A Proof System for Separation Logic with Magic Wand

Wonyeol Lee    Sungwoo Park

July 2013
POSTECH-CSE-2013-7

Department of Computer Science & Engineering
Pohang University of Science and Technology
Republic of Korea, 790-784

Abstract

Separation logic is an extension of Hoare logic which is acknowledged as an enabling technology for large-scale program verification. It features two new logical connectives, separating conjunction and separating implication, but most of the applications of separation logic have exploited only separating conjunction without considering separating implication. Nevertheless the power of separating implication has been well recognized and there is a growing interest in its use for program verification. This paper develops a proof system for full separation logic which supports not only separating conjunction but also separating implication. The proof system is developed in the style of sequent calculus and satisfies the admissibility of cut. The key challenge in the development is to devise a set of inference rules for manipulating heap structures that ensure the completeness of the proof system with respect to separation logic. We show that our proof of completeness directly translates to a proof search strategy.
Keywords: Separation logic, Cut elimination, Proof system, Theorem prover
1 Introduction

Separation logic [28] is an extension of Hoare logic designed to simplify reasoning about programs manipulating mutable data structures with potential pointer aliasing. It features two new logical connectives, separating conjunction \( \star \) and separating implication \( \rightarrow \), whose semantics directly assumes memory heaps structured as a monoid. Separating conjunction allows us to describe properties of two disjoint heaps with a single logical formula: \( A \star B \) means that a given heap can be divided into two disjoint heaps satisfying \( A \) and \( B \) respectively. Separating implication, commonly known as magic wand, allows us to reason about hypothetical heaps extending a given heap: \( A \rightarrow \star B \) means that if a given heap is extended with a disjoint heap satisfying \( A \), the resultant heap satisfies \( B \). The use of the two separating connectives naturally leads to local reasoning in program verification in that we only need to reason locally about those heaps directly affected by the program.

So far, most of the applications of separation logic have exploited only separating conjunction. For example, all existing verification tools based on separation logic, such as Smallfoot [3], Space Invader [11], THOR [23], SLayer [1], HIP [25], jStar [12], Xisa [10], VeriFast [19], Infer [7], and Predator [14], use a decidable fragment by Berdine et al. [2] or its extension which provides only separating conjunction. By virtue of the principle of local reasoning, however, these tools are highly successful in their individual verification domains despite not using separating implication at all.

Although separating implication is not discussed as extensively as separating conjunction in the literature, its power in program verification has nevertheless been well recognized. Just around the inception of separation logic, Yang [29] already gives an elegant proof of the correctness of the Schorr-Waite algorithm which relies crucially on the use of separating implication in the main loop invariant. Krishnaswami [20] shows how to reason abstractly about an iterator protocol with separation logic by exploiting separating implication in the specification of iterators. Maeda et al. [22] adopt the idea of separating implication in extending an alias type system in order to express tail-recursive operations on recursive data structures. Recently Hobor and Villard [17] give a concise proof of the correctness of Cheney’s garbage collector in a proof system based on the ramify rule, a cousin of the frame rule of separation logic, whose premise checks a logical entailment involving separating implication. These promising results arguably suggest that introducing separating implication alone raises the level of technology for program verification as much as separation logic only with separating conjunction improves on Hoare logic.

Despite the potential benefit of separating implication in program verification, however, there is still no practical theorem prover for full separation logic. The state-of-the-art theorem provers for separation logic such as SeLoger [16] and SLP [24] support only separating conjunction, and the labelled tableau calculus by Galmiche and Méry [15] does not directly give rise to a proof search strategy. Because of the unavailability of such a theorem prover, all proofs exploiting separating implication should be manually checked, which can be time-consuming even with the help of lemmas provided by the proof system (as in [17]). Another consequence is that no existing verification tools based on separation logic can fully support backward reasoning by weakest precondition generation, which requires separating implication whenever verifying heap assignments (see Ishtiaq and O’Hearn [18]).

This paper develops a proof system \( \mathbb{P}_{SL} \) for full separation logic which supports not only separating conjunction but also separating implication. Its design is based on the principle of proof by contradiction from classical logic, and we develop its inference rules in the style of sequent calculus. \( \mathbb{P}_{SL} \) uses a new form of sequent, called world sequent, in order to give a complete description of the world of heaps, and its use of world sequents allows us to treat separating implication in the same way that it treats separating conjunction. The key challenge in the development of \( \mathbb{P}_{SL} \) is to devise a set of inference rules for manipulating heap structures so as to correctly analyze separating conjunction and separating implication. We show that \( \mathbb{P}_{SL} \) satisfies the admissibility of cut and that it is sound and complete with respect to separation logic. The proof of completeness directly translates to a proof search strategy, which is the basis for our prototype implementation of \( \mathbb{P}_{SL} \). We show that it is easy to extend \( \mathbb{P}_{SL} \) with new logical connectives and predicates, such as an overlapping conjunction \( A \cup B \) by Hobor and Villard [17].

Separating implication has been commonly considered to be much harder to reason about than separating conjunction, as partially evidenced by lack of theorem provers supporting separating implication and abundance of verification systems supporting separating conjunction. Our development of \( \mathbb{P}_{SL} \), however, suggests that a proof system designed in a principled way can support both logical connectives in a coherent way without requiring distinct treatments. Our prototype implementation of
\( P_{SL} \) also suggests that such a proof system can develop into a practical theorem prover for separation logic. To the best of our knowledge, \( P_{SL} \) is the first proof system for full separation logic that satisfies the admissibility of cut and provides a concrete proof search strategy.

This paper is organized as follows. Section 2 gives preliminaries on separation logic. Section 3 develops our proof system \( P_{SL} \) and Section 4 gives three examples of proving world sequents. Section 6 proves the soundness and completeness of \( P_{SL} \) with respect to separation logic and Section 7 discusses the implementation and extension of \( P_{SL} \). Section 8 discusses related work and Section 9 concludes.

2 Semantics of separation logic

Separation logic extends classical first-order logic with multiplicative formulas from intuitionistic linear logic:

<table>
<thead>
<tr>
<th>formula</th>
<th>( A, B, C )</th>
<th>:=</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \wedge B )</td>
<td>( P \mid \perp \mid \neg A \mid A \lor A \mid )</td>
<td></td>
</tr>
</tbody>
</table>

| primitive formula | \( P \) | := |
| expression | \( E \) | := |
| location expression | \( l \) | := |
| value | \( V \) | := |
| stack variable | \( l_1, l_2, l_3, \ldots \) | := |
| local variable | \( a, b, c \) | := |

\( \perp, \neg A, A \lor B, \) and \( \exists a.A \) are from classical first-order logic. \( I \) is the multiplicative unit. \( A \star B \) is a separating conjunction and \( A \rightarrow B \) is a separating implication. We define \( \top = \neg \perp \), \( A \land B = \neg (\neg A \lor \neg B) \), and \( A \cup B = \neg A \lor B \). We use conventional precedence rules for logical connectives: \( \neg > \star > \lor > \land > \top \). In this work, we do not consider inductively defined predicates.

Primitive formulas include a points-to relation \( [l \mapsto E] \) for describing a singleton heap. All other primitive formulas describe relations between expressions; for simplicity, we consider only an equality relation \( E = E' \). Expressions denote values which include locations \( L \). Location expressions are a special class of expressions that denote locations. In the present work, we allow only locations as values, but it should be straightforward to introduce additional forms of expressions for new types of values such as booleans and integers. We syntactically distinguish between stack variables which originate from the program being verified (and thus may be called global variables) and local variables which are introduced by existential quantifiers (and thus can never appear outside corresponding existential formulas).

We specify the semantics of separation logic with respect to a stack and a heap. A stack \( S \) is a finite partial mapping \( \text{Var} \rightarrow \text{Val} \) from stack variables to values where \( \text{Var} \) denotes the set of stack variables and \( \text{Val} \) denotes the set of values. Given a stack \( S \), we can determine a unique value for every expression \( E \), which we write as \( [E]_S \). A heap \( H \) is a finite partial mapping \( \text{Loc} \rightarrow \text{Val} \) from locations to values where \( \text{Loc} \) denotes the set of locations. We write \( H_1 \odot H_2 \) to mean that heaps \( H_1 \) and \( H_2 \) are disjoint, i.e., \( \text{dom}(H_1) \cap \text{dom}(H_2) = \emptyset \). We write \( H_1 \odot H_2 \) for the union of disjoint heaps \( H_1 \) and \( H_2 \) where \( H_1 \odot H_2 \) is assumed, and \( \epsilon \) for an empty heap. Heaps form a commutative cancellative monoid with \( \odot \) as the associative operation and \( \epsilon \) as the identity:

| (neutrality) | \( H \odot \epsilon = H \) |
| (commutativity) | \( H_1 \odot H_2 = H_2 \odot H_1 \) |
| (associativity) | \( H_1 \odot (H_2 \odot H_3) = (H_1 \odot H_2) \odot H_3 \) |
| (cancellativity) | \( H \odot H_1 = H \odot H_2 \) implies \( H_1 = H_2 \) |

Given a stack \( S \) and a heap \( H \), we obtain the semantics of separation logic from the satisfaction relation \( (S, H) \models A \) for formulas defined as follows:

- \( (S, H) \models [l \mapsto E] \) iff. \( H = [l]_S \rightarrow [E]_S \), i.e., \( H \) is a singleton heap mapping \([l]_S \) to \([E]_S \).
- \( (S, H) \models E = E' \) iff. \([E]_S = [E']_S \).
- \( (S, H) \models \perp \) iff. never.
We observe that the definition for separating implication is symmetric to the definition for separating local variables.

Although the satisfaction relation \((S, H) \models A\) is enough for specifying the semantics of separation logic, we deliberately derive the definition of its negation \((S, H) \not\models A\), which plays an equally important role in the development of our proof system:

\[
\begin{align*}
(S, H) &\not\models [l \mapsto E] \iff H \not\models ([[l]_S \mapsto [E]_S]), \text{i.e., } \text{dom}(H) \neq \{[[l]_S]\} \text{ or } H([[l]_S]) \neq [E]_S. \\
(S, H) &\not\models E = E' \iff [E]_S \neq [E']_S. \\
(S, H) &\not\models \perp \iff \text{always}. \\
(S, H) &\not\models \neg A \iff (S, H) \not\models A. \\
(S, H) &\not\models A \lor B \iff (S, H) \not\models A \text{ and } (S, H) \not\models B. \\
(S, H) &\not\models I \iff \text{dom}(H) \neq \emptyset, \text{i.e., } H \neq \epsilon. \\
(S, H) &\not\models A \ast B \iff H = H_1 \circ H_2 \text{ implies } (S, H_1) \not\models A \text{ or } (S, H_2) \not\models B \text{ for any heaps } H_1 \text{ and } H_2. \\
(S, H) &\not\models A \rightarrow B \iff H_2 = H \circ H_1 \text{ and } (S, H_1) \models A \text{ and } (S, H_2) \not\models B \text{ for some heaps } H_1 \text{ and } H_2. \\
(S, H) &\not\models \exists a. A \iff (S, H) \not\models [V/a]A \text{ for some value } V.
\end{align*}
\]

We observe that the function for separating implication is symmetric to the definition for separating conjunction:

\[
\begin{align*}
(S, H) \models A \ast B &\text{ should find a certain pair of heaps whereas } (S, H) \not\models A \ast B \text{ should analyze an unspecified pair of heaps.} \\
(S, H) \models A \rightarrow B &\text{ should analyze an unspecified pair of heaps whereas } (S, H) \not\models A \rightarrow B \text{ should find a certain pair of heaps.}
\end{align*}
\]

This symmetry suggests that we can incorporate separating implication into the proof system in an analogous way to separating conjunction.

A formula \(A\) is valid, written \(\models A\), if \((S, H) \models A\) holds for every stack \(S\) and heap \(H\). Our proof system \(P_{SL}\) can check the validity of every formula in separation logic.

## 3 Proof system \(P_{SL}\) for separation logic

This section presents the proof system \(P_{SL}\) for separation logic which is developed in the style of sequent calculus. We first explain world sequents, the main judgment in \(P_{SL}\), and then present the inference rules.

### 3.1 World sequents

The design of \(P_{SL}\) is based on the principle of proof by contradiction from classical logic. We describe the state of each heap with a set of true formulas and another set of false formulas. A world sequent in \(P_{SL}\) gives a description of the entire world of heaps, and a derivation of it means that the description is self-contradictory. Hence, in order to check the validity of a formula in separation logic, we use it as a false formula about an arbitrary heap \(w\) (about which nothing is known) and attempt to produce a logical contradiction by proving a world sequent consisting solely of heap \(w\). The definition of world sequents and the principle of proof by contradiction are inherited from the nested sequent calculus for Boolean BI by Park et al. [26].
Since $P_{SL}$ is designed to check the validity of a formula, it assumes an arbitrary stack, which implies that every stack variable denotes an arbitrary value. This in turn implies that in a derivation of a world sequent, we may use a fresh stack variable to denote an arbitrary value. We exploit this interpretation of stack variables in an inference rule for first-order formulas.

A world sequent consists of expression relations $\Theta$, heap relations $\Sigma$, and heap sequents $\Pi$:

- An expression relation $\theta$ is an equality or inequality between two expressions. If we introduce new forms of primitive formulas (e.g., $E < E'$), we should introduce corresponding forms of expression relations.

- We assign a heap variable to each heap, and a heap relation $\sigma$ relates a heap to an empty heap ($w = \epsilon$ and $w \neq \epsilon$), a singleton heap ($w = [l \mapsto E]$ and $w \neq [l \mapsto E]$), or the union of two disjoint heaps ($w = w_1 \cup w_2$). We refer to those heap relation involving an empty heap or a singleton heap as atomic heap relations. As heaps form a commutative (cancellative) monoid, we assume commutativity of $\circ$ and use $w_1 \cup w_2$ and $w_2 \cup w_1$ interchangeably.

- A heap sequent $[\Gamma \Rightarrow \Delta]^w$ describes heap $w$ with truth context $\Gamma$ and falsehood context $\Delta$ which contain true formulas and false formulas, respectively, about heap $w$.

In this way, a world sequent $\Theta; \Sigma \parallel \Pi$ gives a complete description of the world of heaps. We require that no local variable appear in expression relations and heap relations, and that a world sequent contain a unique heap sequent for each heap variable.

A world sequent represents a graph of heaps induced by heap relations. Given a heap relation $w = w_1 \cup w_2$, we say that parent heap $w$ has two child heaps $w_1$ and $w_2$ which are sibling heaps to each other. We can also extend parent-child relations to derive ancestor-descendant relations. If a heap has no pair of child heaps, we call it a terminal heap (where we ignore such a heap relation as $w = w \circ w$, with $w_1 = \epsilon$); otherwise we call it a non-terminal heap. Note that a heap relation $w = \epsilon$ or $w = [l \mapsto E]$ does not immediately mean that $w$ is a terminal heap because we may have another heap relation $w = w_1 \cup w_2$. $P_{SL}$, however, allows us to normalize all heap relations and turn $w$ into a terminal heap.

$P_{SL}$ also uses an expression contradiction judgment $\Theta \vdash \bot$ which is an abbreviation of a particular form of a world sequent $\Theta; \parallel$ and means that expression relations $\Theta$ produce a logical contradiction. Since the definition of expression relations is extensible, we do not give inference rules for the expression contradiction judgment and just assume a decidable system for it. For simplicity, we write $\Theta \vdash E = E'$ for $\Theta, E \neq E' \vdash \bot$, and $\Theta \vdash E \neq E'$ for $\Theta, E = E' \vdash \bot$. We write $\Theta \vdash [l \mapsto E] = [l' \mapsto E']$ for $\Theta \vdash l = l'$ and $\Theta \vdash E = E'$, and $\Theta \vdash [l \mapsto E] \neq [l' \mapsto E']$ for $\Theta \vdash l \neq l'$ or $\Theta \vdash E \neq E'$.

$P_{SL}$ consists of logical rules in Figure 1, structural rules in Figure 2, and heap contradiction rules in Figure 3. The logical rules deal with formulas in heap sequents $\Pi$, the structural rules reorganize graphs of heaps induced by heap relations $\Sigma$, and the heap contradiction rules detect logical contradictions in heap relations $\Sigma$, or heap contradictions. $P_{SL}$ shares the logical rules (for propositional and multiplicative formulas) with the nested sequent calculus for Boolean BI in [26], but the structural rules and the heap contradiction rules are specific to separation logic. We read every inference rule from the conclusion to the premise, and the derivation of a world sequent always terminates with a proof of a logical contradiction. Hence, in order to show the validity of a formula $A$, we try to prove a world sequent $\vdash \bot$.
3.2 Logical rules of $P_{SL}$

Figure 1 shows the logical rules of $P_{SL}$. Except for the rule ExpCont, a logical rule focuses on a principal formula in a heap sequent and either produces a logical contradiction (in the rule $\bot$) or rewrites the world sequent of the conclusion according to the semantics of separation logic in Section 2. For each type of formulas, $P_{SL}$ has both a left rule, which analyzes a true formula about a heap, and a right rule, which analyzes a false formula about a heap, as in a typical sequent calculus. The rules for points-to relations introduce a corresponding heap relation. The rules for propositional and first-order formulas are from first-order classical logic. In the rule $\exists L$, the fresh stack variable $x$ denotes an arbitrary value. In the rules $\exists L$ and $\exists R$, we write $[E/a]A$ for substituting expression $E$ for local variable $x$ in formula $A$. The rules $\exists L$ and $\exists R$ are the only rules that add expression relations, and the rule ExpCont checks if expression relations $\Theta$ produce a logical contradiction.

The rules $IL$ and $IR$ use the fact the $\bot$ is true only at an empty heap. The rules $\star L$ and $\star R$ are based on the following interpretation of multiplicative conjunction $\star$ which closely makes the semantics of separation logic in Section 2:

- $A \star B$ is true at heap $w$ iff. $w \models w_1 \circ w_2$ and $A$ is true at heap $w_1$ and $B$ is true at heap $w_2$ for some heaps $w_1$ and $w_2$.
- $A \star B$ is false at heap $w$ iff. $w \not\models w_1 \circ w_2$ implies that $A$ is false at heap $w_1$ or that $B$ is false at heap $w_2$ for any heaps $w_1$ and $w_2$.

Hence the rule $\star L$ creates (some) fresh child heaps $w_1$ and $w_2$, whereas the rule $\star R$ chooses (any) existing child heaps $w_1$ and $w_2$. Similarly the rules $\rightarrow L$ and $\rightarrow R$ are based on the following interpretation of multiplicative implication $\rightarrow$:

- $A \rightarrow B$ is true at heap $w$ iff. $w \models w_2 \circ w_1$ implies that $A$ is false at heap $w_1$ or that $B$ is true at heap $w_2$ for any heaps $w_1$ and $w_2$.
- $A \rightarrow B$ is false at heap $w$ iff. $w \not\models w_2 \circ w_1$ and $A$ is true at heap $w_1$ and $B$ is false at heap $w_2$ for some heaps $w_1$ and $w_2$.

Hence the rule $\rightarrow L$ chooses (any) existing sibling heap $w_1$ and parent heap $w_2$, whereas as the rule $\rightarrow R$ creates (some) fresh sibling heap $w_1$ and parent heap $w_2$. The rules $\star L$ and $\star R$ are the only logical rules that add parent-child heap relations to extend the graph of heaps, and introduce fresh heap variables $w_1$ and $w_2$ that are not found in the world sequent in the conclusion. The rules $\star R$ and $\rightarrow L$ are the only logical rules that replicate the principal formula into world sequents in the premise.

In the rules $\star R$ and $\rightarrow L$, we allow equalities between heap variables $w_1$, $w_2$, and $w$. Since an equality between these heap variables invalidates the requirement that a world sequent contain a unique heap sequent for each heap variable, we interpret heap sequents for the same heap variable in the rules $\star R$ and $\rightarrow L$ as follows:

- In the conclusion, we implicitly replicate the same heap sequent as necessary.
- In the premise, we combine all changes made to individual heap sequents for the same heap variable to produce a single heap sequent.

For example, an equality $w = w_1$ in the rule $\star R$ yields the following special instance:

\[
\frac{\Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta, A \star B, A]^w, [\Gamma_2 \implies \Delta_2]^w_2}{w \models w \circ w_2 \in \Sigma} \quad \frac{\Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta, A \star B]^w, [\Gamma_2 \implies \Delta_2]^w_2}{\Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta, A \star B]^w, [\Gamma_2 \implies \Delta_2]^w_2}
\]

The rule $\star R$ has two more special instances (corresponding to heap relations $w \models w_1 \circ w_1$ and $w \models w \circ w$), and similarly the rule $\rightarrow L$ has a total of three special instances.

Now we can decompose each individual formula by applying its corresponding logical rule, thus accumulating expression relations and heap relations and creating fresh heaps. When expression relations become self-contradictory, we apply the rule ExpCont to complete the proof search. In order to obtain a complete proof search strategy, however, we should also be able to: 1) enumerate all heap relations $w \models w_1 \circ w_2$ and $w_2 \models w \circ w_1$ for a given heap $w$ for the rules $\star R$ and $\rightarrow L$; 2) produce heap
Rules for points-to relations:
\[
\Theta; \Sigma, w \doteq [l \mapsto E] \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow \Theta; \Sigma, w \nmid [l \mapsto E] \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma \Longrightarrow \Delta, [l \mapsto E]] \Rightarrow \rightarrow R
\]

Rules for propositional formulas:
\[
\begin{align*}
\Theta; \Sigma \parallel \Pi, [\Gamma, \bot \Longrightarrow \Delta] \Rightarrow & L \\
\Theta; \Sigma \parallel \Pi, [\Gamma, \neg A \Longrightarrow \Delta] \Rightarrow & R \\
\Theta; \Sigma \parallel \Pi, [\Gamma, A \Longrightarrow \Delta] \Rightarrow & L \\
\Theta; \Sigma \parallel \Pi, [\Gamma, B \Longrightarrow \Delta] \Rightarrow & R \\
\Theta; \Sigma \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow & L \\
\Theta; \Sigma \parallel \Pi, [\Gamma \Longrightarrow \Delta, A \lor B] \Rightarrow & R
\end{align*}
\]

Rules for multiplicative formulas:
\[
\begin{align*}
\Theta; \Sigma, w \doteq \epsilon \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma, \epsilon \Longrightarrow \Delta] \Rightarrow \rightarrow L \\
\Theta; \Sigma, w \doteq \epsilon \parallel \Pi, [\Gamma \Longrightarrow \Delta, A \lor B] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma, \epsilon \Longrightarrow \Delta, A \lor B] \Rightarrow \rightarrow R
\end{align*}
\]

Rules for first-order formulas:
\[
\begin{align*}
\text{fresh } x & \quad \Theta; \Sigma \parallel \Pi, [\Gamma, [x/a]A \Longrightarrow \Delta] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma, A \Longrightarrow \Delta] \Rightarrow \rightarrow L \\
\text{fresh } w_1, w_2 & \quad \Theta; \Sigma, w \doteq w_1 \circ w_2 \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma, w_1 \circ w_2 \Longrightarrow \Delta] \Rightarrow \rightarrow L \\
\text{fresh } w_1, w_2 & \quad \Theta; \Sigma, w \doteq w_1 \circ w_2 \parallel \Pi, [\Gamma \Longrightarrow \Delta, A \Longrightarrow B] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma, w_1 \circ w_2 \Longrightarrow \Delta, A \Longrightarrow B] \Rightarrow \rightarrow R
\end{align*}
\]

Rules for primitive formulas for expressions:
\[
\begin{align*}
\Theta; E = E'; \Sigma \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow L \\
\Theta; E \neq E'; \Sigma \parallel \Pi, [\Gamma \Longrightarrow \Delta] \Rightarrow \Theta; \Sigma \parallel \Pi, [\Gamma \Longrightarrow \Delta, E = E'] \Rightarrow R \\
\Theta; \bot \Rightarrow \Theta \parallel \Pi \Rightarrow \text{ExpCont}
\end{align*}
\]

Figure 1: Logical rules in the proof system PSL for separation logic
contradictions, for example, from $w = \epsilon$ and $w = [l \mapsto E]$. (We assume that we can make a correct guess on expression $E$ in the rule $\exists R$.) The remaining challenge is to devise a set of structural rules and another set of heap contradiction rules satisfying these two requirements, which would enable us to enumerate all feasible heap relations from those generated by the logical rules and detect all types of heap contradictions.

3.3 Structural rules of $P_{SL}$

The structural rules of $P_{SL}$ are divided into five groups according to their roles in reorganizing graphs of heaps represented by world sequents. The order of the structural rules in Figure 2 roughly follows their use in the proof of the completeness of $P_{SL}$ with respect to separation logic (Theorem 6.4). In a certain sense, we design the structural rules so as to obtain a complete proof search strategy for separation logic when the logical rules are already given as in Figure 1. Below we informally discuss the key properties of the structural rules, which we formally present as part of the proof of the completeness of $P_{SL}$ in Section 6.2.

3.3.1 Rules for disambiguating heap relations

The rules $Disj^{\star}$ and $Disj^{-\star}$ disambiguate heap relations in order to leave only disjoint terminal heaps. Roughly speaking, two heaps are disjoint if they share a common ancestor which has a heap relation separating them.

In the premise of the rule $Disj^{\star}$, child heaps $u_i$ and $v_j$ ($i, j = 1, 2$) share a common parent heap $w$, but their exact relations are unknown. For example, heap $u_1$ may completely subsume, partially overlap with, or be disjoint from heap $v_1$. In general, each pair of child heaps $u_i$ and $v_j$ are allowed to share a common child heap, so the rule $Disj^{\star}$ disambiguates their relations by introducing four fresh terminal heaps, $w_1$ to $w_4$, which are all disjoint from each other:

$$u_1 \triangleright u_2 \triangleright v_1 \triangleright v_2 \quad \Rightarrow \quad u_1 \triangleright u_2 \triangleright v_1 \triangleright v_2 \triangleright w_1 \triangleright w_2 \triangleright w_3 \triangleright w_4$$

Now, for example, we may assume that the intersection of heaps $u_1$ and $v_1$ is represented by heap $w_1$. Note that if heap $u_i$ or $v_j$ is not a terminal heap, the rule $Disj^{\star}$ gives rise to unknown relations between the existing child heaps of $u_i$ or $v_j$ and two of the fresh terminal heaps. The rule $Disj^{\star}$ corresponds to the cross-split axiom for separation algebras [13].

The rule $Disj^{-\star}$ disambiguates relations between two sibling heaps $u_1$ and $u_3$ of heap $u_2$ by introducing three fresh terminal heaps, $v_1$ to $v_3$, which are all disjoint from each other:

$$u_1 \triangleright u_2 \triangleright u_3 \quad \Rightarrow \quad u_1 \triangleright u_2 \triangleright u_3 \triangleright v_1 \triangleright v_2 \triangleright v_3$$

Similarly to the rule $Disj^{\star}$, the rule $Disj^{-\star}$ may give rise to new unknown relations involving heap $w_1$, $w_2$, $u_1$, or $u_3$. Unlike the rule $Disj^{\star}$, however, it also creates a common ancestor $w$ of all heaps. Otherwise the fresh terminal heaps themselves come to have unknown relations, thereby defeating the purpose of applying the rule $Disj^{-\star}$.

Thus the rule $Disj^{\star}$ eliminates unknown relations between child heaps and the rule $Disj^{-\star}$ eliminates unknown relations between sibling heaps of a certain heap, both potentially creating similar unknown
Rules for disambiguating heap relations and leaving only disjoint terminal heaps:

\[
\begin{aligned}
\{ w \doteq u_1 \circ u_2, w \doteq v_1 \circ v_2 \} & \subset \Sigma \quad \text{fresh} \ w_1, w_2, w_3, w_4 \quad \Theta; \Sigma, \quad v_2 \doteq w_2 \circ w_4 \quad \Pi, \quad \vdash \quad \text{Disj*}
\end{aligned}
\]

\[
\begin{aligned}
\{ w_1 \doteq u_1 \circ u_2, w_2 \doteq u_2 \circ u_3 \} & \subset \Sigma \quad \text{fresh} \ w_1, v_1, v_2, v_3 \quad \Theta; \Sigma, \quad u_3 \doteq v_2 \circ v_3 \quad \Pi, \quad \vdash \quad \text{Disj*}
\end{aligned}
\]

Rules for applying associativity of the union of disjoint heaps.

\[
\begin{aligned}
\{ w \doteq u \circ v, u \doteq u_1 \circ u_2 \} & \subset \Sigma \quad \text{fresh} \ u', \quad \Theta; \Sigma, \quad u' \doteq u_2 \circ v, w \doteq u_1 \circ u' \quad \Pi, \quad \vdash \quad \text{Assoc}
\end{aligned}
\]

Rules for propagating atomic heap relations:

\[
\begin{aligned}
\{ w \doteq w \doteq w_1 \circ w_2 \} & \subset \Sigma \quad \Theta; \Sigma, w_1 \doteq \epsilon, w_2 \doteq \epsilon \quad \Pi \quad \vdash \quad \text{Prop}\epsilon
\end{aligned}
\]

\[
\begin{aligned}
\{ w \doteq [l \mapsto E], w \doteq w_1 \circ w_2 \} & \subset \Sigma \quad \Theta; \Sigma, w_1 \doteq [l \mapsto E], w_2 \doteq \epsilon \quad \Pi \quad \vdash \quad \text{Prop}\mapsto
\end{aligned}
\]

\[
\begin{aligned}
\{ w \neq \epsilon, w \doteq w_1 \circ w_2 \} & \subset \Sigma \quad \Theta; \Sigma, w_1 \neq \epsilon \quad \Pi \quad \vdash \quad \text{Prop}\neq
\end{aligned}
\]

Rules for normalizing heap relations:

\[
\begin{aligned}
\Theta; [w / w'] \Sigma, w \doteq u \circ v \quad \Pi, \quad [\Gamma, \Gamma' \mapsto \Delta, \Delta']^w & \quad \text{NormEq}
\end{aligned}
\]

\[
\begin{aligned}
\Theta; \Sigma, w \doteq u \circ v, w' \doteq u \circ v \quad \Pi, \quad [\Gamma \mapsto \Delta, \Gamma' \mapsto \Delta']^w & \quad \text{NormPC}
\end{aligned}
\]

\[
\begin{aligned}
\Theta; [w / w'] \Sigma, w \doteq \epsilon \quad \Pi, \quad [\Gamma, \Gamma' \mapsto \Delta, \Delta']^w & \quad \text{NormEmpty}
\end{aligned}
\]

Rules for creating an empty heap and applying the monoid laws for empty heaps:

\[
\begin{aligned}
\text{fresh} \ w_e \quad \Theta; \Sigma, w_e \doteq \epsilon \quad \Pi, \quad [\vdash \epsilon]^w & \quad \text{EJoin} \quad \Theta; \Sigma, w \doteq w \circ w_e \quad \Pi \quad \vdash \quad \text{ENew}
\end{aligned}
\]

\[
\begin{aligned}
w \doteq w \circ u \doteq \Sigma \quad \Theta; \Sigma, u \doteq \epsilon \quad \Pi \quad \vdash \quad \text{ECancel}
\end{aligned}
\]

Figure 2: Structural rules in the proof system $P_{SL}$ for separation logic
For example, we negate the second clause to derive the rule \( w \) which makes it much easier to develop a complete procedure for producing heap contradictions. From atomic heap relations, we need to inspect only terminal heaps of these graphs, however, we may safely discard heap relation \( \sigma \). Note that although the new heap relations for the child heaps \( w \) in the premise:

3.3.2 Rule for applying associativity of \( \circ \)

The rule Assoc creates new heap relations according to associativity of the union of disjoint heaps. Suppose that we have two heap relations \( w \doteq u \circ v \) and \( u \doteq u_1 \circ u_2 \). The rule Assoc introduces a fresh heap \( u' \) in order to associate two heaps \( u_2 \) and \( v \) which are known to be disjoint but do not have a common parent heap yet; it also assigns heap \( w \) as the common parent heap of heaps \( u_1 \) and \( u' \):

Note that unlike the rules Disj* and Disj→, the rule Assoc creates no fresh terminal heaps.

