Denotational Semantics for Probabilistic and Concurrent Programs

Noam Zilberstein
Cornell University
Ithaca, NY
noamz@cs.cornell.edu

Daniele Gorla
"Sapienza" Università di Roma
Rome, Italy
gorla@di.uniroma1.it

Alexandra Silva
Cornell Univeristy
Ithaca, NY
alexandra.silva@cornell.edu

***Abstract*—We develop a denotational model for programs that have standard programming constructs such as conditionals and while-loops, as well as probabilistic and concurrent commands. Whereas semantic models for languages with either concurrency or randomization are well studied, their combination is limited to languages with bounded loops. Our work is the first to consider both randomization and concurrency for a language with unbounded looping constructs. The interaction between Boolean tests (arising from the control flow structures), probabilistic actions, and concurrent execution creates challenges in generalizing previous work on pomsets and convex languages, prominent models for those effects, individually. To illustrate the generality of our model, we show that it recovers a typical powerdomain semantics for concurrency, as well as the convex powerset semantics for probabilistic nondeterminism.**

I. Introduction

Program semantics is a longstanding research area, underpinning many static analysis and verification techniques. From simple imperative languages, to higher-order functional ones, to the more recent probabilistic and quantum paradigms, the development of programming languages is often accompanied by a study of mathematical objects capturing the semantics of syntactic constructs. These structures not only influence verification techniques, but also how a language can be extended.

Semantics of imperative and functional languages have been studied for decades in several styles—denotational, operational, and axiomatic—and each approach offers different insights. Denotational semantics is often chosen for expressivity and extensibility. For example, adding recursion to a denotational model raises questions about the properties of the underlying *domain*, which motivated the development of domain theory to provide mathematical representations of iterated computations [1], [2]. More expressive program constructs necessitate more involved semantic domains; two prime examples of this are probabilistic and concurrent programs.

Semantics of probabilistic programs has been widely studied, going back to the seminal works of Kozen [3] and McIver and Morgan [4]. To capture probabilistic behavior, domains require some extra algebraic structure to represent the *distribution* of outcomes generated by operations such as sampling and coin flips. This added convex structure brings additional complexity to questions of language extensions and program analysis, but the last decade has brought success stories on taming the complexity through program logics and predicate transformer calculi [4]–[8].

Concurrent programs are prevalent in many areas, with prominent applications in distributed systems. These programs have long been a target of program analysis, as concurrency bugs are incredibly hard to detect and prevent. There have been many approaches to concurrent semantics, some arising from more theoretical process algebra research and some from more practical hardware considerations (*e.g.,* memory models). In terms of denotational semantics, *pomsets* [9]–[12] provide a prominent model, expressing the causality between actions and the parallel branching behavior as a partial order.

However, the situation is more subtle when programs have not only concurrent behavior, but also control-flow-structures—*i.e.,* if-then-else and while-loops. For example, in recent work on Concurrent Kleene Algebra with Tests (CKAT) [13], the proposed model (series-parallel guarded strings) turned out not to adequately capture the behavior of concurrent programs: using CKAT, one can prove that, for any test b and program e, the equation $b \cdot e \cdot \overline{b} = 0$ holds. In other words, no program can change the outcome of any test or, equivalently (and undesirably), any test is an invariant of any program!

Semantics of concurrent and probabilistic programs, separately, are subtle and complex; unsurprisingly, their combination introduces additional challenges. In this paper, we develop a denotational model for programs that mix *both* probabilistic and concurrent constructs. More precisely, we will work with a simple concurrent imperative programming language—whose syntax is shown in Figure 1—where the basic actions can include probabilistic operations such as random sampling. This is not only interesting from a theoretical point of view; for decades, randomization has been used to enhance the capabilities of concurrent programs [14]–[17]. For example, Dijkstra famously showed that the *Dining Philosophers Problem*—a distributed synchronization scheme—has no deterministic solution, but it has a simple randomized one [18].

Despite the importance of mixing randomization and concurrency, there is no semantic model of probabilistic concurrent programs with unbounded looping. Such a model is necessary to analyze behaviors of many concurrent randomized protocols—including the Dining Philosophers—which avoid deadlock with probability 1, but admit an infinite trace whose probability converges to 0. In this paper, we develop such a

model, drawing insights from prior work on concurrent and probabilistic semantics, but solving challenges that are unique to their combination. As a result, our model is consistent with well-established probabilistic semantics (Theorem 2). We illustrate the subtleties through the following program:

$$x :\approx \mathsf{flip}\left(\tfrac{1}{2}\right); ((\texttt{if } x \ \{y := 0\} \ \texttt{else} \ \{y := 1\}) \mid y := 2)$$

The variable x stores the result of a Bernoulli experiment (a fair coin flip) and is then used to control the flow of the parallel execution of two threads that determine the value of y. What should the semantics of such a program be? The value of y depends on the result of the Bernoulli experiment *and* on the parallel execution. Hence, any semantic domain for this program must encompass probability distributions of program states, while at the same time accounting for nondeterminism introduced from parallel branching. As we will show in Example 3, one might attempt to give semantics to this program using a naive composition of distributions and powersets (as monads) but, as it is well known from prior work, such combination needs care and can easily lead to non-compositional semantics [19]–[22].

In this paper, we develop a pomset model in which probabilistic computations can be interpreted. Crucially, our pomset structure also tracks the results of control-flow tests (which can themselves be probabilistic). This is done symbolically by associating a Boolean formula to each node of the pomset, recording the resolution of tests needed to reach that point. We call these new structures *pomsets with formulae*, which capture both parallel composition and guarded branching[1]. As such, this structure encodes many traces, avoiding the need for a set of traces and simultaneously remaining compositional in the presence of probabilistic actions.

Recent work on program logics [23] has proposed pomsets with tests, a *finite* structure with goals similar to ours. Importantly, pomsets with tests cannot give semantics to general purpose looping or recursion operations, which produce traces that are unbounded in size. While we took inspiration from pomsets with tests, our development is substantially different, and those differences are crucial to ensuring good domain-theoretic properties. More precisely, pomsets with formulae have a directed complete partial order (DCPO) structure, and operations such as sequential composition are Scott continuous, so that the semantics of loops can be defined as a least fixed point. Moreover, operations on infinite pomsets can be obtained from operations on their finite approximations.

In a nutshell, the contributions of this paper are:

1) We introduce *pomsets with formulae*, and prove that they have well behaved domain-theoretic properties (Section II).
2) We prove an *extension lemma* similar to that of [24], for extending monotone operations on finite pomsets to continuous operations on infinite pomsets (Lemma 6).

[1]Formulae are more expressive than the usual conflict relation, present, *e.g.,* in event structures: conflict can only exclude execution of certain branches, whereas formulae record the entire outcome of tests that lead to every node.

$$\begin{array}{lll} C ::= & \texttt{skip} & \textit{(No-op)} \\ & \mid C_1; C_2 & \textit{(Sequential Composition)} \\ & \mid C_1 \mid C_2 & \textit{(Parallel Composition)} \\ & \mid \texttt{if } b \ \{C_1\} \ \texttt{else} \ \{C_2\} & \textit{(Guarded Branching)} \\ & \mid \texttt{while } b \ \{C\} & \textit{(Looping)} \\ & \mid a & \textit{(Atomic Actions)} \end{array}$$

Fig. 1. Syntax of a concurrent imperative programming language.

3) We define guarded, sequential, and parallel composition operations on pomset with formulae (Section III) and prove that the first two are Scott continuous.
4) We use our new pomset structure to define the denotational semantics for the concurrent imperative language of Figure 1, with uninterpreted actions (Section IV-A).
5) We define a linearization operation (via the extension lemma), parametric on the computational domain, to convert the pomset semantics into a state transformer (Section IV-B).
6) We instantiate our semantics in a nondeterministic powerdomain model [25], [26], and prove that this gives us the same semantics as pomset languages, *e.g.,* [27], [28].
7) We instantiate our semantics using the convex powerset—a computational domain that supports both randomization and nondeterminism—and prove that the semantics of the sequential fragment of our programming language is the same as that of [8], [29].

We discuss related work and next steps in Section VII. All omitted proofs and details are provided in the appendix.

II. Semantic Structures

In this section, we define a particular kind of labelled partial order, which will form the core of our semantic model. We call this structure a *labeled partial order with formulae* (LPOF), as it includes a special labelling function, assigning a Boolean formula to each node. We start by recalling basic definitions from domain theory; for a complete treatment refer to [30].

A. Preliminaries: Partial orders and DCPOs

Let *nodes* be a countable universe of nodes, that will be denoted by $x, y, z, \ldots$, whereas subsets of *nodes* will be denoted by $N, X, Y, \ldots$.

Definition 1 (Poset)**.** *A partially ordered set, or* poset*, is a set equipped with a partial order: that is, a pair $\langle X, \leq \rangle$ consisting of a set X and an order relation $\leq \subseteq X \times X$ which is reflexive, transitive and antisymmetric. When reflexivity does not hold, the poset is called* strict *and the order relation is usually denoted by $<$.*

Given a strict poset $\langle X, < \rangle$ and any $x \in X$, we define the upward and downward closures of x as

$$x{\uparrow} \triangleq \{y \in X \mid x < y\} \qquad x{\downarrow} \triangleq \{y \in X \mid y < x\};$$

and its set of immediate successors and predecessors as

$$\mathsf{succ}(x) \triangleq \{y \in x\uparrow \mid \nexists z.(x < z < y)\}$$
$$\mathsf{pred}(x) \triangleq \{y \in x\downarrow \mid \nexists z.(y < z < x)\}$$

All the above operations can easily be extended to sets of elements, *e.g.,* $X\uparrow \triangleq \bigcup_{x\in X} x\uparrow$, for any set X. The set of minimal and maximal nodes of a poset is given by:

$$\max \triangleq \{x \in X \mid \mathsf{succ}(x) = \emptyset\} \quad \min \triangleq \{x \in X \mid \mathsf{pred}(x) = \emptyset\}$$

To define our new structures we need a few more definitions on posets that will play a role in proving the existence of fixed points, essential for giving a semantics to unbounded loops. First, we recall the notion of downward closed sets.

Definition 2 (Downward Closure)**.** *Given a poset $\langle X, <\rangle$, a set $Y \subseteq X$ is downward closed, written $Y \subseteq_\downarrow X$, iff for all $y \in Y$ it holds that $y\downarrow \subseteq Y$.*

Second, we need a notion of a poset being finitely preceded.

Definition 3 (Finitely Preceded)**.** *A poset $\langle X, <\rangle$ is* finitely preceded *if there are finitely many elements smaller than each element; that is: $|x\downarrow| < \infty$, for all $x \in X$.*

Third, we define the level of elements in the poset.

Definition 4 (Level [24])**.** *Given a finitely preceded poset $\langle X, <\rangle$, define the level $\mathsf{lev}\colon X \to \mathbb{N}$ of a node as follows:*

$$\mathsf{lev}(x) \triangleq \sup \left\{ n \;\middle|\; \begin{array}{l} \exists x_0, \ldots, x_n \in X.\ x_0 \in \min \ \wedge \\ \quad \wedge\ x_n = x \ \wedge\ \forall i < n.\ x_{i+1} \in \mathsf{succ}(x_i) \end{array} \right\}$$

We also define its inverse $\mathsf{lev}^{-1}\colon \mathbb{N} \to \mathcal{P}(X)$:

$$\mathsf{lev}^{-1}(n) \triangleq \{x \in X \mid \mathsf{lev}(x) = n\}$$

Finally, in order to find fixed points, we will need to have a directed complete partial order structure and a notion of continuity for functions, which we now define.

Definition 5 (DCPO)**.** *Given a poset $\langle X, \leq\rangle$, a subset $D \subseteq X$ is called* directed *iff it is nonempty and for every two elements $x_1, x_2 \in D$, there exists $x \in D$ such that $x_1, x_2 \leq x$.*

$\langle X, \leq\rangle$ is a directed-complete partial order *(*DCPO*) iff, for every directed set D, $\sup D$ exists; furthermore, $\langle X, \leq\rangle$ is called* pointed *if there exists an element $\bot \in X$ such that $\bot \leq x$ for all $x \in X$.*

Definition 6 (Scott Continuity)**.** *Given two* DCPOs *$\langle X, \leq_X\rangle$ and $\langle Y, \leq_Y\rangle$, a function $f\colon X \to Y$ is* Scott Continuous *if it is monotone and preserves suprema of directed sets. That is:*

$$x \leq_X x' \Rightarrow f(x) \leq_Y f(x') \quad \textit{and} \quad \sup_{x\in D} f(x) = f(\sup D)$$

for every $D \subseteq X$ directed.

B. Labelled Partial Orders with Formulae

In this subsection, we introduce Labelled Partial Orders with Formulae (LPOF). We start by defining the class of Boolean formulae using nodes as the variables:

$$\mathit{form} \ni \psi ::= \mathsf{true} \mid \mathsf{false} \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \neg\psi \mid x$$

That is, a formula $\psi \in \mathit{form}$ is either true, false, a conjunction, disjunction, negation, or a Boolean variable $x \in \mathit{nodes}$. Formulae are interpreted by valuations $v\colon \mathit{nodes} \to \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$. Satisfaction is defined in the standard way:

$v \vDash \mathsf{true}$	always
$v \vDash \mathsf{false}$	never
$v \vDash \psi_1 \wedge \psi_2$	iff $v \vDash \psi_1$ and $v \vDash \psi_2$
$v \vDash \psi_1 \vee \psi_2$	iff $v \vDash \psi_1$ or $v \vDash \psi_2$
$v \vDash \neg\psi$	iff $v \nvDash \psi$
$v \vDash x$	iff $v(x) = 1$

The variables of a formula $\mathsf{vars}(\psi)$ are those nodes that appear in the formula. We write:

- $\mathsf{sat}(\psi)$ iff there exists a valuation v such that $v \vDash \psi$;
- $\psi \Rightarrow \psi'$ iff $v \vDash \psi'$, for every $v \vDash \psi$;
- $\psi \Leftrightarrow \psi'$ iff $\psi \Rightarrow \psi'$ and $\psi' \Rightarrow \psi$.

Definition 7 (LPOF)**.** *Let $\langle L, \leq\rangle$ be a pointed and finitely preceded* DCPO *of labels with bottom element $\bot$. A labelled partial order with formulae (*LPOF*) over L is a 4-tuple $\alpha = \langle N, <, \lambda, \varphi\rangle$ where:*

1) $N \subseteq \mathit{nodes}$ is a countable set of nodes;
2) $\langle N, <\rangle$ is a (strict) poset such that:
  a) it is finitely preceded;
  b) every level of the poset has a finite number of nodes, that is: $|\mathsf{lev}^{-1}(n)| < \infty$ for all $n \in \mathbb{N}$; and
  c) it is single-rooted, that is: $|\min| = 1$.
3) $\lambda\colon N \to L$ is a labelling function such that $\mathsf{succ}(x) = \emptyset$ whenever $\lambda(x) = \bot$.
4) $\varphi\colon N \to \mathit{form}$ is a formula function satisfying:
  a) $\mathsf{sat}(\varphi(x))$ and $\mathsf{vars}(\varphi(x)) \subseteq x\downarrow$, for all $x \in N$;
  b) $\varphi(y) \Rightarrow \varphi(x)$, for all $x < y$.

We denote by $\mathit{lpo}(L)$ the set of all LPOFs *over L.*

For some LPOF $\alpha = \langle N, <, \lambda, \varphi\rangle$, we use N_α, $<_\alpha$, λ_α, and φ_α to refer to its constituent parts; similarly, we annotate all functions with the LPOF they refer to, *e.g.,* $x\downarrow_\alpha$, $\mathsf{succ}_\alpha(x)$, $\min_\alpha$, etc. We now explain the conditions in Definition 7.

Conditions (2a) and (2b) are standard; in particular, since a node represents an action of a program, having finitely many predecessors ensures that every action may happen in a finite amount of time. Moreover, asking that every level has a finite number of nodes is a weakening of the usual finite branching property: indeed, we allow a node to have infinitely many successors, but they cannot be all at the same level. We defer the discussion on Condition (2c) to Remark 2 later on.

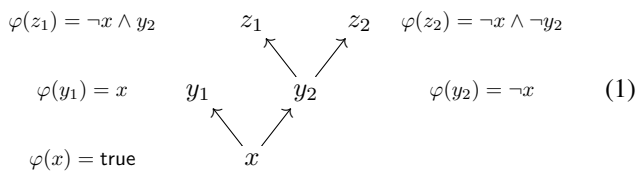
Concerning Condition (3), labels correspond to *Boolean tests* and *actions*, performed during a program execution. Unlike standard pomset models, we allow actions to be probabilistic, and our model is consistent with known sequential probabilistic semantics (Theorem 2). For the moment, we leave labels unspecified, only assuming the presence of $\bot$, used to denote an undetermined computation, which is fundamental in giving the fixed point semantics to while-loops. Since such

loops may not terminate, we cannot consider a node labelled with $\bot$ as a predecessor of any node, since the latter nodes may never be executed (because they can miss one of their causes, if $\bot$ denotes a non-terminating computation).

Condition (4) is about formulae, which are crucial for modeling branching due to guards in conditional constructs (if-then-else and while), such as the following program

$$\texttt{if } b_1 \texttt{ \{}a_1\texttt{\} else \{if } b_2 \texttt{ \{}a_2\texttt{\} else \{}a_3\texttt{\}\}}$$

where b_1 and b_2 are Boolean tests and a_1, a_2, a_3 are atomic actions. This program corresponds to the LPOF below



(1)

In this case, we shall let $\lambda(x) = b_1$, $\lambda(y_1) = a_1$, $\lambda(y_2) = b_2$, $\lambda(z_1) = a_2$ and $\lambda(z_2) = a_3$. Hence, every node w labelled with a Boolean test yields a (binary) branch and its (two) successors have the same formula as w, with an extra conjunct that is either w or $\neg w$, according to the outcome of the test. With this in mind, Condition (4) is quite intuitive:

- Requiring $\mathsf{sat}(\varphi(w))$ amounts to expressing the fact that every node w is reachable, *i.e.,* there exists a truth assignment v such that $v \models \varphi(w)$. The truth assignment tells, for every node labeled with a Boolean test, whether that test passes or not. In our example, z_2 is reachable when both the tests b_1 (labelling x) and b_2 (labelling y_2) fail; this is exactly what satisfiability of $\varphi(z_2)$ entails (viz., that both $\lambda(x)$ and $\lambda(y_2)$ must be false to satisfy $\neg x \wedge \neg y_2$).
- This intuition justifies the other two requirements of Definition 7(4): reachability of a node can only depend on the values of the tests that precede it (and so $\varphi(w)$ can only use nodes that precede w) and, since along a path we shall only add conjuncts, the formula of a higher-level node is stronger than the formulae of all of its predecessors.

Next, we define an order on LPOFs, which we will use for the construction of fixed points to give semantics to unbounded loops. The intuition is that $\alpha \sqsubseteq_{lpo} \beta$ iff β has *more behaviors* than α, meaning that β can be obtained by expanding $\bot$ nodes in α into larger structures (each unfolding of a while-loop is obtained in this way, as shown in Appendix A). Since LPOFs can only be expanded from $\bot$ nodes, the label set L must be pointed; however, the order on L need not be flat: labels themselves can also become larger when passing from α to β (*e.g.,* if actions are nondeterministic assignments from a given set, this corresponds to enlarging the set of possible choices). This can also be useful to model *invariant sensitive execution*, introduced in Probabilistic Concurrent Outcome Logic [23].

