CoLoSL: Concurrent Local Subjective Logic

Azalea Raad, Jules Villard, Philippa Gardner

August 31, 2014

1. CoLoSL Model and Assertions

We formally describe the underlying model of CoLoSL; we present the various ingredients necessary for defining the CoLoSL *worlds*: the building blocks of CoLoSL that track the resources held by each thread, the shared resources accessible to all threads, as well as the ways in which the shared resources may be manipulated by each thread.

We then proceed with the *assertions* of CoLoSL and provide their semantics by relating them to sets of worlds. Finally, we establish the validity of Copy, Forget and Merge principles introduced in the preceding chapters by establishing their truth for all possible worlds and interpretations.

1.1 Worlds

Overview A *world* is a 4-tuple $(l, g, \mathcal{I}, \mathcal{G})$ that is *well-formed* where l and g are *logical states* and $\mathcal{I}$ and $\mathcal{G}$ are *action models*; let us explain the role of each component informally. The *local logical state*, or simply local state, l represents the locally owned resources of a thread. The *shared logical state*, or shared state, g represents the *entire* (global) shared state, accessible to all threads, subject to interferences as described by the action models.

An action model is a partial function from *capabilities* to sets of *actions*. An action is a pair (p, q) of logical states where p is the *pre-state* of the action and q its *post-state*. The *local action model* $\mathcal{I}$ corresponds directly to the (semantic interpretation of) an interference assertion I. The *global action model* $\mathcal{G}$ extrapolates the effect of the relevant actions (those that can take place) in $\mathcal{I}$ to the global shared state g. Although worlds do not put further constraints on the relationship between $\mathcal{I}$ and $\mathcal{G}$, they are linked more tightly in the semantics of assertions (§1.2): as we shall see, $\mathcal{G}$ records at least those actions that matter for the current subjective view.

Finally, the composition of two worlds will be defined whenever their local states are disjoint and they agree on all other three components, hence

have identical knowledge of the shared state and possible interferences.

We proceed by defining logical states, which are CoLoSL's notion of *resource*, in the standard separation logic sense. Logical states have two components: one describes machine states (*e.g.* stacks and heaps); the other represents *capabilities*. The latter are inspired by the capabilities in deny-guarantee reasoning [4]: a thread in possession of a given capability is allowed to perform the associated actions (as prescribed by the *action model* components of each world, defined below), while any capability *not* owned by a thread means that the environment can perform the action.

CoLoSL is parametric in the choice of the separation algebra representing the machine states and capabilities. This allows for suitable *instantiation* of CoLoSL depending on the programs being verified. For instance, in Example **??** the separation algebra of machine states is a standard variable stack; while capabilities are captured as a set of tokens. However, as we demonstrate in the examples of §**??**, our programs often call for a more complex model of machine states and capabilities. For instance, we may need our capabilities to be fractionally owned, where ownership of a *fraction* of a capability grants the right to perform the action to both the thread and the environment, while a fully-owned capability by the thread *denies* the right to the environment to perform the associated action.

In general, the separation algebra of machine states and capabilities can be instantiated with *any* separation algebra (*i.e.* a cancellative, partial commutative monoid [1]) that satisfies the *cross-split* property. This is formalised in the following parametrisations.

Parameter 1 (Machine states separation algebra)**.** Let $(\mathbb{M}, \bullet_{\mathbb{M}}, \mathbf{0}_{\mathbb{M}})$ be any separation algebra with the cross-split property, representing machine states where the elements of $\mathbb{M}$ are ranged over by $m, m_1, \cdots, m_n$.

Parameter 2 (Capability Separation Algebra)**.** Let $(\mathbb{K}, \bullet_{\mathbb{K}}, \mathbf{0}_{\mathbb{K}})$ be any separation algebra with the cross-split property, representing capability resources where the elements of $\mathbb{K}$ are ranged over by $\kappa, \kappa_1, \cdots \kappa_n$.

We can now formalise the notion of *logical states*. As discussed above, a logical state is a pair comprising a machine state and a capability resource.

Definition 1 (Logical states)**.** Given the separation algebra of machine states, $(\mathbb{M}, \bullet_{\mathbb{M}}, \mathbf{0}_{\mathbb{M}})$, and the separation algebra of capabilities, $(\mathbb{K}, \bullet_{\mathbb{K}}, \mathbf{0}_{\mathbb{K}})$, a *logical state* is a pair (m, κ), consisting of a machine state $m \in \mathbb{M}$ and a capability $\kappa \in \mathbb{K}$.

$$\mathsf{LState} \stackrel{\text{def}}{=} \mathbb{M} \times \mathbb{K}$$

We write $l, l_1, \cdots, l_n$ to range over either arbitrary logical states or those representing the local logical state. Similarly, we write $g, g_1, \cdots, g_n$ to range over logical states when representing the shared (global) state. We write $\mathbf{0}_{\mathsf{L}}$ for the logical state $(\mathbf{0}_{\mathbb{M}}, \mathbf{0}_{\mathbb{K}})$. Given a logical state l, we write l_{M} and l_{K} for the first and second projections, respectively. The *composition of logical states* $\circ : \mathsf{LState} \times \mathsf{LState} \rightharpoonup \mathsf{LState}$ is defined component-wise:

$$(m, \kappa) \circ (m', \kappa') \stackrel{\text{def}}{=} (m \bullet_{\mathbb{M}} m', \kappa \bullet_{\mathbb{K}} \kappa')$$

The *separation algebra of logical states* is given by $(\mathsf{LState}, \circ, \mathbf{0}_{\mathsf{L}})$.

Oftentimes, we need to compare two logical states $l_1 \leq l_2$ (or their constituents: $\kappa_1 \leq \kappa_2$, $m_1 \leq m_2$) defined when there exists l such that $l \circ l_1 = l_2$. This is captured in the following definition.

Definition 2 (Ordering)**.** Given any separation algebra $(\mathbb{B}, \bullet_{\mathbb{B}}, \mathbf{0}_{\mathbb{B}})$, the *ordering relation*, $\leq : \mathbb{B} \times \mathbb{B}$, is defined as:

$$\leq \stackrel{\text{def}}{=} \{(b_1, b_2) \mid \exists b.\, b_1 \bullet_{\mathbb{B}} b = b_2\}$$

We write $b_1 \leq b_2$ for $(b_1, b_2) \in \leq$. We also write $b_2 - b_1$ to denote the unique (by cancellativity of separation algebras) element in $\mathbb{B}$ such that $b_1 \bullet_{\mathbb{B}} (b_2 - b_1) = b_2$.

In our formalisms we occasionally need to quantify over *compatible* logical states (similarly, compatible machine states or compatible capabilities), *i.e.* those that can be composed together by $\circ$. Additionally, we often describe two logical states as *disjoint*. We formalise these notions below.

Definition 3 (Compatibility)**.** Given any separation algebra $(\mathbb{B}, \bullet_{\mathbb{B}}, \mathbf{0}_{\mathbb{B}})$, the *compatibility relation*, $\sharp : \mathbb{B} \times \mathbb{B}$, is defined as:

$$\sharp \stackrel{\text{def}}{=} \{(b_1, b_2) \mid \exists b.\, b_1 \bullet_{\mathbb{B}} b_2 = b\}$$

We write $b_1 \sharp b_2$ for $(b_1, b_2) \in \sharp$.

Definition 4 (Disjointness)**.** Given any separation algebra $(\mathbb{B}, \bullet_{\mathbb{B}}, \mathbf{0}_{\mathbb{B}})$, the *disjointness relation*, $\perp : \mathbb{B} \times \mathbb{B}$, is defined as:

$$\perp \stackrel{\text{def}}{=} \{(b_1, b_2) \mid b_1 \sharp b_2 \wedge \forall b \in \mathbb{B}.\, b \leq b_1 \wedge b \leq b_2 \implies b = \mathbf{0}_{\mathbb{B}}\}$$

We write $b_1 \perp b_2$ for $(b_1, b_2) \in \perp$.

Observe that for a separation algebra $(\mathbb{B}, \bullet_{\mathbb{B}}, \mathbf{0}_{\mathbb{B}})$ satisfying the disjointness property[1], the definitions of compatibility and disjointness relations coincide.

We now proceed with the next ingredients of a CoLoSL world, namely, action models. Recall from above that an action is simply a pair of logical states describing the pre- and post-states of the action, while an action model describes the set of actions associated with each capability.

Definition 5 (Actions, action models)**.** The set of *actions*, Action, ranged over by $a, a_1, \cdots, a_n$, is as defined below.

$$\mathsf{Action} \stackrel{\text{def}}{=} \mathsf{LState} \times \mathsf{LState}$$

The set of *action models*, AMod, is defined as follows.

$$\mathsf{AMod} \stackrel{\text{def}}{=} \mathbb{K} \rightharpoonup \mathcal{P}(\mathsf{Action})$$

We write $\mathcal{I}, \mathcal{I}_1, \cdots, \mathcal{I}_n$ to range over action models representing the local action models while writing $\mathcal{G}, \mathcal{G}_1, \cdots, \mathcal{G}_n$ to range over those denoting global action models. We write $\emptyset$ for an action model with empty domain.

The Effect of Actions Given a world $(l, g, \mathcal{I}, \mathcal{G})$, since g represents the *entire* shared state, as part of the well-formedness condition of worlds we require that the actions in $\mathcal{I}$ and $\mathcal{G}$ are *confined* to g. Let us elaborate on the necessity of the confinement condition.

As threads may unilaterally decide to introduce part of their local states into the shared state at any point (by Extend), confinement ensures that existing actions cannot affect future extensions of the shared state. Similarly, we require that the new actions associated with newly shared state are confined to that extension in the same vein, hence extending the shared state cannot retroactively invalidate the views of other threads. However, as we will demonstrate, confinement does not prohibit *referring* to existing parts of the shared state in the new actions; rather, it only safeguards against *mutation* of the already shared resources through new actions.

Through confinement we ensure that the *effect* of actions in the local and global action models are contained to the shared state. In other words, given an action $a = (p, q)$ and a shared state g, whenever the precondition p *agrees* with g then the part of the state mutated by the action is contained in g. Agreement of p and g merely means that p and g agree on the resources they

[1] $\forall b, b' \in \mathbb{B}.\, b \bullet_{\mathbb{B}} b = b' \implies b = b' = \mathbf{0}_{\mathbb{B}}$

have in common. In particular, g need not contain p entirely for a to take effect. This relaxation is due to the fact that other threads may extend the shared state; in particular, the extension may provide the missing resources for p to be contained in the shared state, thus allowing the extending thread to perform action a. Crucially, however, it is not the case that any part of p is allowed to be missing from g. Rather, only parts of p that are not mutated by a (and thus can also be found in q), can escape g's grasp. Mutated parts must always be contained in g, so that extensions need not worry about existing actions interfering with new resources that were never shared beforehand. In order to distinguish between parts that must be contained in g and those that may be missing, we characterise the *active* and *passive* parts of actions, and then continue to define the effect of actions. This will enable us to define our confinement condition.

An action $a = (p, q)$ is typically of the form $(p' \circ c, q' \circ c)$, where part of the state c required for the action is *passive* and acts as a mere *catalyst* for the action: it has to be present for the action to take effect, but is left unchanged by the action. The *active* part of the action is then the pair (p', q'), which should be maximal in the sense that no further, non-empty catalyst can be found in (p', q').

Definition 6 (Active part of an action)**.** Given an action $a = (p, q)$, its *active part*, $\Delta(p, q)$, is defined as the pair (p', q') such that:

$$\exists c.\, p = p' \circ c \wedge q = q' \circ c \wedge \forall c'.\, c' \leq p' \wedge c' \leq q' \Rightarrow c' = \mathbf{0}_{\mathsf{L}}$$

The agreement of an action pre-state p and the shared state g will be defined using the following notion of *intersection* of logical states.

Definition 7 (Intersection)**.** The *intersection* function over logical states, $\sqcap : (\mathsf{LState} \times \mathsf{LState}) \to \mathcal{P}(\mathsf{LState})$, is defined as follows.

$$l_1 \sqcap l_2 \stackrel{\text{def}}{=} \left\{ l \mid \exists l', l'_1, l'_2.\ l_1 = l \circ l'_1 \wedge l_2 = l \circ l'_2 \wedge l \circ l'_1 \circ l'_2 = l' \right\}$$

Observe that when the separation algebra of logical states satisfies the disjointness property, $l_1 \sqcap l_2$ yields at most one element for any l_1 and l_2.

An action pre-state p and a shared state g then agree if their intersection is non-empty, *i.e.* $p \sqcap g \neq \emptyset$. We can now define action confinement.

Definition 8 (Action confinement)**.** An action a is *confined* to a logical state g, written $g \copyright a$, if for all r compatible with g ($g \sharp r$):

$$\mathit{fst}(a) \sqcap g \neq \emptyset \Rightarrow \mathit{fst}(\Delta a) \leq g \wedge \mathit{fst}(\Delta a) \perp r$$

As discussed, only the *active* precondition of a, *i.e.* the part actually mutated by the action, has to be contained in g and must be disjoint from all potential extensions (r) of the logical state g. That is, future extensions of g need not account for existing actions interfering with new resources.

Given a shared state g and a local action model $\mathcal{I}$, we require that all actions of $\mathcal{I}$ are confined in all possible *futures* of g, *i.e.* all shared states resulting from g after any number of applications of actions in $\mathcal{I}$. For that we define *action application* that describes the effect of an action on a logical state. Moreover, for some of the actions in $\mathcal{I}$, the active pre-state may not affect g, that is, its intersection with g may be empty. In that case, we find that even though that action is potentially enabled, we do not need to account for it since it leaves g unchanged. We thus introduce the notion of *visible actions* to quantify over those actions that affect (mutate) g.

Definition 9 (Action application)**.** The *application* of an action a on a logical state g, written $a[g]$, is defined provided that there exists l such that

$$\mathit{fst}(a) \sqcap g \neq \emptyset \wedge g = \mathit{fst}(\Delta a) \circ l \wedge \mathit{snd}(\Delta a) \sharp l$$

When that is the case, we write $a[g]$ for the (uniquely defined) logical state $\mathit{snd}(\Delta a) \circ l$. We write $\mathit{potential}(a, g)$ to denote that $a[g]$ is defined.

Observe that $\mathit{potential}(a, g)$ implies $g \copyright a$.

Definition 10 (Visible actions)**.** An action a is called *visible in* g, written $\mathit{visible}(a, g)$ when

$$\exists p \in (\mathit{fst}(\Delta a) \sqcap g) \,.\, p \neq \mathbf{0}_{\mathsf{L}}$$

We are now ready to define our confinement condition on action models. Inspired by Local RG [5], we introduce the concept of locally fenced action models to capture all possible states reachable from the current state via some number of action applications. A set of states $\mathcal{F}$ *fences* an action model if it is invariant under interferences perpetrated by the corresponding actions. An action model is then confined to a logical state l if it can be fenced by a set of states that includes l. In the following we write $\mathit{rg}(f)$ to denote the *range* (or co-domain) of a function f.

Definition 11 (Locally-fenced action model)**.** An action model $\mathcal{I} \in \mathsf{AMod}$ is *locally fenced* by $\mathcal{F} \in \mathcal{P}(\mathsf{LState})$, written $\mathcal{F} \blacktriangleright \mathcal{I}$, iff for all $g \in \mathcal{F}$ and all $a \in \mathit{rg}(\mathcal{I})$,

$$g \copyright a \wedge (\mathit{potential}(a, g) \Rightarrow a[g] \in \mathcal{F})$$

Definition 12 (Action model confinement)**.** An action model $\mathcal{I}$ is *confined* to a logical state l, written $l \copyright \mathcal{I}$, if there exists a fence $\mathcal{F}$ such that $l \in \mathcal{F}$ and $\mathcal{F} \blacktriangleright \mathcal{I}$.

We are almost in a position to define well-formedness of worlds. Since capabilities enable the manipulation of the shared state through their associated actions in the action models, for a world $(l, g, \mathcal{I}, \mathcal{G})$ to be well-formed the capabilities found in the local state l and shared state g must be *contained* in the action model $\mathcal{I}$. That is, *all* capabilities found in the combined state $l \circ g$ must be accounted for in $\mathcal{I}$. It is not however necessary for the capabilities to be contained in the global action model $\mathcal{G}$, since as discussed at the beginning of this section, $\mathcal{G}$ records the effect of *some* of the actions in $\mathcal{I}$ and may thus track fewer capabilities and actions. We formalise the relationship between the local and global action models in §1.2.

Definition 13 (Capability containment)**.** A capability $\kappa \in \mathbb{K}$, is *contained* in an action model $\mathcal{I} \in \mathsf{AMod}$, written $\kappa \prec \mathcal{I}$ iff

$$\exists K \in \mathcal{P}(\mathbb{K}) \,.\, \kappa = \prod_{\kappa_i \in K}^{\bullet_{\mathbb{K}}} \kappa_i \wedge \forall \kappa_i \in K. \, \exists \kappa' \in \mathsf{dom}(\mathcal{I}) \,.\, \kappa_i \leq \kappa'$$

We can now formalise the notion of well-formedness. A world $(l, g, \mathcal{I}, \mathcal{G})$ is well-formed if l and g are compatible, the capabilities found in $l \circ g$ are contained in the local action model $\mathcal{I}$, actions in the global action model $\mathcal{G}$ correspond to actions in $\mathcal{I}$, and $\mathcal{I}$ is confined to g.

Definition 14 (Well-formedness)**.** A 4-tuple $(l, g, \mathcal{I}, \mathcal{G})$ is *well-formed*, written $\mathsf{wf}(l, g, \mathcal{I}, \mathcal{G})$, iff

$$\begin{array}{l}(\exists m, \kappa. \, l \circ g = (m, \kappa) \wedge \kappa \prec \mathcal{I}) \wedge \\ (\forall \kappa. \, \forall a \in \mathcal{G}(\kappa). \, \exists a' \in \mathcal{I}(\kappa). \, \Delta a = \Delta a') \wedge g \copyright \mathcal{I}\end{array}$$

Observe that consequently we have $g \copyright \mathcal{G}$.

Definition 15 (Worlds)**.** The set of *worlds* is defined as

$$\mathsf{World} \stackrel{\text{def}}{=} \{w \in \mathsf{LState} \times \mathsf{LState} \times \mathsf{AMod} \times \mathsf{AMod} \mid \mathsf{wf}(w)\}$$

The *composition* $w \bullet w'$ of two worlds $\bullet : \mathsf{World} \to \mathsf{World} \rightharpoonup \mathsf{World}$, is defined as follows.

$$(l, g, \mathcal{I}, \mathcal{G}) \bullet (l', g', \mathcal{I}', \mathcal{G}') \stackrel{\text{def}}{=} \begin{cases} (l \circ l', g, \mathcal{I}, \mathcal{G}) & \text{if } g = g', \mathcal{I} = \mathcal{I}', \text{ and } \mathcal{G} = \mathcal{G}' \\ & \text{and } \mathsf{wf}((l \circ l', g, \mathcal{I}, \mathcal{G})) \\ \textit{undefined} & \text{otherwise} \end{cases}$$

The set of worlds with composition $\bullet$ forms a separation algebra with multiple units: all well-formed states of the form $(\mathbf{0}_{\mathsf{L}}, g, \mathcal{I}, \mathcal{G})$. Given a world w, we write w_{L} for the first projection.

1.2 Assertions

Our assertions extend standard assertions from separation logic with *subjective views* and *capability assertions*. We assume an infinite set, LVar, of *logical variables* and a set of *logical environments* $\iota \in \mathsf{LEnv} : \mathcal{P}(\mathsf{LVar} \to \mathsf{Val})$ that associate logical variables with their values.

CoLoSL is parametric with respect to the machine states and capability assertions and can be instantiated with any assertion language over machine states $\mathbb{M}$ and capabilities $\mathbb{K}$. This is captured by the following parameters.

Parameter 3 (Machine state assertions)**.** Assume a set of *machine state assertions* MAssn, ranged over by $\mathcal{M}, \mathcal{M}_1, \cdots, \mathcal{M}_n$ and an associated semantics function:

$$(\!|.|\!)^{M}_{(.)} : \mathsf{MAssn} \to \mathsf{LEnv} \to \mathcal{P}(\mathbb{M})$$

Parameter 4 (Capability assertions)**.** Assume a set of capability assertions KAssn ranged over by $\mathcal{K}, \mathcal{K}_1, \cdots, \mathcal{K}_n$ and an associated semantics function:

$$(\!|.|\!)^{K}_{(.)} : \mathsf{KAssn} \to \mathsf{LEnv} \to \mathcal{P}(\mathbb{K})$$

Definition 16 (Assertion syntax)**.** The assertions of CoLoSL are elements of Assn described by the grammar below, where x ranges over logical variables.

$$\begin{aligned} A &::= \mathit{false} \mid \mathsf{emp} \mid \mathcal{M} \mid \mathcal{K} \\ \mathsf{Assn} \ni P, Q &::= A \mid P \Rightarrow Q \mid \exists x.\, P \mid P * Q \mid P \circledast Q \mid \boxed{P}_I \\ \mathsf{IAssn} \ni I &::= \emptyset \mid \{K : \exists \vec{y}.\, P \rightsquigarrow Q\} \cup I \end{aligned}$$

This syntax follows from standard separation logic, with the exception of subjective views $\boxed{P}_I$. emp is true of the units of $\bullet$. Machine state assertions ($\mathcal{M}$) and capability assertions ($\mathcal{K}$) are interpreted over a world's local state: $\mathcal{M}$ is true of a local state $(m, \mathbf{0}_{\mathbb{K}})$ where m satisfies $\mathcal{M}$; similarly, $\mathcal{K}$ is true of a local state $(\mathbf{0}_{\mathbb{M}}, \kappa)$ where κ satisfies $\mathcal{K}$. $P * Q$ is true of worlds that can be split into two according to $\bullet$ such that one state satisfies P and the other satisfies Q; $P \circledast Q$ is the *overlapping conjunction*, true of worlds can be split three-way according to $\bullet$, such that the $\bullet$-composition of the first two worlds satisfies P and the $\bullet$-composition of the last two satisfy Q [9]; classical predicates and connectives have their standard classical meaning.

Interference assertions I describe actions enabled by a given capability, in the form of a pre- and post-condition.

A subjective view $\boxed{P}_I$ is true of $(l, g, \mathcal{I}, \mathcal{G})$ when $l = \mathbf{0}_{\mathsf{L}}$ and a subjective view s can be found in the global shared state g, *i.e.* $g = s \circ r$ for some *context* r, such that s satisfies P in the standard separation logic sense, and I, $\mathcal{I}$, and $\mathcal{G}$ *agree* given the decomposition s, r, in the following sense:

1. every action in I is reflected in $\mathcal{I}$;
2. every action in $\mathcal{I}$ that is potentially enabled in g and has a visible effect on s is reflected in I;
3. every action in I that is potentially enabled in g and whose pre-state is fully contained in g is reflected in $\mathcal{G}$;
4. the above holds after any number of action applications in $\mathcal{I}$ affecting g

These conditions will be captured by the *action model closure* relation $(\mathcal{I}, \mathcal{G})\downarrow (s, r, \langle\!|I|\!\rangle_\iota)$ given by the upcoming Def. 20 (where $\langle\!|I|\!\rangle_\iota$ is the interpretation of I given a logical environment ι).

The semantics of CoLoSL assertions is given by a forcing relation $w, \iota \vDash P$ between a world w, a logical environment $\iota \in \mathsf{LEnv}$, and a formula P. We use two auxiliary forcing relations. The first one $l, \iota \vDash_{\mathsf{SL}} P$ interprets formulas P in the usual separation logic sense over a logical state l (and ignores shared state assertions). The second one $s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P$ interprets assertions over a *subjective view* s that is part of the global shared state g, subject to action models $\mathcal{I}$ and $\mathcal{G}$. This third form of satisfaction is needed to deal with nesting of subjective views. We often write $\vDash_\dagger$ as a shorthand for $\vDash_{g,\mathcal{I},\mathcal{G}}$ when we do not need to refer to the individual components $g, \mathcal{I}$ and $\mathcal{G}$.

Note that this presentation with several forcing relations differs from the usual CAP presentation [3], where formulas are first interpreted over worlds that are not necessarily well-formed, and then cut down to well-formed ones. The CAP presentation strays from separation logic models in some respects; for instance, in CAP, $*$ is not the adjoint of $-\!\!*$, the "magic wand" connective of separation logic. Although we have omitted this connective from our presentation, its definition in CoLoSL would be standard and satisfy the adjunction with $*$.

Definition 17 (Assertion semantics)**.** Given a logical environment $\iota \in \mathsf{LEnv}$, the semantics of CoLoSL assertions is as follows, where $\langle\!|.|\!\rangle_{(.)} : \mathsf{LEnv} \to \mathsf{AMod}$ denotes the semantics of interference assertions and $(\mathcal{I}, \mathcal{G})\downarrow (s, r, \langle\!|I|\!\rangle_\iota)$ will

be given in Def. 20.

$$
\begin{array}{rll}
(l, g, \mathcal{I}, \mathcal{G}), \iota \vDash A & \text{iff} & l, \iota \vDash_{\mathsf{SL}} A \\
(l, g, \mathcal{I}, \mathcal{G}), \iota \vDash \boxed{P}_I & \text{iff} & l = \mathbf{0}_{\mathbb{M}} \text{ and } \exists s, r.\, g = s \circ r \text{ and} \\
 & & \quad s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P \text{ and } (\mathcal{I}, \mathcal{G}){\downarrow}\,(s, r, \langle\!|I|\!\rangle_\iota) \\
w, \iota \vDash P \Rightarrow Q & \text{iff} & w, \iota \vDash P \text{ implies } w, \iota \vDash Q \\
w, \iota \vDash \exists x.\, P & \text{iff} & \exists v.\, w, [\iota \mid x : v] \vDash P \\
w, \iota \vDash P_1 * P_2 & \text{iff} & \exists w_1, w_2.\, w = w_1 \bullet w_2 \text{ and} \\
 & & \quad w_1, \iota \vDash P_1 \text{ and } w_2, \iota \vDash P_2 \\
w, \iota \vDash P_1 \mathbin{-\!\!*} P_2 & \text{iff} & \exists w', w_1, w_2.\, w = w' \bullet w_1 \bullet w_2 \text{ and} \\
 & & \quad w' \bullet w_1, \iota \vDash P_1 \text{ and } w' \bullet w_2, \iota \vDash P_2
\end{array}
$$

where

$$
\begin{array}{lll}
s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} A & \text{iff} & s, \iota \vDash_{\mathsf{SL}} A \\
s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} \boxed{P}_I & \text{iff} & (s, g, \mathcal{I}, \mathcal{G}), \iota \vDash \boxed{P}_I \\
s, \iota \vDash_\dagger P \Rightarrow Q & \text{iff} & s, \iota \vDash_\dagger P \text{ implies } s, \iota \vDash_\dagger Q \\
s, \iota \vDash_\dagger \exists x.\, P & \text{iff} & \exists v.\, s, [\iota \mid x : v] \vDash_\dagger P \\
s, \iota \vDash_\dagger P_1 * P_2 & \text{iff} & \exists s_1, s_2.\, s = s_1 \circ s_2 \text{ and} \\
 & & \quad s_1, \iota \vDash_\dagger P_1 \text{ and } s_2, \iota \vDash_\dagger P_2 \\
s, \iota \vDash_\dagger P_1 \mathbin{-\!\!*} P_2 & \text{iff} & \exists s', s_1, s_2.\, s = s' \circ s_1 \circ s_2 \text{ and} \\
 & & \quad s' \circ s_1, \iota \vDash_\dagger P_1 \text{ and } s' \circ s_2, \iota \vDash_\dagger P_2
\end{array}
$$

$$
\begin{array}{lll}
l, \iota \vDash_{\mathsf{SL}} \mathit{false} & & \text{never} \\
l, \iota \vDash_{\mathsf{SL}} \mathsf{emp} & \text{iff} & l = \mathbf{0}_{\mathsf{L}} \\
l, \iota \vDash_{\mathsf{SL}} \mathcal{M} & \text{iff} & \exists m.\, l = (m, \mathbf{0}_{\mathbb{K}}) \text{ and } m \in (\!|\mathcal{M}|\!)^M_\iota \\
l, \iota \vDash_{\mathsf{SL}} \mathcal{K} & \text{iff} & \exists \kappa.\, l = (\mathbf{0}_{\mathbb{M}}, \kappa) \text{ and } \kappa \in (\!|\mathcal{K}|\!)^K_\iota \\
l, \iota \vDash_{\mathsf{SL}} \boxed{P}_I & \text{iff} & l = \mathbf{0}_{\mathsf{L}} \\
l, \iota \vDash_{\mathsf{SL}} P \Rightarrow Q & \text{iff} & l, \iota \vDash_{\mathsf{SL}} P \text{ implies } l, \iota \vDash_{\mathsf{SL}} Q \\
l, \iota \vDash_{\mathsf{SL}} \exists x.\, P & \text{iff} & \exists v.\, l, [\iota \mid x : v] \vDash_{\mathsf{SL}} P \\
l, \iota \vDash_{\mathsf{SL}} P_1 * P_2 & \text{iff} & \exists l_1, l_2.\, l = l_1 \circ l_2 \text{ and} \\
 & & \quad l_1, \iota \vDash_{\mathsf{SL}} P_1 \text{ and } l_2, \iota \vDash_{\mathsf{SL}} P_2 \\
l, \iota \vDash_{\mathsf{SL}} P_1 \mathbin{-\!\!*} P_2 & \text{iff} & \exists l', l_1, l_2.\, l = l' \circ l_1 \circ l_2 \text{ and} \\
 & & \quad l' \circ l_1, \iota \vDash_{\mathsf{SL}} P_1 \text{ and } l' \circ l_2, \iota \vDash_{\mathsf{SL}} P_2
\end{array}
$$

and

$$
\langle\!|I|\!\rangle_\iota(\kappa) \stackrel{\text{def}}{=} \left\{ (p, q) \;\middle|\; \begin{array}{l} \kappa : \exists \vec{y}.\, P \rightsquigarrow Q \in I \wedge \\ \exists \vec{v}.\, p, [\iota \mid \vec{y} : \vec{v}] \vDash_{\mathsf{SL}} P \wedge q, [\iota \mid \vec{y} : \vec{v}] \vDash_{\mathsf{SL}} Q \end{array} \right\}
$$

Given a logical environment ι, we write $[\![P]\!]_\iota$ for the set of all worlds satisfying P. That is,

$$
[\![P]\!]_\iota \stackrel{\text{def}}{=} \{w \mid w, \iota \vDash P\}
$$

The satisfaction relation for subjective views in $\vDash_{\mathsf{SL}}$ is purely for convenience, as in CAP [3]: this allows us to write predicates in interference assertions whose definition includes subjective views, as in the example of §3.2. Since actions are not allowed to modify shared state from within another subjective view, the interpretation of such subjective views collapses to emp.

Lemma 1. The following formulas are valid according to the semantics above (where x does not appear free in I):

$$\boxed{P * \boxed{Q}_{I'}}_{I} \Leftrightarrow \boxed{P}_{I} * \boxed{Q}_{I'} \qquad \exists x.\, \boxed{P}_{I} \Leftrightarrow \boxed{\exists x.\, P}_{I}$$
$$(P \Rightarrow Q) \Rightarrow \boxed{P}_{I} \Rightarrow \boxed{Q}_{I} \qquad \boxed{P}_{I} \Rightarrow \mathsf{emp}$$

Proof. Immediate. □

Action Model Closure Let us now turn to the definition of action model closure, as informally introduced at the beginning of this section. First, we need to revisit the effect of actions to take into account the splitting of the global shared state into a subjective state s and a context r.

Definition 18 (Action application (cont.))**.** The *application* of action a on the subjective state s together with the context r, written $a[s, r]$, is defined provided that $a[s \circ r]$ is defined. When that is the case, we write $a[s, r]$ for

$$\left\{ (snd(\Delta a) \circ s', r') \;\middle|\; \begin{array}{l} fst(\Delta a) = p_s \circ p_r \wedge p_s > \mathbf{0}_{\mathsf{L}} \\ \wedge\ s = p_s \circ s' \wedge r = p_r \circ r' \end{array} \right\}$$
$$\cup \left\{ (s, snd(\Delta a) \circ r') \mid r = fst(\Delta a) \circ r' \right\}$$

We observe that this new definition and Def. 9 are linked in the following way:

$$\forall s', r' \in a[s, r].\, s' \circ r' = a[s \circ r]$$

In our informal description of action model closure in Def. 16 we stated that some actions must be *reflected* in some action models. Intuitively, an action is reflected in an action model if for every state in which the action can take place, the action model includes an action with a similar effect that can also occur in that state. In other words, a is reflected in $\mathcal{I}$ if given any state l that contains the pre-state of a ($fst\,(a) \leq l$), there exists an action $a' \in \mathcal{I}$ with the same active part ($\Delta a = \Delta a'$) whose pre-state is also contained in l ($fst\,(a') \leq l$). We proceed with the definition of action reflection.

Definition 19. An action a is *reflected* in a set of actions A from a state l, written $reflected(a, l, A)$, if

$$\forall r.\, fst(a) \leq l \circ r \Rightarrow \exists a' \in A.\, \Delta a' = \Delta a \wedge fst(a') \leq l \circ r$$

Let us now give the formal definition of action model closure. For each condition mentioned at the beginning of this section, we annotate which part of the definition implements them. We write

$$\mathit{enabled}(a, g) \stackrel{\text{def}}{=} \mathit{potential}(a, g) \wedge \mathit{fst}(a) \leq g$$

That is, a can actually happen in g since g holds all the resources in the pre-state of a.