The rule Assoc is crucial for enumerating all heap relations involving a particular heap. The basic observation is that by repeatedly applying the rule Assoc to a graph of heaps, we can eventually obtain another graph of heaps with the same set of terminal heaps such that for each combination of terminal heaps, there is at least one heap subsuming exactly these terminal heaps and no others. By starting with a graph of heaps obtained by repeatedly applying the rules Disj* and Disj→, then, we can enumerate all feasible heap relations \( w \doteq w_1 \circ w_2 \) and \( w_2 \doteq w \circ w_1 \) for a particular heap \( w \) where we assume that heaps \( w_1 \) and \( w_2 \) are in the graph. For the case that \( w_1 \) or \( w_2 \) is an empty heap, however, we need another set of structural rules for dealing with empty heaps. We should also combine heap sequents for the same heap. (Hence we have not yet accomplished the first requirement for obtaining a complete proof search strategy.)

3.3.3 Rules for propagating atomic heap relations

The rules for propagating atomic heap relations, or propagation rules, are designed to propagate all atomic heap relations \( (w \doteq \epsilon, w \neq \epsilon, w \doteq [l \mapsto E], w \neq [l \mapsto E]) \) from non-terminal heaps to terminal heaps. A propagation rule converts an atomic heap relation for a heap \( w \) into semantically equivalent heap relations for its child heaps \( w_1 \) and \( w_2 \) (with \( w \doteq w_1 \circ w_2 \)). It rewrites world sequents according to the following fact on atomic heap relations where we assume \( w \doteq w_1 \circ w_2 \):

- \( w \doteq \epsilon \) iff. \( w_1 \doteq \epsilon \) and \( w_2 \doteq \epsilon \) (for the rules Prop\( \epsilon \) and Prop\( \epsilon \neq \)).
- \( w \doteq [l \mapsto E] \) iff. either \( w_1 \doteq [l \mapsto E] \) and \( w_2 \doteq \epsilon \), or \( w_1 \doteq \epsilon \) and \( w_2 \doteq [l \mapsto E] \) (for the rules Prop\( \mapsto \) and Prop\( \mapsto \neq \)).

For example, we negate the second clause to derive the rule Prop\( \mapsto \neq \) which has four world sequents in the premise:

- \( w \neq [l \mapsto E] \) iff. 1) \( w_1 \neq [l \mapsto E] \) and \( w_1 \neq \epsilon \); 2) \( w_1 \neq [l \mapsto E] \) and \( w_2 \neq [l \mapsto E] \); 3) \( w_2 \neq \epsilon \) and \( w_1 \neq \epsilon \); or 4) \( w_2 \neq \epsilon \) and \( w_2 \neq [l \mapsto E] \).

Note that although the new heap relations for the child heaps \( w_1 \) and \( w_2 \) collectively imply the original heap relation \( \sigma \), we have to preserve \( \sigma \) in every world sequent of the premise because it may still interact with another pair of child heaps \( w_1' \) and \( w_2' \) (with \( w \doteq w_1' \circ w_2' \)). After considering all such interactions, however, we may safely discard \( \sigma \) (by the rule Weaken to be introduced in Section 6.2).

The propagation rules are the first step toward a complete procedure for producing heap contradictions (which detect all types of heap contradictions). Suppose that we repeatedly apply the propagation rules until no more new heap relations arise from atomic heap relations. After discarding atomic heap relations for non-terminal heaps, we obtain a set of graphs of heaps (with the same structure as the original graph) in which atomic heap relations reside only for terminal heaps. Now, in order to produce heap contradictions from atomic heap relations, we need to inspect only terminal heaps of these graphs, which makes it much easier to develop a complete procedure for producing heap contradictions.
3.3.4 Rules for normalizing heap relations

The rules for normalizing heap relations, or normalization rules, merge two identical heaps and isolate empty heaps while simultaneously shrinking the graph of heaps. In the rule `NormEq`, heaps \( w \) and \( w' \) are identical and we merge the two heaps by combining their heap sequents. Here we write \([w/w']\Sigma\) for substituting \( w \) for \( w' \) in every heap relation in \( \Sigma \). Note that the rule `NormEq` implies that \( \circ \) is (partial) deterministic. In the rule `NormPC`, \( v \triangleq \epsilon \) implies that heaps \( w \) and \( u \) are identical. Hence we merge the two heaps by combining their heap sequents and isolate the empty heap \( v \) from the graph of heaps. Similarly, the rule `NormEmpty` merges two empty heaps \( w \) and \( u \) by combining their heap sequents. In effect, it allows us to collect all empty heaps, which do not need to be distinguished for the purpose of proof search, into a single empty heap. Note that the rule `NormEmpty` implies the existence of a single unit of \( \circ \). By repeatedly applying the normalization rules to a graph of heaps, we can eventually obtain an equivalent graph which maintains a unique world sequent for each heap and possibly a unique empty heap isolated from the graph.

It is important that the normalization rules shrink the graph of heaps, but preserve all the properties established by the previous structural rules. For example, if the graph satisfies the property that all terminal heaps are disjoint (established by the rules `Disj*` and `Disj→`), it continues to satisfy the same property after an application of any normalization rule. Hence it is safe to aggressively apply the normalization rules after applying the previous structural rules.

3.3.5 Rules for dealing with empty heaps

The last group of structural rules create an empty heap and apply the monoid laws for empty heaps. We use the rule `ENew` when no rule can directly produce an empty heap. The rule `EJoin`, which is based on neutrality of \( \epsilon \), is sound because extending a heap with an empty heap makes no change. The rule `ECancel` creates an empty heap when a heap is shown to be a child heap of itself. It is based on cancellativity of \( \circ \): we can always generate \( w \triangleq w \circ w \), and \( w \triangleq \epsilon \) by the rules `ENew` and `EJoin`, and \( w \triangleq w \circ u \) and \( w \triangleq w \circ w \) imply \( u \triangleq \epsilon \) by cancellativity of \( \circ \). (Similarly the rule `NormPC` is based on cancellativity of \( \circ \): we can always generate \( w \triangleq w \circ w \) by the rule `EJoin`, and \( w \triangleq u \circ v \) and \( w \triangleq w \circ \epsilon \) imply \( w = u \)). It turns out that we need the rule `ECancel` for the proof of admissibility of cut (Theorem 5.1), but not for the proof of the completeness of \( P_{\text{SL}} \) (Theorem 6.4).

Now we can accomplish the first requirement for obtaining a complete proof search strategy. In order to enumerate all heap relations \( w \triangleq w_1 \circ w_2 \) and \( w_2 \triangleq w \circ w_1 \) for a heap \( w \), we first analyze the graph of heaps obtained by repeatedly applying the previous structural rules. This produces all such heap relations that involve only non-empty heaps. Then we apply the rule `EJoin` as necessary to produce all such heap relations that involve empty heaps.

We may think of the rule `EJoin` as extending heap relations for heap \( w \) with a pair of child heaps \( w \) and \( w_e \), or a pair of sibling heap \( w_e \), and parent heap \( w \). It is the only rule in \( P_{\text{SL}} \) that is capable of creating new heap relations for an arbitrary heap. Thus, whenever an arbitrary heap with no heap relation needs a pair of child heaps or a pair of sibling and parent heaps, we should apply the rule `EJoin` which inevitably reuses an existing empty heap. For example, we prove the validity of \( \top \star \top \) as follows:

\[
\begin{align*}
&; w_e \triangleq \epsilon, w \triangleq w \circ w_e \parallel [\downarrow \implies \top \star \top]^w, [\downarrow \implies \top]^w_\epsilon \quad \downarrow \text{L} \\
&; w_e \triangleq \epsilon, w \triangleq w \circ w_e \parallel [\downarrow \implies \top \star \top]^w, [\downarrow \implies \top]^w_\epsilon \quad \downarrow \text{R} \\
&; w_e \triangleq \epsilon, w \triangleq w \circ w_e \parallel [\downarrow \implies \top \star \top]^w, [\downarrow \implies \top]^w_\epsilon \quad \ast \text{R} \\
&; w_e \triangleq \epsilon \parallel [\downarrow \implies \top \star \top]^w, [\downarrow \implies \top]^w_\epsilon \quad \ast \text{EJoin} \\
&; \quad [\downarrow \implies \top \star \top]^w_\epsilon \quad \text{ENew}
\end{align*}
\]

Note that there is no need to create fresh child heaps \( w_1 \) and \( w_2 \) with \( w \triangleq w_1 \circ w_2 \); if we can prove the world sequent using fresh child heaps about which nothing is known, we should be able to prove it.
equally by reusing an existing empty heap. Similarly we prove the validity of \( \neg (\top \rightarrow \bot) \) as follows:

\[
\vdots \vdash w_c = \epsilon, w = w \circ w_c \parallel [T \rightarrow \bot, \bot \Rightarrow \top]^{w_c}, \vdash \top \Rightarrow \top^{w_c} \quad \Downarrow \text{L}
\]

\[
\vdots \vdash w_c = \epsilon, w = w \circ w_c \parallel [T \rightarrow \bot \Rightarrow \top]^{w_c}, \vdash \bot \Rightarrow \bot^{w_c} \quad \Downarrow \text{EJoin}
\]

\[
\vdots \vdash [\cdot \Rightarrow \neg (\top \rightarrow \bot)]^{w} \quad \Downarrow \text{R}
\]

\[\text{Again we do not create fresh sibling and parent heaps and instead reuse an existing empty heap.}\]

### 3.4 Heap contradiction rules of \( \text{P}_{\text{SL}} \)

The proof system \( \text{P}_{\text{SL}} \) has five rules, \( \text{Cont} \neq \) to \( \text{Cont} \circ \rightarrow \), for producing heap contradictions. In conjunction with the structural rules, these rules enable us to detect all types of heap contradictions, thereby accomplishing the second requirement for obtaining a complete proof strategy.

To see why, assume a world sequent \( \Theta; \Sigma \parallel \Pi \). By repeatedly applying the structural rules in the same order as presented above, we can obtain a semantically equivalent set of world sequents \( \Theta; \Sigma_i \parallel \Pi_i \), \( (i = 1, \ldots, n) \) such that: 1) \( \Sigma_i \) induces a graph of heaps in which all terminal heaps are disjoint; 2) atomic heap relations reside only for terminal heaps and we need to consider only terminal heaps to detect heap contradictions. For an empty heap, we use the rules \( \text{Cont} \neq \) and \( \text{Cont} \circ \rightarrow \) which express the only ways to produce heap contradictions from an empty heap with \( w = \epsilon \). Note that \( w = \epsilon \) and \( w \neq [l \rightarrow E] \) do not produce a heap contradiction because the former implies the latter. For a terminal singleton heap, we use the rules \( \text{Cont} \circ \rightarrow \) and \( \text{Cont} \neq \) which express the only ways to produce heap contradictions from a terminal singleton heap with \( w = [l \rightarrow E] \). Note that \( w \neq [l \rightarrow E] \) implies \( w \neq \epsilon \) always and \( w \neq [l' \rightarrow E'] \) if \( \Theta \vdash [l \rightarrow E] \neq [l' \rightarrow E'] \) holds. We do not need to consider other forms of terminal heaps, for example, those with no atomic heap relations. Finally the rule \( \text{Cont} \circ \rightarrow \) expresses the only way to produce heap contradictions from two disjoint terminal singleton heaps with \( w_1 = [l_1 \rightarrow E_1] \) and \( w_2 = [l_2 \rightarrow E_2] \). This is because if \( \Theta \vdash l_1 \neq l_2 \) holds, the two heap relations are consistent with \( w_1 = w_1 \circ w_2 \) and we cannot produce a heap contradiction. In this way, we can detect all types of heap contradictions in heap relations.

### 4 Examples of proving world sequents

This section presents three examples of proving world sequents in \( \text{P}_{\text{SL}} \). We write \( [l \mapsto \cdot] \) to denote \( [l \mapsto E] \) for some expression \( E \) and assume two distinct location expressions \( l \) and \( l' \) \( (l \neq l') \).

#### 4.1 \( \neg (([l \mapsto \cdot] \circ [l' \mapsto \cdot]) \land ([l \mapsto \cdot] \circ \neg [l' \mapsto \cdot])) \)

The goal formula implies that given a fragment of a heap, we can uniquely determine the remaining fragment. Its proof illustrates that the rule \( \text{Disj}^* \) indirectly applies cancellativity of \( \circ \) to two pairs of child heaps.
We begin with a world sequent $\vdash \parallel [\cdot : \Rightarrow C]^w$ where $C$ is the goal formula. After applying the logical rules, we obtain the following graph of heaps where heap relations are displayed for child heaps:

Then we apply the rule $\text{Disj} \ast$ and the propagation rules $\text{Prop} \rightarrow$ and $\text{Prop} \rightarrow \neq$ to generate $2 \times 2 \times 2 \times 4 = 32$ different world sequents as new goals. All these new goals are immediately provable by the rules $\text{Cont} \epsilon \rightarrow$, $\text{Cont} \epsilon \neq$, and $\text{Cont} \rightarrow \neq$. An example of such a world sequent has heap relations $w_4 \parallel [l' \mapsto \cdot]$, originating from heap $u_2$ by the rule $\text{Prop} \rightarrow$, and $w_4 \neq [l' \mapsto \cdot]$, originating from heap $v_2$ by the rule $\text{Prop} \rightarrow \neq$:

By applying the rule $\text{Cont} \rightarrow \neq$ to heap $w_4$, we complete the proof.

4.2 $A \ast A \supset A$ where $A = -((\top \rightarrow -l)$

The goal formula is valid in separation logic because heaps form a partial deterministic monoid: $H_1 \circ H_2$ may be undefined (when $H_1 \# H_2$ does not hold), but if it is defined, the result is unique. In contrast, the same formula is not valid in Boolean BI, the underlying theory of separation logic, which assumes a non-deterministic monoid [21].

The proof illustrates the use of the rule $\text{EJoin}$ in proving a non-trivial formula. After applying the logical rules to a world sequent $\vdash \parallel [\cdot : \Rightarrow A \ast A \supset A]^w$, we obtain the following graph of heaps:

Since heap $w$ has no sibling and parent heaps, we cannot apply the rule $\rightarrow L$ to $\top \rightarrow -l$ at this point. To make further progress, we apply either the rule $\text{Disj} \rightarrow$ or the rule $\text{EJoin}$ after creating an empty heap. If we apply the rule $\text{EJoin}$, we obtain the following graph:

An application of the rule $\rightarrow L$ to $\top \rightarrow -l$ at heap $w$ generates two new goals, and the interesting case produces $-l$ as a true formula at the same heap (where we omit $\top \rightarrow -l$):
By applying the logical rules to heap \( w \) and the propagation rule \( \text{Prop} \), we obtain the following graph:

Now we can either apply the propagation rule \( \text{Prop} \) to heap \( w \) or use the rule \( \text{NormPC} \) to complete the proof.

4.3 \( \neg((\lfloor l' \mapsto \cdot \rfloor \rightarrow \bot \land (\lfloor l' \mapsto \cdot \rfloor \star \neg((\lfloor l \mapsto \cdot \rfloor \rightarrow \star ((\lfloor l' \mapsto \cdot \rfloor \rightarrow \bot)))))) \)

The goal formula is an example of a formula that cannot be proven without an application of the rule \( \text{Disj} \rightarrow \). After applying the logical rules to a world sequent \( \cdot \parallel \lfloor \cdot \Rightarrow C \rfloor^w \) where \( C \) is the goal formula, we obtain the following graph:

We wish to identify heaps \( w_1 \) and \( w_2 \) (because heaps \( u_1 \) and \( u_3 \) are identical), but there is no way to make further progress toward identifying \( w_1 \) and \( w_2 \) without applying the rule \( \text{Disj} \rightarrow \). Hence we apply the rule \( \text{Disj} \rightarrow \) and the propagation rule \( \text{Prop} \rightarrow \) to heaps \( u_1 \) and \( u_3 \). After eliminating three new goals that are immediately provable, we obtain the following graph:

By applying the rule \( \text{NormPC} \) to merge heaps \( w_1 \) and \( w_2 \), we obtain a heap sequent \( \lfloor l' \mapsto \cdot \rfloor \rightarrow \bot \Rightarrow \lfloor l' \mapsto \cdot \rfloor \rightarrow \bot \), which is easily provable.

5 Admissibility of cut

We state the admissibility of cut in \( P_{\text{SL}} \) as follows:

**Theorem 5.1** (Admissibility of cut).

If \( \Theta; \Sigma \parallel \Pi, [\Gamma \Rightarrow \Delta, C]^u \) and \( \Theta; \Sigma \parallel \Pi, [\Gamma, C \Rightarrow \Delta]^u \), then \( \Theta; \Sigma \parallel \Pi, [\Gamma \Rightarrow \Delta]^u \).

Theorem 5.1 assumes a few properties, such as weakening and contraction, of the expression contradiction judgment \( \Theta \vdash \bot \) (for which we do not provide inference rules). In particular, we assume its own admissibility of cut (where \( \neg \Theta \) denotes the negation of \( \Theta \)): \( \Theta_1, \Theta \vdash \bot \) and \( \Theta_2, \neg \Theta \vdash \bot \). We imply \( \Theta_1, \Theta_2 \vdash \bot \).

The first step in the proof of Theorem 5.1 is to show that it is safe to merge two arbitrary heap sequents:

**Lemma 5.2.** If \( \Theta; \Sigma \parallel \Pi, [\Gamma_1 \Rightarrow \Delta_1]^u, [\Gamma_2 \Rightarrow \Delta_2]^u \), then \( \Theta; [u/v] \Sigma \parallel \Pi, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^u \).
Intuitively the second world sequent inherits every heap relation from the first world sequent, so we should be able to prove the second by the same sequence of rules in the proof of the first or its subsequence.

Next we prove the contraction property for heap relations:

**Proposition 5.3.** If Θ; Σ, σ || Π, then Θ; Σ, σ || Π.

The statement in Proposition 5.3 implicitly assumes that we may apply the rules Disj•, Disj→, and Assoc to the same heap relation. For the rules Disj• and Disj→, the proof for such an application, which essentially has no effect, requires the rules ENew and EJoin, which are necessary for the completeness of PSL anyway. For the rule Assoc, however, the proof for such an application requires the rule ECANCEL, which is unnecessary for the completeness of PSL. Hence, if we never apply the rule Assoc to the same heap relation, we may discard the rule ECANCEL.

To prove Theorem 5.1, we generalize its statement as follows:

**Lemma 5.4.** If Θ1; Σ1 || Π1, [Γ1 \( \Rightarrow \) \( \Delta_1, C \)]w and Θ2; Σ2 || Π2, [Γ2 \( \Rightarrow \) \( \Delta_2 \)]w, then

\[ \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \parallel \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]_w. \]

Here \( \Pi_1 \parallel \Pi_2 \) denotes the result of combining heap sequents for the same heap variable. In conjunction with the contraction property for formulas, Lemma 5.4 implies Theorem 5.1.

### 6 Soundness and completeness of PSL

This section proves the soundness and completeness of the proof system PSL with respect to separation logic. The proof of soundness is straightforward, whereas the proof of completeness uses several subtle properties of graphs of heaps represented by world sequents. In this section, metavariable \( W \) denotes world sequents and heap variables directly refer to heaps. For simplicity, we do not consider first-order formulas.

#### 6.1 Soundness

The soundness property states that a derivation of a world sequent means that its semantic interpretation is self-contradictory. As a special case, we obtain Theorem 6.1:

**Theorem 6.1 (Soundness).** If \( ; \cdot || [\cdot \Rightarrow A]_w \), then \( \models A \).

The key step in the proof of soundness is to show that in any inference rule of PSL, the world sequent in the conclusion is either self-contradictory in itself or semantically implies the disjunction of all world sequents in the premise. Given a stack \( S \), let us write \([W]_S\) for the interpretation of world sequent \( W \) according to the semantics of separation logic (which is formally defined below). We wish to prove that a derivation of \( W \) implies \( \neg [W]_S \), i.e., \([W]_S\) is self-contradictory, for any stack \( S \). Suppose that the last inference rule in the derivation of \( W \) is not an axiom and has world sequents \( W_1, \ldots, W_n \) in its premise \( n \geq 1 \). By induction hypothesis, we have \( \neg [W_1]_S, \ldots, \neg [W_n]_S \), or equivalently, \( \wedge_{i=1, \ldots, n} \neg [W_i]_S \). Then, by proving that \([W]_S\) implies \( \vee_{i=1, \ldots, n} [W_i]_S \), we prove that \( \wedge_{i=1, \ldots, n} \neg [W_i]_S \) implies \( \neg [W]_S \). Now \( \neg [W]_S \) immediately follows.

Formally we define \([W]_S\) using three auxiliary semantic functions \([\theta]_S\), \([\sigma]_S\), and \([\pi]_S\), all of which follow our intuition on world sequents given in Section 3.1:

\[
\begin{align*}
[E = E']_S &= [E]_S = [E']_S \\
[E \neq E']_S &= [E]_S \neq [E']_S \\
[w \triangleright e]_S &= w = e \\
[w \neq e]_S &= w \neq e \\
[w \triangleright l \mapsto E]_S &= w = \langle [l]_S \mapsto [E]_S \rangle \\
[w \neq l \mapsto E]_S &= w \neq \langle [l]_S \mapsto [E]_S \rangle \\
[w \triangleright w_1 \circ w_2]_S &= w = w_1 \circ w_2 \\
[[\Gamma \Rightarrow \Delta]_w]_S &= \wedge_{A \in \Gamma} (S, w) \models A \wedge \wedge_{B \in \Delta} (S, w) \not\models B \\
[\Theta; \Sigma || \Pi]_S &= \wedge_{\theta \in \Theta} [\theta]_S \wedge \wedge_{\sigma \in \Sigma} [\sigma]_S \wedge \wedge_{\pi \in \Pi} [\pi]_S 
\end{align*}
\]

Now we prove the key step in the proof of soundness:
Lemma 6.2. For every inference rule with the conclusion \( W \) and the premise consisting of \( W_1, \ldots, W_n \), it holds that \([W]_S\) implies \( \bigwedge_{i=1,\ldots,n} [W_i]_S\) for any stack \( S \). If \( n = 0 \), we have \( \neg [W]_S \).

As a corollary, we prove that a derivation of a world sequent means that its semantic interpretation is self-contradictory.

Corollary 6.3. If there is a derivation of a world sequent \( W \) in \( P_{SL} \), then \( \neg [W]_S \) holds for any stack \( S \). For the rule ExpCont, we assume that \( \Theta \vdash \bot \) implies \( \neg [\Theta \vdash \bot]_S \).

Then a derivation of \( \vdash \cdot \| : \Rightarrow A \) implies \( (S, w) \models A \):

\[
\neg [\vdash \cdot \| : \Rightarrow A]_w = \neg (S, w) \not\models A = (S, w) \models A
\]

Since \( w \) denotes an arbitrary heap, we have \( \models A \) and Theorem 6.1 follows.

### 6.2 Completeness

The completeness property states that a valid formula in separation logic has a proof of its negation in \( P_{SL} \):

**Theorem 6.4 (Completeness).** If \( \models A \), then \( \vdash \cdot \| : \Rightarrow A \).

For the proof of completeness, we weaken the rules \( *, R \) and \( \rightarrow, L \) by discarding their principal formula in the premise. The original rules are invertible and useless applications with wrong heap relations are safe, but only because the principal formula survives in the premise and we can always try different heap relations without having to backtrack. This is, however, inadequate for the proof of completeness, which must show how to actually find a right heap relation. Hence we weaken the rules \( *, R \) and \( \rightarrow, L \) to directly reflect the semantics of \( * \) and \( \rightarrow \), but also present a scheme for computing the complete set of heap relations for a given heap. We also introduce an explicit weakening rule for eliminating an atomic heap relation (which is admissible) as a new structural rule:

\[
\begin{array}{c}
\sigma \text{ is an atomic heap relation} \\
\Theta; \Sigma, \sigma \parallel \Pi
\end{array}
\begin{array}{c}
\Theta; \Sigma \parallel \Pi
\end{array}
\]

**Weaken**

We use the rule Weaken to eliminate all atomic heap relations at non-terminal heaps when the propagation rules can produce no more new heap relations. As explained in Section 5, we do not use the rule ECancel.

Our proof of Theorem 6.4 uses three new concepts: **canonical world sequents**, disjunctive derivation states, and conjunctive proof goals.

A canonical world sequent \( Z \) is a special form of a world sequent such that if \( \neg [Z]_S \) holds for any stack \( S \), we can construct its derivation using only the rules \( \bot, L, \text{ExpCont} \), and the heap contradiction rules. (In Proposition 6.12, we introduce a class of world sequents that are shown to be canonical.)

A disjunctive derivation state \( \Psi \) for a world sequent \( W \) is a set of world sequents that constitute all the leaves in a partial derivation of \( W \). That is, a disjunctive derivation state \( \Psi = \{W_1, \ldots, W_n\} \) for a world sequent \( W \) means that there is a partial derivation of the following form:

\[
\begin{array}{c}
W_1 \quad \cdots \quad W_n \\
\vdots \\
\overline{W}
\end{array}
\]

We use a reduction judgment \( \Psi \xrightarrow{\mathcal{R}} \Psi' \) to mean that such a partial derivation expands to another partial derivation with disjunctive derivation state \( \Psi' \) by an application of the logical or structural rule \( \mathcal{R} \) to some world sequent \( W_i \) (\( 1 \leq i \leq n \)). That is, we have \( \Psi' = \Psi = \{W_i\} \cup \{W_i^1, \ldots, W_i^m\} \) with:

\[
\begin{array}{c}
W_1 \quad \cdots \quad W_n \\
\vdots \\
\overline{W}
\end{array}
\]

We write \( \Psi \xrightarrow{\star} \Psi' \) for the reflexive and transitive closure of \( \xrightarrow{\cdot} \). For a stack \( S \), we define the interpretation \( [\Psi]_S = \bigvee_{W_i \in \Psi} [W_i]_S \).
A conjunctive proof goal $\Omega$ is a set of disjunctive derivation states for a common world sequent. Given a logical or structural rule $R_i$, we use a reduction judgment $\Omega \xrightarrow{R_i} \Omega'$ to mean that we can generate $\Omega'$ by applying the rule $R_i$ to some disjunctive derivation state $\Psi$ in $\Omega$. That is, we have $\Omega' = \Omega - \{\Psi\} \cup \{\Psi_1, \ldots, \Psi_n\}$ and $\Psi_i \xrightarrow{R_i} \Psi'_i$ for $i = 1, \ldots, n$. If $R_i$ is the rule $\ast R$ or $\rightarrow L$, we have $n \geq 1$ and produce each $\Psi'_i$ by focusing on the same formula in the same heap sequent in the same world sequent in $\Psi$. For all the other rules, we have $n = 1$ and replace $\Psi$ by $\Psi'_1$. We write $\Omega \xrightarrow{\ast} \Omega'$ for the reflexive and transitive closure of $\xrightarrow{\ast}$. For a stack $S$, we define the interpretation $[\Omega]_S = \bigwedge_{W \in \Omega} [\Psi]_S$.

In order to prove Theorem 6.4, assume a goal formula $A$ such that $\models A$. We aim to build a sequence of conjunctive proof goals $\Omega_1, \ldots, \Omega_N$ such that:

- $\Omega_1 = \{\{W\} \mid W = \vdots \, \pipe \vdash A\}^\omega$.
- $\Omega_i \xrightarrow{R_i} \Omega_{i+1}$ for $i = 1, \ldots, N - 1$ where $R_i$ is a logical or structural rule.
- $[\Omega_{i+1}]_S$ implies $[\Omega_i]_S$ for $i = 1, \ldots, N - 1$ and any stack $S$.
- $\Omega_N$ contains only canonical world sequents.

