A Concurrent Specification of POSIX File Systems Technical Report

Gian Ntzik*[1], Pedro da Rocha Pinto†[2], Julian Sutherland‡[2], and Philippa Gardner§[2]

[1]Amadeus
[2]Imperial College London

1 Syntax Encoding

We define the encoding for the syntactic constructs used in the specifications presented in the paper into our core specification language as follows:

$$\begin{aligned}
\textbf{if } P \textbf{ then } \phi \textbf{ else } \psi \textbf{ fi} &\triangleq ([P];\phi) \sqcup ([\neg P];\psi) \quad \text{when } P \text{ is pure} \\
\lambda x_1, x_2 \ldots, x_n.\, \phi &\triangleq \lambda x_1.\, \lambda x_2. \ldots . \lambda x_n.\, \phi \\
f(x_1, x_2 \ldots x_n) &\triangleq ((f x_1) x_2) \ldots x_n \\
\textbf{let } f(\vec{x}) = \phi \textbf{ in } \psi &\triangleq \textbf{let } f = \lambda \vec{x}, \texttt{ret}.\, \phi \textbf{ in } \psi \\
\textbf{letrec } f(\vec{x}) = \phi \textbf{ in } \psi &\triangleq \textbf{let } f = \mu A.\, \lambda \vec{x}, \texttt{ret}.\, \phi \textbf{ in } \psi \\
\texttt{return } x &\triangleq [\texttt{ret} = x] \\
\texttt{return } f(\vec{x}) &\triangleq \textbf{let } y = f(\vec{x}) \textbf{ in } \texttt{return } y \\
\texttt{return } f_1(\vec{x_1}) \sqcap \ldots f_n(\vec{x_n}) &\triangleq \texttt{return } f_1(\vec{x_1}) \sqcap \ldots \texttt{return } f_n(\vec{x_n}) \\
\textbf{let } x = f(\vec{y}) \textbf{ in } \phi &\triangleq \exists x.\, f(\vec{y}, x); \phi \\
\textbf{let } x = f(\vec{y}); \phi &\triangleq \textbf{let } x = f(\vec{y}) \textbf{ in } \phi \\
\textbf{let } w, z = f(\vec{x}) \parallel f(\vec{y}) \textbf{ in } \phi &\triangleq \exists w, z.\, f(\vec{x}, w) \parallel g(\vec{y}, z); \phi \\
\textbf{let } w, z = f(\vec{x}) \parallel f(\vec{y}); \phi &\triangleq \textbf{let } w, z = f(\vec{x}) \parallel f(\vec{y}) \textbf{ in } \phi
\end{aligned}$$

2 Atomicity and Refinement

In this section we formalise the assertion language, the assertion model, the specification language and the associated semantics of refinement. We establish an adequacy theorem (theorem 1) justifying our refinement relation with respect to an operational semantics. The semantics and proofs are inspired by the earlier work of Turon and Wand [**?**] and follow a similar structure. Major differences arise from our combination with TaDA [**?**], as the assertion model and the atomicity semantics are entirely different. We then define general refinement laws, refinement laws for atomicity and encode the concept of hybrid specification statements that combine atomic and non-atomic effects and define associated refinement laws, and show that they are sound.

*gian.ntzik@amadeus.com
†pmd09@doc.ic.ac.uk
‡jhs110@doc.ic.ac.uk
§pg@doc.ic.ac.uk

2.1 Specification Language

Specification programs and the assertions used in specification statements share the same variable environment. We do not distinguish between program variables and logical variables. For the general theory of atomicity and refinement that we develop in this section, we only use a basic set of boolean and integer values and associated expressions. We leave these definitions open-ended and applications of our theory, such as the POSIX specification, can extend these as appropriate.

Definition 1 (Variables and values)**.** *Let* VAR *be a countable set of variables. Let* VAL *be the set of values assigned to variables, at least comprising booleans, integers and the unit value,* $\mathbf{1} \triangleq \{()\}$*.*

$$\text{VAL} \triangleq \mathbb{B} \cup \mathbb{Z} \cup \mathbf{1} \ldots$$

Variable stores, $\rho \in \text{VARSTORE} \triangleq \text{VAR} \to \text{VAL}$*, assign values to variables.*

The variable environment defined above is basic, and will require extensions when defining the semantics of assertions and specification programs. This is because some variables, for functions and recursion, will receive special treatment.

Definition 2 (Expressions)**.** *Expressions,* $e, e' \in \text{EXPR}$*, are defined by the grammar:*

Expressions	$e, e' ::=$	v	*value* $v \in \text{VAL}$
		$\mid x$	*variable* $x \in \text{VAR}$
Boolean Expressions		$\mid \neg e$	*negation*
		$\mid e \wedge e'$	*conjunction*
		$\mid e \vee e'$	*disjunction*
		$\mid e = e'$	*equality*
		$\mid e < e'$	*inequality*
Integer Expressions		$\mid e + e'$	*addition*
		$\mid e - e'$	*subtraction*
		$\mid e \cdot e'$	*multiplication*
		$\mid e \div e'$	*division*
		$\mid \ldots$	

Here we have chosen the $\div$ for division, to distinguish from the path separator used in POSIX. Expressions have a standard, albeit partial, denotational semantics.

Definition 3 (Expression evaluation)**.** *Expression evaluation,* $[\![-]\!]^- : \text{VARSTORE} \to \text{EXPR} \rightharpoonup \text{VAL}$*, is defined as a partial function over expressions parameterised by a variable store:*

$$\begin{aligned}
[\![v]\!]^\rho &\triangleq v \\
[\![x]\!]^\rho &\triangleq \rho(x) \\
[\![\neg e]\!]^\rho &\triangleq \neg\, [\![e]\!]^\rho \quad \textit{if } [\![e]\!]^\rho \in \mathbb{B} \\
[\![e \wedge e']\!]^\rho &\triangleq [\![e]\!]^\rho \wedge [\![e']\!]^\rho \quad \textit{if } [\![e]\!]^\rho \in \mathbb{B} \textit{ and } [\![e']\!]^\rho \in \mathbb{B} \\
[\![e \vee e']\!]^\rho &\triangleq [\![e]\!]^\rho \vee [\![e']\!]^\rho \quad \textit{if } [\![e]\!]^\rho \in \mathbb{B} \textit{ and } [\![e']\!]^\rho \in \mathbb{B} \\
[\![e = e']\!]^\rho &\triangleq [\![e]\!]^\rho = [\![e']\!]^\rho \\
[\![e < e']\!]^\rho &\triangleq [\![e]\!]^\rho < [\![e']\!]^\rho \quad \textit{if } [\![e]\!]^\rho \in \mathbb{Z} \textit{ and } [\![e']\!]^\rho \in \mathbb{Z} \\
[\![e - e']\!]^\rho &\triangleq [\![e]\!]^\rho - [\![e']\!]^\rho \quad \textit{if } [\![e]\!]^\rho \in \mathbb{Z} \textit{ and } [\![e']\!]^\rho \in \mathbb{Z} \\
[\![e \cdot e']\!]^\rho &\triangleq [\![e]\!]^\rho \cdot [\![e']\!]^\rho \quad \textit{if } [\![e]\!]^\rho \in \mathbb{Z} \textit{ and } [\![e']\!]^\rho \in \mathbb{Z} \\
[\![e \div e']\!]^\rho &\triangleq [\![e]\!]^\rho \div [\![e']\!]^\rho \quad \textit{if } [\![e]\!]^\rho \in \mathbb{Z} \textit{ and } [\![e']\!]^\rho \in \mathbb{Z} \setminus \{0\}
\end{aligned}$$

In all other cases, the result is undefined.

When the denotation of an expression is undefined, a fault is triggered in the semantics of our specification language.

Our assertion language is based on TaDA [**?**], extending intuitionistic separation logic [**?**] with regions, guards and abstract predicates. However, we exclude the atomicity tracking components.

Definition 4 (Assertion Language)**.** *Assertions,* $P, Q, R \in$ Assrt*, are defined by the grammar:*

Assertions	$P, Q, R ::=$	false	*falsehood*
		$\mid \mathsf{true}$	*truthfulness*
		$\mid P * Q$	*separating conjunction*
		$\mid P \wedge Q$	*conjunction*
		$\mid P \vee Q$	*disjunction*
		$\mid \neg P$	*negation*
		$\mid \exists x.\, P$	*existential quantification*
		$\mid \forall x.\, P$	*universal quantification*
		$\mid P \Rightarrow Q$	*implication (material)*
		$\mid e \mapsto e'$	*heap cell at* e *storing* e'
		$\mid \mathbf{t}^k_\alpha(\vec{e}, e')$	*shared region* α*, of type* $\mathbf{t}$*, region level* k*, parameterised by* $\vec{e}$ *and with abstract state* e'
		$\mid I(\mathbf{t}^k_\alpha(\vec{e}, e'))$	*shared-region interpretation*
		$\mid [\mathsf{G}(\vec{e})]_\alpha$	*guard* G *for region* α*, parameterised by* $\vec{e}$
		$\mid \mathsf{ap}(\vec{e})$	*application of abstract predicate* ap *to parameters* $\vec{e}$
		$\mid I_a(\mathsf{ap}(\vec{e}))$	*interpretation of abstract predicate*
		$\mid \mathsf{pred} e$	*application of concrete predicate* pred *to* e
		$\mid e$	*expression*
Concrete Predicates	$\mathsf{pred} ::=$	$\lambda x.\, P$	*non-recursive predicate*
		$\mid \mu X.\, \lambda x.\, P$	*recursive predicate*
		$\mid X$	*recursion variable*

Recursion variables, $X \in$ AssrtRecVars*, are taken from a countable set, disjoint from* Var*. The binding precedence, from strongest to weakest, is:* $\neg, *, \wedge, \vee, \forall, \exists, \Rightarrow$*.*

We define a core specification language, based on the atomic specification statement. As we will see later, the hybrid and Hoare specification statements are defined in terms of sequences of atomic statements.

Definition 5 (Specification Language)**.** *The language of specifications,* $\mathcal{L}$*, is defined by the following grammar:*

Specifications	$\phi, \psi ::=$	$\phi ; \psi$	*Sequential composition*
		$\mid \phi \parallel \psi$	*Parallel composition*
		$\mid \phi \sqcup \psi$	*Angelic choice*
		$\mid \phi \sqcap \psi$	*Demonic choice*
		$\mid \exists x.\, \phi$	*Existential quantification*
		$\mid \mathbf{let}\ f = F\ \mathbf{in}\ \phi$	*Function binding*
		$\mid Fe$	*Function application*
		$\mid \forall \vec{x}.\, \langle P, Q \rangle_k$	*Atomic specification statement*
Functions	$F ::=$	f	*Function variable*
		$\mid A$	*Recursion variable*
		$\mid F_l$	*Function literal*
Function Literals	$F_l ::=$	$\mu A.\, \lambda x.\, \phi$	*Recursive function*
		$\mid \lambda x.\, \phi$	*Function*

where $k \in$ RLevel *is defined in section 2.2. Recursion variables,* $A \in$ RecVars*, and function variables,* $f \in$ FuncVars*, are taken from disjoint countable sets, both disjoint from* Var*, and cannot be bound or free in* P *or* Q*. Predicate-recursion variables,* $X \in$ AssrtRecVars*, are also disjoint from* RecVars *and* FuncVars *and cannot be bound or free in* ϕ*. The operator binding precedence, from strongest to weakest, is:* $Fe, \mu A.\,, \lambda x.\,, \sqcap, \sqcup, \exists x.\,;, \parallel$*, with parentheses used to enforce order.*

2.2 Model

Our assertions about the shared state are based on those of TaDA. The same applies to the models of assertions which we develop here. Therefore, the contents of this chapter are heavily based on the model of technical report of TaDA [**?**].

Regions, guards, region levels, atomicity contexts and abstract predicates are merely instrumentation – also referred to as ghost state in the literature – of the concrete shared state, the purpose of which to enable scalable, modular and compositional reasoning for concurrent programs. We take the concrete state shared between threads to be the heap memory.

Definition 6 (Heaps)**.** *Let* Addr *be the set of addresses such that* Addr $\subseteq$ Val*. A heap,* $h \in \text{Heap} \triangleq \text{Addr} \overset{fin}{\rightharpoonup} \text{Val}$*, is a finite partial function from addresses to values. Heaps form a separation algebra* $(\text{Heap}, \uplus, \emptyset)$*, where* $\uplus$ *is the disjoint union of partial functions and* $\emptyset$ *is the partial function with an empty domain. Heaps are ordered by resource ordering,* $h_1 \leq h_2 \overset{def}{\iff} \exists h_3.\, h_1 \uplus h_3 = h_2$*.*

Definition 7 (Guards and Guard Algebras)**.** *Let* Guard *be a set containing all the possible guards. A guard algebra* $\zeta = (\mathcal{G}, \bullet, \mathbf{0}, \mathbf{1})$ *comprises:*

- *a guard carrier set* $\mathcal{G} \subseteq \text{Guard}$*,*
- *a partial, binary, associative and commutative operator* $\bullet : \mathcal{G} \times \mathcal{G} \rightharpoonup \mathcal{G}$*,*
- *an identity element,* $\mathbf{0} \in \mathcal{G}$*, such that* $\forall x \in \mathcal{G}.\, \mathbf{0} \bullet x = x$*,*
- *a maximal element,* $\mathbf{1} \in \mathcal{G}$*, such that* $\forall x \in \mathcal{G}.\, x \leq \mathbf{1}$*, where* $x \leq y \overset{def}{\iff} \exists z.\, x \bullet z = y$*.*

We denote the set of all guard algebras as GAlg. A guard algebra, is a separation algebra with a single unit, $\mathbf{0}$. Given guard algebra $\zeta \in \text{GAlg}$ we denote: the carrier set of guards with $\mathcal{G}_\zeta$, the identity (zero) guard with $\mathbf{0}_\zeta$, the maximal guard with $\mathbf{1}_\zeta$ and the resource ordering $\leq_\zeta$. Let $g_1, g_2 \in \mathcal{G}_\zeta$. We denote with $g_1 \# g_2$ the fact that $g_1 \bullet g_2$ is defined.

Regions have an abstract state. Each region is associated with a labelled transition system, the transitions of which define how the abstract state of the region can be atomically updated. Each transition is labelled by a guard. A thread has the capability to update the abstract state of a region, only if it owns the guard resource that "guards" the transition by which the update is allowed.

Definition 8 (Abstract States and Transition Systems)**.** *Let* AState *be a set containing all the possible region abstract states. Let* $\zeta \in \text{GAlg}$*. A* guard-labelled transition system*,* $\mathcal{T} : \mathcal{G}_\zeta \overset{mono}{\rightarrow} \mathcal{P}(\text{AState} \times \text{AState})$*, is a function mapping guards to abstract state binary relations. The mapping is required to be monotone with respect to guard resource ordering* $(\leq_\zeta)$ *and subset ordering; having more guard resources permits more transitions. The set of all* ζ*-labelled transition systems is denoted by* Asts_ζ*.*

We do not restrict the transition relations for a guard $g \in \mathcal{G}_\zeta$. However, in general we will use the transitive-reflexive closure $\mathcal{T}(g)^*$.

Regions have types, that associate regions of the same type with a guard algebra and a guard-labelled transition system.

Definition 9 (Abstract Region Types)**.** *Let* RTName *be the set of region type names. An* abstract region typing*,*

$$\mathbf{T} \in \text{ARType} \triangleq \text{RTName} \rightarrow \biguplus_{\zeta \in \text{GAlg}} \{\zeta\} \times \text{Asts}_\zeta$$

maps region type names to pairs of guard algebras and guard-labelled transition systems. Let $\mathbf{t} \in \text{RTName}$*. The guard labelled transition system of the region type name* $\mathbf{t}$ *is denoted by* $\mathcal{T}_\mathbf{t}$*.*

Definition 10 (Abstract Predicates)**.** *Let* APName *be the set of abstract predicate names. An* abstract predicate *ap* $\in \text{APName} \times \text{Val}^*$*, comprises an abstract predicate name and a list of parameters. An* abstract predicate bag*,* $b \in \text{APBag} \triangleq \mathcal{M}_{fin}(\text{APName} \times \text{Val}^*)$*, is a finite multiset of abstract predicates. Abstract predicate bags form a separation algebra,* $(\text{APBag}, \cup, \emptyset)$*, where* $\cup$ *is the multiset union and* $\emptyset$ *is the empty multiset. Abstract predicate bags are ordered by the subset order* $\subseteq$*.*

Regions may refer to other regions and circularities may arise. This is a problem for the refinement laws that allow us to open a region, by replacing it with its interpretation. During the derivation of a refinement, if there is a circularity, then the refinement laws could be used to open the same region twice. This is unsound, as it would replicate the resource encapsulated by the region. To prevent this unsoundness, we associate each region and specification statement with a region level.

Definition 11 (Region Levels)**.** *A* region level*,* $k \in \text{RLEVEL} \triangleq \mathbb{N}$*, is a natural number. Levels are ordered by the* $\leq$ *ordering on natural numbers.*

Intuitively, region levels track the nesting depth of regions. The region level associated with a region indicates how deeply the region is nested. The region level associated with a specification statement indicates how far we can look into regions. In order to open a region, we require that the region level associated with the region is less than the region level associated with the specification statement. When a region is opened, the region level associated with the specification statement containing it is decreased. Then, if the same region is encountered again, its region level will be greater that than of the specification statement, and thus it will not be possible to open it again.

Each region has a unique identifier, which is used to identify the region's type, region level and parameters.

Definition 12 (Region Assignments)**.** *Let* RID *be a countable set of region identifiers. A* region assignment*,* $r \in \text{RASS} \triangleq \text{RID} \xrightarrow{fin} \text{RLEVEL} \times \text{RTNAME} \times \text{VAL}^*$*, is a finite partial function from region identifiers to region levels and parameterised region type names. Region assignments are ordered by extension ordering:* $r_1 \leq r_2 \overset{def}{\iff} \forall \alpha \in \text{dom}(r_1).\, r_2(\alpha) = r_1(\alpha)$*.*

In the following definitions, we assume a fixed abstract region typing, $\mathbf{T} \in \text{ARTYPE}$. Each region in a region assignment, is associated with guards from the guard algebra defined in the region typing.

Definition 13 (Guard Assignments)**.** *Let* $r \in \text{RASS}$ *be a region assignment. A* guard assignment*,*

$$\gamma \in \text{GASSN}_r \triangleq \prod_{\alpha \in \text{dom}(r)} \mathcal{G}_{\mathbf{T}(r(\alpha)\downarrow_2)\downarrow_1}$$

is a mapping from the regions declared in the region assignment r *to the guards of the appropriate type for each region. The guards assigned to a region with region identifier* α *are denoted by* $\gamma(\alpha)$*. Guard assignments form a separation algebra,* $(\text{GASSN}_r, \bullet, \lambda\alpha.\, \mathbf{0}_{\mathbf{T}(r(\alpha)\downarrow_2)\downarrow_1})$*, where* $\bullet$ *is the pointwise lift of guard composition:*

$$\gamma_1 \bullet \gamma_2 \triangleq \lambda\alpha.\, \gamma_1(\alpha) \bullet \gamma_2(\alpha)$$

For $\gamma_1 \in \text{GASSN}_{r_1}$ *and* $\gamma_2 \in \text{GASSN}_{r_2}$ *with* $r_1 \leq r_2$*, guard assignments are ordered extensionally:*

$$\gamma_1 \leq \gamma_2 \overset{def}{\iff} \forall \alpha \in \text{dom}(\gamma_1).\, \gamma_1(\alpha) \leq \gamma_2(\alpha)$$

Each region in a region assignment, is associated with an abstract state: the abstract state of the region.

Definition 14 (Region States)**.** *Let* $r \in \text{RASS}$ *be a region assignment. A* region state

$$\beta \in \text{RSTATE}_r \triangleq \text{dom}(r) \to \text{ASTATE}$$

is a mapping from the regions declared in r *to abstract states. For* $\beta_1 \in \text{RSTATE}_{r_1}$ *and* $\beta_2 \in \text{RSTATE}_{r_2}$*, with* $r_1 \leq r_2$*, region states are ordered extensionally:*

$$\beta_1 \leq \beta_2 \overset{def}{\iff} \forall \alpha \in \text{dom}(\beta_1).\, \beta_1(\alpha) = \beta_2(\alpha)$$

Hitherto we have given the semantic definitions required for regions. Now we proceed to develop the semantics definitions for the instrumented states that constitute the models of the assertion language of definition 4. We call these instrumented states *worlds*.

Definition 15 (Worlds)**.** *A* world

$$w \in \text{WORLD} \triangleq \biguplus_{r \in \text{RASS}} (\{r\} \times \text{HEAP} \times \text{APBAG} \times \text{GASSN}_r \times \text{RSTATE}_r)$$

consists of a region assignment, a heap, an abstract predicate bag, a guard assignment and a region state.

Worlds are composed, provided they agree on the region assignment and region state, by composing the heap, abstract predicate bag and guard assignment components in their respective separation algebras. Worlds form a multi-unit separation algebra $(\text{WORLD}, \circ, \mathsf{emp})$*, where*

$$(r, h_1, b_1, \gamma_1, \beta) \circ (r, h_2, b_2, \gamma_2, \beta) \triangleq (r, h_1 \uplus h_2, b_1 \cup b_2, \gamma_1 \bullet \gamma_2, \beta) \qquad \mathsf{emp} \triangleq \left\{(r, \emptyset, \emptyset, \lambda\alpha.\, \mathbf{0}_{\mathbf{T}(r(\alpha)\downarrow_2)\downarrow_1}, \beta)\right\}$$

Worlds are ordered by product order:

$$(r_1, h_1, b_1, \gamma_1, \beta_1) \leq (r_2, h_2, b_2, \gamma_2, \beta_2) \overset{def}{\iff} r_1 \leq r_2 \wedge h_1 \leq h_2 \wedge b_1 \leq b_2 \wedge \gamma_1 \leq \gamma_2 \wedge \beta_1 \leq \beta_2$$

Thus, if $w_1 \leq w_2$, we can get w_2 from w_1 by adding new regions, with arbitrary associated type name and state, and adding new heap, abstract predicates and guards.

Let $w \in \textsc{World}$ be a world. We denote its region assignment component with r_w, its heap component with h_w, its abstract predicates component with b_w, its guard assignment component with γ_w and its region states component with β_w.

Definition 16 (World Predicates)**.** *A* world predicate*, $p, q \in \textsc{WPred} \triangleq \mathcal{P}^{\uparrow}(\textsc{World})$, is a set of worlds that is upwards closed with respect to the world ordering: $\forall w \in p. \exists w'. w \leq w' \Rightarrow w' \in p$. We get composition of world predicates by lifting the composition of worlds[1]:*

$$p * q \triangleq \{w \mid \exists w' \in p, w'' \in q. w = w' \circ w''\}$$

World predicates form a separation algebra, $(\textsc{WPred}, *, \textsc{World})$.

Environment interference is abstracted by the rely relation.

Definition 17 (Rely Relation)**.** *The rely relation, $R \subseteq \textsc{World} \times \textsc{World}$, is the smallest reflexive and transitive relation that satisfies the following rule:*

$$\frac{g \# g' \qquad (s, s') \in \mathcal{T}_{\mathbf{t}(n)}(g')^*}{(r[\alpha \mapsto (k, \mathbf{t}, v)], h, b, \gamma[\alpha \mapsto g], \beta[\alpha \mapsto s]) R (r[\alpha \mapsto (k, \mathbf{t}, v)], h, b, \gamma[\alpha \mapsto g], \beta[\alpha \mapsto s'])}$$

The rely relation rule states that the environment can update a region, if it owns a guard g' for which the update is allowed and as long as that guard is compatible with the thread's own guard g.

Interference is explicitly confined to shared regions.

Definition 18 (Guarantee Relation)**.** *Let $k \in \textsc{RLevel}$ be a region level. The* guarantee relation*, $G_{k;} \subseteq \textsc{World} \times \textsc{World}$, is defined as:*

$$w \; G_{k;} \; w' \overset{def}{\iff}$$
$$\forall \alpha. (\exists k' \geq k. r_w(\alpha) = (k', -, -)) \Rightarrow \beta_w(\alpha) = \beta_{w'}(\alpha)$$

The guarantee relation enforces that regions with level k or higher cannot be modified.

Definition 19 (Stable World Predicates and Views)**.** *A* stable world predicate *is a world predicate that is closed under the rely relation.*

$$p \text{ stable} \overset{def}{\iff} R(p) \subseteq p$$

Stable world predicates are referred to as views*. The set of views is denoted by* $\textsc{View}$.

$$\textsc{View} \triangleq \{p \in \textsc{WPred} \mid R(p) \subseteq p\}$$

Ordering on views is defined by:

$$p \leq q \overset{def}{\iff} \forall w \in p, w' \in q. w \leq w'$$

Lemma 1. *Stable world predicates are closed under $*$, $\cup$ and $\cap$:*

$$p \text{ stable} \wedge q \text{ stable} \Rightarrow p * q \text{ stable}$$
$$p \text{ stable} \wedge q \text{ stable} \Rightarrow p \cup q \text{ stable}$$
$$p \text{ stable} \wedge q \text{ stable} \Rightarrow p \cap q \text{ stable}$$

In the paper, we have given examples of how regions are interpreted into the shared state they encapsulated and how abstract predicates are interpreted to their implementation through respective interpretation functions. We now formally define these interpretation functions.

[1]The result of the composition is upwards closed: any extension to the composition of two worlds can be tracked back and applied to one of the components.

Definition 20 (Region Interpretation)**.** *A* region interpretation

$$I \in \textsc{RInterp} \triangleq (\textsc{RLevel} \times \textsc{RTName} \times \textsc{Val}^*) \times \textsc{RId} \times \textsc{AState} \to \textsc{Assrt}$$

associates an assertion with each abstract state of each parameterised region.

Definition 21 (Abstract Predicate Interpretation)**.** *An* abstract predicate interpretation

$$I_a \in \textsc{APInterp} \triangleq \textsc{APName} \times \textsc{Val}^* \to \textsc{Assrt}$$

associates an assertion with each abstract predicate.

We give a denotational semantics to the assertions of definition 4 in terms of world predicates. We require assertions to be stable and thus the denotations of assertions are required to be views. To ensure the stability of the denotations we use the following auxiliary predicate:

$$\mathsf{stab}(p) \triangleq \begin{cases} p & \text{if } p \in \textsc{View} \\ \emptyset & \text{otherwise} \end{cases}$$

We extend the basic values of definition 1 with views and extend the variable stores analogously. Furthermore, to define the denotation of recursive predicates, we extend the variable stores so that predicate-recursion variables are mapped to functions from values to views.

$$\textsc{Val}_{\mathcal{A}} \triangleq \textsc{Val} \cup \textsc{View} \qquad \textsc{VarStore}_{\mathcal{A}} \triangleq (\textsc{Var} \to \textsc{Val}_{\mathcal{A}}) \uplus (\textsc{AssrtRecVars} \to (\textsc{Val}_{\mathcal{A}} \to \textsc{View}))$$

Definition 22 (Assertion Interpretation)**.** *The assertion interpretation function,* $(\!|-|\!)^- : \textsc{Assrt} \to \textsc{VarStore}_{\mathcal{A}} \to \textsc{View} \cup (\textsc{Val} \to \textsc{View})$*, maps assertions to views, or functions from values to views, within a variable store.*

$$\begin{aligned}
(\!|\mathsf{false}|\!)^\rho &\triangleq \emptyset \\
(\!|\mathsf{true}|\!)^\rho &\triangleq \sqcup\textsc{View} \\
(\!|P * Q|\!)^\rho &\triangleq (\!|P|\!)^\rho * (\!|Q|\!)^\rho \\
(\!|P \wedge Q|\!)^\rho &\triangleq (\!|P|\!)^\rho \cap (\!|Q|\!)^\rho \\
(\!|P \vee Q|\!)^\rho &\triangleq (\!|P|\!)^\rho \cup (\!|Q|\!)^\rho \\
(\!|\neg P|\!)^\rho &\triangleq (\sqcup\textsc{View} \setminus (\!|P|\!)^\rho \\
(\!|\exists x.\, P|\!)^\rho &\triangleq \bigcup_{v \in \textsc{Val}_{\mathcal{A}}} (\!|P|\!)^{\rho[x \mapsto v]} \\
(\!|\forall x.\, P|\!)^\rho &\triangleq \bigcap_{v \in \textsc{Val}_{\mathcal{A}}} (\!|P|\!)^{\rho[x \mapsto v]} \\
(\!|P \Rightarrow Q|\!)^\rho &\triangleq ((\sqcup\textsc{View} \setminus (\!|P|\!)^\rho) \cup (\!|Q|\!)^\rho \\
(\!|e \mapsto e'|\!)^\rho &\triangleq \mathsf{stab}\big(\{w \in \textsc{World} \mid h_w([\![e]\!]^\rho) = [\![e']\!]^\rho\}\big) \\
(\!|\mathbf{t}^k_\alpha(\vec{e}, e')|\!)^\rho &\triangleq \mathsf{stab}\Big(\Big\{w \in \textsc{World} \;\Big|\; r_w(\alpha) = \Big(k, \mathbf{t}, \overrightarrow{[\![e]\!]^\rho}\Big) \wedge \beta_w(\alpha) = [\![e']\!]^\rho\Big\}\Big) \\
(\!|I(\mathbf{t}^k_\alpha(\vec{e}, e'))|\!)^\rho &\triangleq (\!|I((k, \mathbf{t}, \overrightarrow{[\![e]\!]^\rho}), \alpha, [\![e']\!]^\rho)|\!)^\rho \\
(\!|[\mathsf{G}(\vec{e})]_\alpha|\!)^\rho &\triangleq \mathsf{stab}\Big(\Big\{w \in \textsc{World} \;\Big|\; \mathsf{G}\Big(\overrightarrow{[\![e]\!]^\rho}\Big) \leq \gamma_w(\alpha)\Big\}\Big) \\
(\!|\mathsf{ap}(\vec{e})|\!)^\rho &\triangleq \mathsf{stab}\Big(\Big\{w \in \textsc{World} \;\Big|\; (\mathsf{ap}, \overrightarrow{[\![e]\!]^\rho}) \in b_w\Big\}\Big) \\
(\!|I_a(\mathsf{ap}(\vec{e}))|\!)^\rho &\triangleq (\!|I_a(\mathsf{ap}, \overrightarrow{[\![e]\!]^\rho})|\!)^\rho \\
(\!|\lambda x.\, P|\!)^\rho &\triangleq \lambda v.\, (\!|P|\!)^{\rho[x \mapsto v]} \\
(\!|\mu X.\, \lambda x.\, P|\!)^\rho &\triangleq \bigsqcap \{w_f \in \textsc{Val}_{\mathcal{A}} \to \textsc{View} \mid (\!|\lambda x.\, P|\!)^{\rho[X \mapsto w_f]} \leq w_f\} \\
(\!|X|\!)^\rho &\triangleq \rho(X) \\
(\!|\mathsf{pred}(e)|\!)^\rho &\triangleq \mathsf{stab}((\!|\mathsf{pred}|\!)^\rho([\![e]\!]^\rho)) \\
(\!|e|\!)^\rho_{\mathcal{A}} &\triangleq \begin{cases} \sqcup\textsc{View} & \textit{if } [\![e]\!]^\rho = \mathsf{true} \\ [\![e]\!]^\rho & \textit{if } [\![e]\!]^\rho \in \textsc{View} \\ \emptyset & \textit{otherwise} \end{cases}
\end{aligned}$$

Note that on their own, concrete predicates are interpreted as functions from values to views.

Functions from values to views, $\textsc{Val}_{\mathcal{A}} \to \textsc{View}$, are ordered by pointwise extension of the ordering on View. Together with the following lemma, this guarantees the existence of the least fixed point for recursive predicates.

Lemma 2. *For all assertions P and recursion variables X, the function, $(\!|P|\!)^{\rho[X\mapsto -]} : (\textsc{Val}_{\mathcal{A}} \to \textsc{View}) \to \textsc{View}$, is monotonic.*

Proof. By straightforward induction over P. □

Assertions are interpreted as views. These include all the instrumentation in terms of regions, guards, atomicity tracking components and abstract predicates. We now proceed to define the means by which all of the aforementioned instrumentation is reified to concrete heaps.

Definition 23 (Region Collapse)**.** *Let $I \in \textsc{RInterp}$ be a given region interpretation. Given a region level $k \in \textsc{RLevel}$, the* region collapse *of a world $w \in \textsc{World}$, is a set of worlds given by:*

$$w{\downarrow_{k;}} \triangleq \left\{ w \circ (w', \emptyset) \;\middle|\; w' \in \circledast_{\{\alpha \;\mid\; \exists k' < k. r_w(\alpha) = (k', -, -)\}} (\!| I(r_w(\alpha), \alpha, \beta_w(\alpha)) |\!)^{\emptyset} \right\}$$

Region collapse is lifted to world predicates as expected: $p{\downarrow_{k;}} \triangleq \bigcup_{w\in p} w{\downarrow_{k;}}$.

Definition 24 (Abstract Predicate Collapse)**.** *The one-step* abstract predicate collapse *of a world is a set of given worlds given by:*

$$(r, h, b, \gamma, \beta){\downarrow_{1;}} \triangleq \left\{ (r, h, \emptyset, \gamma, \beta) \circ w \;\middle|\; w \in \circledast_{ap\in b} (\!| I_a(ap) |\!)^{\emptyset} \right\}$$

This is lifted to world predicates as expected: $p{\downarrow_{1;}} \triangleq \bigcup_{w\in p} w{\downarrow_{1;}}$. The one-step collapse gives rise to multi-step collapse: $p{\downarrow_{n+1;}} \triangleq (p{\downarrow_{n;}}){\downarrow_{1;}}$. The abstract predicate collapse of a predicate (view), applies the multi-step collapse until all abstract predicates are collapsed:

$$p{\downarrow} \triangleq \{ w \mid \exists n.\, w \in p{\downarrow_{n;}} \wedge b_w = \emptyset \}$$

The above approach to interpreting abstract predicates effectively gives a step-indexed interpretation to the predicates. The concrete interpretation of a predicate is given by the finite unfoldings of the abstract predicate collapse. If a predicate cannot be made fully concrete by finite unfoldings, then it's interpreted as false.

Definition 25 (Reification)**.** *The* reification *operation on worlds collapses the regions and the abstract predicates, and then only retains the heap component:*

$$[\![w]\!]_{k;} \triangleq \{ h_{w'} \mid w' \in w{\downarrow_{k;}}{\downarrow} \}$$

The operation is lifted to world predicates as expected: $[\![p]\!]_{k;} \triangleq \bigcup_{w\in p} [\![w]\!]_{k;}$.