Notationally, Bot_α denotes the set of nodes labelled with $\bot$, *i.e.,* $\mathsf{Bot}_\alpha \triangleq \{x \in N_\alpha \mid \lambda_\alpha(x) = \bot\}$ and nBot_α denotes the complement, *i.e.,* $\mathsf{nBot}_\alpha \triangleq \{x \in N_\alpha \mid \lambda_\alpha(x) \neq \bot\}$.

Definition 8 (Ordering on LPOFs)**.** *We let $\alpha \sqsubseteq_{lpo} \beta$ iff:*

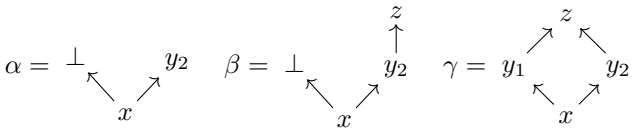
1) $N_\alpha \subseteq_\downarrow N_\beta$*;*

2) $<_\alpha = <_\beta \cap (N_\alpha \times N_\alpha)$*;*

3) $\forall x \in N_\alpha$*, the following hold:*

a) $\lambda_\alpha(x) \leq \lambda_\beta(x)$*;*
b) $\varphi_\alpha(x) = \varphi_\beta(x)$*;*
c) $\mathsf{succ}_\alpha(x) = \mathsf{succ}_\beta(x) \setminus \mathsf{Bot}_\alpha{\uparrow_\beta}$.

Example 1. Consider the following three LPOFs:



where the nodes labelled with $\bot$ in α and β are y_1 (all the other nodes are labelled with non-$\bot$ labels). By following the intuition given above, both α and β are attempts to specify that the computation in γ is truncated at y_1. However, we only have that $\alpha \sqsubseteq_{lpo} \gamma$; this is because, if $\bot$ represents a diverging computation, then z will never have all the causes it needs to be executed and so it must disappear. Indeed, $\beta \not\sqsubseteq_{lpo} \gamma$ because, even though $N_\beta \subseteq_\downarrow N_\gamma$, condition (3c) of Definition 8 is violated: indeed, $z \in \mathsf{succ}_\beta(y_2)$ but $z \notin \mathsf{succ}_\gamma(y_2) \setminus \mathsf{Bot}_\beta{\uparrow_\gamma}$, since $y_1 \in \mathsf{Bot}_\beta$ and $y_1 <_\gamma z$.

We now list the fundamental properties of LPOFs.

Lemma 1. *If $\alpha \sqsubseteq_{lpo} \beta$, then $N_\beta \setminus N_\alpha = \mathsf{Bot}_\alpha{\uparrow_\beta}$.*

Lemma 2. *$\sqsubseteq_{lpo}$ is a partial order.*

Lemma 3. *For any directed set $D \subseteq lpo(L)$, $\sup D$ exists and is given by $\alpha = \langle N_\alpha, <_\alpha, \lambda_\alpha, \varphi_\alpha \rangle$, where:*

$$N_\alpha \triangleq \bigcup_{\beta \in D} N_\beta \qquad <_\alpha \triangleq \bigcup_{\beta \in D} <_\beta$$
$$\lambda_\alpha(x) \triangleq \sup_{\beta \in D\,:\, x \in N_\beta} \lambda_\beta(x) \qquad \varphi_\alpha(x) \triangleq \psi_x$$

and $\psi_x = \varphi_\beta(x)$, for all $\beta \in D$ such that $x \in N_\beta$.

Remark 1. φ_α in Lemma 3 is well defined since, for every $x \in$ *nodes* and every $\beta, \beta' \in D$ that contain x, it must be that $\varphi_\beta(x) = \varphi_{\beta'}(x)$. Indeed, since D is directed, there must exist a $\gamma \in D$ such that $\beta, \beta' \sqsubseteq_{lpo} \gamma$ and, by Definition 8(3b), we have that $\varphi_\beta(x) = \varphi_\gamma(x) = \varphi_{\beta'}(x)$.

Corollary 1. *$\langle lpo(L), \sqsubseteq_{lpo} \rangle$ is a* DCPO.

Proof. Follows immediately from Lemmas 2 and 3. □

Notice that *$lpo(L)$* is *not* pointed, since the node at the root can vary; hence, there is not one single LPOF that is smaller (w.r.t. $\sqsubseteq_{lpo}$) than all other LPOFs. This situation will change (see Lemma 4) when moving to pomsets, that indeed abstract away from the specific nodes involved.

C. Pomsets with Formulae

Definition 9 (LPOF Isomorphism)**.** *Let α and β be* LPOF*s and $f\colon N_\alpha \to N_\beta$ be a bijection between their nodes. Define $f(\alpha) = \langle N, <, \lambda, \varphi\rangle$ as*

$$N \triangleq \{f(x) \mid x \in N_\alpha\} \qquad < \triangleq \{(f(x), f(y)) \mid x <_\alpha y\}$$
$$\lambda \triangleq \lambda_\alpha \circ f^{-1} \qquad \varphi \triangleq f \circ \varphi \circ f^{-1}$$

where $f(\psi)$ syntactically renames the variables of ψ in the obvious way; α and β are isomorphic, written $\alpha \equiv \beta$, iff there is a bijection $f\colon N_\alpha \to N_\beta$ such that $f(\alpha) = \beta$.

Definition 10 (Pomsets with Formulae)**.** *We denote the isomorphism class of α as $[\alpha] \triangleq \{\beta \in \mathit{lpo}(L) \mid \alpha \equiv \beta\}$. A pomset with formulae (or, simply, a pomset) $\boldsymbol{\alpha} \in \mathit{pom}(L)$ is an isomorphism class of* LPOF*s on L:*

$$\mathit{pom}(L) \triangleq \{[\alpha] \mid \alpha \in \mathit{lpo}(L)\}$$

We say that a pomset is finite *if it has finitely many nodes; we denote with $\mathit{pom}_{\mathsf{fin}}(L)$ the set of finite pomsets. Finally, we define an order on pomsets:*

$$\boldsymbol{\alpha} \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta} \quad \textit{iff} \quad \forall \alpha \in \boldsymbol{\alpha}.\, \exists \beta \in \boldsymbol{\beta}.\, \alpha \sqsubseteq_{\mathit{lpo}} \beta$$

Lemma 4. *$\langle \mathit{pom}(L), \sqsubseteq_{\mathit{pom}}\rangle$ is a pointed* DCPO*.*

The semantics of programs depends on a variety of pomset composition operations (*e.g.,* sequential, guarded, and parallel composition). We will also later *linearize* pomsets into state transformer functions. Some of these operations are difficult or impossible to define on infinite structures, so we need ways to extend functions on finite structures to functions on infinite ones. To achieve this, we show that any infinite pomset can be represented as the supremum of its finite approximations, which we now define.

Definition 11 (Finite Approximations)**.** *For any $\boldsymbol{\alpha} \in \mathit{pom}(L)$, let $\lfloor \boldsymbol{\alpha} \rfloor_{\mathsf{fin}} \triangleq \{\boldsymbol{\beta} \in \mathit{pom}_{\mathsf{fin}}(L) \mid \boldsymbol{\beta} \sqsubseteq_{\mathit{pom}} \boldsymbol{\alpha}\}$ be the set of finite approximations of $\boldsymbol{\alpha}$.*

We now prove that the supremum of the finite approximations of $\boldsymbol{\alpha}$ coincides with $\boldsymbol{\alpha}$ itself.

Lemma 5. *For any $\boldsymbol{\alpha} \in \mathit{pom}(L)$, $\boldsymbol{\alpha} = \sup \lfloor \boldsymbol{\alpha} \rfloor_{\mathsf{fin}}$.*

Additionally, the finite approximations of a pomset coincide with the approximation order $\ll$ taken from [30] instantiated to pomsets (see Appendix D for more details); thus, from now on, we let $\boldsymbol{\alpha} \ll \boldsymbol{\beta}$ denote $\boldsymbol{\alpha} \in \lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}}$.

We are now ready to show that every operation f on finite pomsets can be extended to infinite ones by defining the operation on $\boldsymbol{\alpha}$ as the sup of the images through f of all the approximations of $\boldsymbol{\alpha}$. This will be useful in the remainder of the paper, where we define operations on pomsets.

Lemma 6 (Extension)**.** *Let $f\colon \mathit{pom}_{\mathsf{fin}}(L)^n \to T$ be a monotone function on the* DCPO *$\langle T, \leq\rangle$. Then $f^*\colon \mathit{pom}(L)^n \to T$, defined as:*

$$f^*(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n) \triangleq \sup_{\boldsymbol{\alpha}'_1 \ll \boldsymbol{\alpha}_1} \cdots \sup_{\boldsymbol{\alpha}'_n \ll \boldsymbol{\alpha}_n} f(\boldsymbol{\alpha}'_1, \ldots, \boldsymbol{\alpha}'_n)$$

is well-defined and Scott continuous.

III. Pomset Operations

In this section, we define pomset operations that mirror the program syntax from Figure 1. We first define the singleton LPOF on node x with label $\ell \in L$; this generalizes to pomsets and is used to give the semantics to **`skip`** and atomic actions:

$$\langle \ell \rangle_x \triangleq \langle \{x\}, \emptyset, [x \mapsto \ell], [x \mapsto \mathsf{true}]\rangle \qquad \in \mathit{lpo}(L)$$
$$\langle \ell \rangle \triangleq \{\langle \ell \rangle_x \mid x \in \mathit{nodes}\} \qquad \in \mathit{pom}(L)$$

A. Guarded Choice

We define a guarding operation which records the causality between a test and the two branches of computation defined on it. Assuming that $N_\alpha \cap N_\beta = \emptyset$, we let

$$\mathsf{guard}(x, \ell, \alpha, \beta) = \langle N, <, \lambda, \varphi\rangle$$

where $x \notin N_\alpha \cup N_\beta$, and:

$$N \triangleq \{x\} \cup N_\alpha \cup N_\beta$$
$$< \triangleq <_\alpha \cup <_\beta \cup (\{x\} \times (N_\alpha \cup N_\beta))$$
$$\lambda(y) \triangleq \begin{cases} \ell & \text{if } y = x \\ \lambda_\alpha(y) & \text{if } y \in N_\alpha \\ \lambda_\beta(y) & \text{if } y \in N_\beta \end{cases}$$
$$\varphi(y) \triangleq \begin{cases} \mathsf{true} & \text{if } y = x \\ \varphi_\alpha(y) \wedge x & \text{if } y \in N_\alpha \\ \varphi_\beta(y) \wedge \neg x & \text{if } y \in N_\beta \end{cases}$$

As an example, the LPOF depicted in (1) is obtained as

$$\mathsf{guard}\left(x, b_1, \langle a_1\rangle_{y_1}, \mathsf{guard}\left(y_2, b_2, \langle a_2\rangle_{z_1}, \langle a_3\rangle_{z_2}\right)\right)$$

with the difference that, with the guard construction, the formulae associated to all nodes above x now have an extra true conjunct (inherited from $\varphi(x)$, that is true).

For finite pomsets, we define:

$$\mathsf{guard}(\ell, \boldsymbol{\alpha}, \boldsymbol{\beta}) \triangleq \left\{ \mathsf{guard}(x, \ell, \alpha, \beta) \;\middle|\; \begin{array}{c} \alpha \in \boldsymbol{\alpha},\ \beta \in \boldsymbol{\beta}, \\ N_\alpha \cap N_\beta = \emptyset, \\ x \notin N_\alpha \cup N_\beta \end{array} \right\}$$

Monotonicity for the guarding operation holds.

Lemma 7 (Monotonicity of guard)**.** *If $\boldsymbol{\alpha} \sqsubseteq_{\mathit{pom}} \boldsymbol{\alpha}'$ and $\boldsymbol{\beta} \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta}'$, then $\mathsf{guard}(\ell, \boldsymbol{\alpha}, \boldsymbol{\beta}) \sqsubseteq_{\mathit{pom}} \mathsf{guard}(\ell, \boldsymbol{\alpha}', \boldsymbol{\beta}')$.*

For infinite pomsets, guarded branching is defined as:

$$\mathsf{guard}(\ell, \boldsymbol{\alpha}, \boldsymbol{\beta}) \triangleq \sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{guard}(\ell, \boldsymbol{\alpha}', \boldsymbol{\beta}')$$

Corollary 2. *guard is Scott continuous.*

Proof. Immediate from Lemmas 6 and 7. □

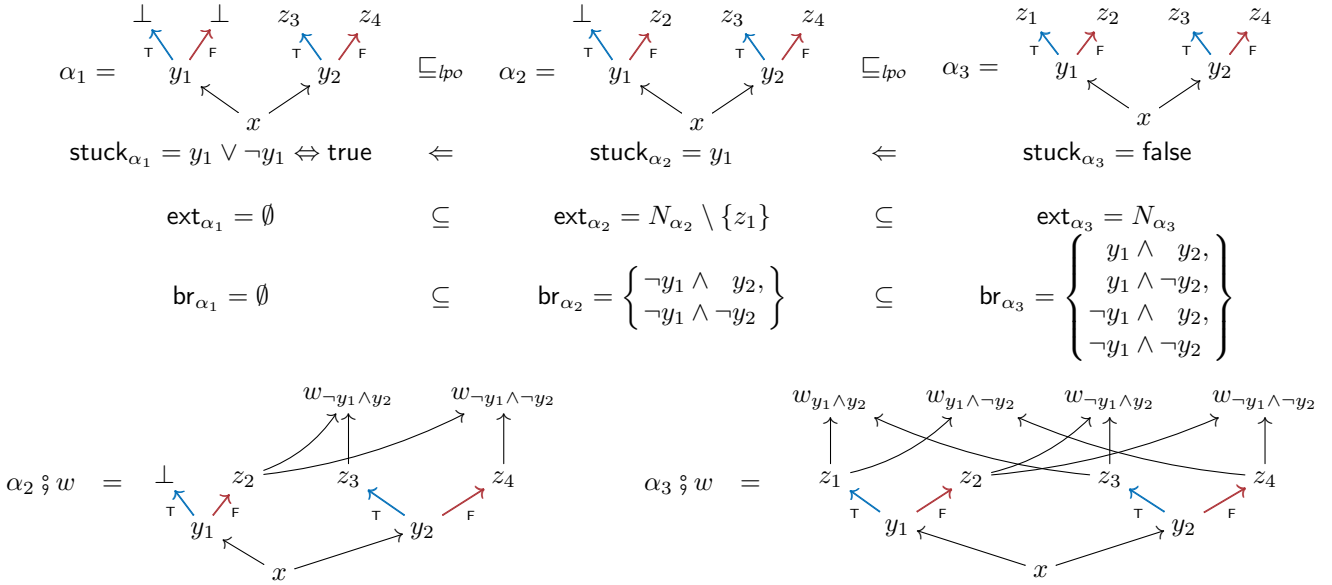


Fig. 2. An example of sequential composition, where $w \xrightarrow{\mathsf{T}} w'$ means that $\varphi(w') \Rightarrow w$ and $w \xrightarrow{\mathsf{F}} w''$ means that $\varphi(w'') \Rightarrow \neg w$ (*i.e.,* this means that w is labeled with a test, whose outcome leads to w' or w'', depending on the Boolean outcome depicted on the colored arc).

B. Sequential Composition

Sequential composition $\alpha \fatsemi \beta$ requires copying β for each maximal formula arising from the execution of α. This copying is essential to correctly account for loops: in the program **while** b **{**a**}** ; a', if we only created a single a' node, then that node would not be finitely preceded; we refer to Figure 4 (and the discussion on it) and to Appendix A for a more detailed account.

We first define sequential composition for finite LPOFs, and then extend our definition to infinite ones using the Extension Lemma (Lemma 6). In the rest of this section, we assume that $\alpha, \beta \in \mathit{lpo}_{\mathsf{fin}}(L)$. We start by defining the notion of a stuck computation, which is a formula indicating which nodes cannot continue to execute, and the set of extensible nodes, which contains those nodes that are not stuck:

$$\mathsf{stuck}_\alpha \triangleq \bigvee_{x \in \mathsf{Bot}_\alpha} \varphi_\alpha(x) \qquad \mathsf{ext}_\alpha \triangleq \{x \in N_\alpha \mid \varphi_\alpha(x) \not\Rightarrow \mathsf{stuck}_\alpha\}$$

Intuitively, $x \in \mathsf{ext}_\alpha$ if there is some computation path that it can take without encountering any $\bot$ nodes. Now, the set of branches contains all of the maximal formulae that can be obtained as the conjunction of formulae of extensible nodes:

$$\mathsf{br}_\alpha \triangleq \left\{ \varphi_\alpha(S) \;\middle|\; \begin{array}{l} \emptyset \subset S \subseteq \mathsf{ext}_\alpha,\ \varphi_\alpha(S) \Rightarrow \neg\mathsf{stuck}_\alpha, \\ \forall T.\ S \subset T \subseteq \mathsf{ext}_\alpha \Rightarrow \neg\mathsf{sat}(\varphi_\alpha(T)) \end{array} \right\}$$

where, for a set $S \subseteq N_\alpha$, we define $\varphi_\alpha(S) \triangleq \bigwedge_{x \in S} \varphi_\alpha(x)$. Since α is finite, then $\mathsf{ext}_\alpha \subseteq N_\alpha$ is finite, and so is S. Note that the formulae of br_α are all pairwise unsatisfiable, otherwise the maximality condition would be violated. The set of branches increases monotonically with the LPOF.

Lemma 8 (Monotonicity of branches)**.** *If* $\alpha \sqsubseteq_{lpo} \beta$*, then* $\mathsf{br}_\alpha = \{\psi \in \mathsf{br}_\beta \mid \psi \Rightarrow \neg\mathsf{stuck}_\alpha\}$.