With such a sequence of conjunctive proof goals, we can build a derivation of $W$ as follows:

- Since we have $\models A$, we have $(S, w) \models A = \neg (S, w) \not\models A = \neg \neg [W]_S = \neg [\Omega_1]_S$ for any stack $S$.
- Since $[\Omega_N]_S$ implies $[\Omega_1]_S$, we have $\neg [\Omega_N]_S$.
- Since every disjunctive derivation state in $\Omega_N$ contains only canonical world sequents, we may write $\Omega_N = \bigcup_j \Psi_j$ and $\Psi_j = \bigcup_k Z_k^j$.
- Since we have $\neg [\Omega_N]_S = \neg \bigwedge_j [\Psi_j]_S = \bigvee_j \neg [\Psi_j]_S$, there exists a disjunctive derivation state $\Psi_j$ such that $\neg [\Psi_j]_S$ holds.
- Since we have $\neg [\Psi_j]_S = \neg \bigvee_k [Z_k^j]_S = \bigwedge_k \neg [Z_k^j]_S$, we have $\neg [Z_k^j]_S$ for each $k$.
- Since $\Omega_1 \xrightarrow{\ast} \Omega_N$, we have $\{W\} \xrightarrow{\ast} \Psi_j$.
- Since we have $\neg [Z_k^j]_S$ for each $k$, there is a derivation of $Z_k^j$ for each $k$ by the definition of canonical world sequents.
- By combining $\{W\} \xrightarrow{\ast} \Psi_j$ and the derivation of $Z_k^j$ for each $k$, we obtain a derivation of $W$.

Below we explain how to build such a sequence of conjunctive proof goals.

### 6.2.1 Completeness of the invertible rules

Suppose that a world sequent in $\Omega^\prime$ contains a formula to which we can apply a rule $R$ other than $\bot L$, $\ast R$, and $\rightarrow L$. By applying the rule $R$, we obtain $\Omega^\prime \xrightarrow{R} \Omega^{\prime+1}$. By Corollary 6.6, $[\Omega^{\prime+1}]_S$ implies $[\Omega^\prime]_S$ for any stack $S$. In this way, we can eliminate all such formulas without losing completeness.

**Proposition 6.5.** Except for the rules $\bot L$, $\text{ExpCont}$, $\ast R$, $\rightarrow L$, and $\text{Weaken}$, every logical or structural rule with the conclusion $W$ and the premise consisting of $W_1, \ldots, W_n$ is invertible in that $\bigvee_{i=1, \ldots, n} [W_i]_S$ implies $[W]_S$ for any stack $S$.

**Corollary 6.6 (Completeness of the invertible rules).**

For such an invertible rule $R$ and any stack $S$, if $\{W\} \xrightarrow{R} \Psi$, then $[\Psi]_S$ implies $[\{W\}]_S$.

### 6.2.2 Completeness of the rules $\ast R$, $\rightarrow L$, and $\text{Weaken}$

We now show how to recover the completeness of the rules $\ast R$ and $\rightarrow L$. We introduce several notations in order to concisely describe properties of graphs of heaps:

- $w \not\rightarrow u$ means that there is a sequence of zero or more child-parent relations from heap $w$ to heap $u$ in the graph: $w = w_0, w_1 = w_0 \circ w_1, \ldots, w_n = w_{n-1} \circ w_n$, and $w_n = u$ for $n \geq 0$. Hence, if $w \not\rightarrow u$, heap $w$ is a descendant of heap $u$, or equivalently, heap $u$ is an ancestor of heap $w$. Note that we allow $w \not\rightarrow w$. 

16
• \( w \uparrow \) means that \( w \) is a terminal heap, i.e., there is no heap relation \( w \vdash u \circ v \).
• \( T(w) \) denotes the set of terminal descendants of heap \( w \), i.e., \( T(w) = \{ v \mid v \uparrow \text{and } v \nrightarrow w \} \).

We assume that heap relations in every world sequent induce not only a graph of heaps but also a special empty heap \( w_\bot \) with heap relation \( w_\bot = \epsilon \) that is separate from the graph. This assumption is safe because we can always generate such a special empty heap with the rule \( \text{ENew} \) if there is none, and combine multiple empty heaps with the rule \( \text{NormEmpty} \) if there are many. We classify world sequents according to the property of graphs of heaps induced by their heap relations (without considering its special empty heap \( w_\bot \)) as follows:

1. Elementary: if \( w = w_1 \circ w_2 \), then \( T(w_1) \cap T(w_2) = \emptyset \).
2. Rooted: elementary and there is a root heap \( w \) such that \( v \nrightarrow w \) for every heap \( v \).
3. Consistent: rooted and if \( w = u_1 \circ u_2 \) and \( w = v_1 \circ v_2 \), then \( T(u_1) \cup T(u_2) = T(v_1) \cup T(v_2) \).
4. Full: consistent and for any non-empty set \( S \) of terminal heaps, there exists at least one heap \( w \) with \( T(w) = S \).
5. \( \ast \)-ready for heap \( w \): full and for any pair of non-empty sets \( S_1 \) and \( S_2 \) of terminal heaps such that \( S_1 \cap S_2 = \emptyset \) and \( S_1 \cup S_2 = T(w) \), there exist heaps \( w_1 \) and \( w_2 \) such that \( w = w_1 \circ w_2 \) with \( T(w_1) = S_1 \) and \( T(w_2) = S_2 \).
6. \( \rightarrow \ast \)-ready for heap \( w \): full and for any pair of non-empty sets \( S_1 \) and \( S_2 \) of terminal heaps such that \( T(w) \cap S_1 = \emptyset \) and \( T(w) \cup S_1 = S_2 \), there exist heaps \( w_1 \) and \( w_2 \) such that \( w_2 = w_1 \circ w_1 \) with \( T(w_1) = S_1 \) and \( T(w_2) = S_2 \).
7. Saturated: full and applications of the propagation rules produce no more new heap relations.
8. Sanitized: full and non-terminal heaps have no atomic heap relations.
9. Normalized: sanitized with no empty heaps and for any non-empty set \( S \) of terminal heaps, there exists a unique heap \( w \) with \( T(w) = S \).
10. Expanded: obtained by applying only the logical rules except \( \ast \text{R} \) and \( \rightarrow \text{L} \) to some consistent world sequent.

Suppose that a world sequent in \( \Omega^i \) contains a false formula \( A \ast B \) or a true formula \( A \rightarrow B \) about heap \( w \). By converting it into normalized world sequents that are also \( \ast \text{R} \)-ready or \( \rightarrow \text{L} \)-ready for heap \( w \) according to Corollary 6.10, we obtain \( \Omega^i \vdash^* \Omega^{i+1} \) such that \( [\Omega^{i+1}]_S \) implies \( [\Omega^i]_S \) for any stack \( S \). By Proposition 6.11, we can apply the rule \( \ast \text{R} \) or \( \rightarrow \text{L} \) to obtain \( \Omega^{i+2} \) with \( \Omega^{i+1} \vdash^* \Omega^{i+2} \) or \( \Omega^{i+1} \vdash^* \Omega^{i+2} \) such that \( [\Omega^{i+2}]_S \) implies \( [\Omega^{i+1}]_S \) for any stack \( S \). In this way, we can eliminate every false formula \( A \ast B \) or true formula \( A \rightarrow B \) without losing completeness.

**Lemma 6.7.** For a world sequent \( W \) of a particular kind, there exists a world sequent \( W' \) of another kind such that:

1. \( \{ W \} \vdash^* \{ W' \} \) by applying only the structural rules;
2. \( [\{ W' \}]_S \) implies \( [\{ W \}]_S \) for any stack \( S \),

where one of the following holds:

1. \( W \) is expanded and \( W' \) is rooted;
2. \( W \) is rooted (generated in step 1) and \( W' \) is consistent;
3. \( W \) is consistent and \( W' \) is full;
4. \( W \) is full and \( W' \) is \( \ast \)-ready for a given heap \( w \);
5. \( W \) is full and \( W' \) is \( \rightarrow \ast \)-ready for a given heap \( w \);
6. \( W \) is sanitized and \( \ast \)-ready \( (\rightarrow \ast \text{R ready}) \) for a given heap \( w \), and \( W' \) is normalized and \( \ast \)-ready \( (\rightarrow \text{L ready}) \) for heap \( w \).

**Lemma 6.8.** For a world sequent \( W \) of a particular kind, there exists a disjunctive derivation state \( \Psi \) such that:

1. \( \{ W \} \vdash^* \Psi \) by applying only the propagation rules;
2. \( [\Psi]_S \) implies \( [\{ W \}]_S \) for any stack \( S \),

where one of the following holds:
5. W is $\star$-ready for a given heap $w$, and every world sequent in $\Psi$ is saturated as well as $\star$-ready for heap $w$.
6. W is $\neg\neg$-ready for a given heap $w$, and every world sequent in $\Psi$ is saturated as well as $\neg\neg$-ready for heap $w$.

**Lemma 6.9** (Completeness of the rule Weaken). For any saturated world sequent $W$, there exists a sanitized world sequent $W'$ such that:

1) $\{W\} \rightarrow^* \{W'\}$ by applying only the rule Weaken;
2) $[\{W'\}]_S$ implies $[\{W\}]_S$ for any stack $S$.

**Corollary 6.10.** For any expanded world sequent $W$ and heap $w$, there exists a disjunctive derivation state $\Psi$ such that:

1) $\{W\} \rightarrow^* \Psi$ by applying only the structural rules;
2) $\Psi$ contains only normalized world sequents that are also $\star$-ready or $\neg\star$-ready for heap $w$;
3) $[\Psi]_S$ implies $[\{W\}]_S$ for any stack $S$.

**Proposition 6.11** (Completeness of the rules $\star R$ and $\rightarrow L$).

Consider a normalized world sequent $W$ that is also $\star$-ready for heap $w$. Suppose that we obtain $\{W\} \rightarrow_R \Psi$, by applying the rule $\star R$ to a false formula $A \star B$ about heap $w$ for each heap relation $w = w_i \circ w' (i = 1, \ldots, n)$, and $\{W\} \rightarrow R \Psi$, by applying the rule $\star R$ to the same formula for another heap relation $w = w \circ w_i$. Then a conjunctive proof goal $\Omega = \{\Psi_1, \ldots, \Psi_n, \Psi_s\}$ satisfies:

1) $[\{W\}] \rightarrow_R \Omega$;
2) $[\Omega]_S$ implies $[\{W\}]_S$ for any stack $S$;
3) No world sequent in $\Omega$ contains the false formula $A \star B$ about heap $w$;
4) Every world sequent in $\Omega$ is still normalized and thus consistent.

Similarly for the rule $\rightarrow L$ where we use each heap relation $w_i = w \circ w'_i$.

### 6.2.3 Completeness of the heap contradiction rules

Suppose that after eliminating all formulas other than $\perp$, we obtain a conjunctive proof goal $\Omega'$ which contains only expanded world sequents. By Corollary 6.10, we can obtain $\Omega' \rightarrow^* \Omega'^{+1}$ such that every world sequent in $\Omega'^{+1}$ is normalized and $[[\Omega'^{+1}]]_S$ implies $[[\Omega']]_S$ for any stack $S$. Proposition 6.12 shows that $\Omega'^{+1}$ contains only canonical world sequents.

**Proposition 6.12** (Completeness of the heap contradiction rules).

A normalized world sequent $W$ with no formulas other than $\perp$ is canonical. That is, if $\neg [W]_S$ holds for any stack $S$, we can construct its derivation using only the rules $\perp L$, ExpCont, and the heap contradiction rules. For the rule ExpCont, we assume that $\neg [\Theta \vdash \perp]_S$ implies $\Theta \vdash \perp$.

Thus we conclude the proof of Theorem 6.4.

### 6.3 Proof search strategy

Figure 4 shows the proof search strategy based on the proof of Theorem 6.4. Given a world sequent of the form $\vdash \| \vdash A^\omega$, we repeatedly apply the logical rules other than the rules $\star R$ and $\rightarrow L$ to obtain expanded world sequents according to Corollary 6.6 (from initial to expanded). Then we apply a series of structural rules to obtain normalized world sequents according to Corollary 6.10 (from expanded to normalized). If no formulas other than $\perp$ remain, we attempt to generate a logical contradiction according to Proposition 6.12 (from normalized to contradiction). Otherwise we apply the rule $\star R$ or $\rightarrow L$ to obtain consistent world sequents according to Proposition 6.11 (from normalized to consistent). By repeatedly applying the logical rules other than the rules $\star R$ and $\rightarrow L$ again, we obtain another set of expanded world sequents according to Corollary 6.6 (from consistent to expanded). In this way, we can eventually decompose all formulas other than $\perp$ and complete the proof search.

As it directly translates to a proof search strategy, the proof of Theorem 6.4 agrees with the result that propositional separation logic is decidable [9]. Since first-order separation logic is undecidable [5], the proof system $\mathbf{PSL}$ becomes undecidable in the presence of the rule $\exists R$. We also remark that in the presence of propositional variables, even propositional separation logic is undecidable [6].
6.4 Validity of formulas in $P_{SL}$

As established by Theorems 6.1 and 6.4, the notion of validity in $P_{SL}$ means that a formula is true (or an assumption of its falsehood leads to a logical contradiction) at an arbitrary heap about which nothing is known. An important implication of assuming such an arbitrary heap is that we in effect assume an arbitrary world of heaps, since even the relationship with an external heap can be thought of as a property of the arbitrary heap, which is assumed to be unknown. In particular, we may not assume the existence of a singleton heap with location $L$ even if the goal formula contains a points-to relation $[L \rightarrow E]$.

As an example, consider a formula

$$A = I \supset \neg ([L \rightarrow E] \star \neg [L \rightarrow E])$$

which states that an empty heap combined with a heap satisfying $[L \rightarrow E]$ cannot be a heap where $[L \rightarrow E]$ is false. According to the semantics of separation logic in Section 2, this formula is valid but only provided that a singleton heap satisfying $[L \rightarrow E]$ is known to exist. The existence of such a singleton heap, however, is an assumption that can be justified only by inspecting the formula at the meta-logical level. Since $P_{SL}$ assumes an arbitrary world of heaps without knowing the existence of such a singleton heap, it fails to prove the above formula.

If we are to show the validity of a formula by exploiting prior knowledge on the world of heaps, we should prove an extended formula that incorporates its meta-logical property as well. For example, in order to show the validity of the above formula $A$ when the existence of a singleton heap satisfying $[L \rightarrow E]$ is already known, we should instead prove an extended formula

$$[L \rightarrow E] \star \top \supset \neg (A \star \top)$$

which states that if the world of heaps contains a heap satisfying $[L \rightarrow E]$, there cannot exist a heap where $A$ is false, i.e., $A$ is true at any heap. An attempt to prove the extended formula produces a world sequent

$$\vdash [\top \implies \cdot]^w, \quad [L \rightarrow E] \implies \cdot^{u_1}, \quad [\top \implies \cdot]^w, \quad [\cdot \implies A]^{v_1}, \quad [\top \implies \cdot]^w, \quad w \doteq u_1 \circ u_2, \quad \vdash w \doteq v_1 \circ v_2$$

which essentially expresses that formula $A$ is false at an arbitrary heap when there exists a heap satisfying $[L \rightarrow E]$. A proof of this world sequent complies with the validity of formula $A$ in separation logic.
7 Discussion

Our prototype implementation of $\mathbb{P}_{\mathbf{SL}}$ (without first-order formulas) is based on the proof of Theorem 6.4, but with a few changes. In particular, it internally uses a different type of normalized world sequents which maintain a unique heap corresponding to each non-empty set of terminal heaps, but permit unknown relations between heaps. The decision is based on the observation that it is the rules $\text{Disj}^*$ and $\text{Disj} \rightarrow$ (for eliminating unknown relations between heaps) that contributes the most to the complexity of graphs of heaps. Thus it selectively applies the rules $\text{Disj}^*$ and $\text{Disj} \rightarrow$ only when it cannot complete the proof search otherwise.

Our experience with the prototype implementation of $\mathbb{P}_{\mathbf{SL}}$ shows that it allows us to incorporate new logical connectives and predicates in a principled way without having to introduce additional structural rules. As an example, consider an overlapping conjunction $A \& B$ by Hobor and Villard [17] which can be defined in the framework of $\mathbb{P}_{\mathbf{SL}}$ as follows:

- $A \& B$ is true at heap $w$ iff. $w \models w_1 \circ v_2$, $w \models v_1 \circ w_2$, $w_1 \models v_1 \circ u$, $w_2 \models u \circ v_2$, and $A$ is true at heap $w_1$ and $B$ is true at heap $w_2$ for some heaps $w_1$, $w_2$, $v_1$, $v_2$, and $u$.

- $A \& B$ is false at heap $w$ iff. $w \models w_1 \circ v_2$, $w \models v_1 \circ w_2$, $w_1 \models v_1 \circ u$, and $w_2 \models u \circ v_2$ implies that $A$ is false at heap $w_1$ or that $B$ is false at heap $w_2$ for any heaps $w_1$, $w_2$, $v_1$, $v_2$, and $u$.

We directly translate this definition into two inference rules for $\&$:

\[
\frac{\text{fresh } w_1, w_2, v_1, v_2, u}{\Theta; \Sigma, w \models u \circ v_2, w_1 \models v_1 \circ u, w_1 \models v_1 \circ u, \{ w_2 \models u \circ v_2 \} \subset \Sigma}{\Theta; \Sigma \parallel \Pi, \Gamma, A \& B \models \Delta} \quad (\&L)
\]

\[
\frac{w \models w_1 \circ v_2, w \models v_1 \circ w_2, w_1 \models v_1 \circ u, \{ w_2 \models u \circ v_2 \} \subset \Sigma}{\Theta; \Sigma \parallel \Pi, \Gamma \models \Delta, A \& B} \quad (\&R)
\]

Note that we obtain the rules $\&L$ and $\&R$ exactly in the same way that we derive the rules $\ast L$ and $\ast R$ from the interpretation of multiplicative conjunction $\ast$. The only difference is that we create five fresh heaps in the rule $\&L$ and try to detect a subgraph consisting of six existing heaps in the rule $\&R$. Equally important is that we need no additional structural or heap contradiction rules because overlapping conjunction does not require new forms of heap relations. Thus, in principle, it is relatively easy to incorporate overlapping conjunction into our prototype implementation of $\mathbb{P}_{\mathbf{SL}}$. Overall we may think of $\mathbb{P}_{\mathbf{SL}}$ as a highly extensible proof system for separation logic.

8 Related work

8.1 Automated verification tools based on separation logic

Separation logic has been the basis for a number of automated verification tools targeting programs using mutable data structures. The first such tool is Smallfoot by Berdine et al. [3] which aims to test the feasibility of automated verification using separation logic. To achieve full automation, it permits no pointer arithmetic and verifies only shape properties of linked lists and trees. Space Invader by Distefano et al. [11] permits pointer arithmetic by integrating the abstract interpretation method into the symbolic execution method in [4]. THOR by Magill et al. [23] is an extension of Space Invader which is capable of tracking the length of linked lists. SLAyer by Berdine et al. [1] is another extension of Space
Invader which uses higher-order predicates to express common properties of nodes in linked lists. The use of higher-order predicates enables SLayer to verify shape properties of composite linked lists such as linked lists of circular linked lists.

There are also several tools supporting arbitrary data structures. HIP by Nguyen and Chin [25] allows users to specify invariants on arbitrary data structures in terms of inductive predicates. Since checking these invariants usually relies on basic properties of inductive predicates that are easy to prove but difficult to discover automatically, HIP requires users to explicitly state such properties in the form of lemmas, which are automatically proven and then applied as necessary. Similarly to HIP, VeriFast by Jacobs et al. [19] relies on user-supplied inductive predicates and lemmas. Unlike HIP, however, VeriFast requires users to provide proofs of these lemmas and specify when to apply them. jStar by Distefano and Parkinson [12] is an extension of Space Invader which exploits user-supplied abstraction rules in order to support arbitrary data structures. Its distinguishing feature is the ability to infer loop invariants automatically. Xisa by Chang and Rival [10] takes a different approach by indirectly specifying invariants on data structures with validation code. Xisa analyzes validation code to extract inductive predicates for describing invariants as well as lemmas for describing their basic properties. Since validation code can be written in common programming languages, users of Xisa do not need the expertise to specify invariants of interest in terms of inductive predicates.

All these tools use as their logical foundation not full separation logic but only its decidable fragment by Berdine et al. [2], which does not include separating implication \( \rightarrow^* \). As shown by Ishtiaq and O'Hearn [18], lack of separating implication implies no support for backward reasoning by weakest precondition generation for those programs performing heap assignments or allocation. As a result, these tools allow only forward reasoning based on symbolic execution as in [4] and do not demonstrate the full potential of separation logic in program verification.

### 8.2 Proof search in full separation logic

Despite the practical importance of separating implication, proof search in full separation logic has not drawn much attention from researchers. Calcagno et al. [8] present a translation from propositional separation logic to first-order logic (with only propositional connectives and no multiplicative connectives) for which a decision procedure already exists. The labelled tableau calculus for separation logic by Galmiche and Méry [15] supports both separating conjunction and separating implication. Similarly to our proof system \( \mathcal{P}_{SL} \), their calculus combines both syntactic (tableau) and semantic (labelled) formulations and uses labels to directly refer to heaps. Although it is shown to be sound and complete, their calculus does not give rise to a proof search strategy. Specifically, in order to check that all branches in a tableau are logically or structurally inconsistent, we need two semantic functions, a measure and an interpretation, for each branch. Their calculus, however, does not explain how to construct such semantic functions for each branch and it is not clear how to extract a concrete proof search strategy.

The closest proof system to ours is the nested sequent calculus \( S_{BBI} \) for Boolean BI by Park et al. [26], which inspired the overall design of \( \mathcal{P}_{SL} \). Similarly to world sequents in \( \mathcal{P}_{SL} \), sequents in \( S_{BBI} \) use a truth context consisting of true formulas and a falsehood context consisting of false formulas, and both systems are based on the principle of proof by contradiction. Because of the similarity in syntactic formulations, their approach to dealing with separating conjunction and separating implication in \( S_{BBI} \) equally applies to our setting for \( \mathcal{P}_{SL} \), which is not surprising considering that separation logic is just an instance of Boolean BI with additional restrictions on the semantic structure. The structural rules of \( \mathcal{P}_{SL} \), however, are specific to separation logic and are designed independently of \( S_{BBI} \). Since \( S_{BBI} \) allows propositional variables, we may use its theorem prover as a supplementary system for our implementation of \( \mathcal{P}_{SL} \).

For theorem provers based on the decidable fragment of separation logic by Berdine et al. [2] (without separating implication), see, for example, SeLoger [16] and SLP [24]. For an isomorphism between (intuitionistic) separation logic and implicit dynamic frames, see [27].

### 9 Conclusion

We have presented a proof system \( \mathcal{P}_{SL} \) for full separation logic with separating implication. Considering the potential benefit of separating implication, we envision that program verification systems in the...
future will provide separating implication and support backward reasoning by weakest precondition generation for their scalability in program verification. We also envision that proof assistants can interface with theorem provers for separation logic and provide a powerful automation tactic for dealing with logical connectives from separation logic. When extended with inductively defined predicates, \( \mathbf{P}_{\mathbf{SL}} \) may serve as a practical foundation for such systems.

References


A Admissibility of cut

We assume that the expression contradiction judgment satisfies the followings:

- Weakening: \( \Theta \vdash \bot \) implies \( \Theta, \theta \vdash \bot \)
- Contraction: \( \Theta, \theta, \theta \vdash \bot \) implies \( \Theta, \theta \vdash \bot \)
- Admissibility of cut: \( \Theta_1, \theta \vdash \bot \) and \( \Theta_2, \neg \theta \vdash \bot \) imply \( \Theta_1, \Theta_2 \vdash \bot \)
- \( \Theta \vdash E_1 = E_2 \) and \( \Theta \vdash E_2 = E_3 \) imply \( \Theta \vdash E_1 = E_3 \)
- \( \Theta \vdash E_1 = E_2 \) and \( \Theta \vdash E_2 \neq E_3 \) imply \( \Theta \vdash E_1 \neq E_3 \)

We maintain the invariant that a world sequent contains a unique heap sequent for each heap variable.

In every structural rule, expression relations \( \Theta \) remain the same in the premise. In every structural rule except the rules \( \text{NormEq} \), \( \text{NormPC} \), and \( \text{NormEmpty} \), heap relations \( \Sigma \) either remain the same or expand in the premise.

We define \( |D| \), the size of a proof \( D \), as the number of rule applications along the longest path in \( D \) without counting the number of applications of the rules \( \text{ENew} \), \( \text{EJoin} \) since these two rules do not increase the complexity of a proof. For a proof \( D \), we define the last inference rule as the rule which is lastly applied in \( D \) except the rules \( \text{ENew} \), \( \text{EJoin} \). Note that when we prove, by induction on \( |D| \), the statements in this section of the form “if \( D :: \Theta; \Sigma || \Pi \), then \( E :: \Theta'; \Sigma' || \Pi' \) and \( |E| \leq |D| \),” it suffices for inductive cases to be proven when the last inference rule of \( D \) is not equal to \( \text{ENew} \), \( \text{EJoin} \).

We define the heap sequent union operation \( \Pi \uplus \Pi' \) as follows:

\[
\Pi \uplus \Pi' = \{ [\Gamma, \Gamma' \implies \Delta, \Delta']^w | [\Gamma \implies \Delta]^w \in \Pi \text{ and } [\Gamma' \implies \Delta']^w \in \Pi' \} \cup \\
\{ [\Gamma \implies \Delta]^w | [\Gamma \implies \Delta]^w \in \Pi \text{ or } [\Gamma \implies \Delta]^w \in \Pi' \text{ but not both} \}
\]

**Proposition A.1** (Weakening of expression relations and heap relations).

If \( D :: \Theta; \Sigma || \Pi \), then \( E :: \Theta; \Sigma || \Pi \) and \( |E| \leq |D| \).

**Proof.** By induction on the size of the proof \( D \). \( \square \)

**Proposition A.2** (Weakening).

If \( D :: \Theta; \Sigma || \Pi, \pi \) and \( |E| \leq |D| \).

**Lemma 5.2** (Merging lemma).

If \( D :: \Theta; \Sigma || \Pi, [\Gamma_1 \implies \Delta_1]^w, [\Gamma_2 \implies \Delta_2]^w \), then \( E :: \Theta; [u/v] \Sigma || \Pi, [\Gamma_1, \Gamma_2 \implies \Delta_1, \Delta_2]^w \) and \( |E| \leq |D| \).

**Proof.** By induction on the size of the proof \( D \). \( \square \)

Case \( R_D = \bot \), \( \text{ExpCont} \), \( \text{Cont}^\epsilon \neq \), \( \text{Cont}^\circ \neq \), \( \text{Cont}^\circ \rightleftharpoons \).

This is a base case, and its proof is trivial.

Case \( R_D \) = remaining logical rules except \( \ast \text{R}, \ast \text{L} \):

We present the proof for the case \( R_D = \bot \). The proofs for other rules are similar to it because, in these logical rules, the number of heap sequents which are in the conclusion and modified in the premise is \( \leq 1 \). There are two subcases: \( R_D \) focuses on (a) \( w \neq u, v \) and (b) \( u \) (or \( v \)).

(Subcase) \( R_D \) focuses on \( w \neq u, v \):

In this subcase, we have \( \Pi = \Pi', [\Gamma, \neg A \implies \Delta]^w \) and

\[
D = \frac{D' :: \Theta; \Sigma || IV', [\Gamma \implies \Delta, A]^w, [\Gamma_1 \implies \Delta_1]^w, [\Gamma_2 \implies \Delta_2]^w \quad \text{\( \ast \text{L} \)}}{\Theta; \Sigma || II', [\Gamma, \neg A \implies \Delta]^w, [\Gamma_1 \implies \Delta_1]^w, [\Gamma_2 \implies \Delta_2]^w}
\]

24
\( \mathcal{E}' :: \Theta; [u/v] \Sigma \parallel \Pi', [\Gamma \rightarrow \Delta, A]^w, [\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2]^w \) and \( |\mathcal{E}'| \leq |\mathcal{D}'| \) by IH on \( \mathcal{D}' \)

\( \mathcal{E} :: \Theta; [u/v] \Sigma \parallel \Pi', [\Gamma_1', \neg A \rightarrow \Delta_1]^w, [\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2]^w \) and \( |\mathcal{E}| \leq |\mathcal{D}| \) by the rule \( \neg \Lambda \)

(Subcase) \( R_D \) focuses on \( u \):
In this subcase, we have \( \Gamma_1 = \Gamma_1', \neg A \) and
\[
\mathcal{D} = \begin{array}{c}
\Delta' :: \Theta; \Sigma \parallel \Pi, [\Gamma_1' \rightarrow \Delta_1, A]^u, [\Gamma_2 \rightarrow \Delta_2]^v \\
\mathcal{E}' :: \Theta; [u/v] \Sigma \parallel \Pi, [\Gamma_1', \neg A \rightarrow \Delta_1]^w, [\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2]^w \end{array}
\]
\( \mathcal{E} :: \Theta; [u/v] \Sigma \parallel \Pi, [\Gamma_1', \Gamma_2 \rightarrow \Delta_1, \Delta_2, A]^w \) and \( |\mathcal{E}'| \leq |\mathcal{D}'| \) by IH on \( \mathcal{D}' \)

\( \mathcal{E} :: \Theta; [u/v] \Sigma \parallel \Pi, [\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2]^w \) and \( |\mathcal{E}| \leq |\mathcal{D}| \) by the rule \( \neg \Lambda \)

Case \( R_D = \ast \mathbb{R}, \ast \Lambda \):
We present the proof for the case \( R_D = \ast \mathbb{R} \). The proof for the case \( R_D = \ast \Lambda \) is similar to it. Suppose \( R_D \) focuses on \( \sigma = w_1 \ast w_2 \circ w_3 \). Then, we have the following subcases except symmetric ones:

(a) when \( w_1 \neq w_2, w_3 \) and \( w_2 \neq w_3 \):

\[
i \text{when } w_1 \neq w_2, w_3 \text{ and } w_2 \neq w_3:
\]
\[
i \text{i) } v_1 \neq u, v \text{ (i = 1, 2, 3)}
\]
\[
i \text{ii) } v_1 = u \text{ and } w_1 \neq u, v \text{ (i = 2, 3)}
\]
\[
i \text{iii) } \left\{ \begin{array}{l}
v_1 = u, w_2 = v \text{ and } w_3 \neq u, v \\
v_1 = u \text{ and } w_2 \neq u, v
\end{array} \right.
\]
\[
i \text{iv) } w_2 = u \text{ and } w_1 \neq u, v \text{ (i = 1, 3)}
\]
\[
i \text{v) } w_2 = u \text{ and } w_3 = v \text{ and } w_1 \neq u, v
\]

(b) when \( w_1 = w_2 = w_3 \):

\[
i \text{i) } w_1 \neq u, v \text{ (i = 1, 3)}
\]
\[
i \text{ii) } w_1 = u \text{ and } w_3 \neq u, v
\]
\[
i \text{iii) } w_1 = u \text{ and } w_3 = v
\]
\[
i \text{iv) } w_3 = u \text{ and } w_1 \neq u, v
\]

Let us present the proofs only for the boxed subcases.