2.3 Operational Semantics

We give a largely standard operational semantics for the specification language of definition 5. The semantics of atomic specification statements are defined via a state transformer on concrete states: the heaps of definition 6.

Definition 26 (Atomic Action State Transformer)**.** *Given a region level, $k \in \textsc{RLevel}$, the* atomic action state transformer, $\mathsf{a}(-,-)_k\,(-) : \textsc{View} \times \textsc{View} \to \textsc{Heap} \to \mathcal{P}(\textsc{Heap})$*, associates precondition and postcondition views to a non-deterministic state transformer:*

$$\mathsf{a}(p,q)_k\,(h) \triangleq$$
$$\left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in p * r. \\ h \in [\![w]\!]_{k;} \wedge \exists w'.\, w\; G_{k;}\; w' \wedge h' \in [\![w']\!]_{k;} \wedge w' \in q * r \end{array} \right\}$$

Given a starting heap, $h \in \textsc{Heap}$, which is contained in the reification of p composed with all possible frames, $\mathsf{a}(p,q)_k\,(h)$ returns the set of heaps that result from the reification of q composed with all possible frames, as long as the result is within the guarantee relation.

The use of a state transformer for the semantics of atomic actions is similar to the semantics of atomic actions in the Views framework [**?**]. The definition of the atomic action state transformer we have given here, corresponds to the definition of the primitive atomic satisfaction judgement in the semantics of TaDA [**?**], that defines the semantics of physically atomic actions.

View-shifts [**?**] are relations between assertions that reify to the same concrete states but may use different instrumentation. In other words, view-shifts allow us to change the view of the underlying state. Examples of view-shifts include the allocation/deallocation of shared regions and the opening/closing of abstract predicates.

Definition 27 (View Shift)**.** *View shifts are defined as follows:*

$$k; \mathcal{A} \vdash P \preccurlyeq Q \overset{def}{\iff}$$
$$\forall \rho.\, \forall r \in \textsc{View}.\, \forall w \in (\!|P|\!)^{\rho} * r.\, \forall h \in [\![w]\!]_{k;}.\, \exists w'.\, w\ G_{k;}\ w' \wedge h \in [\![w']\!]_{k;} \wedge w' \in (\!|Q|\!)^{\rho} * r$$

Lemma 3 (Implications are View Shifts)**.**

$$\frac{\mathcal{A} \vdash P \Rightarrow Q}{\mathcal{A}; k \vdash P \preccurlyeq Q}$$

Definition 28 (Operational Semantics)**.** *Let* $\lightning$ *denote fault. Let outcomes be* $o \in \textsc{Heap}^{\lightning} \triangleq \textsc{Heap} \uplus \{\lightning\}$*. Let configurations be* $\kappa \in \textsc{Configs} \triangleq \left((\mathcal{L} \times \textsc{Heap}) \cup \textsc{Heap}^{\lightning}\right)$*. The single-step operational transition relation,* $\rightsquigarrow \subseteq (\mathcal{L} \times \textsc{Heap}) \times \textsc{Configs}$*, is the smallest relation satisfying the rules:*

(1)
$$\frac{\phi, h \rightsquigarrow \phi', h'}{\phi; \psi, h \rightsquigarrow \phi'; \psi, h'}$$

(2)
$$\frac{\phi, h \rightsquigarrow h'}{\phi; \psi, h \rightsquigarrow \psi, h'}$$

(3)
$$\frac{\phi, h \rightsquigarrow \lightning}{\phi; \psi, h \rightsquigarrow \lightning}$$

(4)
$$\frac{\phi \parallel \psi, h \rightsquigarrow \kappa}{\psi \parallel \phi, h \rightsquigarrow \kappa}$$

(5)
$$\frac{\phi, h \rightsquigarrow \phi', h'}{\phi \parallel \psi, h \rightsquigarrow \phi' \parallel \psi, h'}$$

(6)
$$\frac{\phi, h \rightsquigarrow h'}{\phi \parallel \psi, h \rightsquigarrow \psi, h'}$$

(7)
$$\frac{\phi, h \rightsquigarrow \lightning}{\phi \parallel \psi, h \rightsquigarrow \lightning}$$

(8)
$$\frac{\phi_i, h \rightsquigarrow \kappa \qquad i \in \{1, 2\}}{\phi_1 \sqcup \phi_2, h \rightsquigarrow \kappa}$$

(9)
$$\frac{\forall i \in \{1, 2\}\,.\, \phi_i, h \rightsquigarrow \kappa}{\phi_1 \sqcap \phi_2, h \rightsquigarrow \kappa}$$

(10)
$$\frac{v \in \textsc{Val}_{\mathcal{A}} \qquad \phi\,[v/x]\,, h \rightsquigarrow \kappa}{\exists x.\, \phi, h \rightsquigarrow \kappa}$$

(11)
$$\frac{\phi\,[F/f]\,, h \rightsquigarrow \kappa}{\textbf{let}\ f = F\ \textbf{in}\ \phi, h \rightsquigarrow \kappa}$$

(12)
$$\frac{(\lambda x.\, \phi\,[\mu A.\, \lambda x.\, \phi/A])e, h \rightsquigarrow \kappa}{(\mu A.\, \lambda x.\, \phi)e, h \rightsquigarrow \kappa}$$

(13)
$$\frac{\phi\,[[\![e]\!]\,/x]\,, h \rightsquigarrow \kappa}{(\lambda x.\, \phi)e, h \rightsquigarrow \kappa}$$

(14)
$$\frac{h' \in \bigcup_{\vec{v} \in \overrightarrow{\textsc{Val}}} \mathsf{a}\Big((\!|P|\!)^{[\vec{x} \mapsto \vec{v}]}, (\!|Q|\!)^{[\vec{x} \mapsto \vec{v}]}\Big)_k (h)}{\forall \vec{x}.\, \langle P, Q \rangle_k, h \rightsquigarrow h'}$$

(15)
$$\frac{\forall \vec{v} \in \overrightarrow{\textsc{Val}}.\, \mathsf{a}\Big((\!|P|\!)^{[\vec{x} \mapsto \vec{v}]}, (\!|Q|\!)^{[\vec{x} \mapsto \vec{v}]}\Big)_k (h) = \emptyset}{\forall \vec{x}.\, \langle P, Q \rangle_k, h \rightsquigarrow \lightning}$$

where: $[\![e]\!]$ *denotes the denotation of the expression* e *in the empty variable store, i.e.* e *has no variables;* $\vec{v} \in \overrightarrow{\textsc{Val}}$ *denotes a vector of values; and* $[\vec{x} \mapsto \vec{v}]$ *denotes a function mapping each variable in the vector* $\vec{x}$ *to a value in the vector* $\vec{v}$*. The multi-step operational transition relation,* $\rightsquigarrow^*$*, is defined as the reflexive, transitive closure of* $\rightsquigarrow$*.*

The operational semantics is defined on closed specification programs. A specification program is closed when it has no free variables. This is largely for simplicity; variables are immutable. The operational semantics of a specification program with free variables can be defined with respect to all closing contexts.

2.4 Refinement

We say that a relatively concrete specification program ϕ, *implements* a more abstract specification program ψ, when every behaviour of ϕ is also a behaviour of ψ. Then, any client, or context, interacting with ψ can also interact with ϕ in the same way, without observing different behaviour. Formally, this is expressed as *contextual refinement*, which define in section 2.5.

Reasoning about contextual refinement involves reasoning about all possible contexts, which hinders our ability to derive useful refinement laws to include in a refinement calculus for atomicity. To overcome this difficulty, we additionally define a denotational trace semantics for specification programs, giving raise to a denotational version of refinement, in section 2.6, which we prove sound with respect to contextual refinement in section 2.7. Then, by the compositional nature of denotational semantics, we are able to justify a large selection of refinement laws in section 2.8.

2.5 Contextual Refinement

We consider standard single-holed contexts of specifications. We denote a (single-holed) specification context by C and context application by $C[\phi]$.

Definition 29 (Contextual Refinement). *Let $h \in \textsc{Heap}$.*

$$\phi \sqsubseteq_{\text{op}} \psi \iff \forall C, h, h'. \begin{cases} C[\phi], h \rightsquigarrow^* \lightning \Rightarrow C[\psi], h \rightsquigarrow^* \lightning \\ C[\phi], h \rightsquigarrow^* h' \Rightarrow C[\psi], h \rightsquigarrow^* h' \vee \ C[\psi], h \rightsquigarrow^* \lightning \end{cases}$$

Contextual refinement is given a *partial correctness* interpretation. If ϕ terminates by faulting, then ψ must do the same. On the other hand, if ϕ terminates successfully, then ψ must either successfully terminate with the same result, or fault. Faults are treated as *unspecified behaviour*. They are the most permissible of specifications; everything is a valid refinement of unspecified behaviour. Finally, if ϕ does not terminate, it is still a refinement of ψ, as long as ψ terminates. Hence, $\phi \sqsubseteq_{\text{op}} \psi$ does not guarantee termination of ϕ.

2.6 Denotational Refinement

Following the approach of Turon and Wand [**?**], the denotational model for specification programs is based on Brookes's transition trace model [**?**], adjusted to account for heaps and faults. A *transition trace* is finite sequence of pairs of heaps, (h, h'), called *moves*, possibly terminated by a fault, either due to the specification program faulting on its own accord, $(h, \lightning)$, or due to interference from the environment causing the specification program to fault, $(\lightning, \lightning)$.

Definition 30 (Transition Traces). *Single successful transitions (moves) in a trace are:* $\textsc{Move} \triangleq \textsc{Heap} \times \textsc{Heap}$. *Faulty transitions in a trace are:* $\textsc{Fault} \triangleq \textsc{Heap}^{\lightning} \times \{\lightning\}$. *Transition traces are defined by the regular language:* $\textsc{Trace} \triangleq \textsc{Move}^*; \textsc{Fault}^?$. *The empty trace is denoted by ϵ.*

We use $s, t, u \in \textsc{Trace}$ to range over traces and $S, T, U \subseteq \textsc{Trace}$ to range over sets of traces. Note that sets of traces form a lattice: the powerset lattice. Due to the existence of faults, we extend concatenation of transition traces such that an early fault causes termination.

Definition 31 (Trace Concatenation). *Let $s, t \in \textsc{Trace}$. Concatenation between traces is defined such that a fault on the left discards the trace on the right:*

$$st \triangleq \begin{cases} s & \text{if } \exists u \in \textsc{Trace}.\, s = u(h, \lightning) \vee s = u(\lightning, \lightning) \\ st & \text{otherwise} \end{cases}$$

Trace concatenation is lifted pointwise to sets of traces: $S; T \triangleq \{ st \mid s \in S \wedge t \in T \}$.

Each move in a trace corresponds to a *timeslice* of the execution of a specification program ϕ, where we observe a starting and an ending state from the operational semantics. Arbitrary interference is allowed between discrete timeslices of execution.

Definition 32 (Multi-Step Observed Traces). *The multi-step observed traces relation, $\mathcal{O}[\![-]\!] \subseteq \mathcal{L} \times \mathcal{P}(\textsc{Trace})$, is the smallest relation that satisfies the following rules:*

(16)
$$\frac{}{(\lightning, \lightning) \in \mathcal{O}[\![\phi]\!]}$$

(17)
$$\frac{\phi, h \rightsquigarrow^* o}{(h, o) \in \mathcal{O}[\![\phi]\!]}$$

(18)
$$\frac{\phi, h \rightsquigarrow^* \psi, h' \qquad t \in \mathcal{O}[\![\psi]\!]}{(h, h')t \in \mathcal{O}[\![\phi]\!]}$$

For example, a trace $(h_1, h_1')(h_2, h_2')$ for ϕ comprises two moves. The first, (h_1, h_1') is a timeslice arising from an execution $\phi, h_1 \rightsquigarrow^* \psi, h_1'$. The second, (h_2, h_2') is a timeslice arising from an execution $\psi, h_2 \rightsquigarrow^* \psi', h_2'$. In between the two timeslices, the environment executed some other specification program thus changing h_1' to h_2.

The denotational semantics, defined shortly, provide an alternative mechanism to $\mathcal{O}[\![\phi]\!]$ that is compositional on the structure of ϕ. However, to define the denotations of parallel composition, $\phi \parallel \psi$, in terms of the traces of ϕ and ψ, we need to non-deterministically interleave sets of traces.

Definition 33 (Trace Interleaving). *Let $s, t \in \textsc{Trace}$. The non-deterministic interleaving of s and t, denoted by $s \parallel t$, is the smallest set of traces that satisfies the following rules:*

(19)
$$\frac{s \in t \parallel u}{s \in u \parallel t}$$

(20)
$$\frac{s \in t \parallel u}{(h, h')s \in (h, h')t \parallel u}$$

(21)
$$\frac{}{(h, h')u \in (h, h') \parallel u}$$

(22)
$$\frac{}{(h, \lightning) \in (h, \lightning) \parallel u}$$

The interleaving is lifted pointwise to sets of traces: $T \parallel U \triangleq \{ s \in t \parallel u \mid t \in T \wedge u \in U \}$.

In the model of Brookes, the transition traces $\mathcal{O}[\![\phi]\!]$ of ϕ are closed under *stuttering* and *mumbling* [**?**]. Stuttering adds a move (h, h) to a trace, whereas mumbling merges two moves with a common midpoint. For example, $(h, h')(h', h'')$ is merged by mumbling to (h, h''). The stuttering and mumbling closures correspond to the reflexivity and transitivity of $\rightsquigarrow^*$ respectively.

Definition 34 (Trace Closure)**.** *The trace closure of a set of traces T, denoted by $T^\dagger$, is the smallest set of traces that satisfies the following rules:*

$$\frac{t \in T}{t \in T^\dagger}\ (23) \qquad \frac{st \in T^\dagger}{s(h,h)t \in T^\dagger}\ \textsc{CLStutter} \qquad \frac{s(h,h')(h',o)t \in T^\dagger}{s(h,o)t \in T^\dagger}\ \textsc{CLMumble} \qquad \frac{}{(\lightning,\lightning) \in T^\dagger}\ (24) \qquad \frac{t(h,\lightning) \in T^\dagger}{t(h,h')u \in T^\dagger}\ (25)$$

Let $f : \textsc{Val} \to \mathcal{P}(\textsc{Trace})$. Trace closure is pointwise extended to functions from values to sets of traces: $f^\dagger \triangleq \lambda v.\, f(v)^\dagger$.

The last two rules regarding faults were added to the closure of Brookes [**?**] by Turon and Wand [**?**]. Intuitively, rule (24) captures the fact that the environment of a specification program ϕ, may cause it to fault at any given time. The rule (25) captures the fact that faulting behaviour is permissive: a specification program that terminates with a fault after a trace t, can always be implemented by a specification program that continues after t.

The denotational semantics of specification programs are defined as sets of (closed) traces. We extend the variable stores used for assertions, so that function and recursion variables are mapped to functions from values to sets of traces.

$$\textsc{VarStore}_\mu \triangleq \textsc{VarStore}_{\mathcal{A}} \uplus ((\textsc{FuncVars} \uplus \textsc{RecVars}) \to (\textsc{Val}_{\mathcal{A}} \to \mathcal{P}(\textsc{Trace})))$$

Definition 35 (Denotational Semantics)**.** *The denotational semantics of specification programs are given by the function, $[\![-]\!]^- : \textsc{VarStore}_\mu \to \mathcal{L} \to \mathcal{P}(\textsc{Trace})$, mapping specification programs to sets of traces, within a variable environment.*

$$\begin{aligned}
[\![\phi; \psi]\!]^\rho &\triangleq ([\![\phi]\!]^\rho \,;\, [\![\psi]\!]^\rho)^\dagger \\
[\![\phi \parallel \psi]\!]^\rho &\triangleq ([\![\phi]\!]^\rho \parallel [\![\psi]\!]^\rho)^\dagger \\
[\![\phi \sqcup \psi]\!]^\rho &\triangleq ([\![\phi]\!]^\rho \cup [\![\psi]\!]^\rho)^\dagger \\
[\![\phi \sqcap \psi]\!]^\rho &\triangleq ([\![\phi]\!]^\rho \cap [\![\psi]\!]^\rho)^\dagger \\
[\![\exists x.\, \phi]\!]^\rho &\triangleq \left(\bigcup_{v \in \textsc{Val}_{\mathcal{A}}} [\![\phi]\!]^{\rho[x \mapsto v]} \right)^\dagger \\
[\![\textbf{let } f = F \textbf{ in } \phi]\!]^\rho &\triangleq [\![\phi]\!]^{\rho[f \mapsto [\![F]\!]^\rho]} \\
[\![Fe]\!]^\rho &\triangleq [\![F]\!]^\rho\, [\![e]\!]^\rho \\
[\![f]\!]^\rho &\triangleq \rho(f)^\dagger \\
[\![A]\!]^\rho &\triangleq \rho(A)^\dagger \\
[\![\mu A.\, \lambda x.\, \phi]\!]^\rho &\triangleq \bigcap \left\{ T_f \in \textsc{Val}_{\mathcal{A}} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; T_f = {T_{f'}}^\dagger \wedge [\![\lambda x.\, \phi]\!]^{\rho[A \mapsto T_{f'}]} \subseteq {T_{f'}}^\dagger \right\} \\
[\![\lambda x.\, \phi]\!]^\rho &\triangleq \lambda v.\, [\![\phi]\!]^{\rho[x \mapsto v]} \\
[\![\forall \vec{x}.\, \langle P, Q \rangle_k]\!]^\rho &\triangleq \left(\begin{array}{l} \left\{ (h, h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x} \mapsto \vec{v}]}, (\!|Q|\!)^{\rho[\vec{x} \mapsto \vec{v}]} \right)_k (h) \right\} \\ \cup \left\{ (h, \lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l} \vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x} \mapsto \vec{v}]}, (\!|Q|\!)^{\rho[\vec{x} \mapsto \vec{v}]} \right)_k (h) = \emptyset \\ \wedge\ (\!|Q|\!)^{\rho[\vec{x} \mapsto \vec{v}]} \neq \emptyset \end{array} \right\} \end{array} \right)^\dagger
\end{aligned}$$

The semantics is relatively straightforward. Sequential composition is concatenation and parallel composition is non-deterministic interleaving (of closed traces). Angelic and demonic choice are union and intersection respectively. These correspond to the join and meet of the lattice of trace sets (the powerset lattice) respectively. Existential quantification is the standard set union over all values.

For recursive functions we use the Tarskian least fixed point. Note that functions from values to trace sets, $\textsc{Val}_{\mathcal{A}} \to \mathcal{P}(\textsc{Trace})$, are ordered by the pointwise extension of the ordering on $\mathcal{P}(\textsc{Trace})$. Furthermore, the $\bigcap$

and $\subseteq$ in the fixed-point definition correspond to the meet and partial order of the lattice arising from the pointwise extension of $\mathcal{P}(\textsc{Trace})$ to the function space $\textsc{Val} \to \mathcal{P}(\textsc{Trace})$. Together with the following monotonicity lemma, this guarantees the existence of the fixed point.

Lemma 4. *Let $x_f \in \textsc{FuncVars} \uplus \textsc{RecVars}$, $\phi \in \mathcal{L}$ and $\rho \in \textsc{VarStore}_\mu$. The function $[\![\phi]\!]^{\rho[x_f \mapsto -]} : (\textsc{Val} \to \mathcal{P}(\textsc{Trace})) \to \mathcal{P}(\textsc{Trace})$ is monotonic.*

Proof. By straightforward induction over ϕ. Base cases A, f trivial. Inductive cases follow directly from the induction hypothesis. □

The denotational semantics are defined in terms of sets of finite traces. A finite trace is always terminated either by the implementation itself, or by a fault caused by the environment. Infinite traces are discarded. Consider the denotations of the specification program $(\mu A.\, \lambda x.\, Ax)\,()$. By the least fixpoint and rule (24) of the trace closure (definition 34), the only finite trace for this specification program is $(\lightning, \lightning)$. The denotational semantics of infinite recursion are finite traces that terminate with a fault caused by the environment.

With the denotational semantics defined, we can give a denotational version of refinement based on the partial order of trace sets.

Definition 36 (Denotational Refinement)**.** *$\phi \sqsubseteq_{\text{den}} \psi$ iff, for all closing ρ, $[\![\phi]\!]^\rho \subseteq [\![\psi]\!]^\rho$.*

Throughout this dissertation, unless explicitly stated, we use denotational refinement, writing $\phi \sqsubseteq \psi$ to mean $\phi \sqsubseteq_{\text{den}} \psi$.

2.7 Adequacy

We relate the denotational and operational versions of refinement in two steps. First, we establish the following lemma, showing that denotational semantics produce the same closed trace sets as the operational semantics. The proof details are given appendix A.

Lemma 5. *If ϕ is closed, then $[\![\phi]\!]^\emptyset = (\mathcal{O}[\![\phi]\!])^\dagger$.*

Proof. From corollary 5 established in appendix A. □

Second, with the following theorem we establish that the denotational refinement is a sound approximation of contextual refinement. We are not interested in establishing completeness; all the refinement laws in the subsequent sections are justified by the denotational semantics.

Theorem 1 (Adequacy)**.** *If $\phi \sqsubseteq_{\text{den}} \psi$, then $\phi \sqsubseteq_{\text{op}} \psi$.*

Proof. Let $\phi \sqsubseteq_{\text{den}} \psi$. Let C be a context that closes both ϕ and ψ. We write $C[\phi]$ and $C[\psi]$ for the closed specifications under C. Then, by definition 36, $[\![C[\phi]]\!]^\emptyset \subseteq [\![C[\psi]]\!]^\emptyset$. By lemma 5, $(\mathcal{O}[\![C[\phi]]\!])^\dagger \subseteq (\mathcal{O}[\![C[\psi]]\!])^\dagger$.

- Let $C[\phi], h \rightsquigarrow^* \lightning$. By rule (17) (def. 32), $(h, \lightning) \in \mathcal{O}[\![C[\phi]]\!]$. By definition 34, $(h, \lightning) \in (\mathcal{O}[\![C[\phi]]\!])^\dagger$ and thus $(h, \lightning) \in (\mathcal{O}[\![C[\psi]]\!])^\dagger$. Then by definition 34, either $(h, \lightning) \in \mathcal{O}[\![C[\psi]]\!]$ or there exist h', h'', s such that $(h, h')s(h'', \lightning) \in \mathcal{O}[\![C[\psi]]\!]$. In both cases, by definition 32, we get that $C[\psi], h \rightsquigarrow^* \lightning$ and thus $C[\psi], h \lightning$.
- Let $C[\phi], h \rightsquigarrow^* h'$. By rule (17) (def. 32), $(h, h') \in \mathcal{O}[\![C[\phi]]\!]$. By definition 34, $(h, h') \in (\mathcal{O}[\![C[\phi]]\!])^\dagger$ and thus $(h, h') \in (\mathcal{O}[\![C[\psi]]\!])^\dagger$. Then, by definition 34, we have the following cases:

  i). $(h, h') \in \mathcal{O}[\![C[\psi]]\!]$

  ii). there exist h'', h''', s such that $(h, h'')s(h''', h') \in \mathcal{O}[\![C[\psi]]\!]$

  iii). $(h, \lightning) \in \mathcal{O}[\![C[\psi]]\!]$

  iv). there exist h'', h''', s such that $(h, h'')s(h''', \lightning) \in \mathcal{O}[\![C[\psi]]\!]$

  Cases i) and ii), by definition 32 give that $C[\psi], h \rightsquigarrow^* h'$.

  Cases iii) and iv), by definition 32 give that $C[\psi], h \rightsquigarrow^* \lightning$.

  Therefore, $C[\psi], h \rightsquigarrow^* h' \vee\ C[\psi], h \rightsquigarrow^* \lightning$.

□

2.8 Refinement Laws

Having defined the semantics of refinement in the previous section, we now state the refinement laws that comprise our refinement calculus for reasoning about concurrency. We distinguish the refinement laws into two broad groups: general refinement laws about our core specification language from definition 5, and refinement laws specific to atomic specification statements.

First we define three atomic specifications statements that server as limits of our refinement calculus and behave as the unit of composition operators.

Definition 37 (Primitive specifications)**.**

$$\texttt{abort} \triangleq \langle \mathsf{false}, \mathsf{true} \rangle_k$$
$$\texttt{miracle} \triangleq \langle \mathsf{true}, \mathsf{false} \rangle_k$$
$$\texttt{skip} \triangleq \langle \mathsf{true}, \mathsf{true} \rangle_k$$

The `abort` statement can always faults, since its precondition is never satisfied. It is the most permissive of specifications and serves as the top element in the partial order of refinement. Semantically, it is the set of all possible traces. On the other hand, `miracle` never faults, as its precondition is always satisfied, but also never takes any steps as its postcondition is never satisfied. Semantically, `miracle` does nothing; modulo the closure of definition 34, it is the empty trace ϵ. It is a valid implementation of any specification and serves as the bottom element in the partial order of refinement. Finally, `skip` does not fault, but also does not modify the heap. Note that since the semantics of assertions is intuitionistic, `true` denotes the empty heap. Thus, `skip` acts as the identity of sequential and parallel composition, as well as angelic and demonic choice.

Definition 38 (General Refinement Laws)**.**

Refl
$\phi \sqsubseteq \phi$

Trans
$$\frac{\phi \sqsubseteq \psi' \qquad \psi' \sqsubseteq \psi}{\phi \sqsubseteq \psi}$$

AntiSymm
$$\frac{\phi \sqsubseteq \psi \qquad \psi \sqsubseteq \phi}{\phi \equiv \psi}$$

Skip
$\texttt{skip}; \phi \equiv \phi \equiv \phi; \texttt{skip}$

Assoc
$\phi; (\psi_1; \psi_2) \equiv (\phi; \psi_1); \psi_2$

MinMax
$\texttt{miracle} \sqsubseteq \phi \sqsubseteq \texttt{abort}$

EElim
$\phi\,[e/x] \sqsubseteq \exists x.\, \phi$

EIntro
$$\frac{x \notin free(\phi)}{\exists x.\, \phi \sqsubseteq \phi}$$

AChoiceEq
$\phi \sqcup \phi \equiv \phi$

AChoiceId
$\phi \sqcup \texttt{miracle} \equiv \phi$

AChoiceAssoc
$\phi \sqcup (\psi_1 \sqcup \psi_2) \equiv (\phi \sqcup \psi_1) \sqcup \psi_2$

AChoiceComm
$\phi \sqcup \psi \equiv \psi \sqcup \phi$

AChoiceElim
$\phi \sqsubseteq \phi \sqcup \psi$

AChoiceDstR
$(\phi_1 \sqcup \phi_2); \psi \equiv (\phi_1; \psi) \sqcup (\phi_2; \psi)$

AChoiceDstL
$\psi; (\phi_1 \sqcup \phi_2) \equiv (\psi; \phi_1) \sqcup (\psi; \phi_2)$

DChoiceEq
$\phi \sqcap \phi \equiv \phi$

DChoiceId
$\phi \sqcap \texttt{abort} \equiv \phi$

DChoiceAssoc
$\phi \sqcap (\psi_1 \sqcap \psi_2) \equiv (\phi \sqcap \psi_1) \sqcap \psi_2$

DChoiceComm
$\phi \sqcap \psi \equiv \psi \sqcap \phi$

DChoiceElim
$$\frac{\phi \sqsubseteq \psi_1 \qquad \phi \sqsubseteq \psi_2}{\phi \sqsubseteq \psi_1 \sqcap \psi_2}$$

DChoiceIntro
$\phi \sqcap \psi \sqsubseteq \phi$

DChoiceDstR
$(\phi_1 \sqcap \phi_2); \psi \equiv (\phi_1; \psi) \sqcap (\phi_2; \psi)$

DChoiceDstL
$\psi; (\phi_1 \sqcap \phi_2) \equiv (\psi; \phi_1) \sqcap (\psi; \phi_2)$

AChoiceDstD
$\phi \sqcup (\psi_1 \sqcap \psi_2) \equiv (\phi \sqcup \psi_1) \sqcap (\phi \sqcup \psi_2)$

DChoiceDstA
$\phi \sqcap (\psi_1 \sqcup \psi_2) \equiv (\phi \sqcap \psi_1) \sqcup (\phi \sqcap \psi_2)$

Absorb
$\phi \sqcup (\phi \sqcap \psi) \equiv \phi \equiv \phi \sqcap (\phi \sqcup \psi)$

Demonise
$\phi \sqcap \psi \sqsubseteq \phi \sqcup \psi$

ParSkip
$\phi \parallel \texttt{skip} \equiv \phi$

ParAssoc
$\phi \parallel (\psi_1 \parallel \psi_2) \equiv (\phi \parallel \psi_1) \parallel \psi_2$

ParComm
$\phi \parallel \psi \equiv \psi \parallel \phi$

Exchange
$(\phi \parallel \psi); (\phi' \parallel \psi') \sqsubseteq (\phi; \phi') \parallel (\psi; \psi')$

AChoiceExchange
$(\phi \parallel \psi) \sqcup (\phi' \parallel \psi') \sqsubseteq (\phi \sqcup \phi') \parallel (\psi \sqcup \psi')$

SeqPar
$\phi; \psi \sqsubseteq \phi \parallel \psi$

ParDstLR
$\phi; (\psi_1 \parallel \psi_2) \sqsubseteq (\phi; \psi_1) \parallel \psi_2$

ParDstLL
$\phi; (\psi_1 \parallel \psi_2) \sqsubseteq \psi_1 \parallel (\phi; \psi_2)$

ParDstRL
$(\phi \parallel \psi_1); \psi_2 \sqsubseteq \phi \parallel (\psi_1; \psi_2)$

ParDstRR
$(\phi \parallel \psi_1); \psi_2 \sqsubseteq (\phi; \psi_2) \parallel \psi_1$

EAChoiceEq
$\exists x.\, \phi \equiv \bigsqcup_{v \in \mathrm{Val}} \phi\,[v/x]$

ESeqExt
$$\frac{x \notin free(\phi)}{\exists x.\, \phi; \psi \equiv \phi; \exists x.\, \psi}$$

ESeqDst
$\exists x.\, \phi; \psi \sqsubseteq \exists x.\, \phi; \exists x.\, \psi$

EAChoiceDst
$\exists x.\, \phi \sqcup \psi \equiv (\exists x.\, \phi) \sqcup (\exists x.\, \psi)$

EDChoiceDst
$\exists x.\, \phi \sqcap \psi \equiv (\exists x.\, \phi) \sqcap (\exists x.\, \psi)$

EParDst
$\exists x.\, \phi \parallel \psi \sqsubseteq (\exists x.\, \phi) \parallel (\exists x.\, \psi)$

CMono
$$\frac{\phi \sqsubseteq \psi}{C[\phi] \sqsubseteq C[\psi]}$$

FApplyElim
$\phi\,[e/x] \equiv (\lambda x.\, \phi)\, e$

FApplyElimRec
$\phi\,[(\mu A.\, \lambda x.\, \phi)\,/A]\,[e/x] \equiv (\mu A.\, \lambda x.\, \phi)\, e$

FElim
$F_l \equiv \lambda x.\, F_l x$

FRename
$$\frac{\phi\,[e_1/x] \sqsubseteq \phi\,[e_2/x]}{(\lambda x.\, \phi)\, e_1 \sqsubseteq (\lambda x.\, \phi)\, e_2}$$

FRenameRec
$$\frac{\phi\,[(\mu A.\, \lambda x.\, \phi)\,/A]\,[e_1/x] \sqsubseteq \phi\,[(\mu A.\, \lambda x.\, \phi)\,/A]\,[e_2/x]}{(\mu A.\, \lambda x.\, \phi)\, e_1 \sqsubseteq (\mu A.\, \lambda x.\, \phi)\, e_2}$$

FuncIntro
$$\frac{x \notin free(\phi)}{(\lambda x.\, \phi)\, () \equiv \phi}$$

Inline
$\phi\,[F/f] \equiv \textbf{let } f = F \textbf{ in } \phi$

Ind
$$\frac{\lambda x.\, \phi\,[\psi/A] \sqsubseteq \lambda x.\, \psi}{\mu A.\, \lambda x.\, \phi \sqsubseteq \lambda x.\, \psi}$$

UnrollR
$$\frac{A \notin free(\phi) \cup free(\psi)}{(\mu A.\, \lambda x.\, \psi \sqcup \phi; Ae')\, e \equiv \psi\,[e/x] \sqcup \phi\,[e/x]\,;(\mu A.\, \lambda x.\, \psi \sqcup \phi; Ae')\, e'\,[e/x]}$$

UnrollL
$$\frac{A \notin free(\phi) \cup free(\psi)}{(\mu A.\, \lambda x.\, \psi \sqcup Ae'; \phi)\, e \equiv \psi\,[e/x] \sqcup (\mu A.\, \lambda x.\, \psi \sqcup Ae'; \phi)\, e'\,[e/x]\,;\phi\,[e/x]}$$

RecSeq
$$\frac{A \notin free(\phi) \cup free(\psi_1) \cup free(\psi_2)}{(\mu A.\, \lambda x.\, \psi_1 \sqcup \phi; Ae')\, e; \psi_2\,[e/x] \equiv (\mu A.\, \lambda x.\, \psi_1; \psi_2 \sqcup \phi; Ae')\, e}$$

Many of the refinement laws in definition 38 are familiar from the literature. From left to right, top to bottom, the laws Refl, Trans and AntiSymm reflect the fact that refinement is a partial order. Skip and Assoc state that `skip` is the unit of sequential composition and that sequential composition is associative respectively. The MinMax law defines `miracle` and `abort` as the top and bottom specifications in the partial order of refinement as discussed earlier.

The next two laws concern existential quantification. EElim allows elimination of the quantifier during refinement, by replacing the quantified variable with an expression. Typically, we name refinement laws according with respect to refinement; that is, as if reading the law right to left. Conversely, the EIntro refinement law allows the introduction of an existentially quantified variable.

The next block of refinement laws are about angelic and demonic choice, most of which correspond to the laws of boolean algebra, with the join operator being angelic choice, the meet operator being demonic choice, the bottom element being `miracle` and the top element being `abort`. In fact, the partial order of refinement forms a complete, boolean and atomic lattice. The AChoiceElim refinement law captures the intuition behind the angelic non-deterministic choice: we can choose to refine the choice to one of the two components. On the other hand, the analogous law for demonic choice, DChoiceElim, states that the refinement of a demonic choice must be a refinement of both components. This law is analogous to the conjunction rule of Hoare logic.

Next, is a set of laws regarding parallel composition. The most important refinement law of this block is Exchange, originating from Hoare's algebraic laws [**?**]. The SeqPar law, as well as the subsequent distributivity laws can be derived from Exchange, ParComm and ParSkip.

In the next set of laws we return to existential quantification. The ESeqExt refinement law allows us to increase or decrease the scope of the existentially quantified variable. The rest of the laws for existential quantification concert its distributivity in sequential, non-deterministic choice and parallel composition.