Now, we will need an isomorphic copy $\beta_\psi \equiv \beta$ for each branch $\psi \in \mathsf{br}_\alpha$. These copies will be generated by a function $f \colon \mathsf{br}_\alpha \to [\beta]$ such that the nodes of $f(\psi)$ are disjoint from those of α and of every $f(\psi')$ where $\psi \neq \psi'$. The function f is drawn from the following set:

$$\mathsf{copy}_{\alpha,\beta} \triangleq \left\{ f \colon \mathsf{br}_\alpha \to [\beta] \;\middle|\; \begin{array}{l} \forall \psi \in \mathsf{br}_\alpha.\ (N_{f(\psi)} \cap N_\alpha = \emptyset \wedge \\ \forall \psi' \neq \psi. N_{f(\psi)} \cap N_{f(\psi')} = \emptyset) \end{array} \right\}$$

Given any $\alpha, \beta \in \mathit{lpo}_{\mathsf{fin}}(L)$ and $f \in \mathsf{copy}_{\alpha,\beta}$, we define sequential composition as

$$\alpha \fatsemi_f \beta = \langle N, <, \lambda, \varphi \rangle$$

where $\beta_\psi \triangleq f(\psi)$ and the components are defined as follows:

$$N \triangleq N_\alpha \cup \bigcup_{\psi \in \mathsf{br}_\alpha} N_{\beta_\psi}$$

$$< \triangleq <_\alpha \cup \bigcup_{\psi \in \mathsf{br}_\alpha} (<_{\beta_\psi} \cup (\{x \in N_\alpha \mid \psi \Rightarrow \varphi_\alpha(x)\} \times N_{\beta_\psi}))$$

$$\lambda(x) \triangleq \begin{cases} \lambda_\alpha(x) & \text{if } x \in N_\alpha \\ \lambda_{\beta_\psi}(x) & \text{if } x \in N_{\beta_\psi} \end{cases}$$

$$\varphi(x) \triangleq \begin{cases} \varphi_\alpha(x) & \text{if } x \in N_\alpha \\ \varphi_{\beta_\psi}(x) \wedge \psi & \text{if } x \in N_{\beta_\psi} \end{cases}$$

Example 2. In Figure 2 we give three LPOFs and, for each of them, we provide the stuck nodes, the extensible ones, the branches, and the result of sequentially composing them with the singleton LPOF w. To lighten notation, we assume that α_1, α_2 and α_3 have the same nodes and we only show the $\bot$ labels (associated to z_1 and z_2 in α_1 and just to z_1 in α_2). In α_3, there are no stuck nodes, since all maximal elements are non-$\bot$; hence, all nodes are extensible and the branches include all the possible Boolean combinations of y_1 and y_2 (this requires 4 copies of w when building $\alpha_3 \fatsemi w$). In α_2 the only stuck node is z_1, arising when y_1 is true; all other nodes

are extensible but the only branches to be considered are those in which y_1 is not true (thus, we only need two copies of w here). Finally, in α_1 all nodes are stuck, since the leftmost branch will always lead to $\bot$ and so the whole LPOF cannot be sequentially composed with anything else; for this reason, α_1 has no extensible node and no branch (thus, $\alpha_1 \fatsemi w = \alpha_1$).

For finite pomsets, we define:

$$\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \triangleq \{\alpha \fatsemi_f \beta \mid \alpha \in \boldsymbol{\alpha},\ \beta \in \boldsymbol{\beta},\ f \in \mathsf{copy}_{\alpha,\beta}\}$$

Lemma 9 (Monotonicity of $\fatsemi$)**.** *If $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\alpha}'$ and $\boldsymbol{\beta} \sqsubseteq_{pom} \boldsymbol{\beta}'$, then $\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \sqsubseteq_{pom} \boldsymbol{\alpha}' \fatsemi \boldsymbol{\beta}'$*

For infinite pomsets, sequential composition is defined as:

$$\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \triangleq \sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \boldsymbol{\alpha}' \fatsemi \boldsymbol{\beta}'$$

Corollary 3. *$\fatsemi$ is Scott continuous.*

Proof. Immediate from Lemmas 6 and 9. □

Remark 2. We can now explain Condition (2c) in Definition 7, requiring single-rootedness, which may seem strange in a framework with concurrency; indeed, the usual semantics of two commands put in parallel is obtained by taking the (disjoint) union of their pomsets (and this yields several possible minimum elements in the resulting pomset). However, having multi-rooted LPOFs would make sequential composition not monotone. This is shown by the following example:

$y \sqsubseteq_{lpo} y \quad z$ but $\begin{matrix} y \\ \uparrow \\ x \end{matrix} \not\sqsubseteq_{lpo} \begin{matrix} y & & z \\ & \nwarrow\ \nearrow & \\ & x & \end{matrix}$

We now present a simple yet very reasonable way to model parallel composition with single-rooted LPOFs.

C. Parallel Composition

We finally define a notion of parallel composition; to this aim, we assume that L contains a special (non-$\bot$) label fork that denotes thread forking and denote with nFork_α the set of nodes of α, without its fork-minimal elements (if any):

$$\mathsf{nFork}_\alpha \triangleq N_\alpha \setminus \{x \in \min_\alpha \mid \lambda_\alpha(x) = \mathsf{fork}\}$$

Then, for both finite and infinite LPOFs, we let:

$$\alpha \parallel_x \beta = \langle N, <, \lambda, \varphi \rangle$$

where $x \notin N_\alpha \cup N_\beta$, $N \triangleq \{x\} \cup \mathsf{nFork}_\alpha \cup \mathsf{nFork}_\beta$, and

$$\begin{aligned} < &\triangleq (\{x\} \times (\mathsf{nFork}_\alpha \cup \mathsf{nFork}_\beta)) \\ &\cup\ (<_\alpha \cap (\mathsf{nFork}_\alpha \times \mathsf{nFork}_\alpha)) \\ &\cup\ (<_\beta \cap (\mathsf{nFork}_\beta \times \mathsf{nFork}_\beta)) \end{aligned}$$

$$\lambda(y) \triangleq \begin{cases} \mathsf{fork} & \text{if } y = x \\ \lambda_\alpha(y) & \text{if } y \in \mathsf{nFork}_\alpha \\ \lambda_\beta(y) & \text{if } y \in \mathsf{nFork}_\beta \end{cases}$$

$$\varphi(y) \triangleq \begin{cases} \mathsf{true} & \text{if } y = x \\ \varphi_\alpha(y) & \text{if } y \in \mathsf{nFork}_\alpha \\ \varphi_\beta(y) & \text{if } y \in \mathsf{nFork}_\beta \end{cases}$$

Notice that the branching that arises from $\parallel$ is conceptually (and practically) different from the branching that arises from a test: the branching due to $\parallel$ does not exclude any branch, whereas the two branches due to a test are mutually exclusive. This is apparent by the different way in which we handle the formulae associated to the nodes after the branch: they remain the same under parallel composition, whereas they are extended with a new conjunct (specifying the value of the test at the branch) under a guard.

The complication arising in our handling of parallel composition through fork is that we want one single node labelled with fork, even if we put in parallel many LPOFs. Consider:

$\alpha = \begin{matrix} y_1 & & y_2 \\ & \nwarrow\ \nearrow & \\ & \mathsf{fork} & \end{matrix} \qquad \beta = \begin{matrix} z_1 & z_2 & z_3 \\ & \nwarrow\ \uparrow\ \nearrow & \\ & \mathsf{fork} & \end{matrix}$

resulting by putting in parallel the singleton LPOFs made up, respectively, by nodes y_1 and y_2 and by nodes z_1, z_2 and z_3 (and where fork is the label of the root nodes, say x_1 and x_2 respectively). Now, $\alpha \parallel_x \beta$ is the LPOF

$\begin{matrix} y_1 & y_2 & z_1 & z_2 & z_3 \\ & \nwarrow & \uparrow & \nearrow & \\ & & \mathsf{fork} & & \end{matrix}$

where x_1 and x_2 disappear and the new root is x, labelled fork. As a side note, observe that the way in which we defined parallel composition entails that $\alpha \sqsubseteq_{lpo} \alpha \parallel_x \beta$ if and only if $\alpha = \langle \bot \rangle_x$: the first LPOF has just node x labelled with $\bot$; the second LPOF has node x labelled with fork that proceeds a node labelled with $\bot$ and all the nodes of β. Also, a somewhat unusual notion of associativity for parallel holds:

$$(\alpha \parallel_x \beta) \parallel_y \gamma = \alpha \parallel_y (\beta \parallel_z \gamma)$$

The expectable notion of associativity is recovered for pomsets, where we abstract from the specific node set[2]. Like before, we define parallel composition for pomsets on top of that for LPOFs:

$$\boldsymbol{\alpha} \parallel \boldsymbol{\beta} \triangleq \{\alpha \parallel_x \beta \mid \alpha \in \boldsymbol{\alpha},\ \beta \in \boldsymbol{\beta},\ x \notin N_\alpha \cup N_\beta\}$$

Remark 3. Unlike the previous pomset operations, $\parallel$ is *not* monotone (not for LPOFs nor for pomsets); indeed, consider

$\boldsymbol{\alpha} = \langle \bot \rangle \qquad \boldsymbol{\beta} = \begin{matrix} \ell_1 & & \ell_2 \\ & \nwarrow\ \nearrow & \\ & \mathsf{fork} & \end{matrix} \qquad \boldsymbol{\gamma} = \langle \ell \rangle$

Clearly, $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\beta}$ but

$\boldsymbol{\alpha} \parallel \boldsymbol{\gamma} = \begin{matrix} \bot & & \ell \\ & \nwarrow\ \nearrow & \\ & \mathsf{fork} & \end{matrix} \not\sqsubseteq_{pom} \begin{matrix} \ell_1 & \ell_2 & \ell \\ & \nwarrow\ \uparrow\ \nearrow & \\ & \mathsf{fork} & \end{matrix} = \boldsymbol{\beta} \parallel \boldsymbol{\gamma}$

This is not surprising: if $\parallel$ were monotone, then the function $f(\boldsymbol{\alpha}) = \langle \ell \rangle \parallel \boldsymbol{\alpha}$ would have a fixed point, but that fixed point would be an infinitely branching structure, which is invalid.

[2]We remark that $\fatsemi$ on LPOFs also satisfies a non-straightforward associativity, that however becomes the usual one when passing to pomsets.

$$
\begin{aligned}
[\![\mathbf{skip}]\!] &\triangleq \langle \mathsf{fork} \rangle \\
[\![C_1; C_2]\!] &\triangleq [\![C_1]\!] \fatsemi [\![C_2]\!] \\
[\![C_1 \mid C_2]\!] &\triangleq [\![C_1]\!] \parallel [\![C_2]\!] \\
[\![\mathbf{if}\ b\ \{C_1\}\ \mathbf{else}\ \{C_2\}]\!] &\triangleq \mathsf{guard}(b, [\![C_1]\!], [\![C_2]\!]) \\
[\![\mathbf{while}\ b\ \{C\}]\!] &\triangleq \mathsf{lfp}\left(\Phi_{\langle C,b\rangle}\right) \\
[\![a]\!] &\triangleq \langle a \rangle
\end{aligned}
$$

Fig. 3. Pomset semantics of commands $[\![-]\!] : \mathit{cmd} \to \mathit{pom}$.

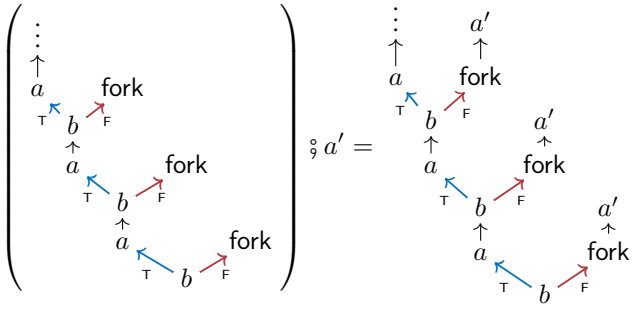


Fig. 4. Semantics of the program $\mathbf{while}\ b\ \{a\}\ ;\ a'$.

IV. Denotational Semantics

We will now define denotational semantics for a basic imperative programming language based on the pomset structure that we defined in the previous sections. Although actions at this stage are uninterpreted, our pomset structure allows for actions with a variety of different *computational effects*, which we will see in Section IV-B. The syntax for program commands $C \in \mathit{cmd}$ is shown in Figure 1.

The atomic actions $a \in \mathit{act}$ are drawn from a DCPO $\langle \mathit{act}, \leq_{\mathit{act}} \rangle$ and tests $b \in \mathit{test}$ are drawn from a set test (with a flat order), which is closed under conjunction, disjunction, and negation. We will first define a compositional denotational model for this language in Section IV-A, and then in Section IV-B we will show how to interpret pomsets as state transformers, so as to learn the input-output behavior of programs with particular effects.

A. Pomset Semantics

We will now discuss the semantics of programs $C \in \mathit{cmd}$ as pomsets, which record both the causality between atomic actions and the branching behavior of tests. We will use a set of labels consisting of actions, tests, fork, and bottom nodes, which forms a pointed DCPO, with $\bot$ as the bottom. Actions are ordered according to $\leq_{\mathit{act}}$; for any other label $\ell \in \mathit{test} \cup \{\mathsf{fork}\}$, we only have $\bot \leq \ell$ and $\ell \leq \ell$. We will also henceforth use pom to refer to pomsets over this label set.

$$
\mathit{label} \triangleq \mathit{act} \cup \mathit{test} \cup \{\mathsf{fork}, \bot\} \qquad \mathit{pom} \triangleq \mathit{pom}(\mathit{label})
$$

The semantic function $[\![-]\!] : \mathit{cmd} \to \mathit{pom}$, which maps a command into a pomset, is shown in Figure 3. Its definition is straightforward using the pomset operations: $\mathbf{skip}$, $C_1; C_2$, and $C_1 \mid C_2$ are interpreted as a singleton fork, sequential composition, and parallel composition, respectively.

While-loops are interpreted as the least fixed point of the characteristic function $\Phi_{\langle C,b\rangle} : \mathit{pom} \to \mathit{pom}$, defined below:

$$
\Phi_{\langle C,b\rangle}(\boldsymbol{\alpha}) \triangleq \mathsf{guard}(b, [\![C]\!] \fatsemi \boldsymbol{\alpha}, [\![\mathbf{skip}]\!])
$$

It is clear that $\Phi_{\langle C,b\rangle}$ is Scott continuous, as it is defined using guard and $\fatsemi$, both of which are Scott continuous (Corollaries 2 and 3). Therefore, by Kleene's fixed point theorem [30, Theorem 2.1.19], $\Phi_{\langle C,b\rangle}$ has a least fixed point given by:

$$
\mathsf{lfp}\left(\Phi_{\langle C,b\rangle}\right) = \sup_{n \in \mathbb{N}} \Phi^n_{\langle C,b\rangle}\left(\bot_{\mathit{pom}}\right)
$$

where $f^0 \triangleq \mathsf{id}$ and $f^{n+1} \triangleq f \circ f^n$. In Figure 4, we show $[\![\mathbf{while}\ b\ \{a\}\ ;\ a']\!]$, containing a loop with a single action in the body. The pomset for the while-loop has a terminating branch for each finite execution, and an infinitely ascending spine corresponding to the case where b never becomes false. Indeed, it is obtained as the supremum of the chain formed by its finite unrollings, which we show in Appendix A.

As discussed in Section III-B, sequentially composing another action a' after this loop results in a copy of a' occurring after each branch, shown on the right of Figure 4. If we would instead attempt to use a single copy of a' as the successor of the entire loop, then it would not be finitely preceded, causing many problems; we refer to Appendix A for a more complete discussion of the technical issues that arise.

The semantics obeys a binary branching property, guaranteeing that test nodes branch into exactly two successors, one where the test passes and another where it fails. Further, variables can only be introduced into formulae after a test. From now on, we assume that all pomsets have this property, which is clearly preserved by the operations of Section III.

Definition 12 (Binary Branching)**.** *A pomset $\boldsymbol{\alpha} \in \mathit{pom}$ has the* binary branching property *if, for every $\alpha \in \boldsymbol{\alpha}$ and every $x \in N_\alpha$ such that $\lambda_\alpha(x) \in \mathit{test}$, we have that $\mathsf{succ}_\alpha(x) = \{y_1, y_2\}$ such that:*

1) $\varphi_\alpha(y_1) = \varphi_\alpha(x) \wedge x$;
2) $\varphi_\alpha(y_2) = \varphi_\alpha(x) \wedge \neg x$; *and*
3) $\mathsf{pred}_\alpha(y_1) = \mathsf{pred}_\alpha(y_2) = \{x\}$

In addition, for every $x \in N_\alpha$ that is not the successor of a test, then $\varphi_\alpha(x) \Leftrightarrow \bigwedge_{y \in \mathsf{pred}_\alpha(x)} \varphi_\alpha(y)$.

B. Linearization

We now discuss how to produce a state transformer from the pomset semantics of Section IV-A, which is useful for understanding the input-output behavior of programs resulting from scheduling concurrent threads. This state transformer is obtained via *linearization*, which consists of considering all interleavings of the threads in order to interpret the semantics of a *complete* program as a function from inputs to collections of outputs. Once the program is linearized, more threads cannot be composed in parallel, as the causality information is gone. Nevertheless, linearization is useful to understand the

$$\mathsf{next}(\alpha,\psi,S) \triangleq \{x \in N_\alpha \setminus S \mid x{\downarrow_\alpha} \subseteq S, \psi \Rightarrow \varphi_\alpha(x)\}$$
$$\mathcal{L}_{lpo}(\alpha,\psi,S)(s) \triangleq \eta(s) \quad \text{if } \mathsf{next}(\alpha,\psi,S) = \emptyset$$
$$\mathcal{L}_{lpo}(\alpha,\psi,S)(s) \triangleq \mathop{\&}_{x \in \mathsf{next}(\alpha,\psi,S)} \begin{cases} \mathcal{L}_{lpo}\left(\alpha,\psi,S\cup\{x\}\right)^\dagger\left(\llbracket\lambda_\alpha(x)\rrbracket_{act}\left(s\right)\right) & \text{if } \lambda_\alpha(x) \in act \\ \mathcal{L}_{lpo}(\alpha,\psi \wedge (\!|x = \llbracket\lambda_\alpha(x)\rrbracket_{test}(s)|\!), S\cup\{x\})(s) & \text{if } \lambda_\alpha(x) \in test \\ \bot_D & \text{if } \lambda_\alpha(x) = \bot \\ \mathcal{L}_{lpo}(\alpha,\psi,S\cup\{x\})(s) & \text{if } \lambda_\alpha(x) = \mathsf{fork} \end{cases}$$

Fig. 5. Linearization for finite LPOFs $\mathcal{L}_{lpo} \colon lpo_{\mathsf{fin}}(label) \times form \times \mathcal{P}(nodes) \to \mathcal{S} \to D(\mathcal{S})$.

program's behavior, and relate the behavior to other semantic models, as we will see in Sections V and VI.

When performing linearization, we will interpret programs in a computational domain D, which must support nondeterminism—in order to model the different ways that threads can be interleaved—but may support additional computational effects too, which we will model using monads. Hence, we assume some familiarity with basic notions of monads and Kleisli categories; for a detailed introduction, we refer to [31]. More precisely, we need the following:

1) A domain of computation D and set of states $\mathcal{S}$, such that $\langle D(\mathcal{S}), \sqsubseteq\rangle$ is a pointed DCPO with bottom $\bot_D$.
2) An associative, commutative, and monotone nondeterminism operator $\& \colon D(\mathcal{S})^2 \to D(\mathcal{S})$.
3) An additive monad in the category **DCPO** $\langle D, \eta, (-)^\dagger\rangle$, meaning that the operations have type:

   $$\eta \colon X \to D(X) \quad (-)^\dagger \colon (X \to D(Y)) \to D(X) \to D(Y)$$

   The monad laws hold:

   $$f^\dagger \circ \eta = f \qquad \eta^\dagger = \mathsf{id} \qquad (g^\dagger \circ f)^\dagger = g^\dagger \circ f^\dagger$$

   The Kleisli extension is Scott Continuous and strict:

   $$\sup_{f\in D}\sup_{d\in D'} f^\dagger(d) = (\sup D)^\dagger(\sup D') \qquad f^\dagger(\bot) = \bot$$

   and it is additive [32], [33] with respect to $\&$:

   $$f^\dagger(d \,\&\, d') = f^\dagger(d) \,\&\, f^\dagger(d')$$

4) An interpretation function for actions and for tests:

   $$\llbracket - \rrbracket_{act} : act \to \mathcal{S} \to D(\mathcal{S}) \qquad \llbracket - \rrbracket_{test} : test \to \mathcal{S} \to \mathbb{B}$$

   s.t. $\llbracket - \rrbracket_{act}$ is monotone: $\llbracket a \rrbracket_{act}(s) \sqsubseteq \llbracket a' \rrbracket_{act}(s)$ if $a \leq_{act} a'$.