(Subcase) \( \sigma = u \ast v \circ w \) and \( w \neq u, v \):
In this subcase, we have \( \Pi = \Pi', \Gamma \Rightarrow \Delta, A \ast B \) and
\[
\mathcal{D} = \begin{array}{c}
\mathcal{D}' :: \Theta; \Sigma \parallel \Pi', [\Gamma \rightarrow \Delta_1, A \ast B]^u, [\Gamma_1 \Rightarrow \Delta_2, A]^v \end{array}
\]
\( \mathcal{E}' :: \Theta; [u/v] \Sigma \parallel \Pi', [\Gamma \Rightarrow \Delta_1, A \ast B]^w, [\Gamma_1 \Rightarrow \Delta_2]^w \) by IH on \( \mathcal{D}' \)

(Subcase) \( \sigma = w \ast u \circ v \) and \( w \neq u, v \):
In this subcase, we have \( \Pi = \Pi', \Gamma \Rightarrow \Delta, A \ast B \) and
\[
\mathcal{D} = \begin{array}{c}
\mathcal{D}' :: \Theta; \Sigma \parallel \Pi', [\Gamma \Rightarrow \Delta, A \ast B]^w, [\Gamma_1 \Rightarrow \Delta_1, A]^u, [\Gamma_2 \Rightarrow \Delta_2]^v \\
\mathcal{E}' :: \Theta; [u/v] \Sigma \parallel \Pi', [\Gamma \Rightarrow \Delta, A \ast B]^w, [\Gamma_1 \Rightarrow \Delta_2]^w \end{array}
\]
\( \mathcal{E} :: \Theta; [u/v] \Sigma \parallel \Pi', [\Gamma \Rightarrow \Delta, A \ast B]^w, [\Gamma_1 \Rightarrow \Delta_1, A]^u \) and \( |\mathcal{E}'| \leq |\mathcal{D}'| \) by IH on \( \mathcal{D}' \)

(Subcase) \( \sigma = u \ast v \circ w \):
In this subcase, we have \( \Delta_1 = \Delta_1', A \ast B \) and
\[
\mathcal{D} = \begin{array}{c}
\mathcal{D}' :: \Theta; \Sigma \parallel \Pi', [\Gamma \Rightarrow \Delta_1', A \ast B]^u, [\Gamma_1 \Rightarrow \Delta_1, A]^v, [\Gamma_2 \Rightarrow \Delta_2]^v \\
\mathcal{E}' :: \Theta; [u/v] \Sigma \parallel \Pi', [\Gamma \Rightarrow \Delta_1, A \ast B]^w, [\Gamma_1 \Rightarrow \Delta_2]^w \end{array}
\]
\( \mathcal{E} :: \Theta; [u/v] \Sigma \parallel \Pi', [\Gamma \Rightarrow \Delta_1, A \ast B]^w, [\Gamma_1 \Rightarrow \Delta_1', A \ast B]^u \) by IH on \( \mathcal{D}' \)

25
Proposition A.3

Case $R_D = \text{NormEq, NormEmpty}$:

We present the proof for the case $R_D = \text{NormPC}$. The proofs for other rules are similar to it. There are two subcases: (a) $w$ is merged with $u$ for $w \neq u, v$, and (b) $v$ is merged with $u$.

(Subcase) $w$ is merged with $u$ for $w \neq u, v$:

In this subcase, we have $\Sigma = \Sigma', w \doteq u \circ v, w \doteq \epsilon, \Pi = \Pi', [\Gamma \implies \Delta]$. We can apply the rule $\text{NormPC}$. In this case, for some function $\pi \in [u/v]$, we have

$$\Pi = \Pi', \Gamma = \Gamma', \Delta = \Delta'. \quad \text{by IH on } \Pi'$$

Case $R_D$ is the last line in the last case because: (a) $[u/v] \Sigma_S \subseteq [u/v] \Sigma$, (b) $\pi$ satisfies $f([u/v] \Sigma_S) = [u/v] f(\Sigma_S)$, (c) $\Pi'$ is heap sequents of fresh heaps, and (d) in the case $R_D$, no heap sequents in the conclusion are modified in the premise.

Proposition A.3 (Contraction of expression relations).

If $D : \Theta, \theta, \Sigma \parallel \Pi$, then $E : \Theta, \theta, \Sigma \parallel \Pi$ and $|E| \leq |D|$.

Proof. By induction on the size of the proof $D$. Let $R_D$ be the last inference rule in the proof $D$.

Case $R_D = \bot \bot, \text{ExpCont}, \text{Cont} \epsilon \rightarrow \bot, \text{Cont} \epsilon \neq \bot, \text{Cont} \epsilon \neq \bot, \text{Cont} \epsilon \rightarrow \bot$:

This is a base case. $\bot$'s in the heap sequents of $\Theta, \theta, \Sigma \parallel \Pi$ still remain in that of $\Theta, \theta, \Sigma \parallel \Pi$ Moreover, the latter world sequent is exactly the same as the former world sequent except the expression relation $\theta$ in the former’s expression relations. Thus, by the contraction property of the expression contradiction judgment, the latter world sequent also has a proof $E = D$.

Case $R_D$ remaining rules:

We present the proof for the case $R_D = \bot \bot$. The proofs for other rules are similar to it since these rules do not focus on the expression relations. In this case, we have $\Pi = \Pi', [\Gamma, E = E' \implies \Delta]$. We can apply the rule $\text{NormPC}$. In this case, for some function $\pi \in [u/v]$, we have

$$\Pi = \Pi', [\Gamma, E = E' \implies \Delta] \quad \text{by IH on } \Pi'$$

$$\Pi = \Pi', [\Gamma, E = E' \implies \Delta] \quad \text{by IH on } \Pi'$$

We can apply the rule $R_D$ in the last line in the last case because: (a) $[u/v] \Sigma_S \subseteq [u/v] \Sigma$, (b) $\pi$ satisfies $f([u/v] \Sigma_S) = [u/v] f(\Sigma_S)$, (c) $\Pi'$ is heap sequents of fresh heaps, and (d) in the case $R_D$, no heap sequents in the conclusion are modified in the premise.

$\square$
Proposition 5.3 (Contraction of heap relations).
If \( D :: \Theta; \Sigma, \sigma \parallel \Pi \), then \( E :: \Theta; \Sigma, \sigma \parallel \Pi \) and \( |E| \leq |D| \).

Proof. By induction on the size of the proof \( D \). Let \( R_D \) be the last inference rule in the proof \( D \).

Case \( R_D = \text{L} \). ExpCont, Cont∗, Cont−→, Cont⇑, Cont⇓, Cont⇑−→.
This is a base case. L’s in the heap sequents of \( \Theta; \Sigma, \sigma \parallel \Pi \) still remain in that of \( \Theta; \Sigma, \sigma \parallel \Pi \).
Moreover, the latter world sequent is exactly the same as the former world sequent except the heap relation \( \sigma \) in the former’s expression relations. Since these rules always do not focus on both \( \sigma, \sigma \), the latter world sequent also has a proof \( E = D \).

Case \( R_D = \text{remaining rules except the rules Disj∗, Disj−→, Assoc, NormEq, NormPC, NormEmpty} \).
We present the proof for \( R_D = \ast R \). The proofs for other rules are similar to it because these rules always do not focus on both \( \sigma, \sigma \).

Now, consider the case when the rule \( \ast R \) focuses on \( \sigma \). In this case, we have \( \sigma = w \equiv w_1 \lor w_2 \).
\( D = \sigma \in \Sigma, \sigma \parallel \Pi', \Gamma \Rightarrow \Delta, A \ast B \parallel \Pi, \Gamma \Rightarrow \Delta_1, A \parallel \Pi, \Gamma \Rightarrow \Delta_1 \parallel \Pi, \Gamma \Rightarrow \Delta_2, B \parallel \Pi, \Gamma \Rightarrow \Delta_2 \parallel \Pi \) and

\[
\begin{align*}
E' :: \Theta, \theta; E = E', \Sigma \parallel \Pi', \Gamma \Rightarrow \Delta \parallel \Pi, |E'| \leq |D'|
\end{align*}
\]

by IH on \( D' \).

We can apply the rule \( = \text{L} \) in the last line because \( \Theta, \theta; \Sigma \parallel \Pi', \Gamma, E \equiv E' \Rightarrow \Delta \parallel \Pi \) is exactly the same as \( \Theta, \theta; \Sigma \parallel \Pi', \Gamma, E = E' \Rightarrow \Delta \parallel \Pi \) except the expression relation \( \theta \) in the latter’s expression relations, which the rule \( = \text{L} \) does not focus on.

\[\square\]
In this case, we have

Similarly to $R_\text{D} = \text{Disj-}\rightarrow$, the only non-trivial case is when the rule $\text{Disj-}\rightarrow$ focuses on both $\sigma, \sigma$. In this case, we have $\sigma = w = u \circ v$ and

$$\mathcal{D} = \{\sigma, \sigma\} \subset \Sigma, \sigma, \sigma \text{ fresh } w', v_1, v_2, v_3 \quad \mathcal{D}' : \Theta; \Sigma, \sigma, \sigma, u \doteq v_2 \circ v_3 \quad \Pi, \vdash [\ldots]^{w'} \text{ and } |\mathcal{E}'| \leq |\mathcal{E}'|$$

by Lemma 5.2 on $w, w'$

Similarly to $R_\text{D} = \text{Disj+}$, the only non-trivial case is when the rule $\text{Disj+}$ focuses on both $\sigma, \sigma$. In this case, we have $\sigma = w = u \circ v$ and there are two subcases.

**Subcase** The fresh heap is the union of $v$ and $v'$:

$$\mathcal{D} = \{\sigma, \sigma\} \subset \Sigma, \sigma, \sigma \text{ fresh } u' \quad \mathcal{D}' : \Theta; \Sigma, \sigma, \sigma, u' \doteq v \circ v, w \doteq w \circ u' \quad \Pi, \vdash [\ldots]^{w'} \text{ by Lemma 5.2 on } v, u'$$

by Lemma 5.2 on $w, w'$

by Proposition A.1, A.2

by Lemma 5.2 on $w, v_3$

by the rule EJoin

by the rule ENew

by IH on $\mathcal{E}'$

by the rule ECancel

**Subcase** The fresh heap is the union of $w$ and $v$:

$$\mathcal{D} = \{\sigma, \sigma\} \subset \Sigma, \sigma, \sigma \text{ fresh } u' \quad \mathcal{D}' : \Theta; \Sigma, \sigma, \sigma, u' \doteq w \circ v, w \doteq v \circ u' \quad \Pi, \vdash [\ldots]^{w'} \text{ by Lemma 5.2 on } w, v_1$$

28
Proposition A.4

Except for the rule \(\text{ExpCont} \), every logical rule of \(\text{P}_{SL}\) is invertible, i.e., if the conclusion has a proof \(D\), then the premise has a proof \(E\) with \(|E| \leq |D|\). Non-trivial cases are the following:

\[\begin{align*}
\text{(Rule } \rightarrow \text{L}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, [l \rightarrow E] \Rightarrow \Delta || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma, w \vdash [l \rightarrow E] || \Pi, \text{ by Lemma 5.2 on } w, u'. \\
\text{(Rule } \rightarrow \text{R}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma, w \vdash [w \rightarrow E] || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \neg \text{L}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, \neg A \Rightarrow A || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \neg \text{R}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, \neg A \Rightarrow A || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \lor \text{L}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, A \lor B \Rightarrow \Delta || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma, w \vdash [w \rightarrow E] || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \lor \text{R}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, A \lor B \Rightarrow \Delta || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma, w \vdash [w \rightarrow E] || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \Rightarrow \text{R}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, A => B || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma, w \vdash [w \rightarrow E] || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \Rightarrow \text{L}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, A => B || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma, w \vdash [w \rightarrow E] || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \forall \text{L}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, l \rightarrow \Delta || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma, w \vdash [w \rightarrow E] || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \forall \text{R}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, l \rightarrow \Delta || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma, w \vdash [w \rightarrow E] || \Pi, \text{ by IH on } \tilde{E}. \\
\text{(Rule } \exists \text{L}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, [x := A] \rightarrow \Delta || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma || \Pi, [x := A] \rightarrow \Delta || \Pi, \text{ for fresh } x. \\
\text{(Rule } \exists \text{R}) & \quad \text{If } D :: \Theta; \Sigma || \Pi, [x := A] \rightarrow \Delta || \Pi, \text{ then } \tilde{E} :: \Theta; \Sigma || \Pi, [x := A] \rightarrow \Delta || \Pi, \text{ for fresh } x.
\end{align*}\]

where \(|E| \leq |D|\) and \(|E'| \leq |D|\).
Proof. Since the rule \( \bot \) has no premise, it is a trivial case. Moreover, by Proposition A.2, the cases for the rules \( \ast R \), \( \rightarrow L \), \( \exists R \) are trivial.

We present the proof for the rule \( =L \). The proofs for other logical rules are similar to it. Suppose \( D :: \Theta; \Sigma || \Pi, [\Gamma, E = E' \implies \Delta]^w \). The proof proceeds by induction on the size of the proof \( D \). Let \( R_D \) be the last inference rule in the proof \( D \).

\begin{itemize}
  \item **Case** \( R_D = \bot L \), \( \text{ExpCont} \), \( \text{Cont} \rightarrow \ast \), \( \text{Cont} \rightarrow \rightarrow \), \( \exists \text{Cont} \rightarrow \ast \), \( \exists \text{Cont} \rightarrow \rightarrow \), \( \exists \text{Cont} \rightarrow \ast \), \( \exists \text{Cont} \rightarrow \rightarrow \). 
    This is a base case. \( \bot 's \) in the heap sequents of \( \Theta; \Sigma || \Pi, [\Gamma, E = E' \implies \Delta]^w \) still remain in that of \( \Theta; \Sigma || \Pi, [\Gamma \implies \Delta]^w \). Moreover, the expression relations and the heap relations of the latter world sequent contain that of the former world sequent. Thus, by the weakening property of the expression contradiction judgment, the latter world sequent also has a proof \( \mathcal{E} = D \).
  \item **Case** \( R_D \) focuses on \( E = E' \); \( \text{In PSL, } =L \) is the only rule focusing on \( E = E' \) in the truth context. Thus, the proof completes.
  \item **Case** \( R_D \) does not focus on \( E = E' \); In this case, for some \( n \), we have
    \[
    D = \frac{D_1 :: \Theta_1; \Sigma_1 || \Pi_1, [\Gamma_1, E = E' \implies \Delta_1]^w \cdots D_n :: \Theta_n; \Sigma_n || \Pi_n, [\Gamma_n, E = E' \implies \Delta_n]^w}{R_D}
    \]
    \[
    \mathcal{E}_i' :: \Theta_i, E = E'; \Sigma_i || \Pi_i, [\Gamma_i \implies \Delta_i]^w \text{ and } |\mathcal{E}_i'| \leq |D_i| \quad (i = 1, \cdots, n)
    \]
    by IH on \( D_i' \)
    \[
    \mathcal{E} :: \Theta, E = E'; \Sigma || \Pi, [\Gamma \implies \Delta]^w \text{ and } |\mathcal{E}| \leq |D|
    \]
    by the rule \( R_D \)

We can apply the rule \( R_D \) in the last line because \( \Theta; \Sigma || \Pi, [\Gamma \implies \Delta]^w \) is exactly the same as \( \Theta; \Sigma || \Pi, [\Gamma, E = E' \implies \Delta]^w \) except the formula \( E = E' \) in the former’s expression relations and the same formula in the latter’s heap sequent of \( w \), which the rule \( R_D \) does not focus on.
\end{itemize}

\( \square \)

**Proposition A.5** (Contraction).

If \( D :: \Theta; \Sigma || \Pi, [\Gamma, A, A \implies \Delta]^w \), then \( \mathcal{E} :: \Theta; \Sigma || \Pi, [\Gamma \implies \Delta]^w \) and \( |\mathcal{E}| \leq |D| \).

If \( D :: \Theta; \Sigma || \Pi, [\Gamma \implies \Delta, A]^w \), then \( \mathcal{E} :: \Theta; \Sigma || \Pi, [\Gamma \implies \Delta, A]^w \) and \( |\mathcal{E}| \leq |D| \).

**Proof.** By simultaneous induction on the size of proofs of \( \Theta; \Sigma || \Pi, [\Gamma, A, A \implies \Delta]^w \) and \( \Theta; \Sigma || \Pi, [\Gamma \implies \Delta, A]^w \).

Let us first prove the former statement. Suppose \( D :: \Theta; \Sigma || \Pi, [\Gamma, A, A \implies \Delta]^w \). Let \( R_D \) be the last inference rule in the proof \( D \).

\begin{itemize}
  \item **Case** \( R_D = \bot L \):
    This is a base case. If \( \bot \) does not focus on \( A \), then \( \bot 's \) in the heap sequents of \( \Theta; \Sigma || \Pi, [\Gamma, A, A \implies \Delta]^w \) still remain in that of \( \Theta; \Sigma || \Pi, [\Gamma, A \implies \Delta]^w \). If \( \bot L \) focuses on \( A \), then \( A = \bot \) holds and at least one \( \bot \) remains in the truth context of \( w \) in \( \Theta; \Sigma || \Pi, [\Gamma, A \implies \Delta]^w \). Thus, we have \( D :: \Theta; \Sigma || \Pi, [\Gamma, A \implies \Delta]^w \).
  \item **Case** \( R_D = \text{ExpCont} \), \( \text{Cont} \rightarrow \ast \), \( \text{Cont} \rightarrow \rightarrow \), \( \exists \text{Cont} \rightarrow \ast \), \( \exists \text{Cont} \rightarrow \rightarrow \), \( \exists \text{Cont} \rightarrow \ast \), \( \exists \text{Cont} \rightarrow \rightarrow \). 
    \( \Theta; \Sigma || \Pi, [\Gamma, A, A \implies \Delta]^w \) is exactly the same as \( \Theta; \Sigma || \Pi, [\Gamma, A \implies \Delta]^w \) except the formula \( A \) in the former’s heap sequents, which \( R_D \) does not focus on. Hence, we have \( D :: \Theta; \Sigma || \Pi, [\Gamma, A \implies \Delta]^w \).
  \item **Case** \( R_D \) focuses on \( A \): \( \text{Subcase} \) \( R_D = \neg L, \forall L, \exists L \):
    We present the proof for the subcase \( R_D = \neg L \). The proofs for other rules are similar to it. In this subcase, we have \( A = \neg B \) and
    \[
    D = \frac{D' :: \Theta_1; \Sigma_1 || \Pi_1, [\Gamma_1, \neg B \implies \Delta, B]^w_1 \cdots D_n :: \Theta_n; \Sigma_n || \Pi_n, [\Gamma_n, \neg B \implies \Delta, B]^w_n}{\neg L}
    \]
    \[
    \mathcal{E}_i' :: \Theta_i; \Sigma_i || \Pi_i, [\Gamma_i \implies \Delta, B]^w_1 \cdots \Delta]^w_n \text{ and } |\mathcal{E}_i'| \leq |D_i'|
    \]
    by Proposition A.4
    \[
    \mathcal{E}_i'' :: \Theta_i; \Sigma_i || \Pi_i, [\Gamma_i \implies \Delta, B]^w_1 \cdots \Delta]^w_n \text{ and } |\mathcal{E}_i''| \leq |\mathcal{E}_i'|
    \]
    by IH on \( \mathcal{E}_i' \)
    \[
    \mathcal{E} :: \Theta; \Sigma || \Pi, [\Gamma, \neg B \implies \Delta]^w \text{ and } |\mathcal{E}| \leq |D|
    \]
    by the rule \( \neg L \)
\end{itemize}
(Subcase) $R_D \Rightarrow \ll_L$: We present the proof for the subcase $R_D \Rightarrow \ll_L$. The proof for the rule $\ll_L$ is similar to it. In this subcase, we have $A = \Gamma \rightarrow E$ and

$$D = \frac{D' : \Theta; \Sigma, w \vdash [l \rightarrow E] \triangleright \Pi, [\Gamma, [l \rightarrow E] \Rightarrow \Delta^w]}{\Theta; \Sigma \parallel \Pi, [\Gamma \rightarrow E], l \rightarrow E \Rightarrow \Delta^w} \Rightarrow \ll_L$$

by Proposition A.4

$E'' : \Theta; \Sigma, w \vdash [l \rightarrow E], w \vdash [l \rightarrow E] \triangleright \Pi, [\Gamma \Rightarrow \Delta^w]$ and $|E'| \leq |D'|$ by Proposition 5.3

$E : \Theta; \Sigma \parallel \Pi, [\Gamma \rightarrow E] \Rightarrow \Delta^w$ and $|E| \leq |D|$ by the rule $\Rightarrow \ll_L$

(Subcase) $R_D = \Rightarrow L$: In this subcase, we have $A = B \Rightarrow C$ and

$$D' = \frac{\text{fresh } w_1, w_2}{\Theta; \Sigma, w \vdash w_1 \circ w_2 \parallel \Pi, [\Gamma, B \rightarrow C \Rightarrow \Delta^w, [B \Rightarrow \Delta^w], [C \Rightarrow \Delta^w]} \ll_L$$

by Proposition A.4

$E_1' : \Theta; \Sigma, w \vdash w_1 \circ w_2, w \vdash w'_1 \circ w'_2 \parallel \Pi, [\Gamma \Rightarrow \Delta^w, [B \Rightarrow \Delta^w], [C \Rightarrow \Delta^w}$ for fresh $w'_1, w'_2$, and $|E_1'| \leq |D'|$

$E_2' : \Theta; \Sigma, w \vdash w_1 \circ w_2, w \vdash w_1 \circ w_2 \parallel \Pi, [\Gamma \Rightarrow \Delta^w, [B \Rightarrow \Delta^w], [C \Rightarrow \Delta^w}$ and $|E_2'| \leq |E_1'|$

$E_3' : \Theta; \Sigma, w \vdash w_1 \circ w_2, w \vdash w_1 \circ w_2 \parallel \Pi, [\Gamma \Rightarrow \Delta^w, [B \Rightarrow \Delta^w], [C \Rightarrow \Delta^w}$ and $|E_3'| \leq |E_2'|$

$E_4' : \Theta; \Sigma, w \vdash w_1 \circ w_2 \parallel \Pi, [\Gamma \Rightarrow \Delta^w, [B \Rightarrow \Delta^w], [C \Rightarrow \Delta^w}$ and $|E_4'| \leq |E_3'|$

by Proposition 5.3

$E_5' : \Theta; \Sigma, w \vdash w_1 \circ w_2 \parallel \Pi, [\Gamma \Rightarrow \Delta^w, [B \Rightarrow \Delta^w], [C \Rightarrow \Delta^w}$ and $|E_5'| \leq |E_4'|$ by IH on $E_4'$

$E_6' : \Theta; \Sigma, w \vdash w_1 \circ w_2 \parallel \Pi, [\Gamma \Rightarrow \Delta^w, [B \Rightarrow \Delta^w], [C \Rightarrow \Delta^w}$ and $|E_6'| \leq |E_5'|$ by IH on $E_5'$

by the rule $\ll_L$

(Subcase) $R_D = \Leftarrow L$: In this subcase, we have $A = B \Leftarrow C$ and

$$D = \frac{D' : \Theta; \Sigma, w \vdash w \circ w_1 \in \Sigma}{\Theta; \Sigma \parallel \Pi, [\Gamma, B \rightarrow C \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w]} \ll_L$$

by IH on $D'$

$E_1' : \Theta; \Sigma \parallel \Pi, [\Gamma, B \Rightarrow C \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w}$ and $|E_1'| \leq |D'|$

$E_2' : \Theta; \Sigma \parallel \Pi, [\Gamma, B \Rightarrow C \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w}$ and $|E_2'| \leq |D'|$

by IH on $D'$

by IH on $D'$

$E : \Theta; \Sigma \parallel \Pi, [\Gamma, B \rightarrow C \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w, [\Gamma \Rightarrow \Delta^w]$ and $|E| \leq |D|$ by the rule $\Rightarrow L$

(Subcase) $R_D = \Rightarrow L$: In this subcase, we have $A = E = E'$ and

$$D = \frac{D' : \Theta; E = E; \Sigma \parallel \Pi, [\Gamma, E = E' \Rightarrow \Delta^w]}{\Theta; \Sigma \parallel \Pi, [\Gamma, E = E' \Rightarrow \Delta^w]} \Rightarrow L$$

by Proposition A.4

$E'' : \Theta; E = E'; \Sigma \parallel \Pi, [\Gamma \Rightarrow \Delta^w]$ and $|E'| \leq |D'|$

$E : \Theta; \Sigma \parallel \Pi, [\Gamma, E = E' \Rightarrow \Delta^w]$ and $|E| \leq |D|$ by the rule $\Rightarrow L$

Case $R_D$ does not focus on $A$: In this case, for some $n$, we have

$$D = \frac{D_1' : \Pi_1, [\Gamma_1, A, A \Rightarrow \Delta_1^w]}{\Theta; \Sigma \parallel \Pi, [\Gamma, A, A \Rightarrow \Delta^w]} = L$$
Let us then prove the latter statement. Suppose \( D :: \Theta; \Sigma \parallel \Pi, [\Gamma, A \implies \Delta] \). Let \( R_D \) be the last inference rule in the proof \( D \).

**Case** \( R_D = \perp L, \text{ExpCont}, \text{Cont} \nRightarrow \text{Cont} \neq \perp, \text{Cont} \nRightarrow \nRightarrow, \text{Cont} \nRightarrow \equiv \text{Cont} \). This is a base case. Since these rules do not focus on the falsehood context, the proof completes.

**Case** \( R_D \) focuses on \( A \):

(Subcase) \( R_D = \forall R; \)

In this subcase, we have \( A = B \lor C \) and

\[
D = \frac{D' :: \Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta, B \lor C] \quad \forall R}{\Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta, B \lor C]}
\]

\( \varepsilon \) by IH on \( \varepsilon' \)

\( \varepsilon' \) by IH on \( \varepsilon' \)

\( \varepsilon \) by the rule \( \forall R \)

(Subcase) \( R_D = \exists R; \)

In this subcase, we have \( A = \exists b.B \) and

\[
D = \frac{D' :: \Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta, \exists b.B] \quad \exists R}{\Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta, \exists b.B]}
\]

\( \varepsilon \) by IH on \( \varepsilon' \)

\( \varepsilon' \) by the rule \( \exists R \)

These subcases are similarly proved to the above subcases for the corresponding left rules.

(Subcase) \( R_D = \ast R, \rightarrow R, \text{IR}, \text{=R}; \)

Each subcase is similarly proved to the above subcase for \( R_D = \rightarrow L, \ast L \), respectively.

**Case** \( R_D \) does not focus on \( A; \)

This case is similarly proved to the above corresponding case.

\[ \square \]

**Lemma A.6.**

If \( \Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta] \), then \( \Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta] \).

**Proof.** By induction on the size of a proof of \( \Theta; \Sigma \parallel \Pi, [\Gamma \implies \Delta] \).

\[ \square \]

**Lemma A.7.**

If \( \Theta_1, E \neq E'; \Sigma_1 \parallel \Pi_1 \) and \( \Theta_2, E = E'; \Sigma_2 \parallel \Pi_2 \), then \( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \parallel \Pi_2 \).

**Proof.** By nested induction on the size of \( \Theta_1, E \neq E'; \Sigma_1 \parallel \Pi_1 \) and \( \Theta_2, E = E'; \Sigma_2 \parallel \Pi_2 \). In the proof, we use the following four assumptions: weakening, contraction, and cut property of the expression contraction judgment, and that either \( \Theta, \theta \vdash \bot \) or \( \Theta, \neg \theta \vdash \bot \) holds.

**Lemma A.8.**

If \( \Theta_1; \Sigma_1, w \neq \epsilon \parallel \Pi_1 \) and \( \Theta_2; \Sigma_2, w = \epsilon \parallel \Pi_2 \), then \( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \parallel \Pi_2 \).

**Proof.** By \( \Theta_1; \Sigma_1, w \neq \epsilon \parallel \Pi_1 \) and \( \Theta_2; \Sigma_2, w = \epsilon \parallel \Pi_2 \), that the either \( \Theta, \theta \vdash \bot \) or \( \Theta, \neg \theta \vdash \bot \) holds.

**Lemma A.8.**

If \( \Theta_1; \Sigma_1, w \neq \epsilon \parallel \Pi_1 \) and \( \Theta_2; \Sigma_2, w = \epsilon \parallel \Pi_2 \), then \( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \parallel \Pi_2 \).
Proof. We prove the former statement by nested induction on the size of (a) a proof $D$ of $\Theta_1; \Sigma_1, w \not\in E \mid \Pi_1$, and (b) a proof $E$ of $\Theta_2; \Sigma_2, w \equiv E \mid \Pi_2$. Similarly, we prove the latter statement by nested induction on the size of (a) a proof $D$ of $\Theta_1; \Sigma_1, w \not\equiv [I \Rightarrow E] \mid \Pi_1$, and (b) a proof $E$ of $\Theta_1; \Sigma_1, w \equiv [I \Rightarrow E] \mid \Pi_2$. In the proof of the latter statement, we use two assumptions: (1) $\Theta \vdash E_1 = E_2$ and $\Theta \vdash E_2 = E_3$ imply $\Theta \vdash E_1 = E_3$; (2) $\Theta \vdash E_1 = E_2$ and $\Theta \vdash E_2 \neq E_3$ imply $\Theta \vdash E_1 \neq E_3$.