The CMono refinement law is obvious, and the most pervasively used law in refinement derivations. It reflects the fact that denotational refinement is contextual refinement, as shown in theorem 1.

The next block of refinement laws is about functions. FApplyElim allows the elimination of a function application by replacing the argument variable with the argument passed to the function. FElim allows us to eliminate indirect function applications. The FRename and FRenameRec allow the refinement of a function application to a different argument, for non-recursive and recursive functions respectively. The FuncIntro law allows the introduction of a function application and Inline allows function definitions to be inlined at the point of application.

The Ind refinement law is standard fixpoint induction.

Finally, the UnrollR and UnrollL laws allow us to do loop unrolling on recursive specification programs. Both rules are useful in derivations of refinements between recursive specifications.

We justify the soundness of our refinement laws by denotational refinement, which in turn is sound with respect to contextual refinement. Transitively, the refinement laws are also sound with respect to contextual refinement.

Theorem 2 (Soundness of General Refinement Laws)**.** *The general refinement laws of definition 38 are sound.*

Proof. By appendix B. □

Apart from the general refinement laws, we also define a set of refinement laws atomic specification statements.

Definition 39 (Refinement Laws for Atomic Specification Statements.)**.**

AUElim
$$\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall x, \vec{x}.\, \langle P, Q\rangle_k$$

AEarly
$$\frac{x \notin free(P)}{\exists x.\, \forall \vec{y}.\, \langle P, Q\rangle_k \equiv \forall \vec{y}.\, \langle P, \exists x.\, Q\rangle_k}$$

AEElim
$$\forall \vec{y}, x.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{y}.\, \langle \exists x.\, P, \exists x.\, Q\rangle_k$$

ADisjunction
$$\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcup \forall \vec{x}.\, \langle P_2, Q_2\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P_1 \vee P_2, Q_1 \vee Q_2\rangle_k$$

AConjunction
$$\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcap \forall \vec{x}.\, \langle P_2, Q_2\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P_1 \wedge P_2, Q_1 \wedge Q_2\rangle_k$$

AFrame
$$\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P * R, Q * R\rangle_k$$

AStutter
$$\forall \vec{x}.\, \langle P, P\rangle_k ; \forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k$$

AMumble
$$\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, P'\rangle_k ; \forall \vec{x}.\, \langle P', Q\rangle_k$$

AInterleave
$$\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \parallel \forall \vec{x}.\, \langle P_2, Q_2\rangle_k \equiv (\forall \vec{x}.\, \langle P_1, Q_1\rangle_k ; \forall \vec{x}.\, \langle P_2, Q_2\rangle_k) \sqcup (\forall \vec{x}.\, \langle P_2, Q_2\rangle_k ; \forall \vec{x}.\, \langle P_1, Q_1\rangle_k)$$

ACons
$$\frac{\forall \vec{x}.\, P \preccurlyeq P' \qquad \forall \vec{x}.\, Q' \preccurlyeq Q}{\forall \vec{x}.\, \langle P', Q'\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k}$$

AChoice
$$\forall \vec{x}.\, \langle P, Q \vee Q'\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k \sqcup \forall \vec{x}.\, \langle P, Q'\rangle_k$$

AUseAtomic
$$\frac{\forall x.\, (x, f(x)) \in \mathcal{T}_{\mathbf{t}}(G)^*}{\begin{array}{l}\forall x, \vec{x}.\, \left\langle I(\mathbf{t}^k_\alpha(\vec{e}, x)) * P(\vec{x}) * [\mathsf{G}]_\alpha, I(\mathbf{t}^k_\alpha(\vec{e}, f(x))) * Q(\vec{x})\right\rangle_k \\ \sqsubseteq \forall x, \vec{x}.\, \left\langle \mathbf{t}^k_\alpha(\vec{e}, x) * P(\vec{x}) * [\mathsf{G}]_\alpha, \mathbf{t}^k_\alpha(\vec{e}, f(x)) * Q(\vec{x})\right\rangle_{k+1}\end{array}}$$

ARLevel
$$\frac{k_1 \leq k_2}{\forall \vec{x}.\, \langle P, Q\rangle_{k_1} \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_{k_2}}$$

The refinement laws stated here have an implicit side condition that requires assertions on both sides of $\sqsubseteq$ are stable.

The AUElim refinement law allows us to refine an atomic specification statement in which a variable is explicitly universally quantified, to an atomic specification in which the variable is free, and thus implicitly universally quantified in the context. The effect is that of turning a variable that is locally bound in the specification statement, to a global variable. The AEarly states that late choice, in the existential quantification in the postcondition, can be refined to early choice. AEarly together with ESeqExt from definition 38, allow us to treat the existential quantifier similarly to the scope extrusion laws of π-calculus. With the AEElim law we can eliminate existential quantification analogously to the existential elimination rule of Hoare logic. The ADisjunction and AConjunction refinement laws are analogous to the conjunction and disjunction rules of Hoare logic. The AFrame refinement law is directly analogous to the frame rule of separation logic [**?**].

The AStutter and AMumble refinement laws are due to the trace closure of definition 34, and originate from Brookes's trace semantics [**?**]. Stuttering reflects the fact that a specification is unable to observe steps of a refinement that do not modify the state. On the other hand, mumbling reflects the fact that a sequence of atomic steps can be implemented by a single atomic step. Note that by setting P' to be P in Mumble we obtain an equivalence for stuttering. The AInterleave states that a parallel composition of two atomic specification statements is observationally equivalent to their interleavings.

The ACons refinement law is directly analogous to the consequence rule of Hoare logic, using view shifts in lieu of implications. We can abstract a specification statement by strengthening the precondition and weakening the postcondition.

So far the refinement laws in definition 39 are due to either Hoare logic or separation logic. They are also present in the refinement calculus of Turon and Wand [**?**] or appear to be admissible by the semantics of their specification language. The next set of laws, however, come from our use of the simplified TaDA model. The AUseAtomic allows us to replace a shared region in an atomic specification statement with its interpretation. We

require ownership of the guard by which the atomic update is allowed for that region in the state transition system. ARLEVEL reflects the fact that we can refine a specification statement of a higher region level to a specification statement of lower region level.

Theorem 3 (Soundness of Atomic Specification Statement Refinement Laws)**.** *The atomic refinement laws of definition 39 are sound.*

Proof. By appendix C. □

2.9 Hybrid Specification Statement

In the previous sections we have defined our specification language and refinement calculus for concurrency. The basic statement of our language is the atomic specification statement, $\forall \vec{x}.\, \langle P, Q \rangle_k$. We now define an encoding of hybrid specification statements that combine atomic and non-atomic effects.

Notation

We often write specification programs that use inline functions: $(\lambda x.\, \phi)\, e$ or $(\mu A.\, \lambda x.\, \phi)\, e$. Several times, the function body, ϕ, may be relatively large. To increase readability in these cases, we use line breaks and whitespace instead of parenthesis to distinguish between the inline function and its application. For example, for a relatively large function body $\phi_{\text{large}}; \psi_{\text{huge}}$, we will write:

$$\begin{array}{l} \mu A.\, \lambda x.\ \phi_{\text{large}}; \\ \qquad\qquad \psi_{\text{huge}} \\ \cdot e \end{array}$$

to mean $(\mu A.\, \lambda x.\, \phi_{\text{large}}; \psi_{\text{huge}})\, e$.

Definition 40 (Hybrid Specification Statement)**.** *We define the* hybrid specification statement *in terms of atomic specification statements as follows:*

$$\begin{array}{l} \forall \vec{x}.\, \exists \vec{y}.\, \{P'\} \langle P(\vec{x}), Q(\vec{x}, \vec{y}) \rangle \{Q'(\vec{x}, \vec{y})\}_k \triangleq \\ \quad \exists p.\, \forall \vec{x} \in \overrightarrow{X}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x}) \rangle_k; \\ \quad \mu A.\, \lambda p.\ \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), p' * P(\vec{x}) \rangle_k; Ap' \\ \qquad \sqcup\ \exists \vec{x}, \vec{y}.\, \exists p''.\ \langle p * P(\vec{x}), p'' * Q(\vec{x}, \vec{y}) \rangle_k; \\ \qquad\qquad\qquad \mu B.\, \lambda p''.\ \exists p'''.\, \langle p'', p''' \rangle_k; Bp''' \\ \qquad\qquad\qquad\qquad \sqcup\ \langle p'', Q'(\vec{x}, \vec{y}) \rangle_k \\ \qquad\qquad\qquad \cdot p'' \\ \quad \cdot p \end{array}$$

The first atomic specification statement solely serves to capture the states satisfied by the private precondition P' into the variable p, so that it can be passed as an argument to the subsequent recursive function. This is a silent atomic step. Indeed, since it does not change the state before the step that immediately follows, by ASTUTTER, the first atomic specification statement is not observable. Furthermore, by ACONS followed by AFRAME the first primitive atomic statement is refined into `skip`, and thus, by SKIP, does not have to implemented at all.

The pattern of definition 40, where we use a silent atomic read to capture the states satisfied by an assertion into a variable, which is then passed as an argument to a function, appears every time we prove various refinements for hybrid specification statements. The following lemma demonstrates that this step is indeed silent and is useful for several of the refinement proofs about atomic specification statements.

Lemma 6 (Assertions as Function Arguments)**.** *When Fe, with p free, does not occur within ϕ, then:*

$$\exists p.\, \forall \vec{x}.\, \langle P, P \wedge p\rangle_k; (\lambda p.\, \forall \vec{x}.\, \langle p, Q\rangle_k; \phi)\, p \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k; \phi\,[P/p]$$

$$\begin{aligned}\exists p.\, \forall \vec{x}.\, \langle P, P \wedge p\rangle_k; (\lambda p.\, \forall \vec{x}.\, \langle p, Q_1\rangle_k; \phi \sqcup \forall \vec{x}.\, \langle p, Q_2\rangle_k; \psi)\, p \\ \sqsubseteq \forall \vec{x}.\, \langle P, Q_1\rangle_k; \phi\,[P/p] \sqcup \forall \vec{x}.\, \langle P, Q_2\rangle_k; \psi\,[P/p]\end{aligned}$$

$$\exists p.\, \forall \vec{x}.\, \langle P, P \wedge p\rangle_k; (\mu A.\, \lambda p.\, \forall \vec{x}.\, \langle p, Q\rangle_k; \phi)\, p \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k; \phi\,[P/p]\,[(\mu A.\, \lambda p.\, \forall \vec{x}.\, \langle p, Q\rangle_k; \phi)/A]$$

$$\begin{aligned}\exists p.\, \forall \vec{x}.\, \langle P, P \wedge p\rangle_k; (\mu A.\, \lambda p.\, \forall \vec{x}.\, \langle p, Q_1\rangle_k; \phi \sqcup \forall \vec{x}.\, \langle p, Q_2\rangle_k; \psi)\, p \\ \sqsubseteq \quad \forall \vec{x}.\, \langle P, Q_1\rangle_k; \phi\,[P/p]\,[(\mu A.\, \lambda p.\, \forall \vec{x}.\, \langle p, Q_1\rangle_k; \phi \sqcup \forall \vec{x}.\, \langle p, Q_2\rangle_k; \psi)/A] \\ \sqcup\ \forall \vec{x}.\, \langle P, Q_2\rangle_k; \psi\,[P/p]\,[(\mu A.\, \lambda p.\, \forall \vec{x}.\, \langle p, Q_1\rangle_k; \phi \sqcup \forall \vec{x}.\, \langle p, Q_2\rangle_k; \psi)/A]\end{aligned}$$

Proof. The first refinement is proven by application of FApplyElim, CMono, AStutter and finally EIntro. The second refinement is proven similarly, with AChoiceDstL before AStutter. The next two refinements are proven in the same way as the first two, except FApplyElimRec is used instead of FApplyElim. □

The general form of the atomic specification statement is a generalisation not only of an atomic update, but also of non-atomic updates, from which we derive a few important specification statements.

Definition 41 (Derived Specification Statements)**.** *The following specification statements are defined as special cases of the atomic specification statement.*

- $\{P,\ Q\}_k \triangleq \forall y \in \mathbf{1}.\, \exists z \in \mathbf{1}.\, \{P\}\langle \mathsf{true}, \mathsf{true}\rangle\{Q\}_k$
- $[P]_k \triangleq \{\mathsf{true},\ P\}_k$
- $\{P\}_k \triangleq \{P,\ P\}_k$

The most important of the derived statements, is the *Hoare specification statement* of the form $\{P,\ Q\}$, which specifies an update from a state satisfying the precondition P, to a state satisfying the postcondition Q, without any atomicity guarantees. Intuitively, it stands for any program that satisfies the Hoare triple $\{P\} - \{Q\}$. The other two derived statements are the *assumption* statement of the form, $[P]$, and the *assertion* statement, $\{P\}$.

According to definitions 41 and 40, the Hoare specification statement, $\{P,\ Q\}$, is a sequence of atomic steps, where the first begins in state P and the last ends in state Q. However, the recursive function part of definition 40 is more complex that what is intuitively necessary for Hoare specification statements. Fortunately, by the following lemma, we show that definition 40, when applied to Hoare specification statements, is observable as much simpler pattern.

Lemma 7 (Hoare Specification Statement Refinement)**.**

$$\begin{aligned}\{P,\ Q\}_k \equiv \exists p.\, \langle P, P \wedge p\rangle_k; \\ \mu A.\lambda p.\ \exists p'.\, \langle p, p'\rangle_k; Ap' \\ \sqcup\ \langle p, Q\rangle_k \\ \cdot p\end{aligned}$$

Proof. We demonstrate a refinement between $\{P,\ Q\}_k$, as given by definition 40, and the simpler form, in both directions. First, we show that:

$$\begin{aligned}\{P,\ Q\}_k \sqsubseteq \exists p.\, \langle P, P \wedge p\rangle_k; \\ \mu A.\lambda p.\ \exists p'.\, \langle p, p'\rangle_k; Ap' \\ \sqcup\ \langle p, Q\rangle_k \\ \cdot p\end{aligned}$$

$\{P,\ Q\}_k \equiv$ by definitions 41 and 40

$\exists p.\ \langle P, P \wedge p\rangle_k;$
$\mu A.\lambda p.\ \exists p'.\ \langle p, p'\rangle_k; Ap'$
$\sqcup\ \exists p''.\ \langle p, p''\rangle_k;$
$\mu B.\lambda p''.\ \exists p'''.\ \langle p'', p'''\rangle_k; Bp'''$
$\sqcup\ \langle p'', Q\rangle_k$
$\cdot p''$
$\cdot p$

$\sqsubseteq$ by Ind and CMono, where the following establishes the premiss

begin with substitute A and α-convert

$\exists p'.\ \forall \vec{x}.\ \langle p, p'\rangle_k; \mu A.\lambda p.\ \exists p''.\ \langle p, p''\rangle_k; Ap''$
$\sqcup\ \langle p, Q\rangle_k$
$\cdot p'$
$\sqcup\ \exists p'.\ \forall \vec{x}.\ \langle p, p'\rangle_k;$
$\mu A.\lambda p.\ \exists p''.\ \langle p, p''\rangle_k; Ap''$
$\sqcup\ \langle p, Q\rangle_k$
$\cdot p'$

$\equiv$ by AChoiceEq

$\exists p'.\ \forall \vec{x}.\ \langle p, p'\rangle_k; \mu A.\lambda p.\ \exists p''.\ \langle p, p''\rangle_k; Ap''$
$\sqcup\ \langle p, Q\rangle_k$
$\cdot p'$

$\sqsubseteq$ by AChoiceElim and CMono

$\forall \vec{x}.\ \langle p, Q\rangle_k$
$\sqcup\ \exists p'.\ \forall \vec{x}.\ \langle p, p'\rangle_k; \mu A.\lambda p.\ \exists p''.\ \langle p, p''\rangle_k; Ap''$
$\sqcup\ \langle p, Q\rangle_k$
$\cdot p'$

$\equiv$ by AChoiceComm and UnrollR

$\mu A.\lambda p.\ \exists p'.\ \langle p, p'\rangle_k; Ap'$
$\sqcup\ \langle p, Q\rangle_k$
$\cdot p$

$\sqsubseteq \exists p.\ \langle P, P \wedge p\rangle_k;$
$\mu A.\lambda p.\ \exists p'.\ \langle p, p'\rangle_k; Ap'$
$\sqcup\ \langle p, Q\rangle_k$
$\cdot p$

Now we show that:

$\exists p.\ \langle P, P \wedge p\rangle_k;$
$\mu A.\lambda p.\ \exists p'.\ \langle p, p'\rangle_k; Ap'$
$\sqcup\ \langle p, Q\rangle_k$
$\cdot p$
$\sqsubseteq \{P,\ Q\}_k$

$\exists p.\ \langle P, P \wedge p\rangle_k;$
$\mu A.\lambda p.\ \exists p'.\ \langle p, p'\rangle_k; Ap'$
$\sqcup\ \langle p, Q\rangle_k$
$\cdot p$

$\sqsubseteq$ AMumble, EElim, Ind and CMono

$\exists p.\ \langle P, P \wedge p\rangle_k;$
$\mu A.\lambda p.\ \exists p'.\ \langle p, p'\rangle_k; Ap'$
$\sqcup\ \exists p''.\ \langle p, p''\rangle_k;$
$\mu B.\lambda p''.\ \exists p'''.\ \langle p'', p'''\rangle_k; Bp'''$
$\sqcup\ \langle p'', Q\rangle_k$
$\cdot p''$
$\cdot p$

$\equiv$ by definitions 41 and 40

$\{P,\ Q\}_k$

□

We define refinement laws for hybrid atomic specification statements, most of which are directly lifted from the laws for atomic specification statements.

Definition 42 (Hybrid Atomic Refinement Laws)**.**

HUElim
$$\forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(x,\vec{x}), Q(x,\vec{x},\vec{y})\rangle\left\{Q'(x,\vec{x},\vec{y})\right\}_k \sqsubseteq \forall x,\vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(x,\vec{x}), Q(x,\vec{x},\vec{y})\rangle\left\{Q'(x,\vec{x},\vec{y})\right\}_k$$

HEarly
$$\frac{x \notin \mathtt{free}(P') \cup \mathtt{free}(P) \qquad y \notin \mathtt{free}(P') \cup \mathtt{free}(P)}{\exists y.\, \exists x.\, \forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k \equiv \forall \vec{x}.\, \exists y, \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), \exists x.\, Q(\vec{x},y,\vec{y})\rangle\left\{Q'(\vec{x},y,\vec{y})\right\}_k}$$

HEElim
$$\forall x,\vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(x,\vec{x}), Q(x,\vec{x},\vec{y})\rangle\left\{P'(x,\vec{x},\vec{y})\right\}_k \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle \exists x.\, P(x,\vec{x}), \exists x.\, Q(x,\vec{x},\vec{y})\rangle\left\{\exists x.\, P'(x,\vec{x},\vec{y})\right\}_k$$

HDisjunction
$$\begin{aligned}&\forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k \sqcup \forall \vec{x}.\, \exists \vec{y}.\, \left\{P''\right\}\langle P'(\vec{x}), Q'(\vec{x},\vec{y})\rangle\left\{Q''(\vec{x},\vec{y})\right\}_k\\ &\quad\sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P' \vee P''\right\}\langle P(\vec{x}) \vee P'(\vec{x}), Q(\vec{x},\vec{y}) \vee Q'(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y}) \vee Q''(\vec{x},\vec{y})\right\}_k\end{aligned}$$

HConjunction
$$\begin{aligned}&\forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k \sqcap \forall \vec{x}.\, \exists \vec{y}.\, \left\{P''\right\}\langle P'(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k\\ &\quad\sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P' \wedge P''\right\}\langle P(\vec{x}) \wedge P'(\vec{x}), Q(\vec{x},\vec{y}) \wedge Q'(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y}) \wedge Q''(\vec{x},\vec{y})\right\}_k\end{aligned}$$

HFrame
$$\forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P' * R'\right\}\langle P(\vec{x}) * R(\vec{x}), Q(\vec{x},\vec{y}) * R(\vec{x})\rangle\left\{Q'(\vec{x},\vec{y}) * R'\right\}_k$$

HStutter
$$\forall \vec{x}.\, \left\{P'\right\}\langle P(\vec{x}), P(\vec{x})\rangle\left\{P''\right\}_k ; \forall \vec{x}.\, \exists \vec{y}.\, \left\{P''\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k$$

HMumble
$$\forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), P'(\vec{x})\rangle\left\{P''\right\}_k ; \forall \vec{x}.\, \exists \vec{y}.\, \left\{P''\right\}\langle P'(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k$$

HInterleave
$$\begin{aligned}&\forall \vec{x}.\, \exists \vec{y}.\, \{I\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{I\}_k \parallel \forall \vec{x}.\, \exists \vec{y}.\, \{I\}\langle P(\vec{x})', Q(\vec{x},\vec{y})'\rangle\{I\}_k\\ \sqsubseteq\ &\left(\forall \vec{x}.\, \exists \vec{y}.\, \{I\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{I\}_k ; \forall \vec{x}.\, \exists \vec{y}.\, \{I\}\langle P(\vec{x})', Q(\vec{x},\vec{y})'\rangle\{I\}_k\right) \sqcup \left(\forall \vec{x}.\, \exists \vec{y}.\, \{I\}\langle P(\vec{x})', Q(\vec{x},\vec{y})'\rangle\{I\}_k ; \forall \vec{x}.\, \exists \vec{y}.\, \{I\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{I\}_k\right)\end{aligned}$$

HStrengthen
$$\forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P' * P(\vec{x}), Q(\vec{x},\vec{y}) * Q'(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P' * P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y}) * Q'(\vec{x},\vec{y})\right\}_k$$

Atomic
$$\forall \vec{x}.\, \left\langle P' * P(\vec{x}), \exists \vec{y}.\, Q'(\vec{x},\vec{y}) * Q(\vec{x},\vec{y})\right\rangle_k \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k$$

HCons
$$\frac{P' \preccurlyeq P'' \qquad \forall \vec{x}.\, P(\vec{x}) \preccurlyeq P'(\vec{x}) \qquad \forall \vec{x},\vec{y}.\, Q''(\vec{x},\vec{y}) \preccurlyeq Q'(\vec{x},\vec{y}) \qquad \forall \vec{x},\vec{y}.\, Q'(\vec{x},\vec{y}) \preccurlyeq Q(\vec{x},\vec{y})}{\forall \vec{x}.\, \exists \vec{y}.\, \left\{P''\right\}\langle P'(\vec{x}), Q'(\vec{x},\vec{y})\rangle\left\{Q''(\vec{x},\vec{y})\right\}_k \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k}$$

HUseAtomic
$$\begin{aligned}&\frac{\forall x.\, (x, f(x)) \in \mathcal{T}_{\mathbf{t}}(\mathsf{G})^*}{\forall x,\vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\left\langle I(\mathbf{t}_\alpha^k(\vec{e},x)) * P(x,\vec{x}) * \left[\mathsf{G}(\vec{e}')\right]_\alpha, I(\mathbf{t}_\alpha^k(\vec{e},f(x))) * Q(x,\vec{x},\vec{y})\right\rangle\left\{Q'(\vec{x},\vec{y})\right\}_k}\\ &\quad\equiv \forall x,\vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\left\langle \mathbf{t}_\alpha^k(\vec{e},x) * P(\vec{x}) * \left[\mathsf{G}(\vec{e}')\right]_\alpha, \mathbf{t}_\alpha^k(\vec{e},f(x)) * Q(\vec{x},\vec{y})\right\rangle\left\{Q'(\vec{x},\vec{y})\right\}_{k+1}\end{aligned}$$

HRLevel
$$\frac{k_1 \le k_2}{\forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_{k_1} \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \left\{P'\right\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\left\{Q'(\vec{x},\vec{y})\right\}_{k_2}}$$

The refinement laws stated here have an implicit side condition that requires assertions on both sides of $\sqsubseteq$ are stable.

Since the hybrid specification statement is just a program in out core specification language, the refinement laws on hybrid statements are simply proven as refinements between specification programs, by using the general laws of our refinement calculus (definition 38) and the laws for atomic specification statements (definition 39).

Theorem 4 (Soundness of Abstract Atomicity Refinement Laws)**.** *The refinement laws for abstract atomicity in definition 42 are sound.*

Proof. By appendix D. □

We complete the development of refinement laws in this section, with laws for Hoare specification statements. At this point, none of these laws should be surprising.

Definition 43 (Hoare Specification Statement Refinement Laws)**.**

$$\textsc{Seq}\ \frac{\phi \sqsubseteq \{P,\, R\}_k \qquad \psi \sqsubseteq \{R,\, Q\}_k}{\phi;\psi \sqsubseteq \{P,\, Q\}_k} \qquad \textsc{Disjunction}\ \frac{\phi \sqsubseteq \{P_1,\, Q_1\}_k \qquad \psi \sqsubseteq \{P_2,\, Q_2\}_k}{\phi \sqcup \psi \sqsubseteq \{P_1 \vee P_2,\, Q_1 \vee Q_2\}_k} \qquad \textsc{Conjunction}\ \frac{\phi \sqsubseteq \{P_1,\, Q_1\}_k \qquad \psi \sqsubseteq \{P_2,\, Q_2\}_k}{\phi \sqcap \psi \sqsubseteq \{P_1 \wedge P_2,\, Q_1 \wedge Q_2\}_k}$$

$$\textsc{Parallel}\ \frac{\phi \sqsubseteq \{P_1,\, Q_1\}_k \qquad \psi \sqsubseteq \{P_2,\, Q_2\}_k}{\phi \parallel \psi \sqsubseteq \{P_1 * P_2,\, Q_1 * Q_2\}_k} \qquad \textsc{Frame}\ \{P,\, Q\}_k \sqsubseteq \{P * R,\, Q * R\}_k \qquad \textsc{EElim}\ \{P,\, Q\}_k \sqsubseteq \{\exists y.\, P,\, \exists y.\, Q\}_k$$

$$\textsc{Early}\ \frac{x \notin free(P)}{\exists x.\, \{P,\, Q\}_k \equiv \{P,\, \exists x.\, Q\}_k} \qquad \textsc{Hybrid}\ \begin{array}{l} \forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\ \sqsubseteq \exists \vec{x}.\ \{P' * P(\vec{x}),\, \exists \vec{y}.\, Q'(\vec{x},\vec{y}) * Q(\vec{x},\vec{y})\}_k \end{array} \qquad \textsc{Cons}\ \frac{P \preccurlyeq P' \qquad Q' \preccurlyeq Q}{\{P',\, Q'\}_k \sqsubseteq \{P,\, Q\}_k}$$

$$\textsc{RLevel}\ \frac{k_1 \leq k_2}{\{P,\, Q\}_{k_1} \sqsubseteq \{P,\, Q\}_{k_2}}$$

The Seq, Disjunction and Conjunction refinement laws directly correspond to the sequence, disjunction and conjunction rules of Hoare logic. Parallel directly corresponds to the parallel rule of separation logic [**?**]. The Frame law corresponds to the frame rule of separation logic [**?**], and is a direct consequence of AFrame. EElim allows existential quantification elimination as the analogous rule in Hoare logic. The Hybrd refinement law allows a non-atomic update defined by the Hoare specification statement, to be implemented atomically by a hybrid specification statement. It is a direct consequence of HStrengthen when the entire atomic component is moved to the non-atomic component. The Cons law, is the consequence rule of Hoare logic for Hoare specification statements, albeit using view-shifts. Finally, RLevel allows the same as ARLevel for Hoare specification statements.

Theorem 5 (Soundness of Hoare Specification Statement Refinement Laws)**.** *The refinement laws for Hoare specification statements, in definition 43, are sound.*

Proof. By appendix E. □

A Adequacy Addendum

In section 2.7, we have defined the semantics of our specification language and refinement both in terms of operational and denotational semantics. With theorem 1, we have established the soundness of denotational refinement with respect to contextual refinement based on the operational semantics. The proof of this theorem relies on lemma 5, which equates the traces obtained by the denotational semantics to the traces obtained by the operational semantics under the stuttering, mumbling and faulting closure.

To prove lemma 5, we establish inequality between operational and denotational traces in both directions. In appendix A.1, we prove that operational traces are contained within denotational traces and in appendix A.2 we prove the reverse. As a stepping stone, in both directions, we will work with *raw* traces, that are not closed by the stuttering, mumbling and faulting closure. This simplifies the proof process by avoiding the need for mumbling and stuttering, not only for lemma 5, but also for the refinement laws.

Before we proceed with the proof, we define the raw denotational semantics, and establish several crucial lemmas.

Definition 44 (Raw Denotational Semantics)**.** *The raw denotational semantics,* $\mathcal{R}[\![-]\!]^- : \textsc{VarStore}_\mu \to \mathcal{L} \to \mathcal{P}(\textsc{Trace})$*, map specification programs to sets of traces, within a variable environment.*

$$
\begin{aligned}
\mathcal{R}[\![\phi; \psi]\!]^\rho &\triangleq \mathcal{R}[\![\phi]\!]^\rho \,;\, \mathcal{R}[\![\psi]\!]^\rho \\
\mathcal{R}[\![\phi \parallel \psi]\!]^\rho &\triangleq \mathcal{R}[\![\phi]\!]^\rho \parallel \mathcal{R}[\![\psi]\!]^\rho \\
\mathcal{R}[\![\phi \sqcup \psi]\!]^\rho &\triangleq \mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi]\!]^\rho \\
\mathcal{R}[\![\phi \sqcap \psi]\!]^\rho &\triangleq \mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\psi]\!]^\rho \\
\mathcal{R}[\![\exists x.\, \phi]\!]^\rho &\triangleq \bigcup_{v \in \textsc{Val}} \mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]} \\
\mathcal{R}[\![\textbf{let } f = F \textbf{ in } \phi]\!]^\rho &\triangleq \mathcal{R}[\![\phi]\!]^{\rho[f \mapsto \mathcal{R}[\![F]\!]^\rho]} \\
\mathcal{R}[\![Fe]\!]^\rho &\triangleq \mathcal{R}[\![F]\!]^\rho\, [\![e]\!]^\rho \\
\mathcal{R}[\![f]\!]^\rho &\triangleq \rho(f) \\
\mathcal{R}[\![A]\!]^\rho &\triangleq \rho(A) \\
\mathcal{R}[\![\mu A.\, \lambda x.\, \phi]\!]^\rho &\triangleq \bigcap \left\{ T_f \in \textsc{Val} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; \mathcal{R}[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto T_f]} \subseteq T_f \right\} \\
\mathcal{R}[\![\lambda x.\, \phi]\!]^\rho &\triangleq \lambda v.\, \mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]} \\
\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_k]\!]^\rho &\triangleq \left\{ (h, h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\big((\!|P|\!)^{\rho[\vec{x} \mapsto \vec{v}]}, (\!|Q|\!)^{\rho[\vec{x} \mapsto \vec{v}]}\big)_k (h) \right\} \\
&\quad \cup \left\{ (h, \lightning) \in \textsc{Heap}^\lightning \;\middle|\; \begin{array}{l} \vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\big((\!|P|\!)^{\rho[\vec{x} \mapsto \vec{v}]}, (\!|Q|\!)^{\rho[\vec{x} \mapsto \vec{v}]}\big)_k (h) = \emptyset \\ \quad \wedge\, (\!|Q|\!)^{\rho[\vec{x} \mapsto \vec{v}]} \neq \emptyset \end{array} \right\} \\
&\quad \cup \left\{ (\lightning, \lightning) \right\}
\end{aligned}
$$

The argument for the existence of the least fixpoint is the same as for the denotation semantics of definition 35.

Lemma 8. $(\lightning, \lightning) \in \mathcal{R}[\![\phi]\!]^\rho$

Proof. Straightforward induction on ϕ. $(\lightning, \lightning) \in \mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_k]\!]^\rho$ by definition 44. All inductive cases follow immediately from the inductive hypothesis. □

Lemma 9 (Function and Recursion Substitution)**.** *If* ψ *is closed, then* $\mathcal{R}[\![\phi]\!]^{\rho[y \mapsto \mathcal{R}[\![\psi]\!]^\rho]} = \mathcal{R}[\![\phi\,[\psi/y]]\!]^\rho$*, where* y *is a recursion variable* A*, or a function variable* f*.*

Proof. Straightforward induction on ϕ. Base case $\mathcal{R}[\![A]\!]$ trivial. Base case $\mathcal{R}[\![f]\!]$ trivial. Base Case $\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_k]\!]$ trivial, as recursion and function variables or not free in P or Q. Inductive cases follow immediately from the induction hypothesis. □

Lemma 10 (Variable Substitution)**.** *If* e *is an expression, where* x *is not free, then* $\mathcal{R}[\![\phi]\!]^{\rho[x \mapsto [\![e]\!]^\rho]} = \mathcal{R}[\![\phi\,[[\![e]\!]^\rho\,/x]]\!]^\rho$*.*

Proof. Straightforward induction on ϕ. □

The semantics of recursion are given as the Tarskian least fixpoint. However, in some proof steps, the Kleenian least fixpoint is more useful. In order to switch to the Kleenian fixpoint we require continuity.

Lemma 11 (Raw Denotation Continuity)**.** $\mathcal{R}[\![\phi]\!]^{\rho[A\mapsto -]}$ *is Scott-continuous.*

Proof. $\mathcal{R}[\![\phi]\!]^{\rho[A\mapsto -]} : (\textsc{Val} \to \mathcal{P}(\textsc{Trace})) \to \mathcal{P}(\textsc{Trace})$.
Let $D \subseteq \textsc{Val} \to \mathcal{P}(\textsc{Trace})$. D is a directed subset of $\textsc{Val} \to \mathcal{P}(\textsc{Trace})$, due to the fact that $\textsc{Val} \to \mathcal{P}(\textsc{Trace})$ is a lattice by pointwise extension of the powerset lattice.
For Scott-continuity we show that: $\sqcup(\mathcal{R}[\![\phi]\!]^{\rho[A\mapsto -]})[D] = \mathcal{R}[\![\phi]\!]^{\rho[A\mapsto \sqcup D]}$ by induction on ϕ.
Base case: Ae.

$$
\begin{aligned}
\sqcup(\mathcal{R}[\![Ae]\!]^{\rho[A\mapsto -]})[D] &= \bigcup_{T_f \in D} \mathcal{R}[\![Ae]\!]^{\rho[A\mapsto T_f]} \\
&= \text{by definition 44} \\
&\quad \bigcup_{T_f \in D} T_f e \\
&= \text{by pointwise extension of the powerset lattice} \\
&\quad \left(\bigsqcup_{T_f \in D} T_f \right) e \\
&= (\sqcup D)\, e \\
&= \text{by definition 44} \\
&\quad \mathcal{R}[\![Ae]\!]^{\rho[A\mapsto \sqcup D]}
\end{aligned}
$$

Base cases $fe, \forall \vec{x}.\, \langle P, Q\rangle_k$ not applicable as the recursion variable A does not appear in these cases.
Cases $\phi;\psi$, $\phi \parallel \psi$, $\phi \sqcup \psi$ and $\phi \sqcap \psi$ from induction hypothesis and by the fact that $;$, $\parallel$, $\cup$ and $\cap$ preserve continuity respectively.
All other cases follow straightforwardly from the inductive hypothesis.