There are many examples of computational domains obeying these properties, depending on which *computational effects* are present. This includes the Hoare, Smyth, and Plotkin powerdomains for nondeterministic computation [25], [26], [30], where the latter two additionally deal with nontermination. It also includes the convex powerset [29], [34] and powerdomain of indexed valuations [19], [35], [36] for mixed probabilistic and nondeterministic computation; and other domains for nondeterminism with exceptions [32], [33]. In Sections V and VI, we will explore how standard semantics in two of those domains can be recovered from our pomset semantics. Note that for any set X, the pointwise extension $\langle X \to D(\mathcal{S}), \sqsubseteq^\bullet\rangle$ is also a pointed DCPO, where $f \sqsubseteq^\bullet g$ iff $f(x) \sqsubseteq g(x)$ for all $x \in X$.

We will start by defining a linearization operation on finite LPOFs, which is shown in Figure 5. Linearization is defined recursively, until there are no more nodes to schedule. It must be defined on finite structures, otherwise the recursion is not well-founded, but we can extend linearization to infinite structures using the extension lemma. In $\mathcal{L}_{lpo}(\alpha,\psi,S)$, the set $S \subseteq N_\alpha$ contains all the nodes that have already been processed and ψ is a path condition, indicating the outcomes of the tests associated to the nodes in S.

The function $\mathsf{next}(\alpha,\psi,S)$ gives the nodes that are ready to be scheduled, which includes all nodes that obey the path condition ψ, and whose predecessors are in S. If $\mathsf{next}(\alpha,\psi,S)$ is empty, then $\mathcal{L}_{lpo}(\alpha,\psi,S)$ simply returns the current state s using the monad unit η. If not, then it nondeterministically selects a next node, where $\&_{i\in I}\, d_i \triangleq d_{i_1} \,\&\, \cdots \,\&\, d_{i_n}$, for any finite index set $I = \{i_1,\ldots,i_n\}$. If the next node is an action, then it interprets the action using $\llbracket - \rrbracket_{act}$, and then uses Kleisli composition to compose the result with the linearization of the remaining LPOF. If the next node is a test, then that test is evaluated and the result is added to the path condition, where $(\!|x = 1|\!) \triangleq x$ and $(\!|x = 0|\!) \triangleq \neg x$. If the next node is $\bot$, then the linearization is $\bot_D$, and fork is treated like a no-op.

We next use $\mathcal{L}_{lpo}$ to define linearization on finite pomsets $\mathcal{L}_{\mathsf{fin}} \colon pom_{\mathsf{fin}}(label) \to \mathcal{S} \to D(\mathcal{S})$. Clearly $\mathcal{L}_{lpo}(\alpha,\mathsf{true},\emptyset) = \mathcal{L}_{lpo}(\beta,\mathsf{true},\emptyset)$ if $\alpha \equiv \beta$, so $\mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha})$ can be defined for any arbitrary representative LPOF $\alpha \in \boldsymbol{\alpha}$. Finally, linearization is extended to infinite pomsets $\mathcal{L} \colon pom \to \mathcal{S} \to D(\mathcal{S})$ by taking the supremum over all of the finite approximations.

$$\mathcal{L}_{\mathsf{fin}}([\alpha]) \triangleq \mathcal{L}_{lpo}(\alpha,\mathsf{true},\emptyset) \qquad \mathcal{L}(\boldsymbol{\alpha}) \triangleq \mathcal{L}^*_{\mathsf{fin}}(\boldsymbol{\alpha})$$

Linearization over finite pomsets is monotonic, and therefore the extension is well-defined and Scott continuous.

Lemma 10 (Monotonicity of $\mathcal{L}_{\mathsf{fin}}$)**.** *If $\boldsymbol{\alpha} \sqsubseteq_{lpo} \boldsymbol{\beta}$, then $\mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha}) \sqsubseteq^\bullet \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\beta})$.*

Corollary 4. *$\mathcal{L} \colon pom \to (\mathcal{S} \to D(\mathcal{S}))$ is Scott continuous.*

Proof. Immediate from Lemmas 6 and 10. □

C. Properties of Linearization

To ensure that linearization gives us a good semantics, we now provide some sanity check lemmas. First, linearizing

skip gives the monad unit and linearizing a singleton pomset has the same behavior as interpreting the single action.

Lemma 11. $\mathcal{L}([\![\textbf{skip}]\!]) = \eta$ *and* $\mathcal{L}([\![a]\!]) = [\![a]\!]_{act}$.

Next, linearizing a sequential composition is equal to performing Kleisli composition on the individual linearizations.

Lemma 12. $\mathcal{L}([\![C_1; C_2]\!]) = \mathcal{L}([\![C_2]\!])^\dagger \circ \mathcal{L}([\![C_1]\!])$.

Linearizing an if-statement is equal to linearizing one of the two branches, depending on the truth of the guard.

Lemma 13.

$$\mathcal{L}([\![\textbf{if } b \ \{C_1\} \textbf{ else } \{C_2\}]\!])(s) = \begin{cases} \mathcal{L}([\![C_1]\!])(s) & \text{if } [\![b]\!]_{test}(s) = 1 \\ \mathcal{L}([\![C_2]\!])(s) & \text{if } [\![b]\!]_{test}(s) = 0 \end{cases}$$

Finally, the linearization of a while-loop is defined as a least fixed point over a different characteristic function $\Psi_{\langle f,b\rangle}$. The function $\Psi_{\langle f,b\rangle}$ inherits Scott continuity from the Kleisli composition operator, therefore the fixed point exists.

Lemma 14. $\mathcal{L}([\![\textbf{while } b \ \{C\}]\!]) = \mathsf{lfp}\left(\Psi_{\langle \mathcal{L}([\![C]\!]),b\rangle}\right)$ *where*

$$\Psi_{\langle f,b\rangle}(g)(s) \triangleq \begin{cases} g^\dagger(f(s)) & \text{if } [\![b]\!]_{test}(\sigma) = 1 \\ \eta(s) & \text{if } [\![b]\!]_{test}(\sigma) = 0 \end{cases}$$

V. The Hoare Powerdomain Instantiation

Powerdomains are commonly used to give semantics to programs that combine nondeterminism with looping or recursion [25], [26]. In this section, we create a powerdomain instantiation of our semantics from Section IV and prove that the resulting model corresponds to well-known models of non-probabilistic concurrent programs [27], [28].

In particular, we work in the *Hoare* or *lower* powerdomain, named as such for its connection to partial correctness and Hoare Logic [37], [38]. Indeed, the Hoare powerdomain identifies the terminating traces of a program, but not that the program *always* terminates. From a concurrency perspective, this corresponds to *safety* properties [39]. For simplicity, we presume that the order over actions $\leq_{act}$ in this section is flat.

A. Pomset Language Semantics

Since typical pomsets do not contain formulae [9], [12], [40], they cannot encode control flow branching introduced by if-statements and while-loops. Concurrent program semantics therefore typically use *pomset languages*—sets of pomsets—where each individual pomset contains only the actions that occur in a single branch of computation [27], [28].

We emulate pomset language semantics through pomsets with formulae. First, we define a label set $\mathit{label}_{\mathsf{PL}} \triangleq \mathit{act}_{\mathsf{PL}} \cup \mathit{test}_{\mathsf{PL}} \cup \{\mathsf{fork}, \bot\}$, whose components are as before, but now $\mathit{test}_{\mathsf{PL}} \triangleq \emptyset$, and instead $\mathit{act}_{\mathsf{PL}} \triangleq \mathit{act} \cup \{\mathsf{assume}\ b \mid b \in \mathit{test}\}$ includes an action $\mathsf{assume}\ b$ for every test $b \in \mathit{test}$, which guarantees that a certain condition holds in the current branch. Pomset languages are sets of finite pomsets over $\mathit{label}_{\mathsf{PL}}$:

$$\mathit{pom}\mathcal{L}\mathit{ang} \triangleq \mathcal{P}(\mathit{pom}_{\mathsf{fin}}(\mathit{label}_{\mathsf{PL}}))$$

The pomset language semantics is shown in Figure 6, where each $\boldsymbol{\alpha} \in [\![C]\!]_{\mathsf{PL}}$ corresponds to a particular sequence of test resolutions. Whenever an assume fails, the trace is eliminated. When a branch occurs, *i.e.,* in an if-statement or while-loop, the set of traces is duplicated to account for both the "true" and "false" branches. Loops are interpreted as the least fixed point of $\Xi_{\langle C,b\rangle}$, which is Scott continuous in the subset $\subseteq$ order, since suprema (given by $\cup$) are associative and commutative. The **skip** appears in the false branch to correspond to $\Phi_{\langle C,b\rangle}$.

Although the semantics is defined using pomsets with tests, $\mathit{test}_{\mathsf{PL}} = \emptyset$, so the formula of every node in $\boldsymbol{\alpha}$ is true. In addition, all pomsets are finite and $\bot$ nodes are never introduced, meaning that $\boldsymbol{\alpha}$ consists only of actions and fork nodes. We therefore get that the definition sequential composition is the standard one, *e.g.,* [9], [24], [40], [41].

Lemma 15. *If* $\mathsf{Bot}_\alpha = \emptyset$, $\varphi_\alpha(x) = \mathsf{true}$ *for all* $x \in N_\alpha$, *and* $f = [\mathsf{true} \mapsto \beta]$, *then:*

$$N_{\alpha ;_f \beta} = N_\alpha \cup N_\beta \quad \textit{and} \quad <_{\alpha ;_f \beta} = <_\alpha \cup <_\beta \cup (N_\alpha \times N_\beta)$$

B. The Hoare Powerdomain and Linearization

The Hoare powerdomain consists of sets of states, ordered by subset inclusion $\langle \mathcal{P}(\mathcal{S}), \subseteq\rangle$, which is known to be a pointed DCPO with supremum given by union $\cup$ and $\emptyset$ as bottom. We also let $\& \triangleq \cup$ and the define the monad operations as $\eta(s) \triangleq \{s\}$ and $f^\dagger(S) \triangleq \bigcup_{s\in S} f(s)$. These operations are well known to be Scott continuous and additive [32]. Accordingly, action evaluation $[\![-]\!]_{act_{\mathsf{PL}}} : \mathit{act}_{\mathsf{PL}} \to \mathcal{S} \to D(\mathcal{S})$ is given by:

$$[\![a]\!]_{act_{\mathsf{PL}}}(s) \triangleq \begin{cases} [\![a]\!]_{act}(s) & \text{if } a \in \mathit{act} \\ \eta(s) & \text{if } a = \mathsf{assume}\ b,\ [\![b]\!]_{test}(s) = 1 \\ \bot_D & \text{if } a = \mathsf{assume}\ b,\ [\![b]\!]_{test}(s) = 0 \end{cases}$$

So assume actions result in $\{s\}$, if the test is true in state s, and $\emptyset$, otherwise. We now define a specialized linearization operation $\mathcal{L}_{\mathsf{PL}} : \mathit{pom}\mathcal{L}\mathit{ang} \to \mathcal{S} \to \mathcal{P}(\mathcal{S})$, which is simply a union over all traces of the standard linearization.

$$\mathcal{L}_{\mathsf{PL}}(S)(s) \triangleq \bigcup_{\boldsymbol{\alpha}\in S} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha})(s)$$

Since all formulae are true, the next nodes to schedule are exactly those whose predecessors have all been processed, *i.e.,* $\mathsf{next}(\alpha, \mathsf{true}, S) = \{x \in N_\alpha \setminus S \mid x{\downarrow_\alpha} \subseteq S\}$. This corresponds to standard interleaving definitions of linearization [40].

C. Equivalence of Semantics

We now prove that pomset language semantics (Figure 6) corresponds exactly to the Hoare powerdomain instance of our semantic model from Section IV. First, we define a translation $\mathsf{tr} : \mathit{pom} \to \mathit{pom}\mathcal{L}\mathit{ang}$, which recovers a pomset language from a pomset with formulae. To define tr, we start with a translation on finite LPOFs, which additionally takes a formula ψ to encode the resolution of tests in the current trace.

$$\mathsf{tr}_{lpo}(\alpha, \psi) = \langle N_\psi, <_\psi, \lambda_\psi, [- \mapsto \mathsf{true}]\rangle$$

where $[- \mapsto \mathsf{true}]$ is the constant true function and

$$N_\psi \triangleq \{x \in N_\alpha \mid \psi \Rightarrow \varphi_\alpha(x)\} \qquad <_\psi \triangleq <_\alpha \cap (N_\psi \times N_\psi)$$

$$
\begin{aligned}
[\![\mathbf{skip}]\!]_{\mathsf{PL}} &\triangleq \{\langle \mathsf{fork}\rangle\} & [\![a]\!]_{\mathsf{PL}} &\triangleq \{\langle a\rangle\}\\
[\![C_1; C_2]\!]_{\mathsf{PL}} &\triangleq \{\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in [\![C_1]\!]_{\mathsf{PL}}, \boldsymbol{\beta} \in [\![C_2]\!]_{\mathsf{PL}}\}\\
[\![C_1 \mid C_2]\!]_{\mathsf{PL}} &\triangleq \{\boldsymbol{\alpha} \parallel \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in [\![C_1]\!]_{\mathsf{PL}}, \boldsymbol{\beta} \in [\![C_2]\!]_{\mathsf{PL}}\}\\
[\![\mathbf{if}\ b\ \{C_1\}\ \mathbf{else}\ \{C_2\}]\!]_{\mathsf{PL}} &\triangleq \{\langle \mathsf{assume}\ b\rangle \fatsemi \boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in [\![C_1]\!]_{\mathsf{PL}}\} \cup \{\langle \mathsf{assume}\ \neg b\rangle \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\beta} \in [\![C_2]\!]_{\mathsf{PL}}\}\\
[\![\mathbf{while}\ b\ \{C\}]\!]_{\mathsf{PL}} &\triangleq \mathsf{lfp}\left(\Xi_{\langle C,b\rangle}\right)
\end{aligned}
$$

$$
\text{where}\quad \Xi_{\langle C,b\rangle}(S) \triangleq \{\langle \mathsf{assume}\ b\rangle \fatsemi \boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in [\![C]\!]_{\mathsf{PL}}, \boldsymbol{\beta} \in S\} \cup \{\langle \mathsf{assume}\ \neg b\rangle \fatsemi [\![\mathbf{skip}]\!]\}
$$

Fig. 6. Standard pomset language semantics $[\![-]\!]_{\mathsf{PL}} : \mathit{cmd} \to \mathit{pomLang}$.

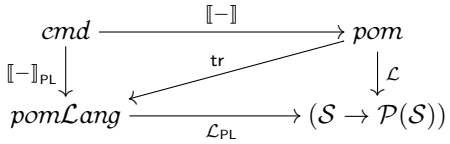
$$
\lambda_\psi(x) \triangleq \begin{cases} \lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \in \mathit{test} \text{ and } \psi \Rightarrow x \\ \mathsf{assume}\ \neg\lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \in \mathit{test} \text{ and } \psi \Rightarrow \neg x \end{cases}
$$

The nodes of $\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi)$ are those nodes $x \in N_\alpha$ such that $\psi \Rightarrow \varphi_\alpha(x)$, indicating that x agrees with the outcomes of the tests described by ψ. Test nodes with label b are converted into either $\mathsf{assume}\ b$ or $\mathsf{assume}\ \neg b$, depending on whether that node is true or false in ψ. We define translation on finite pomsets by taking a union over all branches $\psi \in \mathsf{br}_\alpha$. Finite translation is monotonic—$\mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}) \subseteq \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\beta})$ if $\boldsymbol{\alpha} \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta}$—therefore we can extend to infinite pomsets with the extension lemma.

$$
\mathsf{tr}_{\mathsf{fin}}([\alpha]) \triangleq \{[\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi)] \mid \psi \in \mathsf{br}_\alpha\} \qquad \mathsf{tr}(\boldsymbol{\alpha}) \triangleq \mathsf{tr}^*_{\mathsf{fin}}(\boldsymbol{\alpha})
$$

Using translation, we prove that the two semantics coincide.

Theorem 1 (Equivalence of Semantics)**.** *The following diagram commutes:*



The upper commuting triangle does not depend on the Hoare powerdomain, so it is tempting to say that pomset languages give an adequate model in other domains too. However, as we will see in Example 3, just because a semantic structure exists for a program does not mean it has the desired meaning.

Indeed, the fact that $\mathcal{L} = \mathcal{L}_{\mathsf{PL}} \circ \mathsf{tr}$ relies on two particular properties of the Hoare powerdomain. The first is that, if $h(x) = f(x) \cup g(x)$, then $h^\dagger(S) = f^\dagger(S) \cup g^\dagger(S)$. The analagous property (with $\cup$ replaced by $\&$) is invalid in many domains, particularly when probabilistic computation is present. The second property is that $\mathcal{L}_{\mathsf{PL}}$ is computed as a union over a possibly infinite set (the union is infinite whenever the program contains a while-loop). Unbounded nondeterminism is known to cause problems in many domains—including the Smyth and Plotkin powerdomains [42]–[44]—thus no infinitary version of $\&$ exists. So, although programs in any domain *can* be interpreted as pomset languages, those semantic objects may not give adequate meaning to the program. We will see this more concretely in the next section.

VI. A Probabilistic Concurrency Instantiation

In this section, we develop a probabilistic instantiation of our semantic model. Doing so requires a semantic domain that supports both probabilistic and nondeterministic choice, where nondeterminism models the choices of the scheduler in the linearization procedure. While there are several domains for this combination of effects, we use the *convex powerset*, which is quite well-studied in that it has monad [45] and DCPO [46]–[49] structures, and well behaved equational laws [50]–[52]. We now give a basic account of the convex powerset monad; for a full treatment, we refer to [8, §3].

A discrete probability distribution $\mu \in \mathcal{D}(X) = X \to [0, 1]$ is a map from elements of a set X to probabilities such that $\sum_{x \in X} \mu(x) = 1$. Convex combinations of distributions are defined as $(\mu \oplus_p \nu)(x) = p \cdot \mu(x) + (1 - p) \cdot \nu(x)$. A set of distributions $S \subseteq \mathcal{D}(X)$ is *convex* if $\mu \oplus_p \nu \in S$ for all $\mu, \nu \in S$ and $p \in [0, 1]$. Our computational domain consists of convex sets; however, the DCPO structure depends on a few more requirements, namely that it is upward and *Cauchy* closed [4], which means that it is closed in the product of Euclidean topologies [8]. Now, define the following set:

$$
\mathcal{C}(X) \triangleq \left\{ S \subseteq \mathcal{D}(X_\perp) \;\middle|\; \begin{array}{l} S \text{ is nonempty, convex, upward} \\ \text{closed, and Cauchy closed} \end{array} \right\}
$$

The nondeterminism operator is defined as

$$
S \,\&\, T \triangleq \{\mu \oplus_p \nu \mid \mu \in S, \nu \in T, p \in [0, 1]\}
$$

that is, an element from S or T may be chosen with any probability. Operationally, this correspond to a scheduler that can use randomness to resolve choices, rather than the more standard deterministic scheduler model [35].