Lemma A.9.

Suppose

\[
\frac{\Theta_1; \Sigma_1 \mid \Pi_1 \cdots \Theta_n; \Sigma_n \mid \Pi_n}{\Theta; \Sigma \mid \Pi}
\]

R holds for some rule R in $P_{SL}$.

Then, we have the followings:

- \[
\frac{\Theta_1, \Theta' \mid \Pi_1 \cdots \Theta_n, \Theta'; \Sigma_n \mid \Pi_n}{\Theta, \Theta' \mid \Sigma \mid \Pi} \text{ for any expression relations } \Theta', \text{ and any rule } R.
\]

- \[
\frac{\Theta_1; \Sigma_1, \Sigma' \mid \Pi_1 \cdots \Theta_n; \Sigma_n, \Sigma' \mid \Pi_n}{\Theta; \Sigma, \Sigma' \mid \Pi} \text{ for any heap relations } \Sigma', \text{ any heap sequents } \Pi', \text{ and any rule } R \text{ except NormEq, NormPC, NormEmpty}.
\]

Proof. By case analysis of the rule R.

\[\square\]

Lemma 5.4.

If $D :: \Theta_1; \Sigma_1 \mid \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, C]^w$ and $E :: \Theta_2; \Sigma_2 \mid \Pi_2, [\Gamma_2, C \Rightarrow \Delta_2]^w$, then $\Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \mid \Pi_1 \uplus \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w$.

Proof. By nested induction on the size of (a) cut formula $C$, (b) proof $D$ of $\Theta_1; \Sigma_1 \mid \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, C]^w$, and (c) proof $E$ of $\Theta_2; \Sigma_2 \mid \Pi_2, [\Gamma_2, C \Rightarrow \Delta_2]^w$. Let $R_D$ be the last inference rule in the proof $D$, and $R_E$ be the last inference rule in the proof $E$.

Suppose $|D| = 1$ or $|E| = 1$. Then, we have the following cases.

Case $R_D = \bot$. $\Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \mid \Pi_1 \uplus \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w$ contains the truth contexts of $\Theta_1; \Sigma_1 \mid \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, C]^w$. Thus, the latter world sequent also has a proof.

Case $R_D = \text{ExpCont}$. $\Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \mid \Pi_1 \uplus \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w$ contains the expression relations and the heap relations of $\Theta_1; \Sigma_1 \mid \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, C]^w$. Thus, the latter world sequent also has a proof.

Case $R_E = \bot$. If $R_E$ does not focus on $C$, then $\Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \mid \Pi_1 \uplus \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w$ have at least one $\bot$ in its truth contexts since the union of $\Gamma_2$ and the truth contexts of $\Pi_2$ has at least one $\bot$. Thus, the proof completes. If $R_E$ focuses on $C$, then $\Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \mid \Pi_1 \uplus \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w$ by Lemma A.6

by Proposition A.1, A.2

Case $R_E = \text{ExpCont}$. This case is similarly proved to the case $R_D = \text{ExpCont}$.

Now, suppose $|D| \geq 2$ and $|E| \geq 2$. Then, we have the following cases.

Case $R_D$ and $R_E$ focus on $C$:

(Contcase)

We present the proof for the subcase $R_D = \lor R$. The proof for the rule $\neg R$ is similar to it. In this subcase, we have $C = A \lor B$ and

\[
D = \frac{D' :: \Theta_1; \Sigma_1 \mid \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, A, B]^w \lor R, E \equiv \frac{\varepsilon' :: \Theta_2; \Sigma_2 \mid \Pi_2, [\Gamma_2, A \Rightarrow \Delta_2]^w}{\Theta_2; \Sigma_2 \mid \Pi_2, [\Gamma_2, A \lor B \Rightarrow \Delta_2]^w}}{\Theta_1; \Sigma_1 \mid \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, A \lor B]^w \lor L}
\]

\[\square\]
\[ D'' : \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, B]^w \] by IH on \( A, D', E'_1 \)

\[ \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, 2]^w \] by IH on \( B, D'', E'' \)

\( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \) by Proposition A.3, 5.3, A.5

(Subcase) \( R_D = \Rightarrow \mathbb{R}, \mathbb{R} \)

We present the proof for the subcase \( R_D = \mathbb{R} \). The proof for the rule \( \Rightarrow \mathbb{R} \) is similar to it. In this subcase, we have \( C = 1 \) and

\[ D = \frac{\Theta_1; \Sigma_1, w \neq \epsilon \parallel \Pi_1, [\Gamma_1 \Rightarrow \Delta_1]^w}{\Theta_1; \Sigma_1 \parallel \Pi_1, [\Gamma_1 \Rightarrow \Delta_1]^w} \mathbb{R}L \]

\[ \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \] by Lemma A.8

(Subcase) \( R_D = \ast \mathbb{R}, \ast \mathbb{R} \)

We present the proof for the subcase \( R_D = \ast \mathbb{R} \). The proof for the rule \( \Rightarrow \ast \mathbb{R} \) is similar to it. In this subcase, we have \( C = A \ast B \). Suppose \( \ast \mathbb{R} \) focuses on \( w \equiv u \circ v \in \Sigma_1 \). Without loss of generality, we can assume that there are heap sequents of \( u, v \) in \( \Pi_2 \). Thus, we have \( \Pi_1 = \Pi_1', [\Gamma_1 \Rightarrow \Delta_1]^w, [\Gamma_1 \Rightarrow \Delta_1^v], \Pi_2 = \Pi_2', [\Gamma_2 \Rightarrow \Delta_2]^w, [\Gamma_2 \Rightarrow \Delta_2^v] \), and

\[ D = \frac{\Theta_1; \Sigma_1 \parallel \Pi_1', [\Gamma_1 \Rightarrow \Delta_1, A \ast B]^w}{\Theta_2; \Sigma_2 \parallel \Pi_2', [\Gamma_2 \Rightarrow \Delta_2^w]} \ast L \]

Lemma 5.2 on \( E' \) with \( u, w_1 \) and \( v, w_2 \)

\[ [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w, \]

\[ [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1^u, \Delta_2^v, A]^w, \]

\[ [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, B]^v \]

by IH on \( A \ast B, D'_1, E' \)

\[ \mathcal{E}' : \Theta_2; \Sigma_2 \parallel \Pi_2', [\Gamma_2 \Rightarrow \Delta_2^w] \]

Lemma 5.2 on \( E' \) with \( u, w_1 \) and \( v, w_2 \)

\[ [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, 2]^w, \]

\[ [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1^u, \Delta_2^v, A]^w, \]

\[ [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2, B]^v \]

by IH on \( A, D'^u, E'' \)

\[ \mathcal{E}_1' : \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w, \]

\[ \mathcal{E}_2' : \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w, \]

by Proposition A.3, 5.3, A.5

\( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \) by IH on \( B, D''_1, E''_1 \) and Proposition A.3, 5.3, A.5

(Subcase) \( R_D = \exists \mathbb{R} \)

In this subcase, we have \( C = \exists a.A \) and

\[ D = \frac{\Theta_1; \Sigma_1 \parallel \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, \exists a.A]^w}{\Theta_1, \Sigma_1 \parallel \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, \exists a.A]^w} \exists L \]

\[ \mathcal{E} = \frac{\text{fresh } x}{\Theta_2; \Sigma_2 \parallel \Pi_2, [\Gamma_2, [x/a]A \Rightarrow \Delta_2]^w} \exists L \]

\( \mathcal{D}' : \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \) by IH on \( \exists a.A, D', E' \)

\( \mathcal{E}'' : \Theta_2; \Sigma_2 \parallel \Pi_2, [\Gamma_2, [x/a]A \Rightarrow \Delta_2]^w \) by substituting \( x \) with \( E \)

\( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \) by IH on \( [x/a]A, D'', E'' \)

\( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \lor \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \) by Proposition A.3, 5.3, A.5

34
(Subcase) \( R_D \) does not focus on \( C \):

In this subcase, we have \( C = E = E' \) and

\[
\mathcal{D} = \frac{\Theta_1, E \neq E'; \Sigma_1 \parallel \Pi_1, [\Gamma_1 \Rightarrow \Delta_1]^w}{\Theta_1; \Sigma_1 \parallel \Pi_1, [\Gamma_1 \Rightarrow \Delta_1, E = E']^w}, \quad \mathcal{E} = \frac{\Theta_2, E = E'; \Sigma_2 \parallel \Pi_2, [\Gamma_2 \Rightarrow \Delta_2]^w}{\Theta_2; \Sigma_2 \parallel \Pi_2, [\Gamma_2, E = E' \Rightarrow \Delta_2]^w} = L
\]

\( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \cup \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \) by Lemma A.7

Case \( R_D \) does not focus on \( C \):

We present the proof for the subcase \( R_D = \text{NormEq} \), \( \text{NormPC} \), \( \text{NormEmpty} \):

We present the proof for the subcase \( R_D = \text{NormPC} \). The proofs for other rules are similar to it. In this subcase, we have two possibilities: (a) \( \text{NormPC} \) focuses on \( u \neq v \lor v' \) for some \( u \neq w, v \neq w \) and (b) \( \text{NormPC} \) focuses on \( w \neq v \lor v' \) (or on \( u \neq v \lor v' \)). Let us consider the first possibility. Without loss of generality, we can assume that there are heap sequents of \( u, v \) in \( \Pi_1 \). Then, we have \( \Sigma_1 = \Sigma'_1, u \neq v \lor v' = \epsilon \) and \( \Pi_1 = \Pi'_1 \), \( \Gamma_1u \Rightarrow \Delta_1u \), \( [\Gamma_1v \Rightarrow \Delta_1v] \) and \( \Pi_2 = \Pi'_2 \), \( \Gamma_2u \Rightarrow \Delta_2u \), \( \Pi_2 \Rightarrow \Delta_2v \), and

\[
\mathcal{D} = \frac{\mathcal{D}' :: \Theta_1; [u/v]\Sigma_1v, v' = \epsilon \parallel \Pi_1, \Gamma_1u \Rightarrow \Delta_1u, [\Gamma_1v \Rightarrow \Delta_1v] \land [\Gamma_1 \Rightarrow \Delta_1, C]^w}{\Theta_1; \Sigma_1, u \neq v \lor v' = \epsilon \parallel \Pi_1, \Gamma_1u \Rightarrow \Delta_1u, \Pi'_2, \Gamma_2u \Rightarrow \Delta_2u, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w}
\]

by \( \text{NormPC} \)

\( \mathcal{E}' :: \Theta_2; [u/v]\Sigma_2, v' = \epsilon \parallel \Pi_2, \Gamma_1u \Rightarrow \Delta_1u, \Pi'_2, \Gamma_2u \Rightarrow \Delta_2u, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \)

by \( \text{IH on } C, \mathcal{D}', \mathcal{E}' \)

\( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \cup \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \)

by the rule \( \text{NormPC} \)

The proof for the second possibility is similar to the above.

(Subcase) \( R_D \) = remaining rules:

In this case, for some \( n \), we have

\[
\mathcal{D} = \mathcal{D}' :: \Theta'_1; \Sigma_1' \parallel \Pi_1', [\Gamma'_1 \Rightarrow \Delta'_1, C'^w] \quad \cdots \quad \mathcal{D}' :: \Theta'_n; \Sigma_1' \parallel \Pi_1', [\Gamma'_n \Rightarrow \Delta'_n, C'^w]
\]

by \( \text{IH on } C, \mathcal{D}', \mathcal{E} \)

\( \Theta_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1 \cup \Pi_2, [\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2]^w \)

We can apply the rule \( R_D \) in the last line for the following reason: since \( R_D \) does not focus on \( C \), we have

\[
\Theta'_1; \Sigma_1' \parallel \Pi_1', [\Gamma'_1 \Rightarrow \Delta'_1]^w \quad \cdots \quad \Theta'_n; \Sigma_1' \parallel \Pi_1', [\Gamma'_n \Rightarrow \Delta'_n]^w
\]

and since \( R_D \neq \text{NormEq}, \text{NormPC}, \text{NormEmpty}, \) by using Lemma A.9 we have

\[
\Theta'_1, \Theta_2; \Sigma_1, \Sigma_2 \parallel \Pi_1' \cup \Pi_2, [\Gamma'_1, \Gamma_2 \Rightarrow \Delta'_1, \Delta'_2]^w
\]

\( \cdots \)

by \( \text{IH on } C, \mathcal{D}' \)

Case \( R_E \) does not focus on \( C \):

The proof for this case is similar to the above case when \( R_D \) does not focus on \( C \).
B Soundness and completeness of $P_{SL}$

B.1 Soundness

**Lemma 6.2.** For every inference rule with the conclusion $W$ and the premise consisting of $W_1, \cdots, W_n$, it holds that $[W]_S$ implies $\bigvee_{i=1,\cdots,n} [W_i]_S$ for any stack $S$. If $n=0$, we have $\neg[W]_S$.

**Proof.** The proof is straightforward. Let $R$ be an inference rule for which we want to prove the above statement.

Case $R = \bot$, $\expcon$, $\cont \mapsto$; $\cont \notin \bot$:
It is obvious to prove that $\neg[W]_S$ holds for this case.

Case $R =$ remaining logical rules:
This case is obvious by the definitions of $[W]_S$ and the semantics of separation logic.

Case $R = \disj\ast$, $\disj \mapsto$:
Let us consider the case $R = \disj\ast$. The proof for the rule $\disj \mapsto$ is similar to it. In this case, $W = \Theta; \Pi || \Sigma$ and $\{w \upharpoonright u_1 \circ w_2, w = v_1 \circ v_2\} \subseteq \Sigma$.

Suppose $[W]_S$ holds. By assumption, we get $w = u_1 \circ u_2, w = v_1 \circ v_2$. Let $D_i = \dom(u_i)$, $D'_i = \dom(v_i)$, and $E_{ij} = D_i \cap D'_j$ $(i, j = 1, 2)$. Then, since $D_1 \cup D_2 = D'_1 \cup D'_2$ and $D_1 \cap D_2 = D'_1 \cap D'_2 = \emptyset$ hold, $D_1 = E_{11} \cup E_{12}, D'_2 = E_{21} \cup E_{22}$ and $E_{ij} \cap E_{ij} = \emptyset$ hold for $i, j = 1, 2$ and $(k, l) \neq (i, j)$. Therefore, for some heaps $w_{ij}$ $(i, j = 1, 2)$ whose domain is equal to $E_{ij}$, we have $u_i = w_{i1} \circ w_{i2}, v_j = w_{j1} \circ w_{j2}$. Hence, $[W]_S$ holds.

Case $R = \prop\epsilon$, $\prop \mapsto$, $\prop \notin \epsilon$, $\prop \mapsto \neq$:
Let us consider the case $R = \prop\epsilon$. The proofs for other rules are similar to it. In this case, $W = \Theta; \Pi || \Sigma$ and $\{w \notin \{l \mapsto E\}, w \upharpoonright w_1 \circ w_2\} \subseteq \Sigma$.

Given $w = w_1 \circ w_2$, it is easy to check that $w = \langle [E]_S \mapsto [E]_S \rangle$ iff. $(w_1 \upharpoonright \epsilon \wedge w_2 = ([E]_S \mapsto [E]_S)) \vee (w_1 = ([E]_S \mapsto [E]_S) \wedge w_2 \upharpoonright \epsilon)$. Thus, given $w = w_1 \circ w_2$, we know that $w \notin ([E]_S \mapsto [E]_S)$ iff. $(w_1 \neq \epsilon \vee w_2 \notin ([E]_S \mapsto [E]_S)) \wedge (w_1 \neq ([E]_S \mapsto [E]_S) \vee w_2 \neq \epsilon), i.e., (w_1 \neq \epsilon \wedge w_2 \notin ([E]_S \mapsto [E]_S)) \vee (w_1 \neq \epsilon \wedge w_2 \notin \epsilon) \vee (w_2 \notin ([E]_S \mapsto [E]_S) \wedge w_1 \neq ([E]_S \mapsto [E]_S)) \vee (w_2 \notin ([E]_S \mapsto [E]_S) \wedge w_2 \neq \epsilon)$. Now, suppose $[W]_S$ holds. Then, by assumption, we get $w = w_1 \circ w_2, w \notin ([E]_S \mapsto [E]_S)$, and from the above observation, $\bigvee_{i=1,\cdots,n} [W_i]_S$ holds.

Case $R =$ remaining structural rules and heap contradiction rules:
The proofs for the rules $\cont \mapsto =, \cont \mapsto \neq, \cont \mapsto \epsilon$ are obvious since “$[W]_S$ implies $[W_1]_S$” is just weakening for these rules. The proofs for other rules are immediate from associativity, neutrality, and cancellativity of heaps.

\[\square\]

B.2 Completeness

B.2.1 Completeness of the invertible rules

**Proposition 6.5.** Except for the rules $\bot$, $\expcon$, $\ast R$, $\ast L$, and $\weak$, every logical or structural rule with the conclusion $W$ and the premise consisting of $W_1, \cdots, W_n$ is invertible in that $\bigvee_{i=1,\cdots,n} [W_i]_S$ implies $[W]_S$ for any stack $S$.

**Proof.** The proof is straightforward. Let $R$ be an inference rule for which we want to prove the above statement.

Case $R =$ logical rules except $\ast R$, $\ast L$, $\expcon$:
This case is obvious by the definitions of $[W]_S$ and the semantics of separation logic.

Case $R =$ structural rules:
The proofs for the rules $\normeq$, $\normpc$, $\emptyset$ are immediate from neutrality and cancellativity of heaps. The proofs for other rules are obvious since “$[W]_S$ implies $[W]_S$” is just weakening for these rules.

\[\square\]
B.2.2 Preliminary lemmas

From now on, we assume that there exists a unique heap relation in the following world sequents. This assumption can be achieved by applying the following rule Contraction (which is admissible) whenever we get duplicate heap relations:

\[ \Theta; \Sigma, \sigma \models \Pi \quad \text{Contraction} \]

(Note that if \( \{W\} \uparrow \downarrow \{W'\}\), then \( [[\{W\}]_S \uparrow \downarrow [[\{W'\}]_S \) for any stack \( S \). Thus, we can apply the rule Contraction without losing completeness.) Moreover, we assume that for every heap relation \( w \uparrow u \circ v \) in the following world sequents, \( w, u, v \) are all distinct. In order to keep this assumption, we only need to check that the applications of the rules Assoc, NormPC, NormEq in Lemma 6.7 retains this assumption, and it is easy to check this claim. For details, this claim holds because we apply each of the rules to a world sequent \( W \) in the following specific conditions:

- We apply Assoc only when \( W \) is elementary.
- We apply NormPC only when \( W \) is non-cyclic and \( \rho \)-consistent (which will be defined later), where \( \rho \) is an empty and terminal heap in \( W \).
- We apply NormEq only when \( W \) is elementary and consistent.

Finally, for simple notations, we will use the following notational abuse:

- For a heap \( W \) and a world sequent \( W \), \( w \in W \) means that \( W \) contains a heap sequent for \( w \).
- For a heap relation \( \sigma \) and a world sequent \( W, \sigma \in W \) means that heap relations of \( W \) contain \( \sigma \).

We introduce some more definitions about the properties of graphs of heaps, and then prove auxiliary Lemmas which tells us what properties of world sequents are preserved after applying each of the structural rules. The next subsection will be substantially based on these Lemmas.

Definition B.1.

- \( w \nrightarrow u \) \iff \( w \nrightarrow u \) does not hold.
- \( w \land u \) means that heaps \( w \) and \( u \) are disjoint: there is at least one common ancestor \( v \) of \( w \) and \( u \) along with two heaps \( w' \) and \( u' \) such that \( w \nrightarrow w' \), \( u \nrightarrow u' \), and \( v \circ u' \). We write \( w \land u \) to indicate that heap \( v \) is such a common ancestor of heaps \( w \) and \( u \).
- \( w \lor u \) means that heaps \( w \) and \( u \) share a common descendant \( v \) such that both \( v \nrightarrow w \), \( v \neq u \) and \( v \nrightarrow u \), \( v \neq u \) hold. We write \( w \lor u \) to indicate that heap \( v \) is such a common descendant of heaps \( w \) and \( u \).
- Whenever we are unclear about a world sequent that we are talking about, we will add a subscripted world sequent to existing definitions such as \( w \nrightarrow_W u \), \( w \nleftrightarrow_W u \) and \( T_W (u) \).
- \( W \) is non-cyclic \iff \( \nrightarrow \) is a partial ordering on heaps of \( W \).
- \( W \) is consistent at \( w \) \iff \( w \equiv u_1 \circ u_2 \) and \( w \equiv v_1 \circ v_2 \) then, \( T(u_1) \cup T(u_2) = T(v_1) \cup T(v_2) \).

Lemma B.2. Let \( W \) be any world sequent. Then, the following holds:

- If \( W \) is non-cyclic, then for any heap \( w \), \( |T(w)| \geq 1 \) holds.
- If \( W \) is elementary, then \( W \) is non-cyclic.
- If \( W \) is elementary, then for any heap \( w \) with \( |T(w)| = 1 \), \( w \) is a terminal heap in \( W \).
- If \( W \) is consistent, then for any two terminal heaps \( u \neq v \) in \( W \), \( u \land v \) holds.
- If \( W \) is consistent at a heap \( w \), then for any heaps \( u, v \) in \( W \) with \( w \equiv u \circ v \), \( T(w) = T(u) \cup T(v) \) holds.

Proof. The proof is straightforward. \( \square \)

Lemma B.3 (Preservation lemma for the rules Disj\( \rightarrow \star \), Disj\( \star \)).

- The rule Disj\( \rightarrow \star \) does not preserve elementary-ness.
• The rule Disj* does not preserve elementary-ness but preserves a unique root heap.

• Let \( W, W' \) be any world sequent such that \( W \) is elementary and \( \{ W \} \mapsto \{ W' \} \) by applying only the rule Disj*. Then, for any heaps \( u, v \in W, u \not\nearrow W v \iff u \not\nearrow W' v \) and \( u \land W v \iff u \land W' v \).

Proof. Applying Disj to \( \{ w_1 \equiv u_1 \land u_2, w_2 \equiv u_2 \land u_3, w_2 \equiv u_1 \land u_4 \} \) with \( \{ w_1 \equiv u_1 \land u_2, w_2 \equiv u_2 \land u_3 \} \) is an example for Disj*. Applying Disj* to \( \{ u \equiv w_2 \equiv w_3, w \equiv w_2 \equiv w_3, \} \) with \( \{ w \equiv w_2 \equiv w_3, w \equiv w_1 \equiv w_2 \equiv w_3 \} \) is an example for Disj*. The proofs for the remaining claims are straightforward.

Lemma B.4 (Preservation lemma for the rule Assoc).

• Let \( W, W' \) be any world sequent such that \( \{ W \} \mapsto \{ W' \} \) by applying only the rule Assoc. Then, for any heaps \( u, v \in W, u \not\nearrow W v \iff u \not\nearrow W' v \).

• The rule Assoc preserves consistency, full-ness, **-ready-ness, ↔-ready-ness, and sanitizedness.

Proof. The proof is straightforward.

Lemma B.5 (Preservation lemma for the propagation rules and the rule Weaken).

• The propagation rules preserve **-ready-ness and →-ready-ness.

• The rule Weaken preserves **-ready-ness and →-ready-ness.

Proof. The proof is straightforward.

Lemma B.6 (Preservation lemma for the rule NormEq).

Let \( W \) and \( W' \) be any world sequents such that \( \{ W \} \mapsto \{ W' \} \) by applying the rule NormEq to \( \{ w \equiv w_1 \land w_2, w' \equiv w_1 \land w_2 \} \). Then, \( T_W(u) = T_{W'}([v/w']u) \).

• If \( W \) is consistent, then for any heap \( u \in W \) we have \( T_W(u) = T_{W'}([v/w']u) \).

• If \( W \) is **-ready (resp. →-ready) for heap \( u \in W \) and sanitized, then \( W' \) is **-ready (resp. →-ready) for heap \([v/w']u \in W' \) and sanitized.

Proof. Let us prove the first claim. Suppose that \( W \) is consistent, and choose any heap \( u \in W \). We have the following cases for proving that \( T_W(u) = T_{W'}([v/w']u) \).

Case \( u \neq w' \):

The set of terminal heaps in \( W \) and \( W' \) are the same since NormEq does not create or remove any terminals in \( W \). Thus, it suffices to show that for any terminal heap \( t \in W, t \not\nearrow W u \iff t \not\nearrow W' u \).

(⇒) This direction is trivial because every heap relation in \( W' \) still remains in \( W' \) except that two heaps \( w \) and \( w' \) in \( W \) are merged into a single heap \( w' \) in \( W' \).

(⇐) Suppose \( t \not\nearrow W' u \) holds. Then, there exist heaps \( u_i, v_i \in W' \) (i = 0, ··· , n) such that \( t = u_0, \{ u_1 \equiv u_0 \land v_0, \cdots, u_n \equiv u_{n-1} \land v_{n-1} \} \subset W', u_n = u \) for \( n \geq 0 \). Note that \( u_0(= t) \neq w \).

Let us assume that \( u_n(= u) \neq w \). (We can similarly prove for the situation when \( u_n = w \), so we omit it.) We have the following subcases.

(Subcase) \( u_i \neq w \) for all \( i = 1, \cdots, n - 1 \):

By construction of \( W' \), \( \{ u_1 \equiv u_0 \land v'_0, \cdots, u_n \equiv u_{n-1} \land v'_{n-1} \} \subset W \) holds for some \( v'_i \in \{ u_i, w' \} \) (i = 0, ··· , n - 1). Thus, we get \( t \not\nearrow W u \).

(Subcase) \( u_i = w \) for some \( i = 1, \cdots, n - 1 \):

Without loss of generality, assume that there exists a unique \( i \in \{ 1, \cdots, n - 1 \} \) such that \( u_i = w \). If \( \{ w \equiv u_{i-1} \land v'_{i-1}, u_{i+1} \equiv w \land v'_{i} \} \subset W \) or \( \{ w' \equiv u_{i-1} \land v'_{i-1}, u_{i+1} \equiv w \land v'_{i} \} \subset W \) holds for some \( v'_i \in \{ v_j, w' \} \) (j = i - 1, i), then we clearly get \( t \not\nearrow W u \). So, suppose \( \{ w \equiv u_{i-1} \land v'_{i-1}, u_{i+1} \equiv w \land v'_{i} \} \subset W \) holds for some \( v'_i \in \{ v_j, w' \} \) (j = i - 1, i). (We omit the symmetric case.) By Lemma B.2 with \( W \) consistent and \( w \equiv w_1 \land w_2 \in W \), \( t \not\nearrow W w \) implies either \( t \not\nearrow W w_1 \) or \( t \not\nearrow W w_2 \). Since \( w' \equiv w_1 \land w_2 \in W \) holds and we have \( w' \not\nearrow W u \) by assumption, we finally get \( t \not\nearrow W u \).

38
Case \( u = w' \):

By Lemma B.2 with \( W \) consistent, we have \( T_W(w') = T_W(w_1) \cup T_W(w_2) = T_W(w) \). By the above case, we get \( T_W(w') = T_W(w) \).

Now, let us prove the second claim by using the first claim. Suppose \( W \) is sanitized and \(*\)-ready (resp. \( \rightarrow\)-ready) for heap \( u \in W \). We prove the claim step by step as follows.

1. Elementary:

   Suppose \( v_1 \vdash v_2 \circ v_3 \in W' \) for some heaps \( v_1, v_2, v_3 \in W' \). By construction of \( W' \), we have \( v'_1 \vdash v'_2 \circ v'_3 \in W \) for some \( v'_i \) such that \( v'_i = v_i \) if \( v_i \neq w \), and \( v'_i \in \{ w, w' \} \) if \( v_i = w \) (\( i = 1, 2, 3 \)). By using the first claim and that \( W \) is elementary, \( T_{W'}(v'_2) \cap T_{W'}(v'_3) = T_{W'}(v'_2) \cap T_{W'}(v'_3) = \emptyset \).

2. Rooted:

   It is clear because: (1) \( W \) is rooted, (2) no fresh heaps are added to \( W \), and (3) every heap relation in \( W \) still remains in \( W' \) (except the substitution \( \{ w/w' \} \)).

3. Consistent:

   Suppose \( \{ v_1 \vdash v_2 \circ v_3, v_4 \vdash v_5 \circ v_6 \} \in W' \) for some heaps \( v_1, v_2, v_3, v_4, v_5, v_6 \in W' \) with \( v_1 = v_4 \). By construction of \( W' \), we have \( \{ v'_1 \vdash v'_2 \circ v'_3, v'_4 \vdash v'_5 \circ v'_6 \} \in W \) for some \( v'_i \) such that \( v'_i = v_i \) if \( v_i \neq w \) and \( v'_i \in \{ w, w' \} \) if \( v_i = w \) (\( i = 1, \ldots, 6 \)). By the first claim and Lemma B.2 with \( W \) consistent, \( T_{W'}(v'_i) \cup T_{W'}(v'_{i+2}) = T_{W'}(v'_i) \cup T_{W'}(v'_{i+2}) = T_{W'}(v'_i) \) (\( i = 1, 4 \)). Since we have \( T_W(w) = T_W(w') \) as above, \( T_{W'}(v'_2) \cup T_{W'}(v'_3) = T_{W'}(v'_5) \cup T_{W'}(v'_6) \).

4. Full:

   As mentioned above, the set of terminal heaps are the same in \( W \) and \( W' \). So, let \( S \) be any non-empty set of terminal heaps in \( W' \) (or equivalently, in \( W \)). Since \( W \) is full, there exists a heap \( v \in W \) with \( T_W(v) = S \). By the first claim, we have \( T_{W'}([w/w']v) = S \). Thus, \( W' \) is also full.