Definition 20 (Action model closure)**.** A pair $(\mathcal{I}, \mathcal{G})$ of action models is *closed* under a subjective state s, context r, and action model $\mathcal{I}'$, written $(\mathcal{I}, \mathcal{G})\downarrow (s, r, \mathcal{I}')$, if $\mathcal{I}' \subseteq \mathcal{I}$ (*cf.* property (1) in Def. 16) and $(\mathcal{I}, \mathcal{G})\downarrow_n (s, r, \mathcal{I}')$ for all $n \in \mathbb{N}$, where $\downarrow_n$ is defined recursively as follows:

$$
\begin{array}{ll}
(\mathcal{I}, \mathcal{G})\downarrow_0 (s, r, \mathcal{I}') \stackrel{\text{def}}{\iff} \mathit{true} & \\
(\mathcal{I}, \mathcal{G})\downarrow_{n+1} (s, r, \mathcal{I}') \stackrel{\text{def}}{\iff} & \\
\quad (\forall \kappa.\, \forall a \in \mathcal{I}'(\kappa). & \\
\quad\quad (\mathit{potential}(a, s \circ r) \Rightarrow \forall (s', r') \in a[s, r].\, (\mathcal{I}, \mathcal{G})\downarrow_n (s', r', \mathcal{I}')) \wedge & \textit{(4)} \\
\quad\quad (\mathit{enabled}(a, s \circ r) \Rightarrow (s \circ r, a[s \circ r]) \in \mathcal{G}(\kappa))) \wedge & \textit{(3)} \\
\quad (\forall \kappa.\, \forall a \in \mathcal{I}(\kappa).\, \mathit{potential}(a, s \circ r) \Rightarrow & \\
\quad\quad \mathit{reflected}(a, s \circ r, \mathcal{I}'(\kappa)) \vee & \textit{(2)} \\
\quad\quad \neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\, (\mathcal{I}, \mathcal{G})\downarrow_n (s', r', \mathcal{I}')) & \textit{(4)}
\end{array}
$$

Recall from the semantics of the assertions (Def. 17) that $\mathcal{I}'$ corresponds to the interpretation of an interference assertion I in a subjective view. The first condition($\mathcal{I}' \subseteq \mathcal{I}$) asserts that the subjective action model $\mathcal{I}'$ is contained in $\mathcal{I}$; consequently (from the semantics of subjective assertions), $\mathcal{I}$ represents the superset of all interferences known to subjective views.

The first conjunct (I) states that for any action a known to the subjective view, where it *may* be possible for a to take place on the global state $s \circ r$, then

We make further observations about this definition. First, property (4) makes our assertions robust with respect to future extensions of the shared state, where potentially enabled actions may be become enabled using additional catalyst that is not immediately present. Second, $\mathcal{G}$ only has to record actions from $\mathcal{I}'$ that are enabled in any g resulting from these steps, hence immediately possible. Finally, $\mathcal{I}'$ (and thus interference assertions) need not reflect actions that have no visible effect on the subjective state.

This completes the definition of the semantics of assertions. We can now show that the logical principles of CoLoSL are sound. The proof of the Shift and Extend principles will be delayed until the following chapter, where $\Rrightarrow$ is defined.

Lemma 2. The logical implications Copy, Forget, and Merge are *valid*; *i.e.* true of all worlds and logical interpretations.

Proof. The case of Copy is straightforward from the semantics of assertions. In order to show that the Forget implication is valid, it suffices to show:

$$\forall \iota \in \mathsf{LEnv}.\ \left\{w \mid w, \iota \vDash \boxed{P \circledast Q}_I\right\} \subseteq \left\{w \mid w, \iota \vDash \boxed{P}_I\right\}$$

We first demonstrate that, whenever an action model closure relation holds for a subjective state $s_1 \circ s_2$ and context r, then it also holds for the smaller subjective state s_1, and the larger context extended with the forgotten state. That is,

$$\begin{aligned}\forall s_1, s_2, r \in \mathsf{LState}.\, \forall \mathcal{I}, \mathcal{I}', \mathcal{G} \in \mathsf{AMod}.\\ (\mathcal{I}, \mathcal{G})\downarrow (s_1 \circ s_2, r, \mathcal{I}') \implies (\mathcal{I}, \mathcal{G})\downarrow (s_1, s_2 \circ r, \mathcal{I}')\end{aligned}$$

This is formalised in Lemma 9 of §A. We then proceed to establish the desired result by calculation:

$$\begin{aligned}\left\{w \mid w, \iota \vDash \boxed{P \circledast Q}_I\right\} &= \left\{ \begin{pmatrix}(\mathbf{0}_\mathsf{L},\\ (s \circ r)\\ ,\mathcal{I},\mathcal{G}\end{pmatrix} \;\middle|\; \begin{array}{l}\exists s_p, s_c, s_q.\, s = s_p \circ s_c \circ s_q\\ \wedge r \in \mathsf{LState}\\ \wedge (s_p \circ s_c), \iota \vDash_{(s \circ r),\mathcal{I},\mathcal{G}} P\\ \wedge (s_q \circ s_c), \iota \vDash_{(s \circ r),\mathcal{I},\mathcal{G}} Q\\ \wedge\, (\mathcal{I}, \mathcal{G})\downarrow (s_p \circ s_c \circ s_q,\ r, \langle\!|I|\!\rangle_\iota)\end{array} \right\}\\ \text{By Lemma 9} &\subseteq \left\{ \begin{pmatrix}(\mathbf{0}_\mathsf{L},\\ (s \circ r)\\ ,\mathcal{I},\mathcal{G}\end{pmatrix} \;\middle|\; \begin{array}{l}\exists s_p, s_c, s_q.\, s = s_p \circ s_c \circ s_q\\ \wedge r \in \mathsf{LState}\\ \wedge (s_p \circ s_c), \iota \vDash_{(s \circ r),\mathcal{I},\mathcal{G}} P\\ \wedge (s_q \circ s_c), \iota \vDash_{(s \circ r),\mathcal{I},\mathcal{G}} Q\\ \wedge\, (\mathcal{I}, \mathcal{G})\downarrow (s_p \circ s_c,\ s_q \circ r, \langle\!|I|\!\rangle_\iota)\end{array} \right\}\\ &\subseteq \left\{ \begin{pmatrix}(\mathbf{0}_\mathsf{L},\\ (s_p \circ r)\\ ,\mathcal{I},\mathcal{G}\end{pmatrix} \;\middle|\; \begin{array}{l} s_p, \iota \vDash_{(s_p \circ r),\mathcal{I},\mathcal{G}} P \wedge r \in \mathsf{LState}\\ \wedge\, (\mathcal{I}, \mathcal{G})\downarrow (s_p,\ r, \langle\!|I|\!\rangle_\iota)\end{array} \right\}\\ &= \left\{w \mid w, \iota \vDash \boxed{P}_I\right\}\end{aligned}$$

as required.

Similarly, to show that the Merge implication is valid, it suffices to show:

$$\forall \iota \in \mathsf{LEnv}.\ \left\{w \mid w, \iota \vDash \boxed{P}_{I_1} * \boxed{Q}_{I_2}\right\} \subseteq \left\{w \mid w, \iota \vDash \boxed{P \circledast Q}_{I_1 \cup I_2}\right\}$$

We first demonstrate that, whenever an action model closure relation holds for two (potentially) different subjective states (that may overlap), subject

to two (potentially) different interference relations, then the closure relation also holds for the combined subjective states and their associated interference relations. That is:

$$\begin{array}{l}\forall s_p, s_q, s_c, r \in \mathsf{LState}.\, \forall \mathcal{I}, \mathcal{I}_1, \mathcal{I}_2, \mathcal{G} \in \mathsf{AMod}.\, \forall n \in \mathbb{N}. \\ \quad (\mathcal{I}, \mathcal{G}){\downarrow_n}\ (s_p \circ s_c, s_q \circ r, \mathcal{I}_1)\ \wedge\ (\mathcal{I}, \mathcal{G}){\downarrow_n}\ (s_q \circ s_c, s_p \circ r, \mathcal{I}_2)\ \Longrightarrow \\ \qquad (\mathcal{I}, \mathcal{G}){\downarrow_n}\ (s_p \circ s_c \circ s_q, r, \mathcal{I}_1 \cup \mathcal{I}_2)\end{array}$$

This is formalised in Lemma 10 of §A. We then proceed to establish the desired result by calculation:

$$\begin{array}{rcl} & & \left\{ w \mid w, \iota \vDash \boxed{P}_{I_1} * \boxed{Q}_{I_2} \right\} \\ & = & \left\{ (\mathbf{0}_{\mathsf{L}}, s \circ r, \mathcal{I}, \mathcal{G}) \;\middle|\; \begin{array}{l} \exists s_p, s_c, s_q.\, s = s_p \circ s_c \circ s_q \\ \wedge r \in \mathsf{LState} \\ \wedge (s_p \circ s_c), \iota \vDash_{(s \circ r), \mathcal{I}, \mathcal{G}} P \\ \wedge (s_q \circ s_c), \iota \vDash_{(s \circ r), \mathcal{I}, \mathcal{G}} Q \\ \wedge\, (\mathcal{I}, \mathcal{G}){\downarrow}\, (s_p \circ s_c,\ s_q \circ r, \langle\!| I_1 |\!\rangle_\iota) \\ \wedge\, (\mathcal{I}, \mathcal{G}){\downarrow}\, (s_c \circ s_q,\ s_p \circ r, \langle\!| I_2 |\!\rangle_\iota) \end{array} \right\} \\ \text{By Lemma 10} \subseteq & \left\{ (\mathbf{0}_{\mathsf{L}}, s \circ r, \mathcal{I}, \mathcal{G}) \;\middle|\; \begin{array}{l} \exists s_p, s_c, s_q.\, s = s_p \circ s_c \circ s_q \\ \wedge r \in \mathsf{LState} \\ \wedge (s_p \circ s_c), \iota \vDash_{(s \circ r), \mathcal{I}, \mathcal{G}} P \\ \wedge (s_q \circ s_c), \iota \vDash_{(s \circ r), \mathcal{I}, \mathcal{G}} Q \\ \wedge\, (\mathcal{I}, \mathcal{G}){\downarrow}\, (s_p \circ s_c \circ s_q, r, \langle\!| I_1 \cup I_2 |\!\rangle_\iota) \end{array} \right\} \\ & \subseteq & \left\{ (\mathbf{0}_{\mathsf{L}}, s \circ r, \mathcal{I}, \mathcal{G}) \;\middle|\; \begin{array}{l} s, \iota \vDash_{(s \circ r), \mathcal{I}, \mathcal{G}} P \curlywedge Q \\ \wedge r \in \mathsf{LState} \\ \wedge\, (\mathcal{I}, \mathcal{G}){\downarrow}\, (s, r, \langle\!| I_1 \cup I_2 |\!\rangle_\iota) \end{array} \right\} \\ & = & \left\{ w \mid w, \iota \vDash \boxed{P \curlywedge Q}_{I_1 \cup I_2} \right\} \end{array}$$

□

Note that, the version of FORGET where predicates are conjoined using $\curlywedge$ is also valid, as shown by the following derivation, valid for all P, Q, and I, which uses a property of $\curlywedge$ and WEAKEN in the first step and FORGET in the second:

$$\boxed{P \curlywedge Q}_I \Rightarrow \boxed{P * \mathit{true}}_I \Rightarrow \boxed{P}_I$$

The following examples illustrate the need for both local and global action models in worlds.

Example 1. This example showcases subtle reasoning about subjective views enabled by our logic thanks to the global action model. The following entailment is valid, where $I \stackrel{\text{def}}{=} (\mathsf{a} : \{x \mapsto 0 * y \mapsto 0 \rightsquigarrow x \mapsto 0 * y \mapsto 1\})$:

$$\boxed{(x \mapsto 0 * y \mapsto 0) \vee x \mapsto 1}_I * \boxed{y \mapsto 0}_\emptyset \vdash \boxed{x \mapsto 1}_I * \boxed{y \mapsto 0}_\emptyset$$

Assertion $\boxed{(x \mapsto 0 * y \mapsto 0) \vee x \mapsto 1}_I$ is true of states $(\mathbf{0}_\mathsf{L}, g, \mathcal{I}, \mathcal{G})$ such that parts of g satisfies $x \mapsto 0 * y \mapsto 0$, in which case $\mathcal{G}$ includes the action associated with a in I according to our closure condition, and of states $(\mathbf{0}_\mathsf{L}, g', \mathcal{I}', \mathcal{G}')$ where parts of g' satisfies $x \mapsto 1$, in which case $\mathcal{G}'$ need not contain the action. The second conjunct $\boxed{y \mapsto 0}_\emptyset$ does not allow action a to appear in the global action models, because it potentially affects $y \mapsto 0$ yet is not included in $\emptyset$. Thus, combining the two conjuncts with $*$ allows us to deduce that $x \mapsto 0 * y \mapsto 0$ is not possible, thanks to knowledge captured in the global action model.

Example 2. This example motivates the need for a local action model in worlds. Let $P \stackrel{\text{def}}{=} x \mapsto 0 * \boxed{y \mapsto 0}_I$, where I is as in the previous example. Since $x \mapsto 0$ is present in the local state, and the local and shared states are always disjoint, the action associated with a cannot be carried out by either the current thread or the environment. Thus the global action model $\mathcal{G}$ of worlds satisfying P need not account for this action: *e.g.* $\mathcal{G}(\{\mathsf{a}\}) = \emptyset$ is possible.

Consider now the situation where the current thread extends the shared state with the $x \mapsto 0$ resource previously held locally, and specifies no actions for it, yielding the subjective view $P' \stackrel{\text{def}}{=} \boxed{x \mapsto 0}_\emptyset * \boxed{y \mapsto 0}_I$. Since the new shared state contains both x and y, it might be possible for a thread in possession of a to perform the associated action. In particular, similarly to our previous example, the following entailment is valid:

$$P' * \boxed{y \mapsto 0}_\emptyset \vdash \mathit{false}$$

Without local action models in our worlds, *i.e.* from the global action models of the previous state alone we could not have made that deduction, as some of the global action models "forgot" about the action which was impossible at the time, making the composition of P' with $\boxed{y \mapsto 0}_\emptyset$ satisfiable.

Note that, while sometimes subtle, our semantic conditions are often easy to check in practice. Actions are usually *small* and typically only concern a few cells at a time. Indeed, they are the by-product of atomic sections in program, which are also kept small.

2. CoLoSL Program Logic

This section introduces the core concepts behind our program logic. We start by defining what the rely and guarantee conditions of each thread are in terms of their action models. This allows us to define *stability* against rely conditions, *repartitioning*, which logically represents a thread's atomic actions (and have to be in the guarantee condition), and *semantic implication*. Equipped with these notions, we can justify the SHIFT and EXTEND principles.

2.1 Proof System

Rely The rely relation represents potential interferences from the environment. Although the rely will be different for every program (and indeed, every thread), it is always defined in the same way, which we can break down into three kinds of possible interferences. In the remainder of this section, given a logical state $l = (m, \kappa)$, we write l_{M}, l_{K} for $fst(l)$ and $snd(l)$, respectively.

The first relation, R^{e}, extends the shared state g with new state g', along with a new interference $\mathcal{I}'$ on g'. We proceed with the definition of *action model extension* that captures the extension of the local action model in such a way that respects all previous subjective views of a world; in doing so, the global action model may change provided that the action model closure relation is preserved.

Definition 21 (Action model extension)**.** An action model $\mathcal{G}'$ *extends* $(g, \mathcal{I}, \mathcal{G})$ with $(g', \mathcal{I}')$, written $\mathcal{G}' \uparrow^{(g', \mathcal{I}')} (g, \mathcal{I}, \mathcal{G})$, iff for all $\mathcal{I}_0 \subseteq \mathcal{I}$ and s, r such that $s \circ r = g$:

$$(\mathcal{I}, \mathcal{G})\!\downarrow (s, r, \mathcal{I}_0) \implies \big(\mathcal{I} \cup \mathcal{I}', \mathcal{G}'\big)\!\downarrow \big(s, r \circ g', \mathcal{I}_0\big)$$

Then the extension rely, R^{e}, is defined as

$$R^{\mathsf{e}} \stackrel{\text{def}}{=} \left\{ \begin{pmatrix} (l, g, \mathcal{I}, \mathcal{G}), \\ (l, g \circ g', \mathcal{I} \cup \mathcal{I}', \mathcal{G}') \end{pmatrix} \;\middle|\; \begin{array}{l} g'_{\mathsf{K}} \prec \mathsf{dom}\,(\mathcal{I}') \cup \mathsf{dom}\,(\mathcal{I}) \wedge \\ (\mathcal{I} \cup \mathcal{I}', \mathcal{G}') \downarrow (g', g, \mathcal{I}') \wedge \\ \mathcal{G}' \uparrow^{(g', \mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \end{array} \right\}$$

The second kind of interference is the *update* of the global state according to actions in the global action model whose capability is "owned by the environment", *i.e.*, nowhere to be found in the current local and shared states.

$$R^{\mathsf{u}} \stackrel{\text{def}}{=} \left\{ \left((l, g, \mathcal{I}, \mathcal{G}),\ (l, g', \mathcal{I}, \mathcal{G}) \right) \middle| \exists \kappa.\ (l_{\mathsf{K}} \bullet_{\mathbb{K}} g_{\mathsf{K}}) \sharp \kappa \wedge (g, g') \in \mathcal{G}(\kappa) \right\}$$

The third and last kind of interference is the *shifting* of the local interference relation to a new one that allows more actions while preserving all current subjective views (as expressed by the global action model).

$$R^{\mathsf{s}} \stackrel{\text{def}}{=} \left\{ \left((l, g, \mathcal{I}, \mathcal{G}), (l, g, \mathcal{I} \cup \mathcal{I}', \mathcal{G}) \right) \middle| \mathcal{G} \uparrow^{(\mathbf{0}_{\mathsf{L}}, \mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \right\}$$

Definition 22 (Rely)**.** The *rely* relation $R : \mathcal{P}\,(\mathsf{World} \times \mathsf{World})$ is defined as follows, where $(\cdot)^*$ denotes the reflexive transitive closure:

$$R \stackrel{\text{def}}{=} \left(R^{\mathsf{u}} \cup R^{\mathsf{e}} \cup R^{\mathsf{s}} \right)^*$$

The rely relation enables us to define the stability of assertions with respect to the environment actions.

Definition 23 (Stability)**.** An assertion P is *stable* ($\mathsf{Stable}\,(P)$) if, for all $\iota \in \mathsf{LEnv}$ and $w, w' \in \mathsf{World}$, if $w, \iota \vDash P$ and $(w, w') \in R$, then $w', \iota \vDash P$.

Proving that an assertion is stable is not always obvious, in particular when there are numerous transitions to consider (all those in R^{e}, R^{u}, R^{s}); as it turns out, we only need to check stability against update actions in R^{u}, as expressed by the following lemma.

Lemma 3 (Stability)**.** If an assertion P is stable with respect to actions in R^{u}, then it is stable.

Proof. **TO DO** □

Guarantee We now define the guarantee relation that describes all possible updates the current thread can perform. In some sense, the guarantee relation is the dual of rely: the actions in the guarantee of one thread are included in the rely of concurrently running threads. Thus, it should come as no surprise that transitions in the guarantee can be categorised using three categories which resonate with those of the rely. The *extension* guarantee is similar to the extension rely except that new capabilities corresponding to the new shared region are materialised in the local and shared state, and a part of the local state is moved into the shared state:

$$G^{\mathsf{e}} \stackrel{\text{def}}{=} \left\{ \begin{pmatrix} (l \circ l', g, \mathcal{I}, \mathcal{G}), \\ l \circ (\mathbf{0}_{\mathbb{M}}, \kappa_1), g \circ g', \mathcal{I} \cup \mathcal{I}', \mathcal{G}' \end{pmatrix} \;\middle|\; \begin{array}{l} g' = l' \circ (\mathbf{0}_{\mathbb{M}}, \kappa_2) \wedge \\ \kappa_1 \bullet_{\mathbb{K}} \kappa_2 \prec \mathsf{dom}\,(\mathcal{I}') \wedge \\ \kappa_1 \bullet_{\mathbb{K}} \kappa_2 \perp \mathsf{dom}\,(\mathcal{I}) \wedge \\ (\mathcal{I} \cup \mathcal{I}', \mathcal{G}') \downarrow (g', g, \mathcal{I}') \wedge \\ \mathcal{G}' \uparrow^{(g', \mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \end{array} \right\}$$

The current thread may at any point extend the shared state with some of its locally held resources l'; introduce new interference to describe how the new resources may be mutated ($\mathcal{I}'$) and generate new capabilities ($\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \prec \mathsf{dom}\,(\mathcal{I}_0)$) that facilitate the new interference, with the proviso that the new capabilities are fresh ($\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \perp \mathsf{dom}\,(\mathcal{I})$). The last two conjuncts enforce closure of the new action models and can be justified as in the case of R^{e}.

The *update guarantee* G^{u} is more involved than its R^{u} counterpart, because updates in the guarantee may move resources from the local state into the shared state (similarly to extensions above) at the same time that it mutates them as prescribed by an enabled action. Intuitively, we want to enforce that resources are not created "out of thin air" in the process. This can be expressed as preserving the *orthogonal* of the combination of the local and global states, *i.e.*, the set of states compatible with that combination.

Definition 24 (Orthogonal)**.** Given any separation algebra $(\mathbb{B}, \bullet_{\mathbb{B}}, \mathbf{0}_{\mathbb{B}})$, and an element $b \in \mathbb{B}$, its *orthogonal* $(.)^{\perp}_{\mathbb{B}} : \mathbb{B} \to \mathcal{P}(\mathbb{B})$, is defined as the set of all elements in $\mathbb{B}$ that are compatible with it:

$$(b)^{\perp}_{\mathbb{B}} \stackrel{\text{def}}{=} \left\{ b' \mid b \sharp b' \right\}$$

The *update guarantee* G^{u} can then be defined as follows. When updating the shared state, thread are not allowed to introduce new capabilities, as

this can only be achieved when extending the shared state (through G^{e}).

$$G^{\mathsf{u}} \stackrel{\text{def}}{=} \left\{ ((l,g,\mathcal{I},\mathcal{G}),(l',g',\mathcal{I},\mathcal{G})) \;\middle|\; \begin{array}{l} ((l' \circ s')_{\mathsf{K}})^{\perp}_{\mathbb{K}} = ((l \circ g)_{\mathsf{K}})^{\perp}_{\mathbb{K}} \wedge \\ \left(\begin{array}{l} g = g' \vee \\ \left(\begin{array}{l} \exists \kappa \leq l_{\mathsf{K}}.\, (g,g') \in \mathcal{G}(\kappa) \wedge \\ ((l \circ g)_{\mathsf{M}})^{\perp}_{\mathbb{M}} = ((l' \circ g')_{\mathsf{M}})^{\perp}_{\mathbb{M}} \end{array} \right) \end{array} \right) \end{array} \right\}$$

Lastly, the current thread may extend the local action model by shifting some of the existing interference. This is modelled by the G^{s} relation which is analogous to R^{s}.

$$G^{\mathsf{s}} \stackrel{\text{def}}{=} \left\{ \left((l,g,\mathcal{I},\mathcal{G}),(l,g,\mathcal{I} \cup \mathcal{I}_0,\mathcal{G})\right) \middle| \mathcal{G}\!\uparrow^{(\mathbf{0}_{\mathsf{L}},\mathcal{I}_0)} (g,\mathcal{I},\mathcal{G}) \right\}$$

Definition 25 (Guarantee)**.** The *guarantee* relation $G : \mathcal{P}(\mathsf{World} \times \mathsf{World})$ is defined as

$$G \stackrel{\text{def}}{=} \left(G^{\mathsf{u}} \cup G^{\mathsf{e}} \cup G^{\mathsf{s}}\right)^*$$

Using the guarantee relation, we introduce the notion of *repartitioning* $P \Rrightarrow^{\{R_1\}\{R_2\}} Q$. This relation holds whenever, from any world satisfying P, if whenever parts of the composition of its local and shared states that satisfies R_1 is exchanged for one satisfying R_2, it is possible to split the resulting logical state into a local and shared part again, in such a way that the resulting transition is in G.

Definition 26 (Repartitioning)**.** We write $P \Rrightarrow^{\{R_1\}\{R_2\}} Q$ if, for every $\iota \in \mathsf{LEnv}$, and world w_1 such that $w_1, \iota \vDash P$, there exists states $m_1, m' \in \mathbb{M}$ such that $(m_1, \emptyset), \iota \vDash_{\mathsf{SL}} R_1$ and

- $m_1 \bullet_{\mathbb{M}} m' = (\downarrow(w_1))_{\mathsf{M}}$; and
- for every m_2 where $(m_2, \emptyset), \iota \vDash_{\mathsf{SL}} R_2$, there exists a world w_2 such that $w_2, \iota \vDash Q$; and
  - $m_2 \bullet_{\mathbb{M}} m' = (\downarrow(w_2))_{\mathsf{M}}$; and
  - $(w_1, w_2) \in G$

We write $P \Rrightarrow Q$ for $P \Rrightarrow^{\{\mathsf{emp}\}\{\mathsf{emp}\}} Q$, in which case the repartitioning has no "side effect" and simply shuffles resources around between the local and shared state or modifies the action models. This is the case for (Shift) and (Extend), whose proof will be given in the next section.

2.1.1 Interference Manipulations

In this section we formalise the requirements of the Extend and Shift semantic implications and show that they are valid.

Shared State Extension When extending the shared state using currently owned local resources, one specifies a new interference assertion over these newly shared resources. While in CoLoSL the new interferences may mention parts of the shared state beyond the newly added resources (in particular the existing shared state), they must not allow visible updates to those parts, so as not to invalidate other threads' views of existing resources. We thus impose a locality condition on the newly added behaviour to ensure sound extension of the shared state, similarly to the confinement constraint of local fences of Def. 8. We first motivate this constraint with an example.

Example 3. Let $P \stackrel{\text{def}}{=} x \mapsto 1 * \boxed{y \mapsto 1 \vee y \mapsto 2}_I$ denote the view of the current thread with $I \stackrel{\text{def}}{=} (\mathsf{b} : \{y \mapsto 1 \rightsquigarrow y \mapsto 2\})$. Since the current thread owns the location addressed by x, it can extend the shared state as $Q \stackrel{\text{def}}{=} [\mathsf{a}] * \boxed{(y \mapsto 1 \vee y \mapsto 2) * x \mapsto 1}_{I \cup I'}$ where $I' \stackrel{\text{def}}{=} (\mathsf{a} : x \mapsto 1 \rightsquigarrow x \mapsto 2)$. In extending the shared state, the current thread also extended the interference allowed on the shared state by adding a new action associated with the newly generated capability resource $[\mathsf{a}]$, as given in I', which updates the value of location x. Since location x was previously owned privately by the current thread and was hence not visible to other threads, this new action will not invalidate their view of the shared state, hence this extension is a valid one.

If, on the other hand, I' is replaced with $I'' \stackrel{\text{def}}{=} (\mathsf{a} : \{y \mapsto 1 \rightsquigarrow y \mapsto 3\})$, where location y can be mutated, the situation above is not allowed. Indeed, other threads may rely on the fact that the only updates allowed on location y are done through the $[\mathsf{b}]$ capability as specified in I, and would be spooked by this new possible behaviour they were not aware of (as it is not in I).

In order to ensure sound extension of the shared state, we require in EXTEND that the newly introduced interferences are confined to the locally owned resources.

Definition 27. A set of states $\mathcal{P}$ *contains* an action model $\mathcal{G}$, written $\mathcal{P} \copyright \mathcal{G}$, if

$$\exists \mathcal{F}.\, \mathcal{P} \subseteq \mathcal{F} \wedge \mathcal{F} \blacktriangleright \mathcal{G}$$

The $P \copyright I$ notation used in EXTEND is then a straightforward lift of this definition to assertions $P \in \mathsf{Assn}$ and interference assertions $I \in \mathsf{IAssn}$:

$$P \copyright I \iff^{\text{def}} \forall \iota, \kappa.\ \{l \mid l, \iota \vDash_{\mathsf{SL}} P\} \copyright \langle\!| I |\!\rangle_\iota (\kappa)$$

Definition 28 (Freshness)**.** The capability assertion $\mathcal{K}$ is *fresh* with respect to assertion P and logical variables $\vec{x}$, written $\mathit{fresh}\,(\vec{x}, \mathcal{K}, P)$, iff

1. Free logical variables of P do not clash with $\vec{x}$, *i.e.* $\mathit{fv}(P) \cap \vec{x} = \emptyset$; and

2. For all $\iota \in \mathsf{LEnv}$ and $\kappa \in \mathbb{K}$, there exists $\kappa' \in \mathbb{K}$ such that

$$(\mathbf{0}_{\mathbb{M}}, \kappa'), \iota \vDash_{\mathsf{SL}} \exists \vec{x}. \mathcal{K} \wedge \kappa \perp \kappa'$$

Lemma 4. The semantic implication (EXTEND) is valid.

Proof. It suffices to show that for all $\iota \in \mathsf{LEnv}$ and $w_1 \in \mathsf{World}$,

$$\begin{array}{ll} \text{if} & w_1, \iota \vDash P \wedge P * \mathcal{K}_2 \circledcirc I \wedge \mathit{fresh}\,(\vec{x}, \mathcal{K}_1 * \mathcal{K}_2, P) \\ \text{then} & \exists w_2.\, w_2, \iota \vDash \mathcal{K}_1 * \boxed{P * \mathcal{K}_2}_I \wedge (\downarrow(w_1))_{\mathsf{M}} = (\downarrow(w_2))_{\mathsf{M}} \wedge (w_1, w_2) \in G \end{array}$$

Pick an arbitrary $\iota \in \mathsf{LEnv}$ and $w_1 = (l, g, \mathcal{I}, \mathcal{G})$ such that

$$w_1, \iota \vDash P \tag{2.1}$$
$$P * \mathcal{K}_2 \circledcirc I \tag{2.2}$$
$$\mathit{fresh}\,(\vec{x}, \mathcal{K}_1 * \mathcal{K}_2, P) \tag{2.3}$$

Let $\kappa = f(\mathbf{0}_{\mathbb{K}}, \mathcal{I}, \mathbb{K})$ where the f function is defined recursively as follows.

$$f(\kappa, \mathcal{I}, \emptyset) \stackrel{\text{def}}{=} \kappa$$
$$f(\kappa_1, \mathcal{I}, K \uplus \{\kappa_2\}) \stackrel{\text{def}}{=} \begin{cases} f(\kappa_1 \bullet_{\mathbb{K}} \kappa_2, \mathcal{I}, K) & \kappa_1 \sharp \kappa_2 \wedge \exists \kappa \in \mathsf{dom}\,(\mathcal{I})\,.\, \kappa \not\perp \kappa_2 \\ f(\kappa_1, \mathcal{I}, K) & \text{otherwise} \end{cases}$$

From the definition of κ and function f we have:

$$\forall \kappa_0 \in \mathbb{K}.\ \kappa_0 \leq \kappa \ \ \vee \ \ \kappa_0 \not\sharp \kappa \ \ \vee \ \ \forall \kappa' \in \mathsf{dom}\,(\mathcal{I})\,.\, \kappa_0 \perp \kappa' \tag{2.4}$$

From (2.3) and the definition of *fresh* we know that

$$\exists \kappa'.\, (\mathbf{0}_{\mathbb{M}}, \kappa'), \iota \vDash_{\mathsf{SL}} \exists \vec{x}. \mathcal{K}_1 * \mathcal{K}_2 \ \wedge \ \kappa' \perp \kappa$$

and consequently there exists $\vec{v} \in \mathsf{Val}$ and $\kappa_1, \kappa_2 \in \mathbb{K}$ such that

$$\kappa_1 \in (\!|\mathcal{K}_1[\vec{v}/\vec{x}]|\!)^K_\iota \wedge \kappa_2 \in (\!|\mathcal{K}_2[\vec{v}/\vec{x}]|\!)^K_\iota \wedge \kappa_1 \bullet_{\mathbb{K}} \kappa_2 \perp \kappa \tag{2.5}$$

From (2.2) we have

$$P * \mathcal{K}_2[\vec{v}/\vec{x}] \circledcirc I[\vec{v}/\vec{x}] \tag{2.6}$$

From (2.1) and the definition of $\vDash_{\mathsf{SL}}$ we have $l, \iota \vDash_{\mathsf{SL}} P$. Similarly, from (2.5) and the definitions of $\vDash$ and $\vDash_{\mathsf{SL}}$ we have $(\mathbf{0}_{\mathbb{M}}, \kappa_2) \vDash_{\mathsf{SL}} \mathcal{K}_2$. Consequently we have:

$$l \circ (\mathbf{0}_{\mathbb{M}}, \kappa_2), \iota \vDash_{\mathsf{SL}} P * \mathcal{K}_2 \tag{2.7}$$

From the definitions of w_1 and the well-formedness of worlds we know there exists $\mathcal{F} \in \mathcal{P}(\mathsf{LState})$ such that

$$g \in \mathcal{F} \wedge \mathcal{F} \blacktriangleright \mathcal{I} \tag{2.8}$$

Pick $\mathcal{I}_0 \in \mathsf{AMod}$ such that

$$\mathsf{dom}(\mathcal{I}_0) = \{\kappa \mid (\kappa)^{\perp}_{\mathbb{K}} = (\kappa_1 \bullet_{\mathbb{K}} \kappa_2)^{\perp}_{\mathbb{K}}\} \wedge \forall \kappa \in \mathsf{dom}(\mathcal{I}_0) \,.\, \mathcal{I}_0(\kappa) = \emptyset$$

Let $\mathcal{I}'' \stackrel{\text{def}}{=} \langle\!| I[\vec{v}/\vec{x}] |\!\rangle_{\iota}$ and $\mathcal{I}' = \mathcal{I}'' \cup \mathcal{I}_0$; from (2.6), (2.7) and the definition of ⓒ we know there exists $\mathcal{F}' \in \mathcal{P}(\mathsf{LState})$ such that $l \circ (\mathbf{0}_{\mathbb{M}}, \kappa_2) \in \mathcal{F}' \wedge \mathcal{F}' \blacktriangleright \mathcal{I}''$. Consequently from the definitions of $\blacktriangleright$, $\mathcal{I}'$ and $\mathcal{I}_0$ and since $\forall \kappa \in \mathsf{dom}(\mathcal{I}_0) \,.\, \mathcal{I}_0(\kappa) = \emptyset$ we have

$$l \circ (\mathbf{0}_{\mathbb{M}}, \kappa_2) \in \mathcal{F}' \wedge \mathcal{F}' \blacktriangleright \mathcal{I}' \tag{2.9}$$

Let $\mathcal{G}' \stackrel{\text{def}}{=} (\mathcal{I}, \mathcal{F}) + (\mathcal{I}', \mathcal{F}')$ defined as follows.

$$((\mathcal{I}, \mathcal{F}) + (\mathcal{I}', \mathcal{F}'))(\kappa) \stackrel{\text{def}}{=} \left\{ (f, a[f]) \;\middle|\; \begin{array}{l} (a \in \mathcal{I}(\kappa) \vee a \in \mathcal{I}'(\kappa)) \wedge \\ f \in \mathcal{F} \circ \mathcal{F}' \wedge \mathit{enabled}(a, f) \end{array} \right\}$$

and

$$\mathcal{F} \circ \mathcal{F}' \stackrel{\text{def}}{=} \{ f \circ f' \mid f \in \mathcal{F} \wedge f' \in \mathcal{F}' \}$$

Let $g' = l \circ (\mathbf{0}_{\mathbb{M}}, \kappa_2)$ and $w_2 = ((\mathbf{0}_{\mathbb{M}}, \kappa_1), g \circ g', \mathcal{I} \cup \mathcal{I}', \mathcal{G}')$. Given the definitions of G and G^{e}, it then suffices to show

$$(\downarrow(w_1))_{\mathsf{M}} = (\downarrow(w_2))_{\mathsf{M}} \tag{2.10}$$

$$\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \prec \mathsf{dom}(\mathcal{I}') \tag{2.11}$$

$$\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \perp \mathsf{dom}(\mathcal{I}) \tag{2.12}$$

$$(\mathcal{I} \cup \mathcal{I}', \mathcal{G}') \downarrow (g', g, \mathcal{I}') \tag{2.13}$$

$$\forall s', \mathcal{I}_0 .\, (\mathcal{I}, \mathcal{G}) \downarrow (s', g - s', \mathcal{I}_0) \implies (\mathcal{I} \cup \mathcal{I}', \mathcal{G}') \downarrow (s', (g - s') \circ g', \mathcal{I}_0) \tag{2.14}$$

$$w_2, \iota \vDash \exists \vec{x} .\, \mathcal{K}_1 * \boxed{P * \mathcal{K}_2}_I \tag{2.15}$$

RTS. (2.10) This follows immediately from the definition of $\downarrow(.)$ and the definitions of w_1 and w_2.

