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\}$  and let  $S_p \in \{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\}$  and  $S_f \in \{S \mid S \subseteq f\}$  be witnesses for this. Similarly to **(FR-SL)**, we show that  $S \in \{S \mid S \subseteq p * f\}$ . The second part, namely  $S \neq \emptyset$  follows from  $\mathbf{left\_lives}(S, S_p, S_f)$  and the fact that  $S_p \neq \emptyset$ . Therefore,  $S \in \{S \mid S \subseteq p * f \wedge S \cap (p * f) \neq \emptyset\}$ .

- $\{S \mid S \subseteq p * f \wedge S \cap (p * f) \neq \emptyset\} \subseteq \{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\}$   
Analogously to **(FR-SL)**.

16. **(FR-OSL-2)**  $\{S \mid S \subseteq p_1 \cup p_2 \wedge S \cap p_1 \neq \emptyset \wedge S \cap p_2 \neq \emptyset\} * \{S \mid S \subseteq f\} = \{S \mid S \subseteq (p_1 \cup p_2) * f \wedge S \cap (p_1 * f) \neq \emptyset \wedge S \cap (p_2 * f) \neq \emptyset\}$

First, note that

$$\{S \mid S \subseteq p_1 \cup p_2 \wedge S \cap p_1 \neq \emptyset \wedge S \cap p_2 \neq \emptyset\} = \{S \mid S \subseteq p_1 \wedge S \cap p_1 \neq \emptyset\} \otimes \{S \mid S \subseteq p_2 \wedge S \cap p_2 \neq \emptyset\}.$$

Now, by applying (**⊗-distributivity**), the left side equals

$$(\{S \mid S \subseteq p_1 \wedge S \cap p_1 \neq \emptyset\} * \{S \mid S \subseteq f\}) \otimes (\{S \mid S \subseteq p_2 \wedge S \cap p_2 \neq \emptyset\} * \{S \mid S \subseteq f\})$$

which, after applying **(FR-OSL-1)**, equals

$$\{S \mid S \subseteq p_1 * f \wedge S \cap (p_1 * f) \neq \emptyset\} \otimes \{S \mid S \subseteq p_2 * f \wedge S \cap (p_2 * f) \neq \emptyset\}$$

which equals

$$\{S \mid S \subseteq (p * f) \cup (q * f) \wedge S \cap (p * f) \neq \emptyset \wedge S \cap (q * f) \neq \emptyset\}.$$

Finally, since  $*$  distributes over  $\cup$ , we conclude that the statement is true.

17. **(FR-RSL)**  $\{S \mid S_1 \times S_2 \subseteq \mathbb{R}\} \star \{S \mid S_1 \times S_2 \subseteq \mathbb{F}\} \subseteq \{S \mid S_1 \times S_2 \subseteq \mathbb{R} \boxtimes \mathbb{F}\}$ ,  
 where  $S_i \triangleq \{\langle s, h \rangle \in S \mid s(t) = i\}$ .

Let  $S \in \{S \mid S_1 \times S_2 \subseteq \mathbb{R}\} \star \{S \mid S_1 \times S_2 \subseteq \mathbb{F}\}$  and  $S_R \in \{S \mid S_1 \times S_2 \subseteq \mathbb{R}\}$   
 and  $S_F \in \{S \mid S_1 \times S_2 \subseteq \mathbb{F}\}$  be witnesses for this. In particular,  $S \subseteq S_R \star S_F$ .  
 Then we have

$$S_1 \subseteq (S_R)_1 \star (S_F)_1 \text{ and } S_2 \subseteq (S_R)_2 \star (S_F)_2.$$

Moreover,

$$(S_R)_1 \times (S_R)_2 \subseteq \mathbb{R} \text{ and } (S_F)_1 \times (S_F)_2 \subseteq \mathbb{F}$$

and hence

$$((S_R)_1 \star (S_F)_1) \times ((S_R)_2 \star (S_F)_2) \subseteq \mathbb{R} \boxtimes \mathbb{F}.$$

Therefore,  $S_1 \times S_2 \subseteq \mathbb{R} \boxtimes \mathbb{F}$  and thus  $S \in \{S \mid S_1 \times S_2 \subseteq \mathbb{R} \boxtimes \mathbb{F}\}$ .

### 3.2 Hyper-tripe validity

Before delving into the frame rule, we need to establish what it means for our hyper-triples to be valid, i.e.  $\models_{HSL} \{P\}C\{Q\}$ . This is not straightforward, as we lack an (easily identifiable) natural condition ensuring the soundness of the frame rule, unlike in SL. Specifically, because we aim to support both over- and underapproximate reasoning, we cannot reuse the mechanism used in SL, namely the requirement that the precondition provides the memory required by the program command. Indeed, if we defined validity with this requirement, then

$$\not\models_{HSL} \{\llbracket \exists \langle \mathcal{J} \rangle. \mathcal{J}(x \doteq 0) \rrbracket\} \mathbf{skip} + \mathbf{free}(x) \{\llbracket \exists \langle \mathcal{J} \rangle. \mathcal{J}(x \doteq 0) \rrbracket\},$$

since

$$\forall S \in \llbracket \exists \langle \mathcal{J} \rangle. \mathcal{J}(x \doteq 0) \rrbracket. S \text{ provides the memory required by } C$$

does not hold. That is, we have essentially lost all (non-pure) underapproximate reasoning capabilities. One might suggest a naive solution: to distinguish whether the precondition is  $\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(\mathcal{J}) \rrbracket$  or  $\llbracket \exists \langle \mathcal{J} \rangle. \mathcal{J}(\mathcal{J}) \rrbracket$ , enforcing the requirement in the former case but not in the latter. However, this approach is highly unsatisfactory and artificial.