5. \(*\)-ready (resp. \( \rightarrow\)-ready) for heap \([w/w']u \in W'\):

   Let \( S_1 \) and \( S_2 \) be any non-empty sets of terminal heaps in \( W' \) such that \( S_1 \cap S_2 = \emptyset \) and \( S_1 \cup S_2 = T_{W'}([w/w']u) \) (\( = T_W(u) \) by the first claim). Since \( W \) is \(*\)-ready for \( u \), there exist heaps \( u_1, u_2 \in W \) such that \( u \vdash u_1 \circ u_2 \in W \) and \( T_W(u_1) = S_1, T_W(u_2) = S_2 \). By the first claim, we have \([w/w'](u \vdash u_1 \circ u_2) \in W' \) and \( T_{W'}([w/w']u_1) = S_1, T_{W'}([w/w']u_2) = S_2 \). Thus, \( W' \) is \(*\)-ready for \([w/w']u \). The proof for \( \rightarrow\)-ready is similar.

7. Sanitized:

   Since \( W \) is sanitized, atomic heap relations in \( W \) reside only for terminal heaps in \( W \). As mentioned above, the set of terminal heaps in \( W \) and \( W' \) are the same. Moreover, no new atomic heap relations are added to \( W \). Thus, \( W' \) is sanitized.

\[ \square \]

**Lemma B.7.** Let \( A, B \subseteq C \) be sets and \( x, y \subseteq C \).

- Suppose \( x \notin A \). Then, \( [c/x][x/c]A = A \) for any \( c \subseteq C \).
- Suppose \( x \notin A, y \notin B \). Then, \( [x/y]A = B \iff A = [y/x]B \iff \begin{cases} \alpha \in A \implies [x/y]\alpha \in B \\ \beta \in B \implies [y/x]\beta \in A \end{cases} \).
- Suppose \( x \notin A, x \notin B \). Then, \( [x/y]A = [x/y]B \iff A = B \).
- Suppose \( x \notin A, x \notin B \). Then, \( ([x/y]A) \cap ([x/y]B) = [x/y](A \cap B) \).
- \( ([x/y]A) \cup ([x/y]B) = [x/y](A \cup B) \).

**Proof.** The proof is straightforward. \[ \square \]

**Definition B.8.** For a world sequent \( W \) and a terminal heap \( \rho \) in \( W \), we define the following properties of \( W \).

- \( \rho \)-Elementary: non-cyclic and if \( w \vdash w_1 \circ w_2 \), then \( (T(w_1) - \{ \rho \}) \cap (T(w_2) - \{ \rho \}) = \emptyset \).
- \( \rho \)-Rooted: \( \rho \)-elementary and there is a root heap \( w \) such that \( v \not\succ w \) for every heap \( v \neq \rho \).
- $\rho$-Consistent: $\rho$-rooted and if $w \equiv u_1 \circ u_2$ and $w \equiv v_1 \circ v_2$, then $(T(u_1)-\{\rho\}) \cup (T(u_2)-\{\rho\}) = (T(v_1)-\{\rho\}) \cup (T(v_2)-\{\rho\})$.
- $\rho$-Full: $\rho$-consistent and for any non-empty set $S$ of terminal heaps except $\rho$, there exists at least one heap $w$ with $T(w)-\{\rho\} = S$.
- $\rho$- $\ast$-ready for heap $w$: $\rho$-full and for any pair of non-empty sets $S_1$ and $S_2$ of terminal heaps except $\rho$ such that $S_1 \cap S_2 = \varnothing$ and $S_1 \cup S_2 = T(w)-\{\rho\}$, there exist heaps $w_1$ and $w_2$ such that $w = w_1 \circ w_2$ with $T(w_1)-\{\rho\} = S_1$ and $T(w_2)-\{\rho\} = S_2$.
- $\rho$- $\ast\ast$-ready for heap $w$: $\rho$-full and for any pair of non-empty sets $S_1$ and $S_2$ of terminal heaps except $\rho$ such that $T(w) \cap S_1 = \varnothing$ and $T(w)-\{\rho\} \cup S_1 = S_2$, there exist heaps $w_1$ and $w_2$ such that $w_2 \equiv w \circ w_1$ with $T(w_1)-\{\rho\} = S_1$ and $T(w_2)-\{\rho\} = S_2$.
- $\rho$-Sanitized: $\rho$-full and non-terminal heaps have no atomic heap relations.

Lemma B.9 (Preservation lemma for the rule NormPC).

Let $W$ and $W'$ be any world sequents such that $\{W\} \Rightarrow \{W'\}$ by applying the rule NormPC to $\{w \equiv w_1 \circ w_2, \ w_2 \equiv \epsilon\}$, where $w_2$ is a terminal heap in $W$. Assume that for any $n \geq 2$, there do not exist heaps $v_0, \cdots, v_n$ in $W$ such that $w_1 = v_0, v_1 \equiv v_0 \circ w_2, \cdots, v_n \equiv v_{n-1} \circ w_2, v_n = w \cdots (*)$. Then, we have the followings.

- If $W$ is $w_2$-consistent, then for any heap $u \in W$ we have $[w/w_1](T_W(u) - \{w_2\}) = T_W([w/w_1]u) - \{w_2\}$.
- If $W$ is $w_2$-$\ast$-ready (resp. $w_2$-$\ast\ast$-ready) for heap $u \in W$ and $w_2$-sanitized, then $W'$ is $w_2$-$\ast$-ready (resp. $w_2$-$\ast\ast$-ready) for heap $[w/w_1]u \in W'$ and $w_2$-sanitized.
- Even if $W$ is elementary, $W'$ is not elementary in general.

Proof. An example for the third claim is applying NormPC to $\{w \equiv w_{123} \circ w_4, w \equiv w_1 \circ w_{234}, w_{123} \equiv w_1 \circ w_{23}, w_{234} \equiv w_2 \circ w_4, w_{234} \equiv w_2 \circ w_3, w_{234} \equiv w_3 \circ w_4, w_4 \equiv \epsilon\}$ and $\{w_{234} \equiv w_{23} \circ w_4, w_{234} \equiv w_2 \circ w_3, w_{234} \equiv w_3 \circ w_4, w_4 \equiv \epsilon\}$.

Let us prove the first claim. Suppose that $W$ is $w_2$-consistent, and choose any heap $u \in W$. We have the following for proving that $[w/w_1](T_W(u) - \{w_2\}) = T_W([w/w_1]u) - \{w_2\}$.

Case $u \neq w_1$:
Since $w \not\in T_W(u) - \{w_2\}$ and $w_1 \not\in T_W(u) - \{w_2\}$, by Lemma B.7 it suffices to show that:

(a) For any terminal heap $t \in W - \{w_2\}$, $[w/w_1]t$ is a terminal heap in $W'$.
Moreover, $t \not\nearrow W u \implies [w/w_1]t \not\nearrow W' u$.
(b) For any terminal heap $t \in W' - \{w_2\}$, $[w/w_1]t$ is a terminal heap in $W$.
Moreover, $t \not\nearrow W' u \implies [w/w_1]t \not\nearrow W u$.

The proof is as follows.

(a) Firstly, if $t \neq w_1$, then $t$ is a terminal heap in $W'$ since NormPC does not add any new heap relations to $W$. Let us show that $w$ is a terminal heap in $W'$. It suffices to show that $w = w_1 \circ w_2$ is the only heap relation in $W$ about $w$. Suppose there exists a heap relation $w = w_1' \circ w_2'$ in $W$ different from $w = w_1 \circ w_2$. By the modification of Lemma B.2 with $W$ $w_2$-consistent, $T_W(w) - \{w_2\} = T_W(w_1) - \{w_2\}$ so $T_W(w) = \{w_1, w_2\}$. We get contradiction for each of the following subcases, as follows.

(Subcase) $w_1' = w_1$:
Since $W$ is $w_2$-elementary, $T_W(w_2') \subseteq \{w_2\}$ and, by Lemma B.2 with $W$ non-cyclic, $T_W(w_2') = \{w_2\}$. Since there is no heap relation $\cdot \equiv w_2 \circ w_3$ in $W$, we must have $w_2' = w_2$. This contradicts to that $w = w_1' \circ w_2'$ and $w = w_1 \circ w_2$ are different.

(Subcase) $w_1' \neq w_1$ ($i = 1, 2$):
By the modification of Lemma B.2 with $W$ $w_2$-consistent and $w = w_1 \circ w_2$, we have $w_1 \not\nearrow W w_1'$ without loss of generality (since $w_1 \not\nearrow W w$ and $w_1 \in T_W(w) - \{w_2\}$). Since $w_1' \neq w_1'$, there exist $n \geq 1$ and heaps $\alpha_0, \cdots, \alpha_n$ and $\beta_0, \cdots, \beta_{n-1}$ in $W$ such that $w_1 = \alpha_0, \alpha_1 = \alpha_0 \circ \beta_0, \cdots, \alpha_n = \alpha_{n-1} \circ \beta_{n-1}, \alpha_n = w'$. Since $W$ is $w_2$-elementary, $\beta_i = w_2$ ($i = 0, \cdots, n - 1$). Moreover, since $W$ is $w_2$-elementary, $T_W(w_2') \subseteq \{w_2\}$ and thus $w_2' = w_2$ (by the same argument above). This contradicts to the assumption ($\ast$).
Secondly, if \( t \not\succcurlyeq W u \), then \([w/w_1]t \not\succcurlyeq W u \) because: (1) \( w \prec v_1 \circ v_2 \) is the only heap relation in \( W \) which is removed from \( W \), but (2) a path from \( t \) to \( u \) does not pass through \( v_2 \) (since \( t \not= w_1 \) and \( v_2 \) is a terminal heap in \( W \)) and (3) \( w_1 \) and \( w_1 \) are merged into \( w \) in \( W \).

(b) Firstly, if \( t \not= w \), then \( t \) is a terminal heap in \( W \) since NormPC does not remove any existing heap relations in \( W \) about \( \not= w_1 \); if \( t = w \), then there is no heap relation in \( W \) about \( w_1 \), so \( w_1 \) is a terminal heap in \( W \).

Secondly, suppose \( t \not\preccurlyeq W u \). Then, there exist heaps \( u_1, v_1 \in W' \) (\( i = 0, \cdots, n - 1 \)) such that \( t = u_0, \{ u_0 \circ v_0, \cdots, u_m \circ v_{m-1} \} \subset W', u_n = u \) for \( n \geq 0 \). Assume that \( u_0(=t) \not= w \) and \( u_n(=t) \not= w \). (With the fact that \( w_1 \prec W w \), we can similarly prove the situation when \( w_0 = w \) or \( u_n = w \), so we omit it.) We have the following subcases.

(Subcase) \( u_i \not= w \) for all \( i = 1, \cdots, n - 1 \):

By construction of \( W' \), \( \{ u_1 \circ v_0, \cdots, u_n \circ v_{n-1} \} \subset W \) holds for some \( v_i' \in \{ v_j, w_1 \} \) (\( i = 0, \cdots, n - 1 \)). Thus, we get \( t \not\preccurlyeq W u \).

(Subcase) \( u_i = w \) for some \( i = 1, \cdots, n - 1 \):

Without loss of generality, assume that there exists a unique \( i \in \{ 1, \cdots, n - 1 \} \) such that \( u_i = w \). If \([w \circ v_i', u_i+1 \subseteq w \circ v_i'] \subset W \), or \([w_1 \circ u_i \circ v_i', u_i+1 \subseteq w \circ v_i'] \subset W \), or \([w_1 \circ u_i \circ v_i', u_i+1 \subseteq w \circ v_i'] \subset W \) holds for some \( v_i' \in \{ v_j, w_1 \} \) (\( j = i - 1, i \)), then we clearly get \( t \not\preccurlyeq W u \). So, suppose \( \{ w \circ u_i, u_i+1 \subseteq w \circ v_i' \} \subset W \) holds for some \( v_i' \in \{ v_j, w_1 \} \) (\( j = i - 1, i \)). By the modification of Lemma B.2 with \( W \) \( \mu \), consistent and \( w_1 \circ u_i \circ w_2 \) with \( t \in T_W(w) - \{ w_2 \} \) implies either \( t \not\preccurlyeq W u_1 \) or \( t \not\preccurlyeq W w_2 \). Since \( t \not= w_2 \) and \( w_2 \) is a terminal heap in \( W \), we have \( t \not\preccurlyeq W w_2 \).

Case \( u = w_1 \):

By the modification of Lemma B.2 with \( W \) \( \mu \), consistent, we have \( T_W(w_1) - \{ w_2 \} = (T_W(w_1) - \{ w_2 \}) \cup (T_W(w_2) - \{ w_2 \}) = T_W(w) - \{ w_2 \} \). Also, by the first case above, we have \([w/w_1](T_W(w) - \{ w_2 \}) = T_W((w/w_1) - \{ w_2 \}) \). Thus, we get \([w/w_1](T_W(w_1) - \{ w_2 \}) = T_W((w/w_1) - \{ w_2 \}) \).

Now, let us prove the second claim by using the first claim. Suppose \( W \) is \( \mu \)-ready (resp. \( \mu \)-ready) for heap \( w \) in \( W \). We prove the claim step by step as follows.

1. \( \mu \)-Elementary:

Let us show that \( W' \) is non-cyclic by contradiction. Suppose that there exist heaps \( v, u \in W' \) such that \( v \neq \mu, v \not\succcurlyeq W u \) and \( u \not\preccurlyeq W v \). Then, there exist heaps \( u_i \), \( i = 0, \cdots, n + m - 1 \) such that \( \{ v_i \circ \beta_0, \cdots, \alpha_{n+1} \circ \beta_{n+m-1} \} \subset W' \), \( u_0 = v, u_n = \mu \). We get contradiction for the following cases, as follows.

Case \( \alpha_i \neq w \) for all \( i = 0, \cdots, n + m \):

As in the proof of the first claim’s (b), we get \( v \not\preccurlyeq W u \) and \( u \not\preccurlyeq W v \). However, since \( W \) is non-cyclic, it is contradiction.

Case \( \alpha_i = w \) for some \( i = 0, \cdots, n + m \):

Without loss of generality, assume that \( v = w \) and \( \alpha_i \neq w \) for all \( i = 1, \cdots, n + m - 1 \).

Suppose \( \{ \alpha_i \circ w \circ \beta_0, \cdots, \alpha_{n+1} \circ \beta_{n+m-1} \} \subset W \), or \( \{ \alpha_i \circ w_1 \circ \beta_0, \cdots, \alpha_{n+1} \circ \beta_{n+m-1} \} \subset W \), or \( \{ \alpha_i \circ w_1 \circ \beta_0, \cdots, \alpha_{n+1} \circ \beta_{n+m-1} \} \subset W \) holds for some \( \beta_i \in \{ \beta_0, w_1 \} \) (\( i = 0, n + m - 1 \)). Then, we clearly get either (1) \( v \not\preccurlyeq W u \) and \( u \not\preccurlyeq W v \), or (2) \( v \not\preccurlyeq W \mu \) and \( u \not\preccurlyeq W \mu \). Since \( u \neq \mu \) and \( \alpha_i \neq w \), this contradicts to the assumption (\( \star \)).

Now, suppose \( \{ \alpha_i \circ w_1 \circ \beta_0, \cdots, \alpha_{n+1} \circ \beta_{n+m-1} \} \subset W \) and \( \{ \alpha_i \circ \beta_1, \cdots, \alpha_{n+1} \circ \beta_{n+m-2} \} \subset W \) hold for some \( \beta_i \in \{ \beta_0, w_1 \} \) (\( i = 0, n + m - 1 \)). Since \( T_W(w_1) - \{ w_2 \} = T_W(w_1) - \{ w_2 \} \) (as mentioned above), \( T_W(\beta_i) = \{ w_2 \} \) (\( i = 0, \cdots, n + m - 1 \)) by Lemma B.2 with \( W \) non-cyclic. Since there is no heap relation in \( W \) of the form \( \circ w_2 \circ w_2 \), we must have \( \beta_i = w_2 \) (\( i = 0, \cdots, n + m - 1 \)). However, this contradicts to the assumption (\( \star \)).

Now, suppose \( v_1 \circ v_2 \circ v_3 \in W \) for some heaps \( v_1, v_2, v_3 \in W \) By construction of \( W' \), we have \( v_1' \circ v_2' \circ v_3' \in W \) for some \( v_i' \) such that \( v_i' = v_i \) if \( v_i \neq w \), and \( v_i' \in \{ w, w_1 \} \) if \( v_i = w \) (\( i = 1, 2, 3 \).
By the first claim and Lemma B.7 with \( w \notin T_W(v) \), \([T_W(v_2) - \{ w_2 \}] \cap [T_W(v_2) - \{ w_2 \}] = [(w/w_1)[T_W(v_2') - \{ w_2 \}]) \cap [w/w_1][T_W(v_2) - \{ w_2 \}] = [w/w_1][T_W(v_2') - \{ w_2 \}] \cap [T_W(v_2') - \{ w_2 \}]] \cap (T_W(v_2') - \{ w_2 \}]).

Since \( W \) is \( w_2 \)-elementary, \([T_W(v_2) - \{ w_2 \}] \cap (T_W(v_3) - \{ w_2 \}) = \emptyset \).

2. \( w_2 \)-Rooted:

It is clear because: (1) \( W \) is \( w_2 \)-rooted, (2) no fresh heaps are added to \( W \), (3) \( w = w_1 \circ w_2 \) is the only heap relation in \( W \) which is removed from \( W \), and (4) \( w \) and \( w_1 \) are merged into a single heap in \( W' \).

3. \( w_2 \)-Consistent:

Suppose \( \{ v_1 = v_2 \circ v_3, v_4 = v_5 \circ v_6 \} \subseteq W' \) for some heaps \( v_1, \ldots, v_6 \in W' \) with \( v_i = v_4 \). By construction of \( W' \), we have \( \{ v_1 = v_2 \circ v_3, v_4 = v_5 \circ v_6 \} \subseteq W' \) for some \( v_i' \) such that \( v_i' = v_i \) if \( v_i \neq w \) and \( v_i' \in \{ w, w_1 \} \) if \( v_i = w \) (i = 1, \ldots, 6).

By the first claim and Lemma B.7, \([T_W(v_4) - \{ w_2 \}] \cap [T_W(v_2) - \{ w_2 \}] = [(w/w_1)[T_W(v_2') - \{ w_2 \}] \cup [w/w_1][T_W(v_2') - \{ w_2 \}] \cap [w/w_1][T_W(v_2') - \{ w_2 \}] = [w/w_1][T_W(v_2') - \{ w_2 \}] \cup [T_W(v_2') - \{ w_2 \}] \cup [T_W(v_2') - \{ w_2 \})]. \)

By the modification of Lemma B.2 with \( W - w_2 \)-consistent, we have \( \{ w/w_1\} \cup [T_W(v_2) - \{ w_2 \}] \cup [T_W(v_2') - \{ w_2 \}] \cup [T_W(v_2') - \{ w_2 \})] = \{ w/w_1\} \cup [T_W(v_2') - \{ w_2 \}] \cup [T_W(v_2') - \{ w_2 \})] \cup [T_W(v_2') - \{ w_2 \}) (i = 1, 4). Since we have \( T_W(u_1) \cap \{ w_2 \} = T_W(w) - \{ w_2 \} \) as above, \([T_W(v_2) - \{ w_2 \}] \cup [T_W(v_2) - \{ w_2 \}] = (T_W(v_3) - \{ w_2 \}) \cup [T_W(v_6) - \{ w_2 \})].

4. \( w_2 \)-Full:

Let \( S \) be any non-empty set of terminal heaps in \( W' \) except \( w_2 \). By (b) in the first claim, \([w/w_1]S \) is a non-empty set of terminal heaps in \( W \) except \( w_2 \). Since \( W \) is \( w_2 \)-full, there exists a heap \( v \in W \) with \( T_W(v) - \{ w_2 \} = [w/w_1]S \). By the first claim and Lemma B.7 with \( w_1 \notin S \), we have \( T_W([w/w_1]v) = [w/w_1][w/w_1]S = S \). Thus, \( W' \) is also \( w_2 \)-full.

5. \( w_2 \)-s-ready (resp. \( w_2 \)-\( \star \)-ready) for heap \([w/w_1]u \in W' \):

Let \( S_1 \) and \( S_2 \) be any non-empty sets of terminal heaps in \( W' \) except \( w_2 \) such that \( S_1 \cap S_2 = \emptyset \) and \( S_1 \cup S_2 = T_W([w/w_1]u) - \{ w_2 \} \) (by the first claim). By Lemma B.7 with \( w \notin T_W(u) \), \([w/w_1]S_1 \cup [w/w_1]S_2 = [w/w_1](S_1 \cup S_2) = [w/w_1][T_W(u) - \{ w_2 \})] = T_W(u) - \{ w_2 \} \).

Since \( W \) is \( w_2 \)-\( \star \)-ready for \( u \), there exist heaps \( u_1, u_2 \in W \) such that \( u = u_1 \circ u_2 \in W \) and \( T_W(u_1) - \{ w_2 \} = [w/w_1]S_1, T_W(u_2) - \{ w_2 \} = [w/w_1]S_2 \). Since \( S_i \neq \emptyset \), \( u_i \neq w_2 \) (i = 1, 2); thus, \([w/w_1]u = u_1 \circ u_2 \in W' \). Also, by the first claim and Lemma B.7 with \( w_1 \notin S_i \), \([w/w_1]u \cap \{ w_2 \} = \{ w_2 \} \cap \{ w_2 \} = \emptyset \) (i = 1, 2). Thus, \( W' \) is \( w_2 \)-\( \star \)-ready for \([w/w_1]u \).

The proof for \( w_2 \)-\( \star \)-ready is similar.

7. \( w_2 \)-Sanitized:

Since \( W \) is \( w_2 \)-sanitized, atomic heap relations in \( W \) reside only for terminal heaps in \( W \). By (a) in the first claim, if \( t \in W \) is a terminal heap, then \([w/w_1]t \in W' \) is also a terminal heap. Moreover, no new atomic heap relations are added to \( W \). Thus, \( W' \) is \( w_2 \)-sanitized.

\[ \square \]
B.2.3 Completeness of the rules $\rightarrow R$, $\rightarrow L$, and Weaken

Before we prove Lemma 6.7, we introduce several auxiliary Lemmas about how to apply specific rules to obtain a world sequent having specific properties.

Lemma B.10. Let $W$ be any rooted world sequent, and $W^1$ be any world sequent with $\{W\} \xrightarrow{R} \{W^1\}$ by applying the rule $\rightarrow R$. Let $r$ be the root heap in $W$, and $W^1 = W \cup \{v_1^1 \leq v_1 \circ w_1^1\}$ for some heaps $v_1, w_1^1$, where $v_1 \in W$ and $w_1^1$ are fresh heaps w.r.t. $W$. Suppose $v_n \equiv u_{n-1} \circ v_{n-1} \in W$ for some heaps $v_1, \ldots, v_N, v_{N+1} := r, u_1, \ldots, u_N$ in $W$ $(n = 2, \ldots, N + 1)$. For each $n = 2, \ldots, N + 1$, define a world sequent $W_n$ from $W_n^{n-1}$ as:

- \(\{W_n^{n-1}\} \rightarrow \{W_n\}\) by applying \(\text{the rule Disj} \rightarrow\) on \(\{v_n \equiv u_{n-1} \circ v_{n-1}, w_n^{n-1} \equiv v_{n-1} \circ w_n^{n-1}\}\);  
- \(W_n = W_n^{n-1} \cup \{w_1^n = w_1^{n-1} \circ w_1^n, u_{n-1} = w_1^n \circ u_{n-1}^{n-1}, w_2^{n-1} = w_2^n \circ w_2^{n-1}\}\),

where $w_k^n$ is a fresh heap w.r.t. $W_n^{k-1} (k = 1, \ldots, 4)$. Then, $W_n^{N+1}$ is rooted.

Proof. First, let us prove that there is a unique root in $W_n^{N+1}$. It is easy to show that, for each $n = 1, \ldots, N, w \in W_n$ implies $\exists w \neq r$ or $w \neq w_k^n$ in $W$. (To prove it, just use induction and the rooted-ness of $W$.) From this claim, it is clear that $w_1^{N+1}$ (having $r$ and $w_1^N$ as its children) is a unique root in $W_n^{N+1}$.

Next, let us prove that $W_n^{N+1}$ is elementary. Since we cannot prove it directly (or simply by induction), we must prove stronger statement. Thus, let us prove that the followings hold for each $n = 1, \ldots, N + 1$:

(a) $u_i \neq w$ holds in $W_n$ for any heap $w \in W_n - W$ and any $n \leq i \leq N$;
(b) $w_2^n \neq w$ holds in $W_n$ for any heap $w \in W$;
(c) $W_n$ is elementary.

The proof proceeds by induction on $n$, as follows.

i) $n = 1$:

(a) Note that $W_1 - W = \{w_1^1, w_2^1\}$. 

Since $w_2^1$ is a terminal heap in $W_1$, $u_i \neq w_1^1$ holds in $W_1$ for any $1 \leq i \leq N$. Suppose $u_i \neq w_1^1$ holds in $W_1$ for some $1 \leq i \leq N$. Then, $u_i \neq v_1$ holds in $W$, so $u_i \circ u_i$ holds in $W$. However, this contradicts to $W$ elementary so the claim (a) is true.

(b) The claim (b) is trivial since $w_1^1$ is the only ancestor of $w_2^1$ in $W_1$.

(c) It is easy to check the claim (c) by using the fact that $W$ is elementary.

ii) $n > 1$:

(a) Suppose $w \in W_n^{n-1} - W$. By IH (a), for any $n \leq i \leq N$, $u_i \neq w$ holds in $W_n^{n-1}$ and thus in $W_n$ (because $w \in W_n^{n-1}$).

Now, suppose $w \in W_n^{n} - W_n^{n-1} = \{w_k^n : k = 1, \ldots, 4\}$. For $k = 2, 3, 4$, since $w_k^n$ is a terminal heap in $W_n$, we have $u_i \neq w_k^n$ in $W_n$ for any $n \leq i \leq N$. Suppose $u_i \neq w_k^n$ holds in $W_n$ for some $n \leq i \leq N$. Then, we must have either $u_i \neq v_n$ or $u_i \neq w_k^{n-1}$ in $W_n$. However, the former is false (otherwise, $u_i \neq v_n \in W_n$ because $i \geq n$), and the latter is also false (as proved above). Thus, the claim (a) is proved.

(b) Note that $w_2^{n-1}$ and $w_3^n$ are the only parent heaps of $w_2^n$. Let $w \in W$ be any heap. By IH (b), $w_2^{n-1} \neq w$ holds in $W_n^{n-1}$ and thus in $W_n$ (because $w \in W_n^{n-1}$). Also, $w_3^n$ has no ancestors in $W_n$. Thus, $w_2^n \neq w$ holds in $W_n$, and the claim (b) is true.

(c) Let us prove the claim (c) by contradiction. Suppose that $W_n$ is not elementary. Then, there exist heaps $t, w \in W_n$ such that $t$ is a terminal heap in $W_n$ and $t \neq w \in W_n$. Suppose $t \in W_n^{n-1}$. Since $W_n^{n-1}$ is elementary by IH (c), $w = w_k^n$ holds in $W_n$. (Otherwise, $t \neq w \in W_n^{n-1}$, which is a contradiction.) Then, we must have $t = w_2^n$ or $w_3^n$, which implies $t \notin W_n^{n-1}$ and it is a contradiction.

Now, we know that $t = w_k^n$ for some $k = 2, 3, 4$. We have the following cases for $w$.  


Case $w \in W$:
Suppose $k = 2$. By (b), $t \not\Rightarrow w$ holds in $W^n$, which is a contradiction.
Suppose $k = 3$. Since $w^n$ has no ancestors in $W^n$, we have $u_{n-1} \wedge u_{n-1}$ in $W^n$ and thus in $W^{n-1}$ (because $w \in W^{n-1}$). This contradicts to that $W^{n-1}$ is elementary.
Suppose $k = 4$. By IH (b), $w_{n-1} \not\Rightarrow w$ holds in $W^{n-1}$ and thus in $W^n$. So, $u_{n-1} \wedge u_{n-1}$ holds in $W^n$. Similarly to $k = 3$, we also get a contradiction.

Case $w \in W^{n-1} - W$:
Suppose $k = 2$. Since $w^n$ has no ancestors in $W^n$, we have $w_{2-1} \wedge w_{2-1}$ in $W^n$ and thus in $W^{n-1}$. This contradicts to that $W^{n-1}$ is elementary.
Suppose $k = 3$. Similarly to $k = 2$, $u_{n-1} \wedge u_{n-1}$ holds in $W^n$ and we get a contradiction.
Suppose $k = 4$. By IH (a), $u_{n-1} \not\Rightarrow w$ holds in $W^{n-1}$ and thus in $W^n$. So, we have $w_{2-1} \wedge w_{2-1}$ in $W^n$, and get a contradiction as above for $k = 2$.

Case $w \in W^n - W^{n-1}$:
In this case, We have $w = w^n$. Note that $\{w \vdash w^n \circ w^n, w \vdash w^n \circ v_n\}$ are the only heap relations about $w$ in $W^n$.
Suppose that $t \not\Rightarrow w^n$ and $t \not\Rightarrow w^n$. Then, $t = w^n$. By IH (a), $u_{n-1} \not\Rightarrow w^n$ holds in $W^{n-1}$ and thus in $W^n$. So, $t \not\Rightarrow w^n$ holds in $W^n$, which is a contradiction.
Suppose that $t \not\Rightarrow w^n$ and $t \not\Rightarrow v_n$ holds in $W^n$. Then, $t = w^n$. By IH (b), $w_{n-1} \not\Rightarrow v_n$ holds in $W^n$ and thus in $W^n$. So, $t \not\Rightarrow v_n$ holds in $W^n$, which is a contradiction.

Since we get contradictions for all cases, we conclude that the claim (c) is true.