The domain $\mathcal{C}$ has the necessary properties from Section IV-B, *i.e.,* it is a pointed DCPO, continuous additive monad, etc. [8]. Denotational semantics based on the parallel composition free fragment of the language in Figure 1 can be defined using $\mathcal{C}$ [4], [8], [29], and is shown in Figure 7, and we prove that it is equivalent to our pomset model.

Theorem 2. *For any program $C \in \mathit{cmd}$ that does not contain parallel composition: $\mathcal{L}([\![C]\!]) = [\![C]\!]_{\mathcal{C}}$.*

Proof. By induction on the structure of C. The cases follow directly from Lemmas 11 to 14. □

Some programming languages include constructs such as $C_1 \,\&\, C_2$ and $C_1 \oplus_p C_2$, which nondeterministically or probabilistically execute one of two commands. This can be encoded as a test node in our pomset structure, but linearization

$$
\begin{aligned}
[\![\texttt{skip}]\!]_{\mathcal{C}}(s) &\triangleq \eta(s) \\
[\![C_1; C_2]\!]_{\mathcal{C}}(s) &\triangleq [\![C_2]\!]^{\dagger}_{\mathcal{C}}([\![C_1]\!]_{\mathcal{C}}(s)) \\
[\![\texttt{if } b \ \{C_1\} \ \texttt{else} \ \{C_2\}]\!]_{\mathcal{C}}(s) &\triangleq \begin{cases} [\![C_1]\!]_{\mathcal{C}}(s) \text{ if } [\![b]\!]_{\mathit{test}}(s) = 1 \\ [\![C_2]\!]_{\mathcal{C}}(s) \text{ if } [\![b]\!]_{\mathit{test}}(s) = 0 \end{cases} \\
[\![\texttt{while } b \ \{C\}]\!]_{\mathcal{C}}(s) &\triangleq \mathsf{lfp}(\Psi_{\langle [\![C]\!]_{\mathcal{C}}, b\rangle})(s) \\
[\![a]\!]_{\mathcal{C}}(s) &\triangleq [\![a]\!]_{\mathit{act}}(s)
\end{aligned}
$$

Fig. 7. Convex powerset semantics $[\![-]\!]_{\mathcal{C}} : \mathit{cmd} \to \mathcal{S} \to \mathcal{C}(\mathcal{S})$, where Ψ is defined in Lemma 14.

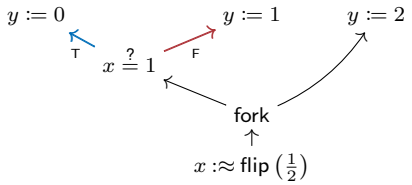
would need to be updated so that test evaluation has type $[\![-]\!]_{\mathit{test}} : \mathit{test} \to \mathcal{S} \to \mathcal{C}(\mathbb{B})$ to support effectful choices.

We now show why the pomset language semantics from Section V does not give an adequate model for probabilistic programs. For the following example, we presume that actions consist of deterministic assignments $x := n$ and (biased) coin flips $x :\approx \mathsf{flip}\,(p)$. The semantics of these actions are standard, as in previous works [8], [23].

Example 3. Let the program C be defined as follows:

$$C = x :\approx \mathsf{flip}\left(\tfrac{1}{2}\right); (\texttt{if } x \ \{y := 0\} \ \texttt{else} \ \{y := 1\} \,|\, y := 2)$$

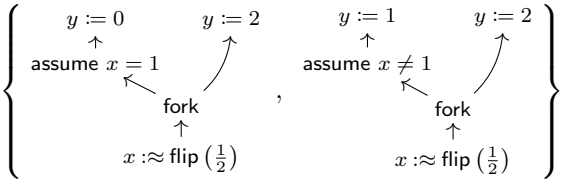
Our semantics based on pomsets with formulae gives the following model of the program, which records the causality between the actions regardless of how the coin flip is resolved:

$[\![C]\!] =$ 

Linearizing this structure, we get the following convex set, where the probabilities p and q are chosen by the scheduler.

$$\mathcal{L}([\![C]\!])(s) = \left\{ \begin{bmatrix} s[y := 0] \mapsto p \\ s[y := 1] \mapsto q \\ s[y := 2] \mapsto 1 - (p + q) \end{bmatrix} \middle| \ 0 \le p, q \le \frac{1}{2} \right\}$$

Linearization provides new insights about the program, which were not obvious in the pomset structure: the scheduler can force $y = 2$ to occur with any probability! Interpreting the program as a pomset language instead gives the following set:

$[\![C]\!]_{\mathsf{PL}} =$ 

Letting $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ be the left and right pomsets, respectively, their linearizations are as follows:

$$\mathcal{L}(\boldsymbol{\alpha}_i)(s) = \left\{ \begin{bmatrix} s[y := i] \mapsto p \\ s[y := 2] \mapsto 1/2 - p \\ \bot \qquad\quad \mapsto 1/2 \end{bmatrix} \middle| \ 0 \le p \le \frac{1}{2} \right\}\uparrow$$

The $\bot$ is introduced when an assume fails, but the upward closure can replace $\bot$ with any other behavior (since $\bot \le t$ for any $t \in \mathcal{S}$). Therefore the execution of these partial traces results in new behaviors that were not present in $\mathcal{L}([\![C]\!])(s)$. While it may be possible to recover $\mathcal{L}([\![C]\!])(s)$ from $\mathcal{L}(\boldsymbol{\alpha}_0)(s)$ and $\mathcal{L}(\boldsymbol{\alpha}_1)(s)$, any procedure to do so would need to identify and remove the new behaviors, and there is no clear way to do this, especially when the program contains probabilistic looping, and the desired result contains a distribution with infinite support. In contrast, the semantics based on our new pomsets with formulae, and the associated linearization procedure, provides a direct way to interpret the meaning of probabilistic concurrent programs, such as the one above.

VII. Discussion and Related Work

There is a rich history of using partial orders in denotational models of concurrent programs [9], [10], [12]. The resulting pomsets have been used as a semantic basis for concurrent separation logic [11], concurrent Kleene algebras [13], [27], [28], [53], and other applications [54]–[56].

While there are links between pomset semantics and operational models of concurrency [57], pomsets are often preferred for their finer notion of concurrency compared to the interleaving semantics offered by operational models. This is especially useful in the context of weak memory [54], [56]. An operational approach to probabilistic concurrency with unbounded looping may be possible, but determining the final probabilities of outcomes would require a transitive closure of the step relation; topological or domain theoretic methods would then be needed to ensure that the limit exists, at which point the approach would be quite similar to our own.

Probabilistic event structures model probabilistic process algebras [58]–[61], whereas we model a full imperative language with sequential composition, control flow, and unbounded loops, for which adding formulae was essential.

Concurrency has previously been combined with other computational effects in limited ways. For example, semantics for probabilistic concurrent programs have been defined in both operational [62]–[64] and denotational [23] styles, but these approaches are limited to programs with bounded looping constructs. The inclusion of unbounded looping adds significant complexity in the probabilistic case. Whereas finite-trace models are sufficient for many non-probabilistic scenarios, unbounded looping in probabilistic programs requires infinite traces, as the probability of termination may only become 1 in the limit. A proper theory of infinite traces requires pomsets to be enriched with a DCPO [24], [65] or metric [41] structure; prior work in this area was very informative to our own development, although the inclusion of deterministic branching in our own pomset structure added significant complexity.

Branching pomsets were recently introduced to model choices in choreographic programs [55], [66], [67]. While branching pomsets can, in theory, contain infinitely many nodes, the authors have not developed the domain theoretic properties needed to approximate infinite structures, which was a significant focus of this paper. Indeed, our extension lemma (Lemma 6) was necessary for linearization, without which we

would not have been able to relate our pomeset model to the known convex powerset semantics [4], [8], [29].

Indeed, while much of this paper was devoted to developing domain theoretic properties of pomsets with formulae, it would be interesting to further explore the connections to probabilistic denotational models. For example, our formulation of the convex powerset is based on the Smyth powerdomain, but other powerdomain constructions exist as well [46]–[49]. In addition, there are other domains for mixing probabilistic and nondeterministic computation including indexed valuations [19], [35], [36] and multisets of distributions [68], [69].

We are also interested in developing program logics on top of our new semantics model. For example, our order over action labels could be used to define *invariant sensitive execution* [23], which is useful for the metatheory of Concurrent Separation Logic style deduction systems for probabilistic programs [70]. Given that our model supports unbounded looping, it will be possible to develop almost sure termination rules in a concurrent context for the first time, paving the way towards formalizing the correctness proofs of distributed probabilistic protocols such as the Dining Philosophers problem [14]–[18]. Whereas pen and paper proofs of these protocols were conducted more than four decades ago, this paper presents the first formal semantic model for those programs.

References

[1] D. Scott, "Outline of a Mathematical Theory of Computation," OUCL, Tech. Rep. PRG02, November 1970.

[2] D. Scott and C. Strachey, "Toward A Mathematical Semantics For Computer Languages," OUCL, Tech. Rep. Prg06, August 1971.

[3] D. Kozen, "Semantics of probabilistic programs," in *20th Annual Symposium on Foundations of Computer Science (SFCS '79)*, 1979, pp. 101–114.

[4] A. McIver and C. Morgan, *Abstraction, Refinement and Proof for Probabilistic Systems*, ser. Monographs in Computer Science. Springer, 2005.

[5] C. Morgan, A. McIver, and K. Seidel, "Probabilistic Predicate Transformers," *ACM Trans. Program. Lang. Syst.*, vol. 18, no. 3, p. 325–353, may 1996.

[6] G. Barthe, T. Espitau, M. Gaboardi, B. Grégoire, J. Hsu, and P.-Y. Strub, "An Assertion-Based Program Logic for Probabilistic Programs," in *Programming Languages and Systems*. Cham: Springer International Publishing, 2018, pp. 117–144.

[7] J. den Hartog, "Verifying probabilistic programs using a hoare like logic," in *Advances in Computing Science — ASIAN'99*, P. S. Thiagarajan and R. Yap, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 113–125.

[8] N. Zilberstein, D. Kozen, A. Silva, and J. Tassarotti, "A demonic outcome logic for randomized nondeterminism," *Proc. ACM Program. Lang.*, vol. 9, no. POPL, Jan 2025. [Online]. Available: https://doi.org/10.1145/3704855

[9] J. L. Gischer, "The equational theory of pomsets," *Theoretical Computer Science*, vol. 61, no. 2, pp. 199–224, 1988. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397588901247

[10] V. R. Pratt, "Semantical Considerations on Floyd-Hoare Logic," in *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, 1976, pp. 109–121.

[11] S. Brookes, "A semantics for concurrent separation logic," in *CONCUR 2004 - Concurrency Theory*, P. Gardner and N. Yoshida, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–34.

[12] J. Grabowski, "On Partial Languages," *Fundamenta Informaticae*, vol. 4, no. 2, pp. 427–498, 1981. [Online]. Available: https://doi.org/10.3233/FI-1981-4210

[13] P. Jipsen and M. A. Moshier, "Concurrent Kleene algebra with tests and branching automata," *Journal of Logical and Algebraic Methods in Programming*, vol. 85, no. 4, pp. 637–652, 2016, relational and algebraic methods in computer science.

[14] M. O. Rabin, "N-process synchronization by 4.log2n-valued shared variable," in *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, 1980, pp. 407–410.

[15] M. O. Rabin, "The choice coordination problem," *Acta Inf.*, vol. 17, no. 2, p. 121–134, Jun. 1982. [Online]. Available: https://doi.org/10.1007/BF00288965

[16] R. Morris, "Counting large numbers of events in small registers," *Commun. ACM*, vol. 21, no. 10, p. 840–842, Oct. 1978. [Online]. Available: https://doi.org/10.1145/359619.359627

[17] S. Hart, M. Sharir, and A. Pnueli, "Termination of probabilistic concurrent program," *ACM Trans. Program. Lang. Syst.*, vol. 5, no. 3, p. 356–380, Jul. 1983. [Online]. Available: https://doi.org/10.1145/2166.357214

[18] D. Lehmann and M. O. Rabin, "On the advantages of free choice: a symmetric and fully distributed solution to the dining philosophers problem," in *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '81. New York, NY, USA: Association for Computing Machinery, 1981, p. 133–138. [Online]. Available: https://doi.org/10.1145/567532.567547

[19] D. Varacca and G. Winskel, "Distributing probability over nondeterminism," *Mathematical Structures in Computer Science*, vol. 16, no. 1, p. 87–113, 2006.

[20] M. Zwart and D. Marsden, "No-Go Theorems for Distributive Laws," in *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2019, pp. 1–13.

[21] M. Zwart, "On the Non-Compositionality of Monads via Distributive Laws," Ph.D. dissertation, University of Oxford, 2020. [Online]. Available: https://ora.ox.ac.uk/objects/uuid:b2222b14-3895-4c87-91f4-13a8d046febb

[22] L. Parlant, "Monad Composition via Preservation of Algebras," Ph.D. dissertation, University College London, 2020. [Online]. Available: https://discovery.ucl.ac.uk/id/eprint/10112228/

[23] N. Zilberstein, A. Silva, and J. Tassarotti, "Probabilistic concurrent reasoning in outcome logic: Independence, conditioning, and invariants," 2024. [Online]. Available: https://arxiv.org/abs/2411.11662

[24] J. J. C. Meyer and E. P. de Vink, "Pomset semantics for true concurrency with synchronization and recursion," in *Mathematical Foundations of Computer Science 1989*, A. Kreczmar and G. Mirkowska, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 360–369.

[25] M. Smyth, "Power domains," *Journal of Computer and System Sciences*, vol. 16, no. 1, pp. 23–36, 1978. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002200007890048X

[26] G. Plotkin, "A powerdomain construction," *SIAM Journal on Computing*, vol. 5, no. 3, pp. 452–487, 1976. [Online]. Available: https://doi.org/10.1137/0205035

[27] T. Kappé, P. Brunet, A. Silva, J. Wagemaker, and F. Zanasi, *Concurrent Kleene Algebra with Observations: From Hypotheses to Completeness*. Springer International Publishing, 2020, p. 381–400. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-45231-5_20

[28] M. R. Laurence and G. Struth, "Completeness Theorems for Bi-Kleene Algebras and Series-Parallel Rational Pomset Languages," in *Relational and Algebraic Methods in Computer Science*, P. Höfner, P. Jipsen, W. Kahl, and M. E. Müller, Eds. Cham: Springer International Publishing, 2014, pp. 65–82.

[29] J. He, K. Seidel, and A. McIver, "Probabilistic models for the guarded command language," *Science of Computer Programming*, vol. 28, no. 2, pp. 171–192, 1997, formal Specifications: Foundations, Methods, Tools and Applications. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642396000196

[30] S. Abramsky and A. Jung, *Domain theory*. USA: Oxford University Press, Inc., 1995, p. 1–168.

[31] S. Awodey, *Category Theory*, ser. Oxford Logic Guides. Ebsco Publishing, 2006. [Online]. Available: https://books.google.co.uk/books?id=IK_sIDI2TCwC

[32] N. Zilberstein, D. Dreyer, and A. Silva, "Outcome Logic: A Unifying Foundation for Correctness and Incorrectness Reasoning," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA1, Apr 2023.

[33] N. Zilberstein, A. Saliling, and A. Silva, "Outcome separation logic: Local reasoning for correctness and incorrectness with computational effects," *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA1, Apr 2024.

[34] C. Morgan, A. McIver, K. Seidel, and J. W. Sanders, "Refinement-oriented probability for CSP," *Form. Asp. Comput.*, vol. 8, no. 6, p. 617–647, nov 1996. [Online]. Available: https://doi.org/10.1007/BF01213492

[35] D. Varacca, "The powerdomain of indexed valuations," in *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, 2002, pp. 299–308.

[36] D. Varacca, " Probability, Nondeterminism and Concurrency: Two Denotational Models for Probabilistic Computation," Ph.D. dissertation, University of Aarhus, Nov 2003. [Online]. Available: https://www.brics.dk/DS/03/14/

[37] C. A. R. Hoare, "An Axiomatic Basis for Computer Programming," *Commun. ACM*, vol. 12, no. 10, p. 576–580, Oct. 1969.

[38] R. W. Floyd, "Assigning Meanings to Programs," in *Mathematical Aspects of Computer Science*, ser. Proceedings of Symposia in Applied Mathematics, vol. 19. Providence, Rhode Island: American Mathematical Society, 1967, pp. 19–32.

[39] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE Transactions on Software Engineering*, vol. SE-3, no. 2, pp. 125–143, 1977.

[40] V. R. Pratt, "Modeling concurrency with partial orders," *Int. J. Parallel Program.*, vol. 15, no. 1, pp. 33–71, 1986. [Online]. Available: https://doi.org/10.1007/BF01379149

[41] J. W. de Bakker and J. H. A. Warmerdam, "Metric pomset semantics for a concurrent language with recursion," in *Semantics of Systems of Concurrent Processes*, I. Guessarian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 21–49.

[42] K. Apt and G. Plotkin, "Countable nondeterminism and random assignment," *J. ACM*, vol. 33, no. 4, p. 724–767, aug 1986. [Online]. Available: https://doi.org/10.1145/6490.6494

[43] R.-J. Back, "Semantics of unbounded nondeterminism," in *Automata, Languages and Programming*, J. de Bakker and J. van Leeuwen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1980, pp. 51–63.

[44] H. Søndergaard and P. Sestoft, "Non-determinism in Functional Languages," *The Computer Journal*, vol. 35, no. 5, pp. 514–523, 10 1992. [Online]. Available: https://doi.org/10.1093/comjnl/35.5.514

[45] B. Jacobs, "Coalgebraic trace semantics for combined possibilitistic and probabilistic systems," *Electronic Notes in Theoretical Computer Science*, vol. 203, no. 5, pp. 131–152, 2008, proceedings of the Ninth Workshop on Coalgebraic Methods in Computer Science (CMCS 2008). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S157106610800337X

[46] R. Tix, K. Keimel, and G. Plotkin, "Semantic domains for combining probability and non-determinism," *Electronic Notes in Theoretical Computer Science*, vol. 222, pp. 3–99, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066109000036

[47] R. Tix, "Convex power constructions for continuous d-cones," *Electronic Notes in Theoretical Computer Science*, vol. 35, pp. 206–229, 2000, workshop on Domains IV. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066105807467

[48] K. Keimel and G. Plotkin, "Mixed powerdomains for probability and nondeterminism," *Logical Methods in Computer Science*, vol. Volume 13, Issue 1, Jan. 2017. [Online]. Available: https://lmcs.episciences.org/2665

[49] R. Tix, "Continuous D-cones: convexity and powerdomain constructions," Ph.D. dissertation, Darmstadt University of Technology, Germany, 1999. [Online]. Available: https://d-nb.info/957239157

[50] M. Mislove, "Nondeterminism and probabilistic choice: Obeying the laws," in *CONCUR 2000 — Concurrency Theory*, C. Palamidessi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 350–365.