RTS. (2.11) This follows trivially from the definitions of $\prec$ and $\mathcal{I}'$ since $\mathcal{I}_0 \subseteq \mathcal{I}'$ and $\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \in \mathsf{dom}(\mathcal{I}_0)$.

RTS. (2.12) There are two cases to consider. Either $\kappa_1 \bullet_{\mathbb{K}} \kappa_2 = \mathbf{0}_{\mathbb{K}}$; or $\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \neq \mathbf{0}_{\mathbb{K}}$. If the former is the case then the desired result holds trivially.

On the other hand, in the latter case where $\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \neq \mathbf{0}_{\mathbb{K}}$, since $\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \perp \kappa$ (2.5), we know

$$\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \sharp \kappa \wedge \kappa_1 \bullet_{\mathbb{K}} \kappa_2 \not\leq \kappa$$

and consequently from (2.4) we have

$$\forall \kappa' \in \mathsf{dom}\,(\mathcal{I})\,.\,\kappa_1 \bullet_{\mathbb{K}} \kappa_2 \perp \kappa'$$

as required.

RTS. (2.13) From (2.8), (2.9), the definitions of $\mathcal{I}'$, $\mathcal{G}'$, g' and Lemma 15, we can despatch this obligation.

RTS. (2.14) From (2.8), (2.9), the definitions of $\mathcal{I}'$, $\mathcal{G}'$, g' and Lemma 16, we can despatch this obligation.

RTS. (2.15)
From (2.5) and the definition of $\vDash_{\mathsf{SL}}$ we have $(\mathbf{0}_{\mathbb{M}}, \kappa_1), \iota \vDash_{\mathsf{SL}} \mathcal{K}_1[\vec{v}/\vec{x}]$ and consequently by the definition of $\vDash$

$$((\mathbf{0}_{\mathbb{M}}, \kappa_1), g \circ g', \mathcal{I} \cup \mathcal{I}', \mathcal{G}'), \iota \vDash \mathcal{K}_1[\vec{v}/\vec{x}] \tag{2.16}$$

Similarly, from (2.5) and the definitions of $\vDash_{\mathsf{SL}}$ and $\vDash$ we have $((\mathbf{0}_{\mathbb{M}}, \kappa_2), g, \mathcal{I}, \mathcal{G}), \iota \vDash \mathcal{K}_2[\vec{v}/\vec{x}]$; consequently from (2.1), the definition of g' and since from (2.3) and the definition of *fresh* we have $fv(P) \cap \vec{x} = \emptyset$, we can conclude

$$(g', g, \mathcal{I}, \mathcal{G}), \iota \vDash (P * \mathcal{K}_2)[\vec{v}/\vec{x}] \tag{2.17}$$

Thus from (2.17), (2.14) - established above - and Lemma 5 (below) we have:

$$g', \iota \vDash_{g \circ g', \mathcal{I} \cup \mathcal{I}', \mathcal{G}'} (P * \mathcal{K}_2)[\vec{v}/\vec{x}] \tag{2.18}$$

On the other hand, from (2.8), (2.9), the definitions of $\mathcal{I}'$, $\mathcal{G}'$, g', Lemma 15 and since $\mathcal{I}' = \mathcal{I}'' \cup \mathcal{I}_0 = \{\!|I[\vec{v}/\vec{x}]|\!\}_\iota \cup \mathcal{I}_0$, we have:

$$\big(\mathcal{I} \cup \mathcal{I}', \mathcal{G}'\big) \downarrow \big(g', g, \{\!|I[\vec{v}/\vec{x}]|\!\}_\iota\big) \tag{2.19}$$

Consequently from (2.18), (2.19) and the definition of $\vDash$

$$(\mathbf{0}_{\mathsf{L}}, g \circ g', \mathcal{I} \cup \mathcal{I}', \mathcal{G}'), \iota \vDash (\boxed{P * \mathcal{K}_2}_I)[\vec{v}/\vec{x}] \tag{2.20}$$

From (2.16), (2.20) and the definitions of w_2 and $\vDash$ we have:

$$w_2, \iota \vDash (\mathcal{K}_1 * \boxed{P * \mathcal{K}_2}_I)[\vec{v}/\vec{x}]$$

and consequently from the definition of $\vDash$

$$w_2, \iota \vDash \exists \vec{x}.\, \mathcal{K}_1 * \boxed{P * \mathcal{K}_2}_I$$

as required.

Lemma 5. for all $P \in \mathsf{Assn}$, $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{I}', \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}$:

$$(s, g, \mathcal{I}, \mathcal{G}), \iota \vDash P \wedge \mathcal{G}' \uparrow^{(s,\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \implies s, \iota \vDash_{g \circ s, \mathcal{I} \cup \mathcal{I}', \mathcal{G}'} P$$

Proof. By induction on the structure of assertion P.

Case $P \stackrel{\text{def}}{=} A$ Immediate.
Case $P \stackrel{\text{def}}{=} P_1 \implies P_2$
Pick an arbitrary $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{I}', \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}$ such that

$$(s, g, \mathcal{I}, \mathcal{G}), \iota \vDash P_1 \implies P_2 \tag{2.21}$$
$$\mathcal{G}' \uparrow^{(s,\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \tag{2.22}$$
$$\forall s, g \in \mathsf{LState}. \forall \mathcal{I}, \mathcal{I}', \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}.$$
$$(s, g, \mathcal{I}, \mathcal{G}), \iota \vDash P_1 \wedge \mathcal{G}' \uparrow^{(s,\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \implies s, \iota \vDash_{g \circ s, \mathcal{I} \cup \mathcal{I}', \mathcal{G}'} P_1 \quad \text{(I.H1)}$$
$$\forall s, g \in \mathsf{LState}. \forall \mathcal{I}, \mathcal{I}', \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}.$$
$$(s, g, \mathcal{I}, \mathcal{G}), \iota \vDash P_2 \wedge \mathcal{G}' \uparrow^{(s,\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \implies s, \iota \vDash_{g \circ s, \mathcal{I} \cup \mathcal{I}', \mathcal{G}'} P_2 \quad \text{(I.H2)}$$

From (2.21) and the definition of $\vDash$ we know $(s, g, \mathcal{I}, \mathcal{G}), \iota \vDash P_1$ implies $(s, g, \mathcal{I}, \mathcal{G}), \iota \vDash P_2$; consequently, from (2.22), (I.H1) and (I.H2) we have: $s, \iota \vDash_{g \circ s, \mathcal{I} \cup \mathcal{I}', \mathcal{G}'} P_1$ implies $s, \iota \vDash_{g \circ s, \mathcal{I} \cup \mathcal{I}', \mathcal{G}'} P_2$. Thus, from the definition of $\vDash_{g \circ s, \mathcal{I} \cup \mathcal{I}', \mathcal{G}'}$ we have:

$$s, \iota \vDash_{g \circ s, \mathcal{I} \cup \mathcal{I}', \mathcal{G}'} P_1 \implies P_2$$

as required.

Cases $P \stackrel{\text{def}}{=} \exists x. P'$**;** $P \stackrel{\text{def}}{=} P_1 \ast P_2$**;** $P \stackrel{\text{def}}{=} P_1 \mathbin{\text{-}\!\!\circledast} P_2$
These cases are analogous to the previous case and are omitted here.

Case $P \stackrel{\text{def}}{=} \boxed{P'}_I$
Pick an arbitrary $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{I}', \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}$ such that

$$(s, g, \mathcal{I}, \mathcal{G}), \iota \vDash \boxed{P'}_I \tag{2.23}$$
$$\mathcal{G}' \uparrow^{(s,\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \tag{2.24}$$

From (2.23) and the definition of $\vDash$ we have:

$$s = \mathbf{0}_{\mathsf{L}} \wedge \exists s', r'.\, g = s' \circ r' \wedge s', \iota \vDash_{g,\mathcal{I},\mathcal{G}} P' \wedge (\mathcal{I}, \mathcal{G})\downarrow \left(s', r', \langle\!|I|\!\rangle_{\iota}\right)$$

Thus from (2.24) we have

$$s = \mathbf{0}_{\mathsf{L}} \wedge \exists s', r'.\, g = s' \circ r' \wedge s', \iota \vDash_{g,\mathcal{I},\mathcal{G}} P' \wedge \left(\mathcal{I} \cup \mathcal{I}', \mathcal{G}'\right)\downarrow \left(s', r' \circ s, \langle\!|I|\!\rangle_{\iota}\right)$$

and consequently from (2.24), Lemma 6 and since $s = \mathbf{0}_{\mathsf{L}}$

$$\begin{aligned} &s = \mathbf{0}_{\mathsf{L}} \wedge \exists s', r'.\, g \circ s = s' \circ r' \wedge s', \iota \vDash_{g\circ s,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P' \\ &\wedge \left(\mathcal{I} \cup \mathcal{I}', \mathcal{G}'\right)\downarrow \left(s', r' \circ s, \langle\!|I|\!\rangle_{\iota}\right) \end{aligned}$$

That is,

$$s, \iota \vDash_{g\circ s,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} \boxed{P'}_{I}$$

as required. □

Lemma 6. for all $P \in \mathsf{Assn}$, $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{I}', \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}$:

$$s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P \wedge \mathcal{G}'\uparrow^{(\mathbf{0}_{\mathsf{L}},\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \implies s, \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P$$

Proof. By induction on the structure of assertion P.

Case $P \stackrel{\text{def}}{=} A$ Immediate.
Case $P \stackrel{\text{def}}{=} P_1 \implies P_2$
Pick an arbitrary $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{I}', \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}$ such that

$$s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P_1 \implies P_2 \tag{2.25}$$

$$\mathcal{G}'\uparrow^{(\mathbf{0}_{\mathsf{L}},\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \tag{2.26}$$

$$\forall s, g \in \mathsf{LState}.\, \forall \mathcal{I}', \mathcal{I}, \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}.$$
$$s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P_1 \wedge \mathcal{G}'\uparrow^{(\mathbf{0}_{\mathsf{L}},\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \implies s, \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P_1 \tag{I.H1}$$

$$\forall s, g \in \mathsf{LState}.\, \forall \mathcal{I}', \mathcal{I}, \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}.$$
$$s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P_2 \wedge \mathcal{G}'\uparrow^{(\mathbf{0}_{\mathsf{L}},\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \implies s, \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P_2 \tag{I.H2}$$

From (2.25) and the definition of $\vDash_{g,\mathcal{I},\mathcal{G}}$ we know $s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P_1$ implies $P_2 \vDash_{g,\mathcal{I},\mathcal{G}}$; consequently, from (2.26), (I.H1) and (I.H2) we have:

$s, \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P_1$ implies $s, \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P_2$. Thus, from the definition of $\vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'}$ we have:

$$s, \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P_1 \implies P_2$$

as required.

Cases $P \stackrel{\text{def}}{=} \exists x.\, P'$**;** $P \stackrel{\text{def}}{=} P_1 * P_2$**;** $P \stackrel{\text{def}}{=} P_1 \mathbin{⍟} P_2$
These cases are analogous to the previous case and are omitted here.

Case $P \stackrel{\text{def}}{=} \boxed{P'}_I$
Pick an arbitrary $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{I}', \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}$ such that

$$s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} \boxed{P'}_I \tag{2.27}$$

$$\mathcal{G}' \uparrow^{(\mathbf{0}_\mathsf{L},\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \tag{2.28}$$

$$\forall s, g \in \mathsf{LState}.\, \forall \mathcal{I}', \mathcal{I}, \mathcal{G}, \mathcal{G}' \in \mathsf{AMod}.$$
$$s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P' \wedge \mathcal{G}' \uparrow^{(\mathbf{0}_\mathsf{L},\mathcal{I}')} (g, \mathcal{I}, \mathcal{G}) \implies s, \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P' \tag{I.H}$$

From (2.27) and the definition of $\vDash_{g,\mathcal{I},\mathcal{G}}$ we have:

$$s = \mathbf{0}_\mathsf{L} \wedge \exists s', r'.\, g = s' \circ r' \wedge s', \iota \vDash_{g,\mathcal{I},\mathcal{G}} P' \wedge (\mathcal{I}, \mathcal{G})\downarrow \left(s', r', \langle\!| I |\!\rangle_\iota\right)$$

Thus from (2.28) and (I.H) we have

$$s = \mathbf{0}_\mathsf{L} \wedge \exists s', r'.\, g = s' \circ r' \wedge s', \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} P' \wedge \left(\mathcal{I} \cup \mathcal{I}', \mathcal{G}'\right)\downarrow \left(s', r', \langle\!| I |\!\rangle_\iota\right)$$

That is,

$$s, \iota \vDash_{g,\mathcal{I}\cup\mathcal{I}',\mathcal{G}'} \boxed{P'}_I$$

as required. □

□

Action Shifting Notice that, according to our rely and guarantee conditions, the local action model may only grow with time. However, that is not to say that the same holds of interference relations in shared state assertions: as seen in §**??**, they may forget about actions that are either redundant or not relevant to the current subjective view via *shifting*. As

for the action model closure relation (Def. 20), this needs to be the case not only from the current subjective view but also for any evolution of it according to potential actions, both from the thread and the environment. To capture this set of possible futures, we refine our notion of local fences (Def. 11), which was defined in the context of the global shared state, to the case where we consider only a subjective state within the global shared state. For this, we need to also refine a second time our notion of action application to ignore the context of a subjective state, which as far as a subjective view is concerned could be anything.

Definition 29 (Subjective action application)**.** The *subjective application* of an action a on a logical state s, written $a(s)$ is defined provided that there exists a context r for which $a[s \circ r]$ is defined. When that is the case, we write $a(s)$ for

$$\left\{ s' \;\middle|\; \exists r.\ s \sharp r \wedge (s', -) \in a[s, r] \right\}$$

Note that, in contrast with $a[l]$, only *parts* of the active precondition has to intersect with the subjective view s for $a(s)$ to apply. Thus, we fabricate a context r that is compatible with the subjective view and satisfies the rest of the precondition.

Definition 30 (Fenced action model)**.** An action model $\mathcal{I} \in \mathsf{AMod}$ is *fenced* by $\mathcal{F} \in \mathcal{P}(\mathsf{LState})$, written $\mathcal{F} \rhd \mathcal{I}$, if, for all $l \in \mathcal{F}$ and all $a \in rg(\mathcal{I})$,

$$a(l) \text{ is defined} \Rightarrow a(l) \subseteq \mathcal{F}$$

In contrast with local fences, fences do not require that actions be confined inside the subjective state. We lift the notion of fences to assertions as follows, given $F \in \mathsf{Assn}$ and $I \in \mathsf{IAssn}$:

$$F \rhd I \iff^{\text{def}} \forall \iota, \kappa.\, \{ l \mid l, \iota \vDash_{\mathsf{SL}} F \} \rhd \langle\!| I |\!\rangle_{\iota}(\kappa)$$

For instance, a possible fence for the interference assertion I_{y} of Fig. **??** is denoted by the following assertion.

$$F_{\mathrm{y}} \stackrel{\text{def}}{=} \bigvee_{v=0}^{10} (x \mapsto v * y \mapsto v) \vee (x \mapsto v+1 * y \mapsto v)$$

Definition 31 (Action shifting)**.** Given $\mathcal{I}, \mathcal{I}' \in \mathsf{AMod}$ and $\mathcal{P} \in \mathcal{P}(\mathsf{LState})$, $\mathcal{I}'$ is a *shifting* of $\mathcal{I}$ with respect to $\mathcal{P}$, written $\mathcal{I} \sqsubseteq^{\mathcal{P}} \mathcal{I}'$, if there exists a fence $\mathcal{F}$ such that

$$\begin{array}{l} \mathcal{P} \subseteq \mathcal{F} \wedge \mathcal{F} \rhd \mathcal{I} \wedge \forall l \in \mathcal{F}.\, \forall \kappa. \\ \quad (\forall a \in \mathcal{I}'(\kappa).\, \mathit{reflected}(a, l, \mathcal{I}(\kappa))) \wedge \\ \quad \forall a \in \mathcal{I}(\kappa).\, a(l) \text{ is defined} \wedge \mathit{visible}(a, l) \Rightarrow \mathit{reflected}(a, l, \mathcal{I}'(\kappa)) \end{array}$$

The first conjunct expresses the fact that the new action model $\mathcal{I}'$ cannot introduce new actions not present in $\mathcal{I}$, and the second one that $\mathcal{I}'$ has to include all the visible potential actions of $\mathcal{I}$. We lift $\sqsubseteq$ to assertions as follows:

$$I \sqsubseteq^P I' \stackrel{\text{def}}{=} \forall \iota, \kappa.\ \langle\!|I|\!\rangle_\iota \sqsubseteq^{\{s \mid s, \iota \vDash_{\mathsf{SL}} P\}} \langle\!|I'|\!\rangle_\iota$$

Lemma 7. The semantic implications SHIFT is valid.

Proof. It suffices to show that for all $\iota \in \mathsf{LEnv}$ and $w_1 \in \mathsf{World}$,

$$\begin{array}{ll} \text{if} & w_1, \iota \vDash \boxed{P}_I \wedge \langle\!|I|\!\rangle_\iota \sqsubseteq^{\{s \mid s, \iota \vDash_{\mathsf{SL}} P\}} \langle\!|I'|\!\rangle_\iota \\ \text{then} & \exists w_2.\ w_2, \iota \vDash \boxed{P}_{I'} \wedge (\downarrow(w_1))_{\mathsf{M}} = (\downarrow(w_2))_{\mathsf{M}} \wedge (w_1, w_2) \in G \end{array}$$

Pick an arbitrary $w_1 = (l, g, \mathcal{I}, \mathcal{G})$ such that

$$w_1, \iota \vDash \boxed{P}_I \tag{2.29}$$

$$\langle\!|I|\!\rangle_\iota \sqsubseteq^{\{s \mid s, \iota \vDash_{\mathsf{SL}} P\}} \langle\!|I'|\!\rangle_\iota \tag{2.30}$$

and let $w_2 = (l, g, \mathcal{I} \cup \langle\!|I'|\!\rangle_\iota, \mathcal{G})$. Given the definition of G, we are then required to show

$$(\downarrow(w_1))_{\mathsf{M}} = (\downarrow(w_2))_{\mathsf{M}} \tag{2.31}$$

$$\forall s', \mathcal{I}_0.\ (\mathcal{I}, \mathcal{G})\downarrow(s', g - s', \mathcal{I}_0) \implies (\mathcal{I} \cup \langle\!|I'|\!\rangle_\iota, \mathcal{G})\downarrow(s', g - s', \mathcal{I}_0) \tag{2.32}$$

$$w_2, \iota \vDash \boxed{P}_{I'} \tag{2.33}$$

From (2.29) and definition of w_1 we know that there exist $s, r \in \mathsf{LState}$ such that

$$g = s \circ r \tag{2.34}$$

$$s, \iota \vDash_{g, \mathcal{I}, \mathcal{G}} P \tag{2.35}$$

$$(\mathcal{I}, \mathcal{G})\downarrow(s, r, \langle\!|I|\!\rangle_\iota) \tag{2.36}$$

From the following lemma and (2.35) we know that

$$s, \iota \vDash_{\mathsf{SL}} P \tag{2.37}$$

Lemma 8. for all $P \in \mathsf{Assn}$, $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{G} \in \mathsf{AMod}$:

$$s, \iota \vDash_{g, \mathcal{I}, \mathcal{G}} P \implies s, \iota \vDash_{\mathsf{SL}} P$$

Proof. By induction on the structure of assertion P. Pick an arbitrary $\iota \in \mathsf{LEnv}$, then

Case $P \stackrel{\text{def}}{=} A$ Immediate.
Case $P \stackrel{\text{def}}{=} P_1 \implies P_2$
Pick an arbitrary $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{G} \in \mathsf{AMod}$ such that

$$s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P_1 \implies P_2 \tag{2.38}$$
$$\forall \iota, s, g, \mathcal{I}, \mathcal{G}.\, s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P_1 \implies s, \iota \vDash_{\mathsf{SL}} P_1 \tag{I.H1}$$
$$\forall \iota, s, g, \mathcal{I}, \mathcal{G}.\, s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P_2 \implies s, \iota \vDash_{\mathsf{SL}} P_2 \tag{I.H2}$$

From (2.38) and the definition of $\vDash_{g,\mathcal{I},\mathcal{G}}$ we know $s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} P_1$ implies $P_2 \vDash_{g,\mathcal{I},\mathcal{G}}$; consequently, from (I.H1) and (I.H2) we have: $s, \iota \vDash_{\mathsf{SL}} P_1$ implies $s, \iota \vDash_{\mathsf{SL}} P_2$. Thus, from the definition of $\vDash_{\mathsf{SL}}$ we have:

$$s, \iota \vDash_{\mathsf{SL}} P_1 \implies P_2$$

as required.

Cases $P \stackrel{\text{def}}{=} \exists x.\, P'$**;** $P \stackrel{\text{def}}{=} P_1 \ast P_2$**;** $P \stackrel{\text{def}}{=} P_1 \mathbin{\text{-}\!\!\circledast} P_2$
These cases are analogous to the previous case and are omitted here.

Case $P \stackrel{\text{def}}{=} \boxed{P'}_I$
Pick an arbitrary $\iota \in \mathsf{LEnv}$, $s, g \in \mathsf{LState}$ and $\mathcal{I}, \mathcal{G} \in \mathsf{AMod}$ such that

$$s, \iota \vDash_{g,\mathcal{I},\mathcal{G}} \boxed{P'}_I \tag{2.39}$$

From (2.39) and the definition of $\vDash_{g,\mathcal{I},\mathcal{G}}$ we know $s, g, \mathcal{I}, \mathcal{G} \vDash \boxed{P'}_I$ and hence, from the definition of $\vDash$, we have $s = \mathbf{0}_{\mathsf{L}}$. Consequently, from the definition of $\vDash_{\mathsf{SL}}$ we have:

$$s, \iota \vDash_{\mathsf{SL}} \boxed{P'}_I$$

as required. □

Consequently, from the definition of $\sqsubseteq$, (2.30) and (2.37) we have:

$$\langle\!| I |\!\rangle_\iota \sqsubseteq^{\{s\}} \langle\!| I' |\!\rangle_\iota \tag{2.40}$$

and thus from (2.36), (2.40) and Lemma 12 we have:

$$\left(\mathcal{J} \cup \left\langle\!\left| I' \right|\!\right\rangle_\iota, \mathcal{G}\right)\downarrow \left(s, r, \left\langle\!\left| I' \right|\!\right\rangle_\iota\right) \tag{2.41}$$

RTS. (2.31) This follows immediately from the definition of $\downarrow(.)$ and the definitions of w_1 and w_2.

RTS. (2.32) This follows immediately from the premise of the goal, (2.34), (2.36), (2.40) and Lemma 13.

RTS. (2.33) From 2.35, 2.32 (established above), and Lemma 6 above we have

$$s, \iota \vDash_{g, \mathcal{J} \cup \langle\!| I' |\!\rangle_\iota, \mathcal{G}} P \tag{2.42}$$

Consequently, from 2.34, 2.41, 2.42 and the definitions of $\vDash$ and w_2 we have

$$w_2, \iota \vDash \boxed{P}_{I'}$$

as required. □

2.2 Programming Language and Proof System

We define the CoLoSL proof system for deriving local Hoare triples for a simple concurrent imperative programming language. The proof system and programming language of CoLoSL are defined as an instantiation of the views framework [2].

Programming Language The CoLoSL programming language is that of the views programming language [2] instantiated with a set of *atomic* commands; It consists of skip, sequential composition, branching, loops and parallel composition. The set of CoLoSL atomic commands comprises an *atomic* construct $\langle . \rangle$ enforcing an atomic behaviour when instantiated with any *sequential* command. The sequential commands of CoLoSL are composed of a set of *elementary* commands, skip, sequential composition, branching and loops excluding atomic constructs and parallel composition. That is, atomic commands cannot be nested or contain parallel composition. CoLoSL is parametric in the choice of elementary commands allowing for suitable *instantiation* of CoLoSL depending on the programs being verified. For instance, in Example **??** the set of elementary commands comprises variable lookup and assignment. We proceed with the formalisation of CoLoSL programming language.

Parameter 5 (Elementary commands)**.** Assume a set of elementary commands Elem, ranged over by $\mathbb{E}, \mathbb{E}_1, \cdots, \mathbb{E}_n$.

Parameter 6 (Elementary command axioms)**.** Given the separation algebra of machine states $(\mathbb{M}, \bullet_{\mathbb{M}}, \mathbf{0}_{\mathbb{M}})$, assume a set of axioms associated with elementary commands:

$$\mathrm{Ax}_{\mathsf{E}} : \mathcal{P}(\mathbb{M}) \times \mathsf{Elem} \times \mathcal{P}(\mathbb{M})$$

Definition 32 (Sequential commands)**.** Given the set of elementary commands Elem, the set of sequential commands Seqs, ranged over by $\mathbb{S}, \mathbb{S}_1, \cdots, \mathbb{S}_n$ is defined inductively as:

$$\mathbb{S} ::= \mathbb{E} \mid \text{ skip } \mid \mathbb{S}_1; \mathbb{S}_2 \mid \mathbb{S}_1 + \mathbb{S}_2 \mid \mathbb{S}^\star$$

Definition 33 (Sequential command axioms)**.** Given the axiomatisation of elementary commands Ax_{E}, the *axioms of sequential commands*:

$$\mathrm{Ax}_{\mathsf{S}} : \mathcal{P}(\mathbb{M}) \times \mathsf{Seqs} \times \mathcal{P}(\mathbb{M})$$

is defined as follows where we write $M, M', M'', \cdots$ to quantify over the elements of $\mathcal{P}(\mathbb{M})$.

$$\begin{array}{rcl}
\mathrm{Ax}_{\mathsf{S}} & \stackrel{\text{def}}{=} & \mathrm{Ax}_{\mathsf{E}} \cup A_{\text{skip}} \cup A_{Seq} \cup A_{Choice} \cup A_{Rec} \\
A_{\text{skip}} & \stackrel{\text{def}}{=} & \{(M, \text{ skip}, M) \mid M \in \mathcal{P}(\mathbb{M})\} \\
A_{Seq} & \stackrel{\text{def}}{=} & \left\{ (M, \mathbb{S}_1; \mathbb{S}_2, M') \;\middle|\; \begin{array}{l} (M, \mathbb{S}_1, M'') \in \mathrm{Ax}_{\mathsf{S}} \wedge \\ (M'', \mathbb{S}_2, M') \in \mathrm{Ax}_{\mathsf{S}} \end{array} \right\} \\
A_{Choice} & \stackrel{\text{def}}{=} & \left\{ (M, \mathbb{S}_1 + \mathbb{S}_2, M') \;\middle|\; \begin{array}{l} (M, \mathbb{S}_1, M') \in \mathrm{Ax}_{\mathsf{S}} \wedge \\ (M, \mathbb{S}_2, M') \in \mathrm{Ax}_{\mathsf{S}} \end{array} \right\} \\
A_{Rec} & \stackrel{\text{def}}{=} & \left\{ (M, \mathbb{S}^\star, M) \;\middle|\; (M, \mathbb{S}, M) \in \mathrm{Ax}_{\mathsf{S}} \right\}
\end{array}$$

Definition 34 (Atomic commands)**.** Given the set of sequential commands Seqs, the set of*atomic commands* Atom, ranged over by $\mathbb{A}, \mathbb{A}_1, \cdots, \mathbb{A}_n$ is as:

$$\mathbb{A} ::= \langle \mathbb{S} \rangle$$

Definition 35 (Atomic command axioms)**.** Given the axioms of sequential commands Ax_{S}, the *axioms of atomic commands*:

$$\mathrm{Ax}_{\mathsf{A}} : \mathcal{P}(\mathsf{World}) \times \mathsf{Atom} \times \mathcal{P}(\mathsf{World})$$

is defined as follows where we write $W, W', \cdots$ to quantify over the elements of $\mathcal{P}(\mathsf{World})$.

$$\mathrm{Ax}_{\mathsf{A}} \stackrel{\text{def}}{=} \left\{ (W, \langle \mathbb{S} \rangle, W') \;\middle|\; \begin{array}{l} (M_1, \mathbb{S}, M_2) \in \mathrm{Ax}_{\mathsf{S}} \wedge \\ W \Rrightarrow^{\{M_1\}\{M_2\}} W' \end{array} \right\}$$

Definition 36 (Programming language)**.** Given the set of atomic commands Atom, the set of CoLoSL *commands* Comm, ranged over by $\mathbb{C}, \mathbb{C}_1, \cdots \mathbb{C}_n$, is defined by the following grammar.

$$\mathbb{C}_{:} := \mathbb{A} \mid \texttt{skip} \mid \mathbb{C}_1; \mathbb{C}_2 \mid \mathbb{C}_1 + \mathbb{C}_2 \mid \mathbb{C}_1 \parallel \mathbb{C}_2 \mid \mathbb{C}^\star$$

Proof Rules Our proof rules are of the form $\vdash \{P\}\ \mathbb{C}\ \{Q\}$ and carry an implicit assumption that the pre- and post-conditions of their judgements are stable. Since we build CoLoSL as an instantiation of the views framework [2], our proof rules correspond to those of [2] with the atomic commands axiomatised as per Def. 35.

Definition 37 (Proof rules)**.** The *proof rules* of CoLoSL are as described below.

$$\frac{}{\vdash \{P\}\ \text{skip}\ \{P\}}\ \textsc{Skip} \qquad \frac{\forall \iota.\ (\llbracket P \rrbracket_\iota,\ \mathbb{A},\ \llbracket Q \rrbracket_\iota) \in \mathrm{Ax}_{\mathsf{A}}}{\vdash \{P\}\ \mathbb{A}\ \{Q\}}\ \textsc{Atom}$$

$$\frac{\vdash \{P\}\mathbb{C}_1\{R\} \quad \vdash \{R\}\mathbb{C}_2\{Q\}}{\vdash \{P\}\ \mathbb{C}_1; \mathbb{C}_2\ \{Q\}}\ \textsc{Seq} \qquad \frac{\vdash \{P_1\}\mathbb{C}_1\{Q_1\} \quad \vdash \{P_2\}\mathbb{C}_2\{Q_2\}}{\vdash \{P_1 \ast P_2\}\ \mathbb{C}_1 \parallel \mathbb{C}_2\ \{Q_1 \ast Q_2\}}\ \textsc{Par}$$

$$\frac{\vdash \{P\}\ \mathbb{C}\ \{Q\}}{\vdash \{P \ast R\}\, \mathbb{C}\, \{Q \ast R\}}\ \textsc{Frame} \qquad \frac{P \Rrightarrow P' \quad \vdash \{P'\}\, \mathbb{C}\, \{Q'\} \quad Q' \Rrightarrow Q}{\vdash \{P\}\ \mathbb{C}\ \{Q\}}\ \textsc{Conseq}$$

$$\frac{\vdash \{P\}\mathbb{C}_1\{Q\} \quad \vdash \{P\}\mathbb{C}_2\{Q\}}{\vdash \{P\}\ \ \mathbb{C}_1 + \mathbb{C}_2\ \ \{Q\}}\ \textsc{Choice} \qquad \frac{\vdash \{P\}\, \mathbb{C}\, \{P\}}{\vdash \{P\}\ \ \mathbb{C}^\star\ \ \{P\}}\ \textsc{Rec}$$

Most proof rules are standard from disjoint concurrent separation logic [7]. In the Conseq judgement, $\Rrightarrow$ denotes the repartitioning of the state as described in Def. 26.