□

The next three lemmas establish properties of the stuttering, mumbling and faulting closure that we rely on in several proof steps.

Lemma 12 (Closure operator)**.** $-^\dagger$ *is a closure operator:*

$$
\begin{aligned}
T \subseteq T^\dagger \quad & (-^\dagger \textit{ is extensive)} \\
T \subseteq U \Rightarrow T^\dagger \subseteq U^\dagger \quad & (-^\dagger \textit{ is increasing)} \\
(T^\dagger)^\dagger = T^\dagger \quad & (-^\dagger \textit{ is idempotent)}
\end{aligned}
$$

Proof. **Idempotent:** $T^\dagger \subseteq (T^\dagger)^\dagger$ follows directly from rule (23).
We show that $(T^\dagger)^\dagger \subseteq T^\dagger$ by induction on the derivation of $t \in T^\dagger$.
Base cases:
Rule (24). $(\lightning,\lightning) \in T^{\dagger\dagger}$ and $(\lightning,\lightning) \in T^\dagger$.
Rule (23). Let $t \in T^{\dagger\dagger}$. By premiss, $t \in T^\dagger$.
Inductive cases:
Rule CLStutter. Let $s(h,h)t \in T^{\dagger\dagger}$. By premiss, $st \in T^{\dagger\dagger}$. By the inductive hypothesis, $st \in T^\dagger$, thus $s(h,h)t \in T^\dagger$.
Rule CLMumble. Let $s(h,o)t \in T^{\dagger\dagger}$. By premiss, $s(h,h')(h',o)t \in T^{\dagger\dagger}$. By the inductive hypothesis, $s(h,h')(h',o)t \in T^\dagger$, thus also $s(h,o)t \in T^\dagger$.
Rule (25). Let $t(h,h')u \in T^{\dagger\dagger}$. By premiss, $t(h,\lightning) \in T^{\dagger\dagger}$. By the inductive hypothesis, $t(h,\lightning) \in T^\dagger$, thus also $t(h,h')u \in T^\dagger$.
Increasing: By induction on the derivation of $t \in T^\dagger$.
Base case:

Rule (24). $(\wr,\wr) \in T^\dagger \Rightarrow (\wr,\wr) \in U^\dagger$ holds trivially.
Rule (23). Let $t \in T^\dagger$. By premiss, $t \in T$. By assumption, $t \in U$. Then, by rule (23), $t \in U^\dagger$.
Inductive cases:
Rule CLStutter. Let $s(h,h)t \in T^\dagger$. By premiss, $st \in T^\dagger$. By the induction hypothesis, $st \in U^\dagger$, from which it follows that $s(h,h)t \in U^\dagger$.
Rule CLMumble. Let $s(h,o)t \in T^\dagger$. By premiss, $s(h,h')(h',o)t \in T^\dagger$. By the induction hypothesis, $s(h,h')(h',o)t \in U^\dagger$, from which it follows that $s(h,o)t \in U^\dagger$.
Rule (25). Let $t(h,h')u \in T^\dagger$. By premiss, $t(h,\wr) \in T^\dagger$. By the induction hypothesis, $t(h,\wr) \in U^\dagger$, from which it follows that $t(h,h')u \in U^\dagger$.
Extensive: $\forall t.\, t \in T$, by rule (23), $t \in T^\dagger$. □

Lemma 13 (Trace Closure Distributivity)**.**

1. $T^\dagger; U^\dagger \subseteq (T;U)^\dagger$
2. $T^\dagger \parallel U^\dagger \subseteq (T \parallel U)^\dagger$
3. $\bigcup({T_i}^\dagger) \subseteq (\bigcup T_i)^\dagger$
4. $\bigcap({T_i}^\dagger) \supseteq (\bigcap T_i)^\dagger$

Proof. (1): First we show that $T^\dagger; U \subseteq (T;U)^\dagger$ by induction on the derivation of $t \in T^\dagger$. Fix $u \in U$.
Base cases:
Rule (24). $(\wr,\wr) \in T^\dagger; U^\dagger \Rightarrow (\wr,\wr) \in (T;U)^\dagger$ holds trivially.
Rule 23. Let $t \in T^\dagger$. By premiss, $t \in T$. By trace concatenation, $tu \in T;U$. Then, by rule 23, $tu \in (T;U)^\dagger$.
Inductive cases:
Rule CLStutter. Let $s(h,h)t \in T^\dagger$. By premiss, $st \in T^\dagger$. By trace concatenation, $stu \in T^\dagger; U$. Then, by the induction hypothesis, $stu \in (T;U)^\dagger$, from which it follows that $s(h,h)tu \in (T;U)^\dagger$.
Rule CLMumble. Let $s(h,o)t \in T^\dagger$. By premiss, $s(h,h')(h',o)t \in T^\dagger$. By trace concatenation, $s(h,h')(h',o)tu \in T^\dagger; U$. Then, by the induction hypothesis, $s(h,h')(h',o)tu \in (T;U)^\dagger$, from which it follows that $s(h,o)t \in (T;U)^\dagger$.
Rule (25). Let $t(h,h')u \in T^\dagger$. By premiss, $t(h,\wr) \in T^\dagger$. By trace concatenation, $t(h,\wr)u = t(h,\wr) \in T^\dagger; U$. Then, by the induction hypothesis, $t(h,\wr)u \in (T;U)^\dagger$, from which it follows that $t(h,h')u \in (T;U)^\dagger$.
Furthermore, $T; U^\dagger \subseteq (T;U)^\dagger$ by induction on the derivation of $u \in U^\dagger$ similarly.
Then, it follows that: $T^\dagger; U^\dagger \subseteq (T; U^\dagger)^\dagger \subseteq ((T;U)^\dagger)^\dagger$. Then, by idempotence $T^\dagger; U^\dagger \subseteq (T;U)^\dagger$.
(2): Similarly to (1).
(3): Fix index I, $n \in I$. By induction on the derivation of $t \in {T_n}^\dagger$.
Base case:
Rule (24). $(\wr,\wr) \in \bigcup({T_i}^\dagger) \iff (\wr,\wr) \in (\bigcup T_i)^\dagger$ holds trivially.
Inductive case:
Rule (23). Let $t \in {T_n}^\dagger$, then $t \in \bigcup({T_i}^\dagger)$. Then, by the induction hypothesis, $t \in (\bigcup T_i)^\dagger$.
Rule CLStutter. Let $s(h,h)t \in \bigcup({T_i}^\dagger)$. By premiss, $st \in \bigcup({T_i}^\dagger)$. Then, by the induction hypothesis, $st \in (\bigcup T_i)^\dagger$, from which it follows that $s(h,h)t \in (\bigcup T_i)^\dagger$.
Rule CLMumble. Let $s(h,o)t \in \bigcup({T_i}^\dagger)$. By premiss, $s(h,h')(h',o)t \in \bigcup({T_i}^\dagger)$. Then, by the induction hypothesis, $s(h,h')(h',o)t \in \bigcup (T_i)^\dagger$, from which it follows that $s(h,o)t \in \bigcup (T_i)^\dagger$.
Rule (25). Let $t(h,h')u \in \bigcup({T_i}^\dagger)$. By premiss, $t(h,\wr) \in \bigcup({T_i}^\dagger)$. Then, by the induction hypothesis, $t(h,\wr) \in \bigcup (T_i)^\dagger$, from which it follows $t(h,h')u \in \bigcup (T_i)^\dagger$.
(4): Similarly to (3). □

Lemma 14. $\left(\bigcap\left({T_i}^\dagger\right)\right)^\dagger = \bigcap\left({T_i}^\dagger\right)$

Proof. By lemma 12 (extensive): $\bigcap\left({T_i}^\dagger\right) \subseteq \left(\bigcap\left({T_i}^\dagger\right)\right)^\dagger$.
By lemma 13: $\bigcap\left({T_i}^{\dagger\dagger}\right) \supseteq \left(\bigcap\left({T_i}^\dagger\right)\right)^\dagger$.
By lemma 12 (idempotence): $\bigcap\left({T_i}^\dagger\right) \supseteq \left(\bigcap\left({T_i}^\dagger\right)\right)^\dagger$. □

The following lemma reflects the fact that the denotational semantics are idempotent with respect to trace closure.

Lemma 15. $(\llbracket\phi\rrbracket^\rho)^\dagger = \llbracket\phi\rrbracket^\rho$

Proof. Straightforward induction on ϕ using lemma 12.
Base case: $\forall\vec{x}.\,\langle P, Q\rangle_k$

$$
\begin{aligned}
(\llbracket\forall\vec{x}.\,\langle P,Q\rangle_k\rrbracket^\rho)^\dagger &= \left(\left(\begin{array}{l}\left\{(h,h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h)\right\} \\ \cup \left\{(h,\lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l}\vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h) = \emptyset \\ \wedge\, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \neq \emptyset\end{array}\right\}\end{array}\right)^\dagger\right)^\dagger \\
&= \text{by lemma 12 (idempotence)} \\
&\quad \left(\begin{array}{l}\left\{(h,h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h)\right\} \\ \cup \left\{(h,\lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l}\vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h) = \emptyset \\ \wedge\, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \neq \emptyset\end{array}\right\}\end{array}\right)^\dagger \\
&= \llbracket\forall\vec{x}.\,\langle P,Q\rangle_k\rrbracket^\rho
\end{aligned}
$$

Base case: A

$$
\begin{aligned}
(\llbracket A\rrbracket^\rho)^\dagger &= \left(\rho(A)^\dagger\right)^\dagger \\
&= \text{by lemma 12 (idempotence)} \\
&\quad \rho(A)^\dagger = \llbracket A\rrbracket^\rho
\end{aligned}
$$

Base case: f, same as A
Case: $\phi;\psi$

$$
\begin{aligned}
(\llbracket\phi;\psi\rrbracket^\rho)^\dagger &= \left((\llbracket\phi\rrbracket^\rho ; \llbracket\psi\rrbracket^\rho)^\dagger\right)^\dagger \\
&= \text{by lemma 12} \\
&\quad \llbracket\phi;\psi\rrbracket^\rho
\end{aligned}
$$

Case: $\phi \parallel \psi$ as previous.
Case: $\phi \sqcup \psi$

$$
\begin{aligned}
(\llbracket\phi \sqcup \psi\rrbracket^\rho)^\dagger &= \left((\llbracket\phi\rrbracket^\rho \cup \llbracket\psi\rrbracket^\rho)^\dagger\right)^\dagger \\
&= \text{by lemma 12} \\
&\quad \llbracket\phi \sqcup \psi\rrbracket^\rho
\end{aligned}
$$

Case: $\phi \sqcap \psi$

$$
\begin{aligned}
(\llbracket\phi \sqcap \psi\rrbracket^\rho)^\dagger &= \left((\llbracket\phi\rrbracket^\rho \cap \llbracket\psi\rrbracket^\rho)^\dagger\right)^\dagger \\
&= \text{by lemma 12} \\
&\quad \llbracket\phi \sqcap \psi\rrbracket^\rho
\end{aligned}
$$

Case: $\exists x.\,\phi$

$$
\begin{aligned}
(\llbracket \phi \rrbracket^\rho)^\dagger &= \left(\left(\bigcup_v \llbracket \phi \rrbracket^{\rho[x \mapsto v]} \right)^\dagger \right)^\dagger \\
&= \text{by lemma 12} \\
&\quad \llbracket \exists x.\, \phi \rrbracket^\rho
\end{aligned}
$$

Case: $\mu A.\, \lambda x.\, \phi$

$$
\begin{aligned}
(\llbracket \mu A.\, \lambda x.\, \phi \rrbracket^\rho)^\dagger &= \left(\bigcap \left\{ T_f \in \textsc{Val} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; \left(\llbracket \lambda x.\, \phi \rrbracket^{\rho[A \mapsto T_f]} \right)^\dagger \subseteq {T_f}^\dagger \right\} \right)^\dagger \\
&= \text{by lemma 14} \\
&\quad \bigcap \left\{ T_f \in \textsc{Val} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; \left(\llbracket \lambda x.\, \phi \rrbracket^{\rho[A \mapsto T_f]} \right)^\dagger \subseteq {T_f}^\dagger \right\} \\
&= \llbracket \mu A.\, \lambda x.\, \phi \rrbracket^\rho
\end{aligned}
$$

Case: $\mathbf{let}\ f = F\ \mathbf{in}\ \phi$

$$
\begin{aligned}
(\llbracket \mathbf{let}\ f = F\ \mathbf{in}\ \phi \rrbracket^\rho)^\dagger &= \left(\llbracket \phi \rrbracket^{\rho[f \mapsto \llbracket F \rrbracket^\rho]} \right)^\dagger \\
&= \text{by induction hypothesis} \\
&\quad \llbracket \phi \rrbracket^{\rho[f \mapsto \llbracket F \rrbracket^\rho]} \\
&= \llbracket \mathbf{let}\ f = F\ \mathbf{in}\ \phi \rrbracket^\rho
\end{aligned}
$$

Case: $\lambda x.\, \phi$

$$
\begin{aligned}
(\llbracket \lambda x.\, \phi \rrbracket^\rho)^\dagger &= \left(\lambda v.\, \llbracket \phi \rrbracket^{\rho[x \mapsto v]} \right)^\dagger \\
&= \text{by induction hypothesis} \\
&\quad \lambda v.\, \llbracket \phi \rrbracket^{\rho[x \mapsto v]} \\
&= \llbracket \lambda x.\, \phi \rrbracket^\rho
\end{aligned}
$$

Case: Fe

$$
\begin{aligned}
(\llbracket Fe \rrbracket^\rho)^\dagger &= (\llbracket F \rrbracket^\rho\, \llbracket e \rrbracket^\rho)^\dagger \\
&= \text{by induction hypothesis} \\
&\quad \llbracket F \rrbracket^\rho\, \llbracket e \rrbracket^\rho \\
&= \llbracket Fe \rrbracket^\rho
\end{aligned}
$$

□

The raw denotational semantics produce the denotational semantics by adding the trace closure, as indicated by the following lemma.

Lemma 16. $\llbracket \phi \rrbracket^\rho = (\mathcal{R} \llbracket \phi \rrbracket^\rho)^\dagger$.

Proof. First, we establish $(\mathcal{R} \llbracket \phi \rrbracket^\rho)^\dagger \subseteq \llbracket \phi \rrbracket^\rho$.
Trivially, $\mathcal{R} \llbracket \phi \rrbracket^\rho \subseteq \llbracket \phi \rrbracket^\rho$.
By lemma 12 (increasing), $(\mathcal{R} \llbracket \phi \rrbracket^\rho)^\dagger \subseteq (\llbracket \phi \rrbracket^\rho)^\dagger$.
By lemma 15, $(\mathcal{R} \llbracket \phi \rrbracket^\rho)^\dagger \subseteq \llbracket \phi \rrbracket^\rho$.
Second, we establish $(\mathcal{R} \llbracket \phi \rrbracket^\rho)^\dagger \supseteq \llbracket \phi \rrbracket^\rho$, by induction on ϕ.

Base case: $(\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho)^\dagger \supseteq [\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho$ by the definitions.
Base case: A. $\mathcal{R}[\![A]\!]^{\rho\dagger} \supseteq \rho(A)^\dagger \supseteq [\![A]\!]^\rho$.
Base case: f, as previous.
Case: $\phi; \psi$.

$$\begin{aligned}
(\mathcal{R}[\![\phi; \psi]\!]^\rho)^\dagger &= (\mathcal{R}[\![\phi]\!]^\rho \,;\mathcal{R}[\![\psi]\!]^\rho)^\dagger \\
&\supseteq \text{by lemma 13} \\
&\quad (\mathcal{R}[\![\phi]\!]^\rho)^\dagger; (\mathcal{R}[\![\psi]\!]^\rho)^\dagger \\
&\supseteq \text{by inductive hypothesis} \\
&\quad [\![\phi]\!]^\rho \,; [\![\psi]\!]^\rho \\
&\supseteq \text{by lemma 12 (increasing on both sides, idempotence on left of } \supseteq) \\
&\quad ([\![\phi]\!]^\rho \,; [\![\psi]\!]^\rho)^\dagger \\
&= \text{by definition 35} \\
&\quad [\![\phi; \psi]\!]^\rho
\end{aligned}$$

Case: $\phi \parallel \phi$, similar to previous.
Case: $\phi \sqcup \psi$

$$\begin{aligned}
(\mathcal{R}[\![\phi \sqcup \psi]\!]^\rho)^\dagger &= (\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi]\!]^\rho)^\dagger \\
&= \text{by lemma 12} \\
&\quad \left((\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi]\!]^\rho)^\dagger\right)^\dagger \\
&\supseteq \text{by lemma 13} \\
&\quad \left((\mathcal{R}[\![\phi]\!]^\rho)^\dagger \cup (\mathcal{R}[\![\psi]\!]^\rho)^\dagger\right)^\dagger \\
&\supseteq \text{by inductive hypothesis} \\
&\quad ([\![\phi]\!]^\rho \cup [\![\psi]\!]^\rho)^\dagger \\
&= \text{by definition 35} \\
&\quad [\![\phi \sqcup \psi]\!]^\rho
\end{aligned}$$

Case: $\phi \sqcap \psi$

$$\begin{aligned}
[\![\phi \sqcap \psi]\!]^\rho &= ([\![\phi]\!]^\rho \cap [\![\psi]\!]^\rho)^\dagger \\
&\subseteq \text{by lemma 13} \\
&\quad ([\![\phi]\!]^\rho)^\dagger \cap ([\![\psi]\!]^\rho)^\dagger \\
&= \text{by lemma 12} \\
&\quad [\![\phi]\!]^\rho \cap [\![\psi]\!]^\rho \\
&\subseteq \text{by inductive hypothesis} \\
&\quad (\mathcal{R}[\![\phi]\!]^\rho)^\dagger \cap (\mathcal{R}[\![\psi]\!]^\rho)^\dagger \\
&\subseteq \text{by lemma 13} \\
&\quad (\mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\psi]\!]^\rho)^\dagger \\
&= \text{by definition 44} \\
&\quad \mathcal{R}[\![\phi \sqcap \psi]\!]^\rho
\end{aligned}$$

Case: $\exists x.\, \phi$, similar to $\phi \sqcup \psi$.
Case: $\mu A.\, \lambda x.\, \phi$

$$
\begin{aligned}
(\mathcal{R}[\![\mu A.\, \lambda x.\, \phi]\!]^\rho)^\dagger &= \left(\bigcap \left\{ T_f \in \textsc{Val} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; \mathcal{R}[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto T_f]} \subseteq T_f \right\}\right)^\dagger \\
&= \text{by lemma 11 and Kleene's fixpoint theorem} \\
&\quad \left(\bigcup \left\{ T_f \in \textsc{Val} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; n \in \mathbb{N} \wedge T_f = (\mathcal{R}[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto \emptyset]})^n \right\}\right)^\dagger \\
&\supseteq \text{by lemma 13} \\
&\quad \bigcup \left\{ T_f \in \textsc{Val} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; n \in \mathbb{N} \wedge T_f = (\mathcal{R}[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto \emptyset]})^n \right\}^\dagger \\
&= \bigcup \left\{ T_f \in \textsc{Val} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; n \in \mathbb{N} \wedge T_f = \left(\left(\mathcal{R}[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto \emptyset]}\right)^\dagger\right)^n \right\} \\
&= \text{by lemma 11 and Kleene's fixpoint theorem} \\
&\quad \bigcap \left\{ T_f \in \textsc{Val} \to \mathcal{P}(\textsc{Trace}) \;\middle|\; \left([\![\lambda x.\, \phi]\!]^{\rho[A \mapsto T_f]}\right)^\dagger \subseteq {T_f}^\dagger \right\} \\
&= \text{by definition 35} \\
&\quad [\![\mu A.\, \lambda x.\, \phi]\!]^\rho
\end{aligned}
$$

All other cases follow directly from the inductive hypothesis. □

Lemma 17 (Trace-Set Interleaving is Associative and Commutative)**.**

$$T \parallel (S \parallel U) = (T \parallel S) \parallel U \qquad\qquad T \parallel U = U \parallel T$$

Proof. Immediate by definition 33. □

A.1 Operational traces are denotational traces

In definition 32 we defined the observed traces on the reflexive, transitive closure of $\rightsquigarrow$. As a stepping stone, we also define single step traces which we relate to the raw denotational semantics.

Definition 45 (Single-Step Observed Traces)**.** *The single-step observed traces relation, $\mathcal{O}_1[\![-]\!] \subseteq \mathcal{L} \times \mathcal{P}(\textsc{Trace})$, is the smallest relation that satisfies the following rules:*

$$\text{(26)}\quad \frac{}{(\lightning, \lightning) \in \mathcal{O}_1[\![\phi]\!]} \qquad \text{(27)}\quad \frac{\phi, h \rightsquigarrow o}{(h, o) \in \mathcal{O}_1[\![\phi]\!]} \qquad \text{(28)}\quad \frac{\phi, h \rightsquigarrow \psi, h' \qquad t \in \mathcal{O}_1[\![\psi]\!]}{(h, h')t \in \mathcal{O}_1[\![\phi]\!]}$$

Traces in $\mathcal{O}_1[\![\phi]\!]$ are not closed by the stuttering, mumbling and faulting closure.

We now relate the operational semantics to the raw denotational semantics: every single step in the operational semantics must be present as a move in the raw denotational semantics. Consequently, the traces observed by $\mathcal{O}_1[\![\phi]\!]$ are contained within $\mathcal{R}[\![\phi]\!]^\emptyset$, when ϕ is closed.

Lemma 18.

- *If $\phi, h \rightsquigarrow \psi, h'$ and $t \in \mathcal{R}[\![\psi]\!]^\emptyset$ then $(h, h')t \in \mathcal{R}[\![\phi]\!]^\emptyset$*
- *If $\phi, h \rightsquigarrow o$ then $(h, o) \in \mathcal{R}[\![\phi]\!]^\emptyset$*

Proof. By induction on the derivation of $\phi, h \rightsquigarrow \kappa$
Base case: rule (14).
Let $\forall \vec{x}.\, \langle P, Q \rangle_k, h \rightsquigarrow h'$. By the premiss, $(h, h') \in \left\{ \mathsf{a}(P([\vec{x} \mapsto \vec{v}]), Q([\vec{x} \mapsto \vec{v}]))_k \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \right\}$. By definition 44, $(h, h') \in \mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_k]\!]^\emptyset$.
Base case: rule (15).
Let $\forall \vec{x}.\, \langle P, Q \rangle_k, h \rightsquigarrow \lightning$. By the premiss,
For all $\vec{v} \in \overrightarrow{\textsc{Val}}$, $\mathsf{a}(P([\vec{x} \mapsto \vec{v}]), Q([\vec{x} \mapsto \vec{v}]))_k\,(h) = \emptyset$. By definition 44, $(h, \lightning) \in \mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_k]\!]^\emptyset$.
Case: rule (1).

Let $s = t; u$, such that, $s \in \mathcal{R}[\![\phi'; \psi]\!]^{\emptyset}$, $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$ and $u \in \mathcal{R}[\![\psi]\!]^{\emptyset}$. By the premiss and the inductive hypothesis, $(h, h')t \in \mathcal{R}[\![\phi]\!]^{\emptyset}$. Then, by definition 44, $(h, h')s = (h, h')tu \in \mathcal{R}[\![\phi; \psi]\!]^{\emptyset}$.
Case: rule (2).
Let $s \in \mathcal{R}[\![\psi]\!]^{\emptyset}$. By the premiss and the inductive hypothesis, $(h, h') \in \mathcal{R}[\![\phi]\!]^{\emptyset}$. Then, by definition 44, $(h, h')s \in \mathcal{R}[\![\phi; \psi]\!]^{\emptyset}$.
Case: rule (3).
By the premiss and the inductive hypothesis, $(h, \lightning) \in \mathcal{R}[\![\phi]\!]^{\emptyset}$. By lemma 8, $(\lightning, \lightning) \in \mathcal{R}[\![\psi]\!]^{\emptyset}$. By trace concatenation $(h, \lightning) = (h, \lightning); (\lightning, \lightning)$. Then, by definition 44, $(h, \lightning) \in \mathcal{R}[\![\phi; \psi]\!]^{\emptyset}$.
Case: rule (4).
By case analysis on κ. First, let $\kappa = \phi', h'$. Let $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$. By the premiss and the inductive hypothesis, $(h, h')t \in \mathcal{R}[\![\phi \parallel \psi]\!]^{\emptyset}$. Then, by definition 44 and lemma 17, $(h, h')t \in \mathcal{R}[\![\psi \parallel \phi]\!]^{\emptyset}$. Second, let $\kappa = o$. By the premiss and the inductive hypothesis, $(h, o) \in \mathcal{R}[\![\phi \parallel \psi]\!]^{\emptyset}$. Then, by definition 44 and lemma 17, $(h, o) \in \mathcal{R}[\![\psi \parallel \phi]\!]^{\emptyset}$.
Case: rule (5).
Let $s \in t \parallel u$, such that $s \in \mathcal{R}[\![\phi' \parallel \psi]\!]^{\emptyset}$, $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$ and $u \in \mathcal{R}[\![\psi]\!]^{\emptyset}$. By the premiss and the inductive hypothesis, $(h, h')t \in \mathcal{R}[\![\phi]\!]^{\emptyset}$. Then, by definition 33, $(h, h')s \in (h, h')t \parallel u$. By definitions 44 and 33, $(h, h')t \parallel u \subseteq \mathcal{R}[\![\phi \parallel \psi]\!]^{\emptyset}$, from which it follows that $(h, h')s \in \mathcal{R}[\![\phi \parallel \psi]\!]^{\emptyset}$.
Case: rule (6).
Let $s \in \mathcal{R}[\![\psi]\!]^{\emptyset}$. By the premiss and the inductive hypothesis, $(h, h') \in \mathcal{R}[\![\phi]\!]^{\emptyset}$. Then, by definition 33, $(h, h')s \in \mathcal{R}[\![\phi \parallel \psi]\!]^{\emptyset}$.
Case: rule (7).
By the premiss and the inductive hypothesis, $(h, \lightning) \in \mathcal{R}[\![\phi]\!]^{\emptyset}$. By lemma 8, $(\lightning, \lightning)\mathcal{R}[\![\psi]\!]^{\emptyset}$. By definition 33, $(h, \lightning) \in (h, \lightning) \parallel (\lightning, \lightning)$. By definition 44 and definition 33, $(h, \lightning) \parallel (\lightning, \lightning) \subseteq \mathcal{R}[\![\phi \parallel \psi]\!]^{\emptyset}$, from which it follows $(h, \lightning) \in \mathcal{R}[\![\phi \parallel \psi]\!]^{\emptyset}$.
Case: rule (8).
Case analysis on κ. First, let $\kappa = \phi', h'$. Let $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$. By the premiss, by the inductive hypothesis, for some $i \in \{0, 1\}$, $(h, h')t \in \mathcal{R}[\![\phi_i]\!]^{\emptyset}$. Thus, $(h, h')t \in \mathcal{R}[\![\phi_i]\!]^{\emptyset} \cup \mathcal{R}\left[\!\left[\phi_{(i+1) \bmod 2}\right]\!\right]^{\emptyset}$. Then, by definition 44, $(h, h')t \in \mathcal{R}[\![\phi_0 \sqcup \phi_1]\!]^{\emptyset}$. Second, let $\kappa = o$. By the premiss, by the inductive hypothesis, for some $i \in \{0, 1\}$, $(h, o) \in \mathcal{R}[\![\phi_i]\!]^{\emptyset}$. Thus, $(h, o) \in \mathcal{R}[\![\phi_i]\!]^{\emptyset} \cup \mathcal{R}\left[\!\left[\phi_{(i+1) \bmod 2}\right]\!\right]^{\emptyset}$. Then, by definition 44, $(h, o) \in \mathcal{R}[\![\phi_0 \sqcup \phi_1]\!]^{\emptyset}$.
Case: rule (9).
Case analysis on κ. First, let $\kappa = \phi', h'$. Let $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$. By the premiss, by the inductive hypothesis, for all $i \in \{0, 1\}$, $(h, h')t \in \mathcal{R}[\![\phi_i]\!]^{\emptyset}$. Thus, $(h, h')t \in \mathcal{R}[\![\phi_i]\!]^{\emptyset} \cap \mathcal{R}\left[\!\left[\phi_{(i+1) \bmod 2}\right]\!\right]^{\emptyset}$. Then, by definition 44, $(h, h')t \in \mathcal{R}[\![\phi_0 \sqcap \phi_1]\!]^{\emptyset}$. Second, let $\kappa = o$. By the premiss, by the inductive hypothesis, for all $i \in \{0, 1\}$, $(h, o) \in \mathcal{R}[\![\phi_i]\!]^{\emptyset}$. Thus, $(h, o) \in \mathcal{R}[\![\phi_i]\!]^{\emptyset} \cap \mathcal{R}\left[\!\left[\phi_{(i+1) \bmod 2}\right]\!\right]^{\emptyset}$. Then, by definition 44, $(h, o) \in \mathcal{R}[\![\phi_0 \sqcap \phi_1]\!]^{\emptyset}$.
Case: rule (10).
Case analysis on κ. First, let $\kappa = \phi', h'$. Let $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$. Fix v. By the premiss and the inductive hypothesis, $(h, h')t \in \mathcal{R}[\![\phi\,[v/x]]\!]^{\emptyset}$. Then, by definition 44, $(h, h')t \in \mathcal{R}[\![\exists x.\, \phi]\!]^{\emptyset}$. Second, let $\kappa = o$. By the premiss and the inductive hypothesis, $(h, o) \in \mathcal{R}[\![\phi\,[v/x]]\!]^{\emptyset}$. Then, by definition 44, $(h, o) \in \mathcal{R}[\![\exists x.\, \phi]\!]^{\emptyset}$.
Case: rule (11).
Case analysis on κ. First, let $\kappa = \phi', h'$. Let $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$. By the premiss and the inductive hypothesis, $(h, h')t \in \mathcal{R}[\![\phi\,[F/f]]\!]^{\emptyset}$. Then, by lemma 9, $(h, h')t \in \mathcal{R}[\![\phi]\!]^{\emptyset[f \mapsto \mathcal{R}[\![F]\!]^{\emptyset}]}$. Then, by definition 44, $(h, h')t \in \mathcal{R}[\![\textbf{let } f = F \textbf{ in } \phi]\!]^{\emptyset}$. Second, let $\kappa = o$. By the premiss and the inductive hypothesis, $(h, o) \in \mathcal{R}[\![\phi\,[F/f]]\!]^{\emptyset}$. Then, by lemma 9, $(h, o) \in \mathcal{R}[\![\phi]\!]^{\emptyset[f \mapsto \mathcal{R}[\![F]\!]^{\emptyset}]}$. Then, by definition 44, $(h, o) \in \mathcal{R}[\![\textbf{let } f = F \textbf{ in } \phi]\!]^{\emptyset}$.
Case: rule (12).
Case analysis on κ. First, let $\kappa = \phi', h'$. Let $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$. By the premiss and the inductive hypothesis, $(h, h')t \in \mathcal{R}[\![(\lambda x.\, \phi\,[\mu A.\, \lambda x.\, \phi/A])\, e]\!]^{\emptyset}$. Then, from the fact that μ is denotationally the least fixpoint, $(h, h')t \in \mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^{\emptyset}$. Second, let $\kappa = o$. By the premiss and the inductive hypothesis, $(h, o) \in \mathcal{R}[\![(\lambda x.\, \phi\,[\mu A.\, \lambda x.\, \phi/A])\, e]\!]^{\emptyset}$. Then, from the fact that μ is denotationally the least fixpoint, $(h, o) \in \mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^{\emptyset}$.
Case: rule (13).
Case analysis on κ. First, let $\kappa = \phi', h'$. Let $t \in \mathcal{R}[\![\phi']\!]^{\emptyset}$. By the premiss and the inductive hypothesis, $(h, h')t \in \mathcal{R}\left[\!\left[\phi\left[[\![e]\!]^{\emptyset}/x\right]\right]\!\right]^{\emptyset}$. Then, by lemma 10, $(h, h')t \in \mathcal{R}[\![\phi]\!]^{\emptyset[x \mapsto [\![e]\!]]}$. Then, by definition 44, $(h, h')t \in \mathcal{R}[\![\lambda x.\, \phi]\!]^{\emptyset}$.

Second, let $\kappa = o$. By the premiss and the inductive hypothesis, $(h, o) \in \mathcal{R}\left[\!\left[\phi\left[\llbracket e \rrbracket^{\emptyset} / x\right]\right]\!\right]^{\emptyset}$. Then, by lemma 10, $(h, o) \in \mathcal{R}\llbracket\phi\rrbracket^{\emptyset[x \mapsto \llbracket e \rrbracket^{\emptyset}]}$. Then, by definition 44, $(h, o) \in \mathcal{R}\llbracket\lambda x.\, \phi\rrbracket^{\emptyset}$. □

Corollary 1. *If $t \in \mathcal{O}_1\llbracket\phi\rrbracket$ then $t \in \mathcal{R}\llbracket\phi\rrbracket^{\emptyset}$.*

Proof. Straightforward induction on the derivation of $t \in \mathcal{O}_1\llbracket\phi\rrbracket$, using lemma 18.
Base case: rule (26).
$(\wr, \wr) \in \mathcal{O}_1\llbracket\phi\rrbracket$ and by lemma 8, $(\wr, \wr) \in \mathcal{R}\llbracket\phi\rrbracket^{\emptyset}$.
Base case: rule (27).
By the premiss and lemma 18, $(h, \kappa) \in \mathcal{R}\llbracket\phi\rrbracket^{\emptyset}$.
Case: rule (28).
By the premisses and lemma 18, $(h, h')t \in \mathcal{R}\llbracket\phi\rrbracket^{\emptyset}$ □

Traces in $\mathcal{O}_1\llbracket\phi\rrbracket$ are obtained by observing every single transition in the operational semantics. They relate to traces in $\mathcal{O}\llbracket\phi\rrbracket$, which are obtained by observing transitions in the transitive and reflexive closure of the operational semantics, by the stuttering, mumbling and faulting closure.