[51] M. Mislove, J. Ouaknine, and J. Worrell, "Axioms for probability and nondeterminism," *Electronic Notes in Theoretical Computer Science*, vol. 96, pp. 7–28, 2004, proceedings of the 10th International Workshop on Expressiveness in Concurrency. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066104050297

[52] F. Bonchi, A. Sokolova, and V. Vignudelli, "Presenting Convex Sets of Probability Distributions by Convex Semilattices and Unique Bases," in *9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), F. Gadducci and A. Silva, Eds., vol. 211. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 11:1–11:18. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CALCO.2021.11

[53] T. Hoare, B. Möller, G. Struth, and I. Wehrman, "Concurrent kleene algebra and its foundations," *J. Log. Algebraic Methods Program.*, vol. 80, no. 6, pp. 266–296, 2011. [Online]. Available: https://doi.org/10.1016/j.jlap.2011.04.005

[54] R. Kavanagh and S. Brookes, "A Denotational Semantics for SPARC TSO," *Electronic Notes in Theoretical Computer Science*, vol. 336, pp. 223–239, 2018, the Thirty-third Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIII). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066118300288

[55] L. Edixhoven, S.-S. Jongmans, J. Proença, and G. Cledou, "Branching Pomsets for Choreographies," *Electronic Proceedings in Theoretical Computer Science*, vol. 365, p. 37–52, Aug. 2022. [Online]. Available: http://dx.doi.org/10.4204/EPTCS.365.3

[56] R. Jagadeesan, A. Jeffrey, and J. Riely, "Pomsets with preconditions: a simple model of relaxed memory," *Proc. ACM Program. Lang.*, vol. 4, no. Oopsla, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3428262

[57] S. Brookes, "Traces, Pomsets, Fairness and Full Abstraction for Communicating Processes," in *CONCUR 2002 — Concurrency Theory*, L. Brim, M. Křetínský, A. Kučera, and P. Jančar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 466–482.

[58] J.-P. Katoen, "Quantitative and Qualitative Extensions of Event Structures," PhD Thesis - Research UT, graduation UT, University of Twente, Netherlands, 1996.

[59] G. Winskel, *Probabilistic and Quantum Event Structures*. Cham: Springer International Publishing, 2014, pp. 476–497. [Online]. Available: https://doi.org/10.1007/978-3-319-06880-0_25

[60] D. Varacca, H. Völzer, and G. Winskel, "Probabilistic event structures and domains," *Theoretical Computer Science*, vol. 358, no. 2, pp. 173–199, 2006, concurrency Theory (CONCUR 2004). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030439750600051X

[61] D. Varacca and N. Yoshida, "Probabilistic π-Calculus and Event Structures," *Electronic Notes in Theoretical Computer Science*, vol. 190, no. 3, pp. 147–166, 2007, proceedings of the Fifth Workshop on Quantitative Aspects of Programming Languages (QAPL 2007). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066107005634

[62] J. Tassarotti and R. Harper, "A Separation Logic for Concurrent Randomized Programs," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, Jan 2019.

[63] J. Tassarotti, "Verifying Concurrent Randomized Algorithms," Ph.D. dissertation, Carnegie Mellon University, Dec 2018. [Online]. Available: https://csd.cmu.edu/academics/doctoral/degrees-conferred/joseph-tassarotti

[64] I. Fesefeldt, J.-P. Katoen, and T. Noll, "Towards Concurrent Quantitative Separation Logic," in *33rd International Conference on Concurrency Theory (CONCUR 2022)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), B. Klin, S. Lasota, and A. Muscholl, Eds., vol. 243. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 25:1–25:24. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CONCUR.2022.25

[65] J.-J. Meyer and E. de Vink, "Applications of compactness in the smyth powerdomain of streams," *Theoretical Computer Science*, vol. 57, no. 2, pp. 251–282, 1988. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397588900424

[66] L. Edixhoven and S.-S. Jongmans, "Realisability of Branching Pomsets," in *Formal Aspects of Component Software*, S. L. Tapia Tarifa and J. Proença, Eds. Cham: Springer International Publishing, 2022, pp. 185–204.

[67] L. Edixhoven, S.-S. Jongmans, J. Proença, and I. Castellani, "Branching pomsets: Design, expressiveness and applications to choreographies," *Journal of Logical and Algebraic Methods in Programming*, vol. 136, p. 100919, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352220823000731

[68] B. Jacobs, "From Multisets over Distributions to Distributions over Multisets," in *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '21. New York, NY, USA: Association for Computing Machinery, 2021.

[69] D. Kozen and A. Silva, *Multisets and Distributions*. Cham: Springer Nature Switzerland, 2024, pp. 168–187. [Online]. Available: https://doi.org/10.1007/978-3-031-61716-4_11

[70] P. W. O'Hearn, "Resources, Concurrency and Local Reasoning," in

CONCUR 2004 - Concurrency Theory. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 49–67.

[71] G. Markowsky, "Chain-complete posets and directed sets with applications," *Algebra Universalis*, vol. 6, pp. 53–68, 12 1976.

[72] W. Różowski, T. Kappé, D. Kozen, T. Schmid, and A. Silva, "Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity," in *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), K. Etessami, U. Feige, and G. Puppis, Eds., vol. 261. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, pp. 136:1–136:20. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2023.136

[73] M. B. Smyth, "Finite approximation of spaces," in *Proceedings of a Tutorial and Workshop on Category Theory and Computer Programming*. Berlin, Heidelberg: Springer-Verlag, 1986, p. 225–241.

Appendix

APPENDIX A
EXAMPLE: UNROLLING LOOPS

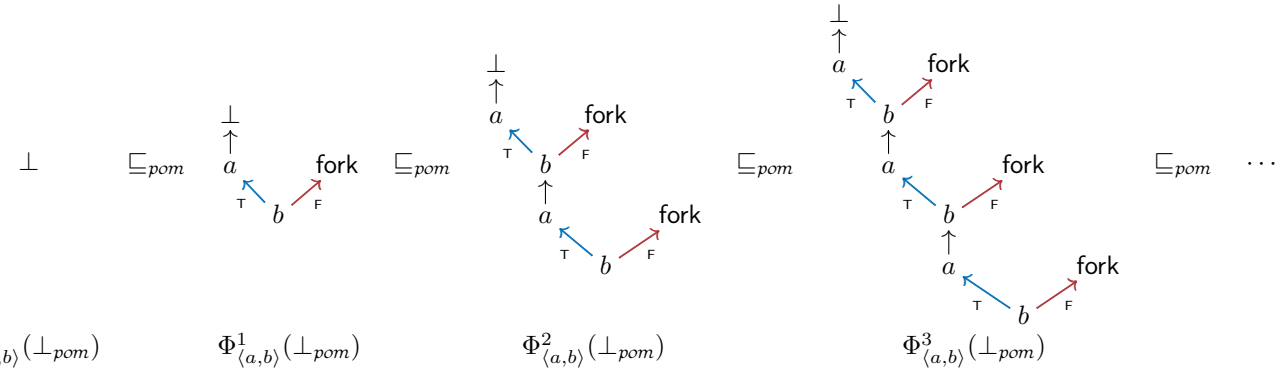
In this section, we demonstrate the semantic structures that are generated as the result of looping programs. We start with a simple program, which is a loop containing a single action as the body:

$$\textbf{while}\ b\ \{a\}$$

As defined in Figure 3, the semantics of this program is given by the following fixed point:

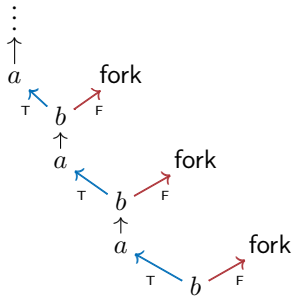
$$[\![\textbf{while}\ b\ \{a\}]\!] = \mathsf{lfp}\left(\Phi_{\langle a,b\rangle}\right) \qquad \text{where} \qquad \Phi_{\langle a,b\rangle}(\boldsymbol{\alpha}) = \mathsf{guard}(b, \langle a\rangle \fatsemi \boldsymbol{\alpha}, [\![\textbf{skip}]\!])$$

By the Kleene fixed point theorem, the least fixed point above is given by the supremum over all the finite iterations of $\Phi_{\langle a,b\rangle}$ applied to $\bot_{pom}$. Unrolling this definition several times, we get the following chain:



The supremum of this chain is clearly an infinite structure with a terminating branch for each $n \in \mathbb{N}$, and an infinitely ascending spine. The $\bot$ node is pushed to a progressively higher level after each unrolling, so in the supremum it does not appear at all.

$$[\![\textbf{while}\ b\ \{a\}]\!] \quad = \quad \mathsf{lfp}\left(\Phi_{\langle a,b\rangle}\right) \quad = \quad \sup_{n\in\mathbb{N}} \Phi^n_{\langle a,b\rangle}(\bot_{pom}) \quad =$$

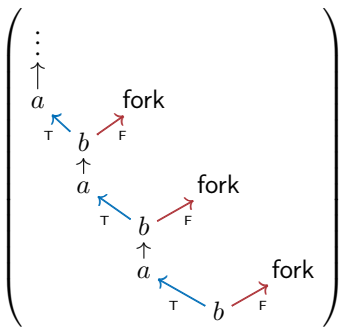


Now suppose that we would like to sequentially compose another action after the loops, to obtain the following program:

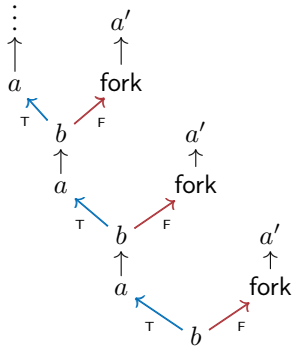
$$\textbf{while}\ b\ \{a\}\ ;\ a'$$

According to the definition of sequential composition from Section III-B, the semantics of this program is obtained by composing a copy of a' after each branch of $[\![\textbf{while}\ b\ \{a\}]\!]$. As we just saw, the branches of $[\![\textbf{while}\ b\ \{a\}]\!]$ correspond to all of the terminating traces of the loop, and therefore we make countably many copies of a'.

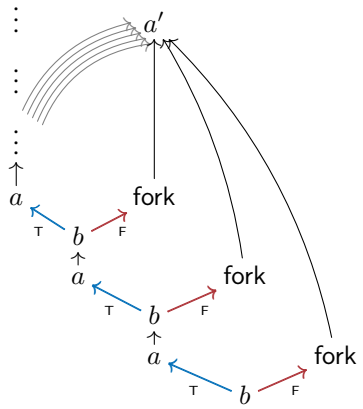
$$[\![\textbf{while}\ b\ \{a\}\ ;\ a']\!] \quad =$$



$$\fatsemi \quad a' \quad =$$



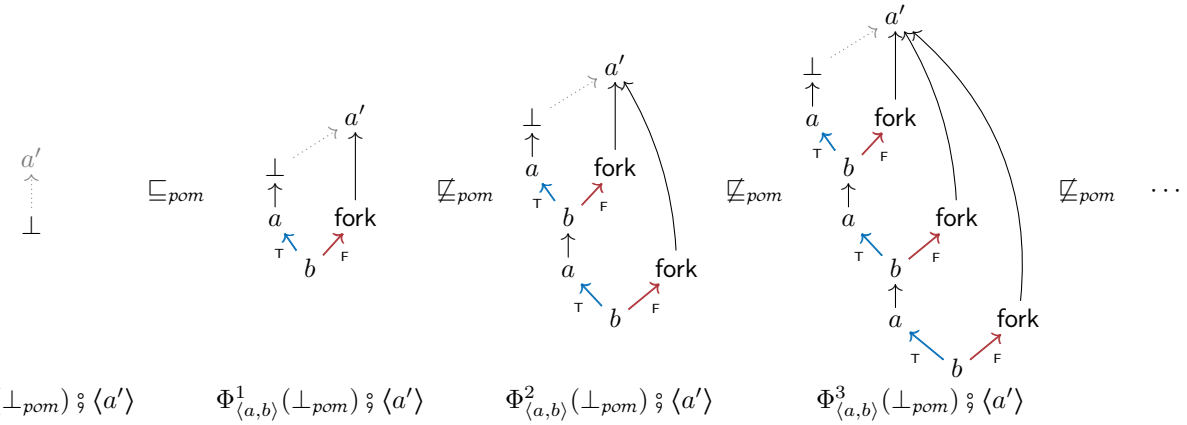
We investigated the possibility of defining sequential composition without such a copying, similar to what was done for finite structures in [23]. However, this approach does not translate to the infinite case. To see why not, consider the following structure, obtained by composing a' after **while** b $\{a\}$, without copying.



The node labelled as a' is not finitely preceded; it has countably many predecessors, one for each terminating branch of the loop. Relaxing the finitely preceded requirement for LPOFs is not an option; it breaks many of the proofs from Appendix D, preventing us from approximating a pomset using finite structures and invalidating the extension lemma. Without the extension lemma, it is unclear how linearization could be defined, as well-founded recursion relies on the structure being finite. Scott continuity of sequential composition comes into question as well. Scott continuity of $\fatsemi$ means that the following equation holds:

$$\left(\sup_{n\in\mathbb{N}} \Phi^n_{\langle a,b\rangle}(\bot_{pom})\right) \fatsemi \langle a'\rangle \qquad \stackrel{?}{=} \qquad \sup_{n\in\mathbb{N}} \left(\Phi^n_{\langle a,b\rangle}(\bot_{pom}) \fatsemi \langle a'\rangle\right)$$

We investigated two possible expansions for the latter expression. The first possibility retained the requirement that nothing can proceed $\bot$; it is shown as the sequence below, ignoring the grey dotted arrows. The second possibility allowed $\bot$ nodes to have successors, and is shown below, where the grey dotted arrows indicate standard causality.



The causality from $\bot$ to a' is problematic, since $\bot$ could be expanded into an infinite structure, again causing a' to not be finitely preceded. But the lack of causality from $\bot$ to a' is also problematic, as in the next approximation, $\bot$ will be replaced by the test b, which is supposed to be scheduled before a'. This complicates monotonicity of linearization, as the allowable interleavings can change as the structure grows.

In any case, the sequence above does not form a chain—at least, not according to our current definition of $\sqsubseteq_{pom}$. The reason why this is not a chain is because the node labelled a' gains more predecessors with each successive structure. We investigated several alternative definitions of $\sqsubseteq_{pom}$, which did allow for this sort of expansion, however each one had significant problems—*e.g.,* some were not transitive, and thus not partial orders; some were not DCPOs; and some caused $\Phi_{\langle C,b\rangle}$ to not be monotone, meaning that loops could not be interpreted as fixed points using the Kleene fixed point theorem or the Knaster-Tarski theorem. In the course of our investigation, we found that the copying of structures in sequential composition was the best way to obtain a sound structure, obeying all the necessary laws.

Interestingly, this copying behavior corresponds to the following equational law of Probabilistic Guarded Kleene Algebra with Tests [72], where sequencing a command C after a guarded choice is equivalent to including C in both branches:

$$\textbf{if}\ b\ \{C_1\}\ \textbf{else}\ \{C_2\}\ ;\ C \quad \equiv \quad \textbf{if}\ b\ \{C_1; C\}\ \textbf{else}\ \{C_2; C\}$$

APPENDIX B
POMSETS WITH TESTS

A. Labelled Partial Orders

Lemma B.1. *Let $\alpha \sqsubseteq_{lpo} \beta$; then,* $\mathsf{nBot}_\alpha \subseteq \mathsf{nBot}_\beta$.

Proof. By Def. 8 (Item 1 and 3a), we have that $\lambda_\alpha(x) \leq \lambda_\beta(x)$, for all $x \in N_\alpha \subseteq N_\beta$. Hence, if $x \in \mathsf{nBot}_\alpha$, then $\bot < \lambda_\alpha(x) \leq \lambda_\beta(x)$ and so $x \in \mathsf{nBot}_\beta$. □

Lemma 1. *If $\alpha \sqsubseteq_{lpo} \beta$, then $N_\beta \setminus N_\alpha = \mathsf{Bot}_\alpha{\uparrow_\beta}$.*

Proof. We show the set inclusion in both ways.

First, take any $x \in N_\beta \setminus N_\alpha$. Since x is finitely proceeded, there must be a positive integer n and nodes $x_1, \ldots, x_n \in N_\beta$ such that x_1 is the root of β, $x_n = x$, and $x_{i+1} \in \mathsf{succ}_\beta(x_i)$ for all $1 \leq i < n$. Since $\alpha \sqsubseteq_{lpo} \beta$, then x_1 must also be the root of α. Now, let i be smallest index such that $x_i \in N_\alpha$ and $x_{i+1} \notin N_\alpha$ (note that $1 < i < n$ since $x_1 \in N_\alpha$ but $x = x_n \notin N_\alpha$). From $\alpha \sqsubseteq_{lpo} \beta$, we know that $\mathsf{succ}_\alpha(x_i) = \mathsf{succ}_\beta(x_i) \setminus \mathsf{Bot}_\alpha{\uparrow_\beta}$; so, $x_{i+1} \in \mathsf{Bot}_\alpha{\uparrow_\beta}$ that allows us to conclude $x \in \mathsf{Bot}_\alpha{\uparrow_\beta}$, being $x_{i+1} <_\beta x_n = x$.

Now, take $x \in \mathsf{Bot}_\alpha{\uparrow_\beta}$, i.e. there exists $z \in \mathsf{Bot}_\alpha$ such that $z <_\beta x$. If $x \in N_\alpha$, being $<_\alpha = <_\beta \cap (N_\alpha \times N_\alpha)$, we would have that $z <_\alpha x$, in contradiction with $z \in \mathsf{Bot}_\alpha$. □

Lemma 2. *$\sqsubseteq_{lpo}$ is a partial order.*

Proof. Reflexivity is trivial. For antisymmetry, we have that $\alpha \sqsubseteq_{lpo} \beta$ and $\beta \sqsubseteq_{lpo} \alpha$ imply that $N_\alpha = N_\beta$; hence:

$$<_\alpha = <_\beta \cap (N_\alpha \times N_\alpha) = <_\beta \cap (N_\beta \times N_\beta) = <_\beta$$

Then, let $x \in N_\alpha (= N_\beta)$. Trivially, $\varphi_\alpha(x) = \varphi_\beta(x)$; furthermore, $\lambda_\alpha(x) \leq \lambda_\beta(x)$ and $\lambda_\beta(x) \leq \lambda_\alpha(x)$, so, $\lambda_\alpha(x) = \lambda_\beta(x)$, by antisimmetry of $\leq$.

For transitivity, let us assume that $\alpha \sqsubseteq_{lpo} \beta$ and $\beta \sqsubseteq_{lpo} \gamma$, and prove that $\alpha \sqsubseteq_{lpo} \gamma$. By hypothesis, $N_\alpha \subseteq_\downarrow N_\beta \subseteq_\downarrow N_\gamma$, so $N_\alpha \subseteq N_\gamma$; we have to prove that N_α is downwards closed (w.r.t. to γ). Take any $x \in N_\alpha$ and $y <_\gamma x$. Since $N_\alpha \subseteq N_\beta$, $x \in N_\beta$ and, being $N_\beta \subseteq_\downarrow N_\gamma$, then $y \in N_\beta$. Further, since $<_\beta = <_\gamma \cap (N_\beta \times N_\beta)$, then $y <_\beta x$. Therefore $y \in N_\alpha$, since $N_\alpha \subseteq_\downarrow N_\beta$.