2.3 Operational Semantics

We define the operational semantics of CoLoSL in terms of a set of *concrete* (low-level) states. Recall that the states of CoLoSL, namely worlds, are parametric in the separation algebra of machine states $(\mathbb{M}, \bullet_{\mathbb{M}}, \mathbf{0}_{\mathbb{M}})$. As such, we require that the choice of concrete states is also parametrised in CoLoSL. Since CoLoSL is an instantiation of the views framework, its operational semantics is as defined in [2] provided with the set of concrete states and the *semantics of atomic commands*. The semantics of atomic commands are defined in terms of the *interpretation of sequential and elementary commands*.

Finally, as CoLoSL can be instantiated with any set of elementary commands Elem, the interpretation of elementary commands is also a parameter in CoLoSL. We proceed with the formalisation of the ingredients necessary for defining the operational semantics of atomic commands.

Parameter 7 (Concrete states)**.** Assume a set of concrete states $\mathcal{S}$ ranged over by $\sigma, \sigma_1, \cdots, \sigma_n$.

Parameter 8 (Elementary command interpretation)**.** Given the set of concrete states $\mathcal{S}$, assume an *elementary interpretation function* associating each elementary command with a non-deterministic state transformer:

$$[\![.]\!]_{\mathsf{E}}(.) : \mathsf{Elem} \to \mathcal{S} \to \mathcal{P}(\mathcal{S})$$

We lift the interpretation function to a set of concrete states such that for $S \in \mathcal{P}(\mathcal{S})$:

$$[\![\mathbb{E}]\!]_{\mathsf{E}}(S) \stackrel{\text{def}}{=} \bigcup_{\sigma \in S} ([\![\mathbb{E}]\!]_{\mathsf{E}}(\sigma))$$

Definition 38 (Sequential command interpretation)**.** Given the interpretation function of elementary commands $[\![.]\!]_{\mathsf{E}}(.)$, the *interpretation function for sequential commands*:

$$[\![.]\!]_{\mathsf{S}}(.) : \mathsf{Seqs} \to \mathcal{S} \to \mathcal{P}(\mathcal{S})$$

is defined inductively over the structure of sequential commands as follows:

$$\begin{array}{rcl}
[\![\mathbb{E}]\!]_{\mathsf{S}}(\sigma) & \stackrel{\text{def}}{=} & [\![\mathbb{E}]\!]_{\mathsf{E}}(\sigma) \\
[\![\,\mathrm{skip}]\!]_{\mathsf{S}}(\sigma) & \stackrel{\text{def}}{=} & \{\sigma\} \\
[\![\mathbb{S}_1; \mathbb{S}_2]\!]_{\mathsf{S}}(\sigma) & \stackrel{\text{def}}{=} & \{\sigma_2 \mid S = [\![\mathbb{S}_1]\!]_{\mathsf{S}}(\sigma) \ \wedge \ \sigma_2 \in [\![\mathbb{S}_2]\!]_{\mathsf{S}}(S)\} \\
[\![\mathbb{S}_1 + \mathbb{S}_2]\!]_{\mathsf{S}}(\sigma) & \stackrel{\text{def}}{=} & [\![\mathbb{S}_1]\!]_{\mathsf{S}}(\sigma) \cup [\![\mathbb{S}_2]\!]_{\mathsf{S}}(\sigma) \\
[\![\mathbb{S}^\star]\!]_{\mathsf{S}}(\sigma) & \stackrel{\text{def}}{=} & [\![\,\mathrm{skip} + \mathbb{S}; \mathbb{S}^\star]\!]_{\mathsf{S}}(\sigma)
\end{array}$$

where we lift the interpretation function to a set of concrete states such that for $S \in \mathcal{P}(\mathcal{S})$:

$$[\![\mathbb{S}]\!]_{\mathsf{S}}(S) \stackrel{\text{def}}{=} \bigcup_{\sigma \in S} ([\![\mathbb{S}]\!]_{\mathsf{S}}(\sigma))$$

Definition 39 (Atomic Command Interpretation)**.** Given the sequential command interpretation function $[\![.]\!]_{\mathsf{S}}(.)$, the *interpretation function for atomic commands*:

$$[\![.]\!]_{\mathsf{A}}(.) : \mathsf{Atom} \to \mathcal{S} \to \mathcal{P}(\mathcal{S})$$

is defined as :

$$[\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}}(\sigma) \stackrel{\text{def}}{=} [\![\mathbb{S}]\!]_{\mathsf{S}}(\sigma)$$

We lift the interpretation function to a set of concrete states such that for $S \in \mathcal{P}(\mathcal{S})$:

$$[\![\mathbb{S}]\!]_{\mathsf{A}}(S) \stackrel{\text{def}}{=} \bigcup_{\sigma \in S} ([\![\mathbb{S}]\!]_{\mathsf{A}}(\sigma))$$

2.4 Soundness

In order to establish the soundness of CoLoSL program logic, we relate its proof judgements to its operational semantics. To this end, we relate the CoLoSL states, *i.e.* worlds, to the concrete states by means of a *reification function*. The reification of worlds is defined in terms of relating (reifying) machine states in $\mathbb{M}$ to concrete states in $\mathcal{S}$. As such, given that CoLoSL is parametric in the choice of underlying machine states, the *reification of machine states* is also a parameter in CoLoSL.

Since CoLoSL is an instantiation of the views framework [2], its soundness follows immediately from the soundness of views, provided that the atomic axioms are sound with respect to their operational semantics. Recall that the axioms of atomic commands are defined in terms of the axioms pertaining to elementary commands which are parametrised in CoLoSL. Thus, to ensure the soundness of atomic commands, we require that the elementary axioms are also sound with respect to their operational semantics. We proceed with

Parameter 9 (Machine state reification)**.** Given the separation algebra of machine states $(\mathbb{M}, \bullet_{\mathbb{M}}, \mathbf{0}_{\mathbb{M}})$, assume a *reification function* $\lfloor . \rfloor_{\mathbb{M}} : \mathbb{M} \to \mathcal{P}(\mathcal{S})$, relating machine states to sets of concrete states.

Parameter 10 (Elementary soundness)**.** Given the separation algebra of machine states $(\mathbb{M}, \bullet_{\mathbb{M}}, \mathbf{0}_{\mathbb{M}})$,and their reification function $\lfloor . \rfloor_{\mathbb{M}}$, assume for each elementary command $\mathbb{E} \in \mathsf{Elem}$, its axiom $(M_1, \mathbb{E}, M_2) \in \mathrm{Ax}_{\mathsf{E}}$ and any given machine state $m \in \mathbb{M}$ the following *soundness* property holds.

$$[\![\mathbb{E}]\!]_{\mathsf{E}}(\lfloor M_1 \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) \subseteq \lfloor M_2 \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}$$

Definition 40 (Reification)**.** The *reification of worlds*, $\lfloor . \rfloor_W : \mathsf{World} \to \mathcal{P}(\mathcal{S})$ is defined as:

$$\lfloor (l, g, \mathcal{G}, \mathcal{I}) \rfloor_W \stackrel{\text{def}}{=} \lfloor (l \circ g)_{\mathsf{M}} \rfloor_{\mathbb{M}}$$

Theorem 1 (Atomic command soundness)**.** For all $\mathbb{A} \in \mathsf{Atom}$, $(W_1, \mathbb{A}, W_2) \in \mathrm{Ax}_{\mathsf{A}}$ and $w \in \mathsf{World}$:

$$[\![\mathbb{A}]\!]_{\mathsf{A}} (\lfloor W_1 \bullet \{w\} \rfloor_W) \subseteq \lfloor W_2 \bullet_{\mathbb{M}} R(\{w\}) \rfloor_W$$

Proof. By induction over the structure of $\mathbb{A}$.

Case $\langle \mathbb{S} \rangle$
Pick an arbitrary $\mathbb{S} \in \mathsf{Seqs}$, $w \in \mathsf{World}$ and $W_1, W_2 \in \mathcal{P}(\mathsf{World})$ such that $(W_1, \langle \mathbb{S} \rangle, W_2) \in \mathrm{Ax}_{\mathsf{A}}$. From the definition of Ax_{A} we then know there exists $M_1, M_2 \in \mathcal{P}(\mathbb{M})$ such that:

$$(M_1, \mathbb{S}, M_2) \in \mathrm{Ax}_{\mathsf{S}} \wedge W_1 \Rrightarrow^{\{M_1\}\{M_2\}} W_2 \tag{2.43}$$

RTS.

$$[\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}} (\lfloor W_1 \bullet \{w\} \rfloor_W) \subseteq \lfloor W_2 \bullet R(\{w\}) \rfloor_W$$

Proof. Pick an arbitrary $w_1 = (l, g, \mathcal{I}, \mathcal{G}) \in W_1$; it then suffices to show that there exists $w_2 \in W_2$ and $w' \in R(w)$ such that

$$[\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}} (\lfloor w_1 \bullet w \rfloor_W) = \lfloor w_2 \bullet w' \rfloor_W \tag{2.44}$$

Given the $w_1 = (l_1, g_1, \mathcal{I}_1, \mathcal{G}_1)$, from the definitions of $[\![.]\!]_{\mathsf{A}}(.)$ and $\lfloor . \rfloor_W$, and the properties of $\bullet$ and $\bullet_{\mathbb{M}}$ we have:

$$\begin{aligned} [\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}} (\lfloor w_1 \bullet w \rfloor_W) &= [\![\mathbb{S}]\!]_{\mathsf{S}} (\lfloor w_1 \bullet w \rfloor_W) \\ &= [\![\mathbb{S}]\!]_{\mathsf{S}} \left(\lfloor (l_1 \circ g_1)_{\mathsf{M}} \bullet_{\mathbb{M}} (w_{\mathsf{L}})_{\mathsf{M}} \rfloor_{\mathbb{M}} \right) \end{aligned} \tag{2.45}$$

On the other hand, from (2.43) and the definition of $\Rrightarrow$ we know there exists $m_1 \in M_1$ and $m' \in \mathbb{M}$ such that

$$m_1 \circ m' = (l_1 \circ g_1)_{\mathsf{M}} \wedge \tag{2.46}$$

$$\begin{aligned} &\forall m_2 \in M_2. \exists w_2 = (l_2, g_2, \mathcal{I}_2, \mathcal{G}_2) \in W_2. \\ &\quad m_2 \circ m' = (l_2 \circ g_2)_{\mathsf{M}} \wedge (w_1, w_2) \in G \end{aligned} \tag{2.47}$$

Consequently from (2.45) and (2.46) we have:

$$[\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}} (\lfloor w_1 \bullet w \rfloor_W) = [\![\mathbb{S}]\!]_{\mathsf{S}} \left(\lfloor m_1 \bullet_{\mathbb{M}} m' \bullet_{\mathbb{M}} (w_{\mathsf{L}})_{\mathsf{M}} \rfloor_{\mathbb{M}} \right) \tag{2.48}$$

From (2.43) and Lemma 18 we can rewrite (2.48) as

$$[\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}} (\lfloor w_1 \bullet w \rfloor_W) \subseteq \lfloor M_2 \bullet_{\mathbb{M}} \{m' \bullet_{\mathbb{M}} (w_{\mathsf{L}})_{\mathsf{M}}\} \rfloor_{\mathbb{M}}$$

That is, there exits $m_2 \in M_2$ such that

$$[\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}} (\lfloor w_1 \bullet w \rfloor_W) = \left\lfloor m_2 \bullet_{\mathbb{M}} m' \bullet_{\mathbb{M}} (w_{\mathsf{L}})_{\mathsf{M}} \right\rfloor_{\mathbb{M}} \tag{2.49}$$

From (2.47) we know there exists $w_2 \in \mathsf{World}$ such that

$$w_2 = (l_2, g_2, \mathcal{I}_2, \mathcal{G}_2) \in W_2 \wedge m_2 \circ m' = (l_2 \circ g_2)_{\mathsf{M}} \tag{2.50}$$

$$(w_1, w_2) \in G \tag{2.51}$$

From the definition of $\lfloor . \rfloor_W$ and the properties of $\bullet_{\mathbb{M}}$ and $\bullet$ we can thus rewrite (2.49) as

$$[\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}} (\lfloor w_1 \bullet w \rfloor_W) = \lfloor (l_2 \circ w_{\mathsf{L}}, g_2, \mathcal{I}_2, \mathcal{G}_2) \rfloor_W \tag{2.52}$$

From (2.51) and Lemma 19 we know there exists $w' \in \mathsf{World}$ such that

$$w' = (w_{\mathsf{L}}, g_2, \mathcal{I}_2, \mathcal{G}_2) \wedge w' \in R(w) \tag{2.53}$$

Consequently, from (2.50), (2.52) and (2.53) we know there exists $w_2 \in W_2$ and $w' \in R(w)$ such that

$$[\![\langle \mathbb{S} \rangle]\!]_{\mathsf{A}} (\lfloor w_1 \bullet w \rfloor_W) = \left\lfloor w_2 \bullet w' \right\rfloor_W$$

as required.

□

3. Examples

3.1 Concurrent Spanning Tree

Programs manipulating arbitrary graphs present the following significant challenge for compositional verification: because of deep sharing between different components of a graph, changes to one subgraph may affect other subgraphs which may point into it. This makes it hard to reason about updates to each subgraph in isolation. In a concurrent setting, this difficulty compounds with the fact that threads working on different parts of the graph may affect each other in ways that are difficult to reason about locally to each subgraph. Central to those issues is the fact that different subgraphs present *unspecified sharing* and may overlap in arbitrary ways. We now demonstrate on a concurrent spanning tree example that CoLoSL may be just the tool you need in this case, as it naturally deals with arbitrary overlapping views of the shared state, and allows one to tailor interferences to a given subjective view.

Our example, presented in Fig. 3.4, operates on a *directed binary graph* (henceforth simply *graph*), *i.e.* a directed graph where each node has at most two successors, called its left and right children. The program computes a in-place spanning tree of the graph, *i.e.* a tree that covers all nodes of the graphs from a given root, concurrently as follows: each time a new node is encountered, two new threads are created that each prune the edges of the left and right children recursively. A mark bit is associated to each node to keep track of which ones have already been visited. Each thread returns whether the root of the subgraph it was operating on had already been visited; if so, the parent thread removes the link from its root node to the corresponding child. Intuitively, it is allowed to do so because, since the child was marked by another thread, the child has to be already reachable via some other path in the graph.

We will prove that, given a shared graph as input, the program always returns a tree, *i.e.* all sharing and cycles have been appropriately removed.

Pleasingly, the CoLoSL specification achieves maximal locality and allows us to reason only on the current subgraph manipulated by a thread, instead of the whole graph. Because of arbitrary sharing between the two, a global specification would be unpleasant indeed! However, we do not establish that the final tree indeed spans the original graph. The reason it does is subtle indeed, as are the invariants required, but in ways unrelated to our main issue, which is to provide tight specifications for each thread.

To reason about this program, following Hobor and Villard [6], we use two representations of graphs. The first is a mathematical representation $\gamma = (V, E)$ where V is a finite set of vertices and $E : V \to (V \uplus \{\text{ null}\}) \times (V \uplus \{\text{ null}\})$ is a function associating each vertex with at most two successors, where null denotes the absence of an edge from the node. We write $n \in \gamma$ for $n \in V$, $\gamma(n)$ for $E(n)$ and $|\gamma|$ for $|V|$.

Mathematical graphs are connected to a second, in-memory representation by an inductive predicate $\mathsf{graph}\,(x, \gamma)$, denoting a spatial (in-heap) graph rooted at address x corresponding to the mathematical graph γ. The predicate definition uses the overlapping conjunction to account for the sharing between the left and right children and for potential cycles in the graph, as shown in Fig. 3.2. The basic action we allow on spatial graphs is to *mark* a node, changing its mark field from 0 to 1 and claiming ownership of its left and right pointers in the process. Such an action is allowed by a *marking* capability of the form $\mathsf{a_m}\,(n, e)$ where n denotes the vertex (address) and e the edge via which vertex n is visited. For instance, the capabilities associated with marking of vertex z in Fig. 3.1 are $\mathsf{a_m}\,(z, y.r)$ and $\mathsf{a_m}\,(z, w.l)$. Note that the parameterisation of our actions are merely a notational convenience and can be substituted for their full definitions. Given a graph at root address x, in order to account for the ability to mark the root vertex x, we introduce a logical (virtual) root edge $\mathcal{R}$ into x as depicted in Fig. 3.1 together with its associated marking capability $\mathsf{a_m}\,(x, \mathcal{R})$. The shared state contains node x which can be either unmarked ($\mathsf{U}\,(x, l, r)$) or marked ($\mathsf{M}\,(x)$); as well as the left and right subgraphs captured recursively by $\mathsf{G}\,(l, \gamma)$ and $\mathsf{G}\,(r, \gamma)$.

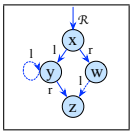

Figure 3.1: A graph.

Each vertex is represented as three consecutive cells in the heap tracking the mark bit and the addresses of the left (l) and right (r) subgraphs. For brevity, we write $x \mapsto m, l, r$ for $x \mapsto m * x + 1 \mapsto l * x + 2 \mapsto r$, and $x.m$, $x.l$, and $x.r$ for x, $x + 1$, and $x + 2$, respectively. When vertex x is in the unmarked state, the whole cell $x \mapsto 0, l, r$ and the capabilities to mark the children reside in the shared state. In the marked state, the shared state only

$$\mathsf{graph}\,(x,\gamma) \stackrel{\text{def}}{=} [\mathsf{a_m}\,(x,\mathcal{R})] * \boxed{\mathsf{G}\,(x,\gamma)}_{I_\gamma} \qquad I_\gamma \stackrel{\text{def}}{=} \bigcup_{n\in\gamma} I(n)$$
$$\mathsf{G}\,(x,\gamma) \stackrel{\text{def}}{=} (x = \text{ null} \wedge \mathsf{emp}) \vee x \in \gamma \wedge \exists l, r.\, \gamma(x) = (l,r) \wedge (\mathsf{U}\,(x,l,r) \vee \mathsf{M}\,(x)) \circledast \mathsf{G}\,(l,\gamma) \circledast \mathsf{G}\,(r,\gamma)$$
$$\mathsf{U}\,(x,l,r) \stackrel{\text{def}}{=} x \mapsto 0,l,r * [\mathsf{a_m}\,(l,x.l)] * [\mathsf{a_m}\,(r,x.r)]$$
$$\mathsf{M}\,(x) \stackrel{\text{def}}{=} x \mapsto 1$$
$$I(n) \stackrel{\text{def}}{=} \{\mathsf{a_m}\,(n,-) : \exists l, r.\, \mathsf{U}\,(n,l,r) \rightsquigarrow \mathsf{M}\,(n)\}$$

Figure 3.2: Global specification of the graph predicate.

$$\mathsf{g}\,(x,\gamma) \stackrel{\text{def}}{=} (x = \text{ null} \wedge \mathsf{emp}) \vee x \in \gamma \wedge \exists l, r.\, \gamma(x) = (l,r) \wedge \boxed{\mathsf{U}\,(x,l,r) \vee \mathsf{M}\,(x)}_{I(x)} * \mathsf{g}\,(l,\gamma) * \mathsf{g}\,(r,\gamma)$$
$$\mathsf{t}\,(x,\gamma) \stackrel{\text{def}}{=} (x = \text{ null} \wedge \mathsf{emp}) \vee x \in \gamma \wedge \exists l, r.\, \gamma(x) = (l,r) \wedge \boxed{\mathsf{M}\,(x)}_{I(x)} * \exists l' \in \{l, \text{ null}\}.\, \exists r' \in \{r, \text{ null}\}.\, [\mathsf{a_m}\,(l,x.l)] * x.l \mapsto l' * \mathsf{t}\,(l',\gamma) * [\mathsf{a_m}\,(r,x.r)] * x.r \mapsto r' * \mathsf{t}\,(r',\gamma)$$

Figure 3.3: Local specification of the graph predicate.

contains $x.m \mapsto 1$: the left and right subgraphs (pointers and capabilities) have been claimed by the thread who marked the node, while other threads need not access the children of x once they see that x is already marked. The atomic CAS instruction prevents several threads to concurrently mark the same node and claim ownership of the same resource.

The interference associated with the graph is described as the union of interferences pertaining to the vertices of the graph ($n \in \gamma$). For each vertex $n \in \gamma$, the only permitted action is that of marking n which can be carried out by any of the marking capabilities associated with node n ($\mathsf{a_m}\,(n,-)$). Note that the anonymous quantification $-$ is yet another notational shorthand and can be substituted for the following more verbose definition.

$$I(n) \stackrel{\text{def}}{=} \bigcup_{p\in\gamma} \left(\bigcup_{e\in\{p.l,p.r,\mathcal{R}\}} \mathsf{a_m}\,(n,e) : \exists l, r.\, \mathsf{U}\,(n,l,r) \rightsquigarrow \mathsf{M}\,(n) \right)$$

Fig. 3.4 shows an in place concurrent algorithm for calculating a spanning tree of a graph. The $\mathsf{graph}\,(x,\gamma)$ predicate defined in Fig. 3.2 is a *global* account of the graph in that it captures all vertices and the interference associated with them. However, our spanning tree algorithm operates *locally* as it is called upon recursively for each node. That is, for each span(n)

call (where $\mathrm{n} \mapsto n$ and $n \in \gamma$), the footprint of the call is limited to node n. Moreover, in order to reason about the concurrent recursive calls $\mathrm{span(x.l)} \parallel \mathrm{span(x.r)}$, we need to *split* the state into two $\star$-composed states prior to the calls, pass each constituent state onto the relevant thread and combine the resulting states by $\star$ composition through an application of the PAR rule. We thus provide a *local* specification of the graph, $\mathsf{g}(x,\gamma)$ as defined in Fig. 3.3 such that for all $n, p \in \gamma$ and $e \in \{p.l, p.r, \mathcal{R}\}$

$$\left\{ \mathrm{n} \mapsto n \star \mathrm{b} \mapsto - \star [\mathsf{a_m}(n,e)] \star \mathsf{g}(n,\gamma) \right\}$$
$$\mathrm{b{:=}\ span(n)}$$
$$\left\{ \mathrm{n} \mapsto n \star [\mathsf{a_m}(n,e)] \star (\mathrm{b} \mapsto 1 \star \mathsf{t}(n,\gamma) \vee \mathrm{b} \mapsto 0 \star \mathsf{t}(\mathrm{null},\gamma)) \right\}$$

The definition of the $\mathsf{g}(x,\gamma)$ predicate is similar to that of $\boxed{\mathsf{G}(x,\gamma)}_{I_\gamma}$ except that the global view $\boxed{\mathsf{G}(x,\gamma)}_{I_\gamma}$ that describes the resources associated with all $|\gamma|$ vertices has been replaced by $|\gamma|$ $\star$-composed more local views, each describing the resources of a vertex $n \in \gamma$. Moreover, the interference of each local view concerning a vertex $n \in \gamma$ has been shifted from I_γ to $I(n)$ as to reflect only those actions that affect n.

Similarly, the $\mathsf{t}(x,\gamma)$ predicate represents a *tree* rooted at x, as is standard in separation logic [8], and consists of $|\gamma|$ subjective views one for each vertex in γ. The assertion of each subjective view reflects that the corresponding vertex (x) has been marked $\boxed{\mathsf{M}(x)}_{I(x)}$. The resources associated with each node x, namely the left and right pointers and the corresponding marking capabilities have been claimed by the marking thread and moved into the local state. The vertex addressed by the left pointer of x (*i.e.* l') corresponds to either the initial value prior to marking (l where $\gamma(x) = (l,r)$) or null when l has more than one predecessors and has been marked by another thread, making the whole predicate stable against actions of the program and the environment.

We now demonstrate how to obtain the local specification $\mathsf{g}(x,\gamma)$ from the global specification of Fig. 3.2. When expanding the definition of $\mathsf{G}(x,\gamma)$, there are two cases to consider depending on whether or not $x = \mathrm{null}$. In what follows we only consider the case where $x \neq \mathrm{null}$ since the derivation in the case of $x = \mathrm{null}$ is trivial. Let P and Q predicates be defined as below.

$$P \stackrel{\text{def}}{=} \circledast_{n\in\gamma} (\gamma(n) = (l,r) \wedge (\mathsf{U}(n,l,r) \vee \mathsf{M}(n)))$$
$$Q \stackrel{\text{def}}{=} \circledast_{n\in\gamma} \left(\gamma(n) = (l,r) \wedge \boxed{\mathsf{U}(n,l,r) \vee \mathsf{M}(n)}_{I(n)}\right)$$

From the definitions of $\mathsf{G}(x,\gamma)$ and $\mathsf{g}(x,\gamma)$ we then have:

$$\mathsf{G}(x,\gamma) \iff P \qquad\qquad \mathsf{g}(x,\gamma) \iff Q$$

In order to derive the local specification $\mathsf{g}\,(x, \gamma)$ from the global specification $\mathsf{G}\,(x, \gamma)$, it thus suffices to show $\boxed{P}_{I_\gamma} \Rrightarrow Q$ as demonstrated below.

$$
\begin{aligned}
\boxed{P}_{I_\gamma} &\overset{(\textsc{Copy})}{\Longrightarrow} \underbrace{\boxed{P}_{I_\gamma} * \cdots * \boxed{P}_{I_\gamma}}_{|\gamma| \text{ times}} \\
&\overset{(\textsc{Forget})}{\Longrightarrow} \circledast_{n \in \gamma} \left(\gamma(n) = (l, r) \wedge \boxed{\mathsf{U}\,(n, l, r) \vee \mathsf{M}\,(n)}_{I_\gamma} \right) \\
&\overset{(\textsc{Shift})}{\Rrightarrow} \circledast_{n \in \gamma} \left(\gamma(n) = (l, r) \wedge \boxed{\mathsf{U}\,(n, l, r) \vee \mathsf{M}\,(n)}_{I(n)} \right) \\
&\overset{\text{def}}{\Longleftrightarrow} Q
\end{aligned}
$$

3.2 Set Module

In this section we consider a concurrent set module as described in [3]. We first produce the set specification in CoLoSL and then compare it against its specification in Concurrent Abstract Predicates (CAP) described in [3]. We demonstrate that our CoLoSL reasoning considerably improves on CAP by producing a *more concise* specification and allowing for *more local* reasoning. **TO DO** actual reasoning .

We implement a set as a sorted singly-linked list with no duplicate elements (since it represents a set) and one lock per node as described in [3]. Our set module provides three functionalities: contains(v), add(v) and remove(v). As the name suggests, the contains(v) function checks whether value v is contained in the set; the add(v) and remove(v) functions add/remove value v to/from the list, respectively.

Fig. 3.8-3.9 illustrate a C-like implementation of the set operations. All three operations proceed by traversing the sorted list from the head address and locating the first node in the list holding a value v' greater than or equal to v (through the locate procedure). The algorithm for locating such a node begins by locking the head node; it then moves down the list by hand-over-hand locking whereby first the node following the one currently held is locked and subsequently the previously locked node is released.

We proceed with the CoLoSL specification of the set module. Fig. 3.5 shows a specification of the set module in CoLoSL similar to that of Concurrent Abstract Predicates (CAP) defined in [3]. All three operations proceed by traversing the sorted list at head address h and locating the first node in the list holding a value v' greater than or equal to v. The algorithm

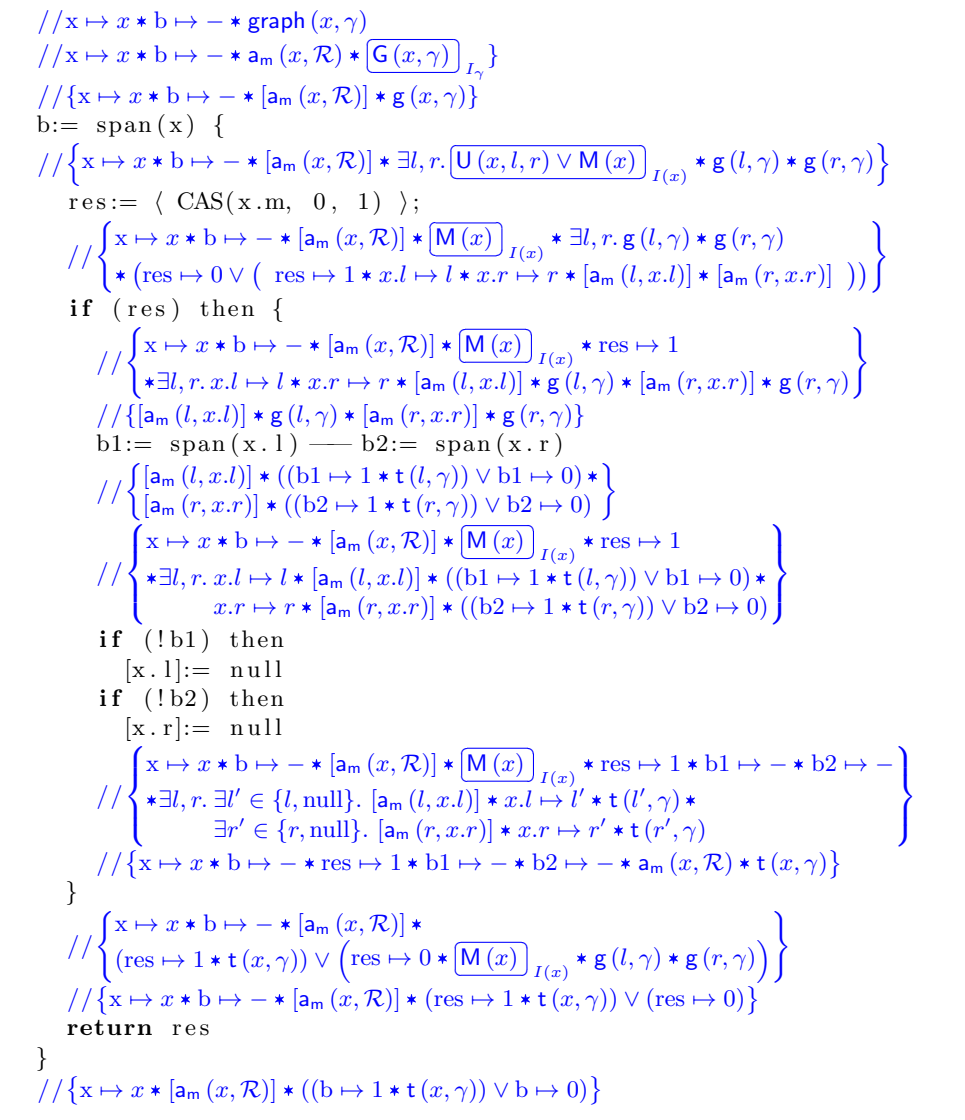

Figure 3.4: Concurrent Spanning Tree Implementation

for locating such a node begins by locking the head node of the list. It then moves down the list by hand-over-hand locking whereby first the node following the one currently held is locked and then the previously locked node is released. No thread can access a node locked by another thread or traverse past it. As such, no thread can overtake an other thread in accessing the list.

$$
\begin{array}{rl}
\mathsf{in}\,(h,v) & \stackrel{\text{def}}{=} \exists\pi.\,\mathsf{isLock}\,(h,\pi) * h.\mathsf{V} \Rrightarrow - * \mathsf{L}_{\in}\,(h,v) \\
\mathsf{out}\,(h,v) & \stackrel{\text{def}}{=} \exists\pi.\,\mathsf{isLock}\,(h,\pi) * h.\mathsf{V} \Rrightarrow - * \mathsf{L}_{\notin}\,(h,v)
\end{array}
$$

$$
\begin{array}{rl}
\mathsf{L}_{\in}\,(h,v) & \stackrel{\text{def}}{=} \exists L.\,\mathsf{sorted}\,(L) \wedge v \in L \wedge \mathsf{lsg}\,(h,\,\mathrm{null}, -\infty :: L + +\{\infty\}, h) \\
\mathsf{L}_{\notin}\,(h,v) & \stackrel{\text{def}}{=} \exists L.\,\mathsf{sorted}\,(L) \wedge v \notin L \wedge \mathsf{lsg}\,(h,\,\mathrm{null}, -\infty :: L + +\{\infty\}, h)
\end{array}
$$

$$
\begin{array}{rl}
\mathsf{lsg}\,(x,y,L,h) & \stackrel{\text{def}}{=} (L = [] \wedge x = y \wedge \mathsf{emp}) \vee \\
& \quad \exists z, v, L'.\, L = v :: L' \wedge \boxed{\mathsf{node}\,(x,v,z)}_{I(h,x)} * \mathsf{lsg}\,(z,y,L',h)
\end{array}
$$

$$
\begin{array}{rl}
\mathsf{node}\,(a,v,b) & \stackrel{\text{def}}{=} \mathsf{U}\,(a,v,b) \vee \mathsf{L}\,(a,v,b) \\
\mathsf{U}\,(a,v,b) & \stackrel{\text{def}}{=} \mathsf{link}\,(a,b) * \mathsf{val}\,(a,v,b) \\
\mathsf{L}\,(a,v,b) & \stackrel{\text{def}}{=} \mathsf{locked}\,(a) * \mathsf{val}\,(a,v,b)
\end{array}
$$

$$
\begin{array}{rl}
\mathsf{link}\,(a,b) & \stackrel{\text{def}}{=} a.next \mapsto b * \mathsf{isLock}\,(b,1) * a.\mathsf{M} \Rrightarrow - \\
\mathsf{val}\,(a,v,b) & \stackrel{\text{def}}{=} a.value \mapsto v * a.\mathsf{N} \Rrightarrow b \\
\mathsf{isLock}\,(a,\pi) & \stackrel{\text{def}}{=} a.\mathsf{L} \stackrel{\pi}{\Rrightarrow} - * \boxed{a.lock \mapsto 0 * a.\mathsf{U} \Rrightarrow - \vee a.lock \mapsto 1}_{L(a)} \\
\mathsf{locked}\,(a) & \stackrel{\text{def}}{=} a.\mathsf{U} \Rrightarrow - * \boxed{a.lock \mapsto 1}_{L(a)}
\end{array}
$$

Figure 3.5: CoLoSL specification of the concurrent set module where the definitions of $I(h,a)$ and $L(a)$ are provided in Fig. 3.7.