Lemma 19. *If $t \in \mathcal{O}\llbracket\phi\rrbracket$ then $t \in (\mathcal{O}_1\llbracket\phi\rrbracket)^{\dagger}$.*

Proof. Induction on the derivation of $t \in \mathcal{O}\llbracket\phi\rrbracket$, with nested induction on steps $\kappa \rightsquigarrow^* \kappa'$.
Base case: rule (16).
$(\wr, \wr) \in \mathcal{O}\llbracket\phi\rrbracket$ and by definition 34, $(\wr, \wr) \in (\mathcal{O}_1\llbracket\phi\rrbracket)^{\dagger}$.
Base case: rule (17).
By induction on the derivation of the premiss.
Nested base case: $\phi, h \rightsquigarrow o$
By definition 45, $(h, o) \in \mathcal{O}_1\llbracket\phi\rrbracket$. Then by definition 34, $(h, o) \in (\mathcal{O}_1\llbracket\phi\rrbracket)^{\dagger}$.
Nested case: $\phi, h, \rightsquigarrow \psi, h'$ and $\psi, h' \rightsquigarrow^* o$
By the inductive hypothesis, $(h', o) \in (\mathcal{O}_1\llbracket\psi\rrbracket)^{\dagger}$. Then, by definition 34, there exist h'', h''', t such that if $t = \epsilon$, then $h''' = h''$, and $(h', h'')t(h''', o) \in \mathcal{O}_1\llbracket\psi\rrbracket$. Then, by definition 45, $(h, h')(h', h'')t(h''', o) \in \mathcal{O}_1\llbracket\phi\rrbracket$. Then, by definition 34, $(h, o) \in (\mathcal{O}\llbracket\phi\rrbracket)^{\dagger}$.
Case: rule (18).
Let $t \in \mathcal{O}\llbracket\psi\rrbracket$. By induction on the derivation of the premiss.
Nested base case: $\phi, h \rightsquigarrow \psi, h'$ By the inductive hypothesis $t \in (\mathcal{O}_1\llbracket\psi\rrbracket)^{\dagger}$. By definition 34, there exists $u \in \mathcal{O}_1\llbracket\psi\rrbracket$ such that $t \in (\mathcal{O}_1\llbracket\psi\rrbracket)^{\dagger}$. Then, by definition 45, $(h, h')u \in \mathcal{O}_1\llbracket\phi\rrbracket$. Then, by definition 34, $(h, h')t \in (\mathcal{O}_1\llbracket\phi\rrbracket)^{\dagger}$.
Nested case: $\phi, h \rightsquigarrow \phi', h''$ and $\phi', h'' \rightsquigarrow^* \psi, h'$. By the inductive hypothesis, $(h'', h')t \in (\mathcal{O}_1\llbracket\phi'\rrbracket)^{\dagger}$. By definition 34, there exists u, v, h''', h'''', such that $(h'', h''')u(h'''', h')v \in \mathcal{O}_1\llbracket\phi'\rrbracket$. Then, by definition 45, $(h, h'')(h'', h''')u(h'''', h')v \in \mathcal{O}_1\llbracket\phi\rrbracket$. Then, by definition 34, $(h, h')t \in (\mathcal{O}_1\llbracket\phi\rrbracket)^{\dagger}$. □

Corollary 2. *If $t \in \mathcal{O}\llbracket\phi\rrbracket$ then $t \in \llbracket\phi\rrbracket^{\emptyset}$.*

Proof.

$$
\begin{aligned}
\mathcal{O}\llbracket\phi\rrbracket &\subseteq \text{by lemma 19} \\
&\quad (\mathcal{O}_1\llbracket\phi\rrbracket)^{\dagger} \\
&\subseteq \text{by corollary 1 and lemma 12} \\
&\quad \left(\mathcal{R}\llbracket\phi\rrbracket^{\emptyset}\right)^{\dagger} \\
&= \text{by lemma 16} \\
&\quad \llbracket\phi\rrbracket^{\emptyset}
\end{aligned}
$$

□

A.2 Denotational traces are operational traces

The following two intermediate lemmas, establish compositional properties for $\mathcal{O}_1[\![\phi]\!]$.

Lemma 20. $\mathcal{O}_1[\![\phi]\!]; \mathcal{O}_1[\![\psi]\!] \subseteq \mathcal{O}_1[\![\phi; \psi]\!]$

Proof. We prove that if $t \in \mathcal{O}_1[\![\phi]\!]$ and $u \in \mathcal{O}_1[\![\psi]\!]$, then $tu \in \mathcal{O}_1[\![\phi; \psi]\!]$, by induction on the derivation of $t \in \mathcal{O}_1[\![\phi]\!]$.
Base case: rule (26).
$(\natural, \natural) \in \mathcal{O}_1[\![\phi]\!]$, $(\natural, \natural) \in \mathcal{O}_1[\![\psi]\!]$ and $(\natural, \natural) \in \mathcal{O}_1[\![\phi; \psi]\!]$.
Base case: rule (27).
Let $u \in \mathcal{O}_1[\![\psi]\!]$. Let $(h, o) \in \mathcal{O}_1[\![\phi]\!]$. Case analysis on o. First, let $o = \natural$. Then, by the premiss and definition 28, $\phi; \psi, h \rightsquigarrow \natural$, by which $(h, \natural) \in \mathcal{O}_1[\![\phi; \psi]\!]$. By trace concatenation, $(h, \natural) = (h, \natural)u$, by which $(h, \natural)u \in \mathcal{O}_1[\![\phi; \psi]\!]$. Second, let $o = h'$. By definition 28, $\phi; \psi, h \rightsquigarrow \psi, h'$. Then, by rule (28), $(h, h')u \in \mathcal{O}_1[\![\phi; \psi]\!]$.
Case: rule (28).
Let $u \in \mathcal{O}_1[\![\psi]\!]$. Let $(h, h')t \in \mathcal{O}_1[\![\phi]\!]$. Then, by the premiss, there exists ϕ', such that $\phi, h \rightsquigarrow \phi', h'$ and $t \in \mathcal{O}_1[\![\phi']\!]$. By the induction hypothesis, $tu \in \mathcal{O}_1[\![\phi'; \psi]\!]$. From the premiss and definition 28, $\phi; \psi, h \rightsquigarrow \phi'; \psi, h'$. Then, $(h, h')tu \in \mathcal{O}_1[\![\phi; \psi]\!]$. □

Lemma 21. $\mathcal{O}_1[\![\phi]\!] \parallel \mathcal{O}_1[\![\psi]\!] \subseteq \mathcal{O}_1[\![\phi \parallel \psi]\!]$.

Proof. We prove that if $t \in \mathcal{O}_1[\![\phi]\!]$, $u \in \mathcal{O}_1[\![\psi]\!]$, and $s \in t \parallel u$, then $s \in \mathcal{O}_1[\![\phi \parallel \psi]\!]$, by induction on the derivation of $t \in \mathcal{O}_1[\![\phi]\!]$.
Base case: rule (26). $(\natural, \natural) \in \mathcal{O}_1[\![\phi]\!]$, $(\natural, \natural) \in \mathcal{O}_1[\![\psi]\!]$ and $(\natural, \natural) \in \mathcal{O}_1[\![\phi \parallel \psi]\!]$.
Base case: rule (27).
Let $u \in \mathcal{O}_1[\![\psi]\!]$. Let $(h, o) \in \mathcal{O}_1[\![\phi]\!]$. Case analysis on o. First, let $o = \natural$. By definition 33, $(h, \natural) \in (h, \natural) \parallel u$. Then, by premiss and by definition 28, $\phi \parallel \psi, h \rightsquigarrow \natural$, by which $(h, \natural) \in \mathcal{O}_1[\![\phi \parallel \psi]\!]$. Second, let $o = h'$. By definition 33, $(h, h')u \in (h, h') \parallel u$. Then, by premiss and by definition 28, $\phi \parallel \psi, h \rightsquigarrow \psi, h'$. Then, by rule (28), $(h, h')u \in \mathcal{O}_1[\![\phi \parallel \psi]\!]$.
Case: rule (28).
Let $u \in \mathcal{O}_1[\![\psi]\!]$. Let $(h, h')s \in \mathcal{O}_1[\![\phi]\!]$. Then, by the premiss, there exists ϕ', such that $\phi, h \rightsquigarrow \phi', h'$ and $s \in \mathcal{O}_1[\![\phi']\!]$. Let $w \in s \parallel u$. By definition 33, $(h, h')w \in (h, h')s \parallel u$. By the inductive hypothesis, $w \in \mathcal{O}_1[\![\phi' \parallel \psi]\!]$. From the premiss and definition 28, $\phi \parallel \psi, h \rightsquigarrow \phi' \parallel \psi, h'$. Then, $(h, h')w \in \mathcal{O}_1[\![\phi \parallel \psi]\!]$. □

Lemma 22. $\mathcal{O}_1[\![\phi]\!] \cup \mathcal{O}_1[\![\psi]\!] \subseteq \mathcal{O}_1[\![\phi \sqcup \psi]\!]$.

Proof. We prove that if $t \in \mathcal{O}_1[\![\phi]\!]$ or $t \in \mathcal{O}_1[\![\phi]\!]$, then $t \in \mathcal{O}_1[\![\phi \sqcup \psi]\!]$, by proving: a). if $t \in \mathcal{O}_1[\![\phi]\!]$, then $t \in \mathcal{O}_1[\![\phi \sqcup \psi]\!]$, and b). if $t \in \mathcal{O}_1[\![\psi]\!]$, then $t \in \mathcal{O}_1[\![\phi \sqcup \psi]\!]$, each by induction on the derivation of $t \in \mathcal{O}_1[\![\phi]\!]$.
Proof of a).
Base case: rule (26). $(\natural, \natural) \in \mathcal{O}_1[\![\phi]\!]$, $(\natural, \natural) \in \mathcal{O}_1[\![\psi]\!]$ and $(\natural, \natural) \in \mathcal{O}_1[\![\phi \sqcup \psi]\!]$.
Base case: rule (27).
Let $(h, o) \in \mathcal{O}_1[\![\phi]\!]$. Then, by premiss and definition 28, $\phi \sqcup \psi, h \rightsquigarrow o$. It follows that, $(h, o) \in \mathcal{O}_1[\![\phi \sqcup \psi]\!]$.
Case: rule (28).
Let $u \in \mathcal{O}_1[\![\psi]\!]$. Let $(h, h')t \in \mathcal{O}_1[\![\phi]\!]$. Then, by the premiss, there exists ϕ', such that $\phi, h \rightsquigarrow \phi', h'$ and $t \in \mathcal{O}_1[\![\phi']\!]$. By the induction hypothesis, $t \in \mathcal{O}_1[\![\phi' \sqcup \psi]\!]$. From definition 28, let $\phi \sqcup \psi, h \rightsquigarrow \phi', h'$. It follows that, $(h, h')t \in \mathcal{O}_1[\![\phi \sqcup \psi]\!]$.
Proof of b). As in a). □

Lemma 23. $\bigcup_v \mathcal{O}_1[\![\phi\,[v/x]]\!] \subseteq \mathcal{O}_1[\![\exists x.\, \phi]\!]$.

Proof. We prove that if $t \in \bigcup_v \mathcal{O}_1[\![\phi\,[v/x]]\!]$, then $t \in \mathcal{O}_1[\![\exists x.\, \phi]\!]$, by induction on the length of t.
Let $v \in \textsc{Val}$.
Base case: $(\natural, \natural)$, trivial.
Base case: $(h, o) \in \mathcal{O}_1[\![\phi\,[v/x]]\!]$.
By rule (27), $\phi\,[v/x], h \rightsquigarrow o$. Then, by definition 28, $\exists x.\, \phi, h \rightsquigarrow o$. It follows that, $(h, o) \in \mathcal{O}_1[\![\exists x.\, \phi]\!]$.
Case: $(h, h')t \in \mathcal{O}_1[\![\phi\,[v/x]]\!]$.
By rule (28), there exists ψ such that $\phi\,[v/x], \phi \rightsquigarrow \psi, h'$ and $t \in \mathcal{O}_1[\![\psi]\!]$. By definition 28, $\exists x.\, \phi, h \rightsquigarrow \psi, h'$. By rule (28), $(h, h')t \in \mathcal{O}_1[\![\exists x.\, \phi]\!]$.

□

Lemma 24. $\mathcal{O}_1[\![\phi\,[F/f]]\!] \subseteq \mathcal{O}_1[\![\textbf{let } f = F \textbf{ in } \phi]\!]$.

Proof. By induction on the length of $t \in \mathcal{O}_1[\![\phi\,[F/f]]\!]$.

□

Lemma 25. $\mathcal{O}_1[\![\phi\left[[\![e]\!]^{\emptyset}/x\right]]\!] \subseteq \mathcal{O}_1[\![(\lambda x.\,\phi)\,e]\!]$.

Proof. Similarly, to lemma 24. □

Lemma 26. $\mathcal{O}_1[\![\phi\,[\mu A.\,\lambda x.\,\phi/A]\,[[\![e]\!]/x]]\!] \subseteq \mathcal{O}_1[\![(\mu A.\,\lambda x.\,\phi)\,e]\!]$.

Proof. Similarly to lemma 25.

□

The following lemma is a direct consequence of the fact that recursion is semantically the least fixpoint.

Lemma 27. $\mathcal{R}[\![\lambda x.\,\phi]\!]^{\rho[A\mapsto\mathcal{R}[\![\mu A.\lambda x.\phi]\!]^{\rho}]} = \mathcal{R}[\![\mu A.\,\lambda x.\,\phi]\!]^{\rho}$

Proof. By induction on ϕ.
Base case : Ae.

$$
\begin{aligned}
&\mathcal{R}[\![\lambda x.\,Ae]\!]^{\rho[A\mapsto\mathcal{R}[\![\mu A.\lambda x.Ae]\!]^{\rho}]}\\
&= \lambda v.\,\mathcal{R}[\![Ae]\!]^{\rho[A\mapsto\mathcal{R}[\![\mu A.\lambda x.Ae]\!]^{\rho}][x\mapsto v]}\\
&= \lambda v.\left(\mathcal{R}[\![A]\!]^{\rho[A\mapsto\mathcal{R}[\![\mu A.\lambda x.Ae]\!]^{\rho}][x\mapsto v]}\right)[\![e]\!]^{\rho[A\mapsto\mathcal{R}[\![\mu A.\lambda x.Ae]\!]^{\rho}][x\mapsto v]}\\
&= \lambda v.\,(\mathcal{R}[\![\mu A.\,\lambda x.\,Ae]\!]^{\rho})\,[\![e]\!]^{\rho[x\mapsto v]}\\
&= \text{by definition 44, lemma 11 and Kleene's fixpoint theorem}\\
&\quad \lambda v.\left(\bigcup\left\{T_f\in \textsc{Val}\to\mathcal{P}(\textsc{Trace}) \;\middle|\; n\in\mathbb{N}\wedge T_f = (\mathcal{R}[\![\lambda x.\,Ae]\!]^{\rho[A\mapsto\emptyset]})^n\right\}\right)[\![e]\!]^{\rho[x\mapsto v]}\\
&= \lambda v.\left(\bigcup\left\{T_f\in \textsc{Val}\to\mathcal{P}(\textsc{Trace}) \;\middle|\; n\in\mathbb{N}\wedge T_f = (\lambda v.\left(\mathcal{R}[\![A]\!]^{\rho[A\mapsto\emptyset][x\mapsto v]}\right)[\![e]\!]^{\rho[x\mapsto v]})^n\right\}\right)[\![e]\!]^{\rho[x\mapsto v]}\\
&= \bigcup\left\{T_f\in \textsc{Val}\to\mathcal{P}(\textsc{Trace}) \;\middle|\; n\in\mathbb{N}\wedge T_f = (\lambda v.\left(\mathcal{R}[\![A]\!]^{\rho[A\mapsto\emptyset][x\mapsto v]}\right)[\![e]\!]^{\rho[x\mapsto v]})^{n+1}\right\}\\
&= \text{by Kleene's fixpoint theorem}\\
&\quad \mathcal{R}[\![\mu A.\,\lambda x.\,\phi]\!]^{\rho}
\end{aligned}
$$

All other cases follow directly from the inductive hypothesis. □

We now establish the reverse of corollary 1: $\mathcal{R}[\![\phi]\!]^{\emptyset} \subseteq \mathcal{O}_1[\![\phi]\!]$. This is difficult to prove directly by induction over ϕ. Specifically, substructures of ϕ extend the variable environment, for example $\mathcal{R}[\![\exists x.\,\phi]\!]^{\emptyset} = \bigcup_v \mathcal{R}[\![\phi]\!]^{\emptyset[x\mapsto v]}$, and thus we cannot directly apply the inductive hypothesis. The solution is to generalise the property for arbitrary variable stores. Then, the property we wish to prove takes the form $\mathcal{R}[\![\phi]\!]^{\rho} \subseteq \mathcal{O}_1[\![C[\phi]]\!]$, where C is a context that closes ϕ according to the bindings in ρ. In order to precisely state this property, we first formally define closing contexts induced by variable stores.

Definition 46 (Closing contexts)**.**

$$
\begin{array}{lrl}
\textit{Syntactic environments:} & \eta ::= \emptyset \mid x\mapsto e:\eta \mid f\mapsto F:\eta \mid A\mapsto\phi & \\
\textit{Syntactic environment application:} & \theta(\emptyset\phi) & \triangleq \phi\\
& \theta((x\mapsto e:\eta)\phi) & \triangleq \theta(\eta(\phi\left[[\![e]\!]^{\omega(\eta)}/x\right]))\\
& \theta((f\mapsto F:\eta)\phi) & \triangleq \theta(\eta(\phi\,[F/f]))\\
& \theta((A\mapsto\psi:\eta)\phi) & \triangleq \theta(\eta(\phi\,[\mu A.\,\lambda x.\,\psi/A]))\\
\textit{Syntactic environment erasure:} & \omega(\emptyset) & \triangleq \emptyset\\
& \omega(x\mapsto e:\eta) & \triangleq \omega(\eta)[x\mapsto [\![e]\!]^{\omega(\eta)}]\\
& \omega(f\mapsto F:\eta) & \triangleq \omega(\eta)[f\mapsto \mathcal{R}[\![F]\!]^{\omega(\eta)}]\\
& \omega(A\mapsto\psi:\eta) & \triangleq \omega(\eta)[A\mapsto \mathcal{R}[\![\mu A.\,\lambda x.\,\psi]\!]^{\omega(\eta)}]
\end{array}
$$

Intuitively, a syntactic environment, η, represents the closing context. We use η, to list the variable bindings that we need to introduce in ϕ. The syntactic environment application, $theta(\eta\ \phi)$, applies the syntactic environment η to ϕ by substituting the variables in ϕ with their bound values. The syntactic environment erasure, $\omega(\eta)$, erases η to a variable store that binds variables according to η.

Given definition 46, the reverse of corollary 1 is: $\mathcal{R}[\![\phi]\!]^{\omega(\eta)} \subseteq \mathcal{O}_1[\![\theta(\eta\phi)]\!]$.

Lemma 28. $\mathcal{R}[\![\phi]\!]^{\omega(\eta)} \subseteq \mathcal{O}_1[\![\theta(\eta\phi)]\!]$.

Proof. By induction on ϕ.
Base case: $\forall \vec{x}.\, \langle P, Q\rangle_k$.

$$
\begin{aligned}
\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^{\omega(\eta)} &= \text{by definition 44} \\
&\quad \left\{ (h, h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\omega(\eta)[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\omega(\eta)[\vec{x}\mapsto\vec{v}]} \right)_k (h) \right\} \\
&\quad \cup \left\{ (h, \lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l} \vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\omega(\eta)[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\omega(\eta)[\vec{x}\mapsto\vec{v}]} \right)_k (h) = \emptyset \\ \quad \wedge (\!|Q|\!)^{\omega(\eta)[\vec{x}\mapsto\vec{v}]}_{\neq} \emptyset \end{array} \right\} \\
&\quad \cup \left\{ (\lightning, \lightning) \right\} \\
&= \text{by induction on } \eta \text{ and definition 45} \\
&\quad \mathcal{O}_1[\![\theta(\eta(\forall \vec{x}.\, \langle P, Q\rangle_k))]\!]
\end{aligned}
$$

Case: $\phi; \psi$.

$$
\begin{aligned}
\mathcal{R}[\![\phi; \psi]\!]^{\omega(\eta)} &= \text{by definition 44} \\
&\quad \mathcal{R}[\![\phi]\!]^{\omega(\eta)} ; \mathcal{R}[\![\psi]\!]^{\omega(\eta)} \\
&\subseteq \text{by induction hypothesis} \\
&\quad \mathcal{O}_1[\![\theta(\eta\phi)]\!]; \mathcal{O}_1[\![\theta(\eta\psi)]\!] \\
&\subseteq \text{by induction over } \eta \text{ and lemma 20} \\
&\quad \mathcal{O}_1[\![\theta(\eta(\phi; \psi))]\!]
\end{aligned}
$$

Case: $\phi \parallel \psi$.

$$
\begin{aligned}
\mathcal{R}[\![\phi \parallel \psi]\!]^{\omega(\eta)} &= \text{by definition 44} \\
&\quad \mathcal{R}[\![\phi]\!]^{\omega(\eta)} \parallel \mathcal{R}[\![\psi]\!]^{\omega(\eta)} \\
&\subseteq \text{by induction hypothesis} \\
&\quad \mathcal{O}_1[\![\theta(\eta\phi)]\!] \parallel \mathcal{O}_1[\![\theta(\eta\psi)]\!] \\
&\subseteq \text{by induction over } \eta \text{ and lemma 21} \\
&\quad \mathcal{O}_1[\![\theta(\eta(\phi \parallel \psi))]\!]
\end{aligned}
$$

Case: $\phi \sqcup \psi$.

$$
\begin{aligned}
\mathcal{R}[\![\phi \sqcup \psi]\!]^{\omega(\eta)} &= \text{by definition 44} \\
&\quad \mathcal{R}[\![\phi]\!]^{\omega(\eta)} \cup \mathcal{R}[\![\psi]\!]^{\omega(eta)} \\
&\subseteq \text{by induction hypothesis} \\
&\quad \mathcal{O}_1[\![\theta(\eta\phi)]\!] \cup \mathcal{O}_1[\![\theta(\eta\psi)]\!] \\
&\subseteq \text{by induction over } \eta \text{ and lemma 22} \\
&\quad \mathcal{O}_1[\![\phi \sqcup \psi]\!]
\end{aligned}
$$

Case: $\phi \sqcap \psi$, similarly to previous.
Case: $\exists x.\, \phi$.

$$
\begin{aligned}
\mathcal{R}[\![\exists x.\, \phi]\!]^{\omega(\eta)} &= \text{by definition 44} \\
&\quad \bigcup_v \mathcal{R}[\![\phi]\!]^{\omega(\eta)[x \mapsto v]} \\
&= \bigcup_v \mathcal{R}[\![\phi]\!]^{\omega(x \mapsto v : \eta)} \\
&\subseteq \text{by induction hypothesis} \\
&\quad \bigcup_v \mathcal{O}_1[\![\theta((x \mapsto v : \eta)\phi)]\!] \\
&= \text{by syntactic environment application (definition 46)} \\
&\quad \bigcup_v \mathcal{O}_1[\![\theta(\eta(\phi\,[v/x]))]\!] \\
&\subseteq \text{by induction over } \eta \text{ and lemma 23} \\
&\quad \mathcal{O}_1[\![\theta(\eta(\exists x.\, \phi))]\!]
\end{aligned}
$$

Case: $\mathbf{let}\ f = F\ \mathbf{in}\ \phi$

$$
\begin{aligned}
\mathcal{R}[\![\mathbf{let}\ f = F\ \mathbf{in}\ \phi]\!]^{\omega(\eta)} &= \text{by definition 44} \\
&\quad \mathcal{R}[\![\phi]\!]^{\omega(\eta)[f \mapsto \mathcal{R}[\![F]\!]^{\emptyset}]} \\
&= \mathcal{R}[\![\phi]\!]^{\omega(f \mapsto F : \eta)} \\
&\subseteq \text{by induction hypothesis} \\
&\quad \mathcal{O}_1[\![\theta((f \mapsto F : \eta)\phi)]\!] \\
&= \text{by syntactic environment application (definition 46)} \\
&\quad \mathcal{O}_1[\![\theta(\eta(\phi\,[F/f]))]\!] \\
&\subseteq \text{by induction over } \eta \text{ and lemma 24} \\
&\quad \mathcal{O}_1[\![\theta(\eta(\mathbf{let}\ f = F\ \mathbf{in}\ \phi))]\!]
\end{aligned}
$$

Case: $(\lambda x.\, \phi)\, e$.

$$
\begin{aligned}
\mathcal{R}[\![(\lambda x.\, \phi)\, e]\!]^{\omega(\eta)} &= \text{by definition 44} \\
&\quad \left(\mathcal{R}[\![\lambda x.\, \phi]\!]^{\omega(\eta)}\right) [\![e]\!]^{\omega(\eta)} \\
&= \left(\lambda v.\, \mathcal{R}[\![\phi]\!]^{\omega(\eta)[x \mapsto v]}\right) [\![e]\!]^{\omega(\eta)} \\
&= \mathcal{R}[\![\phi]\!]^{\omega(\eta)[x \mapsto [\![e]\!]^{\omega(\eta)}]} \\
&= \mathcal{R}[\![\phi]\!]^{\omega(x \mapsto [\![e]\!]^{\omega(\eta)} : \eta)} \\
&\subseteq \text{by the induction hypothesis} \\
&\quad \mathcal{O}_1[\![\theta((x \mapsto [\![e]\!]^{\omega(\eta)} : \eta)\phi)]\!] \\
&= \text{by syntactic environment application (definition 46)} \\
&\quad \mathcal{O}_1[\![\theta(\eta(\phi\,\left[[\![e]\!]^{\omega(\eta)}/x\right]))]\!] \\
&\subseteq \text{by induction over } \eta \text{ and lemma 25} \\
&\quad \mathcal{O}_1[\![\theta(\eta((\lambda x.\, \phi)\, e))]\!]
\end{aligned}
$$

Case: $(\mu A.\, \lambda x.\, \phi)\, e$.

$$
\begin{aligned}
\mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^{\omega(\eta)} &= \text{by lemma 27} \\
&\quad \mathcal{R}[\![(\lambda x.\, \phi)\, e]\!]^{\omega(\eta)[A \mapsto \mathcal{R}[\![\mu A.\lambda x.\phi]\!]^{\omega(\eta)}]} \\
&= \mathcal{R}[\![\phi]\!]^{\omega(\eta)[A \mapsto \mathcal{R}[\![\mu A.\lambda x.\phi]\!]^{\omega(\eta)}][x \mapsto [\![e]\!]^{\omega(\eta)}]} \\
&= \mathcal{R}[\![\phi]\!]^{\omega(x \mapsto [\![e]\!]^{\omega(\eta)} : A \mapsto \phi : \eta)} \\
&\subseteq \text{by the induction hypothesis} \\
&\quad \mathcal{O}_1[\![\theta((x \mapsto [\![e]\!]^{\omega(\eta)} : A \mapsto \phi : \eta)\phi)]\!] \\
&= \text{by syntactic environment application (definition 46)} \\
&\quad \mathcal{O}_1[\![\theta(\eta(\phi\,[\mu A.\, \lambda x.\, \phi / A]\, \left[[\![e]\!]^{\omega(\eta)} / x \right]))]\!] \\
&\subseteq \text{by induction over } \eta \text{ and lemma 26} \\
&\quad \mathcal{O}_1[\![\theta(\eta((\mu A.\, \lambda x.\, \phi)\, e))]\!]
\end{aligned}
$$

Case: Ae. By induction on η.
Case: fe. By induction on η. □

The following lemma reflects the fact that $\mathcal{O}[\![\phi]\!]$ are obtained from the reflexive, transitive closure of a single step in the operational semantics.

Lemma 29. *If $t \in \mathcal{O}_1[\![\phi]\!]$, then $t \in \mathcal{O}[\![\phi]\!]$.*

Proof. By induction on the derivation of $t \in \mathcal{O}_1[\![\phi]\!]$.
Base case: rule (26).
Trivial; $(\wr, \wr) \in \mathcal{O}_1[\![\phi]\!]$ and $(\wr, \wr) \in \mathcal{O}[\![\phi]\!]$.
Base case: rule (27).
Let $(h, o) \in \mathcal{O}_1[\![\phi]\!]$. By the premiss, $\phi, h \rightsquigarrow o$, from which it follows that $\phi, h \rightsquigarrow^* o$. Then, by rule (17), $(h, o) \in \mathcal{O}[\![\phi]\!]$.
Case: rule (28).
Let $(h, h')t \in \mathcal{O}_1[\![\phi]\!]$. By the premiss, there exists ψ, such that $\phi, h \rightsquigarrow \psi, h'$ and $t \in \mathcal{O}_1[\![\psi]\!]$. It follows that $\phi, h \rightsquigarrow^* \psi, h'$. From the induction hypothesis, $t \in \mathcal{O}[\![\psi]\!]$. Then, by rule (18), $(h, h')t \in \mathcal{O}[\![\phi]\!]$. □

Corollary 3. *If $t \in \mathcal{O}_1[\![\phi]\!]$, then $t \in (\mathcal{O}[\![\phi]\!])^\dagger$.*

Proof. By lemma 29, $t \in \mathcal{O}_1[\![\phi]\!] \Rightarrow t \in \mathcal{O}[\![\phi]\!]$. By rule (23), $t \in \mathcal{O}[\![\phi]\!] \Rightarrow t \in (\mathcal{O}[\![\phi]\!])^\dagger$. Then, by transitivity $t \in \mathcal{O}_1[\![\phi]\!] \Rightarrow t \in (\mathcal{O}[\![\phi]\!])^\dagger$. □

Corollary 4. *If ϕ is closed, then $[\![\phi]\!]^\emptyset \subseteq (\mathcal{O}[\![\phi]\!])^\dagger$.*

Proof.

$$
\begin{aligned}
[\![\phi]\!]^\emptyset &= \text{by lemma 16} \\
&\quad \left(\mathcal{R}[\![\phi]\!]^\emptyset\right)^\dagger \\
&\subseteq \text{by lemma 28 and lemma 12 (increasing property)} \\
&\quad (\mathcal{O}_1[\![\phi]\!])^\dagger \\
&\subseteq \text{by lemma 29 and lemma 12 (increasing property)} \\
&\quad (\mathcal{O}[\![\phi]\!])^\dagger
\end{aligned}
$$

□

Corollary 5. *If ϕ is closed, then $[\![\phi]\!]^\emptyset = (\mathcal{O}[\![\phi]\!])^\dagger$.*

Proof. From corollary 2, $\mathcal{O}[\![\phi]\!] \subseteq [\![\phi]\!]^\emptyset$.
By lemma 12 (increasing), $(\mathcal{O}[\![\phi]\!])^\dagger \subseteq \left([\![\phi]\!]^\emptyset\right)^\dagger$.
By lemma 15, $(\mathcal{O}[\![\phi]\!])^\dagger \subseteq [\![\phi]\!]^\emptyset$.
Then, by corollary 4, $[\![\phi]\!]^\emptyset = (\mathcal{O}[\![\phi]\!])^\dagger$. □

B Proofs of General Refinement Laws

Lemma 30 (Refl). $\phi \sqsubseteq \phi$

Proof. Immediate, by definition 36 and reflexivity of $\subseteq$. □

Lemma 31 (Trans). *If $\phi \sqsubseteq \psi'$, and $\psi' \sqsubseteq \psi$, then $\phi \sqsubseteq \psi$.*

Proof. Immediate, by definition 36 and transitivity of $\subseteq$. □

Lemma 32 (AntiSymm). *If $\phi \sqsubseteq \psi$, and $\psi \sqsubseteq \phi$, then $\phi \equiv \psi$.*

Proof. Immediate, by definition 36 and anti-symmetricity of $\subseteq$. □

Lemma 33 (Skip). $\texttt{skip}; \phi \equiv \phi \equiv \phi; \texttt{skip}$

Proof. Let ρ such that it closes ϕ.
By definition 37, $\mathcal{R}[\![\texttt{skip}]\!]^\rho = \mathcal{R}[\![\langle \mathsf{true}, \mathsf{true}\rangle_k]\!]^\rho$.
Then by definitions 44 and 26,
$\mathcal{R}[\![\forall \vec{x}.\, \langle \mathsf{true}, \mathsf{true}\rangle_k]\!]^\rho = \{(h,h) \mid h \in \textsc{Heap}\} \cup \{(\lightning, \lightning)\}$.
By rules CLStutter and CLMumble, $(\mathcal{R}[\![\texttt{skip}]\!]^\rho ; \mathcal{R}[\![\phi]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi]\!]^\rho ; \mathcal{R}[\![\texttt{skip}]\!]^\rho)^\dagger$.
By lemma 16, $[\![\texttt{skip}; \phi]\!]^\rho = [\![\phi]\!]^\rho = [\![\phi; \texttt{skip}]\!]^\rho$.
By definition 36, $\texttt{skip}; \phi \equiv \phi \equiv \phi; \texttt{skip}$. □

Lemma 34 (Assoc). $\phi; (\psi_1; \psi_2) \equiv (\phi; \psi_1); \psi_2$

Proof. Let ρ such that it closes ϕ, ψ_1 and ψ_2.
By definitions 44 and 31, $\mathcal{R}[\![\phi; (\psi_1; \psi_2)]\!]^\rho = \mathcal{R}[\![(\phi; \psi_1); \psi_2]\!]^\rho$.
By lemma 12, $(\mathcal{R}[\![\phi; (\psi_1; \psi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\phi; \psi_1); \psi_2]\!]^\rho)^\dagger$.
By lemma 16, $[\![\phi; (\psi_1; \psi_2)]\!]^\rho = [\![(\phi; \psi_1); \psi_2]\!]^\rho$. □

Lemma 35. $\texttt{miracle} \sqsubseteq \phi$

Proof. Let ρ such that it closes ϕ.
By definition 37, $\mathcal{R}[\![\texttt{miracle}]\!]^\rho = \mathcal{R}[\![\langle \mathsf{true}, \mathsf{false}\rangle_k]\!]^\rho$.
By definitions 44 and 26, $\mathcal{R}[\![\langle \mathsf{true}, \mathsf{false}\rangle_k]\!]^\rho = \{(\lightning, \lightning)\}$.
By lemma 8, $(\lightning, \lightning) \in \mathcal{R}[\![\phi]\!]^\rho$. Therefore $\mathcal{R}[\![\texttt{miracle}]\!]^\rho \subseteq \mathcal{R}[\![\phi]\!]^\rho$.
By lemma 12, $(\mathcal{R}[\![\texttt{miracle}]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\phi]\!]^\rho)^\dagger$.
By lemma 16, $[\![\texttt{miracle}]\!]^\rho \subseteq [\![\phi]\!]^\rho$.
Thus, by definition 36, $\texttt{miracle} \sqsubseteq \phi$. □