The approach we have adopted is standard (e.g., see the third condition in RSL's validity). Rather than searching for a "natural" condition, we simply take the weakest definition (which is at least as strong as HHL's validity) that ensures a sound frame rule. That is, we "embed" the frame rule into the definition of validity:

$$\models_{HSL} \{P\}C\{Q\} \stackrel{def}{\iff} \forall S. \forall f. S \in P \star \{S \mid S \subseteq f\} \Rightarrow \llbracket C \rrbracket[S] \in Q \star \{S \mid S \subseteq f\}.$$

Note that **this definition is only a rough outline**, meant for illustration, since we quantify over all frames and not only those that satisfy " $\mathbf{fv}(f) \cap \mathbf{md}(C) = \emptyset$ ". That is, according to this definition

$$\not\models_{HSL} \{\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(x \doteq 0) \rrbracket\} x := \llbracket x + 1 \rrbracket \{\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(x \doteq 1) \rrbracket\},$$

since for  $f \Leftarrow \llbracket x \doteq 0 \rrbracket$ ,  $\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(x \doteq 0) \rrbracket \star \{S \mid S \subseteq \llbracket x \doteq 0 \rrbracket\} = \llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(x \doteq 0) \rrbracket$ , whereas  $\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(x \doteq 1) \rrbracket \star \{S \mid S \subseteq \llbracket x \doteq 0 \rrbracket\} = \{\emptyset\}$ . The fix for this is straightforward—one simply needs to introduce a syntax and define the semantic condition " $\text{fv}(f) \cap \text{md}(C) = \emptyset$ ". In this thesis, we do neither, as this is not our focus.

We point out that there is a more fundamental problem with this definition (or rather, with the state model it rests upon!): None of the encodings used in HHL, work in the separation variant. That is, for example,

$$\models_{SL} \{\llbracket \mathcal{P} \rrbracket\} C \{\llbracket \mathcal{Q} \rrbracket\} \iff \models_{HSL} \{\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(\mathcal{P}) \rrbracket\} C \{\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(\mathcal{Q}) \rrbracket\}$$

does not hold. Specifically, recall the discussion at the end of subsection 2.2, where we demonstrated that SL's validity definition is not the weakest one (which is at least as strong as HL's validity) that supports a sound frame rule. The issue arises from this distinction—HSL's validity is defined as the weakest one that supports a sound frame rule, whereas SL's validity is not. Formally,

$$\not\models_{SL} \{\llbracket \top \rrbracket\} x := [y] \{\llbracket \top \rrbracket\},$$

whereas, assuming we have added the condition " $\text{md}(C) \cap \text{fv}(f) = \emptyset$ " to the definition of  $\models_{HSL}$ ,

$$\models_{HSL} \{\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket\} x := [y] \{\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket\}.$$

The former doesn't hold, since the precondition doesn't provide the memory required by  $x := [y]$ . The latter holds, since no matter what frame  $f$  (such that " $\text{md}(C) \cap \text{fv}(f) = \emptyset$ " holds) we add to  $\llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket$ , the only thing the program changes is  $x$ , i.e. for any  $S \in \llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket \star \{S \mid S \subseteq f\}$ , it follows that  $\llbracket C \rrbracket[S]$  is the same as  $S$  but with changed  $x$ . Hence by the " $\text{md}(C) \cap \text{fv}(f) = \emptyset$ ", it follows that  $\llbracket C \rrbracket[S] \in \llbracket \forall \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket \star \{S \mid S \subseteq f\}$ .

One might expected that this definition (at least) works for SSIL, as, similarly to SSIL, it does not require the precondition to provide the memory required by the program command. However, that's not the case either, since, HSL considers all paths simultaneously, whereas SSIL focuses on specific paths. Formally,

$$\models_{SSIL} \{\llbracket \top \rrbracket\} \mathbf{skip} + \text{free}(x) \{\llbracket \top \rrbracket\},$$

whereas

$$\not\models_{HSL} \{\llbracket \exists \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket\} \mathbf{skip} + \text{free}(x) \{\llbracket \exists \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket\}.$$

The former holds, since for any state, we have a terminating execution path (via  $\mathbf{skip}$ ) to  $\llbracket \top \rrbracket$ . The latter doesn't hold, since for  $f \Leftarrow \llbracket x \mapsto 5 \rrbracket$  and  $S \Leftarrow \{\langle s_x^1, \emptyset_1^5 \rangle \in \llbracket \exists \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket \star \{S \mid S \subseteq f\}$ , the resulting  $\llbracket C \rrbracket[S] = \{\langle s_x^1, \emptyset_1^5 \rangle, \langle s_x^1, \emptyset \rangle\} \notin \llbracket \exists \langle \mathcal{J} \rangle. \mathcal{J}(\top) \rrbracket \star \{S \mid S \subseteq f\}$ , since it considers both paths and, in particular,  $\langle s_x^1, \emptyset \rangle$  lacks the required by the frame heap  $\llbracket x \mapsto 5 \rrbracket$ .

The essence of the issue lies at a deeper level. The primary concern is not the definition of validity itself (with the added condition " $\text{md}(C) \cap \text{fv}(f) = \emptyset$ "), but rather the state model itself. Specifically, we do not account for errors. In

[Zilberstein et al. 2024], they demonstrate that by incorporating errors, a sound frame rule can be achieved, which accommodates both under- and overapproximation. However, this is beyond the scope of the current thesis.

Our goal is to develop a separating conjunction (which we already did) and as general as possible frame rule, which is at least as strong as the frame rules found in these four logics. It is important to note that we work under the assumption that the encodings, used in HHL, would also be applicable to HSL, once the state model is generalized to account for errors. This assumption is reasonable, as the encodings capture the very essence of the logics.

### 3.3 The strongest postcondition

In the context of this subsection assume that the condition "  $\text{md}(C) \cap \text{fv}(f) = \emptyset$ " is added to the definition of  $\models_{HSL}$ .

Unlike the previously discussed logics, it is unclear whether our logic supports strongest postcondition. That is, it may contain multiple minimal postconditions, but not a strongest one. Recall that in HHL, for each  $S$  in the precondition  $P$ , we had exactly one resulting  $\llbracket C \rrbracket[S]$ , which was required to be included in the postcondition  $Q$ . However, that's not the case anymore due to the interplay between the allocation axiom, the requirement for a sound frame rule, and the fact that HSL supports underapproximate reasoning.