We provide a similar specification for the set module in CoLoSL as illustrated in Fig. 3.5. In what follows we first give a description of the predicates of Fig. 3.5 and subsequently contrast our specification with that of CAP. Since CoLoSL is parametric in the separation algebra of capabilities and capability assertions, we instantiate it with a *ghost fractional heap* to represent the separation algebra of capabilities and write heap-like assertions of the form $a \stackrel{\pi}{\Rrightarrow} b$ to denote capability a with permission π. Moreover, our capabilities are *stateful* in that they can capture some additional information (b). As we demonstrate below, this leads to conciser specifications. For readability we write $a \Rrightarrow b$ for $a \stackrel{1}{\Rrightarrow} b$, $a \stackrel{\pi}{\Rrightarrow} -$ for $\exists b.\, a \stackrel{\pi}{\Rrightarrow} b$ and $a \Rrightarrow -$ for $\exists b.\, a \stackrel{1}{\Rrightarrow} b$.

isLock(a,π) **/ locked**(a) Every node in the singly-linked list is protected by a lock that is to be acquired prior to its modification. The $\mathsf{isLock}\,(a,\pi)$ predicate is similar to that in [3] and states that the lock at address a can be acquired with permission $\pi \in (0,1]$. Multiple threads may attempt to acquire the lock at once and thus we use the π argument to reflect this sharing where

1 denotes exclusive right to acquisition while $\pi \in (0,1)$ accounts for sharing of the lock between multiple threads. The $\mathsf{isLock}\,(a,\pi)$ predicate asserts that the thread's local state contains the capability $a.\mathsf{L} \overset{\pi}{\Rightarrow} -$ to acquire the lock and that the lock resides in the shared state where at any one point either the lock is unlocked ($a.lock \mapsto 0$) and the shared state contains the capability to unlock it ($a.\mathsf{U} \Rightarrow -$); or it is locked ($a.lock \mapsto 1$) and the unlocking capability has been claimed by the locking thread.

The $\mathsf{locked}\,(a)$ predicate asserts that the thread's local state contains the exclusive capability to unlock the lock ($[a.\mathsf{U} \Rightarrow -]$); and that the a is in the locked state ($a.lock \mapsto 1$).

$\mathsf{in}(h,v)$ / $\mathsf{out}(h,v)$ The $\mathsf{in}\,(h,v)$ predicate states that the set at head address h contains value v and captures exclusive right to alter the set with respect to v by changing whether v belongs to it. It asserts that the thread owns some capability (π) to acquire the lock associated with head address ($\mathsf{isLock}\,(h,\pi)$) and that the thread owns the exclusive capability to alter the set with respect to value v ($h.\mathsf{V} \Rightarrow -$) should it own the capability to modify the address at which value v is stored (*cf.* the description of $\mathsf{U}\,(a,v,b)$ predicate). The underlying singly-inked list is captured by the $\mathsf{L}_{\in}\,(h,v)$ predicate. The $\mathsf{L}_{\in}\,(h,v)$ predicate uses an auxiliary carrier sorted list L (such that $v \in L$) to capture the contents of the singly-linked list via the lsg predicate. For simpler implementation, we extend L with two sentinel values $-\infty$ and ∞, one at each end to avoid corner cases such as removing the first/last element of the list. The $\mathsf{out}\,(h,v)$ predicate is analogous to $\mathsf{in}\,(h,v)$ and corresponds to the case where the set does not contain v.

$\mathsf{lsg}(x,y,L,h)$ This predicate is defined inductively and describes a *segment* of the list at h that starts at address x and extends upto (but not including) address y and contains the elements in the mathematical list L. When L is empty, it corresponds to a void segment and yields no resources (emp); otherwise, it is defined as the composition of the first node of the segment at address x, $\boxed{\mathsf{node}\,(x,v,z)}_{I(h,x)}$, and the tail of the list segment ($\mathsf{lsg}\,(z,y,L',h)$). The $\mathsf{node}\,(x,v,z)$ predicate describes a node at address x with value v and successor z and can be either unlocked ($\mathsf{U}\,(x,v,z)$) or locked ($\mathsf{L}\,(x,v,z)$).

$\mathsf{U}(a,v,b)$ / $\mathsf{L}(a,v,b)$ The $\mathsf{U}\,(a,v,b)$ predicate states that the node at address a is unlocked, it contains value v and comes before the node at address b. The statement of the $\mathsf{L}\,(a,v,b)$ predicate is analogous and describes the

node when locked. A thread can alter the set at h with respect to the node at address a with value v, only when it holds both the value capability $h.\mathsf{V} \Rrightarrow -$ and the modification capability $a.\mathsf{M} \Rrightarrow -$ in its local state. When the node is in the unlocked state, no thread can modify it and thus the modification capability $a.\mathsf{M} \Rrightarrow -$ as well as the next pointer of the node ($a.next \mapsto b$) lie in the shared state. Recall that when traversing the list by hand-over-hand locking, no thread can overtake an other in traversing the list. Consequently, a node can only be locked by a thread that has already acquired the lock associated with the previous node. Therefore, when the node a is in the unlocked state, the exclusive capability to lock its successor at address b ($\mathsf{isLock}\,(b, 1)$) also lies in the shared state. When a thread successfully locks the node at a, it claims the next pointer, the modification capability and the locking capability pertaining to its successors ($\mathsf{link}\,(a, b)$), and moves them into its local state. In return, it transfers the $\mathsf{locked}\,(a)$ resource to the shared state as evidence that it is indeed the locking thread.

In both locked and unlocked states, the shared state contains the value field ($a.value \mapsto v$) and the "successor" capability $a.\mathsf{N} \Rrightarrow b$. This capability is used to track the current successor of a even when the node is locked and its next pointer has been claimed by the locking thread.

CoLoSL vs. CAP Specification The definitions of most of the predicates in Fig. 3.5 are analogous to the correspondingly-named CAP predicates specified in [3]. The difference however lies in the definitions of the $\mathsf{L}_{\in}$, $\mathsf{L}_{\notin}$, and lsg predicates. Fig. 3.6 presents the definitions of these predicates in CAP as specified in [3].

There are two main advantages to the CoLoSL specification of the set module. In the CAP specification, all elements of the set (represented as a singly-linked list) reside in a *single* shared region labelled s. Each node at a given address a, is associated with an (*infinite*) set of update capabilities of the form $[\mathrm{U}(a, b, v)]$ for *all* possible addresses b and *all* possible values v. This is to capture all potential successor addresses b and all potential values v that may be stored at address a. For instance, since the value v' may be contained in the list at address x before address y, there must exist an update capability $[\mathrm{U}(x, y, v')]$ to accommodate possible modifications of value v' with respect to x and y. In order to modify a node, a thread can acquire the lock associated with the node and subsequently claim the relevant update capability. Since the capabilities associated with a region are *all* allocated at the point of region creation, the shared region is required to keep track of *all* (infinitely many) possible update capabilities $\mathrm{LGAP}(a, b, v)$ associated with

$$\mathsf{L}_{\lhd}(h,v,s) \stackrel{\text{def}}{=} \exists L, S.\, \mathsf{sorted}(L) \wedge v \lhd L \wedge \mathsf{lsg}(h,\ \text{null}, S, -\infty :: L + +\{\infty\}) \\ * (\mathsf{gaps}(S,s) \wedge \mathsf{mygaps}(v,s)) \qquad \text{where } \lhd = \in \text{ or } \lhd = \notin$$

$$\mathsf{lsg}(x,y,S,L) \stackrel{\text{def}}{=} (L = [] \wedge S = \emptyset \wedge x = y \wedge \mathsf{emp}) \vee \\ (\exists z, v, L'.\, L = v :: L' \wedge \mathsf{U}(x,v,z) * \mathsf{lsg}(z,y,S,L')) \vee \\ \begin{pmatrix} \exists z, v, L', S'.\, L = v :: L' \wedge S = \{(x,z)\} \uplus S' \wedge \\ \mathsf{L}(x,v,z) * \mathsf{lsg}(z,y,S',L') \end{pmatrix}$$

$$\mathsf{gaps}(S,s) \stackrel{\text{def}}{=} \circledast (x,y) \notin S. \circledast v.\, [\text{LGAP}(x,y,v)]_1^s * \\ \circledast (x,y) \in S. \exists w.\, \circledast v \neq w.\, [\text{LGAP}(x,y,v)]_1^s \wedge \\ \forall x,y,w,z.\, (x,y) \in S \wedge (w,z) \in S \implies (x = w \iff y = z)$$

$$\mathsf{mygaps}(v,s) \stackrel{\text{def}}{=} \circledast x, y.\, [\text{LGAP}(x,y,v)]_1^s * true$$

Figure 3.6: CAP predicates of the set module that contrast with the CoLoSL specification of Fig. 3.5.

all addresses a, all possible successor addresses b and all values v. This is captured by the $\mathsf{gaps}(S,s)$ and $\mathsf{mygaps}(S,s)$ predicates through the infinite multiplicative star operator $\circledast$. The $\mathsf{gaps}(S,s)$ predicate uses an auxiliary mathematical set S to track those nodes of the list that are currently locked and thus infer which LGAP capabilities reside in the shared state and which ones have been removed. This results in an inevitably cluttered and a rather counter-intuitive specification since it seems unnatural to account for the capabilities pertaining to addresses not in the domain of the set.

On the other hand, in the CoLoSL specification we use a combination of the $a.\mathsf{M} \Rrightarrow -$ and $a.\mathsf{N} \Rrightarrow b$ capabilities to achieve the same effect for each address a. Since CoLoSL allows for dynamic extension of the shared state, the capabilities associated with each address a are generated upon *extension* of the shared state with those addresses, thus avoiding the need to account for capabilities concerning those addresses absent from the list.

Moreover, since the ghost heaps used to represent the separation algebra of capabilities are *stateful*, rather than having a distinct capability to modify the element at address a before address b for each possible successor address b, we appeal to a single capability of the form $a.\mathsf{N} \Rrightarrow b$ whereby the capability is modified accordingly to reflect the changes to the successor address. For instance, when node at address a is redirected to point from address b to a new location at address c, $a.\mathsf{N} \Rrightarrow b$ is also updated to $a.\mathsf{N} \Rrightarrow c$.

$$I(h,a) \stackrel{\text{def}}{=} \left\{ \begin{array}{c} true : \{\mathsf{link}\,(a,b) \rightsquigarrow \mathsf{locked}\,(a.lock)\} \\ a.\mathsf{M} \Rrightarrow - : \{\exists v, b.\, \mathsf{L}\,(a,v,b) \rightsquigarrow \mathsf{U}\,(a,v,b)\} \\ a.\mathsf{M} \Rrightarrow - * h.\mathsf{V} \Rrightarrow - : \\ \quad \left\{ \begin{array}{c} \exists v_0, b, c.\ \ \mathsf{L}\,(a,v_0,b) * \mathsf{L}\,(b,v,c) \\ \rightsquigarrow \mathsf{L}\,(a,v_0,c) * \mathsf{L}\,(b,v,c) \end{array} \right\} \cup \\ \quad \left\{ \begin{array}{c} \exists v_0, b, c.\ \ v_0 < v \wedge \mathsf{L}\,(b,v_0,c) * \mathsf{L}\,(a,v,c) \\ \rightsquigarrow \mathsf{L}\,(b,v_0,c) \end{array} \right\} \\ \quad \left\{ \begin{array}{c} \exists v_1, v_2, c, d.\ \ v_1 < v < v_2 \wedge \mathsf{L}\,(a,v_1,c) * \mathsf{val}\,(b,v,c) * \mathsf{val}\,(c,v_2,d) \\ \rightsquigarrow \mathsf{L}\,(a,v_1,b) * \mathsf{val}\,(b,v,c) * \mathsf{val}\,(c,v_2,d) \end{array} \right\} \end{array} \right\}$$

$$L(a) \stackrel{\text{def}}{=} \left\{ \begin{array}{c} a.\mathsf{L} \stackrel{-}{\Rrightarrow} - : \left\{ a \mapsto 0 * a.\mathsf{U} \Rrightarrow - \rightsquigarrow a \mapsto 1 \right\} \\ a.\mathsf{U} \stackrel{1}{\Rrightarrow} - : \left\{ a \mapsto 1 \rightsquigarrow a \mapsto 0 * a.\mathsf{U} \Rrightarrow - \right\} \\ a.\mathsf{L} \stackrel{1}{\Rrightarrow} - * a.\mathsf{U} \stackrel{1}{\Rrightarrow} - : \left\{ a \mapsto 1 \rightsquigarrow a.\mathsf{L} \stackrel{1}{\Rrightarrow} - * a.\mathsf{U} \stackrel{1}{\Rrightarrow} - \right\} \end{array} \right\}$$

Figure 3.7: Interference associated with nodes in the concurrent set.

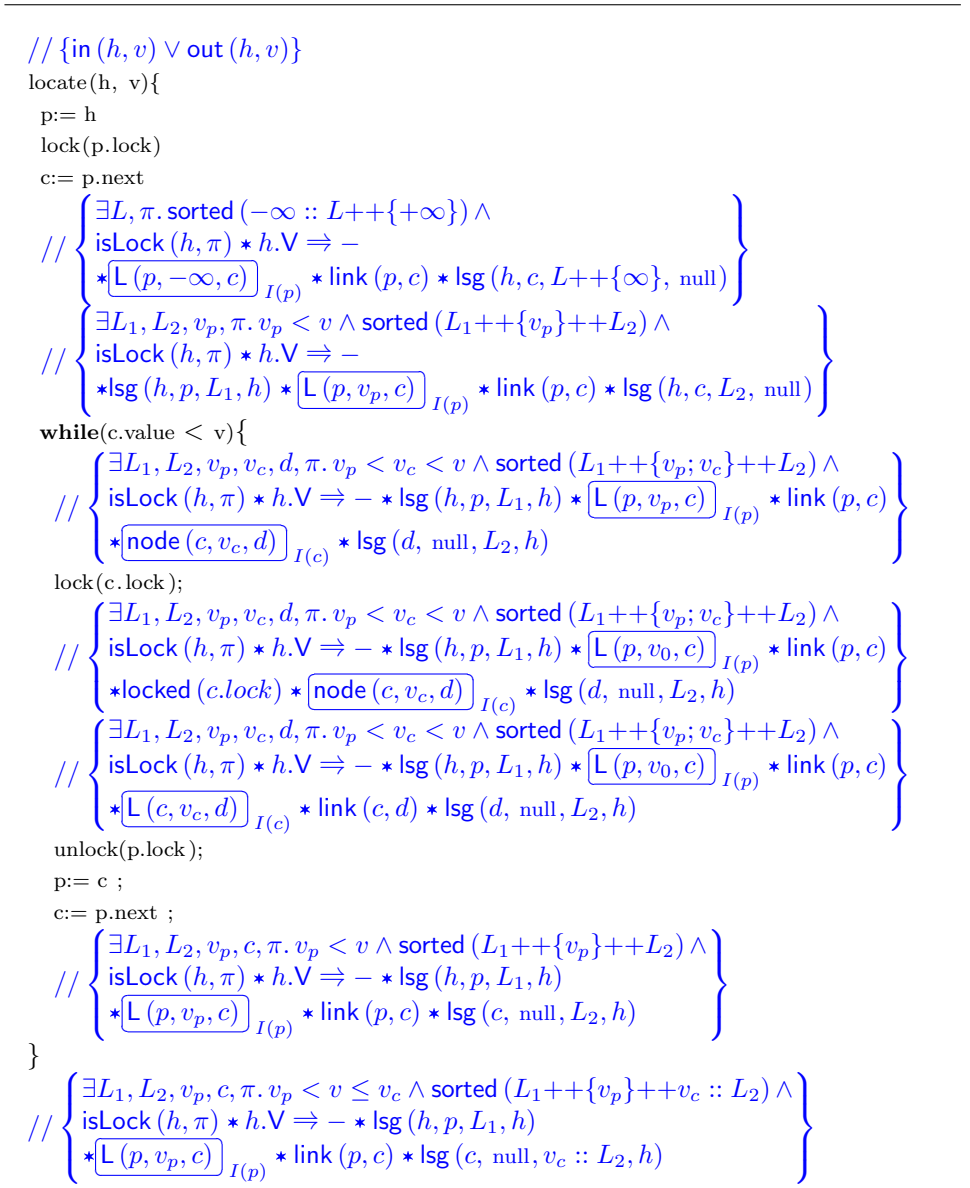

Figure 3.8: Implementation of the set locate operation.

$// \{\mathsf{in}\,(h,v) \vee \mathsf{out}\,(h,v)\}$

remove(h, v){

(p, c):= locate(h, v);

$$// \left\{ \begin{array}{l} \exists L_1, L_2, v_p, \pi.\, v_p < v \leq v_c \wedge \mathsf{sorted}\,(L_1 + +\{v_p\} + +v_c :: L_2) \wedge \\ \mathsf{isLock}\,(h,\pi) * h.\mathsf{V} \Rrightarrow - * \mathsf{lsg}\,(h,p,L_1,h) \\ * \boxed{\mathsf{L}\,(p,v_p,c)}_{I(p)} * \mathsf{link}\,(p,c) * \mathsf{lsg}\,(c,\ \mathrm{null}, v_c :: L_2, h) \end{array} \right\}$$

if (c.value == v) then{

lock(c.lock) ;

$$// \left\{ \begin{array}{l} \exists L_1, L_2, v_p, d, \pi.\, \mathsf{sorted}\,(L_1 + +\{v_p\} + +v_c :: L_2) \wedge \\ h.\mathsf{L} \overset{\pi}{\Rrightarrow} - * h.\mathsf{V} \Rrightarrow - * \mathsf{lsg}\,(h,p,L_1,h) \\ * \boxed{\mathsf{L}\,(p,v_p,c)}_{I(p)} * \mathsf{link}\,(p,c) * \boxed{\mathsf{L}\,(c,v,d)}_{I(c)} * \mathsf{link}\,(c,d) * \mathsf{lsg}\,(d,\ \mathrm{null}, L_2, h) \end{array} \right\}$$

z:= c.next ;

[p.next]:= z ;

$$// \left\{ \begin{array}{l} \exists L_1, L_2, v_p, d.\, \mathsf{sorted}\,(L_1 + +\{v_p\} + +L_2) \wedge \\ h.\mathsf{L} \overset{\pi}{\Rrightarrow} - * h.\mathsf{V} \Rrightarrow - * \mathsf{lsg}\,(h,p,L_1,h) \\ * \boxed{\mathsf{L}\,(p,v_p,c)}_{I(p)} * p.\mathsf{M} \Rrightarrow - * p.next \mapsto d * \mathsf{isLock}\,(c,1) \\ * \boxed{\mathsf{L}\,(c,v,d)}_{I(c)} * c.\mathsf{M} \Rrightarrow - * c.next \mapsto d * \mathsf{isLock}\,(d,1) * \mathsf{lsg}\,(d,\ \mathrm{null}, L_2, h) \end{array} \right\}$$

/* Use the MERGE principle. */

$$// \left\{ \begin{array}{l} \exists L_1, L_2, v_p, d, \pi.\, \mathsf{sorted}\,(L_1 + +\{v_p\} + +L_2) \wedge \\ h.\mathsf{L} \overset{\pi}{\Rrightarrow} - * h.\mathsf{V} \Rrightarrow - * \mathsf{lsg}\,(h,p,L_1,h) \\ * \boxed{\mathsf{L}\,(p,v_p,c) * \mathsf{L}\,(c,v,d)}_{I(p) \cup I(c)} * p.\mathsf{M} \Rrightarrow - * p.next \mapsto d * \mathsf{isLock}\,(c,1) \\ * c.\mathsf{M} \Rrightarrow - * c.next \mapsto d * \mathsf{isLock}\,(d,1) * \mathsf{lsg}\,(d,\ \mathrm{null}, L_2, h) \end{array} \right\}$$

/* Apply the actions of $h.\mathsf{V} \Rrightarrow - * p.\mathsf{M} \Rrightarrow -$ cpability in $I(p)$ in order. */

$$// \left\{ \begin{array}{l} \exists L_1, L_2, v_p, d, \pi.\, \mathsf{sorted}\,(L_1 + +\{v_p\} + +L_2) \wedge \mathsf{isLock}\,(h,\pi) * h.\mathsf{V} \Rrightarrow - * \\ \mathsf{lsg}\,(h,p,L_1,h) * \boxed{\mathsf{L}\,(p,v_p,d)}_{I(p) \cup I(c)} * p.\mathsf{M} \Rrightarrow - * p.next \mapsto d * \mathsf{isLock}\,(c,1) \\ * \mathsf{L}\,(c,v,d) * c.\mathsf{M} \Rrightarrow - * c.next \mapsto d * \mathsf{isLock}\,(d,1) * \mathsf{lsg}\,(d,\ \mathrm{null}, L_2, h) \end{array} \right\}$$

/* Use the SHIFT principle. */

$$// \left\{ \begin{array}{l} \exists L_1, L_2, v_p, d, \pi.\, \mathsf{sorted}\,(L_1 + +\{v_p\} + +L_2) \wedge \mathsf{isLock}\,(h,\pi) * h.\mathsf{V} \Rrightarrow - * \\ \mathsf{lsg}\,(h,p,L_1,h) * \boxed{\mathsf{L}\,(p,v_p,d)}_{I(p)} * p.\mathsf{M} \Rrightarrow - * p.next \mapsto d * \mathsf{isLock}\,(c,1) \\ * \mathsf{L}\,(c,v,d) * c.\mathsf{M} \Rrightarrow - * \mathsf{isLock}\,(d,1) * \mathsf{lsg}\,(d,\ \mathrm{null}, L_2, h) \end{array} \right\}$$

unlock(p) ;

$// \{\mathsf{out}\,(h,v) * \mathsf{L}\,(c,v,d) * \mathsf{isLock}\,(c,1)\}$

$// \{\mathsf{out}\,(h,v) * c.value \mapsto v * c.next \mapsto d * c.lock \mapsto 1\}$

dispose(c, 3) ;

$// \{\mathsf{out}\,(h,v)\}$

}

}

$// \{\mathsf{out}\,(h,v)\}$

Figure 3.9: Implementation of the set remove operation.

Bibliography

[1] Cristiano Calcagno, Peter W. O'Hearn, and Hongseok Yang. Local action and abstract separation logic. In *LICS*, pages 366–378, 2007.

[2] Thomas Dinsdale-Young, Lars Birkedal, Philippa Gardner, Matthew J. Parkinson, and Hongseok Yang. Views: compositional reasoning for concurrent programs. In *POPL*, pages 287–300, 2013.

[3] Thomas Dinsdale-Young, Mike Dodds, Philippa Gardner, Matthew J. Parkinson, and Viktor Vafeiadis. Concurrent abstract predicates. In Theo D'Hondt, editor, *ECOOP*, volume 6183 of *LNCS*, pages 504–528. Springer, 2010.

[4] Mike Dodds, Xinyu Feng, Matthew J. Parkinson, and Viktor Vafeiadis. Deny-guarantee reasoning. In Giuseppe Castagna, editor, *ESOP*, volume 5502 of *Lecture Notes in Computer Science*, pages 363–377. Springer, 2009.

[5] Xinyu Feng. Local rely-guarantee reasoning. In Zhong Shao and Benjamin C. Pierce, editors, *POPL*, pages 315–327. ACM, 2009.

[6] Aquinas Hobor and Jules Villard. The ramifications of sharing in data structures. In *POPL*, pages 523–536, 2013.

[7] Peter W. O'Hearn. Resources, concurrency, and local reasoning. *Theor. Comput. Sci.*, 375(1-3):271–307, 2007.

[8] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS*, pages 55–74. IEEE Computer Society, 2002.

[9] John C. Reynolds. A short course on separation logic. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/fox-19/member/jcr/wwwaac2003/notes7.ps, 2003.

A. Auxiliary Lemmata

Lemma 9 (FORGET-Closure)**.** For all $\mathcal{I}, \mathcal{I}', \mathcal{G} \in \mathsf{AMod}$ and $s_1, s_2, r \in \mathsf{LState}$

$$(\mathcal{I}, \mathcal{G})\!\downarrow (s_1 \circ s_2, r, \mathcal{I}') \implies (\mathcal{I}, \mathcal{G})\!\downarrow (s_1, s_2 \circ r, \mathcal{I}')$$

Proof. Pick an arbitrary $\mathcal{I}, \mathcal{I}', \mathcal{G} \in \mathsf{AMod}$ and $s_1, s_2, r \in \mathsf{LState}$ such that

$$(\mathcal{I}, \mathcal{G})\!\downarrow \left(s_1 \circ s_2, r, \mathcal{I}'\right) \tag{A.1}$$

From the definition of $\downarrow$, it then suffices to show

$$\mathcal{I}' \subseteq \mathcal{I} \tag{A.2}$$
$$\forall n \in \mathbb{N}.\ (\mathcal{I}, \mathcal{G})\!\downarrow_n \left(s_1, s_2 \circ r, \mathcal{I}'\right) \tag{A.3}$$

RTS. (A.2)
This follows trivially from (A.1) and the definition of $\downarrow$.

RTS. (A.3)
Rather than proving (A.3) directly, we first establish the following.

$$\forall n \in \mathbb{N}. \forall s_1, s_2, r \in \mathsf{LState}.$$
$$(\mathcal{I}, \mathcal{G})\!\downarrow_n \left(s_1 \circ s_2, r, \mathcal{I}'\right) \implies (\mathcal{I}, \mathcal{G})\!\downarrow_n \left(s_1, s_2 \circ r, \mathcal{I}'\right) \tag{A.4}$$

We can then despatch (A.3) from (A.1) and (A.4); since for an arbitrary $n \in \mathbb{N}$, from (A.1) and the definition of $\downarrow$ we have $(\mathcal{I}, \mathcal{G})\!\downarrow_n (s_1 \circ s_2, r, \mathcal{I}')$ and consequently from (A.4) we derive $(\mathcal{I}, \mathcal{G})\!\downarrow_n (s_1, s_2 \circ r, \mathcal{I}')$ as required.

RTS. (A.4)
We proceed by induction on the number of steps n.

Base case $n = 0$
Pick an arbitrary $s_1, s_2, r \in \mathsf{LState}$. We are then required to show $(\mathcal{I}, \mathcal{G})\!\downarrow_0$

$(s_1, s_2 \circ r, \mathcal{I}')$ which follows trivially from the definition of $\downarrow_0$.

Inductive Step Pick an arbitrary $n \in \mathbb{N}$ and $s_1, s_2, r \in \mathsf{LState}$ such that

$$(\mathcal{I}, \mathcal{G})\downarrow_n \left(s_1 \circ s_2, r, \mathcal{I}'\right) \tag{A.5}$$

$$\forall s_1, s_2, r \in \mathsf{LState}.\quad (\mathcal{I}, \mathcal{G})\downarrow_{(n-1)} \left(s_1 \circ s_2, r, \mathcal{I}'\right) \implies (\mathcal{I}, \mathcal{G})\downarrow_{(n-1)} \left(s_1, s_2 \circ r, \mathcal{I}'\right) \tag{I.H.}$$

RTS.

$$\forall \kappa.\, \forall a \in \mathcal{I}'(\kappa).\quad (\mathit{potential}(a, s_1 \circ s_2 \circ r) \Rightarrow \forall (s', r') \in a[s_1, s_2 \circ r].\ (\mathcal{I}, \mathcal{G})\downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right)) \wedge \tag{A.6}$$

$$(\mathit{enabled}(a, s_1 \circ s_2 \circ r) \Rightarrow (s_1 \circ s_2 \circ r, a[s_1 \circ s_2 \circ r]) \in \mathcal{G}(\kappa)) \wedge \tag{A.7}$$

$$\forall \kappa.\, \forall a \in \mathcal{I}(\kappa).\, \mathit{potential}(a, s_1 \circ s_2 \circ r) \Rightarrow \mathit{reflected}(a, s_1 \circ s_2 \circ r, \mathcal{I}'(\kappa)) \vee \neg \mathit{visible}(a, s_1) \wedge \forall (s', r') \in a[s_1, s_2 \circ r].\ (\mathcal{I}, \mathcal{G})\downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right) \tag{A.8}$$

RTS. (A.6)
Pick an arbitrary κ, $a \in \mathcal{I}'(\kappa)$ and (s', r') such that

$$\mathit{potential}(a, s_1 \circ s_2 \circ r) \tag{A.9}$$

$$(s', r') \in a[s_1, s_2 \circ r] \tag{A.10}$$

Then from the definition of $a[s_1, s_2 \circ r]$ and by the cross-split property we know there exists $ps_1, ps_2, p_r, s_1', s_2', r''' \in \mathsf{LState}$ such that :

$$\mathit{fst}\,(\Delta a) = ps_1 \circ ps_2 \circ p_r \wedge s_1 = ps_1 \circ s_1' \wedge s_2 = ps_2 \circ s_2' \wedge r = p_r \circ r'' \wedge \begin{pmatrix} (ps_1 > \mathbf{0}_{\mathsf{L}} \wedge s' = \mathit{snd}\,(\Delta a) \circ s_1' \wedge r' = s_2' \circ r'') \\ \vee\, (ps_1 = \mathbf{0}_{\mathsf{L}} s' = s_1 \wedge r' = \mathit{snd}\,(\Delta a) \circ s_2' \circ r'') \end{pmatrix} \tag{A.11}$$

and consequently from the definition of $a[s_1 \circ s_2, r]$

$$\mathit{fst}\,(\Delta a) = ps_1 \circ ps_2 \circ p_r \wedge s_1 = ps_1 \circ s_1' \wedge s_2 = ps_2 \circ s_2' \wedge r = p_r \circ r'' \wedge \begin{pmatrix} (s' = \mathit{snd}\,(\Delta a) \circ s_1' \wedge r' = s_2' \circ r'' \wedge (\mathit{snd}\,(\Delta a) \circ s_1' \circ s_2', r'') \in a[s_1 \circ s_2, r]) \\ \vee \begin{pmatrix} ps_1 = \mathbf{0}_{\mathsf{L}} s' = s_1 \wedge r' = \mathit{snd}\,(\Delta a) \circ s_2' \circ r'' \wedge \\ \begin{pmatrix} (ps_2 = \mathbf{0}_{\mathsf{L}} \wedge (s_1' \circ s_2', \mathit{snd}\,(\Delta a) \circ r'') \in a[s_1 \circ s_2, r]) \\ \vee (ps_2 > \mathbf{0}_{\mathsf{L}} \wedge (s_1' \circ \mathit{snd}\,(\Delta a) \circ s_2', r'') \in a[s_1 \circ s_2, r]) \end{pmatrix} \end{pmatrix} \end{pmatrix}$$

That is,

$$\exists s''.\,(s' \circ s'', r' - s'') \in a[s_1 \circ s_2, r] \tag{A.12}$$

Then from (A.5) and (A.12) we have:

$$(\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s' \circ s'', r' - s'', \mathcal{I}'\right)$$

Finally from (I.H.)

$$(\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(\mathit{snd}(s', (r' - s'') \circ s'', \mathcal{I}'\right)$$

and consequently

$$(\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right)$$

as required.

RTS. (A.7)
Pick an arbitrary κ and $a \in \mathcal{I}'(\kappa)$ such that

$$\mathit{enabled}(a, s_1 \circ s_2 \circ r) \tag{A.13}$$

Then from (A.5) and (A.13) we have:

$$(s_1 \circ s_2 \circ r, a[s_1 \circ s_2 \circ r]) \in \mathcal{G}(\kappa)$$

as required.