Lemma 36. $\phi \sqsubseteq \texttt{abort}$

Proof. Let ρ such that it closes ϕ.
By definition 37, $\mathcal{R}[\![\texttt{abort}]\!]^\rho = \mathcal{R}[\![\langle \mathsf{false}, \mathsf{true}\rangle_k]\!]^\rho$.
By definitions 44 and 26, $\mathcal{R}[\![\langle \mathsf{false}, \mathsf{true}\rangle_k]\!]^\rho = (\textsc{Heap} \times \{\lightning\}) \cup \{(\lightning, \lightning)\}$.
By definition 34, and specifically rule (25),
$(\mathcal{R}[\![\langle \mathsf{false}, \mathsf{true}\rangle_k]\!]^\rho)^\dagger = \textsc{Move}^*; \textsc{Fault}^?$,
Thus, by lemma 16, $[\![\langle \mathsf{false}, \mathsf{true}\rangle_k]\!]^\rho = \textsc{Trace}$, is the set of all possible traces; the top element of the refinement lattice.
Therefore, $[\![\phi]\!]^\rho \subseteq [\![\texttt{abort}]\!]^\rho$, and by definition 36, $\phi \sqsubseteq \texttt{abort}$. □

Corollary 6 (MinMax). $\texttt{miracle} \sqsubseteq \phi \sqsubseteq \texttt{abort}$

Proof. By lemma 35 and lemma 36. □

Lemma 37 (EElim). $\phi\,[e/x] \sqsubseteq \exists x.\, \phi$

Proof. Let ρ such that it closes ϕ.
By definition 44, $\mathcal{R}[\![\exists x.\, \phi]\!]^\rho = \bigcup_v \mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]}$.
By lemma 9, $\mathcal{R}[\![\phi\,[e/x]]\!] = \mathcal{R}[\![\phi]\!]^{\rho[x \mapsto [\![e]\!]^\rho]}$.
Let $v' = [\![e]\!]^\rho$.
Then, $\mathcal{R}[\![\phi\,[e/x]]\!]^\rho \subseteq \mathcal{R}[\![\exists x.\, \phi]\!]^\rho$.
By lemma 12, $(\mathcal{R}[\![\phi\,[e/x]]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\exists x.\, \phi]\!]^\rho)^\dagger$.
By lemma 16, $[\![\phi\,[e/x]]\!]^\rho \subseteq [\![\exists x.\, \phi]\!]^\rho$.
Thus, by definition 36, $\phi\,[e/x] \sqsubseteq \exists x.\, \phi$. □

Lemma 38 (EIntro)**.** *If $x \notin free(\phi)$, then $\exists x.\, \phi \sqsubseteq \phi$*

Proof. Let ρ such that it closes ϕ.
Directly by definition 44 and $x \notin free(\phi)$, $\mathcal{R}[\![\exists x.\, \phi]\!]^\rho \subseteq \mathcal{R}[\![\phi]\!]^\rho$.
By lemma 12, $(\mathcal{R}[\![\exists x.\, \phi]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\phi]\!]^\rho)^\dagger$.
By lemma 16, $[\![\exists x.\, \phi]\!]^\rho \subseteq [\![\phi]\!]^\rho$.
Thus, by definition 36, $\exists x.\, \phi \sqsubseteq \phi$. □

Lemma 39 (AChoiceEq)**.** $\phi \sqcup \phi \equiv \phi$

Proof. Let ρ such that it closes ϕ.
$\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\phi]\!]^\rho = \mathcal{R}[\![\phi]\!]^\rho$.
By lemma 12, $(\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\phi]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi]\!]^\rho)^\dagger \cup (\mathcal{R}[\![\phi]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcup \phi]\!]^\rho = [\![\phi]\!]^\rho$.
Thus, by definition 36, $\phi \sqcup \phi \equiv \phi$. □

Lemma 40 (AChoiceId)**.** $\phi \sqcup \texttt{miracle} \equiv \phi$

Proof. Let ρ such that it closes ϕ.
By definitions, $\mathcal{R}[\![\texttt{miracle}]\!]^\rho = \{(\lightning, \lightning)\}$.
By lemma 8, $(\lightning, \lightning) \in \mathcal{R}[\![\phi]\!]^\rho$.
Therefore, $\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\texttt{miracle}]\!]^\rho = \mathcal{R}[\![\phi]\!]^\rho$.
Thus, $\mathcal{R}[\![\phi \sqcup \texttt{miracle}]\!]^\rho = \mathcal{R}[\![\phi]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\phi \sqcup \texttt{miracle}]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcup \texttt{miracle}]\!]^\rho = [\![\phi]\!]^\rho$.
Thus, by definition 36, $\phi \sqcup \texttt{miracle} \equiv \phi$. □

Lemma 41 (AChoiceAssoc)**.** $\phi \sqcup (\psi_1 \sqcup \psi_2) \equiv (\phi \sqcup \psi_1) \sqcup \psi_2$

Proof. Let ρ such that it closes ϕ, ψ_1 and ψ_2.
By associativity of set union, $\mathcal{R}[\![\phi]\!]^\rho \cup (\mathcal{R}[\![\psi_1]\!]^\rho \cup \mathcal{R}[\![\psi_2]\!]^\rho) = (\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi_1]\!]^\rho) \cup \mathcal{R}[\![\psi_2]\!]^\rho$.
Thus, $\mathcal{R}[\![\phi \sqcup (\psi_1 \sqcup \psi_2)]\!]^\rho = \mathcal{R}[\![(\phi \sqcup \psi_1) \sqcup \psi_2]\!]^\rho$.
Then, by lemma 12 $(\mathcal{R}[\![\phi \sqcup (\psi_1 \sqcup \psi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\phi \sqcup \psi_1) \sqcup \psi_2]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcup (\psi_1 \sqcup \psi_2)]\!]^\rho = [\![(\phi \sqcup \psi_1) \sqcup \psi_2]\!]^\rho$.
Thus, by definition 36, $\phi \sqcup (\psi_1 \sqcup \psi_2) \equiv (\phi \sqcup \psi_1) \sqcup \psi_2$. □

Lemma 42 (AChoiceComm)**.** $\phi \sqcup \psi \equiv \psi \sqcup \phi$

Proof. Let ρ such that it closes ϕ and ψ.
By commutativity of set union, $\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi]\!]^\rho = \mathcal{R}[\![\psi]\!]^\rho \cup \mathcal{R}[\![\phi]\!]^\rho$.
Therefore, $\mathcal{R}[\![\phi \sqcup \psi]\!]^\rho = \mathcal{R}[\![\psi \sqcup \phi]\!]^\rho$.
Then, by lemma 12 $(\mathcal{R}[\![\phi \sqcup \psi]\!]^\rho)^\dagger = (\mathcal{R}[\![\psi \sqcup \phi]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcup \psi]\!]^\rho = [\![\psi \sqcup \phi]\!]^\rho$.
Thus, by definition 36, $\phi \sqcup \psi \equiv \psi \sqcup \phi$. □

Lemma 43 (AChoiceElim)**.** $\phi \sqsubseteq \phi \sqcup \psi$

Proof. Let ρ such that it closes ϕ and ψ.
By set theory, $\mathcal{R}[\![\phi]\!]^\rho \subseteq \mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi]\!]^\rho$.
Therefore, $\mathcal{R}[\![\phi]\!]^\rho \subseteq \mathcal{R}[\![\phi \sqcup \psi]\!]^\rho$.
Thus, by lemma 12, $(\mathcal{R}[\![\phi]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\phi \sqcup \psi]\!]^\rho)^\dagger$.
Thus, by lemma 16, $[\![\phi]\!]^\rho \subseteq [\![\phi \sqcup \psi]\!]^\rho$.
Thus, by definition 36, $\phi \sqsubseteq \phi \sqcup \psi$. □

Lemma 44. $(T_1 \cup T_2); T_3 = (T_1; T_3) \cup (T_2; T_3)$

Proof. By definitions, $(T_1 \cup T_2); T_3 = \{st \mid s \in T_1 \cup T_2, t \in T_3\}$.
Also, $T_1; T_3 = \{st \mid s \in T_1, t \in T_3\}$.
Also, $T_2; T_3 = \{st \mid s \in T_2, t \in T_3\}$.
Then, $(T_1; T_3) \cup (T_2; T_3) = \{st \mid s \in T_1 \cup T_2, t \in T_3\}$ □

Corollary 7 (AChoiceDstR)**.** $(\phi_1 \sqcup \phi_2); \psi \equiv (\phi_1; \psi) \sqcup (\phi_2; \psi)$

Proof. Let ρ such that it closes ϕ and ψ.
By lemma 44, $(\mathcal{R}[\![\phi_1]\!]^\rho \cup \mathcal{R}[\![\phi_2]\!]^\rho) ; \mathcal{R}[\![\psi]\!]^\rho = (\mathcal{R}[\![\phi_1]\!]^\rho ; \mathcal{R}[\![\psi]\!]^\rho) \cup (\mathcal{R}[\![\phi_2]\!]^\rho ; \mathcal{R}[\![\psi]\!]^\rho)$.
Thus, $((\mathcal{R}[\![\phi_1]\!]^\rho \cup \mathcal{R}[\![\phi_2]\!]^\rho) ; \mathcal{R}[\![\psi]\!]^\rho)^\dagger = ((\mathcal{R}[\![\phi_1]\!]^\rho ; \mathcal{R}[\![\psi]\!]^\rho) \cup (\mathcal{R}[\![\phi_2]\!]^\rho ; \mathcal{R}[\![\psi]\!]^\rho))^\dagger$.
By definition 44, $(\mathcal{R}[\![(\phi_1 \sqcup \phi_1); \psi]\!]^\rho)^\dagger = (\mathcal{R}[\![(\phi_1; \psi) \sqcup (\phi_2; \psi)]\!]^\rho)^\dagger$.
By lemma 16, $[\![(\phi_1 \sqcup \phi_1); \psi]\!]^\rho = [\![(\phi_1; \psi) \sqcup (\phi_2; \psi)]\!]^\rho$.
Thus, by definition 36, $(\phi_1 \sqcup \phi_1); \psi \equiv (\phi_1; \psi) \sqcup (\phi_2; \psi)$. □

Lemma 45. $T_1; (T_2 \cup T_3) = (T_1; T_2) \cup (T_1; T_3)$

Proof. By definition, $T_1; (T_2 \cup T_3) = \{st \mid s \in T_1, t \in T_2 \cup T_3\}$.
Also, $T_1; T_2 = \{st \mid s \in T_1, t \in T_2\}$.
Also, $T_1; T_3 = \{st \mid s \in T_1, t \in T_3\}$.
Then, $(T_1; T_2) \cup (T_1; T_3) = \{st \mid s \in T_1, t \in T_2 \cup T_3\}$. □

Corollary 8 (AChoiceDstL)**.** $\psi; (\phi_1 \sqcup \phi_2) \equiv (\psi; \phi_1) \sqcup (\psi; \phi_2)$

Proof. Let ρ such that it closes ϕ_1, ϕ_2 and ψ.
By lemma 45, $\mathcal{R}[\![\psi]\!]^\rho ; (\mathcal{R}[\![\phi_1]\!]^\rho \cup \mathcal{R}[\![\phi_2]\!]^\rho) = (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_1]\!]^\rho) \cup (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_2]\!]^\rho)$.
Thus, $(\mathcal{R}[\![\psi]\!]^\rho ; (\mathcal{R}[\![\phi_1]\!]^\rho \cup \mathcal{R}[\![\phi_2]\!]^\rho))^\dagger = ((\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_1]\!]^\rho) \cup (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_2]\!]^\rho))^\dagger$.
By definition 44, $(\mathcal{R}[\![\psi; (\phi_1 \sqcup \phi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\psi; \phi_1) \sqcup (\psi; \phi_2)]\!]^\rho)^\dagger$.
By lemma 16, $[\![\psi; (\phi_1 \sqcup \phi_2)]\!]^\rho = [\![(\psi; \phi_1) \sqcup (\psi; \phi_2)]\!]^\rho$.
Thus, by definition 36, $\psi; (\phi_1 \sqcup \phi_2) \equiv (\psi; \phi_1) \sqcup (\psi; \phi_2)$. □

Lemma 46 (DChoiceEq)**.** $\phi \sqcap \phi \equiv \phi$

Proof. Let ρ such that it closes ϕ.
By set intersection, $\mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\phi]\!]^\rho = \mathcal{R}[\![\phi]\!]^\rho$.
Therefore, $\mathcal{R}[\![\phi \sqcap \phi]\!]^\rho = \mathcal{R}[\![\phi]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\phi \sqcap \psi]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcap \psi]\!]^\rho = [\![\phi]\!]^\rho$.
Thus, by definition 36, $\phi \sqcap \phi \equiv \phi$. □

Lemma 47 (DChoiceId)**.** $\phi \sqcap \texttt{abort} \equiv \phi$

Proof. Let ρ such that it closes ϕ.
By definitions, $[\![\texttt{abort}]\!]^\rho = \textsc{Trace}$.
Then, by the fact that Trace is the top element in the $\mathcal{P}(\textsc{Trace})$ lattice, and by set intersection, $[\![\phi]\!]^\rho \cap [\![\texttt{abort}]\!]^\rho = [\![\phi]\!]^\rho$.
Then, by lemma 12, $([\![\phi]\!]^\rho \cap [\![\texttt{abort}]\!]^\rho)^\dagger = ([\![\phi]\!]^\rho)^\dagger$.
Then, by lemma 15 and definition 35, $[\![\phi \sqcap \texttt{abort}]\!]^\rho = [\![\phi]\!]^\rho$.
Thus, by definition 36, $\phi \sqcap \texttt{abort} \equiv \phi$. □

Lemma 48 (DChoiceAssoc)**.** $\phi \sqcap (\psi_1 \sqcap \psi_2) \equiv (\phi \sqcap \psi_1) \sqcap \psi_2$

Proof. Let ρ such that it closes ϕ, ψ_1 and ψ_2.
By associativity of set intersection, $\mathcal{R}[\![\phi]\!]^\rho \cap (\mathcal{R}[\![\psi_1]\!]^\rho \cap \mathcal{R}[\![\psi_2]\!]^\rho) = (\mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\psi_1]\!]^\rho) \cap \mathcal{R}[\![\psi_2]\!]^\rho$.
Thus, $\mathcal{R}[\![\phi \sqcap (\psi_1 \sqcap \psi_2)]\!]^\rho = \mathcal{R}[\![(\phi \sqcap \psi_1) \sqcap \psi_2]\!]^\rho$.
Then, by lemma 12 $(\mathcal{R}[\![\phi \sqcap (\psi_1 \sqcap \psi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\phi \sqcap \psi_1) \sqcap \psi_2]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcap (\psi_1 \sqcap \psi_2)]\!]^\rho = [\![(\phi \sqcap \psi_1) \sqcap \psi_2]\!]^\rho$.
Thus, by definition 36, $\phi \sqcap (\psi_1 \sqcap \psi_2) \equiv (\phi \sqcap \psi_1) \sqcap \psi_2$. □

Lemma 49 (DChoiceComm)**.** $\phi \sqcap \psi \equiv \psi \sqcap \phi$

Proof. Let ρ such that it closes ϕ and ψ.
By commutativity of set intersection, $\mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\psi]\!]^\rho = \mathcal{R}[\![\psi]\!]^\rho \cap \mathcal{R}[\![\phi]\!]^\rho$.
Therefore, $\mathcal{R}[\![\phi \sqcap \psi]\!]^\rho = \mathcal{R}[\![\psi \sqcap \phi]\!]^\rho$.
Then, by lemma 12 $(\mathcal{R}[\![\phi \sqcap \psi]\!]^\rho)^\dagger = (\mathcal{R}[\![\psi \sqcap \phi]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcap \psi]\!]^\rho = [\![\psi \sqcap \phi]\!]^\rho$.
Thus, by definition 36, $\phi \sqcap \psi \equiv \psi \sqcap \phi$. □

Lemma 50 (DChoiceElim)**.** *If $\phi \sqsubseteq \psi_1$ and $\phi \sqsubseteq \psi_2$, then $\phi \sqsubseteq \psi_1 \sqcap \psi_2$.*

Proof. Assume premisses hold. Then,

$$
\begin{aligned}
\psi_1 \sqcap \psi_2 &\sqsupseteq \text{by first premiss and CMono} \\
&\quad \phi \sqcap \psi_2 \\
&\sqsupseteq \text{by second premiss and CMono} \\
&\quad \phi \sqcap \phi \\
&\equiv \text{by DChoiceEq} \\
&\quad \phi
\end{aligned}
$$

□

Lemma 51 (DChoiceIntro)**.** $\phi \sqcap \psi \sqsubseteq \phi$

Proof. Let ρ such that it closes ϕ and ψ.
By set intersection, $\mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\psi]\!]^\rho \subseteq \mathcal{R}[\![\phi]\!]^\rho$.
Then, by definition 44, $\mathcal{R}[\![\phi \sqcap \psi]\!]^\rho \subseteq \mathcal{R}[\![\phi]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\phi \sqcap \psi]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\phi]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcap \psi]\!]^\rho \subseteq [\![\phi]\!]^\rho$.
Thus, by definition 36, $\phi \sqcap \psi \sqsubseteq \phi$. □

Lemma 52. $(T_1 \cap T_2); T_3 = (T_1; T_3) \cap (T_2; T_3)$

Proof. By definitions, $(T_1 \cap T_2); T_3 = \{st \mid s \in T_1 \cap T_2, t \in T_3\}$.
Also, $T_1; T_3 = \{st \mid s \in T_1, t \in T_3\}$.
Also, $T_2; T_3 = \{st \mid s \in T_2, t \in T_3\}$.
Then, $(T_1; T_3) \cap (T_2; T_3) = \{st \mid s \in T_1 \cap T_2, t \in T_3\}$ □

Corollary 9 (DChoiceDstR)**.** $(\phi_1 \sqcap \phi_2); \psi \equiv (\phi_1; \psi) \sqcap (\phi_2; \psi)$.

Proof. Let ρ such that it closes ϕ_1, ϕ_2 and ψ.
By lemma 52, $\mathcal{R}[\![\psi]\!]^\rho ; (\mathcal{R}[\![\phi_1]\!]^\rho \cap \mathcal{R}[\![\phi_2]\!]^\rho) = (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_1]\!]^\rho) \cap (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_2]\!]^\rho)$.
Thus, $(\mathcal{R}[\![\psi]\!]^\rho ; (\mathcal{R}[\![\phi_1]\!]^\rho \cap \mathcal{R}[\![\phi_2]\!]^\rho))^\dagger = ((\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_1]\!]^\rho) \cap (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_2]\!]^\rho))^\dagger$.
By definition 44, $(\mathcal{R}[\![\psi; (\phi_1 \sqcap \phi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\psi; \phi_1) \sqcap (\psi; \phi_2)]\!]^\rho)^\dagger$.
By lemma 16, $[\![\psi; (\phi_1 \sqcap \phi_2)]\!]^\rho = [\![(\psi; \phi_1) \sqcap (\psi; \phi_2)]\!]^\rho$.
Thus, by definition 36, $\psi; (\phi_1 \sqcap \phi_2) \equiv (\psi; \phi_1) \sqcap (\psi; \phi_2)$. □

Lemma 53. $T_1; (T_2 \cap T_3) = (T_1; T_2) \cap (T_1; T_3)$

Proof. By definition, $T_1; (T_2 \cap T_3) = \{st \mid s \in T_1, t \in T_2 \cap T_3\}$.
Also, $T_1; T_2 = \{st \mid s \in T_1, t \in T_2\}$.
Also, $T_1; T_3 = \{st \mid s \in T_1, t \in T_3\}$.
Then, $(T_1; T_2) \cap (T_1; T_3) = \{st \mid s \in T_1, t \in T_2 \cap T_3\}$. □

Corollary 10 (DChoiceDstL). $\psi;(\phi_1 \sqcap \phi_2) \equiv (\psi;\phi_1) \sqcap (\psi;\phi_2)$

Proof. Let ρ such that it closes ϕ_1, ϕ_2 and ψ.
By lemma 45, $\mathcal{R}[\![\psi]\!]^\rho ; (\mathcal{R}[\![\phi_1]\!]^\rho \cap \mathcal{R}[\![\phi_2]\!]^\rho) = (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_1]\!]^\rho) \cap (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_2]\!]^\rho)$.
Thus, $(\mathcal{R}[\![\psi]\!]^\rho ; (\mathcal{R}[\![\phi_1]\!]^\rho \cap \mathcal{R}[\![\phi_2]\!]^\rho))^\dagger = ((\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_1]\!]^\rho) \cap (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\phi_2]\!]^\rho))^\dagger$.
By definition 44, $(\mathcal{R}[\![\psi;(\phi_1 \sqcap \phi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\psi;\phi_1) \sqcap (\psi;\phi_2)]\!]^\rho)^\dagger$.
By lemma 16, $[\![\psi;(\phi_1 \sqcap \phi_2)]\!]^\rho = [\![(\psi;\phi_1) \sqcap (\psi;\phi_2)]\!]^\rho$.
Thus, by definition 36, $\psi;(\phi_1 \sqcap \phi_2) \equiv (\psi;\phi_1) \sqcap (\psi;\phi_2)$. □

Lemma 54 (AChoiceDstD). $\phi \sqcup (\psi_1 \sqcap \psi_2) \equiv (\phi \sqcup \psi_1) \sqcap (\phi \sqcup \psi_2)$

Proof. Let ρ such that it closes ϕ, ψ_1 and ψ_2.
By distributivity of set union over set intersection,

$$\mathcal{R}[\![\phi]\!]^\rho \cup (\mathcal{R}[\![\psi_1]\!]^\rho \cap \mathcal{R}[\![\psi_2]\!]^\rho) = (\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi_1]\!]^\rho) \cap (\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi_2]\!]^\rho)$$

Therefore, by definition 44, $\mathcal{R}[\![\phi \sqcup (\psi_1 \sqcap \psi_2)]\!]^\rho = \mathcal{R}[\![(\phi \sqcup \psi_1) \sqcap (\phi \sqcup \psi_2)]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\phi \sqcup (\psi_1 \sqcap \psi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\phi \sqcup \psi_1) \sqcap (\phi \sqcup \psi_2)]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcup (\psi_1 \sqcap \psi_2)]\!]^\rho = [\![(\phi \sqcup \psi_1) \sqcap (\phi \sqcup \psi_2)]\!]^\rho$.
Thus, by definition 36, $\phi \sqcup (\psi_1 \sqcap \psi_2) \equiv (\phi \sqcup \psi_1) \sqcap (\phi \sqcup \psi_2)$. □

Lemma 55 (DChoiceDstA). $\phi \sqcap (\psi_1 \sqcup \psi_2) \equiv (\phi \sqcap \psi_1) \sqcup (\phi \sqcap \psi_2)$

Proof. Let ρ such that it closes ϕ, ψ_1 and ψ_2.
By distributivity of set intersection over set union,

$$\mathcal{R}[\![\phi]\!]^\rho \cap (\mathcal{R}[\![\psi_1]\!]^\rho \cup \mathcal{R}[\![\psi_2]\!]^\rho) = (\mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\psi_1]\!]^\rho) \cup (\mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\psi_2]\!]^\rho)$$

Therefore, by definition 44, $\mathcal{R}[\![\phi \sqcap (\psi_1 \sqcup \psi_2)]\!]^\rho = \mathcal{R}[\![(\phi \sqcap \psi_1) \sqcup (\phi \sqcap \psi_2)]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\phi \sqcap (\psi_1 \sqcup \psi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\phi \sqcap \psi_1) \sqcup (\phi \sqcap \psi_2)]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcap (\psi_1 \sqcup \psi_2)]\!]^\rho = [\![(\phi \sqcap \psi_1) \sqcup (\phi \sqcap \psi_2)]\!]^\rho$.
Thus, by definition 36, $\phi \sqcap (\psi_1 \sqcup \psi_2) \equiv (\phi \sqcap \psi_1) \sqcup (\phi \sqcap \psi_2)$. □

Lemma 56 (Absorb). $\phi \sqcup (\phi \sqcap \psi) \equiv \phi \equiv \phi \sqcap (\phi \sqcup \psi)$.

Proof. Let ρ such that it closes ϕ and ψ.
By the absorption property in the lattice of powersets,

$$\mathcal{R}[\![\phi]\!]^\rho \cup (\mathcal{R}[\![\phi]\!]^\rho \cap \mathcal{R}[\![\psi]\!]^\rho) = \mathcal{R}[\![\phi]\!]^\rho = \mathcal{R}[\![\phi]\!]^\rho \cap (\mathcal{R}[\![\phi]\!]^\rho \cup \mathcal{R}[\![\psi]\!]^\rho)$$

Therefore, by definition 44, $\mathcal{R}[\![\phi \sqcup (\phi \sqcap \psi)]\!]^\rho = \mathcal{R}[\![\phi]\!]^\rho = \mathcal{R}[\![\phi \sqcap (\phi \sqcup \psi)]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\phi \sqcup (\phi \sqcap \psi)]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi \sqcap (\phi \sqcup \psi)]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \sqcup (\phi \sqcap \psi)]\!]^\rho = [\![\phi]\!]^\rho = [\![\phi \sqcap (\phi \sqcup \psi)]\!]^\rho$.
Thus, by definition 36, $\phi \sqcup (\phi \sqcap \psi) \equiv \phi \equiv \phi \sqcap (\phi \sqcup \psi)$. □

Lemma 57 (Demonise). $\phi \sqcap \psi \sqsubseteq \phi \sqcup \psi$

Proof.

$$\begin{aligned} \phi \sqcap \psi &\sqsubseteq \text{ by DChoiceIntro} \\ &\quad \phi \\ &\sqsubseteq \text{ by AChoiceElim} \\ &\quad \phi \sqcup \psi \end{aligned}$$

□

Lemma 58 (ParSkip). $\phi \parallel \texttt{skip} \equiv \phi$

Proof. By definitions, and CLStutter and CLMumble rules. □

Lemma 59 (ParAssoc). $\phi \parallel (\psi_1 \parallel \psi_2) \equiv (\phi \parallel \psi_1) \parallel \psi_2$

Proof. Let ρ such that it closes ϕ, ψ_1 and ψ_2.
By lemma 17, $\mathcal{R}[\![\phi]\!]^\rho \parallel (\mathcal{R}[\![\psi_1]\!]^\rho \parallel \mathcal{R}[\![\psi_2]\!]^\rho) = (\mathcal{R}[\![\phi]\!]^\rho \parallel \mathcal{R}[\![\psi_1]\!]^\rho) \parallel \mathcal{R}[\![\psi_2]\!]^\rho$.
Therefore, by definition 44, $\mathcal{R}[\![\phi \parallel (\psi_1 \parallel \psi_2)]\!]^\rho = \mathcal{R}[\![(\phi \parallel \psi_1) \parallel \psi_2]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\phi \parallel (\psi_1 \parallel \psi_2)]\!]^\rho)^\dagger = (\mathcal{R}[\![(\phi \parallel \psi_1) \parallel \psi_2]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\phi \parallel (\psi_1 \parallel \psi_2)]\!]^\rho = [\![(\phi \parallel \psi_1) \parallel \psi_2]\!]^\rho$.
Thus, by definition 36, $\phi \parallel (\psi_1 \parallel \psi_2) \equiv (\phi \parallel \psi_1) \parallel \psi_2$. □

Lemma 60 (ParComm)**.** $\phi \parallel \psi \equiv \psi \parallel \phi$

Proof. Let ρ such that it closes ϕ and ψ.
By definition 35, $[\![\phi \parallel \psi]\!]^\rho = ([\![\phi]\!]^\rho \parallel [\![\psi]\!]^\rho)^\dagger$.
By lemma 17, $([\![\phi]\!]^\rho \parallel [\![\psi]\!]^\rho)^\dagger = ([\![\psi]\!]^\rho \parallel [\![\phi]\!]^\rho)^\dagger$.
Thus, by definition 35, $[\![\phi \parallel \psi]\!]^\rho = [\![\psi \parallel \phi]\!]^\rho$.
Therefore, by definition 36, $\phi \parallel \psi \equiv \psi \parallel \phi$. □

Lemma 61 (Exchange)**.** $(\phi \parallel \psi); (\phi' \parallel \psi') \sqsubseteq (\phi; \phi') \parallel (\psi; \psi')$

Proof. Let ρ such that is closes ϕ, ϕ', ψ, ψ'.
By definition 44, $\mathcal{R}[\![(\phi \parallel \psi); (\phi' \parallel \psi')]\!]^\rho = (\mathcal{R}[\![\phi]\!]^\rho \parallel \mathcal{R}[\![\psi]\!]^\rho); (\mathcal{R}[\![\phi']\!]^\rho \parallel \mathcal{R}[\![\psi']\!]^\rho)$.
Let $t_1 \in \mathcal{R}[\![\phi]\!]^\rho$, $u_1 \in \mathcal{R}[\![\psi]\!]^\rho$, $t_2 \in \mathcal{R}[\![\phi']\!]^\rho$, $u_2 \in \mathcal{R}[\![\psi']\!]^\rho$.
Let $s \in t_1 \parallel u_1$ and $w \in t_2 \parallel u_2$.
We prove that $sw \in t_1 t_2 \parallel u_1 u_2$ by induction on the derivation of $s \in t \parallel u$.
Base case: $(h, \wr) \in (h, \wr) \parallel u_1 u_2$, directly by rule (22).
Base case: $(h, h') u_1 u_2 \in (h, h') \parallel u_1 u_2$, directly by rule (21).
Case $(h, h') sw \in (h, h') t_1 t_2 \parallel u_1 u_2$:
By rule (20), $(h, h') s \in (h, h') t_1 \parallel u_1$.
Then, by the induction hypothesis: $(h, h') sw \in (h, h') t_1 t_2 \parallel u_1 u_2$.
Case $sw \in u_1 u_2 \parallel t_1 t_2$:
By rule (19), $s \in u_1 \parallel t_1$ and $w \in u_2 \parallel t_2$.
Then, by the induction hypothesis: $sw \in u_1 u_2 \parallel t_1 t_2$.
Therefore, $(\mathcal{R}[\![\phi]\!]^\rho \parallel \mathcal{R}[\![\psi]\!]^\rho); (\mathcal{R}[\![\phi']\!]^\rho \parallel \mathcal{R}[\![\psi']\!]^\rho) \subseteq (\mathcal{R}[\![\phi]\!]^\rho ; \mathcal{R}[\![\phi']\!]^\rho) \parallel (\mathcal{R}[\![\psi]\!]^\rho ; \mathcal{R}[\![\psi']\!]^\rho)$.
Thus, by definition 44, $\mathcal{R}[\![(\phi \parallel \psi); (\phi' \parallel \psi')]\!]^\rho \subseteq \mathcal{R}[\![(\phi; \phi') \parallel (\psi; \psi')]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![(\phi \parallel \psi); (\phi' \parallel \psi')]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![(\phi; \phi') \parallel (\psi; \psi')]\!]^\rho)^\dagger$.
By lemma 16, $[\![(\phi \parallel \psi); (\phi' \parallel \psi')]\!]^\rho \subseteq [\![(\phi; \phi') \parallel (\psi; \psi')]\!]^\rho$.
Therefore, by definition 36: $(\phi \parallel \psi); (\phi' \parallel \psi') \sqsubseteq (\phi; \phi') \parallel (\psi; \psi')$. □

Lemma 62. $(S \parallel T) \cup (S' \parallel T') \subseteq (S \cup S') \parallel (T \cup T')$

Proof. Immediate by definition 33. □

Corollary 11 (AChoiceExchange)**.** $(\phi \parallel \psi) \sqcup (\phi' \parallel \psi') \equiv (\phi \sqcup \phi') \parallel (\psi \sqcup \psi')$

Proof. Let ρ such that it closes ϕ, ϕ', ψ and ψ'.
By lemma 62, $(\mathcal{R}[\![\phi]\!]^\rho \parallel \mathcal{R}[\![\psi]\!]^\rho) \cup (\mathcal{R}[\![\phi']\!]^\rho \parallel \mathcal{R}[\![\psi']\!]^\rho) \subseteq (\mathcal{R}[\![\phi]\!]^\rho \sqcup \mathcal{R}[\![\phi']\!]^\rho) \parallel (\mathcal{R}[\![\psi]\!]^\rho \sqcup \mathcal{R}[\![\psi']\!]^\rho)$.
Therefore, by definition 44, $\mathcal{R}[\![(\phi \parallel \psi) \sqcup (\phi' \parallel \psi')]\!]^\rho \subseteq \mathcal{R}[\![(\phi \sqcup \phi') \parallel (\psi \sqcup \psi')]\!]^\rho$.
Then, by lemma 12, $\left(\mathcal{R}[\![(\phi \parallel \psi) \sqcup (\phi' \parallel \psi')]\!]^\rho\right)^\dagger \subseteq \left(\mathcal{R}[\![(\phi \sqcup \phi') \parallel (\psi \sqcup \psi')]\!]^\rho\right)^\dagger$.
Then, by lemma 16, $[\![(\phi \parallel \psi) \sqcup (\phi' \parallel \psi')]\!]^\rho \subseteq [\![(\phi \sqcup \phi') \parallel (\psi \sqcup \psi')]\!]^\rho$.
Thus, by definition 36, $(\phi \parallel \psi) \sqcup (\phi' \parallel \psi') \sqsubseteq (\phi \sqcup \phi') \parallel (\psi \sqcup \psi')$. □

Lemma 63 (SeqPar)**.** $\phi; \psi \sqsubseteq \phi \parallel \psi$

Proof.

$$\begin{aligned}
\phi;\psi \equiv & \text{ by ParSkip and CMono} \\
& (\phi \parallel \texttt{skip});(\psi \parallel \texttt{skip}) \\
\sqsubseteq & \text{ by Exchange and CMono rules} \\
& (\phi;\texttt{skip}) \parallel (\psi;\texttt{skip}) \\
\equiv & \text{ by Skip and CMono} \\
& \phi \parallel \psi
\end{aligned}$$

□

Lemma 64 (ParDstLR)**.** $\phi;(\psi_1 \parallel \psi_2) \sqsubseteq (\phi;\psi_1) \parallel \psi_2$

Proof.

$$\begin{aligned}
\phi;(\psi_1 \parallel \psi_2) \equiv & \text{ by ParSkip and CMono} \\
& (\phi \parallel \texttt{skip});(\psi_1 \parallel \psi_2) \\
\sqsubseteq & \text{ by Exchange} \\
& (\phi;\psi_1) \parallel (\texttt{skip};\psi_2) \\
\equiv & \text{ by Skip and CMono} \\
& (\phi;\psi_1) \parallel \psi_2
\end{aligned}$$

□

Lemma 65 (ParDstLL)**.** $\phi;(\psi_1 \parallel \psi_2) \sqsubseteq \psi_1 \parallel (\phi;\psi_2)$

Proof.

$$\begin{aligned}
\phi;(\psi_1 \parallel \psi_2) \equiv & \text{ by ParSkip, ParComm and CMono} \\
& (\texttt{skip} \parallel \phi);(\psi_1 \parallel \psi_2) \\
\sqsubseteq & \text{ by Exchange} \\
& (\texttt{skip};\psi_1) \parallel (\phi;\psi_2) \\
\equiv & \text{ by Skip and CMono} \\
& \psi_1 \parallel (\phi;\psi_2)
\end{aligned}$$

□

Lemma 66 (ParDstRL)**.** $(\phi \parallel \psi_1);\psi_2 \sqsubseteq \phi \parallel (\psi_1;\psi_2)$

Proof.

$$\begin{aligned}
(\phi \parallel \psi_1);\psi_2 \equiv & \text{ by ParSkip, ParComm and CMono} \\
& (\phi \parallel \psi_1);(\texttt{skip} \parallel \psi_2) \\
\sqsubseteq & \text{ by Exchange} \\
& (\phi;\texttt{skip}) \parallel (\psi_1;\psi_2) \\
\equiv & \text{ by Skip and CMono} \\
& \phi \parallel (\psi_1;\psi_2)
\end{aligned}$$

□

Lemma 67 (ParDstRR)**.** $(\phi \parallel \psi_1);\psi_2 \sqsubseteq (\phi;\psi_2) \parallel \psi_1$

Proof.

$$\begin{aligned}
(\phi \parallel \psi_1); \psi_2 &\equiv \text{by SKIP and CMONO} \\
&\quad (\phi \parallel \psi_1); (\psi_2 \parallel \texttt{skip}) \\
&\sqsubseteq \text{by EXCHANGE} \\
&\quad (\phi; \psi_2) \parallel (\psi_1; \texttt{skip}) \\
&\equiv \text{by SKIP and CMONO} \\
&\quad (\phi; \psi_2) \parallel \psi_1
\end{aligned}$$

□

Lemma 68 (ACHOICEEQ)**.** $\exists x.\, \phi \equiv \bigsqcup_{v \in \text{VAL}} \phi\,[v/x]$

Proof. Let ρ such that it closes ϕ.
By definition 44, $\mathcal{R}[\![\exists x.\, \phi]\!]^\rho = \bigcup_{v \in \text{VAL}} \mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]}$.
By lemma 10, $\mathcal{R}[\![\exists x.\, \phi]\!]^\rho = \bigcup_{v \in \text{VAL}} \mathcal{R}[\![\phi\,[v/x]]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\exists x.\, \phi]\!]^\rho)^\dagger = \left(\bigcup_{v \in \text{VAL}} \mathcal{R}[\![\phi\,[v/x]]\!]^\rho\right)^\dagger$.
Therefore, by definition 44, $(\mathcal{R}[\![\exists x.\, \phi]\!]^\rho)^\dagger = \left(\mathcal{R}\left[\!\left[\bigsqcup_{v \in \text{VAL}} \phi\,[v/x]\right]\!\right]^\rho\right)^\dagger$.
Then, by lemma 16, $[\![\exists x.\, \phi]\!]^\rho = \left[\!\left[\bigsqcup_{v \in \text{VAL}} \phi\,[v/x]\right]\!\right]^\rho$.
Thus, by definition 36, $\exists x.\, \phi \equiv \bigsqcup_{v \in \text{VAL}} \phi\,[v/x]$. □

Lemma 69 (ESEQEXT)**.** *If $x \notin free(\phi)$, then $\exists x.\, \phi; \psi \equiv \phi; \exists x.\, \psi$.*

Proof. Let ρ such that it closes ϕ and ψ.
By definition 44, $\mathcal{R}[\![\exists x.\, \phi; \psi]\!]^\rho = \bigcup_v \mathcal{R}[\![\phi; \psi]\!]^{\rho[x \mapsto v]} = \bigcup_v \left(\mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]}; \mathcal{R}[\![\psi]\!]^{\rho[x \mapsto v]}\right)$.
By the premiss, $\mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]} = \mathcal{R}[\![\phi]\!]^\rho$.
Therefore, $\bigcup_v \left(\mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]}; \mathcal{R}[\![\psi]\!]^{\rho[x \mapsto v]}\right) = \mathcal{R}[\![\phi]\!]^\rho; \bigcup_v \mathcal{R}[\![\psi]\!]^{\rho[x \mapsto v]}$.
Thus, by definition 44, $\mathcal{R}[\![\exists x.\, \phi; \psi]\!]^\rho = \mathcal{R}[\![\phi; \exists x.\, \psi]\!]^\rho$.
By lemma 12, $(\mathcal{R}[\![\exists x.\, \phi; \psi]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi; \exists x.\, \psi]\!]^\rho)^\dagger$.
By lemma 16, $[\![\exists x.\, \phi; \psi]\!]^\rho = [\![\phi; \exists x.\, \psi]\!]^\rho$.
Thus, by definition 36, $\exists x.\, \phi; \psi \equiv \phi; \exists x.\, \psi$. □

Lemma 70 (ESEQDST)**.** $\exists x.\, \phi; \psi \sqsubseteq (\exists x.\, \phi); (\exists x.\, \psi)$.