First, recall that the semantic frame rule of SSIL is unsound, e.g.

$$\frac{\models_{SSIL} \{\llbracket \text{emp} \rrbracket\} x := \text{alloc}() \{\llbracket x \doteq 42 \rrbracket\}}{\not\models_{SSIL} \{\llbracket \text{emp} \rrbracket * \llbracket 42 \mapsto 42 \rrbracket\} x := \text{alloc}() \{\llbracket x \doteq 42 \rrbracket * \llbracket 42 \mapsto 42 \rrbracket\}},$$

and we argued (see section 2.3) that this is common issue of underapproximate logics. In HSL, we ensured that the semantic frame rule is sound:

$$\models_{HSL} \{\llbracket \exists(j). j(\top) \rrbracket\} x := \text{alloc}() \{\llbracket \exists(j). j(x \doteq 42) \rrbracket\},$$

since the frame rule is embedded. That is, let  $f \Leftarrow \llbracket 42 \mapsto 42 \rrbracket$ . Then for  $S \in \llbracket \exists(j). j(\top) \rrbracket * \{S \mid S \subseteq \llbracket 42 \mapsto 42 \rrbracket\}$ , it follows that all  $\langle s, h \rangle \in S$  are such that location 42 is allocated. Therefore the  $x := \text{alloc}()$  command cannot allocate location 42 and subsequently store it in  $x$ . Hence, the resulting  $\llbracket x := \text{alloc}() \rrbracket[S]$  cannot satisfy the postcondition  $\llbracket \exists(j). j(x \doteq 42) \rrbracket$ . However, the semantic soundness of the frame rule may come at the cost of losing the existence of a strongest postcondition—though we hope this is not the case. If it turns out that a strongest postcondition does not exist, this would further hint at a fundamental incompatibility between underapproximation, the frame rule (based on the proposed  $\star$ <sup>47</sup>), and completeness. We leave this for further exploration and instead focus on explaining why a strongest postcondition may fail to exist.

Note that the axioms still (as in HHL) follow the general patten of

$$\models_{HSL} \{P\} C \underbrace{\{\{\llbracket C \rrbracket[S] \mid S \in P\}\}}_Q.$$

<sup>47</sup>It may be the case that we have missed a crucial property that must be satisfied by  $\star$ .

That is, for each  $S \in P$ , we require that  $\llbracket C \rrbracket [S] \in Q$ . The only exception is the allocation axiom. In order to show a simpler example, assume we have a more precise version of  $x := \text{alloc}()$ , namely  $x := \text{alloc}(e)$ , where  $\llbracket x := \text{alloc}(e) \rrbracket = \{ \langle \langle s, h \rangle, \langle s_x^l, h_l^{\llbracket e \rrbracket(s)} \rangle \mid (l \notin \text{Dom}(h) \vee h(l) = \perp) \wedge l \neq 0 \}$ . Let  $s \in \text{Stacks}$  be arbitrary, such that  $s(x) = 7$ . We have (independent of the choice of  $s$ ) that

$$\not\models_{HSL} \underbrace{\{ \langle \langle s, \emptyset \rangle \rangle \}}_P \overset{S_P}{x := \text{alloc}(5)} \underbrace{\{ \langle \langle s_x^l, \emptyset_l^5 \rangle \mid l \neq 0 \}}_Q \overset{\text{sp}(x:=\text{alloc}(5), S_P)}{}$$

even though the only element  $S \Leftarrow \{ \langle s, \emptyset \rangle \} \in P$  has its denotational image  $\llbracket x := \text{alloc}(5) \rrbracket [S]$  belong to  $Q$ , i.e.  $\llbracket x := \text{alloc}(5) \rrbracket [S] = \{ \langle s_x^l, \emptyset_l^5 \rangle \mid l \neq 0 \} \in Q$ . The reason for that is the embeded frame rule and the witness preserving nature of  $\star$ . More precisely, let  $f \Leftarrow \llbracket 42 \mapsto 42 \rrbracket$ . Then  $P \star f = \{ \langle \langle s, \emptyset_{s(x)}^{42} \rangle \rangle \}$ , but the denotational image of its only element,  $\{ \langle s, \emptyset_{s(x)}^{42} \rangle \}$ , does not belong to  $Q \star \{ S \mid S \subseteq f \}$ . That is,

$$\llbracket x := \text{alloc}(5) \rrbracket [ \{ \langle s, \emptyset_7^{42} \rangle \} ] = \{ \langle s_x^l, \emptyset_{7,l}^{42,5} \rangle \mid l \neq 0 \wedge l \neq 7 \} \notin \{ \langle \langle s_x^l, \emptyset_l^5 \rangle \rangle \mid l \neq 0 \} \star \{ S \mid S \subseteq \llbracket 42 \mapsto 42 \rrbracket \}.$$

Indeed, notice that

$$\{ \langle \langle s_x^l, \emptyset_l^5 \rangle \rangle \mid l \neq 0 \} \star \{ S \mid S \subseteq \llbracket 42 \mapsto 42 \rrbracket \} = \emptyset,$$

which stems from the fundamental design choice made for the  $\star$ , namely the preservation of the witnesses (formalized with `left_lives` and `right_lives`). More precisely,  $\langle s_x^{42}, \emptyset_{42}^5 \rangle \in \{ \langle \langle s_x^l, \emptyset_l^5 \rangle \rangle \mid l \neq 0 \}$ , which is the only element of  $\{ \langle \langle s_x^l, \emptyset_l^5 \rangle \rangle \mid l \neq 0 \}$ , cannot "continue its life", since every  $S_f \in \{ S \mid S \subseteq \llbracket 42 \mapsto 42 \rrbracket \}$  has already location 42 allocated for all  $\sigma_f \in S_f$ , thus

$$\forall S_f \in \{ S \mid S \subseteq \llbracket 42 \mapsto 42 \rrbracket \}. \not\# S. \text{left\_lives}(S, \{ \langle \langle s_x^l, \emptyset_l^5 \rangle \rangle \mid l \neq 0 \}, S_f).$$

In order to resolve this issue, more precise alternatives to `left_lives` and `right_lives`, which allow some states to not "continue their lives", are required. However, this falls beyond the scope of this thesis.

**Discussion 3.13.** Since we haven't formally defined  $\models_{HSL}$  with the condition " $\text{md}(C) \cap \text{fv}(f) = \emptyset$ " added, we refrain from further exploration of the allocation axiom (and HSL in general). We, however, strongly believe that there exists a strongest postcondition for the allocation axiom and that the proposed  $\star$  satisfies the 12<sup>th</sup> property for said program command, namely

$$\text{SP}(P, x := \text{alloc}()) \star F = \text{SP}(P \star F, x := \text{alloc}()).$$

We strongly believe that the strongest postcondition  $\text{SP}(\{p\}, x := \text{alloc}())$  exists and is the set of all finitely smaller subsets of the denotational image  $\llbracket x := \text{alloc}() \rrbracket [p]$ . For  $P$  with cardinality more than 1, we apply the trick from subsection 2.7.1. Notice the finitely smaller requirement—it stems from the fact that we model the heaps as finite partial functions, rather than just partial functions.