RTS. (A.8)
Pick an arbitrary κ and $a \in \mathcal{I}(\kappa)$ such that

$$\mathit{potential}(a, s_1 \circ s_2 \circ r) \tag{A.14}$$

Then from (A.5) we have:

$$\begin{array}{l} \mathit{reflected}(a, s_1 \circ s_2 \circ r, \mathcal{I}'(\kappa)) \vee \\ \neg \mathit{visible}(a, s_1 \circ s_2) \wedge \forall (s', r') \in a[s_1 \circ s_2, r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right) \end{array}$$

If the first disjunct is the case then the desired result holds trivially. On the other hand if the second disjunct is the case then we have:

$$\neg \mathit{visible}(a, s_1 \circ s_2) \wedge \forall (s', r') \in a[s_1 \circ s_2, r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right) \tag{A.15}$$

Pick an arbitrary $s', r' \in \mathsf{LState}$ such that

$$(s', r') \in a[s_1, s_2 \circ r] \tag{A.16}$$

Then from (A.15), (A.16) and the definitions of *visible* and $a[s_1, s_2 \circ r]$ we know there exists $r'' \in \mathsf{LState}$ such that

$$\neg \mathit{visible}(a, s_1) \tag{A.17}$$
$$s' = s_1 \wedge r' = s_2 \circ r'' \tag{A.18}$$
$$(s_1 \circ s_2, r'') \in a[s_1 \circ s_2, r] \tag{A.19}$$

From (A.15) and (A.19) we then have:

$$(\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s_1 \circ s_2, r'', \mathcal{I}'\right) \tag{A.20}$$

and consequently from (I.H.) and (A.18)

$$(\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right) \tag{A.21}$$

Finally from (A.16), (A.17) and (A.21) we have

$$\neg \mathit{visible}(a, s_1) \wedge \forall (s', r') \in a[s_1, s_2 \circ r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right)$$

as required. □

Lemma 10 (MERGE-Closure)**.** For all $\mathcal{I}, \mathcal{I}_1, \mathcal{I}_2, \mathcal{G} \in \mathsf{AMod}$ and $s_p, s_c, s_q, r \in \mathsf{LState}$,

$$(\mathcal{I}, \mathcal{G})\!\downarrow (s_p \circ s_c, s_q \circ r, \mathcal{I}_1) \wedge (\mathcal{I}, \mathcal{G})\!\downarrow (s_q \circ s_c, s_p \circ r, \mathcal{I}_2) \implies (\mathcal{I}, \mathcal{G})\!\downarrow (s_p \circ s_c \circ s_q, r, \mathcal{I}_1 \cup \mathcal{I}_2)$$

Proof. Pick an arbitrary $\mathcal{I}, \mathcal{I}_1, \mathcal{I}_2, \mathcal{G} \in \mathsf{AMod}$ and $s_p, s_c, s_q, r \in \mathsf{LState}$ such that

$$(\mathcal{I}, \mathcal{G})\!\downarrow (s_p \circ s_c, s_q \circ r, \mathcal{I}_1) \wedge (\mathcal{I}, \mathcal{G})\!\downarrow (s_q \circ s_c, s_p \circ r, \mathcal{I}_2) \tag{A.22}$$

From the definition of $\downarrow$, it then suffices to show

$$\mathcal{I}_1 \cup \mathcal{I}_2 \subseteq \mathcal{I} \tag{A.23}$$
$$\forall n \in \mathbb{N}.\ (\mathcal{I}, \mathcal{G})\!\downarrow_n (s_p \circ s_c \circ s_q, r, \mathcal{I}_1 \cup \mathcal{I}_2) \tag{A.24}$$

RTS. (A.23)
Since from (A.22) and the definition of $\downarrow$ we have $\mathcal{I}_1 \subseteq \mathcal{I} \wedge \mathcal{I}_2 \subseteq \mathcal{I}$, we can

thus conclude $\mathcal{I}_1 \cup \mathcal{I}_2 \subseteq \mathcal{I}$ as required.

RTS. (A.24)
Rather than proving (A.24) directly, we first establish the following.

$$\begin{aligned}
&\forall n \in \mathbb{N}.\, \forall s_p, s_c, s_q, r \in \mathsf{LState}.\\
&\quad (\mathcal{I}, \mathcal{G})\!\downarrow_n (s_p \circ s_c, s_q \circ r, \mathcal{I}_1) \wedge (\mathcal{I}, \mathcal{G})\!\downarrow_n (s_c \circ s_q, s_p \circ r, \mathcal{I}_2)\\
&\qquad \Longrightarrow (\mathcal{I}, \mathcal{G})\!\downarrow_n (s_p \circ s_c \circ s_q, r, \mathcal{I}_1 \cup \mathcal{I}_2) \qquad \text{(A.25)}
\end{aligned}$$

We can then despatch (A.24) from (A.22) and (A.25); since for an arbitrary $n \in \mathbb{N}$, from (A.22) and the definition of $\downarrow$ we have $(\mathcal{I}, \mathcal{G}) \downarrow_n (s_p \circ s_c, s_q \circ r, \mathcal{I}_1) \wedge (\mathcal{I}, \mathcal{G})\!\downarrow_n (s_c \circ s_q, s_p \circ r, \mathcal{I}_2)$ and consequently from (A.25) we derive $(\mathcal{I}, \mathcal{G})\!\downarrow_n (s_p \circ s_c \circ s_q, r, \mathcal{I}_1 \cup \mathcal{I}_2)$ as required.

RTS. (A.25)
We proceed by induction on the number of steps n.

Base case $n = 0$
Pick an arbitrary $s_1, s_2, r \in \mathsf{LState}$. We are then required to show $(\mathcal{I}, \mathcal{G})\!\downarrow_0 (s_1, s_2 \circ r, \mathcal{I}')$ which follows trivially from the definition of $\downarrow_0$.

Inductive Step Pick an arbitrary $s_p, s_q, s_c, r \in \mathsf{LState}$ and $n \in \mathbb{N}$, such that

$$(\mathcal{I}, \mathcal{G})\!\downarrow_n (s_p \circ s_c, s_q \circ r, \mathcal{I}_1) \qquad \text{(A.26)}$$

$$(\mathcal{I}, \mathcal{G})\!\downarrow_n (s_q \circ s_c, s_p \circ r, \mathcal{I}_2) \qquad \text{(A.27)}$$

$$\begin{aligned}
&\forall s_p, s_q, s_c, r \in \mathsf{LState}.\\
&\quad (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} (s_p \circ s_c, s_q \circ r, \mathcal{I}_1) \;\wedge\; (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} (s_q \circ s_c, s_p \circ r, \mathcal{I}_2)\\
&\qquad \Longrightarrow (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} (s_p \circ s_c \circ s_q, r, \mathcal{I}_1 \cup \mathcal{I}_2) \qquad \text{(I.H.)}
\end{aligned}$$

RTS.

$$\begin{aligned}
&\forall \kappa.\, \forall a \in (\mathcal{I}_1 \cup \mathcal{I}_2)\,(\kappa).\\
&\quad (\mathit{potential}(a, s_p \circ s_c \circ s_q \circ r) \Rightarrow\\
&\qquad \forall (s', r') \in a[s_p \circ s_c \circ s_q, r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \big(s', r', \mathcal{I}_1 \cup \mathcal{I}_2\big)) \wedge \qquad \text{(A.28)}\\
&\quad (\mathit{enabled}(a, s_p \circ s_c \circ s_q \circ r) \Rightarrow\\
&\qquad\quad (s_p \circ s_c \circ s_q \circ r, a[s_p \circ s_c \circ s_q \circ r]) \in \mathcal{G}(\kappa)) \wedge \qquad \text{(A.29)}\\
&\forall \kappa.\, \forall a \in \mathcal{I}(\kappa).\, \mathit{potential}(a, s_p \circ s_c \circ s_q \circ r) \Rightarrow\\
&\quad \mathit{reflected}(a, s_p \circ s_c \circ s_q \circ r, (\mathcal{I}_1 \cup \mathcal{I}_2)\,(\kappa)) \vee\\
&\quad \neg \mathit{visible}(a, s_p \circ s_c \circ s_q) \wedge \forall (s', r') \in a[s_p \circ s_c \circ s_q, r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \big(s', r', \mathcal{I}_1 \cup \mathcal{I}_2\big)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(A.30)}
\end{aligned}$$

RTS. (A.28)
Pick an arbitrary κ, $a = (p, q) \in (\mathcal{I}_1 \cup \mathcal{I}_2)(\kappa)$ and s', r' such that

$$potential(a, s_p \circ s_c \circ s_q \circ r) \tag{A.31}$$
$$(s', r') \in a[s_p \circ s_c \circ s_q, r] \tag{A.32}$$

Then from the definition of $a[s_p \circ s_c \circ s_q]$ and by the cross-split property we know there exists $p_p, p_c, p_q, s'_p, s'_c, s'_q \in \mathsf{LState}$ such that :

$$\begin{array}{l} s' = s_p \circ s_c \circ s_q \vee \\ \left(\begin{array}{l} (p_p > \mathbf{0}_{\mathsf{L}} \vee p_c > \mathbf{0}_{\mathsf{L}} \vee p_q > \mathbf{0}_{\mathsf{L}}) \wedge s' = snd\,(\Delta a) \circ s'_p \circ s'_c \circ s'_q \\ \wedge fst\,(\Delta a) = p_p \circ p_c \circ p_q \circ p_r \\ \wedge s_p = p_p \circ s'_p \wedge s_c = p_c \circ s'_c \wedge s_q = p_q \circ s'_q \wedge r = p_r \circ r' \end{array} \right) \end{array} \tag{A.33}$$

and consequently from the definitions of $a[s_p \circ s_c, s_q \circ r]$ and $a[s_c \circ s_q, s_p \circ r]$ we have:

$$\begin{array}{l} \left(\begin{array}{l} s' = s_p \circ s_c \circ s_q \wedge \\ (s_p \circ s_c, s_q \circ r') \in a[s_p \circ s_c, s_q \circ r] \wedge (s_c \circ s_q, s_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \end{array} \right) \\ \vee \left(\begin{array}{l} s' = snd\,(\Delta a) \circ s'_p \circ s'_c \circ s'_q \wedge \\ \left(\begin{array}{l} \left(\begin{array}{l} ((p_c > \mathbf{0}_{\mathsf{L}} \vee (p_c = \mathbf{0}_{\mathsf{L}} \wedge p_p > \mathbf{0}_{\mathsf{L}} \wedge p_q > \mathbf{0}_{\mathsf{L}}))) \wedge \\ (snd\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r') \in a[s_p \circ s_c, s_q \circ r] \wedge \\ (snd\,(\Delta a) \circ s'_c \circ s'_q, s'_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \end{array} \right) \\ \vee \left(\begin{array}{l} p_p > \mathbf{0}_{\mathsf{L}} \wedge p_c = p_q = \mathbf{0}_{\mathsf{L}} \wedge \\ (snd\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r') \in a[s_p \circ s_c, s_q \circ r] \wedge \\ (s'_c \circ s'_q, snd\,(\Delta a) \circ s'_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \end{array} \right) \\ \vee \left(\begin{array}{l} p_q > \mathbf{0}_{\mathsf{L}} \wedge p_c = p_p = \mathbf{0}_{\mathsf{L}} \wedge \\ (s'_p \circ s'_c, snd\,(\Delta a) \circ s'_q \circ r') \in a[s_p \circ s_c, s_q \circ r] \wedge \\ (snd\,(\Delta a) \circ s'_c \circ s'_q, s'_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \end{array} \right) \end{array} \right) \end{array} \right) \end{array} \tag{A.34}$$

From the definition of $\mathcal{I}_1 \cup \mathcal{I}_2$ we know:

$$a \in \mathcal{I}_1(\kappa) \vee a \in \mathcal{I}_2(\kappa)$$

There are two cases to consider:

Case 1. $a \in \mathcal{I}_1(\kappa)$

From (A.34), the assumption of case 1, (A.26) and (A.31) we have:

$$\begin{pmatrix} s' = s_p \circ s_c \circ s_q \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} (s_p \circ s_c, s_q \circ r', \mathcal{I}_1) \wedge (s_c \circ s_q, s_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \end{pmatrix} \vee \left(\begin{array}{l} s' = snd\,(\Delta a) \circ s'_p \circ s'_c \circ s'_q \wedge \\ \left(\begin{array}{l} \left(\begin{array}{l} ((p_c > \mathbf{0_L} \vee (p_c = \mathbf{0_L} \wedge p_p > \mathbf{0_L} \wedge p_q > \mathbf{0_L}))) \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(snd\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (snd\,(\Delta a) \circ s'_c \circ s'_q, s'_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \end{array} \right) \\ \vee \left(\begin{array}{l} p_p > \mathbf{0_L} \wedge p_c = p_q = \mathbf{0_L} \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(snd\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (s'_c \circ s'_q, snd\,(\Delta a) \circ s'_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \end{array} \right) \\ \vee \left(\begin{array}{l} p_q > \mathbf{0_L} \wedge p_c = p_p = \mathbf{0_L} \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(s'_p \circ s'_c, snd\,(\Delta a) \circ s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (snd\,(\Delta a) \circ s'_c \circ s'_q, s'_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \end{array} \right) \end{array} \right) \end{array} \right) \tag{A.35}$$

Since $\mathcal{I}_1 \subseteq \mathcal{I}$, we know $a \in \mathcal{I}(\kappa)$. Consequently, from (A.27) and (A.31) we have:

$$\begin{array}{l} \mathit{reflected}(a, s_p \circ s_c \circ s_q \circ r, \mathcal{I}_2(\kappa)) \vee \\ \neg \mathit{visible}(a, s_c \circ s_q) \wedge (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} (s_c \circ s_q, a[s_p \circ s_c \circ s_q \circ e] - s_c \circ s_q, \mathcal{I}_2) \end{array}$$

There are two cases to consider:

Case 1.1.

$$\neg \mathit{visible}(a, s_c \circ s_q) \wedge (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} (s_c \circ s_q, a[s_p \circ s_c \circ s_q \circ e] - s_c \circ s_q, \mathcal{I}_2)$$

By definition of *visible* and from the assumption of case 1.1 and (A.35) we have:

$$\begin{pmatrix} s' = s_p \circ s_c \circ s_q \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} (s_p \circ s_c, s_q \circ r', \mathcal{I}_1) \wedge (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} (s_c \circ s_q, s_p \circ r', \mathcal{I}_2) \end{pmatrix} \vee \begin{pmatrix} s' = snd\,(\Delta a) \circ s'_p \circ s'_c \circ s'_q \wedge p_c = p_q = \mathbf{0_L} \wedge s'_c = s_c \wedge s'_q = s_q \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(snd\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(s'_c \circ s'_q, snd\,(\Delta a) \circ s'_p \circ r', \mathcal{I}_2\right) \end{pmatrix} \tag{A.36}$$

From (A.36) and (I.H.) we have:

$$(\mathcal{I},\mathcal{G})\downarrow_{(n-1)} (s', r', \mathcal{I}_1 \cup \mathcal{I}_2)$$

as required.

Case 1.2.

$$\mathit{reflected}(a, s_p \circ s_c \circ s_q \circ r, \mathcal{I}_2(\kappa))$$

From (A.31), the definition of *potential* and by Lemma 23 we have:

$$\begin{aligned}
&\exists l.\, \mathit{fst}\,(a) < s_p \circ s_c \circ s_q \circ r \circ l \wedge \\
&\exists l.\, \mathit{fst}\,(\Delta a) \circ l = s_p \circ s_c \circ s_q \circ r \wedge \mathit{snd}\,(\Delta a) \sharp l
\end{aligned}$$

and thus from the assumption of case 1.1.2. we know there exists $a' \in \mathcal{I}_2(\kappa)$ such that:

$$\begin{aligned}
&\Delta a' = \Delta a \wedge \\
&\exists l.\, \mathit{fst}\left(a'\right) < s_p \circ s_c \circ s_q \circ r \circ l \wedge \\
&\exists l.\, \mathit{fst}\left(\Delta a'\right) \circ l = s_p \circ s_c \circ s_q \circ r \wedge \mathit{snd}\left(\Delta a'\right) \sharp l
\end{aligned}$$

and consequently from the definition of *potential* and Lemma 23 we have:

$$\Delta a' = \Delta a \wedge \mathit{potential}(a', s_p \circ s_c \circ s_q \circ s_r) \tag{A.37}$$

On the other hand, from the definition of $a[s_c \circ s_q, s_p \circ r]$ and since $\Delta a' = \Delta a$(A.37), from (A.35) we have:

$$\begin{aligned}
&\begin{pmatrix} s' = s_p \circ s_c \circ s_q \wedge \\ (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ (s_p \circ s_c, s_q \circ r', \mathcal{I}_1) \wedge (s_c \circ s_q, s_p \circ r') \in a'[s_c \circ s_q, s_p \circ r] \end{pmatrix} \\
&\vee \left(\begin{array}{l}
s' = \mathit{snd}\,(\Delta a) \circ s'_p \circ s'_c \circ s'_q \wedge \\
\left(\begin{array}{l}
\begin{pmatrix} ((p_c > \mathbf{0_L} \vee (p_c = \mathbf{0_L} \wedge p_p > \mathbf{0_L} \wedge p_q > \mathbf{0_L}))) \wedge \\ (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(\mathit{snd}\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (\mathit{snd}\,(\Delta a) \circ s'_c \circ s'_q, s'_p \circ r') \in a'[s_c \circ s_q, s_p \circ r] \end{pmatrix} \\
\vee \begin{pmatrix} p_p > \mathbf{0_L} \wedge p_c = p_q = \mathbf{0_L} \wedge \\ (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(\mathit{snd}\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (s'_c \circ s'_q, \mathit{snd}\,(\Delta a) \circ s'_p \circ r') \in a'[s_c \circ s_q, s_p \circ r] \end{pmatrix} \\
\vee \begin{pmatrix} p_q > \mathbf{0_L} \wedge p_c = p_p = \mathbf{0_L} \wedge \\ (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s'_p \circ s'_c, \mathit{snd}\,(\Delta a) \circ s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (\mathit{snd}\,(\Delta a) \circ s'_c \circ s'_q, s'_p \circ r') \in a'[s_c \circ s_q, s_p \circ r] \end{pmatrix}
\end{array} \right)
\end{array} \right)
\end{aligned} \tag{A.38}$$

and thus from (A.27) we have:

$$\begin{pmatrix} s' = s_p \circ s_c \circ s_q \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} (s_p \circ s_c, s_q \circ r', \mathcal{I}_1) \wedge (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} (s_c \circ s_q, s_p \circ r', \mathcal{I}_2) \end{pmatrix}$$
$$\vee \left(\begin{array}{l} s' = snd\,(\Delta a) \circ s'_p \circ s'_c \circ s'_q \wedge \\ \left(\begin{array}{l} \left(\begin{array}{l} ((p_c > \mathbf{0}_{\mathsf{L}} \vee (p_c = \mathbf{0}_{\mathsf{L}} \wedge p_p > \mathbf{0}_{\mathsf{L}} \wedge p_q > \mathbf{0}_{\mathsf{L}}))) \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(snd\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(snd\,(\Delta a) \circ s'_c \circ s'_q, s'_p \circ r', \mathcal{I}_2\right) \end{array} \right) \\ \vee \left(\begin{array}{l} p_p > \mathbf{0}_{\mathsf{L}} \wedge p_c = p_q = \mathbf{0}_{\mathsf{L}} \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(snd\,(\Delta a) \circ s'_p \circ s'_c, s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(s'_c \circ s'_q, snd\,(\Delta a) \circ s'_p \circ r', \mathcal{I}_2\right) \end{array} \right) \\ \vee \left(\begin{array}{l} p_q > \mathbf{0}_{\mathsf{L}} \wedge p_c = p_p = \mathbf{0}_{\mathsf{L}} \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(s'_p \circ s'_c, snd\,(\Delta a) \circ s'_q \circ r', \mathcal{I}_1\right) \wedge \\ (\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(snd\,(\Delta a) \circ s'_c \circ s'_q, s'_p \circ r', \mathcal{I}_2\right) \end{array} \right) \end{array} \right) \end{array} \right) \tag{A.39}$$

From (A.39) and (I.H.) we have:

$$(\mathcal{I},\mathcal{G})\downarrow_{(n-1)} \left(s', r', \mathcal{I}_1 \cup \mathcal{I}_2\right)$$

as required.

Case 2. $a \in \mathcal{I}_2(\kappa)$
The proof of this case is analogous to that of previous case and is omitted here.

RTS. (A.29)
Pick an arbitrary κ and $a \in (\mathcal{I}_1 \cup \mathcal{I}_2)(\kappa)$ such that

$$enabled(a, s_p \circ s_c \circ s_q \circ r) \tag{A.40}$$

There are two cases to consider:

Case 1. $a \in \mathcal{I}_1(\kappa)$
From assumption of case 1, (A.26) and (A.40) we then have:

$$(s_p \circ s_c \circ s_q \circ r, a[s_p \circ s_c \circ s_q \circ r]) \in \mathcal{G}(\kappa)$$

as required.

Case 2. $a \in \mathcal{I}_2(\kappa)$
The proof of this case is analogous to that of previous case and is omitted here.

RTS. (A.30)
Pick an arbitrary κ and $a \in \mathcal{I}(\kappa)$ such that

$$\mathit{potential}(a, s_p \circ s_c \circ s_q \circ r) \tag{A.41}$$

From (A.26) and (A.41) we have:

$$\begin{array}{l}\mathit{reflected}(a, s_p \circ s_c \circ s_q \circ r, \mathcal{I}_1(\kappa)) \vee \\ \neg\mathit{visible}(a, s_p \circ s_c) \wedge \forall(s', r') \in a[s_p \circ s_c, s_q \circ r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_1\right)\end{array}$$

and consequently from the definition of $\mathcal{I}_1 \cup \mathcal{I}_2$ we have:

$$\begin{array}{l}\mathit{reflected}(a, s_p \circ s_c \circ s_q \circ r, (\mathcal{I}_1 \cup \mathcal{I}_2)(\kappa)) \vee \\ \neg\mathit{visible}(a, s_p \circ s_c) \wedge \forall(s', r') \in a[s_p \circ s_c, s_q \circ r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_1\right)\end{array} \tag{A.42}$$

Similarly, from (A.27) and (A.41) we have:

$$\begin{array}{l}\mathit{reflected}(a, s_p \circ s_c \circ s_q \circ r, (\mathcal{I}_1 \cup \mathcal{I}_2)(\kappa)) \vee \\ \neg\mathit{visible}(a, s_c \circ s_q) \wedge \forall(s', r') \in a[s_c \circ s_q, s_p \circ r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_2\right)\end{array} \tag{A.43}$$

From (A.42) and (A.43) we have:

$$\begin{array}{l}\mathit{reflected}(a, s_p \circ s_c \circ s_q \circ r, (\mathcal{I}_1 \cup \mathcal{I}_2)(\kappa)) \vee \\ \neg\mathit{visible}(a, s_p \circ s_c) \wedge \forall(s', r') \in a[s_p \circ s_c, s_q \circ r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_1\right) \\ \neg\mathit{visible}(a, s_c \circ s_q) \wedge \forall(s', r') \in a[s_c \circ s_q, s_p \circ r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_2\right)\end{array} \tag{A.44}$$

If the first disjunct is the case then the desired result holds trivially. On the other hand, in case of the second disjunct we have:

$$\begin{array}{l}\neg\mathit{visible}(a, s_p \circ s_c) \wedge \neg\mathit{visible}(a, s_c \circ s_q) \\ \wedge\forall(s', r') \in a[s_p \circ s_c, s_q \circ r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} (s', r', \mathcal{I}_1) \\ \wedge\forall(s', r') \in a[s_c \circ s_q, s_p \circ r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} (s', r', \mathcal{I}_2)\end{array} \tag{A.45}$$

From (A.45) and by definition of *visible* we have:

$$\neg\mathit{visible}(a, s_p \circ s_c \circ s_q) \tag{A.46}$$

Pick an arbitrary $s', r' \in \mathsf{LState}$ such that

$$(s', r') \in a[s_p \circ s_c \circ s_q, r] \tag{A.47}$$

Then from (A.46), (A.47) and the definitions of $a[s_p \circ s_c, s_q \circ r]$ and $a[s_c \circ s_q, s_p \circ r]$ we know

$$s' = s_p \circ s_c \circ s_q \tag{A.48}$$

$$(s_p \circ s_c, s_q \circ r') \in a[s_p \circ s_c, s_q \circ r] \wedge (s_c \circ s_q, s_p \circ r') \in a[s_c \circ s_q, s_p \circ r] \tag{A.49}$$

Consequently from (A.45) and (A.49) we have

$$(\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s_p \circ s_c, s_q \circ r', \mathcal{I}_1\right) \wedge (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s_c \circ s_q, s_p \circ r', \mathcal{I}_2\right)$$

and thus from (I.H.) and (A.48)

$$(\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_1 \cup \mathcal{I}_2\right) \tag{A.50}$$

Finally, from (A.46), (A.47) and (A.50) we have:

$$\neg \mathit{visible}(a, s_p \circ s_c \circ s_q) \wedge$$
$$\forall (s', r') \in a[s_p \circ s_c \circ s_q, r].\ (\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_1 \cup \mathcal{I}_2\right)$$

as required.

□

Lemma 11 (Shift-Fence)**.** For all $\mathcal{I}_1, \mathcal{I}_2 \in \mathsf{AMod}$, $s, s', r \in \mathsf{LState}$ and $a \in rg(\mathcal{I}_1)$:

$$\mathcal{I}_1 \sqsubseteq^{\{s\}} \mathcal{I}_2 \wedge \left(s' \in a(s) \vee (s', -) \in a[s, r]\right) \implies \mathcal{I}_1 \sqsubseteq^{\{s'\}} \mathcal{I}_2$$

Proof. Pick an arbitrary $\mathcal{I}_1, \mathcal{I}_2 \in \mathsf{AMod}$, $s, s', r \in \mathsf{LState}$ and $a \in rg(\mathcal{I}_1)$ such that:

$$\mathcal{I}_1 \sqsubseteq^{\{s\}} \mathcal{I}_2 \tag{A.51}$$

RTS. $\mathcal{I}_1 \sqsubseteq^{\{s'\}} \mathcal{I}_2$.
There are two cases to consider:
Case 1. $s' \in a(s)$
From the definition of $\sqsubseteq^{\{s\}}$ and (A.51) we know there exists a fence $\mathcal{F}$ such that:

$$s \in \mathcal{F} \tag{A.52}$$

$$\mathcal{F} \rhd \mathcal{I}_1 \tag{A.53}$$

$$\forall l \in \mathcal{F}.\, \forall \kappa.\, \forall a \in \mathcal{I}_2(\kappa).\, \mathit{reflected}(a, l, \mathcal{I}_1(\kappa)) \wedge$$
$$\forall a \in \mathcal{I}_1(\kappa).\, a(l) \text{ is defined } \wedge \mathit{visible}(a, l) \implies \mathit{reflected}(a, l, \mathcal{I}_2(\kappa)) \tag{A.54}$$

By definition of $\rhd$ and from (A.52)-(A.53) and assumption of case 1. we have:

$$s' \in \mathcal{F} \tag{A.55}$$

Finally by definition of $\sqsubseteq^{\{s'\}}$ and (A.53)-(A.55) we have

$$\mathcal{I}_1 \sqsubseteq^{\{s'\}} \mathcal{I}_2$$

as required.

Case 2. $(s', -) \in a[s, r]$
From the definitions of $a[s, r]$ and $a(s)$ and from the assumption of the case we know $s' \in a(s)$. The rest of the proof is identical to that of case 1. $\square$

Lemma 12 (SHIFT-Closure-1)**.** For all $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}, \mathcal{G} \in \mathsf{AMod}$ and $s, r \in \mathsf{LState}$,

$$(\mathcal{I}, \mathcal{G})\!\downarrow (s, r, \mathcal{I}_1) \wedge \mathcal{I}_1 \sqsubseteq^{\{s\}} \mathcal{I}_2 \implies (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\!\downarrow (s, r, \mathcal{I}_2)$$

Proof. Pick an arbitrary $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}, \mathcal{G} \in \mathsf{AMod}$ and $s, r \in \mathsf{LState}$ such that

$$(\mathcal{I}, \mathcal{G})\!\downarrow (s, r, \mathcal{I}_1) \tag{A.56}$$
$$\mathcal{I}_1 \sqsubseteq^{\{s\}} \mathcal{I}_2 \tag{A.57}$$

From the definition of $\downarrow$, it then suffices to show

$$\mathcal{I}_2 \subseteq \mathcal{I} \cup \mathcal{I}_2 \tag{A.58}$$
$$\forall n \in \mathbb{N}.\ (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\!\downarrow_n (s, r, \mathcal{I}_2) \tag{A.59}$$

RTS. (A.58)
This holds trivially from the definition of $\mathcal{I} \cup \mathcal{I}_2$.

RTS. (A.59)
Rather than proving (A.59) directly, we first establish the following.

$$\forall n \in \mathbb{N}. \forall s, r \in \mathsf{LState}.$$
$$(\mathcal{I}, \mathcal{G})\!\downarrow_n (s, r, \mathcal{I}_1) \wedge \mathcal{I}_1 \sqsubseteq^{\{s\}} \mathcal{I}_2 \implies (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\!\downarrow_n (s, r, \mathcal{I}_2) \tag{A.60}$$

We can then despatch (A.59) from (A.56), (A.57) and (A.60); since for an arbitrary $n \in \mathbb{N}$, from (A.56) and the definition of $\downarrow$ we have $(\mathcal{I}, \mathcal{G})\!\downarrow_n (s, r, \mathcal{I}_1)$ and consequently from (A.57) and (A.60) we derive $(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\!\downarrow_n (s, r, \mathcal{I}_2)$ as

required.

RTS. (A.60)
We proceed by induction on the number of steps n.

Base case $n = 0$
Pick an arbitrary $s, r \in \mathsf{LState}$. We are then required to show $(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\downarrow_0 (s, r, \mathcal{I}_2)$ which follows trivially from the definition of $\downarrow_0$.

Inductive Case
Pick an arbitrary $s, r \in \mathsf{LState}$ such that:

$$(\mathcal{I}, \mathcal{G})\downarrow_n (s, r, \mathcal{I}_1) \tag{A.61}$$
$$\mathcal{I}_1 \sqsubseteq^{\{s\}} \mathcal{I}_2 \tag{A.62}$$
$$\forall s, r \in \mathsf{LState}.$$
$$(\mathcal{I}, \mathcal{G})\downarrow_{(n-1)} (s, r, \mathcal{I}_1) \wedge \mathcal{I}_1 \sqsubseteq^{\{s\}} \mathcal{I}_2 \implies (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\downarrow_{(n-1)} (s, r, \mathcal{I}_2) \tag{I.H}$$

RTS.

$$\forall \kappa.\, \forall a \in \mathcal{I}_2(\kappa).$$
$$(\mathit{potential}(a, s \circ r) \Rightarrow$$
$$\forall (s', r') \in a[s, r].\, (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\downarrow_{(n-1)} \left(s', r', \mathcal{I}_2\right)) \wedge \tag{A.63}$$
$$(\mathit{enabled}(a, s \circ r) \Rightarrow (s \circ r, a[s \circ r]) \in \mathcal{G}(\kappa)) \wedge \tag{A.64}$$
$$\forall \kappa.\, \forall a \in (\mathcal{I} \cup \mathcal{I}_2)\,(\kappa).\, \mathit{potential}(a, s \circ r) \Rightarrow$$
$$\mathit{reflected}(a, s \circ r, \mathcal{I}_2(\kappa)) \vee$$
$$\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\, (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\downarrow_{(n-1)} \left(s', r', \mathcal{I}_2\right) \tag{A.65}$$

RTS. (A.63)
Pick an arbitrary κ, $a \in \mathcal{I}_2(\kappa)$ and $s', r' \in \mathsf{LState}$ such that:

$$\mathit{potential}(a, s \circ r) \tag{A.66}$$
$$(s', r') \in a[s, r] \tag{A.67}$$

From (A.66), the definition of *potential* and by Lemma 23 we know there exists l such that

$$\mathit{fst}\,(a) < s \circ r \circ l \wedge$$
$$\exists l'.\, \mathit{fst}\,(\Delta a) \circ l' = s \circ r \wedge \mathit{snd}\,(\Delta a) \sharp l'$$

and thus from (A.62) we know there exists $a' \in \mathcal{I}_1(\kappa)$ such that:

$$\begin{array}{l} \Delta a' = \Delta a \wedge \\ \mathit{fst}\left(a'\right) < s \circ r \circ l \wedge \\ \exists l'.\, \mathit{fst}\left(\Delta a'\right) \circ l' = s \circ r \wedge \mathit{snd}\left(\Delta a'\right) \sharp\, l' \end{array}$$

and consequently from the definition of *potential* and Lemma 23 we have:

$$\Delta a' = \Delta a \wedge \mathit{potential}(a', s \circ r) \tag{A.68}$$

Since $\Delta a' = \Delta a$, from the definition of $a[s, r]$ we know $a[s, r] = a'[s, r]$. Thus, from (A.61), (A.66), (A.67) and (A.68) we have:

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_1\right) \tag{A.69}$$

From (A.62), (A.67), Lemma 11 and since $a[s, r] = a'[s, r]$, we have

$$\mathcal{I}_1 \sqsubseteq^{\{s'\}} \mathcal{I}_2 \tag{A.70}$$

Finally, from (A.69), (A.70) and (I.H) we have:

$$(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_2\right)$$

as required.

RTS. (A.64)
Pick an arbitrary κ and $a \in \mathcal{I}_2(\kappa)$ such that:

$$\mathit{enabled}(a, s \circ r) \tag{A.71}$$

From (A.61) and (A.71) we then have:

$$(s \circ r, a[s \circ r]) \in \mathcal{G}(\kappa)$$

as required.

RTS. (A.65)
Pick an arbitrary κ and $a \in (\mathcal{I} \cup \mathcal{I}_2)(\kappa)$ such that:

$$\mathit{potential}(a, s \circ r) \tag{A.72}$$

If $a \in \mathcal{I}_2(\kappa)$, then it is trivially the case that *reflected*$(a, s \circ r)$ and thus the desired result (A.65) holds. On the other hand, if $a \in \mathcal{I}(\kappa)$, then from (A.61) and (A.72) we have:

$$\begin{array}{l} \mathit{reflected}(a, s \circ r, \mathcal{I}_1(\kappa)) \vee \\ \neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_1\right) \end{array}$$

There are two cases to consider:
Case 1.

$$\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} (s', r', \mathcal{I}_1)$$

From the assumption of case 1, (A.62) and (I.H) we have:

$$\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_{(n-1)} (s', r', \mathcal{I}_2)$$

as required.

Case 2.

$$\mathit{reflected}(a, s \circ r, \mathcal{I}_1(\kappa))$$

Pick an arbitrary $l \in \mathsf{LState}$ such that:

$$\mathit{fst}\,(a) \leq s \circ r \circ l \tag{A.73}$$

Then from the assumption of case 2 and the definition of *reflected* we have:

$$\exists a' \in \mathcal{I}_1(\kappa).\, \mathit{fst}\left(a'\right) \leq s \circ r \circ l \wedge \Delta a' = \Delta a \tag{A.74}$$

Since either $\mathit{visible}(a', s)$ or $\neg \mathit{visible}(a', s)$, there are two cases to consider:
Case 2.1. $\mathit{visible}(a', s)$
From (A.72) and by definition of *potential* we know $a[s \circ r]$ is defined; from (A.74), and the definition of $a'[s \circ r]$ we know that $a'[s \circ r]$ is also defined. Consequently, from the definition of $a'(s)$, we know $a'(s)$ is also defined. Thus, from the assumptions of case 2.1, (A.62), (A.74) and from the definition of $\sqsubseteq^{\{s\}}$ we have

$$\exists a'' \in \mathcal{I}_2(\kappa).\, \mathit{fst}\left(a''\right) \leq s \circ r \circ l \wedge \Delta a'' = \Delta a' \tag{A.75}$$

Finally, from (A.73), (A.74), (A.75) and by definition of *reflected* we have:

$$\mathit{reflected}(a, s \circ r, \mathcal{I}_2(\kappa))$$

as required.