Proof. Let ρ such that it closes ϕ and ψ modulo x.
By definitions, $\bigcup_{v \in \text{VAL}} \mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]}; \mathcal{R}[\![\psi]\!]^{\rho[x \mapsto v]} \subseteq \left(\bigcup_{v \in \text{VAL}} \mathcal{R}[\![\phi]\!]^{\rho[x \mapsto v]}\right); \left(\bigcup_{v \in \text{VAL}} \mathcal{R}[\![\psi]\!]^{\rho[x \mapsto v]}\right)$.
By definition 44, $\mathcal{R}[\![\exists x.\, \phi; \psi]\!]^\rho \subseteq \mathcal{R}[\![(\exists x.\, \phi); (\exists x.\, \psi)]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\exists x.\, \phi; \psi]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![(\exists x.\, \phi); (\exists x.\, \psi)]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\exists x.\, \phi; \psi]\!]^\rho \subseteq [\![(\exists x.\, \phi); (\exists x.\, \psi)]\!]^\rho$.
Thus, by definition 36, $\exists x.\, \phi; \psi \sqsubseteq (\exists x.\, \phi); (\exists x.\, \psi)$. □

Lemma 71 (EACHOICEDST)**.** $\exists x.\, \phi \sqcup \psi \equiv (\exists x.\, \phi) \sqcup (\exists x.\, \psi)$

Proof. Direct, via EACHOICEEQ and ACHOICEASSOC. □

Lemma 72 (EPARDST)**.** $\exists x.\, \phi \parallel \psi \equiv (\exists x.\, \phi) \parallel (\exists x.\, \psi)$

Proof. Direct, via EACHOICEEQ and ACHOICEEXCHANGE. □

Lemma 73 (CMONO)**.** *Let C be a specification context. If $\phi \sqsubseteq \psi$, then $C[\phi] \sqsubseteq C[\psi]$.*

Proof. By straightforward induction on $C[-]$.

□

Lemma 74 (FAPPLYELIM)**.** $\phi\,[e/x] \equiv (\lambda x.\, \phi)\, e$

Proof. Let ρ such that it closes ϕ, modulo x.
By definition 35, $[\![(\lambda x.\, \phi)\, e]\!]^\rho = [\![\phi]\!]^{\rho[x \mapsto [\![e]\!]^\rho]}$.
Then, by lemma 10, $[\![(\lambda x.\, \phi)\, e]\!]^\rho = [\![\phi\, [e/x]]\!]^\rho$.
Thus, by definition 36, $\phi\, [e/x] \equiv (\lambda x.\, \phi)\, e$. □

Lemma 75 (FApplyElimRec)**.** $\phi\, [(\mu A.\, \lambda x.\, \phi)\, /A]\, [e/x] \equiv (\mu A.\, \lambda x.\, \phi)\, e$

Proof. Let ρ such that it closes ϕ, modulo x and A, and e.
By definition 44, $\mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^\rho = (\mathcal{R}[\![\mu A.\, \lambda x.\, \phi]\!]^\rho)\, [\![e]\!]^\rho$.
Then, by lemma 27, $\mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^\rho = \left(\mathcal{R}[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto \mathcal{R}[\![\mu A.\lambda x.\phi]\!]^\rho]}\right) [\![e]\!]^\rho$.
By definition 44, $\mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^\rho = \mathcal{R}[\![\phi]\!]^{\rho[A \mapsto \mathcal{R}[\![\mu A.\lambda x.\phi]\!]^\rho][x \mapsto [\![e]\!]^\rho]}$.
Then, by lemma 9, $\mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^\rho = \mathcal{R}[\![\phi\, [(\mu A.\, \lambda x.\, \phi)\, /A]]\!]^{\rho[x \mapsto [\![e]\!]^\rho]}$.
Then, by lemma 10, $\mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^\rho = \mathcal{R}[\![\phi\, [(\mu A.\, \lambda x.\, \phi)\, /A]\, [e/x]]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^\rho)^\dagger = (\mathcal{R}[\![\phi\, [(\mu A.\, \lambda x.\, \phi)\, /A]\, [e/x]]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![(\mu A.\, \lambda x.\, \phi)\, e]\!]^\rho = [\![\phi\, [(\mu A.\, \lambda x.\, \phi)\, /A]\, [e/x]]\!]^\rho$.
Thus, by definition 36, $\phi\, [(\mu A.\, \lambda x.\, \phi)\, /A]\, [e/x] \equiv (\mu A.\, \lambda x.\, \phi)\, e$. □

Lemma 76 (FElim)**.** $F_l \equiv \lambda x.\, F_l x$

Proof. Let ρ such that it closes F_l.
Case analysis on F_l.
Case $\lambda x.\, \phi$, immediate by definition 35.
Case $\mu A.\, \lambda x.\, \phi$:
By definition 44, $\mathcal{R}[\![\lambda x.\, (\mu A.\, \lambda x.\, \phi)\, x]\!]^\rho = \lambda v.\, (\mathcal{R}[\![\mu A.\, \lambda x.\, \phi]\!]^\rho)\, v$.
By lemma 27, $\mathcal{R}[\![\lambda x.\, (\mu A.\, \lambda x.\, \phi)\, x]\!]^\rho = \lambda v.\, \left(\mathcal{R}[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto \mathcal{R}[\![\mu A.\lambda x.\phi]\!]^\rho]}\right) v$.
Then, by definition 44, $\mathcal{R}[\![\lambda x.\, (\mu A.\, \lambda x.\, \phi)\, x]\!]^\rho = \mathcal{R}[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto \mathcal{R}[\![\mu A.\lambda x.\phi]\!]^\rho]}$
Therefore, by lemma 27, $\mathcal{R}[\![\lambda x.\, (\mu A.\, \lambda x.\, \phi)\, x]\!]^\rho = \mathcal{R}[\![\mu A.\, \lambda x.\, \phi]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\lambda x.\, (\mu A.\, \lambda x.\, \phi)\, x]\!]^\rho)^\dagger = (\mathcal{R}[\![\mu A.\, \lambda x.\, \phi]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\lambda x.\, (\mu A.\, \lambda x.\, \phi)\, x]\!]^\rho = [\![\mu A.\, \lambda x.\, \phi]\!]^\rho$.
Thus, by definition 36, $F_l \equiv \lambda x.\, F_l x$. □

Lemma 77 (FRename)**.** *If* $\phi\, [e_1/x] \sqsubseteq \phi\, [e_2/x]$*, then* $(\lambda x.\, \phi)\, e_1 \sqsubseteq (\lambda x.\, \phi)\, e_2$*.*

Proof. By FApplyElim and CMono. □

Lemma 78 (FRenameRec)**.** *If* $\phi\, [(\mu A.\, \lambda x.\, \phi)\, /A]\, [e_1/x] \sqsubseteq \phi\, [(\mu A.\, \lambda x.\, \phi)\, /A]\, [e_2/x]$*, then* $(\mu A.\, \lambda x.\, \phi)\, e_1 \sqsubseteq (\mu A.\, \lambda x.\, \phi)\, e_2$*.*

Proof. By FApplyElimRec and CMono. □

Lemma 79 (FuncIntro)**.** *If* $x \notin free(\phi)$*, then* $(\lambda x.\, \phi)\, () \equiv \phi$*.*

Proof. Immediate by definition 35. □

Lemma 80 (Inline)**.** $\phi\, [F/f] \equiv \mathbf{let}\ f = F\ \mathbf{in}\ \phi$

Proof. Immediate by definition 35 and lemma 9. □

Lemma 81 (Ind)**.** *If* $\lambda x.\, \phi\, [\lambda x.\, \psi/A] \sqsubseteq \lambda x.\, \psi$*, then* $\mu A.\, \lambda x.\, \phi \sqsubseteq \lambda x.\, \psi$*.*

Proof. By lemma 4, the function $[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto -]} : (\textsc{Val} \to \mathcal{P}(\textsc{Trace})) \to (\textsc{Val} \to \mathcal{P}(\textsc{Trace}))$ is monotonic. Furthermore, the function space $\textsc{Val} \to \mathcal{P}(\textsc{Trace})$ is a complete lattice.
By the premiss and definition 36, for all closing ρ, $[\![\lambda x.\, \phi\, [\lambda x.\, \psi/A]]\!]^\rho \subseteq [\![\lambda x.\, \psi]\!]^\rho$.
By lemma 9, $[\![\lambda x.\, \phi]\!]^{\rho[A \mapsto [\![\lambda x.\psi]\!]^\rho]} \subseteq [\![\lambda x.\, \psi]\!]^\rho$.
Then, by the fixpoint induction theorem, $\mu\, [\![\lambda x.\, \phi]\!]^{\rho[A \mapsto -]} \subseteq [\![\lambda x.\, \psi]\!]^\rho$.
Thus, by definition 35, $[\![\mu A.\, \lambda x.\, \phi]\!]^\rho \sqsubseteq [\![\lambda x.\, \psi]\!]^\rho$.
Therefore, by definition 36, $\mu A.\, \lambda x.\, \phi \sqsubseteq \lambda x.\, \psi$. □

Lemma 82 (UnrollR)**.** *If* $A \notin free(\phi;) \cup free(\psi)$*, then*
$(\mu A.\, \lambda x.\, \psi \sqcup \phi; Ae')\, e \equiv \psi\, [e/x] \sqcup \phi\, [e/x]\, ; (\mu A.\, \lambda x.\, \psi \sqcup \phi; Ae'')\, e'$.

Proof. By FApplyElimRec. □

Lemma 83 (UnrollL). *If $A \notin free(\phi;) \cup free(\psi)$, then*
$(\mu A.\, \lambda x.\, \psi \sqcup Ae';\phi)\, e \equiv \psi\,[e/x] \sqcup \phi\,[e/x]\,;(\mu A.\, \lambda x.\, \psi \sqcup Ae'';\phi)\, e'$.

Proof. By FApplyElimRec. □

Lemma 84 (RecSeq). *If $A \notin free(\phi) \cup free(\psi_1) \cup free(\psi_2)$, then*

$$(\mu A.\, \lambda x.\, \psi_1 \sqcup \phi; Ae')\, e; \psi_2\,[e/x] \equiv (\mu A.\, \lambda x.\, \psi_1; \psi_2 \sqcup \phi; Ae')\, e$$

Proof. First, we have the following:

$$\begin{aligned}
&\lambda x.\, \psi_1; \psi_2 \sqcup \phi; (\lambda x.\, (\mu A.\, \lambda x.\, \psi_1 \sqcup \phi; Ae')\, x; \psi_2\,[e/x])\, e' \\
\equiv\ &\text{by FApplyElim} \\
&\lambda x.\, \psi_1; \psi_2 \sqcup \phi; (\mu A.\, \lambda x.\, \psi_1 \sqcup \phi; Ae')\, e'; \psi_2\,[e/x] \\
\equiv\ &\text{by AChoiceDstR} \\
&\lambda x.\, (\psi_1 \sqcup \phi; (\mu A.\, \lambda x.\, \psi_1 \sqcup \phi; Ae')\, e')\,; \psi_2\,[e/x] \\
\equiv\ &\text{by UnrollR and CMono} \\
&\lambda x.\, (\mu A.\, \lambda x.\, \psi_1 \sqcup \phi; Ae')\, x; \psi_2\,[e/x]
\end{aligned}$$

Therefore, by Ind, $\mu A.\, \lambda x.\, \psi_1; \psi_2 \sqcup \phi; Ae' \equiv \lambda x.\, (\mu A.\, \lambda x.\, \psi_1 \sqcup \phi; Ae')\, x; \psi_2\,[e/x]$.
Then, by CMono and FApplyElim, $(\mu A.\, \lambda x.\, \psi_1; \psi_2 \sqcup \phi; Ae')\, e \equiv (\mu A.\, \lambda x.\, \psi_1 \sqcup \phi; Ae')\, e; \psi_2\,[e/x]$. □

C Primitive Atomic Refinement Laws Proofs

Lemma 85 (AUElim). $\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall y, \vec{x}.\, \langle P, Q\rangle_k$

Proof. Let ρ such that it closes $\forall \vec{x}.\, \langle P, Q\rangle_k$. Let $\vec{v} \in \overrightarrow{\text{Val}}$, of the same length as $\vec{x}$ and let $v \in$ Val. By pointwise extension, $(\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \subseteq (\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}][y\mapsto v]}$, and $(\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \subseteq (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}][y\mapsto v]}$.
Therefore, by definition 44, $\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall y, \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho)^\dagger \subseteq \mathcal{R}[\![\forall y, \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho$.
Then, by lemma 16, $[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho \subseteq [\![\forall y, \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho$.
Thus, by definition 36, $\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall y, \vec{x}.\, \langle P, Q\rangle_k$. □

Lemma 86 (AEarly). *If $x \notin free(P)$, then $\exists x. \forall \vec{y}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{y}.\, \langle P, \exists x.\, Q\rangle_k$*

Proof. Let ρ such that it closes both specifications. Let $\vec{v} \in \overrightarrow{\text{Val}}$ and $v \in$ Val.
By premiss and definition 26, $\mathsf{a}\big((\!|P|\!)^{\rho[x\mapsto v][\vec{y}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[x\mapsto v][\vec{y}\mapsto\vec{v}]}\big)_k = \mathsf{a}\big((\!|P|\!)^{\rho[\vec{y}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[x\mapsto v][\vec{y}\mapsto\vec{v}]}\big)_k$.
By definitions, $\mathsf{a}\big((\!|P|\!)^{\rho[\vec{y}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[x\mapsto v][\vec{y}\mapsto\vec{v}]}\big)_k \subseteq \mathsf{a}\big((\!|P|\!)^{\rho[\vec{y}\mapsto\vec{v}]}, (\!|\exists x.\, Q|\!)^{\rho[\vec{y}\mapsto\vec{v}]}\big)_k$.
Thus, by definition 44, $\mathcal{R}[\![\forall \vec{y}.\, \langle P, Q\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{y}.\, \langle P, \exists x.\, Q\rangle_k]\!]^\rho$.
Then, by lemma 12, lemma 16 and definition 36, $\forall \vec{y}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{y}.\, \langle P, \exists x.\, Q\rangle_k$.
Then, by CMono, $\exists x. \forall \vec{y}.\, \langle P, Q\rangle_k \sqsubseteq \exists x. \forall \vec{y}.\, \langle P, \exists x.\, Q\rangle_k$.
By the premiss and EIntro, $\exists x. \forall \vec{y}.\, \langle P, \exists x.\, Q\rangle_k \sqsubseteq \forall \vec{y}.\, \langle P, \exists x.\, Q\rangle_k$.
Thus, by Trans, $\exists x. \forall \vec{y}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{y}.\, \langle P, \exists x.\, Q\rangle_k$. □

Lemma 87 (AEElim). $\forall \vec{y}, x.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{y}.\, \langle \exists x.\, P, \exists x.\, Q\rangle_k$

Proof. Let ρ such that it closes both specifications. Let $\vec{v} \in \overrightarrow{\text{Val}}$ and $v \in$ Val.
By definitions 26 and 22, $\mathsf{a}\big((\!|P|\!)^{\rho[\vec{y}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{y}\mapsto\vec{v}]}\big)_k \subseteq \mathsf{a}\big((\!|\exists x.\, P|\!)^{\rho[\vec{y}\mapsto\vec{v}]}, (\!|\exists x.\, Q|\!)^{\rho[\vec{y}\mapsto\vec{v}]}\big)_k$.
Therefore, by definition 44, $\mathcal{R}[\![\forall \vec{y}, x.\, \langle P, Q\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{y}.\, \langle \exists x.\, P, \exists x.\, Q\rangle_k]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\forall \vec{y}, x.\, \langle P, Q\rangle_k]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\forall \vec{y}.\, \langle \exists x.\, P, \exists x.\, Q\rangle_k]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\forall \vec{y}, x.\, \langle P, Q\rangle_k]\!]^\rho \subseteq [\![\forall \vec{y}.\, \langle \exists x.\, P, \exists x.\, Q\rangle_k]\!]^\rho$.
Thus, by definition 36, $\forall \vec{y}, x.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{y}.\, \langle \exists x.\, P, \exists x.\, Q\rangle_k$. □

Lemma 88 (ADisjunction). $\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcup \forall \vec{x}.\, \langle P_2, Q_2\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P_1 \vee P_2, Q_1 \vee Q_2\rangle_k$

Proof. Let ρ such that it closes both specifications.
By definition 26, $\mathsf{a}(\llparenthesis P_1 \rrparenthesis^\rho, \llparenthesis Q_1 \rrparenthesis^\rho)_k \cup \mathsf{a}(\llparenthesis P_2 \rrparenthesis^\rho, \llparenthesis Q_2 \rrparenthesis^\rho)_k \subseteq \mathsf{a}(\llparenthesis P_1 \vee P_2 \rrparenthesis^\rho, \llparenthesis Q_1 \vee Q_2 \rrparenthesis^\rho)_k$.
Therefore, by definition 44, $\mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k]\!]^\rho \cup \mathcal{R}[\![\forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{x}.\, \langle P_1 \vee P_2, Q_1 \vee Q_2\rangle_k]\!]^\rho$.
Thus, $\mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcup \forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{x}.\, \langle P_1 \vee P_2, Q_1 \vee Q_2\rangle_k]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcup \forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\forall \vec{x}.\, \langle P_1 \vee P_2, Q_1 \vee Q_2\rangle_k]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcup \forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho \subseteq [\![\forall \vec{x}.\, \langle P_1 \vee P_2, Q_1 \vee Q_2\rangle_k]\!]^\rho$.
Thus, by definition 36, $\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcup \forall \vec{x}.\, \langle P_2, Q_2\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P_1 \vee P_2, Q_1 \vee Q_2\rangle_k$. □

Lemma 89 (AConjunction)**.** $\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcap \forall \vec{x}.\, \langle P_2, Q_2\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P_1 \wedge P_2, Q_1 \wedge Q_2\rangle_k$

Proof. Let ρ such that it closes both specifications.
By definition 26, $\mathsf{a}(\llparenthesis P_1 \rrparenthesis^\rho, \llparenthesis Q_1 \rrparenthesis^\rho)_k \cap \mathsf{a}(\llparenthesis P_2 \rrparenthesis^\rho, \llparenthesis Q_2 \rrparenthesis^\rho)_k \subseteq \mathsf{a}(\llparenthesis P_1 \wedge P_2 \rrparenthesis^\rho, \llparenthesis Q_1 \wedge Q_2 \rrparenthesis^\rho)_k$.
Therefore, by definition 44, $\mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k]\!]^\rho \cap \mathcal{R}[\![\forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{x}.\, \langle P_1 \wedge P_2, Q_1 \wedge Q_2\rangle_k]\!]^\rho$.
Thus, $\mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcap \forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{x}.\, \langle P_1 \wedge P_2, Q_1 \wedge Q_2\rangle_k]\!]^\rho$.
Then, by lemma 12, $(\mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcap \forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\forall \vec{x}.\, \langle P_1 \wedge P_2, Q_1 \wedge Q_2\rangle_k]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcap \forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho \subseteq [\![\forall \vec{x}.\, \langle P_1 \wedge P_2, Q_1 \wedge Q_2\rangle_k]\!]^\rho$.
Thus, by definition 36, $\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \sqcap \forall \vec{x}.\, \langle P_2, Q_2\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P_1 \wedge P_2, Q_1 \wedge Q_2\rangle_k$. □

Lemma 90 (AFrame)**.** $\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P * R, Q * R\rangle_k$

Proof. Let ρ such that it closes both specifications.
Let $p, r \in$ View. By definition 19, $p \leq p * r$.
Therefore, by definition 26:

$$
\begin{aligned}
\mathsf{a}(\llparenthesis P \rrparenthesis^\rho, \llparenthesis Q \rrparenthesis^\rho)_k(h) &= \left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in \llparenthesis P \rrparenthesis^\rho * r.\, h \in [\![w]\!]_{k;} \\ \wedge\, \exists w'.\, w\ G_{k;}\ w' \wedge h' \in [\![w']\!]_{k;} \wedge w' \in \llparenthesis Q \rrparenthesis^\rho * r \end{array} \right\} \\
&\subseteq \left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in \llparenthesis P \rrparenthesis^\rho * \llparenthesis R \rrparenthesis^\rho * r.\, h \in [\![w]\!]_{k;} \\ \wedge\, \exists w'.\, w\ G_{k;}\ w' \wedge h' \in [\![w']\!]_{k;} \wedge w' \in \llparenthesis Q \rrparenthesis^\rho * \llparenthesis R \rrparenthesis^\rho * r \end{array} \right\} \\
&= \text{by definition 22} \\
&\quad \left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in \llparenthesis P * R \rrparenthesis^\rho * r.\, h \in [\![w]\!]_{k;} \\ \wedge\, \exists w'.\, w\ G_{k;}\ w' \wedge h' \in [\![w']\!]_{k;} \wedge w' \in \llparenthesis Q * R \rrparenthesis^\rho * r \end{array} \right\} \\
&= \mathsf{a}(\llparenthesis P * R \rrparenthesis^\rho, \llparenthesis Q * R \rrparenthesis^\rho)_k(h)
\end{aligned}
$$

Therefore, by definition 44, $\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{x}.\, \langle P * R, Q * R\rangle_k]\!]^\rho$.
By lemma 12, $(\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\forall \vec{x}.\, \langle P * R, Q * R\rangle_k]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho \subseteq [\![\forall \vec{x}.\, \langle P * R, Q * R\rangle_k]\!]^\rho$.
Thus, by definition 36, $\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P * R, Q * R\rangle_k$. □

Lemma 91 (AStutter)**.** $\forall \vec{x}.\, \langle P, P\rangle_k ; \forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k$

Proof. Let ρ such that it closes $\forall \vec{x}.\, \langle P, Q\rangle_k$.
By definition 34, and specifically the CLStutter rule,
$(\mathcal{R}[\![\forall \vec{x}.\, \langle P, P\rangle_k ; \forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho)^\dagger \subseteq (\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho)^\dagger$.
Then, by lemma 16, $[\![\forall \vec{x}.\, \langle P, P\rangle_k ; \forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho \subseteq [\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho$.
Thus, by definition 36, $\forall \vec{x}.\, \langle P, P\rangle_k ; \forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k$. □

Lemma 92 (AMumble)**.** $\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, P'\rangle_k ; \forall \vec{x}.\, \langle P', Q\rangle_k$

Proof. Let ρ such that it closes both specifications.
By definition 34, and specifically the CLMumble rule,
$(\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho)^\dagger \subseteq \big(\mathcal{R}[\![\forall \vec{x}.\, \langle P, P'\rangle_k ; \forall \vec{x}.\, \langle P', Q\rangle_k]\!]^\rho\big)^\dagger$.
Then, by lemma 16, $[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho \subseteq [\![\forall \vec{x}.\, \langle P, P'\rangle_k ; \forall \vec{x}.\, \langle P', Q\rangle_k]\!]^\rho$.
Thus, by definition 36, $\forall \vec{x}.\, \langle P, Q\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, P'\rangle_k ; \forall \vec{x}.\, \langle P', Q\rangle_k$. □

Lemma 93 (AInterleave)**.**

$$
\begin{aligned}
&\forall \vec{x}.\, \langle P_1, Q_1\rangle_k \parallel \forall \vec{x}.\, \langle P_2, Q_2\rangle_k \\
&\sqsubseteq (\forall \vec{x}.\, \langle P_1, Q_1\rangle_k ; \forall \vec{x}.\, \langle P_2, Q_2\rangle_k) \sqcup (\forall \vec{x}.\, \langle P_2, Q_2\rangle_k ; \forall \vec{x}.\, \langle P_1, Q_1\rangle_k)
\end{aligned}
$$

Proof. Let ρ such that it closes both specifications. Let $m_1, m_2 \in (\textsc{Heap} \times \textsc{Heap}^{\lightning}) \cup \{\lightning, \lightning\}$. Then, by definition 33, $m_1 \parallel m_2 = \{m_1 m_2, m_2 m_1\}$.
Therefore, by definition 44,

$$\begin{aligned}&\mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k]\!]^\rho \parallel \mathcal{R}[\![\forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho \\ =\ &(\mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k]\!]^\rho ; \mathcal{R}[\![\forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho) \cup (\mathcal{R}[\![\forall \vec{x}.\, \langle P_2, Q_2\rangle_k]\!]^\rho ; \mathcal{R}[\![\forall \vec{x}.\, \langle P_1, Q_1\rangle_k]\!]^\rho)\end{aligned}$$

Then, the result is established by lemma 12, lemma 16 and definition 36. □

Lemma 94 (ACons)**.** *If $P \Rightarrow P'$ and $Q' \Rightarrow Q$, then $\forall \vec{x}.\, \langle P', Q'\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k$.*

Proof. Let ρ such that it closes both specifications.
From the first premiss, when P is satisfied, then $(\!|P|\!)^\rho \subseteq (\!|P'|\!)^\rho$.
From the second premiss, when Q' is satisfied, then $(\!|Q'|\!)^\rho \subseteq (\!|Q'|\!)^\rho$.
Then, from definition 26, it follows that for all $h \in \textsc{Heap}$, $\mathsf{a}((\!|P'|\!)^\rho, (\!|Q'|\!)^\rho)_k(h) \subseteq \mathsf{a}((\!|P|\!)^\rho, (\!|Q|\!)^\rho)_k(h)$.
Therefore, $\mathcal{R}[\![\forall \vec{x}.\, \langle P', Q'\rangle_k]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho$.
Then, by lemma 12, $\left(\mathcal{R}[\![\forall \vec{x}.\, \langle P', Q'\rangle_k]\!]^\rho\right)^\dagger \subseteq (\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho)^\dagger$.
Thus, by lemma 16, and definition 36, $\forall \vec{x}.\, \langle P', Q'\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k$. □

Lemma 95 (AChoice)**.** $\forall \vec{x}.\, \langle P, Q \vee Q'\rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q\rangle_k \sqcup \forall \vec{x}.\, \langle P, Q'\rangle_k$

Proof. Let ρ such that it closes both specifications.
By definition 22 and definition 26,
$\mathsf{a}((\!|P|\!)^\rho, (\!|Q \vee Q'|\!)^\rho)_k = \mathsf{a}((\!|P|\!)^\rho, (\!|Q|\!)^\rho \cup (\!|Q'|\!)^\rho)_k = \mathsf{a}((\!|P|\!)^\rho, (\!|Q|\!)^\rho)_k \cup \mathsf{a}((\!|P|\!)^\rho, (\!|Q'|\!)^\rho)_k$.
By definition 44,
$\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \vee Q'\rangle_k]\!]^\rho =$

$$\begin{aligned}&\left\{(h,h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q \vee Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h)\right\} \\ \cup &\left\{(h,\lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l}\vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q \vee Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h) = \emptyset \\ \wedge\ (\!|Q \vee Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}_{\neq} \emptyset\end{array}\right\} \\ \cup &\left\{(\lightning,\lightning)\right\} \\ = \text{ by definition 22}& \\ &\left\{(h,h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \cup (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h)\right\} \\ \cup &\left\{(h,\lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l}\vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \cup (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h) = \emptyset \\ \wedge\ (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \cup (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \neq \emptyset\end{array}\right\} \\ \cup &\left\{(\lightning,\lightning)\right\} \\ = &\left\{(h,h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h)\right\} \\ \cup &\left\{(h,h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h)\right\} \\ \cup &\left\{(h,\lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l}\vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \cup (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h) = \emptyset \\ \wedge\ (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \cup (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \neq \emptyset\end{array}\right\} \\ \cup &\left\{(\lightning,\lightning)\right\} \cup \left\{(\lightning,\lightning)\right\} \\ \subseteq &\left\{(h,h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h)\right\} \\ \cup &\left\{(h,h') \in \textsc{Move} \;\middle|\; \vec{v} \in \overrightarrow{\textsc{Val}} \wedge h' \in \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h)\right\} \\ \cup &\left\{(h,\lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l}\vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h) = \emptyset \\ \wedge\ (\!|Q|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \neq \emptyset\end{array}\right\} \\ \cup &\left\{(h,\lightning) \in \textsc{Heap}^{\lightning} \;\middle|\; \begin{array}{l}\vec{v} \in \overrightarrow{\textsc{Val}} \wedge \mathsf{a}\left((\!|P|\!)^{\rho[\vec{x}\mapsto\vec{v}]}, (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]}\right)_k(h) = \emptyset \\ \wedge\ (\!|Q'|\!)^{\rho[\vec{x}\mapsto\vec{v}]} \neq \emptyset\end{array}\right\} \\ \cup &\left\{(\lightning,\lightning)\right\} \cup \left\{(\lightning,\lightning)\right\} \\ = \text{ by definition 44}& \\ &\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q\rangle_k]\!]^\rho \sqcup \mathcal{R}[\![\forall \vec{x}.\, \langle P, Q'\rangle_k]\!]^\rho\end{aligned}$$

Then, by lemma 12,
$\left(\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \vee Q' \rangle_k]\!]^\rho\right)^\dagger \subseteq \left(\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_k \sqcup \forall \vec{x}.\, \langle P, Q' \rangle_k]\!]^\rho\right)^\dagger$.
By lemma 16, $[\![\forall \vec{x}.\, \langle P, Q \vee Q' \rangle_k]\!]^\rho \subseteq [\![\forall \vec{x}.\, \langle P, Q \rangle_k \sqcup \forall \vec{x}.\, \langle P, Q' \rangle_k]\!]^\rho$.
Thus, by definition 36, $\forall \vec{x}.\, \langle P, Q \vee Q' \rangle_k \sqsubseteq \forall \vec{x}.\, \langle P, Q \rangle_k \sqcup \forall \vec{x}.\, \langle P, Q' \rangle_k$ □