This completes the proof for this Lemma.

\begin{lemma}
Let $W$ be any rooted world sequent with a heap $w$ such that:

- For some $n \geq 1$, $w \equiv u_{1,1} \circ u_{1,2} \circ \cdots, w \equiv u_{n,1} \circ u_{n,2}, w \equiv v_1 \circ v_2$ are all the heap relations in $W$ at $w$.
- $v_1, v_2$ are terminal heaps in $W$ and $w$ is the only parent heap of $v_1$ or $v_2$ in $W$.
- Let $T_w(u_1) \cup T_w(u_2) = \{t_1, \cdots, t_m\}$ where $u_1 = u_{1,1}$ and $u_2 = u_{1,2}$.
  Then, $\{t_1, \cdots, t_m\} = T_w(u_1) \cup T_w(u_2)$ and $\{t_1, \cdots, t_m\} \cap \{v_1, v_2\} = \emptyset$ ($i = 1, \cdots, n$).
- For any descendant heap $\alpha$ (in $W$) of $u_{i,j}$ or $v_{i,j}$, $W$ is consistent at $\alpha$ ($i = 1, \cdots, n$ and $j = 1, 2$).

Then, there exists a rooted world sequent $W'$ such that:

(a) $\{W\} \Rightarrow^* \{W'\}$ by applying only \boxed{the rule Disj*};
(b) 1) For any heap relation $\alpha' \equiv \cdots \in W' - W$, $\alpha' \not\Rightarrow u_j$ or $\alpha' \not\Rightarrow v_j$ holds in $W'$ for some $1 \leq j \leq 2$;
   2) Moreover, for any terminal heap $t' \in W' - W$ in $W'$, $t' \not\Rightarrow t_k$ holds in $W'$ for some $1 \leq k \leq m$;
(c) $v_1, v_2$ are not terminal heaps in $W'$;
(d) $T_{W'}(u_1) \cup T_{W'}(u_2) = \bigcup_{1 \leq k \leq m} T_{W'}(t_k)$ and $T_{W'}(u_1) \cup T_{W'}(u_2) = T_{W'}(u_1) \cup T_{W'}(u_2) = T_{W'}(v_1) \cup T_{W'}(v_2)$ ($i = 1, \cdots, n$);
(e) For any descendant heap (in $W'$) $\alpha'$ of $u_{i,j}$ or $v_{i,j}$, $W'$ is consistent at $\alpha'$ ($i = 1, \cdots, n$ and $j = 1, 2$).

\end{lemma}

\begin{proof}
The proof proceeds by induction on $n$. Note that $n \geq 2$ by Lemma B.2 with $W$ elementary.

i) $m = 2$:
Let $W'$ be a world sequent obtained by applying $\text{Disj^*}$ to $W$ with $w \equiv u_1 \circ u_2, w \equiv v_1 \circ v_2$. Then, $W' = W' \cup \{u_1 \equiv \alpha_1 \circ \alpha_2, u_2 \equiv \alpha_3 \circ \alpha_4, v_1 \equiv \alpha_5 \circ \alpha_6, v_2 \equiv \alpha_7 \circ \alpha_8\}$ for fresh heaps $\alpha_1, \cdots, \alpha_8$.

Let us show that $W'$ is rooted. By Lemma B.3, it suffices to show that $W'$ is elementary. Suppose not. Then, $\alpha' \wedge W' \alpha'$ for some $\alpha' \in W'$. (From now on, we will omit the use of the Lemma.) Without loss of generality, we can assume $\alpha' = \alpha_1'$.
Since $W'$ is elementary and $u_1, v_1$ are the only parent heaps of $\alpha_1'$ in $W'$, we must have $u_1 \wedge W' v_1$.
By the second assumption, $u_1 \wedge W' v_1$ and this implies $w \wedge W' v_1$ and thus $w \wedge W' t_k$. Since this is a contradiction, we conclude that $W'$ is elementary.

\end{proof}
Now, let us show that $W'$ satisfies (b)-(e). Firstly, (b) and (c) are clearly true. Secondly, since $W$ is elementary and $\{t_1, t_2\} = T_W(u_{i,1}) \cup T_W(u_{i,2})$, $\{u_{i,1}, u_{i,2}\} = \{t_1, t_2\}$ $(i = 1, \ldots, n)$ by Lemma B.2. From this and that $u_1, u_2, v_1, v_2$ are terminal heaps in $W$ (by the second assumption), $T_W(u_1) \cup T_W(u_2) = \{\alpha_1, \ldots, \alpha_4\} = T_W(v_1) \cup T_W(v_2)$; thus, (d) is true. Finally, since $u_1, u_2, v_1, v_2$ are all distinct heaps (by the third assumption), there is a unique heap relation in $W'$ at each of these heaps. From this, (e) is clearly true.

ii) $m > 2$:

Let $W'$ be a world sequent obtained by applying Disj* to $W$ with $\{w \equiv u_1 \circ u_2, w \equiv v_1 \circ v_2\}$. Then, $W' = W \cup \{u_1 \equiv \alpha_1 \circ \alpha_2, u_2 \equiv \alpha_3 \circ \alpha_4, v_1 \equiv \alpha_1 \circ \alpha_3, v_2 \equiv \alpha_2 \circ \alpha_4\}$ for fresh heaps $\alpha_1, \ldots, \alpha_4$. By the same argument in the above, we know that $W'$ is rooted.

Suppose that $u_1$ and $u_2$ are not terminal heaps in $W$. (The other cases are similarly proved, so we omit it.) Then, $\{u_1 \equiv \beta_1 \circ \beta_2, u_2 \equiv \beta_3 \circ \beta_4\} \subset W$ for some heaps $\beta_1, \ldots, \beta_4$ in $W$. Without loss of generality, we can assume that $T_W(\beta_1) \cup T_W(\beta_2) = \{t_1, \ldots, t_i\}$ and $T_W(\beta_3) \cup T_W(\beta_4) = \{t_{i+1}, \ldots, t_m\}$ for some $0 < l < m$. Since $\alpha_1 \neq \beta_j$ in $W$ $(i, j = 1, \ldots, 4)$, $T_W(\beta_1) \cup T_W(\beta_2) = T_W(\beta_3) \cup T_W(\beta_4)$.

First, by the assumptions of $W$ and by the construction of $W'$, $W'$ with $u_1$ satisfies the assumptions of this Lemma. Moreover, since $l < m$, we can apply induction. By IH on $W'$ with $u_1$, there exists a rooted world sequent $W^3$ such that:

(a2) $\{W'\} \rightarrow^* \{W^3\}$ by applying only the rule Disj*;
(b2) 1) For any heap relation $\alpha^2 \equiv \cdots \equiv \in W^2 - W', \alpha^2 \not\in u_1$ in $W$;
2) Moreover, for any terminal heap $t^2 \in W^2 - W'$ in $W^2$, $t^2 \not\in t_k$ in $W^2$ for some $0 < k \leq l$;
(c2) $\alpha_1, \alpha_2$ are not terminal heaps in $W^2$;
(d2) $T_{W^3}(u_1) = \bigcup_{0 < k \leq l} T_{W^3}(t_k);$  
(e2) For any descendant heap (in $W^2$) $\alpha^2$ of $u_1$, $W^2$ is consistent at $\alpha^2$.

Next, by the assumptions of $W$ and the construction of $W'$, and by (b2)-1 that with a descendant heap of $u_1$ are not a descendant of $u_2$ (since $W^2$ is elementary), $W^2$ with $u_2$ satisfies the assumptions of this Lemma; also, we have $T_{W^2}(\beta_1) \cup T_{W^2}(\beta_2) = T_{W^2}(\beta_3) \cup T_{W^2}(\beta_4)$. Moreover, since $m - l < m$, we can apply induction. By IH on $W^2$ with $u_2$, there exists a rooted world sequent $W^3$ such that:

(a3) $\{W^2\} \rightarrow^* \{W^3\}$ by applying only the rule Disj*;
(b3) 1) For any heap relation $\alpha^3 \equiv \cdots \equiv \in W^3 - W^2, \alpha^3 \not\in u_2$ in $W$;
2) Moreover, for any terminal heap $t^3 \in W^3 - W^2$ in $W^3$, $t^3 \not\in t_k$ in $W^3$ for some $0 < k \leq m$;
(c3) $\alpha_3, \alpha_4$ are not terminal heaps in $W^3$;
(d3) $T_{W^3}(u_2) = \bigcup_{0 < k \leq m} T_{W^3}(t_k);$  
(e3) For any descendant heap (in $W^3$) $\alpha^3$ of $u_2$, $W^3$ is consistent at $\alpha^3$.

Now, let $W' := W^3$ and let us show that $W'$ satisfies (b)-(e).

(b) (b-1) is true by (b2)-1, (b3)-1, and the construction of $W^3$. (b-2) is true by (b2)-2, (b3)-2, and the fact that $\alpha^3_j$ is not a terminal heap in $W^3$ (due to (c2) and (c3)) $(j = 1, \ldots, 4)$.

(c) This is trivial from the construction of $W^3$.

(d) Firstly, by (b3)-1 with that a descendant heap of $u_2$ are not a descendant of $u_1$ (since $W^3$ is elementary), we have $T_{W^3}(u_1) = T_{W^3}(u_1)$ and $T_{W^3}(t_k) = T_{W^3}(t_k)$ $(k = 1, \ldots, l)$, and thus $T_{W^3}(u_1) = \bigcup_{0 < k \leq l} T_{W^3}(t_k)$ due to (d2). From this and (d3), we get $T_{W^3}(u_1) \cup T_{W^3}(u_2) = \bigcup_{0 < k \leq m} T_{W^3}(t_k)$. Secondly, by (c2) with (b3)-1 and by (e3), $W'$ is consistent at $u_1$ and $u_2$. Thus, we have $T_{W^3}(u_1) \cup T_{W^3}(u_2) = \bigcup_{0 < k \leq l} T_{W^3}(\alpha_1) = T_{W^3}(u_1) \cup T_{W^3}(u_2)$. Finally, we have $T_{W^3}(u_{i,1}) \cup T_{W^3}(u_{i,2}) = \bigcup_{0 < k \leq m} T_{W^3}(t_k)$ $(i = 1, \ldots, n)$. (c) direction is trivial and (c) direction is true by (b-2) and Lemma B.3 with $W'$ elementary.) Therefore, $W'$ satisfies (d).

(e) By (e2) with (b3)-1 and by (e3) and (b1), $W'$ satisfies (e).

This completes the proof for this Lemma.
Lemma B.12. Let \( W \) be any world sequent with heaps \( u, v \) satisfying one of the followings: 1) \( u \land v \); 2) \( u \not\rightarrow v \). Then, there exists a world sequent \( W' \) with a heap \( w' \) such that:

- \( \{ W \} \rightarrow^* \{ W' \} \) by applying only \( \text{the rule Assoc; } \)
- The corresponding one of the followings holds:
  1) A heap relation \( w' \triangleq u \circ v \) is in \( W' \).
  2) A heap relation \( v \triangleq u \circ w' \) is in \( W' \).

Proof. The proof is straightforward.

Lemma B.13. Let \( W \) be any consistent world sequent and \( w \in W \) be any heap. Suppose \( T_W(w) = \{ w_1, \ldots, w_n \} \) for some \( n \geq 2 \). Then, there exists a world sequent \( W' \) with a heap \( w' \) such that:

- \( \{ W \} \rightarrow^* \{ W' \} \) by applying only \( \text{the rule Assoc; } \)
- A heap relation \( w \triangleq w_1 \circ w' \) is in \( W' \);
- For any heap \( u' \in W' - W, |T_W(u')| < n \) holds.

Proof. The proof is straightforward.

Lemma B.14. Let \( W \) be a world sequent which is sanitized and \( * \)-ready (resp. \( \rightarrow * \)-ready) for a given heap \( u \in W \). (We will omit the phrase “(resp. \( \rightarrow * \)-ready)” from now on for simplicity.) Let \( w_1 \) be the special empty heap in \( W \). Then, there exists a world sequent \( W' \) such that:

(a) \( \{ W \} \rightarrow^* \{ W' \} \) by applying only \( \text{the rules NormPC and NormEmpty; } \)
(b) \( W' \) is \( * \)-ready for \( u \) and sanitized;
(c) There is no empty heap except \( w_1 \) in \( W' \).

Proof. The proof proceeds by induction on \( n := (\text{the number of empty heaps in } W \text{ except } w_1) \).

i) \( n = 0 \): There is nothing to prove.

ii) \( n > 0 \):
   - Let \( \rho \neq w_1 \) be an empty heap in \( W \). Since \( W \) is sanitized, \( \rho \) is a terminal heap in \( W \). Note that \( W \) is \( \rho \)-\( * \)-ready for \( u \) and \( \rho \)-sanitized since \( W \) is sanitized and \( * \)-ready for \( u \).

   First, by exploiting only a weaker fact that \( W \) is \( \rho \)-\( * \)-ready for \( u \) and \( \rho \)-sanitized (not by using a stronger fact that \( W \) is sanitized and \( * \)-ready for \( u \)), let us show that there exists a world sequent \( W^2 \) such that:

   a. \( \{ W \} \rightarrow^* \{ W^2 \} \) by applying only the rule NormPC;
   b. \( W^2 \) is \( \rho \)-\( * \)-ready for \( u \) and \( \rho \)-sanitized;
   c. There is no heap relation of the form \( \cdot \triangleq \cdot \circ \rho \) in \( W^2 \).

The proof proceeds by induction on \( m := (\text{the number of heap relations in } W \text{ of the form } \cdot \triangleq \cdot \circ \rho) \).

i. \( m = 0 \): There is nothing to prove.

ii. \( m > 0 \):

   By using the fact that \( W \) is non-cyclic (since \( W \) is \( \rho \)-elementary), it is easy to show that there exists heaps \( w, w_1 \in W \) with \( w \triangleq w_1 \circ \rho \in W \) such that: for any \( i \geq 2 \), there do not exist heaps \( v_0, v_1, \ldots, v_n \in W \) with \( w_1 = v_0, \{ v_i = v_0 \circ \rho, \ldots, v_i = v_{i-1} \circ \rho \} \subset W, v_i = w_2 \).

   First, let \( W^1 \) be a world sequent obtained by applying NormPC to \( W \) with \( \{ w \triangleq w_1 \circ \rho, w_2 \triangleq \epsilon \} \). Then, by Lemma B.9, \( W^1 \) is \( \rho \)-\( * \)-ready for \( u \) and \( \rho \)-sanitized. Moreover, the number of heap relations in \( W^1 \) of the form \( \cdot \triangleq \cdot \circ \rho \) is \( m - 1 \).

   Therefore, by IH on \( W^1 \), there exists a world sequent \( W^2 \) such that:
   - \( \{ W^1 \} \rightarrow^* \{ W^2 \} \) by applying only the rule NormPC;
   - \( W^2 \) satisfies b. and c.
Thus, we now have a world sequent $W^2$ satisfying $a$, $b$, and $c$.

Next, let $W^3$ be a world sequent obtained by applying $\text{NormEmpty}$ to $W^2$ with $\rho$ and $w_\epsilon$. Since $W^2$ satisfies $a$ and $b$, and $\rho$ is a terminal heap in $W^2$, $W^3$ is $\ast$-ready for $u$ and sanitized. Moreover, the number of empty heaps (except $w_\epsilon$) in $W^3$ is $n - 1$.

Therefore, by IH on $W^3$, there exists a world sequent $W'$ such that:

$\Rightarrow \in \{W^3\} \Rightarrow^{*} \{W'\}$ by applying only the rules $\text{NormPC}$ and $\text{NormEmpty}$;

$W'$ satisfies (b) and (c).

This completes the proof for this Lemma. $\blacksquare$

**Lemma B.15.** Let $W$ be a world sequent which is sanitized, $\ast$-ready (resp. $\rightarrow$-ready) for a given heap $u \in W$, and has no empty heap except its special empty heap $w_\epsilon$. (We will omit the phrase "(resp. $\rightarrow$-ready)" from now on for simplicity.) Then, for any $n \geq 1$, there exists a world sequent $W'$ such that:

(a) $\{W\} \Rightarrow^{*} \{W'\}$ by applying only the rules $\text{NormEq}$ and $\text{Assoc}$;

(b) $W'$ is $\ast$-ready for $u$ and sanitized;

(c) There is no empty heap except $w_\epsilon$ in $W'$;

(d) For any $m \leq n$ and any non-empty set $S = \{w_1, \ldots, w_m\}$ of terminal heaps in $W'$, there exists a unique heap $w' \in W'$ with $T_{W'}(w') = S$;

(e) For any heap $w' \in W' - W$, $|T_{W'}(w')| < n$.

**Proof.** By Lemma B.6 and B.4 and by the fact that $\text{NormEq}$ and $\text{Assoc}$ do not create heap relations $\cdot \cong \epsilon$, we do not need to prove (b) and (c), i.e., we only need to prove (d) and (e). Also, by these Lemmas, note that $\text{NormEq}$ and $\text{Assoc}$ do not change the set of terminal descendants of each heap. The proof proceeds by induction on $k$.

i) $k = 1$:

Let $\alpha \in W$ be any heap with $T_W(\alpha) = \{w_1\}$. By Lemma B.2 with $W$ elementary, $\alpha$ is a terminal heap and $\alpha = w_1$. Therefore, $W' = W$ satisfies (d) and (e).

ii) $k > 1$:

Let $\alpha, \beta \in W$ be any heaps with $\alpha \neq \beta$ and $T_W(\alpha) = T_W(\beta) = \{w_1, \ldots, w_n\}$.

First, by Lemma B.13, there exists a world sequent $W^1$ with heaps $\alpha_1$ and $\beta_1$ such that:

$\Rightarrow \in \{W\} \Rightarrow^{*} \{W^1\}$ by applying only the rule $\text{Assoc}$;

$\Rightarrow \{\alpha \cong w_1 \circ \alpha_1, \beta \cong w_1 \circ \beta_1\} \subseteq W^1$;

For any heap $w^1 \in W^1$, $|T_{W^1}(w^1)| < n$.

By Lemma B.2 with $W$ consistent, $T_{W^1}(\alpha_1) = T_{W^1}(\beta_1) = \{w_2, \ldots, w_n\}$.

Next, by IH on $W^1$ with $n - 1$, there exists a world sequent $W^2$ such that:

$\Rightarrow \in \{W^1\} \Rightarrow^{*} \{W^2\}$ by applying only the rules $\text{NormEq}$ and $\text{Assoc}$;

For any $m \leq n - 1$ and any non-empty set $S = \{w_1^1, \ldots, w_m^1\}$ of terminal heaps in $W^1$, there exists a unique heap $w^2 \in W^2$ with $T_{W^2}(w^2) = S$;

For any heap $w^2 \in W^2$, $|T_{W^2}(w^2)| < n - 1$.

Thus, we have $\{\alpha \cong w_1 \circ \gamma^2, \beta \cong w_1 \circ \gamma^2\} \subseteq W^2$ for some heap $\gamma^2 \in W^2$.

Finally, we get a world sequent $W^3$ by applying $\text{NormEq}$ with $\{\alpha \cong w_1 \circ \gamma^2, \beta \cong w_1 \circ \gamma^2\}$. So, in $W^3$, $\alpha$ and $\beta$ are merged into a single heap.

Since we did not create any fresh heap w.r.t. $W$ with $|T(\gamma)| = n$, we get a world sequent $W'$ satisfying (d) and (e) by repeating the above procedure.

This completes the proof for this Lemma. $\blacksquare$
From now on, we prove the remaining Lemmas and Propositions.

Lemma 6.7. For a world sequent \( W \) of a particular kind, there exists a world sequent \( W' \) of another kind such that:

1) \( \{ W \} \rightarrow^* \{ W' \} \) by applying only the structural rules;
2) \( \llbracket \{ W' \} \rrbracket_S \implies \llbracket \{ W \} \rrbracket_S \) for any stack \( S \),

where one of the following holds:

1. \( W \) is expanded and \( W' \) is rooted;
2. \( W \) is rooted (generated in step 1) and \( W' \) is consistent;
3. \( W \) is consistent and \( W' \) is full;
4. \( W \) is full and \( W' \) is \( \ast \)-ready for a given heap \( w \);
4'. \( W \) is full and \( W' \) is \( \rightarrow \ast \)-ready for a given heap \( w \);
8. \( W \) is sanitized and \( \ast \)-ready (\( \rightarrow \ast \)-ready) for a given heap \( w \), and
   \( W' \) is normalized and \( \ast \)-ready (\( \rightarrow \ast \)-ready) for heap \( w \).

Proof. In the proof, we will use only the structural rules \( \text{Disj} \rightarrow \ast \), \( \text{Disj} \ast \), \( \text{Assoc} \), \( \text{NormPC} \), \( \text{NormEq} \), \( \text{NormEmpty} \). Since 2) naturally holds by Corollary 6.6, we only need to prove 1) for each case.

For the cases 1 and 2, it suffices to show the following: if \( W \) be any consistent world sequent and \( W' \) be any world sequent with \( \{ W \} \rightarrow \{ W' \} \) by applying \( R = \ast L \) or \( \rightarrow R \), then there exists a rooted world sequent \( W^{N+1} \) and a consistent world sequent \( W' \) such that \( \{ W' \} \rightarrow^* \{ W^{N+1} \} \rightarrow^* \{ W \} \) by applying only the rules \( \text{Disj} \rightarrow \ast \) and \( \text{Disj} \ast \).

The main idea of the proof for the cases 1 and 2 (i.e. how to apply the rules \( \text{Disj} \rightarrow \ast \) and \( \text{Disj} \ast \) to get a consistent world sequent from an expanded world sequent) is shown in Figure 5 and 6. The former (resp. the latter) figure shows an example of applying the rule \( \text{Disj} \rightarrow \ast \) (resp. \( \text{Disj} \ast \)) to a given root node (resp. consistent) world sequent from an expanded world sequent. Note that the way to apply the rule \( \text{Disj} \ast \) to get a consistent world sequent is not obvious at all. As shown in Figure 7, it would be impossible to obtain a consistent world sequent if we mistakenly apply the rule \( \text{Disj} \ast \) even once — in Figure 7, if we apply the rule \( \text{Disj} \ast \) to \( \{ w \} \rightarrow \{ w^2 \} \) in the third world sequent only because \( \text{T} \{ w \} \cup \text{T} \{ w^2 \} \neq \text{T} \{ w^5 \} \cup \text{T} \{ w_6 \} \), then we cannot obtain a consistent world sequent no matter how we apply the rule \( \text{Disj} \ast \) (since the situation of \( w \) in \( \{ w \} \rightarrow \{ w^5 \} \rangle \) is fourth world sequent exactly happens to \( u_3 \) \( (j = 1, \cdots, 12) \) and to their descendant heaps in the following world sequents.)

1, 2. Suppose \( R = \ast L \). Then, \( W^1 = W \cup \{ w \downarrow \circ \} \) for some heap \( w \in W \). Since \( W^1 \) is rooted and \( W^1 \) with \( w \) satisfies the assumptions of Lemma B.11, \( W^{N+1} \) := \( W^1 \) and there exists a world sequent \( W' \) by the Lemma such that:

\( \triangleright \{ W^1 \} \rightarrow^* \{ W' \} \) by applying only the rule \( \text{Disj} \ast \);
\( \triangleright \) New heap relations in \( W' \) w.r.t. \( W^1 \) are at descendants of \( w \);
\( \triangleright \) \( W' \) is consistent at any descendant heap of \( w \).

From these and \( W \) consistent, we conclude that \( W' \) is consistent.

Now, suppose \( R = \rightarrow R \). Then, \( W^1 = W \cup \{ w_1^1 \downarrow v_1 \circ w_2^1 \} \) for some heaps \( v_1, w_1^1, w_2^1 \), where \( v_1 \in W \) and \( w_1^1, w_2^1 \in W^1 \). Since \( W \) is rooted, there exists a root heap \( r \) in \( W \). Since \( v_1 \nrightarrow r \) in \( W \), there exist heaps \( v_2, \cdots, v_N \) and \( u_1, \cdots, u_N \) such that \( v_n \downarrow u_{n-1} \circ v_{n-1} \in W (n = 2, \cdots, N + 1) \), where \( u_{N+1} := r \). For each \( n = 2, \cdots, N + 1 \), we inductively construct a world sequent \( W^n \) from \( W^{n-1} \) such that:

\( \triangleright \{ W^{n-1} \} \rightarrow \{ W^n \} \) by applying the rule \( \text{Disj} \rightarrow \ast \) on \( \{ v_n \downarrow u_{n-1} \circ v_{n-1}, s_{n-1} \downarrow u_n \circ w_{n-1}^n \downarrow s_{n-1} \circ w_{n-1}^n \} \);
\( \triangleright W^n = W^{n-1} \cup \{ w_1^n \downarrow w_2^n \circ v_n, u_{n-1}^n \downarrow w_3^n \circ w_{n-1}^n, u_{n-1}^n \downarrow w_2^n \circ w_{n-1}^n \} \).
Figure 5: An example of applying the rule $\text{Disj} \rightarrow \star$ to get a rooted world sequent. (The blue means newly created heap relations and the dashed is used just for easy visualization.)
Figure 6: An example of applying the rule $\text{Disj}^*$ to get a consistent world sequent. (The blue means newly created heap relations and the dashed is used just for easy visualization.)
Figure 7: An example of mistakenly applying the rule Disj*. 
(The blue means newly created heap relations and the dashed is used just for easy visualization.)
where $w^n_k$ is a fresh heap w.r.t. $W^{n-1}$ ($k = 1, \cdots, 4$). Then, by Lemma B.10, $W^{n+1}$ is rooted.

By $W$ consistent and the construction of $W^{N+1}$, it is easy to check: if $W^{N+1}$ is not consistent at $w^{N+1}_n$, then $w^{N+1}_n \in \{u_1, \cdots, u_N, w^3_1, \cdots, w^3_{N+1}\}$; moreover, if $W^{N+1}$ is not consistent at $u_n$ for some $1 \leq n \leq N$, then $W^{N+1}$ with $u_n$ satisfies the assumptions of Lemma B.11. Thus, by applying Lemma B.11 repeatedly to $u_n$’s, we get a world sequent $W'$ such that:

- $\{W^{N+1}\} \Rightarrow^* \{W'\}$ by applying only the rule Disj$*$;
- New heap relations in $W'$ w.r.t. $W^{N+1}$ are at descendants of $u_n$ for some $1 \leq n \leq N$;
- $W'$ is consistent at any descendant heap of $u_n$ ($n = 1, \cdots, N$).

Note that by $W^{N+1}$ elementary and by (b-1) of the Lemma, we can repeatedly apply the Lemma more than once to get $W'$ as above. Since new heap relations in $W'$ w.r.t. $W^{N+1}$ are at descendants of $u_n$ ($1 \leq n \leq N$), $w^3_1, \cdots, w^3_{N+1}$ are the only possible heaps where $W'$ could be not consistent.

Let us prove that $W'$ is, in fact, consistent even at $w^n_k$ ($k = 2, \cdots, N$). It suffices to show that if $W'$ is consistent at $w^{n-1}_n$, then $W'$ is consistent at $w^n_k$. Note that $w^n_k \equiv w^n_1 \circ w^{n-1}_k$ and $w^n_k \equiv w^n_2 \circ v_n$ are the only heap relations in $W'$ at $w^n_k$. Thus, we only need to check that $T_{W'}(w^n_1) \cup T_{W'}(w^{n-1}_k) = T_{W'}(w^n_1) \cup T_{W'}(w^n_2) \cup T_{W'}(v_n)$, and this is true because: by Lemma B.2 with $W'$ consistent at $w^{n-1}_k$ and $w^n_2$, we have $T_{W'}(w^n_1) \cup T_{W'}(w^{n-1}_k) = T_{W'}(w^n_2) \cup (T_{W'}(w^{n-1}_k) \cup T_{W'}(w^n_2))$; moreover, by the Lemma with $W'$ consistent at $v_n$ and $u_{n-1}$, we have $T_{W'}(v_n) \cup T_{W'}(v_n) = T_{W'}(w^n_2) \cup (T_{W'}(u_{n-1}) \cup T_{W'}(v_n))$.

For the cases 3, 4, and 4’, assume that $r$ is a root heap of a world sequent $W$. Note that, in any world sequent obtained by applying the rule Assoc to $W$, the heap $r$ is still a root heap.

3. In this case, we use only the rule Assoc; hence, the following world sequents are all consistent by Lemma B.4.

Note that the set of terminal heaps of a world sequent obtained by applying Assoc’s to $W$ is equal to that of $W$, as the rule Assoc does not create fresh terminals; thus, let $S' = \{u_1, \cdots, u_n\}$ be any non-empty set of terminal heaps in $W$. It suffices to show that, by applying Assoc’s to $W$, we can construct a world sequent $W'$ with a heap $w'$ such that $T(w') = S'$. The proof proceeds by induction on $n = |S'|$.

- i) $n = 1, 2$:
  There is nothing to prove for $n = 1$, so let $n = 2$. Since $W$ is consistent, $u_1 \wedge u_2$ holds by Lemma B.2. Then, by Lemma B.12, there exists a world sequent $W'$ with a heap $w'$ such that:
  - $\{W\} \Rightarrow^* \{W'\}$ by applying only the rule Assoc;
  - $w' \equiv u_1 \circ u_2$ holds in $W'$.

- ii) $n > 2$:
  By IH on $S' - \{u_n\}$, there exists a world sequent $W^1$ with a heap $w^1$ such that:
  - $\{W\} \Rightarrow^* \{W^1\}$ by applying only the rule Assoc;
  - $T(w^1) = S' - \{u_n\}$ holds in $W^1$.

By Lemma B.12 with $w^1 \not\supset r$, there exists a world sequent $W^2$ with a heap $w^2$ such that:

- $\{W^1\} \Rightarrow^* \{W^2\}$ by applying only the rule Assoc;
- $r \equiv w^1 \circ w^2$ holds in $W^2$.