Case 2.2. $\neg \mathit{visible}(a', s)$
From (A.72), (A.74) and by definition of *potential* we have:

$$\mathit{potential}(a', s \circ r) \tag{A.76}$$

Consequently, from (A.61), (A.74), and (A.76)

$$\forall (s', r') \in a'[s, r].\ (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s', r', \mathcal{I}_1\right)$$

and since $\Delta a = \Delta a'$ (A.74), by definition of $a[s, r]$ we have:

$$\forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s', r', \mathcal{I}_1\right)$$

Consequently, from (A.62) and (I.H) we have:

$$\forall (s', r') \in a[s, r].\ (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s', r', \mathcal{I}_2\right) \tag{A.77}$$

On the other hand, since $\Delta a = \Delta a'$ (A.74), from the definition of *visible* and the assumption of case 2.2. we have:

$$\neg \mathit{visible}(a, s) \tag{A.78}$$

Finally, from (A.77) and (A.78) we have:

$$\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s', r', \mathcal{I}_2\right)$$

as required.

□

Lemma 13 (SHIFT-Closure-2)**.** For all $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2, \mathcal{I}, \mathcal{G} \in \mathsf{AMod}$ and $s_1, r_1, s_0, r_0 \in \mathsf{LState}$

$$\begin{array}{l} (\mathcal{I}, \mathcal{G}){\downarrow}\ (s_1, r_1, \mathcal{I}_1) \wedge \mathcal{I}_1 \sqsubseteq^{\{s_1\}} \mathcal{I}_2 \wedge \\ (\mathcal{I}, \mathcal{G}){\downarrow}\ (s_0, r_0, \mathcal{I}_0) \wedge s_1 \circ r_1 = s_0 \circ r_0 \implies \\ \qquad\qquad\qquad\qquad\qquad\qquad (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}){\downarrow}\ (s_0, r_0, \mathcal{I}_0) \end{array}$$

Proof. Pick an arbitrary $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2, \mathcal{I}, \mathcal{G} \in \mathsf{AMod}$ and $s_1, r_1, s_0, r_0 \in \mathsf{LState}$ such that

$$(\mathcal{I}, \mathcal{G}){\downarrow}\ (s_1, r_1, \mathcal{I}_1) \tag{A.79}$$

$$\mathcal{I}_1 \sqsubseteq^{\{s_1\}} \mathcal{I}_2 \tag{A.80}$$

$$(\mathcal{I}, \mathcal{G}){\downarrow}\ (s_0, r_0, \mathcal{I}_0) \tag{A.81}$$

$$s_1 \circ r_1 = s_0 \circ r_0 \tag{A.82}$$

From the definition of $\downarrow$, it then suffices to show

$$\mathcal{I}_0 \subseteq \mathcal{I} \cup \mathcal{I}_2 \tag{A.83}$$

$$\forall n \in \mathbb{N}.\ (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_n (s_0, r_0, \mathcal{I}_0) \tag{A.84}$$

RTS. (A.83)
From (A.81) and the definition of $\downarrow$ we have $\mathcal{I}_0 \subseteq \mathcal{I}$ and consequently $\mathcal{I}_0 \subseteq \mathcal{I} \cup \mathcal{I}_2$ as required.

RTS. (A.84)
Rather than proving (A.84) directly, we first establish the following.

$$\begin{array}{l} \forall n \in \mathbb{N}.\, \forall s_1, r_1, s_0, r_0 \in \mathsf{LState}. \\ \quad (\mathcal{I}, \mathcal{G}) \downarrow_n (s_1, r_1, \mathcal{I}_1) \wedge \mathcal{I}_1 \sqsubseteq^{\{s_1\}} \mathcal{I}_2 \wedge \\ \quad (\mathcal{I}, \mathcal{G}) \downarrow_n (s_0, r_0, \mathcal{I}_0) \wedge s_1 \circ r_1 = s_0 \circ r_0 \implies \\ \qquad\qquad (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_n (s_0, r_0, \mathcal{I}_0) \end{array} \tag{A.85}$$

We can then despatch (A.84) from (A.79)-(A.82) and (A.85); since for an arbitrary $n \in \mathbb{N}$, from (A.79), (A.81) and the definition of $\downarrow$ we have $(\mathcal{I}, \mathcal{G}) \downarrow_n (s_1, r_1, \mathcal{I}_1) \wedge (\mathcal{I}, \mathcal{G}) \downarrow_n (s_0, r_0, \mathcal{I}_0)$ and consequently from (A.80), (A.82) and (A.85) we derive $(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_n (s_0, r_0, \mathcal{I}_0)$ as required.

RTS. (A.85)
We proceed by induction on the number of steps n.

Base case $n = 0$
Pick an arbitrary $s_1, r_1, s_0, r_0 \in \mathsf{LState}$. We are then required to show $(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_0 (s_0, r_0, \mathcal{I}_0)$ which follows trivially from the definition of $\downarrow_0$.

Inductive Case
Pick an arbitrary $n \in \mathbb{N}$ and $s_1, r_1, s_0, r_0 \in \mathsf{LState}$ such that:

$$(\mathcal{I}, \mathcal{G}) \downarrow_n (s_1, r_1, \mathcal{I}_1) \tag{A.86}$$

$$\mathcal{I}_1 \sqsubseteq^{\{s_1\}} \mathcal{I}_2 \tag{A.87}$$

$$(\mathcal{I}, \mathcal{G}) \downarrow_n (s_0, r_0, \mathcal{I}_0) \tag{A.88}$$

$$s_1 \circ r_1 = s_0 \circ r_0 \tag{A.89}$$

$$\begin{array}{l} \forall s_1', r_1', s_0', r_0' \in \mathsf{LState}. \\ \quad (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} (s_1', r_1', \mathcal{I}_1) \wedge \mathcal{I}_1 \sqsubseteq^{\{s_1'\}} \mathcal{I}_2 \wedge \\ \quad (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} (s_0', r_0', \mathcal{I}_0) \wedge s_1' \circ r_1' = s_0' \circ r_0' \implies \\ \qquad\qquad (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_{(n-1)} (s_0', r_0', \mathcal{I}_0) \end{array} \tag{I.H}$$

RTS.

$$\forall\kappa.\,\forall a \in \mathcal{I}_0(\kappa).$$
$$(\mathit{potential}(a, s_0 \circ r_0) \Rightarrow$$
$$\forall (s', r') \in a[s_0, r_0].\ (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)) \tag{A.90}$$
$$\wedge\ \mathit{enabled}(a, s_0 \circ r_0) \Rightarrow (s_0 \circ r_0, a[s_0 \circ r_0]) \in \mathcal{G}(\kappa)) \wedge \tag{A.91}$$
$$\forall\kappa.\,\forall a \in (\mathcal{I} \cup \mathcal{I}_2)\,(\kappa).\,\mathit{potential}(a, s_0 \circ r_0) \Rightarrow$$
$$\mathit{reflected}(a, s_0 \circ r_0, \mathcal{I}_0(\kappa)) \vee$$
$$\neg \mathit{visible}(a, s_0) \wedge \forall (s', r') \in a[s_0, r_0].\ (\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right) \tag{A.92}$$

RTS. (A.90)
Pick an arbitrary κ, $a \in \mathcal{I}_0(\kappa)$ and (s', r') such that

$$\mathit{potential}(a, s_0 \circ r_0) \tag{A.93}$$
$$(s', r') \in a[s_0, r_0] \tag{A.94}$$

Then from (A.88) and (A.93)-(A.94) we have:

$$(\mathcal{I}, \mathcal{G})\!\downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right) \tag{A.95}$$

From (A.94) and the definition of $a[s_0, r_0]$, we know that $\mathit{fst}\,(\Delta a) \leq s_0 \circ r_0$ and consequently from (A.89) we have $\mathit{fst}\,(\Delta a) \leq s_1 \circ r_1$. Thus from Lemma 24 we know there exists $p_s, p_r \in \mathsf{LState}$ such that :

$$\mathit{fst}\,(\Delta a) = p_s \circ p_r \wedge p_s \leq s_1 \wedge p_r \leq r_1$$

From the definition of $a[s_1, r_1]$ we then have

$$(p_s = \mathbf{0}_{\mathsf{L}} \wedge (s_1, \mathit{snd}\,(\Delta a) \circ (r_1 - p_r)) \in a[s_1, r_1])$$
$$\vee (p_s > \mathbf{0}_{\mathsf{L}} \wedge (\mathit{snd}\,(\Delta a) \circ (s_1 - p_s), r_1 - p_r) \in a[s_1, r_1])$$

That is, there exists $s'', r'' \in \mathsf{LState}$ such that

$$(s'', r'') \in a[s_1, r_1] \tag{A.96}$$

From (A.87), (A.96) and Lemma 11 we have:

$$\mathcal{I}_1 \sqsubseteq^{\{s''\}} \mathcal{I}_2 \tag{A.97}$$

From (A.89), (A.94), (A.96) and Lemma 14 we have:

$$s' \circ r' = s'' \circ r'' \tag{A.98}$$

Since $a \in \mathcal{I}_0(\kappa)$ and $\mathcal{I}_0 \subseteq \mathcal{I}$, we know $a \in \mathcal{I}(\kappa)$; consequently, from (A.96) and (A.86) we have:

$$\begin{aligned}&\mathit{reflected}(a, s_1 \circ r_1, \mathcal{I}_1(\kappa)) \vee \\ &\neg \mathit{visible}(a, s_1) \wedge \forall (s'', r'') \in a[s_1, r_1].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s'', r'', \mathcal{I}_1\right)\end{aligned}$$

There are two cases to consider:
Case 1. $\neg \mathit{visible}(a, s_1) \wedge \forall (s'', r'') \in a[s_1, r_1].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} (s'', r'', \mathcal{I}_1)$
From the assumption of the case and (A.96) we have

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s'', r'', \mathcal{I}_1\right) \tag{A.99}$$

Consequently, from (A.95), (A.97), (A.98), (A.99) and (I.H) we have:

$$(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)$$

as required.

Case 2. $\mathit{reflected}(a, s_1 \circ r_1, \mathcal{I}_1(\kappa))$
From (A.89) and (A.93) we have $\mathit{potential}(a, s_1 \circ r_1)$ and consequently from the definition of *potential* we know there exists $l \in \mathsf{LState}$ such that $\mathit{fst}\,(a) \leq s_1 \circ r_1 \circ l$. Thus from the assumption of the case and the definition of *reflected* we have:

$$\exists a' \in \mathcal{I}_1(\kappa).\, \Delta a = \Delta a' \wedge \mathit{fst}\left(a'\right) \leq s_1 \circ r_1 \circ l \tag{A.100}$$

From (A.89) and (A.93) we have $\mathit{potential}(a, s_1 \circ r_1)$. Consequently, from (A.100) and the definition of *potential* we have:

$$\mathit{potential}(a', s_1 \circ r_1) \tag{A.101}$$

On the other hand, from (A.96), (A.100) and the definition of $a'[s_1, r_1]$ we know $(s'', r'') \in a'[s_1, r_1]$. Thus from (A.86), (A.100) and (A.101) we have:

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s'', r'', \mathcal{I}_1\right) \tag{A.102}$$

Finally, from (A.95), (A.97), (A.98), (A.102) and (I.H) we have:

$$(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)$$

as required.

RTS. (A.91)
Pick an arbitrary κ and $a \in \mathcal{I}_0(\kappa)$ such that

$$\mathit{enabled}(a, s_0 \circ r_0)$$

Then from (A.88) we have:

$$(s_0 \circ r_0, a[s_0 \circ r_0]) \in \mathcal{G}(\kappa)$$

as required.

RTS. (A.92)
Pick an arbitrary κ and $a \in (\mathcal{I} \cup \mathcal{I}_2)(\kappa)$ such that

$$\mathit{potential}(a, s_0 \circ r_0) \tag{A.103}$$

There are two cases to consider:

Case 1. $a \in \mathcal{I}(\kappa)$
Then from (A.88) and assumption of case 1. we have:

$$\begin{array}{l}\mathit{reflected}(a, s_0 \circ r_0, \mathcal{I}_0(\kappa)) \vee \\ \neg \mathit{visible}(a, s_0) \wedge \forall (s', r') \in a[s_0, r_0].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} (s', r', \mathcal{I}_0)\end{array}$$

In the case of the first disjunct the desired result holds trivially. On the other hand, in the case of the second disjunct we have:

$$\neg \mathit{visible}(a, s_0) \wedge \forall (s', r') \in a[s_0, r_0].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right) \tag{A.104}$$

Pick an arbitrary s', r' such that

$$(s', r') \in a[s_0, r_0] \tag{A.105}$$

Then from (A.88) and (A.103)-(A.105) we have:

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right) \tag{A.106}$$

From (A.105) and the definition of $a[s_0, r_0]$, we know that $\mathit{fst}\,(\Delta a) \leq s_0 \circ r_0$ and consequently from (A.89) we have $\mathit{fst}\,(\Delta a) \leq s_1 \circ r_1$. Thus from Lemma 24 we know there exists $p_s, p_r \in \mathsf{LState}$ such that :

$$\mathit{fst}\,(\Delta a) = p_s \circ p_r \wedge p_s \leq s_1 \wedge p_r \leq r_1$$

From the definition of $a[s_1, r_1]$ we then have

$$(p_s = \mathbf{0}_{\mathsf{L}} \wedge (s_1, \mathit{snd}\,(\Delta a) \circ (r_1 - p_r)) \in a[s_1, r_1])$$
$$\vee (p_s > \mathbf{0}_{\mathsf{L}} \wedge (\mathit{snd}\,(\Delta a) \circ (s_1 - p_s), r_1 - p_r) \in a[s_1, r_1])$$

That is, there exists $s'', r'' \in \mathsf{LState}$ such that

$$(s'', r'') \in a[s_1, r_1] \tag{A.107}$$

From (A.87), (A.107) and Lemma 11 we have:

$$\mathcal{I}_1 \sqsubseteq^{\{s''\}} \mathcal{I}_2 \tag{A.108}$$

From (A.89), (A.105), (A.107) and Lemma 14 we have:

$$s' \circ r' = s'' \circ r'' \tag{A.109}$$

Since $a \in \mathcal{I}(\kappa)$ (assumption of case 1), from (A.107) and (A.86) we have:

$$\mathit{reflected}(a, s_1 \circ r_1, \mathcal{I}_1(\kappa)) \vee$$
$$\neg \mathit{visible}(a, s_1) \wedge \forall (s'', r'') \in a[s_1, r_1].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s'', r'', \mathcal{I}_1\right)$$

There are two cases to consider:
Case 1. $\neg \mathit{visible}(a, s_1) \wedge \forall (s'', r'') \in a[s_1, r_1].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} (s'', r'', \mathcal{I}_1)$
From the assumption of the case and (A.107) we have

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s'', r'', \mathcal{I}_1\right) \tag{A.110}$$

Consequently, from (A.106), (A.108), (A.109), (A.110) and (I.H) we have:

$$(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)$$

as required.

Case 2. $\mathit{reflected}(a, s_1 \circ r_1, \mathcal{I}_1(\kappa))$
From (A.89) and (A.103) we have $\mathit{potential}(a, s_1 \circ r_1)$ and consequently from the definition of *potential* we know there exists $l \in \mathsf{LState}$ such that $\mathit{fst}\,(a) \leq s_1 \circ r_1 \circ l$. Thus from the assumption of the case and the definition of *reflected* we have:

$$\exists a' \in \mathcal{I}_1(\kappa).\ \Delta a = \Delta a' \wedge \mathit{fst}\left(a'\right) \leq s_1 \circ r_1 \circ l \tag{A.111}$$

From (A.89) and (A.103) we have $potential(a, s_1 \circ r_1)$. Consequently, from (A.111) and the definition of *potential* we have:

$$potential(a', s_1 \circ r_1) \tag{A.112}$$

On the other hand, from (A.107), (A.111) and the definition of $a'[s_1, r_1]$ we know $(s'', r'') \in a'[s_1, r_1]$. Thus from (A.86), (A.111) and (A.112) we have:

$$(\mathcal{I}, \mathcal{G})\downarrow_{(n-1)} \left(s'', r'', \mathcal{I}_1\right) \tag{A.113}$$

Finally, from (A.106), (A.108), (A.109), (A.113) and (I.H) we have:

$$(\mathcal{I} \cup \mathcal{I}_2, \mathcal{G})\downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)$$

as required. □

Lemma 14 (action-application)**.** For all $a \in \mathsf{LState} \times \mathsf{LState}$ and s_1 , r_1, s_2, r_2, s_1', r_1', s_2', $r_2' \in \mathsf{LState}$,

$$s_1 \circ r_1 = s_2 \circ r_2 \wedge (s_1', r_1') \in a[s_1, r_1] \wedge (s_2', r_2') \in a[s_2, r_2] \implies s_1' \circ r_1' = s_2' \circ r_2'$$

Proof. Take arbitrary $a \in \mathsf{LState} \times \mathsf{LState}$ and s_1 , r_1, s_2, r_2, s_1', r_1', s_2', $r_2' \in \mathsf{LState}$ such that

$$s_1 \circ r_1 = s_2 \circ r_2 \tag{A.114}$$
$$(s_1', r_1') \in a[s_1, r_1] \tag{A.115}$$
$$(s_2', r_2') \in a[s_2, r_2] \tag{A.116}$$

Then from (A.115), and the definitions of $a[s_1, r_1]$ and $a[s_1 \circ r_1]$ we have:

$$a[s_1 \circ r_1] = s_1' \circ r_1' \tag{A.117}$$

Similarly, from (A.116) we have:

$$a[s_2 \circ r_2] = s_2' \circ r_2' \tag{A.118}$$

Finally, from (A.114), (A.117) and (A.118) we have:

$$s_1' \circ r_1' = s_2' \circ r_2'$$

as required. □

Lemma 15 (EXTEND-Closure-1)**.** For all $\mathcal{I}, \mathcal{I}_e, \mathcal{I}_0 \in \mathsf{AMod}$ such that $\forall \kappa \in \mathsf{dom}\,(\mathcal{I}_0)\,.\,\mathcal{I}_0(\kappa) = \emptyset$; and for all $g, s_e \in \mathsf{LState}$ and $\mathcal{F}, \mathcal{F}_e \in \mathcal{P}\,(\mathsf{LState})$,

$$g \in \mathcal{F} \wedge \mathcal{F} \blacktriangleright \mathcal{I} \wedge s_e \in \mathcal{F}_e \wedge \mathcal{F}_e \blacktriangleright \mathcal{I}_e \cup \mathcal{I}_0 \implies (\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, (\mathcal{I}, \mathcal{F}) + (\mathcal{I}_e \cup \mathcal{I}_0, \mathcal{F}_e))\!\downarrow (s_e, g, \mathcal{I}_e)$$

where

$$((\mathcal{I}_1, \mathcal{F}_1) + (\mathcal{I}_2, \mathcal{F}_2))\,(\kappa) \stackrel{\text{def}}{=} \left\{ (f', a[f']) \;\middle|\; \begin{array}{l} (a \in \mathcal{I}_1(\kappa) \vee a \in \mathcal{I}_2(\kappa)) \wedge \\ f' \in \mathcal{F}_1 \circ \mathcal{F}_2 \wedge \mathit{enabled}(a, f') \end{array} \right\}$$

and

$$\mathcal{F}_1 \circ \mathcal{F}_2 \stackrel{\text{def}}{=} \{f_1 \circ f_2 \mid f_1 \in \mathcal{F}_1 \wedge f_2 \in \mathcal{F}_2\}$$

Proof. Pick an arbitrary $\mathcal{I}, \mathcal{I}_e, \mathcal{I}_0 \in \mathsf{AMod}$, $g, s_e \in \mathsf{LState}$ and $\mathcal{F}, \mathcal{F}_e \in \mathcal{P}\,(\mathsf{LState})$ such that

$$\forall \kappa \in \mathsf{dom}\,(\mathcal{I}_0)\,.\,\mathcal{I}_0(\kappa) = \emptyset \tag{A.119}$$
$$g \in \mathcal{F} \wedge s_e \in \mathcal{F}_e \tag{A.120}$$
$$\mathcal{F} \blacktriangleright \mathcal{I} \wedge \mathcal{F}_e \blacktriangleright \mathcal{I}_e \cup \mathcal{I}_0 \tag{A.121}$$

From the definition of $\downarrow$, it then suffices to show

$$\mathcal{I}_e \subseteq \mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0 \tag{A.122}$$
$$\forall n \in \mathbb{N}.\ (\mathcal{I} \cup \mathcal{I}_e, (\mathcal{I}, \mathcal{F}) + (\mathcal{I}_e, \mathcal{F}_e))\!\downarrow_n (s_e, g, \mathcal{I}_e) \tag{A.123}$$

RTS. (A.122)
This holds trivially from the definition of $\mathcal{I} \cup \mathcal{I}_e$.

RTS. (A.123)
Rather than proving (A.123) directly, we first establish the following.

$$\forall n \in \mathbb{N}.\, \forall g, s_e \in \mathsf{LState}.\ g \in \mathcal{F} \wedge s_e \in \mathcal{F}_e \implies (\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, (\mathcal{I}, \mathcal{F}) + (\mathcal{I}_e \cup \mathcal{I}_0, \mathcal{F}_e))\!\downarrow_n (s_e, g, \mathcal{I}_e) \tag{A.124}$$

We can then despatch (A.123) from (A.120) and (A.124); since for an arbitrary $n \in \mathbb{N}$, from (A.120) and (A.124) we have $(\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, (\mathcal{I}, g) + (\mathcal{I}_e \cup \mathcal{I}_0, s_e))\!\downarrow_n (s_e, g, \mathcal{I}_e)$ as required.

RTS. (A.124)
Let $\mathcal{G} = (\mathcal{I}, \mathcal{F}) + (\mathcal{I}_e \cup \mathcal{I}_0, \mathcal{F}_e)$. We proceed by induction on the number of

steps n.

Base case $n = 0$
Pick an arbitrary $g, s_e \in \mathsf{LState}$. We are then required to show $(\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, \mathcal{G})\downarrow_0 (s_e, g, \mathcal{I}_e)$ which follows trivially from the definition of $\downarrow_0$.

Inductive case
Pick an arbitrary $n \in \mathbb{N}$ and $g, s_e \in \mathsf{LState}$ such that

$$g \in \mathcal{F} \tag{A.125}$$
$$s_e \in \mathcal{F}_e \tag{A.126}$$
$$\forall g', s'_e.\, g' \in \mathcal{F} \wedge s'_e \in \mathcal{F}_e \implies (\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, \mathcal{G})\downarrow_{(n-1)} \left(s'_e, g', \mathcal{I}_e\right) \tag{I.H}$$

RTS.

$$\begin{aligned}
&\forall \kappa.\, \forall a \in \mathcal{I}_e(\kappa).\\
&\quad (\mathit{potential}(a, s_e \circ g) \Rightarrow\\
&\qquad \forall (s', r') \in a[s_e, g].\ (\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, \mathcal{G})\downarrow_{(n-1)} \left(s', r', \mathcal{I}_e\right)) \qquad \text{(A.127)}\\
&\quad \mathit{enabled}(a, s_e \circ g) \Rightarrow (s_e \circ g, a[s_e \circ g]) \in \mathcal{G}(\kappa)) \qquad \text{(A.128)}\\
&\forall \kappa.\, \forall a \in (\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0)\,(\kappa).\, \mathit{potential}(a, s_e \circ g) \Rightarrow\\
&\ \mathit{reflected}(a, s_e \circ g, \mathcal{I}_e(\kappa)) \vee\\
&\ \neg \mathit{visible}(a, s_e) \wedge \forall (s', r') \in a[s_e, g].\ (\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, \mathcal{G})\downarrow_{(n-1)} \left(s', r', \mathcal{I}_e\right)
\end{aligned} \tag{A.129}$$

RTS. A.127
Pick an arbitrary κ, $a \in \mathcal{I}_e(\kappa)$ and (s', r') such that

$$\mathit{potential}(a, s_e \circ g) \tag{A.130}$$
$$(s', r') \in a[s_e, g] \tag{A.131}$$

Since from (A.130) and the definition of *potential* we have $s_e \circ g \sqcap \mathit{fst}\,(a) \neq \emptyset$ and consequently, $s_e \sqcap \mathit{fst}\,(a) \neq \emptyset$, from (A.126) we have:

$$\mathit{fst}\,(\Delta a) \leq s_e \wedge \mathit{fst}\,(\Delta a) \perp g \tag{A.132}$$

From (A.130), (A.132) and the definition of *potential* we have:

$$\mathit{potential}(a, s_e) \tag{A.133}$$

On the other hand, from (A.131), (A.132) and the definitions of $a[s_e, g]$ and $\perp$, we have:

$$r' = g \tag{A.134}$$
$$a[s_e] = s' \tag{A.135}$$

Consequently, from (A.121), (A.126), (A.133), (A.135) and the definition of $\blacktriangleright$ we have:

$$s' \in \mathcal{F}_e \tag{A.136}$$

Finally, from (A.125), (A.136) and (I.H) we have:

$$(\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, \mathcal{G}) \downarrow_{(n-1)} \left(s', g, \mathcal{I}_e\right)$$

and consequently from (A.134)

$$(\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_e\right)$$

as required.

RTS. A.128
Pick an arbitrary κ, $a \in \mathcal{I}_e(\kappa)$ such that

$$\mathit{enabled}(a, s_e \circ g) \tag{A.137}$$

Then from (A.125)-(A.126) and the definition of $(\mathcal{I}, \mathcal{F}) + (\mathcal{I}_e, \mathcal{F}_e)$ we have:

$$(s_e \circ g, a[s_e \circ g]) \in \mathcal{G}$$

as required.

RTS. A.129
Pick an arbitrary κ, $a \in (\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0)(\kappa)$ such that

$$\mathit{potential}(a, s_e \circ g) \tag{A.138}$$

Since from (A.119) we know $\mathcal{I}_0(\kappa) = \emptyset$ we know $a \in \mathcal{I}_e(\kappa) \vee a \in \mathcal{I}(\kappa)$, and thus there are two cases to consider.

If the first disjunct is the case, the desired result holds trivially since from the definition of *reflected* and (A.138) we have $\mathit{reflected}(a, s_e \circ g, \mathcal{I}_e(\kappa))$ as required.

On the other hand, if the second disjunct is the case, then since from (A.138) and the definition of *potential* we have $s_e \circ g \sqcap \mathit{fst}\,(a) \neq \emptyset$ and consequently, $g \sqcap \mathit{fst}\,(a) \neq \emptyset$, from (A.125) we have:

$$\mathit{fst}\,(\Delta a) \leq g \wedge \mathit{fst}\,(\Delta a) \perp s_e \tag{A.139}$$

From (A.138), (A.139) and the definition of *potential* we have:

$$\mathit{potential}(a, g) \tag{A.140}$$

On the other hand, from (A.139) and the definitions of $a[s_e \circ g]$ and $\perp$, we know there exists g' such that:

$$a[s_e \circ g] = s_e \circ g' \tag{A.141}$$
$$a[g] = g' \tag{A.142}$$

Consequently, from (A.121), (A.125), (A.140), (A.142) and the definition of $\blacktriangleright$ we have:

$$g' \in \mathcal{F} \tag{A.143}$$

Finally, from (A.126), (A.143), (I.H) we have:

$$(\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, \mathcal{G})\!\downarrow_{(n-1)} \left(s_e, g', \mathcal{I}_e\right)$$

and consequently from (A.141)

$$(\mathcal{I} \cup \mathcal{I}_e \cup \mathcal{I}_0, \mathcal{G})\!\downarrow_{(n-1)} (s_e, a[s_e \circ g] - s_e, \mathcal{I}_e)$$

as required. □

Lemma 16 (EXTEND-closure-2)**.** For all $\mathcal{I}_0, \mathcal{I}, \mathcal{I}_e, \mathcal{G} \in \mathsf{AMod}$, $s, g, s_e \in \mathsf{LState}$ and $\mathcal{F}, \mathcal{F}_e \in \mathcal{P}\,(\mathsf{LState})$

$$\begin{array}{c} g \in \mathcal{F} \wedge \mathcal{F} \blacktriangleright \mathcal{I} \wedge s_e \in \mathcal{F}_e \wedge \mathcal{F}_e \blacktriangleright \mathcal{I}_e \wedge (\mathcal{I}, \mathcal{G})\!\downarrow (s, g - s, \mathcal{I}_0) \\ \implies (\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}')\!\downarrow (s, (g - s) \circ s_e, \mathcal{I}_0) \end{array}$$

where $\mathcal{G}' = (\mathcal{I}, \mathcal{F}) + (\mathcal{I}_e, \mathcal{F}_e)$ and

$$((\mathcal{I}, \mathcal{F}) + (\mathcal{I}_e, \mathcal{F}_e))\,(\kappa) \stackrel{\text{def}}{=} \left\{ (f', a[f']) \;\middle|\; \begin{array}{l} (a \in \mathcal{I}(\kappa) \vee a \in \mathcal{I}_e(\kappa)) \wedge \\ f' \in \mathcal{F} \circ \mathcal{F}_e \wedge \mathit{enabled}(a, f') \end{array} \right\}$$

and

$$\mathcal{F} \circ \mathcal{F}_e \stackrel{\text{def}}{=} \{f \circ f_e \mid f \in \mathcal{F} \wedge f_e \in \mathcal{F}_e\}$$

Proof. Pick an arbitrary $\mathcal{I}_0, \mathcal{I}, \mathcal{I}_e, \mathcal{G} \in \mathsf{AMod}$, $s, g, s_e \in \mathsf{LState}$ and $\mathcal{F}, \mathcal{F}_e \in \mathcal{P}(\mathsf{LState})$ such that

$$g \in \mathcal{F} \wedge s_e \in \mathcal{F}_e \tag{A.144}$$
$$\mathcal{F} \blacktriangleright \mathcal{I} \wedge \mathcal{F}_e \blacktriangleright \mathcal{I}e \tag{A.145}$$
$$(\mathcal{I}, \mathcal{G}) \downarrow (s, g - s, \mathcal{I}_0) \tag{A.146}$$

From the definition of $\downarrow$, it then suffices to show

$$\mathcal{I}_0 \subseteq \mathcal{I} \cup \mathcal{I}_2 \tag{A.147}$$
$$\forall n \in \mathbb{N}.\ \left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_n (s, (g - s) \circ s_e, \mathcal{I}_0) \tag{A.148}$$

RTS. (A.147)
Since from (A.146) and the definition of $\downarrow$ we have $\mathcal{I}_0 \subseteq \mathcal{I}$, we can consequently derive $\mathcal{I}_0 \subseteq \mathcal{I} \cup \mathcal{I}_e$ as required.

RTS. (A.148)
Rather than proving (A.148) directly, we first establish the following.

$$\begin{aligned} &\forall n \in \mathbb{N}.\, \forall s, g, s_e \in \mathsf{LState}. \\ &\quad g \in \mathcal{F} \wedge s_e \in \mathcal{F}_e \wedge (\mathcal{I}, \mathcal{G}) \downarrow_n (s, g - s, \mathcal{I}_0) \implies \\ &\qquad \left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_n (s, (g - s) \circ s_e, \mathcal{I}_0) \end{aligned} \tag{A.149}$$

We can then despatch (A.148) from (A.144)-(A.146) and (A.149); since for an arbitrary $n \in \mathbb{N}$, from (A.146) and the definition of $\downarrow$ we have $(\mathcal{I}, \mathcal{G}) \downarrow_n (s, g - s, \mathcal{I}_0)$ and consequently from (A.144) and (A.149) we derive $(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}') \downarrow_n (s, (g - s) \circ s_e, \mathcal{I}_0)$ as required.

RTS. (A.149)
Pick an arbitrary $g', s'_e \in \mathsf{LState}$ such that $g' \copyright \mathcal{I} \wedge s'_e \copyright \mathcal{I}_e$ and let $\mathcal{G}' = (\mathcal{I}, g') + (\mathcal{I}_e, s'_e)$. We proceed by induction on the number of steps n.

Base case $n = 0$
Pick an arbitrary $s, g, s_e \in \mathsf{LState}$. We are then required to show $(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}') \downarrow_0 (s, (g - s) \circ s_e, \mathcal{I}_0)$ which follows trivially from the definition of $\downarrow_0$.