Finally, we show yet another fundamental flaw of the state model. That is, the 12<sup>th</sup> property, namely **(operational-coherence)**, does not hold for the error-prone commands  $[x] := e$ ,  $y := [x]$  and  $\text{free}(x)$ . Indeed, consider  $\text{SP}(\llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\text{emp}) \rrbracket, \text{free}(x))$ . Even though a general definition for  $\text{SP}(P, C)$  is non-trivial (and perhaps doesn't even exist), this is not the case when  $C$  is a base command, different that allocation. In that case, it is easy to see that  $\text{SP}(P, C) = \{\llbracket C \rrbracket[S] \mid S \in P\}$ . Therefore,

$$\text{SP}(\llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\text{emp}) \rrbracket, \text{free}(x)) = \{\emptyset\}$$

and hence

$$\text{SP}(\llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\text{emp}) \rrbracket, \text{free}(x)) \star \{S \mid S \subseteq \llbracket x \mapsto 1 \rrbracket\} = \{\emptyset\}.$$

However,

$$\llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\text{emp}) \rrbracket \star \{S \mid S \subseteq \llbracket x \mapsto 1 \rrbracket\} = \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(x \mapsto 1) \rrbracket$$

and thus

$$\text{SP}(\llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(\text{emp}) \rrbracket \star \{S \mid S \subseteq \llbracket x \mapsto 1 \rrbracket\}, \text{free}(x)) = \llbracket \forall \langle \mathcal{J} \rangle . \mathcal{J}(x \mapsto \perp) \rrbracket \supset \{\emptyset\}.$$

Similarly to how  $\text{sp}(p, C) * f = \text{sp}(p * f, C)$  failed to hold for the less expressive state model but held for the more expressive state model (which incorporates  $\perp$ ; see subsection 2.2.4), we strongly believe that for a more expressive state model—one that accounts for errors—the property **(operational-coherence)** will hold, at least for the base program commands. Note that this issue has significant implications: there are no axioms for error-prone commands, which do not provide the memory required by the program. That is, for any  $Q$  (even  $\llbracket \top \rrbracket$ )

$$\not\equiv_{HSL} \{\llbracket \forall \langle \mathcal{J} \rangle . (\text{emp}) \rrbracket\} \text{free}(x) \{Q\}.$$

and, more importantly,

$$\not\equiv_{HSL} \{\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \mapsto 5) \rrbracket\} \text{free}(x) \{\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \mapsto \perp) \rrbracket\},$$

since for  $f \Leftarrow \{\langle s_x^1, \emptyset \rangle, \langle s_x^2, \emptyset_2^2 \rangle\}$ <sup>48</sup> and  $S \Leftarrow \{\langle s_x^1, \emptyset_1^1 \rangle, \langle s_x^2, \emptyset_2^2 \rangle\} \in \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \mapsto 5) \rrbracket \star \{S \mid S \subseteq f\}$ , we have that  $\llbracket \text{free}(x) \rrbracket[S] = \{\langle s_x^1, \emptyset_1^1 \rangle, \langle s_x^2, \emptyset_2^1 \rangle\} \notin \llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \mapsto \perp) \rrbracket \star \{S \mid S \subseteq f\}$ . The same counterexample can be used to show that

$$\not\equiv_{HSL} \{\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \mapsto 5) \rrbracket\} \text{free}(x) \{\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(\top) \rrbracket\}$$

and even

$$\not\equiv_{HSL} \{\llbracket \exists \langle \mathcal{J} \rangle . \mathcal{J}(x \mapsto 5) \rrbracket\} \text{free}(x) \{\llbracket \top \rrbracket\}.$$

A syntactically expressible variant of the counterexample consists of the same  $S$  and  $f \Leftarrow \llbracket (x \doteq 1 \Rightarrow \text{emp}) \wedge (x \doteq 2 \Rightarrow 2 \mapsto 2) \rrbracket$ .

<sup>48</sup>Since  $\text{md}(\text{free}(x)) = \emptyset$ , then any  $f$  satisfies the condition " $\text{md}(\text{free}(x)) \cap \text{fv}(f) = \emptyset$ ".

### 3.4 The Frame rule

#### 3.4.1 Soundness

In the context of this subsection, we assume that the condition " $\text{md}(C) \cap \text{fv}(f) = \emptyset$ " is **not** added to the definition of  $\models_{HSL}$ .

Recall that our goal is to obtain a frame rule, sound for any downward closed frame (see subsection 2.7.3). As previously mentioned, we achieve this by embedding the frame rule into the definition of validity:

$$\models_{HSL} \{P\}C\{Q\} \stackrel{def}{\iff} \forall S. \forall f. S \in P \star \{S \mid S \subseteq f\} \Rightarrow \llbracket C \rrbracket[S] \in Q \star \{S \mid S \subseteq f\}.$$

However, instead of embedding arbitrary downward closed frame  $F$  into the validity definition, we embedded an arbitrary power set frame  $\{S \mid S \subseteq f\}$ , which is both downward closed and closed under infinite union. It turns out that this is sufficient to establish the soundness of the frame rule for arbitrary downward closed frames, which further reinforces the idea that our  $\star$  is well-grounded.

We begin by proving that the rule

$$\frac{\text{for all } i \in I. \models_{HSL} \{P_i\}C\{Q_i\}}{\models_{HSL} \left\{ \bigcup_{i \in I} P_i \right\} C \left\{ \bigcup_{i \in I} Q_i \right\}} \text{ (Idx-Union)}$$

is still (as in HHL) sound. Indeed, let  $S$  and  $f$  be such that  $S \in (\bigcup_{i \in I} P_i) \star \{S \mid S \subseteq f\}$ . That is, by (**U-distributivity**),  $S \in \bigcup_{i \in I} (P_i \star \{S \mid S \subseteq f\})$ . Let  $i \in I$  such that  $S \in P_i \star \{S \mid S \subseteq f\}$ . Then, by the assumption,  $\llbracket C \rrbracket[S] \in Q_i \star \{S \mid S \subseteq f\} \subseteq \bigcup_{i \in I} (Q_i \star \{S \mid S \subseteq f\})$ . Therefore, by (**U-distributivity**), we conclude that  $\llbracket C \rrbracket[S] \in (\bigcup_{i \in I} Q_i) \star \{S \mid S \subseteq f\}$ . Thus, the rule (Idx-Union) is sound.