Lemma 96 (ARLevel)**.** *If $k_1 \leq k_2$, then $\forall \vec{x}.\, \langle P, Q \rangle_{k_1} \sqsubseteq \forall \vec{x}.\, \langle P, Q \rangle_{k_2}$*

Proof. Let ρ that closes $\forall \vec{x}.\, \langle P, Q \rangle_-$.
Fix $h \in$ Heap. By definition 26,

$$\begin{aligned}
\mathsf{a}(\langle\!|P|\!\rangle^\rho, \langle\!|Q|\!\rangle^\rho)_{k_1}(h) &= \left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in \langle\!|P|\!\rangle^\rho * r.\, h \in \mathbb{L}w\mathbb{J}_{k_1;} \\ \wedge\, \exists w'.\, w\ G_{k_1;}\ w' \wedge h' \in \mathbb{L}w'\mathbb{J}_{k_1;\wedge} w' \in \langle\!|Q|\!\rangle^\rho * r \end{array} \right\} \\
&= \text{by } \mathbb{L}w\mathbb{J}_{k_1;} = \mathbb{L}w\mathbb{J}_{k_2;}, \mathbb{L}w'\mathbb{J}_{k_1;} = \mathbb{L}w'\mathbb{J}_{k_2;} \text{ as all regions are opened} \\
&\quad \left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in \langle\!|P|\!\rangle^\rho * r.\, h \in \mathbb{L}w\mathbb{J}_{k_2;} \\ \wedge\, \exists w'.\, w\ G_{k_1;}\ w' \wedge h' \in \mathbb{L}w'\mathbb{J}_{k_2;\wedge} w' \in \langle\!|Q|\!\rangle^\rho * r \end{array} \right\} \\
&\subseteq \text{by } \ G_{k_1;} \subseteq G_{k_2;} \\
&\quad \left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in \langle\!|P|\!\rangle^\rho * r.\, h \in \mathbb{L}w\mathbb{J}_{k_2;} \\ \wedge\, \exists w'.\, w\ G_{k_2;}\ w' \wedge h' \in \mathbb{L}w'\mathbb{J}_{k_2;\wedge} w' \in \langle\!|Q|\!\rangle^\rho * r \end{array} \right\} \\
&= \mathsf{a}(\langle\!|P|\!\rangle^\rho, \langle\!|Q|\!\rangle^\rho)_{k_2}(h)
\end{aligned}$$

From this and definition 44, $\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_{k_1}]\!]^\rho \subseteq \mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_{k_2}]\!]^\rho$.
By lemma 12, $\left(\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_{k_1}]\!]^\rho\right)^\dagger \subseteq \left(\mathcal{R}[\![\forall \vec{x}.\, \langle P, Q \rangle_{k_2}]\!]^\rho\right)^\dagger$.
Then, by lemma 16, $[\![\forall \vec{x}.\, \langle P, Q \rangle_{k_1}]\!]^\rho \subseteq [\![\forall \vec{x}.\, \langle P, Q \rangle_{k_2}]\!]^\rho$.
Thus, by definition 36, $\forall \vec{x}.\, \langle P, Q \rangle_{k_1} \sqsubseteq \forall \vec{x}.\, \langle P, Q \rangle_{k_2}$. □

Lemma 97 (AUseAtomic)**.** *If $\forall x \in X.\, (x, f(x)) \in \mathcal{T}_{\mathbf{t}}(G)^*$, then*

$$\forall x \in X.\, \left\langle I(\mathbf{t}^k_\alpha(x)) * P * [\mathsf{G}]_\alpha, I(\mathbf{t}^k_\alpha(f(x))) * Q \right\rangle_k \equiv \forall x \in X.\, \left\langle \mathbf{t}^k_\alpha(x) * P * [\mathsf{G}]_\alpha, \mathbf{t}^k_\alpha(f(x)) * Q \right\rangle_{k+1}$$

Proof. Assume the premiss.
Fix ρ such that it closes both specifications. Fix $v \in X$.
Let $P' = I(\mathbf{t}^k_\alpha(x)) * P * [\mathsf{G}]_\alpha$ and $Q' = I(\mathbf{t}^k_\alpha(f(x))) * Q$.

$$\mathsf{a}\left(\langle\!|P'|\!\rangle^{\rho[x\mapsto v]}, \langle\!|Q'|\!\rangle^{\rho[x\mapsto v]}\right)_k(h) = \left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in \langle\!|P'|\!\rangle^{\rho[x\mapsto v]} * r.\, h \in \mathbb{L}w\mathbb{J}_{k;} \\ \wedge\, \exists w'.\, w\ G_{k;}\ w' \wedge h' \in \mathbb{L}w'\mathbb{J}_{k;\wedge} w' \in \langle\!|Q'|\!\rangle^{\rho[x\mapsto v]} * r \end{array} \right\}$$

Let $\overline{w} \in \langle\!|\mathbf{t}^k_\alpha(x) * P * [\mathsf{G}]_\alpha|\!\rangle^\rho$. Then, $\mathbb{L}w\mathbb{J}_k = \mathbb{L}\overline{w}\mathbb{J}_{k+1}$.
Let $\overline{w'} \in \langle\!|\mathbf{t}^k_\alpha(x) * Q|\!\rangle^\rho$. From the guarantee, $\overline{w}\ G_{k+1;}\ \overline{w'}$, and we have that $\mathbb{L}w'\mathbb{J}_k = \mathbb{L}\overline{w'}\mathbb{J}_{k+1}$.
Therefore,

$$\begin{aligned}
&\left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall w \in \langle\!|P'|\!\rangle^{\rho[x\mapsto v]} * r.\, h \in \mathbb{L}w\mathbb{J}_{k;} \\ \wedge\, \exists w'.\, w\ G_{k;}\ w' \wedge h' \in \mathbb{L}w'\mathbb{J}_{k;} \wedge w' \in \langle\!|Q'|\!\rangle^{\rho[x\mapsto v]} * r \end{array} \right\} \\
=&\left\{ h' \in \textsc{Heap} \;\middle|\; \begin{array}{l} \forall r \in \textsc{View}.\, \forall \overline{w} \in \langle\!|\overline{P'}|\!\rangle^{\rho[x\mapsto v]} * r.\, h \in \mathbb{L}\overline{w}\mathbb{J}_{k+1;} \\ \wedge\, \exists \overline{w}'.\, \overline{w}\ G_{k+1;}\ \overline{w'} \wedge h' \in \mathbb{L}\overline{w'}\mathbb{J}_{k+1;} \wedge \overline{w'} \in \langle\!|\overline{Q'}|\!\rangle^{\rho[x\mapsto v]} * r \end{array} \right\} \\
=&\mathsf{a}\left(\langle\!|\overline{P'}|\!\rangle^{\rho[x\mapsto v]}, \langle\!|\overline{Q'}|\!\rangle^{\rho[x\mapsto v]}\right)_{k+1}(h)
\end{aligned}$$

where $\overline{P'} = \mathbf{t}^k_\alpha(x) * P * [\mathsf{G}]_\alpha$ and $\overline{Q'} = \mathbf{t}^k_\alpha(f(x)) * Q$.
Thus, from definition 35,

$$[\![\forall x \in X.\, \left\langle I(\mathbf{t}^k_\alpha(x)) * P * [\mathsf{G}]_\alpha, I(\mathbf{t}^k_\alpha(f(x))) * Q \right\rangle_k]\!]^\rho = \left[\!\!\left[\forall x \in X.\, \left\langle \mathbf{t}^k_\alpha(x) * P * [\mathsf{G}]_\alpha, \mathbf{t}^k_\alpha(f(x)) * Q \right\rangle_{k+1}\right]\!\!\right]^\rho$$

Then, from definition 36,

$$\forall x \in X.\, \left\langle I(\mathbf{t}^k_\alpha(x)) * P * [\mathsf{G}]_\alpha, I(\mathbf{t}^k_\alpha(f(x))) * Q \right\rangle_k \equiv \forall x \in X.\, \left\langle \mathbf{t}^k_\alpha(x) * P * [\mathsf{G}]_\alpha, \mathbf{t}^k_\alpha(f(x)) * Q \right\rangle_{k+1}$$

□

D Proofs of Hybrid Specification Statement Refinement Laws

Lemma 98.

$$\begin{array}{c}\forall \vec{x}.\, \langle P' * P(\vec{x}), P'' * P(\vec{x})\rangle_k; \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\ \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k\end{array}$$

Proof.

$$\begin{array}{l}\forall \vec{x}.\, \langle P' * P(\vec{x}), P'' * P(\vec{x})\rangle_k; \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\ \sqsubseteq \text{ by lemma 6, ACons, AEarly and FApplyElimRec} \\ \quad \exists p'.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P'' \wedge p' * P(\vec{x})\rangle_k; \\ \quad \mu A.\, \lambda p.\ \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\ \qquad \sqcup \exists \vec{x}, \vec{y}.\, \exists p''.\ \langle p * P(\vec{x}), p'' * Q(\vec{x},\vec{y})\rangle_k; \\ \qquad\qquad \mu B.\, \lambda p''.\ \exists p'''.\, \langle p'', p'''\rangle_k; Bp''' \\ \qquad\qquad\qquad \sqcup \langle p'', Q'(\vec{x},\vec{y})\rangle_k \\ \qquad\qquad \cdot p'' \\ \quad \cdot p' \\ \sqsubseteq \text{ by AStutter, ACons, AEarly and CMono} \\ \quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\ \qquad \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), P'' \wedge p' * P(\vec{x})\rangle_k; \\ \qquad \mu A.\, \lambda p.\ \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\ \qquad\quad \sqcup \exists \vec{x}, \vec{y}.\, \exists p''.\ \langle p * P(\vec{x}), p'' * Q(\vec{x},\vec{y})\rangle_k; \\ \qquad\qquad \mu B.\, \lambda p''.\ \exists p'''.\, \langle p'', p'''\rangle_k; Bp''' \\ \qquad\qquad\qquad \sqcup \langle p'', Q'(\vec{x},\vec{y})\rangle_k \\ \qquad\qquad \cdot p'' \\ \qquad \cdot p' \\ \sqsubseteq \text{ by AChoiceElim, AChoiceComm, UnrollR and CMono} \\ \quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\ \qquad \mu A.\, \lambda p.\ \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\ \qquad\quad \sqcup \exists \vec{x}, \vec{y}.\, \exists p''.\ \langle p * P(\vec{x}), p'' * Q(\vec{x},\vec{y})\rangle_k; \\ \qquad\qquad \mu B.\, \lambda p''.\ \exists p'''.\, \langle p'', p'''\rangle_k; Bp''' \\ \qquad\qquad\qquad \sqcup \langle p'', Q'(\vec{x},\vec{y})\rangle_k \\ \qquad\qquad \cdot p'' \\ \qquad \cdot p \\ \equiv \text{ by definition 40} \\ \quad \forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k\end{array}$$

□

Lemma 99 (HUseAtomic)**.**

$$\begin{array}{c}\forall x, \vec{x}.\, \exists \vec{y}.\, \{P'\}\left\langle I(\mathbf{t}^k_\alpha(\vec{e}, x)) * P(x,\vec{x}) * [\mathsf{G}(\vec{e}')]_\alpha, I(\mathbf{t}^k_\alpha(\vec{e}, f(x))) * Q(x,\vec{x},\vec{y})\right\rangle\{Q'(\vec{x},\vec{y})\}_k \\ \sqsubseteq \forall x, \vec{x}.\, \exists \vec{y}.\, \{P'\}\left\langle \mathbf{t}^k_\alpha(\vec{e}, x) * P(x,\vec{x}) * [\mathsf{G}(\vec{e}')]_\alpha, \mathbf{t}^k_\alpha(\vec{e}, f(x)) * Q(x,\vec{x},\vec{y})\right\rangle\{Q'(\vec{x},\vec{y})\}_{k+1}\end{array}$$

Proof. By definition 40, AFrame, AUseAtomic, RLevel and CMono. □

Lemma 100 (HFrame)**.**

$$\begin{array}{c}\forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\ \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \{P' * R'\}\langle P(\vec{x}) * R(\vec{x}), Q(\vec{x},\vec{y}) * R(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y}) * R'\}_k\end{array}$$

Proof. By definition 40, AFrame and CMono. □

Lemma 101 (HStrengthen)**.**

$$\begin{array}{c}\forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P' * P(\vec{x}), Q'(\vec{x},\vec{y}) * Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\ \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \{P' * P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y}) * Q'(\vec{x},\vec{y})\}_k\end{array}$$

Proof. By definition 40, AFrame and CMono. □

Lemma 102 (Atomic)**.**

$$\begin{array}{c}\forall \vec{x}.\, \langle P' * P(\vec{x}), \exists \vec{y}.\, Q'(\vec{x},\vec{y}) * Q(\vec{x},\vec{y})\rangle_k \\ \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k\end{array}$$

Proof. Trivial, using AMumble and Ind. □

Lemma 103 (HEElim)**.**

$$\begin{array}{c}\forall x, \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(x,\vec{x}), Q(x,\vec{x},\vec{y})\rangle\{P'(x,\vec{x},\vec{y})\}_k \\ \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle \exists x.\, P(x,\vec{x}), \exists x.\, Q(x,\vec{x},\vec{y})\rangle\{\exists x.\, P'(x,\vec{x},\vec{y})\}_k\end{array}$$

Proof. By AEElim and CMono. □

Lemma 104 (HStutter)**.**

$$\begin{array}{c}\forall \vec{x}.\, \{P'\}\langle P(\vec{x}), P(\vec{x})\rangle\{P''\}_k; \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\ \sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k\end{array}$$

Proof. First,

$$\begin{array}{l}\quad \forall \vec{x}.\, \{P'\}\langle P(\vec{x}), P(\vec{x})\rangle\{P''\}_k; \\ \equiv \text{ by definition 40} \\ \quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\ \qquad \mu A.\, \lambda p.\ \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\ \qquad\qquad \sqcup\ \exists \vec{x}.\, \exists p''.\ \forall \vec{x}.\, \langle p * P(\vec{x}), p'' * P(\vec{x})\rangle_k; \\ \qquad\qquad\qquad \mu B.\, \lambda p''.\ \exists p'''.\, \langle p'', p'''\rangle_k; Bp''' \\ \qquad\qquad\qquad\qquad \sqcup\ \langle p'', P''\rangle_k \\ \qquad\qquad\qquad \cdot p'' \\ \qquad \cdot p \\ \sqsubseteq \text{ by AFrame, AUElim, EElim and CMono} \\ \quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\ \qquad \mu A.\, \lambda p.\ \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\ \qquad\qquad \sqcup\ \exists \vec{x}.\, \exists p''.\, \langle p * P(\vec{x}), p'' * P(\vec{x})\rangle_k; \\ \qquad\qquad\quad \mu B.\, \lambda p''.\ \exists \vec{x}.\, \exists p'''.\, \forall \vec{x}.\, \langle p'' * P(\vec{x}), p''' * P(\vec{x})\rangle_k; Bp''' \\ \qquad\qquad\qquad\quad \sqcup\ \forall \vec{x} \in \overrightarrow{X}.\, \langle p'' * P(\vec{x}), P'' * P(\vec{x})\rangle_k \\ \qquad\qquad\quad \cdot p'' \\ \qquad \cdot p \\ \sqsubseteq \text{ by AChoiceComm, AChoiceElim, UnrollR and CMono} \\ \quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\ \qquad \mu A.\, \lambda p.\ \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\ \qquad\qquad \sqcup\ \mu B.\, \lambda p''.\ \exists \vec{x}.\, \exists p'''.\, \forall \vec{x}.\, \langle p'' * P(\vec{x}), p''' * P(\vec{x})\rangle_k; Bp''' \\ \qquad\qquad\qquad \sqcup\ \forall \vec{x}.\, \langle p'' * P(\vec{x}), P'' * P(\vec{x})\rangle_k \\ \qquad\qquad\quad \cdot p'' \\ \qquad \cdot p \\ \sqsubseteq \text{ by Ind, for premiss: } \alpha\text{-conversion, AEElim, EElim, AChoiceElim and UnrollR} \\ \quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\ \qquad \mu A.\, \lambda p.\ \exists \vec{x}.\, \exists p'.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\ \qquad\qquad \sqcup\ \forall \vec{x}.\, \langle p * P(\vec{x}), P'' * P(\vec{x})\rangle_k \\ \qquad \cdot p\end{array}$$

Then,

$$
\begin{array}{l}
\quad \forall \vec{x}.\, \{P'\}\langle P(\vec{x}), P(\vec{x})\rangle\{P''\}_k; \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\
\sqsubseteq \text{ by CMono} \\
\quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\
\qquad \mu A.\, \lambda p.\ \exists \vec{x}.\, \exists p'.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\
\qquad\qquad \sqcup\ \forall \vec{x}.\, \langle p * P(\vec{x}), P'' * P(\vec{x})\rangle_k \\
\qquad \cdot p; \\
\qquad \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\
\equiv \text{ by AChoiceComm and RecSeq} \\
\quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\
\qquad \mu A.\, \lambda p.\ \exists \vec{x}.\, \exists p'.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\
\qquad\qquad \sqcup\ \forall \vec{x}.\, \langle p * P(\vec{x}), P'' * P(\vec{x})\rangle_k; \\
\qquad\qquad\quad \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\
\qquad \cdot p \\
\sqsubseteq \text{ by lemma 98 and CMono} \\
\quad \exists p.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\
\qquad \mu A.\, \lambda p.\ \exists \vec{x}.\, \exists p'.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\
\qquad\qquad \sqcup\ \forall \vec{x}.\, \exists \vec{y}.\, \{p\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\
\qquad \cdot p \\
\sqsubseteq \text{ by Ind, for premiss: lemma 98, CMono and AChoiceEq, AStutter and definition 40} \\
\quad \forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k
\end{array}
$$

□

Lemma 105 (HMumble)**.**

$$
\begin{array}{c}
\forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \\
\sqsubseteq \forall \vec{x}.\, \{P'\}\langle P(\vec{x}), P'(\vec{x})\rangle\{P''\}_k; \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P'(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k
\end{array}
$$

Proof. By AMumble steps in definition 40, create the recursive function for $\forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P'(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k$. Then, apply RecSeq. □

Lemma 106 (HDisjunction)**.**

$$
\begin{array}{l}
\forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \sqcup \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P'(\vec{x}), Q'(\vec{x},\vec{y})\rangle\{Q''(\vec{x},\vec{y})\}_k \\
\sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \{P' \vee P''\}\langle P(\vec{x}) \vee P'(\vec{x}), Q(\vec{x},\vec{y}) \vee Q'(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y}) \vee Q''(\vec{x},\vec{y})\}_k
\end{array}
$$

Proof. First we observe the following:

$$
\begin{array}{l}
\forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \sqsubseteq \\
\exists p, p''.\, \forall \vec{x}.\, \langle P' * P(\vec{x}), P' \wedge p * P(\vec{x})\rangle_k; \\
\mu A.\, \lambda p.\ \exists p'.\, \forall \vec{x}.\, \langle p * P(\vec{x}), p' * P(\vec{x})\rangle_k; Ap' \\
\qquad \sqcup\ \exists \vec{x}, \vec{y}.\ \langle p * P(\vec{x}), p'' * Q(\vec{x},\vec{y})\rangle_k \\
\cdot p; \\
\mu B.\, \lambda p''.\ \exists p'''.\, \langle p'', p'''\rangle_k; Bp''' \\
\qquad \sqcup\ \langle p'', Q'(\vec{x},\vec{y})\rangle_k \\
\cdot p''
\end{array}
$$

Each recursive function above is structurally similar to the recursive function of a Hoare specification statement according to lemma 7. Thus we proceed as in lemma 113. □

Lemma 107 (HConjunction)**.**

$$
\begin{array}{l}
\forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}), Q(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y})\}_k \sqcap \forall \vec{x}.\, \exists \vec{y}.\, \{P''\}\langle P'(\vec{x}), Q'(\vec{x},\vec{y})\rangle\{Q''(\vec{x},\vec{y})\}_k \\
\sqsubseteq \forall \vec{x}.\, \exists \vec{y}.\, \{P' \wedge P''\}\langle P(\vec{x}) \wedge P'(\vec{x}), Q(\vec{x},\vec{y}) \wedge Q'(\vec{x},\vec{y})\rangle\{Q'(\vec{x},\vec{y}) \wedge Q''(\vec{x},\vec{y})\}_k
\end{array}
$$

Proof. Similarly to lemma 106. □

Lemma 108 (HCons)**.** *If* $P' \Rightarrow P''$*, and* $\forall \vec{x} \in \overrightarrow{X}. P(\vec{x}) \Rightarrow P'(\vec{x})$*, and* $\forall \vec{x} \in \overrightarrow{X}, \vec{y} \in \overrightarrow{Y}. Q''(\vec{x}, \vec{y}) \Rightarrow Q'(\vec{x}, \vec{y})$*, and* $\forall \vec{x} \in \overrightarrow{X}. Q'(\vec{x}, \vec{y}) \Rightarrow Q(\vec{x}, \vec{y})$*, then*

$$\begin{array}{l} \forall \vec{x}. \exists \vec{y}. \{P''\}\langle P'(\vec{x}), Q'(\vec{x}, \vec{y})\rangle\{Q''(\vec{x}, \vec{y})\}_k \\ \sqsubseteq \forall \vec{x}. \exists \vec{y}. \{P'\}\langle P(\vec{x}), Q(\vec{x}, \vec{y})\rangle\{Q'(\vec{x}, \vec{y})\}_k \end{array}$$

Proof. By definition 40, ACons and CMono. □

Lemma 109 (HRLevel)**.** *If* $k_1 \leq k_2$*, then*

$$\begin{array}{l} \forall \vec{x}. \exists \vec{y}. \{P'\}\langle P(\vec{x}), Q(\vec{x}, \vec{y})\rangle\{Q'(\vec{x}, \vec{y})\}_{k_1} \\ \sqsubseteq \forall \vec{x}. \exists \vec{y}. \{P'\}\langle P(\vec{x}), Q(\vec{x}, \vec{y})\rangle\{Q'(\vec{x}, \vec{y})\}_{k_2} \end{array}$$

Proof. By RLevel on definition 40. □

E Proofs of Hoare-Statement Refinement Laws

Lemma 110 (Seq)**.** *If* $\phi \sqsubseteq I \vdash \{P,\ R\}_k$ *and* $\psi \sqsubseteq I \vdash \{R,\ Q\}_k$*, then* $\phi; \psi \sqsubseteq I \vdash \{P,\ Q\}_k$*.*

Proof. Follows directly from the premisses, definition 41, AStutter and CMono. □

Lemma 111.

$$\begin{array}{rl} I \vdash \{P,\ Q\}_k \equiv \exists p.\ \langle P * I, P \wedge p * I\rangle_k; & \\ \qquad \mu A.\lambda p.\ \exists p'.\ \langle p * I, p' * I\rangle_k; Ap' & \\ \qquad\qquad \sqcup\ \langle p * I, Q * I\rangle_k & \\ \qquad \cdot p & \end{array}$$

Proof. Similarly to lemma 7. □

Lemma 112 (Disjunction)**.** *If* $\phi \sqsubseteq I \vdash \{P_1,\ Q_1\}_k$ *and* $\psi \sqsubseteq I \vdash \{P_2,\ Q_2\}_k$*, then* $\phi \sqcup \psi \sqsubseteq I \vdash \{P_1 \vee P_2,\ Q_1 \vee Q_2\}_k$*.*

Proof.

$$
\begin{array}{l}
\phi \sqcup \psi \sqsubseteq \text{ by CMono} \\
\quad I \vdash \{P_1,\, Q_1\}_k \sqcup I \vdash \{P_2,\, Q_2\}_k \\
\sqsubseteq \text{ by lemma 111, lemma 6, FApplyElimRec, AChoiceComm and CMono} \\
\quad \begin{pmatrix} \langle P_1 * I, Q_1 * I\rangle_k \\ \sqcup\, \exists p'.\, \langle P_1 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_1\}_k \end{pmatrix} \sqcup \begin{pmatrix} \langle P_2 * I, Q_2 * I\rangle_k \\ \sqcup\, \exists p'.\, \langle P_2 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_2\}_k \end{pmatrix} \\
\sqsubseteq \text{ by AChoiceAssoc, EAChoiceDst and CMono} \\
\quad \langle P_1 * I, Q_1 * I\rangle_k \sqcup \langle P_2 * I, Q_2 * I\rangle_k \\
\quad \sqcup\, \exists p'.\, (\langle P_1 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_1\}_k) \sqcup (\langle P_2 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_2\}_k) \\
\sqsubseteq \text{ by PDisjunction, Cons and CMono} \\
\quad \langle P_1 \vee P_2 * I, Q_1 \vee Q_2 * I\rangle_k \\
\quad \sqcup\, \exists p'.\, (\langle P_1 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_1\}_k) \sqcup (\langle P_2 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_2\}_k) \\
\sqsubseteq \text{ by HCons and CMono} \\
\quad \langle P_1 \vee P_2 * I, Q_1 \vee Q_2 * I\rangle_k \\
\quad \sqcup\, \exists p'.\, (\langle P_1 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_1 \vee Q_2\}_k) \sqcup (\langle P_2 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_1 \vee Q_2\}_k) \\
\sqsubseteq \text{ by AChoiceDstR and CMono} \\
\quad \langle P_1 \vee P_2 * I, Q_1 \vee Q_2 * I\rangle_k \\
\quad \sqcup\, \exists p'.\, (\langle P_1 * I, p' * I\rangle_k \sqcup \langle P_2 * I, p' * I\rangle_k) ; I \vdash \{p',\, Q_1 \vee Q_2\}_k \\
\sqsubseteq \text{ by PDisjunction, Cons and CMono} \\
\quad \langle P_1 \vee P_2 * I, Q_1 \vee Q_2 * I\rangle_k \\
\quad \sqcup\, \exists p'.\, \langle P_1 \vee P_2 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_1 \vee Q_2\}_k \\
\sqsubseteq \text{ by AChoiceComm, FApplyElimRec, lemma 6 and lemma 111} \\
\quad I \vdash \{P_1 \vee P_2,\, Q_1 \vee Q_2\}_k
\end{array}
$$

□

Lemma 113 (Conjunction)**.** *If* $\phi \sqsubseteq I \vdash \{P_1,\, Q_1\}_k$ *and* $\psi \sqsubseteq I \vdash \{P_2,\, Q_2\}_k$*, then* $\phi \sqcap \psi \sqsubseteq I \vdash \{P_1 \wedge P_2,\, Q_1 \wedge Q_2\}_k$*.*

Proof. Similarly to lemma 114.

□

Lemma 114 (Parallel)**.** *If* $\phi \sqsubseteq I \vdash \{P_1,\, Q_1\}_k$ *and* $\psi \sqsubseteq I \vdash \{P_2,\, Q_2\}_k$*, then* $\phi \parallel \psi \sqsubseteq I \vdash \{P_1 * P_2,\, Q_1 * Q_2\}_k$*.*

Proof. First we show the following:

$$
\begin{array}{l}
\quad I \vdash \{P_1 * P_2,\, Q_1 * Q_2\}_k \\
\equiv \text{ by lemma 111 and lemma 6} \\
\quad \langle P_1 * P_2 * I, Q_1 * Q_2 * I\rangle_k \\
\quad \sqcup\, \exists p'.\, \langle P_1 * P_2 * I, p' * I\rangle_k ; I \vdash \{p',\, Q_1 * Q_2\}_k \\
\sqsupseteq \text{ by AChoiceElim} \\
\quad \langle P_1 * P_2 * I, Q_1 * Q_2 * I\rangle_k \\
\equiv \text{ by Stutter and Ind} \\
\quad \exists p.\ \langle P_1 * P_2 * I, (P_1 * P_2) \wedge p * I\rangle_k ; \\
\qquad (\mu A.\, \lambda p.\, \langle p * I, Q_1 * Q_2 * I\rangle_k \sqcup A p)\, p
\end{array}
$$

Next, for the recursive function derived above:

$$
\begin{aligned}
&\langle P_1 * P_2 * I, Q_1 * Q_2 * I\rangle_k \sqcup (I \vdash \{P_1,\, Q_1\}_k \parallel I \vdash \{P_1,\, Q_2\}_k) \\
&\sqsupseteq \text{ by ACHOICEELIM} \\
&\quad I \vdash \{P_1,\, Q_1\}_k \parallel I \vdash \{P_1,\, Q_2\}_k
\end{aligned}
$$

Thus, by IND and TRANS:

$$I \vdash \{P_1,\, Q_1\}_k \parallel I \vdash \{P_1,\, Q_2\}_k \sqsubseteq I \vdash \{P_1 * P_2,\, Q_1 * Q_2\}_k$$

The result follows by the premisses, CMONO and TRANS.

□

Lemma 115 (FRAME)**.** $I \vdash \{P,\, Q\}_k \sqsubseteq I \vdash \{P * R,\, Q * R\}_k$

Proof. By definition 41, AFRAME and CMONO. □

Lemma 116 (EELIM)**.**

$$\frac{\text{EELIMHOARE}}{I \vdash \{P,\, Q\}_k \sqsubseteq I \vdash \{\exists y.\, P,\, \exists y.\, Q\}_k}$$

Proof. By definitions 41 and 40, EPELIM and CMONO. □

Lemma 117 (EARLY)**.** *If* $x \notin \texttt{free}(P) \cup \texttt{free}(I)$*, then*

$$\exists x.\, I \vdash \{P,\, Q\}_k \equiv I \vdash \{P,\, \exists x.\, Q\}_k$$

Proof. By definition 41 and EAATOM. □

Lemma 118 (HYBRID Proof)**.**

$$
\begin{aligned}
&\forall \vec{x}.\, \exists \vec{y}.\, \{P'\}\langle P(\vec{x}) * I(\vec{x}), Q(\vec{x},\vec{y}) * I(\vec{x})\rangle\{Q'(\vec{x},\vec{y})\}_k \\
&\sqsubseteq \exists \vec{x} \in \overrightarrow{X}.\, I(\vec{x}) \vdash \left\{P' * P(\vec{x}),\, \exists \vec{y} \in \overrightarrow{Y}.\, Q'(\vec{x},\vec{y}) * Q(\vec{x},\vec{y})\right\}_k
\end{aligned}
$$

Proof. Direct, by definition 41 and HSTRENGTHEN.

□

Lemma 119 (CONS)**.** *If* $P \Rightarrow P'$ *and* $Q' \Rightarrow Q$*, then* $I \vdash \{P',\, Q'\}_k \sqsubseteq I \vdash \{P,\, Q\}_k$*.*

Proof. By definition 41 and ACONS. □

Lemma 120 (RLEVEL)**.** *If* $k_1 \leq k_2$*, then* $\{P,\, Q\}_{k_1} \sqsubseteq \{P,\, Q\}_{k_2}$*.*

Proof. By definition 41 and ARLEVEL. □

F Extra Proof Sketches

Figure 1 contains a full proof sketch of the `lock`() operation, including the full detail of derivation of a specification for the call to the `open`() POSIX operation.

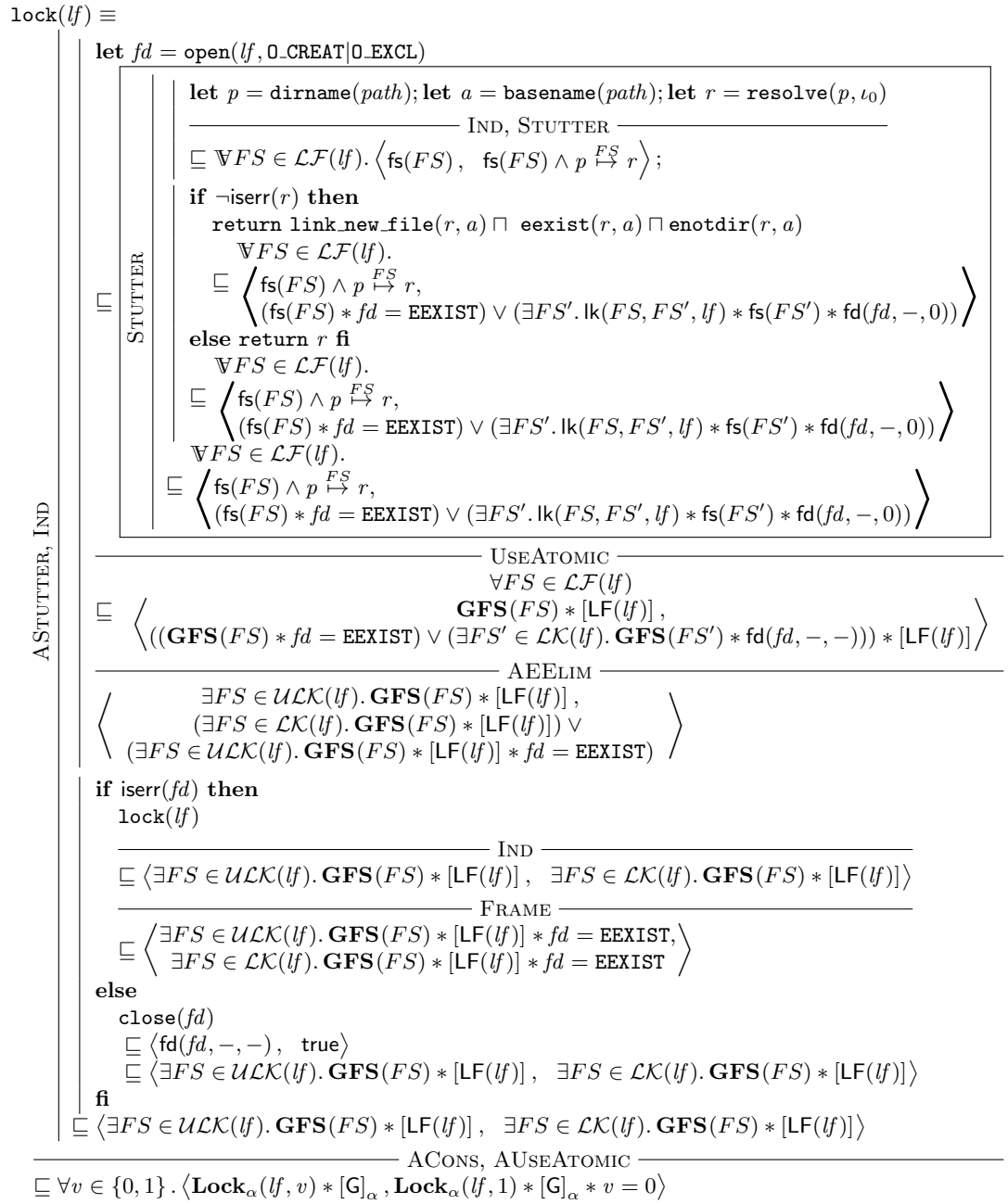

Figure 1: Complete `lock()` specification proof sketch.