For the second item of Definition 8,

$$\begin{array}{rcl} <_\alpha & = & <_\beta \cap (N_\alpha \times N_\alpha) \\ & = & (<_\gamma \cap (N_\beta \times N_\beta)) \cap (N_\alpha \times N_\alpha) \\ & = & <_\gamma \cap ((N_\beta \cap N_\alpha) \times (N_\beta \cap N_\alpha)) \\ & = & <_\gamma \cap (N_\alpha \times N_\alpha) \end{array}$$

Finally, let $x \in N_\alpha$. Trivially, $\varphi_\alpha(x) = \varphi_\beta(x) = \varphi_\gamma(x)$ and $\lambda_\alpha(x) \leq \lambda_\beta(x) \leq \lambda_\gamma(x)$; so, $\lambda_\alpha(x) \leq \lambda_\gamma(x)$, by transitivity of $\leq$. Moreover, we have:

$$\begin{array}{rcl} \mathsf{succ}_\alpha(x) & = & \mathsf{succ}_\beta(x) \setminus \mathsf{Bot}_\alpha{\uparrow_\beta} \\ & = & \left(\mathsf{succ}_\gamma(x) \setminus \mathsf{Bot}_\beta{\uparrow_\gamma}\right) \setminus \mathsf{Bot}_\alpha{\uparrow_\beta} \\ & = & \mathsf{succ}_\gamma(x) \setminus \left(\mathsf{Bot}_\beta{\uparrow_\gamma} \cup \mathsf{Bot}_\alpha{\uparrow_\beta}\right) \\ & = & \mathsf{succ}_\gamma(x) \setminus \mathsf{Bot}_\alpha{\uparrow_\gamma} \end{array}$$

where the last equality holds thanks to Lemma 1: indeed, $\mathsf{Bot}_\alpha{\uparrow_\beta} = N_\beta \setminus N_\alpha$, $\mathsf{Bot}_\beta{\uparrow_\gamma} = N_\gamma \setminus N_\beta$ and $\mathsf{Bot}_\alpha{\uparrow_\gamma} = N_\gamma \setminus N_\alpha$; we conclude since $(N_\gamma \setminus N_\beta) \cup (N_\beta \setminus N_\alpha) = (N_\gamma \setminus N_\alpha)$. □

Lemma B.2. *Let $N_\beta \subseteq_\downarrow N_\alpha$ and $<_\beta = <_\alpha \cap (N_\beta \times N_\beta)$. Then:*

1) *if $y \in \mathsf{succ}_\alpha(x)$ and $x <_\beta y$, then $y \in \mathsf{succ}_\beta(x)$;*
2) *if $y \in \mathsf{succ}_\beta(x)$, then $y \in \mathsf{succ}_\alpha(x)$;*
3) *$\mathsf{succ}_\beta(x) \subseteq \mathsf{succ}_\alpha(x) \setminus \mathsf{Bot}_\beta{\uparrow_\alpha}$, for every $x \in N_\beta$.*

Proof. For the first claim, if it was $x <_\beta z <_\beta y$, then $x <_\alpha z <_\alpha y$ (being $<_\beta \subseteq <_\alpha$), in contradiction with $y \in \mathsf{succ}_\alpha(x)$.

For the second claim, $x <_\beta y$ and $<_\beta \subseteq <_\alpha$ imply that $x <_\alpha y$. Now, by contradiction, assume that $x <_\alpha z <_\alpha y$. Since $y \in N_\beta \subseteq_\downarrow N_\alpha$, then $z \in N_\beta$; hence, being $<_\beta = <_\alpha \cap (N_\beta \times N_\beta)$, we would have that $x <_\beta z <_\beta y$, in contradiction with $y \in \mathsf{succ}_\beta(x)$. So, $y \in \mathsf{succ}_\alpha(x)$.

For the third claim, let $y \in \mathsf{succ}_\beta(x)$; by the second claim, $y \in \mathsf{succ}_\alpha(x)$. If it was $y \in \mathsf{Bot}_\beta{\uparrow_\alpha}$, then there would exist $z \in \mathsf{Bot}_\beta$ such that $z <_\alpha y$; being $<_\beta = <_\alpha \cap (N_\beta \times N_\beta)$, we would have that $z <_\beta y$, i.e. $z \notin \mathsf{Bot}_\beta$. □

Lemma B.3. *If $D \subseteq \mathit{lpo}(L)$ is a directed set, then $x{\downarrow_\beta} = x{\downarrow_{\beta'}}$ for any $\beta, \beta' \in D$ such that $x \in N_\beta \cap N_{\beta'}$.*

Proof. Take any $\beta, \beta' \in D$ such that $x \in N_\beta$ and $x \in N_{\beta'}$. Since D is directed, there must be a $\gamma \in D$ such that $\beta \sqsubseteq_{\mathit{lpo}} \gamma$ and $\beta' \sqsubseteq_{\mathit{lpo}} \gamma$. Now, take any $y <_\beta x$, clearly this means that $y <_\gamma x$, and since $N_{\beta'} \subseteq_\downarrow N_\gamma$, then $y \in N_{\beta'}$, and so $y \in x{\downarrow_{\beta'}}$, so we have showed that $x{\downarrow_\beta} \subseteq x{\downarrow_{\beta'}}$. By an equivalent argument, we get that $x{\downarrow_{\beta'}} \subseteq x{\downarrow_\beta}$, therefore $x{\downarrow_\beta} = x{\downarrow_{\beta'}}$. □

Lemma B.4. *For any directed set $D \subseteq \mathit{lpo}(L)$ and $n \in \mathbb{N}$, there exists a finite set X such that $\{x \mid \mathsf{lev}_\beta(x) = n\} \subseteq X$ for all $\beta \in D$.*

Proof. The proof is by induction on n. Let $n = 1$, take any $\beta \in D$, and let x be the root of β. Since D is directed, then x is the root of every $\beta' \in D$, therefore letting $X = \{x\}$, we have that $\{y \mid \mathsf{lev}_\beta(y) = 1\} = X$ for all $\beta \in D$.

Now, suppose that the claim holds for all $k < n$, and let X_k be the finite set such that $\{x \mid \mathsf{lev}_\beta(x) = k\} \subseteq X_k$ for each $k < n$. Also, let $X = \bigcup_{k<n} X_k$, so $\{x \mid \mathsf{lev}_\beta(x) < n\} \subseteq X$, which is also finite. This means that only finitely many $\bot$ nodes can appear before level n. Now construct A as follows: for each $x \in X$, select a $\beta \in D$ such that $\lambda_\beta(x) \neq \bot$, if such a β exists. So $|A| \leq |X|$, which is clearly finite. Since A is a finite subset of D, which is directed, then there must be some $\gamma \in D$ that is an upper bound of all the elements in A. This means that for all $x \in X$, $x \notin \mathsf{Bot}_\beta$ for some $\beta \in D$ iff $x \notin \mathsf{Bot}_\gamma$.

Now let $Y = \{x \mid \mathsf{lev}_\gamma(x) = n\}$. Take any $\beta \in D$, so there is a $\beta' \in D$ that is an upper bound of β and γ. By construction, β' can only extend γ after level n, and so the elements of γ at level n must be the same as the elements of β' and since $\beta \sqsubseteq_{\mathit{lpo}} \beta'$, then $\{x \mid \mathsf{lev}_{\beta'}(x) = n\} \subseteq Y$. □

Lemma 3. *For any directed set $D \subseteq \mathit{lpo}(L)$, $\sup D$ exists and is given by $\alpha = \langle N_\alpha, <_\alpha, \lambda_\alpha, \varphi_\alpha \rangle$, where:*

$$N_\alpha \triangleq \bigcup_{\beta \in D} N_\beta \qquad <_\alpha \triangleq \bigcup_{\beta \in D} <_\beta$$

$$\lambda_\alpha(x) \triangleq \sup_{\beta \in D \,:\, x \in N_\beta} \lambda_\beta(x) \qquad \varphi_\alpha(x) \triangleq \psi_x$$

and $\psi_x = \varphi_\beta(x)$, for all $\beta \in D$ such that $x \in N_\beta$.

Proof. We need to show that:

1) $\alpha \in \mathit{lpo}(L)$;
2) $\beta \sqsubseteq_{\mathit{lpo}} \alpha$, for all $\beta \in D$; and
3) $\alpha \sqsubseteq_{\mathit{lpo}} \gamma$, for all γ s.t. $\beta \sqsubseteq_{\mathit{lpo}} \gamma$ (for all $\beta \in D$).

We start with property (1). First, clearly α is single-rooted, since D is a directed set and therefore every $\beta \in D$ must have the same root. We now show that every element of N_α is finitely proceeded. Take any $x \in N_\alpha$, so $x \in N_\beta$ for some $\beta \in D$. By Lemma B.3, we know that $x{\downarrow_\beta} = x{\downarrow_{\beta'}}$ for all $\beta' \in D$, therefore $x{\downarrow_\alpha} = \bigcup_{\beta' \in A} x{\downarrow_{\beta'}} = x{\downarrow_\beta}$, which much be finite.

Next, we show that finitely many elements appear at level n. By Lemma B.4, we know that there is a finite set X such that $\{x \mid \mathsf{lev}_\beta(x) = n\} \subseteq X$ for all $\beta \in D$. Following from Lemma B.3, $\mathsf{lev}_\beta(x) = \mathsf{lev}_{\beta'}(x) = \mathsf{lev}_\alpha(x)$ for all $\beta, \beta' \in D$ and $x \in N_\alpha$, so the elements at level n in α are the elements at level n from all $\beta \in D$. This gives us:

$$\{x \mid \mathsf{lev}_\alpha(x) = n\} = \bigcup_{\beta \in D} \{x \mid \mathsf{lev}_\beta(x) = n\} \subseteq X$$

Now, take any $x \in N_\alpha$. If $\lambda_\alpha(x) = \bot$, then it must be that $\lambda_\beta(x) = \bot$ for all $\beta \in D$ such that $x \in N_\beta$. This means that x has no successors in any $\beta \in D$, therefore it has no successors in α.

If $x <_\alpha y$, then $x <_\beta y$ for some $\beta \in D$. By Remark 1, $\varphi_\beta(x) = \varphi_{\beta'}(x)$ for all $\beta' \in D$ that contain x (and similarly for y); therefore, $\varphi_\alpha(y) = \varphi_\beta(y) \Rightarrow \varphi_\beta(x) = \varphi_\alpha(x)$. Finally, take any $x \in N_\alpha$, so $x \in N_\beta$ for some $\beta \in D$. By construction, $\varphi_\alpha(x) = \varphi_\beta(x) = \psi_x$ and, by Lemma B.3, $x{\downarrow_\alpha} = x{\downarrow_\beta}$; so, since $\mathsf{vars}(\varphi_\beta(x)) \subseteq x{\downarrow_\beta}$, then $\mathsf{vars}(\varphi_\alpha(x)) \subseteq x{\downarrow_\alpha}$.

We next prove property (2): we choose any $\beta \in D$ and show all conditions of $\sqsubseteq_{\mathit{lpo}}$.

First, we prove that $N_\beta \subseteq_\downarrow N_\alpha$; the inclusion is by construction. Then, take $x \in N_\beta$ and $y <_\alpha x$; by construction, there must exist a $\beta' \in D$ such that $y <_{\beta'} x$. Take $\gamma \in D$ to be an upper bound of β and β', which must exist since D is a directed set. So $\beta' \sqsubseteq_{\mathit{lpo}} \gamma$ and therefore $y <_\gamma x$. Since N_β is a downward closed subset of N_γ (being $\beta \sqsubseteq_{\mathit{lpo}} \gamma$), it must be that $y \in N_\beta$.

For the order, we proceed as follows:

$$\begin{aligned} <_\alpha \cap (N_\beta \times N_\beta) &= \left(\bigcup_{\beta' \in D} <_{\beta'} \right) \cap (N_\beta \times N_\beta) \\ &= \bigcup_{\beta' \in D} (<_{\beta'} \cap (N_\beta \times N_\beta)) \end{aligned}$$

$$\overset{\ddagger}{=} <_\beta$$

For the step (‡), we prove that, for every $\beta' \in D$ and every $(x,y) \in <_{\beta'} \cap (N_\beta \times N_\beta)$, we have that $(x,y) \in <_\beta$. Let γ be the upper bound of β and β'; then, $x <_\gamma y$ (being $x <_{\beta'} y$ and $\beta' \sqsubseteq_{lpo} \gamma$). Since $<_\beta = <_\gamma \cap (N_\beta \times N_\beta)$ (because $\beta \sqsubseteq_{lpo} \gamma$), we conclude $x <_\beta y$.

Now, fix $x \in N_\beta$; by Remark 1, $\varphi_\alpha(x) = \psi_x = \varphi_\beta(x)$ and, by definition of sup, $\lambda_\beta(x) \leq \lambda_\alpha(x)$. Finally, by Lemma B.2(3), $\mathsf{succ}_\beta(x) \subseteq \mathsf{succ}_\alpha(x) \setminus \mathsf{Bot}_\beta{\uparrow_\alpha}$. We now show the reverse inclusion. Suppose that $y \in \mathsf{succ}_\alpha(x)$ and $y \notin \mathsf{Bot}_\beta{\uparrow_\alpha}$; by construction, there must be some $\beta' \in D$ such that $x <_{\beta'} y$. By Lemma B.2(1), $y \in \mathsf{succ}_{\beta'}(x)$. Since D is directed, there is a $\gamma \in D$ that is an upper bound of β and β'. Since $\beta' \sqsubseteq_{lpo} \gamma$, then $\mathsf{succ}_{\beta'}(x) = \mathsf{succ}_\gamma(x) \setminus \mathsf{Bot}_{\beta'}{\uparrow_\gamma}$; so, $y \in \mathsf{succ}_\gamma(x)$. Since $\beta \sqsubseteq_{lpo} \gamma$, then we also have $\mathsf{succ}_\beta(x) = \mathsf{succ}_\gamma(x) \setminus \mathsf{Bot}_\beta{\uparrow_\gamma}$. Since $y \notin \mathsf{Bot}_\beta{\uparrow_\alpha}$, then y also cannot be in $\mathsf{Bot}_\beta{\uparrow_\gamma}$ since $<_\gamma \subseteq <_\alpha$ by construction; therefore, it must be that $y \in \mathsf{succ}_\beta(x)$.

Let us now move to property (3): we fix γ satisfying $\beta \sqsubseteq_{lpo} \gamma$ (for all $\beta \in D$) and show that $\alpha \sqsubseteq_{lpo} \gamma$.

We know that $N_\beta \subseteq N_\gamma$ for all $\beta \in D$, therefore $N_\alpha = \bigcup_{\beta \in D} N_\beta \subseteq N_\gamma$. We now need to show that N_α is downwards closed. Take any $x \in N_\alpha$ and $y <_\gamma x$. Since $x \in N_\alpha$, then $x \in N_\beta$ for some $\beta \in D$. Since $N_\beta \subseteq_\downarrow N_\gamma$, then $y \in N_\beta$, and therefore $y \in N_\alpha$.

Second,

$$\begin{aligned}
<_\gamma \cap (N_\alpha \times N_\alpha) &= <_\gamma \cap \left(\bigcup_{\beta \in D} N_\beta \times \bigcup_{\beta \in D} N_\beta \right) \\
&\overset{\dagger}{=} <_\gamma \cap \left(\bigcup_{\beta \in D} (N_\beta \times N_\beta) \right) \\
&= \bigcup_{\beta \in D} (<_\gamma \cap (N_\beta \times N_\beta)) \\
&= \bigcup_{\beta \in D} <_\beta \;=\; <_\alpha
\end{aligned}$$

where the step marked with (†) follows using two set inclusions. The first is $\bigcup_{\beta \in D}(N_\beta \times N_\beta) \subseteq \bigcup_{\beta \in D} N_\beta \times \bigcup_{\beta \in D} N_\beta$, which is immediate by the properties of cartesian product. For the reverse inclusion, take any $(x,y) \in \bigcup_{\beta \in D} N_\beta \times \bigcup_{\beta \in D} N_\beta$, which means that there exist $\beta', \beta'' \in D$ such that $x \in N_{\beta'}$ and $y \in N_{\beta''}$. Since D is a directed set, there exists $\hat{\beta} \in D$ that is an upper bound of β' and β''; thus, $N_{\beta'} \subseteq N_{\hat{\beta}}$ and $N_{\beta''} \subseteq N_{\hat{\beta}}$ and so $N_{\beta'} \times N_{\beta''} \subseteq N_{\hat{\beta}} \times N_{\hat{\beta}}$. The latter implies that $(x,y) \in N_{\hat{\beta}} \times N_{\hat{\beta}} \subseteq \bigcup_{\beta \in D}(N_\beta \times N_\beta)$, as desired.

To conclude, take any $x \in N_\alpha$; since $\beta \sqsubseteq_{lpo} \gamma$ (for all β), we know that $\varphi_\beta(x) = \varphi_\gamma(x)$ and $\lambda_\beta(x) \leq \lambda_\gamma(x)$, for all β that contain x. This allows us to obtain that $\varphi_\alpha(x) = \varphi_\gamma(x)$ and $\lambda_\alpha(x) \leq \lambda_\gamma(x)$.