**Theorem 3.14.** *The semantic frame rule*

$$\frac{\models_{HSL} \{P\}C\{Q\} \quad \forall S \in F. \forall S' \subseteq S. S' \in F}{\models_{HSL} \{P \star F\}C\{Q \star F\}} \text{ (Frame)}$$

is sound.

*Proof.* First, using (**associativity**), (**FR-SL**) and  $\models_{HSL} \{P\}C\{Q\}$ , we obtain that for any  $f$ ,

$$\models_{HSL} \{P \star \{S \mid S \subseteq f\}\}C\{Q \star \{S \mid S \subseteq f\}\}$$

and now by the (Idx-Union) rule, we obtain

$$\models_{HSL} \left\{ \bigcup_{f \in F} (P \star \{S \mid S \subseteq f\}) \right\} C \left\{ \bigcup_{f \in F} (Q \star \{S \mid S \subseteq f\}) \right\}$$

and hence by (**U-distributivity**) and (**commutativity**)

$$\models_{HSL} \left\{ P \star \left( \bigcup_{f \in F} \{S \mid S \subseteq f\} \right) \right\} C \left\{ Q \star \left( \bigcup_{f \in F} \{S \mid S \subseteq f\} \right) \right\}$$

Finally, since  $F$  is downward closed, it follows that  $F = \bigcup_{f \in F} \{S \mid S \subseteq f\}$ . Therefore,

$$\models_{HSL} \{P \star F\}C\{Q \star F\}.$$

□

Of course, once " $\text{md}(C) \cap \text{fv}(f) = \emptyset$ " is added to the definition of  $\models_{HSL}$ , we will be able to prove (at most) the usual frame rule, which has " $\text{md}(C) \cap \text{fv}(F) = \emptyset$ " in the assumption. Finally, as discussed in subsection 2.4, where we demonstrate that the frame rule becomes unsound if the frame contains  $\perp$  syntactically, we need to further weaken the embedding of the frame rule in the definition of  $\models_{HSL}$  by quantifying only over frames that do not syntactically contain  $\perp$  and hence obtain a frame rule, which takes as an additional assumption that the frame does not contain syntactically  $\perp$ . That is,

$$\frac{\models_{HSL} \{P\}C\{Q\} \quad \forall S \in F. \forall S' \subseteq S. S' \in F \quad \text{"fv}(F) \cap \text{md}(C) = \emptyset \quad \text{"no } \perp \text{ in } F\text{"}}{\models_{HSL} \{P \star F\}C\{Q \star F\}} \text{ (Frame)}$$

### 3.4.2 Expressivity

Recall that none of the encodings used in HHL apply in our case due to the weak (non-erroneous) state model. However, we will provide high-level semantic<sup>49</sup> arguments explaining why HSL's frame rule (alongside other HSL axioms and rules) should be sufficient to capture the frame rules of SL, SSIL and OSL. Moreover, we will present an initial idea for deriving RSL's frame rule, which has not been fully explored and may or may not succeed. Crucially, we assume that the final encodings (that will be used in HSL with erroneous state model) will match those used in HHL. That is, we assume that for

$$\text{ENC}_{SL}(p) \Leftarrow \{S \mid S \subseteq p\}$$

$$\text{ENC}_{SSIL}(p) \Leftarrow \{S \mid S \cap p \neq \emptyset\}$$

$$\text{ENC}_{OSL}(p) \Leftarrow \{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\}, \text{ for the base case hasassertion } \mathcal{P}^+(p) \text{ in OSL}$$

$$\text{ENC}_{RSL}(R, t) \Leftarrow \{S \mid \forall s_1 \in S. \forall s_2 \in S. s_1(t) = 1 \Rightarrow s_2(t) = 2 \Rightarrow [s_2^1] \in R\}$$

we have

$$\models_{SL} \{p\}C\{q\} \iff \models_{HSL} \{\text{ENC}_{SL}(p)\}C\{\text{ENC}_{SL}(q)\}$$

$$\models_{SSIL} \{p\}C\{q\} \iff \models_{HSL} \{\text{ENC}_{SSIL}(p)\}C\{\text{ENC}_{SSIL}(q)\}$$

$$\models_{OSL} \{P\}C\{Q\} \iff \models_{HSL} \{P\}C\{Q\}, \text{ for OSL hasassertions } P \text{ and } Q$$

$$\models_{RSL} \{R\}C\{S\} \iff \models_{HSL} \{\text{ENC}_{RSL}(R)\}C\{\text{ENC}_{RSL}(S)\},$$

where there is a subtle detail in the RSL case, discussed in the corresponding bullet point.

<sup>49</sup>We trust that the reader can easily construct the syntactic variants.



- SL

The frame rule of SL

$$\frac{\models_{SL} \{p\}C\{q\} \quad \text{"md}(C) \cap \text{fv}(f) = \emptyset \quad \text{"no } \perp \text{ in } f\text{"}}{\models_{SL} \{p * f\}C\{q * f\}} \text{ (Frame)}$$

can be expressed in HSL as follows

$$\frac{\models_{HSL} \{\{S \mid S \subseteq p\}\}C\{\{S \mid S \subseteq q\}\} \quad \text{"md}(C) \cap \text{fv}(\{S \mid S \subseteq f\}) = \emptyset \quad \text{"no } \perp \text{ in } \{S \mid S \subseteq f\}\text{"}}{\models_{HSL} \{\{S \mid S \subseteq p\} * \{S \mid S \subseteq f\}\}C\{\{S \mid S \subseteq q\} * \{S \mid S \subseteq f\}\}} \text{ (Frame)}$$