Since we have $u_n \notin T(w^1)$ and $W^2$ is consistent, $u_n \in T(w^2)$ holds in $W^2$ by Lemma B.2. Thus, by Lemma B.12 with $w^1 \wedge u_n$, there exists a world sequent $W'$ with a heap $w'$ such that:

- $\{W^2\} \Rightarrow^* \{W'\}$ by applying only the rule Assoc;
- $w' \equiv w^1 \circ u_n$ (i.e., $T(w') = S'$) holds in $W'$.

This completes the proof for the case 3.

4. In this case, we use only the rule Assoc; hence, the following world sequents are all full by Lemma B.4.
Let $w$ be any heap in $W$. As in the case 3, the set of terminal heaps of all world sequents below is equal to that of $W$; thus, let $S'_1, S'_2$ be any non-empty sets of terminal heaps in $W$ with $S'_1 \cap S'_2 = \emptyset$ and $S'_1 \cup S'_2 = T(w)$. It suffices to show that, by applying Assoc's to $W$, we can construct a world sequent $W'$ with heaps $w'_1, w'_2$ such that $w \vdash w'_1 \circ w'_2$, $T(w'_1) = S'_1$ and $T(w'_2) = S'_2$ hold in $W'$.

The proof proceeds by induction on $n = |T(w)|$.

i) $n = 2$:

We have $S'_1 = \{w_1\}, S'_2 = \{w_2\}$ for some heaps $w_1 \neq w_2$ in $W$. By Lemma B.2 with $W$ consistent, $w_1 \wedge w_2$. By Lemma B.2 with $W$ elementary, a heap relation $w \vdash w_1 \circ w_2$ is already contained in $W$.

ii) $n > 2$:

Since $w$ is not a terminal heap, there exists heaps $w_1, w_2$ in $W$ with $w \vdash w_1 \circ w_2$. Let $S_i = T(w_i)$ and $S_{ij} = S_i \cap S'_j (i, j = 1, 2)$. Note that $S_i \cap S_j = \emptyset$ and $S_i \cup S_j = T(w)$ hold since $W$ is consistent. We can assume that $S_i \neq S'_j$ for any $i, j = 1, 2$. (Otherwise, there is nothing to prove.) We have the following cases.

\textbf{Case} $S_{ij} = \emptyset$ for some $i, j$:

Without loss of generality, let $i = 1$ and $j = 2$. By assumptions, $S_1 \subseteq S'_1$ holds so $S'_1 - S_1$ and $S'_2$ are not empty. Thus, by IH on $w_2$, there exists a world sequent $W'$ with heaps $w'_1, w'_2$ such that:

$\vdash \{W\} \leftrightarrow \{W'\}$ by applying only the rule Assoc;
$w_2 \vdash w'_1 \circ w'_2$ and $T(w'_1) = S'_1 - S_1$, $T(w'_2) = S'_2$ hold in $W'$.

By applying Assoc to $W'$, we get a world sequent $W''$ with a heap $w'_1$ such that $w \vdash w'_1 \circ w_1$ and $w'_1 \vdash w_1 \circ w'_2$ hold in $W'$. Therefore, by letting $w'_2 = w_2$, we have $T(w'_j) = S'_j$ in $W'' (j = 1, 2)$.

\textbf{Case} $S_{ij} \neq \emptyset$ for any $i, j = 1, 2$:

By IH on $w_i$'s with $S_i = S_{i1} \cup S_{i2} (i = 1, 2)$, there exists a world sequent $W'$ with heaps $w'_1, w'_2$ such that:

$\vdash \{W\} \leftrightarrow \{W'\}$ by applying only the rule Assoc;
$w_i \vdash w'_1 \circ w'_2$ and $T(w'_i) = S'_i - S_i$ hold in $W' (i, j = 1, 2)$.

It is easy to show that there exists a world sequent $W''$ with heaps $w'_1, w'_2$ such that:

$\vdash \{W\} \leftrightarrow \{W''\}$ by applying only the rule Assoc;
$w \vdash w'_1 \circ w'_2$, $w'_1 \vdash w_1 \circ w'_2$ hold in $W'' (j = 1, 2)$.

Since $S'_i = S_{ij} \cup S_{i2}$ holds, we have $T(w'_j) = S'_j$ in $W'' (j = 1, 2)$.

This completes the proof for the case 4.

4'. In this case, we use only the rule Assoc; hence, the following world sequents are all full by Lemma B.4.

Let $w$ be any heap in $W$. As in the case 3, the set of terminal heaps of all world sequents below is equal to that of $W$; thus, let $S'_1, S'_2$ be any non-empty sets of terminal heaps in $W$ with $T(w) \cap S'_1 = \emptyset$ and $T(w) \cup S'_1 = S'_2$. It suffices to show that, by applying Assoc's to $W$, we can construct a world sequent $W'$ with heaps $w'_1, w'_2$ such that $w \vdash w'_1 \circ w'_2$, $T(w'_1) = S'_1$ and $T(w'_2) = S'_2$ hold in $W'$.

By Lemma B.12 with $w \not\vdash r$, there exists a world sequent $W'$ with a heap $w'$ such that:

$\vdash \{W\} \leftrightarrow \{W'\}$ by applying only the rule Assoc;
$w \vdash w' \circ w'$ holds in $W'$.

Since $W'$ is consistent and $T(w) \cap S'_1 = \emptyset$, $S'_1 \subset T(w')$ holds in $W'$ by Lemma B.2. By this Lemma's case 4 on $w'$, there exists a world sequent $W''$ with a heap $w'_1, w'_2$ such that:

$\vdash \{W\} \leftrightarrow \{W''\}$ by applying only the rule Assoc;
$w'_1 \vdash w'_1 \circ w'_2$ and $T(w'_1) = S'_1$ holds in $W''$.

By applying Assoc to $W''$, we get a world sequent $W'''$ with a heap $w'_2$ such that $w'_2 \vdash w \circ w'_2$ holds in $W'''$. By letting $w'_1 = w'_1$, we have $T(w'_j) = S'_j$ in $W''' (j = 1, 2)$. This completes the proof for the case 4'.
8. In this case, we use only the rules NormPC, NormEmpty, NormEq and Assoc.

Let \( W \) be a world sequent which is sanitized and \( \star \)-ready (resp. \( \rightarrow \)-ready) for a given heap \( u \in W \). We will omit the phrase “(resp. \( \rightarrow \)-ready)” from now on for simplicity. Let \( w_\epsilon \) be the special empty heap in \( u \).

By Lemma B.14, there exists a world sequent \( W^1 \) such that:

- \( \{ W \} \rightarrow^* \{ W^1 \} \) by applying only the rules NormPC and NormEmpty;
- \( W^1 \) is \( \star \)-ready for \( u \) and sanitized;
- There is no empty heap except \( w_\epsilon \) in \( W^1 \).

Let \( n \) be the number of terminal heaps in \( W^1 \). By Lemma B.15 with \( n \), there exists a world sequent \( W'' \) such that:

- \( \{ W^1 \} \rightarrow^* \{ W'' \} \) by applying only the rules NormEq and Assoc;
- \( W'' \) is \( \star \)-ready for \( u \) and sanitized;
- There is no empty heap except \( w_\epsilon \) in \( W'' \);
- For any \( m \leq n \) and any non-empty set \( S^1 = \{ w_1^1, \cdots , w_m^1 \} \) of terminal heaps in \( W^1 \), there exists a unique heap \( w' \in W'' \) with \( T_{W''}(w') = S^1 \);
- For any heap \( w' \in W'' - W^1 \), \( |T_{W''}(w')| \leq n \).

Since NormEq and Assoc do not create any new terminals, we conclude that \( W'' \) is normalized and \( \star \)-ready for \( u \). This completes the proof for the case 8.

\( \square \)

**Lemma 6.8.** For a world sequent \( W \) of a particular kind, there exists a disjunctive derivation state \( \Psi \) such that:

1. \( \{ W \} \rightarrow^* \Psi \) by applying only the propagation rules;
2. \( \| \Psi \|_S \) implies \( \{ \{ W \} \|_S \) for any stack \( S \),

where one of the following holds:

5. \( W \) is \( \star \)-ready for a given heap \( w \), and every world sequent in \( \Psi \) is saturated as well as \( \star \)-ready for heap \( w \).
6. \( W \) is \( \rightarrow \)-ready for a given heap \( w \), and every world sequent in \( \Psi \) is saturated as well as \( \rightarrow \)-ready for heap \( w \).

**Proof.** In the proof, we will use only the propagation rules. Thus, by Corollary 6.6 and Lemma B.5, we do not have to prove that 2) holds and one of \( \star \)-ready-ness and \( \rightarrow \)-ready-ness is preserved. In other words, it suffices to construct a disjunctive derivation state satisfying 1) such that every world sequent in it is saturated.

Before starting to prove, we introduce several definitions to simplify our arguments. First, for any \( a_i, a_i' \in \mathbb{N} \) \((i = 1, 2, 3)\), define an ordering \(<\) on \( \mathbb{N} \times \mathbb{N} \) and \((\mathbb{N} \times \mathbb{N}) \times \mathbb{N} \) as follows:

- \((a_1, a_2) < (a_1', a_2')\) iff \( a_1 < a_1' \) or \( a_1 = a_1' \) and \( a_2 < a_2' \);
- \(((a_1, a_2), a_3) < ((a_1', a_2'), a_3')\) iff \((a_1, a_2) < (a_1', a_2')\) or \((a_1, a_2) = (a_1', a_2')\) and \( a_3 < a_3' \).

Next, for any world sequent \( W \), define \( \alpha(W) \) be the set of atomic heap relations in \( W \) such that the application of some propagation rule to them produces new heap relations. Note that heap relations in \( \alpha(W) \) must reside for non-terminals heaps. Finally, for any two world sequents \( W, W'' \), we write \( W \equiv W'' \) iff non-atomic heap relations of \( W \) and that of \( W'' \) are the same.

Let \( W \) be any world sequent which is \( \star \)-ready for a given heap \( w \). Let \( w_1, \cdots , w_N \) be a topological ordering of non-terminal heaps in \( W \) with partial ordering \( \prec^\prime \), i.e., \( w_N \) is the root heap in \( W \) and \( w_i \) is a heap having no ancestors in \( W - \{ w_{i+1}, \cdots , w_N \} \) \((n = 1, \cdots , N - 1)\). Now, for any world sequent \( W'' \) with \( W \equiv W'' \), define \( \chi(W'') \in \mathbb{N} \times \mathbb{N} \) as:

- \( \chi(W''; i) := \) the number of heap relations at \( w_i \) in \( \alpha(W'') \);
- \( \chi(W'') := \left\{ \begin{array}{ll} (m, \chi(W''; m)), & \text{if } m > 0, \\ (0, 0), & \text{if } m = 0, \end{array} \right. \)
where \( m := \max\{i : i = 1, \cdots, N \text{ and } \chi(W'; i) > 0\} \in \mathbb{N} \) (here, we use \( \max \phi := 0 \)). Now, for any disjunctive derivation state \( \Psi' \) such that \( W \equiv W' \) for any \( W' \in \Psi' \), define \( \chi(\Psi') \in (\mathbb{N} \times \mathbb{N}) \times \mathbb{N} \) as:

\[
\begin{align*}
\triangleright & \ \chi(\Psi'; (i, j)) := \text{the number of world sequents } W' \text{ in } \Psi' \text{ with } \chi(W') = (i, j); \\
\triangleright & \ \chi(\Psi') := (m, \chi(\Psi'; m)),
\end{align*}
\]

where \( m := \max\{\chi(W') : W' \in \Psi'\} \in \mathbb{N} \times \mathbb{N} \).

Let us start to prove this Lemma. It suffices to show the following: for any disjunctive derivation state \( \Psi' \) with \( \{W\} \rightarrow^* \{\Psi\} \) by applying only the propagation rules, there exists a disjunctive derivation state \( \Psi \) such that:

(a) \( \{\Psi\} \rightarrow^* \{\Psi\} \) by applying only the propagation rules;

(b) Every world sequent in \( \Psi \) is saturated.

The proof proceeds by induction on \( \chi(\Psi') \).

i) \( \chi(\Psi') = ((0, 0), k) \) for any \( k > 0 \):
In this case, the application of any propagation rules to any world sequent in \( \Psi \) does not produce any new heap relations, \( i.e. \), every world sequent in \( \Psi' \) is already saturated. Thus, \( \Psi = \Psi' \).

ii) \( \chi(\Psi') \geq ((1, 1), 1) \):
Let \((i, j), k) = \chi(\Psi') \) and \( W' \in \Psi' \) be a world sequent with \( \chi(W') = (i, j) \). Since \( i > 0 \), there exists an atomic heap relation \( \sigma_i = \alpha(W') \) at \( w_i \). By applying one of the propagation rules to \( \sigma_i \) at \( w_i \) in \( W' \), we get \( \Psi^2 = \Psi' - \{W'\} \cup \{W_1', \cdots, W_M', W_{k'}\} \) for some \( M \) and world sequents \( W_1', \cdots, W_k', W_{k'} \). We see \( \sigma_i = \alpha(W') \) and \( \sigma_i \notin \alpha(W_{k'}) \) hold \((m = 1, \cdots, M) \). Also, every heap relation produced by the above application of a propagation rule resides for a heap which has topological order less than \( i \). From these observations, it is easy to check that we have \( \chi(W_{k'}) < \chi(W') \) \((m = 1, \cdots, M) \) and thus \( \chi(\Psi^2) < \chi(\Psi') \). By IH on \( \Psi^2 \), we get a desired disjunctive derivation state \( \Psi \).

This completes the proof of this Lemma. \( \square \)

**Lemma 6.9 (Completeness of the rule Weaken).**
For a world sequent \( W \) of a particular kind, there exists a world sequent \( W' \) of another kind such that:

1) \( \{W\} \rightarrow^* \{W'\} \) by applying only [the rule Weaken];

2) \( \llbracket \{W'\} \rrbracket_S \) implies \( \llbracket \{W\} \rrbracket_S \) for any stack \( S \),

where the following holds:

7. \( W \) is saturated and \( \star\)-ready (\( \hookrightarrow\)-ready) for a given heap \( w \), and \( W' \) is sanitized and \( \star\)-ready (\( \hookrightarrow\)-ready) for heap \( w \).

**Proof.** In the proof, we will use only the rule Weaken. Thus, by Lemma B.5, we do not have to prove that and one of \( \star\)-ready-ness and \( \hookrightarrow\)-ready-ness is preserved. In other words, it suffices to construct a sanitized world sequent satisfying 1) and 2).

Let \( w_1, \cdots, w_N \) be the reverse of topological ordering of non-terminal heaps in \( W \) with partial ordering \( \nearrow \), \( i.e. \), \( w_1 \) is the root heap in \( W \) and \( w_n \) is a heap having no ancestors in \( W = \{w_1, \cdots, w_{n-1}\} \) \((n = 2, \cdots, N) \). For each \( n = 1, \cdots, N \), we inductively construct a world sequent \( W^n \) from \( W^{n-1} \) such that

\[
\begin{align*}
\triangleright & \ \{W^{n-1}\} \rightarrow^* \{W^n\} \text{ by applying the rule Weaken to atomic heap relations only at } w_n; \\
\triangleright & \ \text{In } W^n, \text{ there is no atomic heap relation at } w_n,
\end{align*}
\]

where we define \( W^0 := W \). Let \( W' := W^N \). Then, by the above construction, we naturally have that \( W' \) is sanitized and 1) is true.

Now, let us prove that 2) is also true, or equivalently that \( \llbracket W^n \rrbracket_S \) implies \( \llbracket W^{n-1} \rrbracket_S \) for any stack \( S \) \((n = 1, \cdots, N) \). To prove it, take any \( 1 \leq n \leq N \) and arbitrary stack \( S \). Without loss of generality, we can assume that in \( W^{n-1} \) there is only one atomic heap relation \( \sigma \) at \( w_n \). Then, it suffices to show that \( \llbracket W^n \rrbracket_S \) implies \( \llbracket \sigma \rrbracket_S \) since \( \llbracket W^{n-1} \rrbracket_S = \llbracket W^n \rrbracket_S \land \llbracket \sigma \rrbracket_S \).

Suppose we have \( \llbracket W^n \rrbracket_S \), and let us prove that \( \llbracket \sigma \rrbracket_S \) holds. There are four cases for \( \sigma \); however, we show the proof only for the most complicated case and other cases are similarly proved.
Case \( \sigma = w_n \neq [l \mapsto E] \) for some \( l, E \):

Since \( w_n \) is a non-terminal heap, there exist heaps \( u, v \) with \( w_n = u \circ v \). Since \( W \) is saturated, application of the rule \( \text{Prop} \rightarrow \not\exists \) to \( W \) with \( \langle \sigma, w_n = u \circ v \rangle \) does not produce any new atomic heap relations at \( u, v \). Since all atomic heap relations at \( u, v \) in \( W \) still remain in \( W^n \) (by the above construction), we have the following subcases.

(Subcase) \( \{ u \neq \epsilon, u \neq [l \mapsto E] \} \subset W^n \):
Since we have \( u \neq \epsilon \), \( \text{dom}(u) \neq \emptyset \) holds. Since we have \( u \neq \langle \emptyset \rangle, \emptyset \rangle \), either \( \text{dom}(u) \neq \{ \emptyset \} \) or \( \{ \emptyset \} = \emptyset \). If the former holds, then we have \( \text{dom}(w_n) \neq \{ \emptyset \} \); if the latter holds, then we have \( w_n(\emptyset) \neq \emptyset \). Thus, we have \( w_n \neq \langle \emptyset \\rangle \rightarrow \emptyset \).

(Subcase) \( \{ u \neq \epsilon, v \neq \epsilon \} \subset W^n \):
Since we have \( u \neq \epsilon \) and \( v \neq \epsilon \), \( |\text{dom}(u)| \geq 1 \) and \( |\text{dom}(v)| \geq 1 \) hold. By definition of \( \circ \), we have \( \text{dom}(u) \cap \text{dom}(v) = \emptyset \) and \( |\text{dom}(w_n)| \geq 2 \), thereby having \( w_n \neq \langle \emptyset \rangle \rightarrow \emptyset \).

(Subcase) \( \{ u \neq [l \mapsto E], v \neq [l \mapsto E] \} \subset W^n \):
Suppose either \( u \neq \emptyset \) or \( v \neq \emptyset \). Assume \( \text{dom}(u) \neq \emptyset \) and \( \text{dom}(v) \neq \emptyset \). Then, we obtain \( \text{dom}(w_n) \neq \emptyset \).

Hence, we have \( w_n \neq \langle \emptyset \rangle \rightarrow \emptyset \).

(Subcase) \( \{ v \neq \epsilon, v \neq [l \mapsto E] \} \subset W^n \):
This subcase is symmetric to the first one.

To conclude, \( W^n \) satisfies all the desired properties, and this completes the proof.

**Proposition 6.11** (Completeness of the rules \( \star\text{R} \) and \( \rightarrow\text{L} \)).

Consider a normalized world sequent \( W \) that is also \( \star \)-ready for heap \( w \). Suppose that we obtain \( \{ W \} \vdash \Psi \), by applying the rule \( \star\text{R} \) to a false formula \( A \star B \) about heap \( w \) for each heap relation \( w = w_1 \circ w_2 \) \( (i = 1, \ldots, n) \), and \( \{ W \} \vdash \Psi \), by applying the rule \( \star\text{R} \) to the same formula for another heap relation \( w = w \circ w_2 \). Then a conjunctive proof goal \( \Omega = \{ \Psi_1, \ldots, \Psi_n, \Psi_e \} \) satisfies:

1) \( \{ W \} \vdash \Omega \);
2) \( \{ \Omega \} \) implies \( \{ \{ W \} \} \) for any stack \( S \);
3) No world sequent in \( \Omega \) contains the false formula \( A \star B \) about heap \( w \);
4) Every world sequent in \( \Omega \) is still normalized and thus consistent.

Similarly for the rule \( \rightarrow\text{L} \) where we use each heap relation \( w_i = w \circ w_2 \).

**Proof.** Since \( W \) is \( \star \)-ready for heap \( w \), it contains all potential pairs of child heaps for \( w \) (that do not involve \( w_0 \)) and thus \( \{ \{ W \} \} \) and \( \{ \Omega \} \) are equivalent. Since we only eliminate the false formula \( A \star B \) about heap \( w \), every world sequent in \( \Omega \) has the same structure as the original world sequent \( W \) and thus continues to be normalized. Similarly for the rule \( \rightarrow\text{L} \).

**B.2.4 Completeness of the heap contradiction rules**

**Proposition 6.12** (Completeness of the heap contradiction rules).

A normalized world sequent \( W \) with no formulas other than \( \bot \) is canonical. That is, if \( \neg \{ W \} \) holds for any stack \( S \), we can construct its derivation using only the rules \( \bot\text{L} \), \( \text{ExpCont} \), and the heap contradiction rules. For the rule \( \text{ExpCont} \), we assume that \( \neg [\theta] \vdash \bot \) implies \( \neg \bot \).

**Proof.** Let \( W = \Theta; \Sigma || \Pi \) for some \( \Theta, \Sigma, \Pi \), and let \( S \) be any stack. For any set \( A \) of expression relations, heap relations or heap sequents, define \( [A] \) as follows:

\[
[A] := \bigwedge_{a \in A} [a].
\]

Then, we have the following cases since \( \neg [W] = \neg [\theta] \vee \neg [\Sigma] \vee \neg [\Pi] \).

**Case** \( [\Pi] \) is contradictory:
Since only formulas in \( W \) are \( \bot \), there exists a heap \( w \) such that \( \neg (S, w) \vdash \bot \) holds. Thus, \( W \) contains a heap sequent \( [\Gamma, \bot \Rightarrow \Delta] w \) for some \( \Gamma \) and \( \Delta \), thereby having a derivation of \( W \) by applying the rule \( \bot\text{L} \) to \( w \).

56
Case \([\Theta]_S\) is contradictory:
In this case, we have \(\neg[\Theta]_S = \neg[\Theta]_S\). Thus, by assumption, we have \(\Theta \vdash \bot\), thereby having a derivation of \(W\) by using the rule \(\text{ExpCont}\).

Case \([\Theta]_S\) holds and \([\Theta]_S \land \{\Pi\}_S\) is contradictory:
To simplify notations, for a heap \(w\) define \(\Sigma(w) := \{\text{heap relations of the form } \dot \omega \in \cdot \in \cdot \in \cdot \in \Sigma\}\). We write \([\Pi]_S = \Pi\] and \([\Pi]_S = \Pi\] for \([\Pi]_S = \Pi\] and \([\Pi]_S = \Pi\]. Let \(w_r\) be the special empty heap in \(W\). We have the following subcases by analyzing where a contradiction occurs.

(Subcase) \([\Theta]_S \land \{\Sigma(w_r)\}_S\) is contradictory:
There are three possibilities on \(\Sigma(w_r)\) as follows.

i) \(w_r \not\in \Sigma(w_r)\):
We get a derivation of \(W\) by applying the rule \(\text{Cont} \not\in \rightarrow w_r\).

ii) \(w_r \in [l \rightarrow E] \in \Sigma(w_r)\) for some \(l, E\):
We get a derivation of \(W\) by applying the rule \(\text{Cont} \rightarrow \rightarrow w_r\).

iii) \(\Sigma(w_r)\) does not contain heap relations of the forms i) and ii):
Let us show the claim that \([\Theta]_S \land \{\Sigma(w_r)\}_S\) does not have a contradiction. Since \(W\) is normalized, \(w_r\) is an isolated node in the graph of heaps induced by \(W\); therefore, \(\Sigma(w_r)\) does not contain heap relations of the form \(w_r \not\in u \circ v\). From this, \(\Sigma(w_r)\) contains heap relations only of the forms \(w_r \not\in \epsilon\) and \(w_r \not\in [l \rightarrow E]\), which implies the above claim.

(Subcase) \([\Theta]_S \land \{\Sigma(w)\}_S\) is contradictory for some terminal heap \(w\) in \(W\):
We have the following possibilities on \(\Sigma(w)\).

i) \(w \equiv [l \rightarrow E] \in \Sigma(w)\) for some \(l, E\):
(a) \(v \not\in [l' \rightarrow E'] \in \Sigma(w)\) for some \(l', E'\) with \([l \rightarrow E] = [l' \rightarrow E']\):
Since we have \([l']_S = [l']_S\) and \([E]_S = [E']_S\), we have \(\neg[\Theta]_S = [l']_S\) and \(\neg[\Theta]_S = [E']_S\).
By assumption, we have \(\Theta \vdash l = l'\) and \(\Theta \vdash E = E'\) (i.e. \(\Theta \vdash [l \rightarrow E] = [l' \rightarrow E']\)).
Thus, we have a derivation of \(W\) by applying the rule \(\text{Cont} \equiv \rightarrow w\).

(b) \(w \equiv [l' \rightarrow E'] \in \Sigma(w)\) for some \(l', E'\) with \([l \rightarrow E] = [l' \rightarrow E']\):
Since we have \([l']_S = [l']_S\) or \([E]_S = [E']_S\), we have \(\Theta \vdash [l \rightarrow E] \equiv [l' \rightarrow E']\) by using the argument of (a). Thus, we have a derivation of \(W\) by applying the rule \(\text{Cont} \equiv \rightarrow w\).

(c) Otherwise:
Let us show the claim that \([\Theta]_S \land \{\Sigma(w)\}_S\) does not have a contradiction. Since \(W\) is normalized, \(w \not\in \epsilon \in \Sigma(w)\) holds, and since \(w\) is a terminal heap, \(\Sigma(w)\) does not contain heap relations of the form \(w \not\in u \circ v\). Moreover, by the assumption of this possibility, \(w \not\in [l' \rightarrow E'] \in \Sigma(w)\) implies \([l \rightarrow E] = [l' \rightarrow E']\), and \(w \equiv [l' \rightarrow E'] \in \Sigma(w)\) implies \([l \rightarrow E] = [l' \rightarrow E']\); thus, heap relations of the forms \(w \not\in [l' \rightarrow E']\) and \(w \equiv [l' \rightarrow E']\) (as well as \(w \not\in \epsilon\)) do not make a contradiction of \([\Theta]_S \land \{\Sigma(w)\}_S\). Hence, the above claim is proved.

ii) \(\Sigma(w)\) does not contain heap relations of the form i):
By the same argument of i) (c), \(\Sigma(w)\) contains heap relations only of the forms \(w \not\in \epsilon\) and \(w \not\in [l \rightarrow E]\), which implies that \([\Theta]_S \land \{\Sigma(w)\}_S\) does not have a contradiction.

(Subcase) \([\Theta]_S \land \{\Sigma(w)\}_S\) is not contradictory for any empty or terminal heap \(w\) in \(W\):
Let \(w_1, \cdots, w_N\) be terminal heaps in \(W\) such that \(w_i \equiv [l_i \rightarrow E_i] \in \Sigma(w_i)\) for some \(l_i\) and \(E_i\) (\(i = 1, \cdots, N\)), and let \(w_{N+1}, \cdots, w_{N+M}\) be the remaining terminal heaps in \(W\), where \(N, M \geq 0\). By the assumption of this subcase and the arguments of the above subcase, the followings hold:

- If \(\sigma \in \Sigma(w_r)\), then
  - \(\sigma = w_r \equiv \epsilon\), or
  - \(\sigma = w \equiv [l \rightarrow E]\) for some \(l, E\);
If $\sigma \in \Sigma(w_i) (1 \leq i \leq N)$, then
\begin{itemize}
    \item $\sigma = w \neq \epsilon$, or
    \item $\sigma = w \neq [l \mapsto E]$ for some $l, E$ with $[[l \mapsto E]]_S \neq [[l_i \mapsto E_i]]_S$, or
    \item $\sigma = w \neq [l \mapsto E]$ for some $l, E$ with $[[l \mapsto E]]_S = [[l_i \mapsto E_i]]_S$.
\end{itemize}

If $\sigma \in \Sigma(w_i) (N + 1 \leq i \leq M)$, then
\begin{itemize}
    \item $\sigma = w \neq \epsilon$, or
    \item $\sigma = w \neq [l \mapsto E]$ for some $l, E$.
\end{itemize}

Now, let us consider the following cases.

i) $[l_i]_S = [l_j]_S$ for some $i \neq j$:
Since $W$ is full, there exists a heap $w_{ij}$ in $W$ such that $w_{ij} \vdash L \circ R \in \Sigma$. From $[l_i]_S = [l_j]_S$, we get $\Theta \vdash l_i = l_j$ by a similar argument in the above subcase. Thus, we have a derivation of $W$ by applying the rule $\text{Cont} \, \circ \mapsto$ to $w_{ij}$.

ii) $[l_i]_S \neq [l_j]_S$ for all $i \neq j$:
Let us show the claim that $[\Theta]_S \land [\Pi]_S$ does not have a contradiction. To do so, first assume $[\Theta]_S \land [\Pi]_S$. Note that we cannot get any contradiction from terminal or empty heaps in $W$ by the assumption of this subcase. So, let $w$ be any non-terminal heap in $W$. Since $W$ is sanitized, $\Sigma(w)$ contains heap relations only of the form $w = u \circ v$. Without loss of generality, assume that $\Sigma(w) \neq \emptyset$. So, we have $w = w_{k_1} \circ \cdots \circ w_{k_n}$ for some $n \geq 2$ and $\{k_1, \cdots, k_n\} \subset \{1, \cdots, N + M\}$. Since $W$ is consistent, we conclude that this is the only information on the heap $w$ which we can obtain from $[\Pi]_S$. (That is, if we get $w = t_1 \circ \cdots \circ t_m$ from $[\Pi]_S$ for some $m \geq 2$ and terminal heaps $t_i$ in $W$ ($i = 1, \cdots, m$), then we have $n = m$ and $\{w_{k_1}, \cdots, w_{k_n}\} = \{t_1, \cdots, t_m\}$.) Since $W$ is elementary, $k_i \neq k_j$ for all $i \neq j$. Finally, since we know $[l_k]_S \neq [l_k]_S$, $w$ is a valid heap having no contradictions. In sum, we cannot derive any semantic contradiction from heaps in $W$, and thus $[\Theta]_S \land [\Pi]_S$ is not contradictory.