Inductive case

Pick an arbitrary $n \in \mathbb{N}$ and $s, r, g, s_e \in \mathsf{LState}$ such that

$$g \in \mathcal{F} \tag{A.150}$$
$$s_e \in \mathcal{F}_e \tag{A.151}$$
$$g = s \circ r \tag{A.152}$$
$$(\mathcal{I}, \mathcal{G}) \downarrow_n (s, r, \mathcal{I}_0) \tag{A.153}$$
$$\forall s'', g'', s''_e.\, g'' \in \mathcal{F} \wedge s''_e \in \mathcal{F}_e \wedge (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s'', g'' - s'', \mathcal{I}_0\right)$$
$$\implies \left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_{(n-1)} \left(s'', (g'' - s'') \circ s''_e, \mathcal{I}_0\right) \tag{I.H}$$

RTS.

$$\forall \kappa.\, \forall a \in \mathcal{I}_0(\kappa).$$
$$(\mathit{potential}(a, g \circ s_e) \Rightarrow$$
$$\forall (s', r') \in a[s, r \circ s_e].\, \left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)) \tag{A.154}$$
$$(\mathit{enabled}(a, g \circ s_e) \Rightarrow (g \circ s_e, a[g \circ s_e]) \in \mathcal{G}'(\kappa)) \tag{A.155}$$
$$\forall \kappa.\, \forall a \in (\mathcal{I} \cup \mathcal{I}_e)\,(\kappa).\, \mathit{potential}(a, g \circ s_e) \Rightarrow$$
$$\mathit{reflected}(a, g \circ s_e, \mathcal{I}_0(\kappa)) \vee$$
$$\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r \circ s_e].\, \left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right) \tag{A.156}$$

RTS. (A.154)
Pick an arbitrary κ, $a \in \mathcal{I}_0(\kappa)$ and (s', r') such that

$$\mathit{potential}(a, g \circ s_e) \tag{A.157}$$
$$(s', r') \in a[s, r \circ s_e] \tag{A.158}$$

Since from (A.157) and the definition of *potential* we have $s_e \circ g \sqcap \mathit{fst}\,(a) \neq \emptyset$ and consequently, $g \sqcap \mathit{fst}\,(a) \neq \emptyset$, from (A.150) we have:

$$\mathit{fst}\,(\Delta a) \leq g \wedge \mathit{fst}\,(\Delta a) \perp s_e \tag{A.159}$$

From (A.157), (A.159) and the definition of *potential* we have:

$$\mathit{potential}(a, g) \tag{A.160}$$

On the other hand, from (A.158), (A.159) and the definitions of $a[s, r \circ s_e]$ and $\perp$, we know there exists r'':

$$r' = r'' \circ s_e \tag{A.161}$$
$$(s', r'') \in a[s, r] \tag{A.162}$$

From (A.162) and the definitions of $a[s, r]$ and $a[s \circ r]$, we know $s' \circ r'' = a[s \circ r]$. Consequently, from (A.145), (A.150), (A.160), the definition of $\blacktriangleright$ and since $g = s \circ r$ (A.152), we have:

$$s' \circ r'' \in \mathcal{F} \tag{A.163}$$

On the other hand, from (A.153), (A.160), (A.157) and (A.162) we have:

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r'', \mathcal{I}_0\right) \tag{A.164}$$

Finally, from (A.151), (A.163), (A.164) and (I.H) we have:

$$\left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_{(n-1)} \left(s', r'' \circ s_e, \mathcal{I}_0\right)$$

and consequently from (A.161)

$$\left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)$$

as required.

RTS. (A.155)
Pick an arbitrary κ, $a \in \mathcal{I}_0(\kappa)$ such that

$$\mathit{enabled}(a, g \circ s_e) \tag{A.165}$$

Then from (A.150)-(A.151) and the definition of $(\mathcal{I}, \mathcal{F}) + (\mathcal{I}_e, \mathcal{F})$ we have:

$$(g \circ s_e, a[g \circ s_e]) \in \mathcal{G}'$$

as required.

RTS. (A.156)
Pick an arbitrary κ, $a \in (\mathcal{I} \cup \mathcal{I}_e)(\kappa)$ such that

$$\mathit{potential}(a, g \circ s_e) \tag{A.166}$$

Since $a \in \mathcal{I}_e(\kappa) \vee a \in \mathcal{I}(\kappa)$, there are two cases to consider.

Case 1. $a \in \mathcal{I}_e(\kappa)$
Since from (A.166) and the definition of *potential* we have $g \circ s_e \sqcap \mathit{fst}\,(a) \neq \emptyset$ and consequently, $s_e \sqcap \mathit{fst}\,(a) \neq \emptyset$, from (A.151) and the assumption of the case we have:

$$\mathit{fst}\,(\Delta a) \leq s_e \wedge \mathit{fst}\,(\Delta a) \perp g \tag{A.167}$$

and consequently from the definition of *visible* and (A.152) we have

$$\neg \mathit{visible}(a, s) \tag{A.168}$$

From (A.166), (A.167) and the definition of *potential* we have:

$$\mathit{potential}(a, s_e) \tag{A.169}$$

Pick an arbitrary $s', r' \in \mathsf{LState}$ such that

$$(s', r') \in a[s, r \circ s_e] \tag{A.170}$$

From (A.152), (A.167) and the definitions of $a[s, r \circ s_e]$ and $\bot$, we know there exists s'_e such that:

$$s' = s \wedge r' = r \circ s'_e \tag{A.171}$$
$$a[s_e] = s'_e \tag{A.172}$$

Consequently, from (A.145), (A.151), (A.169), (A.172) and the definition of $\blacktriangleright$ we have:

$$s'_e \in \mathcal{F}_e \tag{A.173}$$

From (A.152), (A.171) and Lemma 17 we have:

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} (s', r, \mathcal{I}_0) \tag{A.174}$$

From (A.150), (A.173), (A.174), (I.H) we have:

$$(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}') \downarrow_{(n-1)} (s', r \circ s'_e, \mathcal{I}_0)$$

and thus from (A.171)

$$(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}') \downarrow_{(n-1)} (s', r', \mathcal{I}_0) \tag{A.175}$$

Finally, from (A.168), (A.170) and (A.171) we have:

$$\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r \circ s_e].\ (\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}') \downarrow_{(n-1)} (s', r', \mathcal{I}_0)$$

as required.

Case 2. $a \in \mathcal{I}(\kappa)$
Since from (A.166) and the definition of *potential* we have $g \circ s_e \sqcap \mathit{fst}\,(a) \neq \emptyset$

and consequently, $g \sqcap \mathit{fst}\,(a) \neq \emptyset$, from (A.150) and the assumption of the case we have:

$$\mathit{fst}\,(\Delta a) \leq g \wedge \mathit{fst}\,(\Delta a) \perp s_e \tag{A.176}$$

From (A.166), (A.176) and the definition of *potential* we have:

$$\mathit{potential}(a, g) \tag{A.177}$$

Consequently, from (A.153), the assumption of the case we have:

$$\begin{array}{l} \mathit{reflected}(a, g, \mathcal{I}_0(\kappa)) \vee \\ \neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right) \end{array}$$

There are two cases to consider:

Case 2.1. $\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} (s', r', \mathcal{I}_0)$
Pick an arbitrary $s', r' \in \mathsf{LState}$ such that

$$(s', r') \in a[s, r \circ s_e] \tag{A.178}$$

From (A.176), the assumption of case and the definitions of $a[s, r \circ s_e]$, $\perp$ and *visible*, we know there exists r'' such that:

$$s = s' \wedge r' = r'' \circ s_e \wedge (s', r'') \in a[s, r] \tag{A.179}$$

From (A.145), (A.150), (A.152), (A.177), (A.179), and the definition of $\blacktriangleright$ we have:

$$s' \circ r'' \in \mathcal{F} \tag{A.180}$$

On the other hand, from (A.179) and the assumption of the case we have:

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r'', \mathcal{I}_0\right) \tag{A.181}$$

Finally, from (A.151), (A.180), (A.181), (A.179) and (I.H) we have:

$$\left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)$$

and thus from the assumption of the case and (A.178) we have

$$\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r \circ s_e].\ \left(\mathcal{I} \cup \mathcal{I}_e, \mathcal{G}'\right) \downarrow_{(n-1)} \left(s', r', \mathcal{I}_0\right)$$

as required.

Case 2.2. *reflected*$(a, g, \mathcal{I}_0(\kappa))$
Pick an arbitrary $l \in \mathsf{LState}$ such that

$$\mathit{fst}\,(a) \leq g \circ s_e \circ l \tag{A.182}$$

From the assumption of the case and the definition of *reflected* we then have

$$\exists a' \in \mathcal{I}_0(\kappa).\, \Delta a' = \Delta a \wedge \mathit{fst}\,\left(a'\right) \leq g \circ s_e \circ l \tag{A.183}$$

Thus from (A.182), (A.183) and the definition of *reflected* we have:

$$\mathit{reflected}(a, g \circ s_e, \mathcal{I}_0(\kappa))$$

as required. □

Lemma 17. For all $\mathcal{I}, \mathcal{G}, \mathcal{I}' \in \mathsf{AMod}$ and $n \in \mathbb{N}^+$

$$\forall s, r \in \mathsf{LState}.\ (\mathcal{I}, \mathcal{G}){\downarrow_n}\ \left(s, r, \mathcal{I}'\right) \implies (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s, r, \mathcal{I}'\right)$$

Proof. Pick an arbitrary $s, r \in \mathsf{LState}$ and proceed by induction on number of steps n.

Base case: n = 1
Pick an arbitrary $s, r \in \mathsf{LState}$. We are then required to show $(\mathcal{I}, \mathcal{G}){\downarrow_0}\ (s, r, \mathcal{I}')$ which trivially follows from the definition of $\downarrow_0$.

Inductive case
Pick an arbitrary $s, r \in \mathsf{LState}$ such that

$$(\mathcal{I}, \mathcal{G}){\downarrow_{(n+1)}}\ \left(s, r, \mathcal{I}'\right) \tag{A.184}$$

$$\forall s', r' \in \mathsf{LState}.\ (\mathcal{I}, \mathcal{G}){\downarrow_n}\ \left(s', r', \mathcal{I}'\right) \implies (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s', r', \mathcal{I}'\right) \tag{I.H}$$

RTS.

$$\forall \kappa.\, \forall a \in \mathcal{I}'(\kappa).$$

$$(\mathit{potential}(a, s \circ r) \Rightarrow \forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s', r', \mathcal{I}'\right)) \tag{A.185}$$

$$(\mathit{enabled}(a, s \circ r) \Rightarrow (s \circ r, a[s \circ r]) \in \mathcal{G}(\kappa)) \tag{A.186}$$

$$\forall \kappa.\, \forall a \in \mathcal{I}(\kappa).\, \mathit{potential}(a, s \circ r) \Rightarrow$$
$$\mathit{reflected}(a, s \circ r, \mathcal{I}'(\kappa)) \vee$$
$$\neg \mathit{visible}(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}){\downarrow_{(n-1)}}\ \left(s', r', \mathcal{I}'\right) \tag{A.187}$$

RTS. (A.185)
Pick an arbitrary κ, $a \in \mathcal{I}'(\kappa)$ and (s', r') such that

$$potential(a, s \circ r) \tag{A.188}$$
$$(s', r') \in a[s, r] \tag{A.189}$$

Then from (A.188)-(A.189) and (A.184) we have:

$$(\mathcal{I}, \mathcal{G}) \downarrow_n \left(s', r', \mathcal{I}'\right)$$

and thus from (I.H)

$$(\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right)$$

as required.

RTS. (A.186)
Pick an arbitrary κ and $a \in \mathcal{I}'(\kappa)$ such that

$$enabled(a, s \circ r) \tag{A.190}$$

Then from (A.184) and (A.190) we have:

$$(s \circ r, a[s \circ r]) \in \mathcal{G}(\kappa)$$

as required.

RTS. (A.187)
Pick an arbitrary κ and $a \in \mathcal{I}(\kappa)$ such that

$$potential(a, s \circ r) \tag{A.191}$$

Then from (A.5) we have:

$$\begin{aligned} &reflected(a, s \circ r, \mathcal{I}'(\kappa)) \vee \\ &\neg visible(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}) \downarrow_n \left(s', r', \mathcal{I}'\right) \end{aligned}$$

and consequently from (I.H.)

$$\begin{aligned} &reflected(a, s \circ r, \mathcal{I}'(\kappa)) \vee \\ &\neg visible(a, s) \wedge \forall (s', r') \in a[s, r].\ (\mathcal{I}, \mathcal{G}) \downarrow_{(n-1)} \left(s', r', \mathcal{I}'\right) \end{aligned}$$

as required.

□

Lemma 18 (Sequential command soundness)**.** For all $\mathbb{S} \in \mathsf{Seqs}$, $(M_1, \mathbb{S}, M_2) \in \mathrm{Ax}_{\mathsf{S}}$ and $m \in \mathbb{M}$:

$$[\![\mathbb{S}]\!]_{\mathsf{S}} (\lfloor M_1 \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) \subseteq \lfloor M_2 \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}$$

Proof. By induction over the structure of $\mathbb{S}$. Pick an arbitrary $m \in \mathbb{M}$.

Case $\mathbb{E}$
This follows immediately from parameter 10.

Case skip
RTS.

$$[\![\mathbb{S}]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) \subseteq \lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}$$

Proof.

$$\begin{array}{rll} [\![\mathbb{S}]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) = & \lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}} \\ \subseteq & \lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}} \end{array}$$

as required.

Case $\mathbb{S}_1; \mathbb{S}_2$
RTS.

$$[\![\mathbb{S}_1; \mathbb{S}_2]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) \subseteq \left\lfloor M' \bullet_{\mathbb{M}} \{m\} \right\rfloor_{\mathbb{M}}$$

where $(M, \mathbb{S}_1, M''), (M'', \mathbb{S}_2, M') \in \mathrm{Ax}_{\mathsf{S}}$.

Proof.

$$\begin{array}{rll} [\![\mathbb{S}_1; \mathbb{S}_2]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) = & [\![\mathbb{S}_2]\!]_{\mathsf{S}} ([\![\mathbb{S}_1]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}})) \\ \text{(I.H.)} \quad \subseteq & [\![\mathbb{S}_2]\!]_{\mathsf{S}} (\lfloor M'' \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) \\ \text{(I.H.)} \quad \subseteq & \lfloor M' \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}} \end{array}$$

as required.

Case $\mathbb{S}_1 + \mathbb{S}_2$
RTS.

$$[\![\mathbb{S}_1 + \mathbb{S}_2]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) \subseteq \left\lfloor M' \bullet_{\mathbb{M}} \{m\} \right\rfloor_{\mathbb{M}}$$

where $(M, \mathbb{S}_1, M'), (M, \mathbb{S}_2, M') \in \mathrm{Ax}_{\mathsf{S}}$.

Proof.

$$\begin{array}{rll} [\![\mathbb{S}_1 + \mathbb{S}_2]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) = & [\![\mathbb{S}_1]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) \cup [\![\mathbb{S}_2]\!]_{\mathsf{S}} (\lfloor M \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}}) \\ \text{(I.H.)} \quad \subseteq & \lfloor M' \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}} \cup \lfloor M' \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}} \\ \subseteq & \lfloor M' \bullet_{\mathbb{M}} \{m\} \rfloor_{\mathbb{M}} \end{array}$$

as required.

Case $\mathbb{S}^\star$
RTS.

$$[\![\mathbb{S}^\star]\!]_{\mathsf{S}}\left(\lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}}\right) \subseteq \lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}}$$

where $(M, \mathbb{S}, M) \in \mathrm{Ax}_{\mathsf{S}}$.

Proof.

$$\begin{array}{rll} [\![\mathbb{S}^\star]\!]_{\mathsf{S}}\left(\lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}}\right) = & [\![\,\mathrm{skip} + \mathbb{S}; \mathbb{S}^\star]\!]_{\mathsf{S}}\left(\lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}}\right) \\ = & [\![\,\mathrm{skip}]\!]_{\mathsf{S}}\left(\lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}}\right) \cup [\![\mathbb{S}; \mathbb{S}^\star]\!]_{\mathsf{S}}\left(\lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}}\right) \\ \text{(I.H.)} \quad \subseteq & \lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}} \cup \lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}} \\ \subseteq & \lfloor M \bullet_{\mathbb{M}} \{m\}\rfloor_{\mathbb{M}} \end{array}$$

as required.

□

Lemma 19. For all $w_1, w_2, w, w' = (l', g', \mathcal{I}', \mathcal{G}') \in \mathsf{World}$,

$$w_1 \bullet w_2 = w \wedge (l', g', \mathcal{I}', \mathcal{G}') \in G(w_1) \implies ((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R(w_2)$$

Proof. Pick an arbitrary w_1, w_2, w and (l_1', s', ς') such that:

$$w_1 \bullet w_2 = w \tag{A.192}$$
$$(l', g', \mathcal{I}', \mathcal{G}') \in G(w_1) \tag{A.193}$$

RTS.

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R(w_2)$$

* From (A.193) and by definition of G we know:

$$(l', g', \mathcal{I}', \mathcal{G}') \in (G^{\mathsf{u}} \cup G^{\mathsf{e}} \cup G^{\mathsf{s}})^\star (w_2) \tag{A.194}$$

From (A.192), (A.194) and by Lemmata 20, 21 and 22 we have:

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in (R^{\mathsf{u}} \cup R^{\mathsf{e}} \cup R^{\mathsf{s}})^\star (w_2)$$

and consequently

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R(w_2)$$

as required. □

Lemma 20. For all $w_1, w_2, w, w' = (l', g', \mathcal{I}', \mathcal{G}') \in \mathsf{World}$,

$$w_1 \bullet w_2 = w \wedge (l', g', \mathcal{I}', \mathcal{G}') \in G^{\mathsf{u}}(w_1) \implies ((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R^{\mathsf{u}}(w_2)$$

where we write $R^{\mathsf{u}}(w)$ to denote $\{w' \mid (w, w') \in R^{\mathsf{u}}(w)\}$.

Proof. Pick an arbitrary $W_1 = (l_1, g_1, \mathcal{I}_1, \mathcal{G}_1)$, $w_2 = (l_2, g_2, \mathcal{I}_2, \mathcal{G}_2), w$ and $(l', g', \mathcal{I}', \mathcal{G}')$ such that:

$$w_1 \bullet w_2 = w \tag{A.195}$$
$$(l', g', \mathcal{I}', \mathcal{G}') \in G^{\mathsf{u}}(w_1) \tag{A.196}$$

RTS.

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R^{\mathsf{u}}(w_2)$$

* From (A.195) we know:

$$g_1 = g_2 \tag{A.197}$$
$$\mathcal{I}_1 = \mathcal{I}_2 \wedge \mathcal{G}_1 = \mathcal{G}_2 \tag{A.198}$$

By definition of $G^{\mathsf{u}}$and from (A.196) and (A.198) we know:

$$\mathcal{I}' = \mathcal{I}_1 = \mathcal{I}_2 \wedge \mathcal{G}' = \mathcal{G}_1 = \mathcal{G}_2 \tag{A.199}$$
$$(l_1 \circ g_1)_{\mathsf{K}})^{\perp}_{\mathbb{K}} = \left((l' \circ g')_{\mathsf{M}}\right)^{\perp}_{\mathbb{K}} \tag{A.200}$$
$$g' = g_1 \vee \left(\begin{array}{l} \exists \kappa \leq (l_1)_{\mathsf{K}}. (g_1, g') \in \mathcal{G}_1(\kappa) \wedge \\ ((l_1 \circ g_1)_{\mathsf{M}})^{\perp}_{\mathbb{M}} = ((l' \circ g')_{\mathsf{M}})^{\perp}_{\mathbb{M}} \end{array} \right)$$

There are two cases to consider:

Case 1. $g_1 = g'$
From (A.197) and the assumption of the case we know $g' = g_2$. Consequently, from (A.199) we have:

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') = (l_2, g_2, \mathcal{I}_2, \mathcal{G}_2) \tag{A.201}$$

By definition of R^{u} and from (A.201) we can conclude:

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R^{\mathsf{u}}(l_2, g_2, \mathcal{I}_2, \mathcal{G}_2) \tag{A.202}$$

as required.

Case 2.

$$\exists \kappa \leq (l_1)_{\mathsf{K}}.\, (g_1, g') \in \mathcal{G}_1(\kappa) \tag{A.203}$$

$$((l_1 \circ g_1)_{\mathsf{M}})^{\perp}_{\mathbb{M}} = \big((l' \circ g')_{\mathsf{M}}\big)^{\perp}_{\mathbb{M}} \tag{A.204}$$

From (A.195), (A.197) and (A.198) we know that

$$w = (l_1 \circ l_2, g_2, \mathcal{I}_2, \mathcal{G}_2) \tag{A.205}$$

Since $\mathsf{wf}\,(w)$ (by definition of World) and from (A.197) we know:

$$(l_1 \circ l_2 \circ g_2)_{\mathsf{K}} = (l_1)_{\mathsf{K}} \bullet_{\mathbb{K}} (l_2)_{\mathsf{K}} \bullet_{\mathbb{K}} (g_2) = (l_1 \circ g_1)_{\mathsf{K}} \bullet_{\mathbb{K}} (l_2)_{\mathsf{K}} \text{ is defined} \tag{A.206}$$

$$(l_1 \circ l_2 \circ g_1)_{\mathsf{M}} = (l_1 \bullet_{\mathbb{M}} g_1)_{\mathsf{M}} \bullet_{\mathbb{M}} (l_2)_{\mathsf{M}} \text{ is defined} \tag{A.207}$$

Since $\kappa_1 \leq (l_1)_{\mathsf{K}}$ (A.203), from (A.206) and Lemma 25, we know:

$$\kappa \sharp\ (l_2)_{\mathsf{K}} \bullet_{\mathbb{K}} (g_2)_{\mathsf{K}} \tag{A.208}$$

From (A.197), (A.204) and (A.207) we know

$$(l' \circ g')_{\mathsf{M}} \bullet_{\mathbb{M}} (l_2)_{\mathsf{M}} = (l' \circ l_2 \circ g')_{\mathsf{M}} \quad \text{is defined} \tag{A.209}$$

From (A.200) and (A.206) we know

$$(l' \circ g')_{\mathsf{K}} \bullet_{\mathbb{K}} (l_2)_{\mathsf{K}} = (l' \circ l_2 \circ g')_{\mathsf{K}} \quad \text{is defined} \tag{A.210}$$

From (A.209) and (A.210) we know $l'_1 \circ l_2 \circ g'$ is defined and consequently:

$$l_2 \circ g' \quad \text{is defined} \tag{A.211}$$

From (A.198), (A.203), (A.208), (A.211) and by definition of R^{u}, we have:

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') = (l_2, g_2, \mathcal{I}_2, \mathcal{G}_2) \in R^{\mathsf{u}}(l_2, s_2, \mathcal{I}_2, \mathcal{G}_2)$$

as required. □

Lemma 21. For all $w_1, w_2, w, w' = (l', g', \mathcal{I}', \mathcal{G}') \in \mathsf{World}$,

$$w_1 \bullet w_2 = w \wedge w' \in G^{\mathsf{e}}(w_1) \implies ((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R^{\mathsf{e}}(w_2)$$

Proof. Pick an arbitrary $w_1 = (l_1, g_1, \mathcal{I}_1, \mathcal{G}_1)$, $w_2 = (l_2, g_2, \mathcal{I}_2, \mathcal{G}_2), w$ and $w' = (l', g', \mathcal{I}', \mathcal{G}')$ such that:

$$w_1 \bullet w_2 = w \tag{A.212}$$
$$w' \in G^{\mathsf{e}}(w_1) \tag{A.213}$$

RTS.

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R^{\mathsf{e}}(w_2)$$

From (A.212) we know:

$$g_1 = g_2 \wedge \mathcal{I}_1 = \mathcal{I}_2 \wedge \mathcal{G}_1 = \mathcal{G}_2 \tag{A.214}$$

By definition of G^{e} and from (A.213) and (A.214) we know there exists $l_3, l_4, g'' \in \mathsf{LState}$, $\kappa_1, \kappa_2 \in \mathbb{K}$ and $\mathcal{I}''$ such that

$$\begin{array}{l} l_1 = l_3 \circ l_4 \wedge l_1' = l_3 \circ (\mathbf{0}_{\mathbb{M}}, \kappa_1) \\ \wedge g'' = l_4 \circ (\mathbf{0}_{\mathbb{M}}, \kappa_2) \wedge g' = g_2 \circ g'' \\ \wedge \kappa_1 \bullet_{\mathbb{K}} \kappa_2 \prec \mathsf{dom}\,(\mathcal{I}'') \wedge \kappa_1 \bullet_{\mathbb{K}} \kappa_2 \perp \mathsf{dom}\,(\mathcal{I}_2) \\ \wedge \mathcal{I}' = \mathcal{I}_2 \cup \mathcal{I}'' \wedge \mathcal{G}' \uparrow^{(g'', \mathcal{I}'')} (g_2, \mathcal{I}_2, \mathcal{G}_2) \wedge (\mathcal{I}', \mathcal{G}') \downarrow (g'', g_2, \mathcal{I}'') \end{array} \tag{A.215}$$

From (A.215) and the definition of R^{e} we have:

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R^{\mathsf{e}}(w_2)$$

as required. □

Lemma 22. For all $w_1, w_2, w, w' = (l', g', \mathcal{I}', \mathcal{G}') \in \mathsf{World}$,

$$w_1 \bullet w_2 = w \wedge w' \in G^{\mathsf{s}}(w_1) \implies ((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R^{\mathsf{s}}(w_2)$$

Proof. Pick an arbitrary $w_1 = (l_1, g_1, \mathcal{I}_1, \mathcal{G}_1)$, $w_2 = (l_2, g_2, \mathcal{I}_2, \mathcal{G}_2), w$ and $w' = (l', g', \mathcal{I}', \mathcal{G}')$ such that:

$$w_1 \bullet w_2 = w \tag{A.216}$$
$$w' \in G^{\mathsf{s}}(w_1) \tag{A.217}$$

RTS.

$$((w_2)_{\mathsf{L}}, g', \mathcal{I}', \mathcal{G}') \in R^{\mathsf{s}}(w_2)$$

From (A.216) we know:

$$g_1 = g_2 \wedge \mathcal{I}_1 = \mathcal{I}_2 \wedge \mathcal{G}_1 = \mathcal{G}_2 \tag{A.218}$$

By definition of $G^{\mathsf{s}}$and from (A.217), and (A.218) we know there exists $\mathcal{J}'' \in \mathsf{AMod}$ such that

$$\mathcal{J}' = \mathcal{I}_2 \cup \mathcal{J}'' \wedge \mathcal{G}_2 \uparrow^{(\mathbf{0}_{\mathsf{L}}, \mathcal{J}'')} (g_2, \mathcal{I}_2, \mathcal{G}_2) \tag{A.219}$$

From (A.219) and the definition of R^{s} we have:

$$((w_2)_{\mathsf{L}}, g', \mathcal{J}', \mathcal{G}') \in R^{\mathsf{s}}(w_2)$$

as required. □

Lemma 23. Given any separation algebra $(\mathbb{A}, \bullet_{\mathbb{A}}, \mathbf{0}_{\mathbb{A}})$ with the cross-split property, for any $a, b \in \mathbb{A}$:

$$\exists c \in \mathbb{A}.\, a \leq b \bullet_{\mathbb{A}} c \iff a \sqcap_{\mathbb{A}} b \neq \emptyset$$

Proof $\Longrightarrow$. We proceed with proof by contradiction. Take arbitrary $a, b, c \in \mathbb{A}$ such that

$$a \leq b \circ c \tag{A.220}$$

and assume

$$a \sqcap_{\mathbb{A}} b = \emptyset \tag{A.221}$$

From (A.221) and by definition of $\sqcap_{\mathbb{A}}$, we have:

$$\neg \exists d, e, f, g.\, a = d \bullet_{\mathbb{A}} e \wedge b = e \bullet_{\mathbb{A}} f \wedge d \bullet_{\mathbb{A}} e \bullet_{\mathbb{A}} f = g \tag{A.222}$$

From (A.220) we have $\exists h.\, a \bullet_{\mathbb{A}} h = b \bullet_{\mathbb{A}} c$ and consequently by the cross-split property we have:

$$\exists ab, ac, hb, hc, t.\, ab \bullet_{\mathbb{A}} ac = a \tag{A.223}$$

$$hb \bullet_{\mathbb{A}} hc = h \tag{A.224}$$

$$ab \bullet_{\mathbb{A}} hb = b \tag{A.225}$$

$$ac \bullet_{\mathbb{A}} hc = c \tag{A.226}$$

$$t = ab \bullet_{\mathbb{A}} ac \bullet_{\mathbb{A}} hb \bullet_{\mathbb{A}} hc \tag{A.227}$$

From (A.227) we have:

$$\exists s.\, ab \bullet_{\mathbb{A}} ac \bullet_{\mathbb{A}} hb = s \tag{A.228}$$

From (A.223), (A.225) and (A.228) we have:

$$\exists d, e, f, g.\, a = d \bullet_{\mathbb{A}} e \wedge b = e \bullet_{\mathbb{A}} f \wedge d \bullet_{\mathbb{A}} e \bullet_{\mathbb{A}} f = g \tag{A.229}$$

From (A.222) and (A.229) we derive a contradiction and can hence deduce:

$$a \sqcap_{\mathbb{A}} b \neq \emptyset$$

as required.

Proof $\Leftarrow$. Take arbitrary $a, b \in \mathbb{A}$ such that

$$a \sqcap_{\mathbb{A}} b \neq \emptyset$$

Then by definition of $\sqcap_{\mathbb{A}}$ we have:

$$\begin{aligned} \exists d, e, f \in \mathbb{A}.\, a &= d \bullet_{\mathbb{A}} e \\ b &= e \bullet_{\mathbb{A}} f \\ & d \bullet_{\mathbb{A}} e \bullet_{\mathbb{A}} f \text{ is defined} \end{aligned}$$

and thus by definition of $\leq$ we have $a \leq b \bullet_{\mathbb{A}} d$ and consequently

$$\exists c.\, a \leq b \bullet_{\mathbb{A}} c$$

as required. □

Lemma 24. Given any separation algebra $(\mathcal{M}, \bullet_{\mathcal{M}}, \mathbf{0}_{\mathcal{M}})$ with the cross-split property:

$$\forall a, b, c \in \mathcal{M}.\, a \leq b \bullet_{\mathcal{M}} c \implies \exists m, n.\, a = m \bullet_{\mathcal{M}} n \ \wedge\ m \leq b \ \wedge\ n \leq c$$

Proof. Pick an arbitrary $a, b, c \in \mathcal{M}$. Since $a \leq b \bullet_{\mathcal{M}} c$, we know $\exists d \in \mathcal{M}$. such that:

$$a \bullet_{\mathcal{M}} d = b \bullet_{\mathcal{M}} c \tag{A.230}$$

By the cross-split property of $\mathcal{M}$, We can deduce: $\exists ab, ac, db, dc \in \mathcal{M}$. such that:

$$a = ab \bullet_{\mathcal{M}} ac \tag{A.231}$$
$$b = ab \bullet_{\mathcal{M}} db \tag{A.232}$$
$$c = ac \bullet_{\mathcal{M}} dc \tag{A.233}$$
$$d = db \bullet_{\mathcal{M}} dc$$

Since $ab \leq b$ (A.232) and $ac \leq c$ (A.233), from (A.231) we can deduce:

$$\exists m, n \in \mathcal{M}.\, a = m \bullet_{\mathcal{M}} n \ \wedge\ m \leq b \ \wedge\ n \leq c$$

as required. □

Lemma 25. Given any separation algebra $(\mathcal{M}, \bullet_{\mathcal{M}}, \mathbf{0}_{\mathcal{M}})$,

$$\forall a, b, c, d \in \mathcal{M}.\, a \bullet_{\mathcal{M}} b = d \ \wedge\ c \leq b \implies \exists f \in \mathcal{M}.\, a \bullet_{\mathcal{M}} c = f$$

Proof. Pick an arbitrary $a, b, c, d \in \mathcal{M}$ such that:

$$a \bullet_{\mathcal{M}} b = d \tag{A.234}$$
$$c \leq b \tag{A.235}$$

From (A.235), we have:

$$\exists e \in \mathcal{M}.\, c \bullet_{\mathcal{M}} e = b \tag{A.236}$$

and consequently from (A.234) we have:

$$a \bullet_{\mathcal{M}} c \bullet_{\mathcal{M}} e = d \tag{A.237}$$

Since $e \leq d$ (A.237), we have:

$$\exists f \in \mathcal{M}.\, e \bullet_{\mathcal{M}} f = d \tag{A.238}$$

From (A.237), (A.238) and cancellativity of separation algebras we have:

$$a \bullet_{\mathcal{M}} c = f \tag{A.239}$$

and thus

$$\exists f \in \mathcal{M}.\, a \bullet_{\mathcal{M}} c = f \tag{A.240}$$

as required. □

Ode To Q

Dear Q, this is P, let's call off this fight
Though we may differ, we're both in the right
Contend we may yet consistent our ways
In concert our shared fates unfold with sleight

$$\boxed{P}_{I_1} \qquad \boxed{Q}_{I_2}$$
$$\boxed{P}_{I_1} * \boxed{Q}_{I_2} = :)$$
$$I_1 = \{a : P \rightsquigarrow P'\} \quad I_2 = \{a : Q \rightsquigarrow Q'\}$$
$$I_1 \cup I_2 = :)$$

I'm lost in the dark, you see all that's light
Your wisdom I crave, what say we unite?
In tandem evolving we gain what we yearned
Though strong our liaison we part with delight

$$\boxed{P}_{I_1} : (\quad * \quad \boxed{Q}_{I_2} :)$$
$$\boxed{P \uplus Q}_{I_1 \cup I_2} :)$$
$$\boxed{P' \uplus Q'}_{I_1 \cup I_2}$$
$$\boxed{P'}_{I_1 \cup I_2} * \boxed{Q'}_{I_1 \cup I_2}$$

That which was complete is now marred and trite
We seek out that simple yet stable rite
Your memory now dimmed, your spirit I bear
In mine for always and this I recite:

$$\boxed{P'}_{I_1 \cup I_2} :(\quad * \quad \boxed{Q'}_{I_1 \cup I_2} :($$
$$\boxed{P'}_{I_1} :) \quad * \quad \boxed{Q'}_{I_2} :)$$
$$\boxed{P'}_{I_1}$$
$$\boxed{P'}_{I_1} * ?\boxed{Q'}_{I_2} \vee \boxed{Q''}_{I_3} \vee \boxed{Q'''}_{I_3} \vee \cdots$$

Our discord our strength, conciseness our might
Tangled at heart though disjointed at sight
CoLoSL our feat, our effort but slight
Sound our ideas, our paper pray cite!

Concurrent
Local
Subjective
Logic