Now, using property **(FR-SL)**, namely  $\{S \mid S \subseteq p\} * \{S \mid S \subseteq f\} = \{S \mid S \subseteq p * f\}$  and the fact that our assertions have no free SVars variables and are obtained via quantification of assertions (hence " $\text{fv}(\{S \mid S \subseteq f\}) = \text{fv}(f)$ " and " $\perp$  in  $\{S \mid S \subseteq f\}$  iff  $\perp$  in  $f$ "), we conclude that HSL's frame rule gives directly SL's frame rule:

$$\frac{\models_{HSL} \{\text{ENC}_{SL}(p)\}C\{\text{ENC}_{SL}(q)\} \quad \text{"md}(C) \cap \text{fv}(f) = \emptyset \quad \text{"no } \perp \text{ in } f\text{"}}{\models_{HSL} \{\text{ENC}_{SL}(p * f)\}C\{\text{ENC}_{SL}(q * f)\}} \text{ (Frame)}$$

For simplicity, in the remaining bullet points, we will ignore the filtering assumptions of the frame rule, namely " $\text{md}(C) \cap \text{fv}(f) = \emptyset$ " and " $\text{no } \perp \text{ in } f$ ", as they are analogous.

- SSIL

Using HSL's frame rule, we obtain

$$\frac{\models_{HSL} \{\{S \mid S \cap p \neq \emptyset\}\}C\{\{S \mid S \cap q \neq \emptyset\}\}}{\models_{HSL} \{\{S \mid S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\}\}C\{\{S \mid S \cap q \neq \emptyset\} * \{S \mid S \subseteq f\}\}} \text{ (Frame)}$$

and using property **(FR-SSIL)**, namely  $\{S \mid S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\} \subseteq \{S \mid S \cap (p * f) \neq \emptyset\}$ , and the consequence rule (which is easily verified sound), we obtain

$$\frac{\models_{HSL} \{\{S \mid S \cap p \neq \emptyset\}\}C\{\{S \mid S \cap q \neq \emptyset\}\}}{\models_{HSL} \{\{S \mid S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\}\}C\{\{S \mid S \cap q \neq \emptyset\} * \{S \mid S \subseteq f\}\}} \text{ (Frame)}$$

$$\frac{}{\models_{HSL} \{\{S \mid S \cap p \neq \emptyset\} * \{S \mid S \subseteq f\}\}C\{\{S \mid S \cap (q * f) \neq \emptyset\}\}} \text{ (Cons)}$$

That is, we obtained a little weaker version of SSIL's frame rule. To make it as strong as SSIL's one, we need to generalize the precondition to  $\{S \mid S \cap (p * f) \neq \emptyset\}$ . To achieve this, we introduce the following axiom

$$\frac{}{\models_{HSL} \{P\}C\{\top\}} \text{ (Triv-Post)}$$

and rule

$$\frac{\models_{HSL} \{P\}C\{Q\} \quad \models_{HSL} \{P'\}C\{Q'\}}{\models_{HSL} \{P \otimes P'\}C\{Q \otimes Q'\}} \text{ (Join)}$$

As already discussed (see subsection 3.3), (Triv-Post) is unsound for the current state model. However, once errors are incorporated, the definition of the  $\star$  can be adjusted (while keeping the frame embedding definition of  $\models_{HSL}$ ) in an appropriate manner to restore its soundness. The (Join) rule is easily proven sound, using the property ( $\otimes$ -**distributivity**). Now, incorporating (Triv-Post), (Join) and (Frame), we obtain

$$\frac{\frac{\frac{\models_{HSL} \{\{S \mid S \cap p \neq \emptyset\}\} C \{\{S \mid S \cap q \neq \emptyset\}\}}{\models_{HSL} \{\{S \mid S \cap p \neq \emptyset\} \star \{S \mid S \subseteq f\}\} C \{\{S \mid S \cap q \neq \emptyset\} \star \{S \mid S \subseteq f\}\}} \text{(Frame)}}{\models_{HSL} \{(\{S \mid S \cap p \neq \emptyset\} \star \{S \mid S \subseteq f\}) \otimes \llbracket \top \rrbracket\} C \{(\{S \mid S \cap q \neq \emptyset\} \star \{S \mid S \subseteq f\}) \otimes \llbracket \top \rrbracket\}} \text{(Join)}}{\models_{HSL} \{\llbracket \top \rrbracket\} C \{\llbracket \top \rrbracket\}} \text{(Triv-Post)}$$

Finally, using the fact that  $(\{S \mid S \cap p \neq \emptyset\} \star \{S \mid S \subseteq f\}) \otimes \llbracket \top \rrbracket = \{S \mid S \cap (p * f) \neq \emptyset\}$ , which is easily verified (see lemma `add_unconstrained` in `HyperStar.thy`), we conclude that

$$\frac{\models_{HSL} \{\text{ENC}_{SSIL}(p)\} C \{\text{ENC}_{SSIL}(q)\}}{\models_{HSL} \{\text{ENC}_{SSIL}(p * f)\} C \{\text{ENC}_{SSIL}(q * f)\}} \text{(Frame)(Triv-Post)(Join)}$$

- OSL

The rule of OSL for the base case hassertion

$$\frac{\models_{OSL} \{\{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\}\} C \{\{S \mid S \subseteq q \wedge S \cap q \neq \emptyset\}\}}{\models_{OSL} \{\{S \mid S \subseteq p * f \wedge S \cap (p * f) \neq \emptyset\}\} C \{\{S \mid S \subseteq q * f \wedge S \cap (q * f) \neq \emptyset\}\}} \text{(Frame)}$$

follows directly from HSL's frame rule and (**FR-OSL-1**), namely

$$\{S \mid S \subseteq p \wedge S \cap p \neq \emptyset\} \star \{S \mid S \subseteq f\} = \{S \mid S \subseteq p * f \wedge S \cap (p * f) \neq \emptyset\}.$$

The inductive steps incorporate properties ( $\cup$ -**distributivity**) and ( $\otimes$ -**distributivity**).

- RSL

This case is especially challenging as there are two major obstacles involved. The first one being that RSL (see [Yang 2007]) reasons about simultaneous divergence, which in essence is not  $\forall\forall$ -program hyperproperty. In this thesis we focus on an alternative of RSL, where simultaneous divergence is not considered, i.e. with the third condition in the definition of  $\models_{RSL}$  removed. If one is to capture RSL in its original  $\forall\forall$ - and  $\forall\exists$ -form, then the state model should further be expanded by adding  $\uparrow$  (divergence). The second obstacle arises from the fundamental difference between the types on which ours  $\star$  and theirs  $\boxtimes$  operate—hassertions and rassertions, respectively. That is, their  $\boxtimes$  can easily differentiate whether a heap is from the first or from the second coordinate, whereas ours has no way of differentiating that—there are no coordinates in the first place. The workaround we did was to encode in the store (in a variable  $t \notin \text{md}(C)$ ) the coordinate from which a state, particularly a heap, originates. This

solution, however, has the drawback that it can only speak of "rectangular" assertions, i.e. those which are cartesian product of two assertions, e.g.  $R \Leftarrow \{[\sigma_{21}^{11}], [\sigma_{22}^{12}]\}$  is not rectangular as  $R \subset \{\sigma_{11}, \sigma_{12}\} \times \{\sigma_{21}, \sigma_{22}\}$ . In this bullet point we set aside the first problem and focus on the  $\forall\forall$ -variant of RSL and give an initial idea of how a potential solution to the second problem may look like.

We begin by illustrating a simplified variant of the second problem and its solution. Recall the naive star

$$P \star'_{\text{naive}} Q \Leftarrow \{S_p * S_q \mid S_p \in P \wedge S_q \in Q\}.$$

It suffers a **similar** problem with rectangularness. Indeed, consider  $p \Leftarrow \{s, \emptyset_1^1\}, \langle s, \emptyset_2^2 \rangle$  and  $q \Leftarrow \{s, \emptyset_3^3\}, \langle s, \emptyset_4^4 \rangle$  for some fixed  $s$ . Then

$$\begin{aligned} \{S \mid S \subseteq p\} \star'_{\text{naive}} \{S \mid S \subseteq q\} &= \{\emptyset, & // 0 \times 0 \\ & \{s, \emptyset_{1,3}^{1,3}\}, \{s, \emptyset_{1,4}^{1,4}\}, \{s, \emptyset_{2,3}^{2,3}\}, \{s, \emptyset_{2,4}^{2,4}\}, \{s, \emptyset_{1,3}^{1,3}\}, & // 1 \times 1 \\ & \{s, \emptyset_{1,3}^{1,3}\}, \langle s, \emptyset_{1,4}^{1,4} \rangle, \{s, \emptyset_{2,3}^{2,3}\}, \langle s, \emptyset_{2,4}^{2,4} \rangle, & // 1 \times 2 \\ & \{s, \emptyset_{1,3}^{1,3}\}, \langle s, \emptyset_{2,3}^{2,3} \rangle, \{s, \emptyset_{1,4}^{1,4}\}, \langle s, \emptyset_{2,4}^{2,4} \rangle, & // 2 \times 1 \\ & \{s, \emptyset_{1,3}^{1,3}\}, \langle s, \emptyset_{1,4}^{1,4} \rangle, \langle s, \emptyset_{2,3}^{2,3} \rangle, \langle s, \emptyset_{2,4}^{2,4} \rangle\}. & // 2 \times 2 \end{aligned}$$

Notice that we have all  $0 \times 0$ ,  $1 \times 1$ ,  $1 \times 2$ ,  $2 \times 1$  and  $2 \times 2$  subsets of  $p * q$ , but miss the "serrated" subsets, i.e. those with cardinality 3. That is, we have

$$\{S \mid S \subseteq p\} \star'_{\text{naive}} \{S \mid S \subseteq q\} \subseteq \{S \mid S \subseteq p * q\}$$

and not the stronger version with equality. However, it is not an arbitrary subset. It is a subset, which contains the  $0 \times 0$  subset, i.e.  $\emptyset$ , and, more importantly, all  $1 \times 1$  subsets:

$$\forall \sigma \in p * q. \{\sigma\} \in \{S \mid S \subseteq p\} \star'_{\text{naive}} \{S \mid S \subseteq q\}.$$

That is, using HSL's frame rule and the consequence rule we obtain

$$\frac{\frac{\frac{\vdash_{\text{HSL}} \{\{S \mid S \subseteq p\}\} C \{\{S \mid S \subseteq q\}\}}{\vdash_{\text{HSL}} \{\{S \mid S \subseteq p\} \star'_{\text{naive}} \{S \mid S \subseteq f\}\} C \{\{S \mid S \subseteq q\} \star'_{\text{naive}} \{S \mid S \subseteq f\}\}}{\vdash_{\text{HSL}} \{\{\emptyset, \{\sigma\}\}\} C \{\{S \mid S \subseteq q * f\}\}} \quad (\text{Frame})}{\quad} \quad (\text{Cons}) ,$$

for any  $\sigma \in p * f$ . Now, in order to continue with the illustration, let's replace  $\star'_{\text{naive}}$  with  $\star$  in the reasoning we did, which can clearly be made. This change is made because we will need the soundness of the following rule

$$\frac{\text{for all } i \in I. \vdash_{\text{HSL}} \{P_i\} C \{Q_i\}}{\vdash_{\text{HSL}} \{\bigotimes_{i \in I} P_i\} C \{\bigotimes_{i \in I} Q_i\}} \quad (\text{Idx-Join})$$

However, we believe that this rule is actually unsound for  $\star'_{\text{naive}}$ , while for  $\star$ , the soundness is easily verified using the property ( **$\bigotimes$ -distributivity**)

(see `Indexed_join_rule_sound` in `HyperFrameRule.thy`). That is, we have obtained a weaker version of SL's frame rule, namely

$$\frac{\frac{\models_{HSL} \{\{S \mid S \subseteq p\}\} C \{\{S \mid S \subseteq q\}\}}{\models_{HSL} \{\{S \mid S \subseteq p\} \star \{S \mid S \subseteq f\}\} C \{\{S \mid S \subseteq q\} \star \{S \mid S \subseteq f\}\}} \text{(Frame)}}{\models_{HSL} \{\{\emptyset, \{\sigma\}\}\} C \{\{S \mid S \subseteq q \star f\}\}} \text{(Cons)},$$

for any  $\sigma \in p \star f$ . For the sake of the example, let's overlook the fact that SL's frame rule was already obtained before the application of the consequence rule. Therefore, by applying the (Idx-Join) rule, we obtain

$$\frac{\text{for all } \sigma \in p \star f. \models_{HSL} \{\{\emptyset, \{\sigma\}\}\} C \{\{S \mid S \subseteq q \star f\}\}}{\models_{HSL} \left\{ \left( \bigotimes_{\sigma \in p \star f} \{\emptyset, \{\sigma\}\} \right) \right\} C \left\{ \left( \bigotimes_{\sigma \in p \star f} \{S \mid S \subseteq q \star f\} \right) \right\}} \text{(Idx-Join)}.$$

Finally, note that the precondition  $\bigotimes_{\sigma \in p \star f} \{\emptyset, \{\sigma\}\} = \{S \mid S \subseteq p \star f\}$  and the postcondition  $\bigotimes_{\sigma \in p \star f} \{S \mid S \subseteq q \star f\} = \{S \mid S \subseteq q \star f\}$ . That is, we obtained the (encoding of the) conclusion of SL's frame rule. What we demonstrated is that, essentially, it suffices that  $\star$  yields all  $1 \times 1$  subsets, as the (Idx-Join) rule then allows us to obtain all subsets.

For the RSL's frame rule, we speculate that the same principle might work. That is, for  $t \notin \text{md}(C)$  and  $S_i \rightleftharpoons \{\langle s, h \rangle \in S \mid s(t) = i\}$ , by applying (**FR-RSL**), we obtain

$$\frac{\frac{\models_{HSL} \{\{S \mid S_1 \times S_2 \subseteq \mathbb{R}\}\} C \{\{S \mid S_1 \times S_2 \subseteq \mathbb{S}\}\}}{\models_{HSL} \{\{S \mid S_1 \times S_2 \subseteq \mathbb{R}\} \star \{S \mid S_1 \times S_2 \subseteq \mathbb{F}\}\} C \{\{S \mid S_1 \times S_2 \subseteq \mathbb{S}\} \star \{S \mid S_1 \times S_2 \subseteq \mathbb{F}\}\}} \text{(Frame)}}{\models_{HSL} \{\{\emptyset, \{\sigma_1, \sigma_2\}\}\} C \{\{S \mid S_1 \times S_2 \subseteq \mathbb{S} \boxtimes \mathbb{F}\}\}} \text{(Cons)},$$

for any  $\begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \in \mathbb{R} \boxtimes \mathbb{F}$ . The issue here is that the postcondition is not union closed, so the application of the (Join-Idx) rule may result in a weaker postcondition than the starting one. Whether this has a simple workaround, is a fundamental issue, or is a flaw in the definition of  $\star$  remains an open question.

## References

- Flavio Ascari, Roberto Bruni, Roberta Gori, and Francesco Logozzo. Sufficient incorrectness logic: Sil and separation sil, 2024. URL <https://arxiv.org/abs/2310.18156>.
- Callum Bannister, Peter Höfner, and Gerwin Klein. Backwards and forwards with separation logic. In Jeremy Avigad and Assia Mahboubi, editors, *Interactive Theorem Proving*, pages 68–87, Cham, 2018. Springer International Publishing. ISBN 978-3-319-94821-8.
- Nick Benton. Simple relational correctness proofs for static analyses and program transformations. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, page 14–25, New York, NY, USA, 2004. Association for Computing Machinery. URL <https://doi.org/10.1145/964001.964003>.
- E. M. Clarke, P. Aczel, J. V. Tucker, and J. C. Shepherdson. The characterization problem for hoare logics [and discussion]. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 312:423–440, 1984. URL <http://www.jstor.org/stable/37443>.
- Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *2008 21st IEEE Computer Security Foundations Symposium*, pages 51–65, 2008. URL <https://doi.org/10.1109/CSF.2008.7>.
- Stephen A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM J. Comput.*, 7:70–90, 1978. URL <https://api.semanticscholar.org/CorpusID:9829063>.
- Thibault Dardinier and Peter Müller. Hyper hoare logic: (dis-)proving program hyperproperties. *Proc. ACM Program. Lang.*, 8(PLDI), June 2024. URL <https://doi.org/10.1145/3656437>.
- J. A. Goguen and J. Meseguer. Security policies and security models. In *1982 IEEE Symposium on Security and Privacy*, pages 11–11, 1982. doi: 10.1109/SP.1982.10014.
- David Harel. *First-Order Dynamic Logic*, volume 68 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.
- C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, oct 1969. URL <https://doi.org/10.1145/363235.363259>.
- Peter W. O’Hearn. Incorrectness logic. *Proc. ACM Program. Lang.*, 4(POPL): 1–32, dec 2019. URL <https://doi.org/10.1145/3371078>.
- Azalea Raad, Josh Berdine, Hoang-Hai Dang, Derek Dreyer, Peter O’Hearn, and Jules Villard. *Local Reasoning About the Presence of Bugs: Incorrectness Separation Logic*, pages 225–252. Springer International Publishing, 07 2020. URL [https://doi.org/10.1007/978-3-030-53291-8\\_14](https://doi.org/10.1007/978-3-030-53291-8_14).

- J.C. Reynolds. Separation logic: a logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, pages 55–74, 2002. URL <https://doi.org/10.1109/LICS.2002.1029817>.
- Joseph Robert Shoenfield. *Mathematical Logic*. Addison-Wesley, 1967.
- Makoto Tatsuta, Wei-Ngan Chin, and Mahmudul Faisal Al Ameen. Completeness and expressiveness of pointer program verification by separation logic. *Information and Computation*, 267:1–27, 2019. ISSN 0890-5401. doi: <https://doi.org/10.1016/j.ic.2019.03.002>.
- Hongseok Yang. Relational separation logic. *Theoretical Computer Science*, 375(1):308–334, 2007. URL <https://doi.org/10.1016/j.tcs.2006.12.036>.
- Noam Zilberstein, Derek Dreyer, and Alexandra Silva. Outcome logic: A unifying foundation for correctness and incorrectness reasoning. *Proc. ACM Program. Lang.*, 7(OOPSLA1), April 2023. URL <https://doi.org/10.1145/3586045>.
- Noam Zilberstein, Angelina Saliling, and Alexandra Silva. Outcome separation logic: Local reasoning for correctness and incorrectness with computational effects. *Proc. ACM Program. Lang.*, 8(OOPSLA1), April 2024. URL <https://doi.org/10.1145/3649821>.