Finally, by Lemma B.2(3) (with α in place of β and γ in place of α therein), $\mathsf{succ}_\alpha(x) \subseteq \mathsf{succ}_\gamma(x) \setminus \mathsf{Bot}_\alpha{\uparrow_\gamma}$. We now show the reverse inclusion too. Let $y \in \mathsf{succ}_\gamma(x)$ and $y \notin \mathsf{Bot}_\alpha{\uparrow_\gamma}$. Since y is finitely proceeded, we know that $y{\downarrow_\gamma}$ is finite, therefore $y{\downarrow_\gamma} \cap N_\alpha$ is finite and $\lambda_\alpha(z) \neq \bot$ for each $z \in y{\downarrow_\gamma} \cap N_\alpha$ since $y \notin \mathsf{Bot}_\alpha{\uparrow_\gamma}$. We now construct a set A as follows: for each $z \in y{\downarrow_\gamma} \cap N_\alpha$, select a $\beta_z \in D$ such that $z \in N_{\beta_z}$ and $\lambda_{\beta_z}(z) \neq \bot$, which must exist since $\lambda_\alpha(z) \neq \bot$. Since A is a finite subset of D, which is directed, there is an element $\beta \in D$ which is an upper bound of all the elements in A; therefore, $y{\downarrow_\gamma} \cap N_\alpha \subseteq N_\beta$ and $\lambda_\beta(z) \neq \bot$ for each $z \in y{\downarrow_\gamma} \cap N_\alpha$. Since $\beta \in D$, we also know that $\beta \sqsubseteq_{lpo} \gamma$ and, since $x <_\gamma y$ and $x \in N_\alpha$, then $x \in N_\beta$; hence, $\mathsf{succ}_\beta(x) = \mathsf{succ}_\gamma(x) \setminus \mathsf{Bot}_\beta{\uparrow_\gamma}$, so either $y \in \mathsf{succ}_\beta(x)$ or $y \in \mathsf{Bot}_\beta{\uparrow_\gamma}$. In the first case, we are done since $\beta \sqsubseteq_{lpo} \alpha$ and therefore $\mathsf{succ}_\beta(x) \subseteq \mathsf{succ}_\alpha(x)$ by Lemma B.2(3). The second case is a contradiction, since it implies that there is some $z \in \mathsf{Bot}_\beta$ such that $z <_\gamma y$; hence, $z \in y{\downarrow_\gamma} \cap N_\beta \subseteq y{\downarrow_\gamma} \cap N_\alpha$, where the last inclusion holds since $\beta \sqsubseteq_{lpo} \alpha$. This implies that $\lambda_\alpha(z) \neq \bot$, i.e. $z \notin \mathsf{Bot}_\beta$. □

B. Truncation of Lpos

First, we define some notation for lpos, similar to that of pomsets:

$$lpo_{\mathsf{fin}}(L) \triangleq \{\alpha \in lpo(L) \mid |N_\alpha| < \infty\} \qquad \lfloor \alpha \rfloor_{\mathsf{fin}} \triangleq \{\beta \in lpo_{\mathsf{fin}}(L) \mid \beta \sqsubseteq_{lpo} \alpha\}$$

Definition 13 (Truncation)**.** *Let $\alpha \in lpo(L)$ and $n \in \mathbb{N}$. Then, $\lfloor \alpha \rfloor_n = \langle N, <, \lambda, \varphi \rangle$ where*

- $N \triangleq \{x \in N_\alpha \mid \mathsf{lev}_\alpha(x) \leq n\}$;
- $< \triangleq <_\alpha \cap (N \times N)$;
- *for all $x \in N$, let $\lambda(x) \triangleq \lambda_\alpha(x)$, if $\mathsf{lev}_\alpha(x) < n$, and $\lambda(x) \triangleq \bot$, otherwise;*
- *for all $x \in N$, let $\varphi(x) \triangleq \varphi_\alpha(x)$.*

Essentially, the truncation of α with n consists in keeping α unchanged for all nodes $x \in N_\alpha$ such that $\mathsf{lev}_\alpha(x) < n$ and changing the labels of all nodes $x \in \mathsf{lev}^{-1}(n)$ to $\bot$. For any $\alpha \in \mathit{lpo}(L)$, $\lfloor \alpha \rfloor_0$ is a singleton LPOF with $\bot$ at the root.

For a set of LPOFs $S \subseteq \mathit{lpo}(L)$, we also let $\lfloor S \rfloor_n = \{\lfloor \alpha \rfloor_n \mid \alpha \in S\}$. Note that since a pomset is a set of LPOFs, this definition applies to pomsets too.

Lemma B.5. *For any $\alpha \in \mathit{lpo}(L)$ and $n > 0$, it holds that $\lfloor \alpha \rfloor_n \in \lfloor \alpha \rfloor_{\mathsf{fin}}$.*

Proof. To lighten notation, let $\beta = \lfloor \alpha \rfloor_n$. First, note that N_β is finite, since all nodes occur at level at most n, and there can only be finitely many nodes up to that level.

Now, we prove that $\beta \in \mathit{lpo}(L)$. First, β is single rooted, is finitely proceeded and has finitely many nodes at every finite level, because it inherits these properties from α. For the same reason, the conditions on φ_β are satisfied. Finally, for any $x \in N_\beta$ such that $\lambda_\beta(x) = \bot$, we consider two cases:

1) $\mathsf{lev}_\alpha(x) < n$: by construction, $\lambda_\beta(x) = \lambda_\alpha(x)$ and so $\mathsf{succ}_\alpha(x) = \emptyset$. By the definition of $<_\beta$, we have that $\mathsf{succ}_\beta(x) \subseteq \mathsf{succ}_\alpha(x)$ and so also $\mathsf{succ}_\beta(x) = \emptyset$.
2) $\mathsf{lev}_\alpha(x) = n$: obviously x has no successors, since everything above level n has been deleted.

Finally, we show that $\beta \sqsubseteq_{\mathit{lpo}} \alpha$. First, we show that $N_\beta \subseteq_\downarrow N_\alpha$. Clearly $N_\beta \subseteq N_\alpha$ by construction. Now, take any $y \in N_\beta$ and $z <_\alpha y$. Since $\mathsf{lev}_\alpha(y) \leq n$, then $\mathsf{lev}_\alpha(z) < n$ and therefore $z \in N_\beta$. By construction, we also have that $<_\beta = <_\alpha \cap (N_\beta \times N_\beta)$, $\lambda_\beta(y) \leq \lambda_\alpha(y)$, and $\varphi_\beta(y) = \varphi_\alpha(y)$ for all $y \in N_\beta$. It remains only to show the condition on successors. Take any $y \in N_\beta$. By Lemma B.2(3), $\mathsf{succ}_\beta(y) \subseteq \mathsf{succ}_\alpha(y) \setminus \mathsf{Bot}_\beta{\uparrow_\alpha}$; we now show the reverse inclusion. Suppose that $z \in \mathsf{succ}_\alpha(y)$ and $z \notin \mathsf{Bot}_\beta{\uparrow_\alpha}$. Since $y \in N_\beta$, then $\mathsf{lev}_\alpha(z) \leq n+1$. However, it cannot be that $\mathsf{lev}_\alpha(z) = n+1$, since that would imply that there exists $w \in \mathsf{pred}_\alpha(z)$ such that $\mathsf{lev}_\alpha(w) = n$, but then $\lambda_\beta(w) = \bot$, which is a contradiction. So, $\mathsf{lev}_\alpha(z) \leq n$, which means that $z \in N_\beta$. □

An immediate consequence of Lemma B.5 is that $\lfloor \alpha \rfloor_n \in \mathit{lpo}(L)$ and $\lfloor \alpha \rfloor_n \sqsubseteq_{\mathit{lpo}} \alpha$.

Lemma B.6 (Monotonicity of truncation)**.** *If $\alpha \sqsubseteq_{\mathit{lpo}} \beta$, then $\lfloor \alpha \rfloor_n \sqsubseteq_{\mathit{lpo}} \lfloor \beta \rfloor_n$, for every $n \in \mathbb{N}$.*

Proof. By Lemma B.3, we have that $\mathsf{lev}_\alpha(x) = \mathsf{lev}_\beta(x)$, for every $x \in N_\alpha \subseteq N_\beta$; hence, $N_{\lfloor \alpha \rfloor_n} \subseteq N_{\lfloor \beta \rfloor_n}$. For downward closure, let $x \in N_{\lfloor \alpha \rfloor_n}$ and $y <_{\lfloor \beta \rfloor_n} x$; thus, $y \in x{\downarrow_{\lfloor \beta \rfloor_n}}$ and again by Lemma B.3 we easily conclude.

For the order relation, we have that

$$\begin{aligned}
<_{\lfloor \beta \rfloor_n} \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n}) &= (<_\beta \cap (N_{\lfloor \beta \rfloor_n} \times N_{\lfloor \beta \rfloor_n})) \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n}) \\
&= <_\beta \cap ((N_{\lfloor \beta \rfloor_n} \times N_{\lfloor \beta \rfloor_n}) \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n})) \\
&= <_\beta \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n}) \\
&= <_\beta \cap ((N_\alpha \times N_\alpha) \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n})) \\
&= (<_\beta \cap (N_\alpha \times N_\alpha)) \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n}) \\
&= <_\alpha \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n}) = <_{\lfloor \alpha \rfloor_n}
\end{aligned}$$

where all steps follow by definition of truncation, associativity of $\cap$, $\alpha \sqsubseteq_{\mathit{lpo}} \beta$, $N_{\lfloor \alpha \rfloor_n} \subseteq N_{\lfloor \beta \rfloor_n}$ and $N_{\lfloor \alpha \rfloor_n} \subseteq N_\alpha$.

Then, let $x \in N_{\lfloor \alpha \rfloor_n}$. By definition and $\alpha \sqsubseteq_{\mathit{lpo}} \beta$, we have that $\lambda_{\lfloor \alpha \rfloor_n}(x) = \lambda_\alpha(x) \leq \lambda_\beta(x) = \lambda_{\lfloor \beta \rfloor_n}(x)$ and $\varphi_{\lfloor \alpha \rfloor_n}(x) = \varphi_\alpha(x) = \varphi_\beta(x) = \varphi_{\lfloor \beta \rfloor_n}(x)$. By Lemma B.2(3), $\mathsf{succ}_{\lfloor \alpha \rfloor_n}(x) \subseteq \mathsf{succ}_{\lfloor \beta \rfloor_n}(x) \setminus \mathsf{Bot}_{\lfloor \alpha \rfloor_n}{\uparrow_{\lfloor \beta \rfloor_n}}$; let us prove the reverse inclusion. So, let $y \in \mathsf{succ}_{\lfloor \beta \rfloor_n}(x)$; this means that $\mathsf{lev}_\beta(x) < n$ and $\mathsf{lev}_\beta(y) \leq n$. By assumption, $x \in N_\alpha$. If also $y \in N_\alpha$, then $y \in \mathsf{succ}_{\lfloor \alpha \rfloor_n}(x)$, because $\mathsf{lev}_\beta(y) = \mathsf{lev}_\alpha(y)$ and by Lemma B.2(1). If $y \notin N_\alpha$, then $y \in \mathsf{Bot}_\alpha{\uparrow_\beta}$, because $\alpha \sqsubseteq_{\mathit{lpo}} \beta$; this means that there exists $z \in \mathsf{Bot}_\alpha$ such that $z <_\beta y$. Hence, $\mathsf{lev}_\alpha(z) = \mathsf{lev}_\beta(z) < \mathsf{lev}_\beta(y) = \mathsf{lev}_\alpha(y) \leq n$; thus, $z \in N_{\lfloor \alpha \rfloor_n} \subseteq N_{\lfloor \beta \rfloor_n}$ and so $z <_{\lfloor \beta \rfloor_n} y$. By definition of truncation, $\lambda_{\lfloor \alpha \rfloor_n}(z) = \lambda_\alpha(z) = \bot$; hence, we can conclude that $y \in \mathsf{Bot}_{\lfloor \alpha \rfloor_n}{\uparrow_{\lfloor \beta \rfloor_n}}$. □

We extend the truncation to a set of lpos in the obvious way: $\lfloor D \rfloor_n \triangleq \{\lfloor \beta \rfloor_n \mid \beta \in D\}$.

Lemma B.7 (Scott continuity of truncation)**.** *If D is a directed set of* LPOFs*, then $\sup \lfloor D \rfloor_n$ exists and it is $\lfloor \sup D \rfloor_n$, for every $n \in \mathbb{N}$.*

Proof. By Lemma 3, $\sup D = \alpha$, where:

$$N_\alpha \triangleq \bigcup_{\beta \in D} N_\beta \qquad <_\alpha \triangleq \bigcup_{\beta \in D} <_\beta \qquad \lambda_\alpha(x) \triangleq \sup_{\beta \in D \,:\, x \in N_\beta} \lambda_\beta(x) \qquad \varphi_\alpha(x) \triangleq \psi_x$$

We now want to prove that γ, given by

$$N_\gamma \triangleq \bigcup_{\beta \in D} N_{\lfloor \beta \rfloor_n} \qquad <_\gamma \triangleq \bigcup_{\beta \in D} <_{\lfloor \beta \rfloor_n} \qquad \lambda_\gamma(x) \triangleq \sup_{\beta \in D \,:\, x \in N_{\lfloor \beta \rfloor_n}} \lambda_{\lfloor \beta \rfloor_n}(x) \qquad \varphi_\gamma(x) \triangleq \psi_x$$

is the sup of $\lfloor D \rfloor_n$ and that it coincides with $\lfloor \alpha \rfloor_n$.

The fact that $\gamma = \sup \lfloor D \rfloor_n$ follows by Lemma 3, once we prove that $\lfloor D \rfloor_n$ is directed; so, take $\beta_1, \beta_2 \in \lfloor D \rfloor_n$ and prove that they admit an upper bound in $\lfloor D \rfloor_n$. By definition, there exist $\beta_1', \beta_2' \in D$ such that $\beta_1 = \lfloor \beta_1' \rfloor_n$ and $\beta_2 = \lfloor \beta_2' \rfloor_n$. Since D is directed, there exist a $\beta \in D$ such that $\beta_1', \beta_2' \sqsubseteq_{lpo} \beta$. By Lemma B.6, $\beta_1, \beta_2 \sqsubseteq_{lpo} \lfloor \beta \rfloor_n$ and, by definition, $\lfloor \beta \rfloor_n \in \lfloor D \rfloor_n$.

We are left with proving that $\gamma = \lfloor \alpha \rfloor_n$. First, $x \in N_\gamma$ iff there exists $\beta \in D$ such that $x \in N_{\lfloor \beta \rfloor_n}$, i.e. $x \in N_\beta$ and $\mathsf{lev}_\beta(x) = n$; this happens iff $x \in \bigcup_{\beta \in D} N_\beta$ and $\mathsf{lev}_\alpha(x) = n$, being $\mathsf{lev}_\alpha(x) = \mathsf{lev}_\beta(x)$ (since $\beta \sqsubseteq_{lpo} \alpha$), i.e. $x \in N_{\lfloor \alpha \rfloor_n}$. Then,

$$
\begin{aligned}
<_{\lfloor \alpha \rfloor_n} &= <_\alpha \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n}) \\
&= \left(\bigcup_{\beta \in D} <_\beta \right) \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n}) \\
&= \bigcup_{\beta \in D} (<_\beta \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n})) \\
&= \bigcup_{\beta \in D} ((<_\alpha \cap (N_\beta \times N_\beta)) \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n})) \\
&= \bigcup_{\beta \in D} (<_\alpha \cap ((N_\beta \times N_\beta) \cap (N_{\lfloor \alpha \rfloor_n} \times N_{\lfloor \alpha \rfloor_n})) \\
&\overset{\dagger}{=} \bigcup_{\beta \in D} (<_\alpha \cap (N_{\lfloor \beta \rfloor_n} \times N_{\lfloor \beta \rfloor_n})) \\
&\overset{\ddagger}{=} \bigcup_{\beta \in D} <_{\lfloor \beta \rfloor_n} = <_\gamma
\end{aligned}
$$

Step (†) holds because $N_{\lfloor \alpha \rfloor_n} = N_\gamma = \bigcup_{\beta' \in D} N_{\lfloor \beta' \rfloor_n}$ and so $N_{\lfloor \alpha \rfloor_n} \cap N_\beta = (\bigcup_{\beta' \in D} N_{\lfloor \beta' \rfloor_n}) \cap N_\beta = \bigcup_{\beta' \in D} (N_{\lfloor \beta' \rfloor_n} \cap N_\beta) = N_{\lfloor \beta \rfloor_n}$, once we prove that $N_{\lfloor \beta' \rfloor_n} \cap N_\beta \subseteq N_{\lfloor \beta \rfloor_n}$, for every $\beta' \in D$ (indeed, obviously $N_{\lfloor \beta \rfloor_n} \subseteq \bigcup_{\beta' \in D} (N_{\lfloor \beta' \rfloor_n} \cap N_\beta)$). Fix $\beta' \neq \beta$ and $x \in N_{\lfloor \beta' \rfloor_n} \cap N_\beta$; this means that $\mathsf{lev}\beta'(x) \leq n$, and so $\mathsf{lev}_\beta(x) = \mathsf{lev}_\alpha(x) = \mathsf{lev}\beta'(x) \leq n$, since $\beta, \beta' \sqsubseteq_{lpo} \alpha$. This allows us to conclude that $x \in N_{\lfloor \beta \rfloor_n}$, as desired. Step (‡) is proved by double inclusion. Trivially, $x <_{\lfloor \beta \rfloor_n} y$ means that $x <_\beta y$ and $\mathsf{lev}_\beta(x), \mathsf{lev}_\beta(y) \leq n$; hence, $x, y \in N_{\lfloor \beta \rfloor_n}$ and, since $\beta \sqsubseteq_{lpo} \alpha$, we also have $x <_\alpha y$, as desired. Conversely, if $x <_\alpha y$ and $x, y \in N_{\lfloor \beta \rfloor_n}$, then $x, y \in N_\beta$ and, since $\beta \sqsubseteq_{lpo} \alpha$, also $x <_\beta y$; thus, by definition, $x <_{\lfloor \beta \rfloor_n} y$.

Finally, fix $x \in N_\gamma = N_{\lfloor \alpha \rfloor_n}$. By definition, $\varphi_\gamma(x) = \varphi_{\lfloor \alpha \rfloor_n}(x)$. For the labeling, first observe that $\mathsf{lev}_\gamma(x) = \mathsf{lev}_{\lfloor \beta \rfloor_n}(x) = \mathsf{lev}_\alpha(x)$, for every $\beta \in D$ such that $x \in N_\beta$. If $\mathsf{lev}_\gamma(x) < n$, then by construction $\lambda_\gamma(x) = \sup_{\beta \in D \,:\, x \in N_{\lfloor \beta \rfloor_n}} \lambda_{\lfloor \beta \rfloor_n}(x) = \sup_{\beta \in D \,:\, x \in N_\beta} \lambda_\beta(x) = \lambda_\alpha(x) = \lambda_{\lfloor \alpha \rfloor_n}(x)$. Otherwise, $x \in N_{\lfloor \beta \rfloor_n}$ and $\lambda_{\lfloor \beta \rfloor_n}(x) = \bot$, for all β such that $x \in N_\beta$; hence, $\lambda_\gamma(x) = \sup_{\beta \in D \,:\, x \in N_{\lfloor \beta \rfloor_n}} \lambda_{\lfloor \beta \rfloor_n}(x) = \bot = \lambda_{\lfloor \alpha \rfloor_n}(x)$. □

APPENDIX C
POMSETS

The following lemma provides alternative characterizations of the pomset order that will be convenient in some proofs and definitions later on.

Lemma C.1. *The following three statements are equivalent:*

1) $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\beta}$;
2) $\exists \alpha \in \boldsymbol{\alpha}, \beta \in \boldsymbol{\beta}.\ \alpha \sqsubseteq_{lpo} \beta$; *and*
3) $\forall \beta \in \boldsymbol{\beta}.\ \exists \alpha \in \boldsymbol{\alpha}.\ \alpha \sqsubseteq_{lpo} \beta$

Proof. Clearly (1) implies (2) since $\boldsymbol{\alpha}$ is nonempty.

We now show that (2) implies (3), assume that there exist $\alpha \in \boldsymbol{\alpha}$ and $\beta \in \boldsymbol{\beta}$ such that $\alpha \sqsubseteq_{lpo} \beta$. Now, take any $\beta' \in \boldsymbol{\beta}$, so $\beta \equiv \beta'$ and therefore there exists a bijection $f \colon N_\beta \to N_{\beta'}$ that satisfies the properties in Definition 9. Since $\alpha \sqsubseteq_{lpo} \beta$, then $N_\alpha \subseteq N_\beta$, and so we can construct a bijection g such that $g(x) = f(x)$ for all $x \in N_\alpha$. Let $\alpha' \equiv \alpha$ be the lpo generated via the bijection g, so clearly $\alpha' \sqsubseteq_{lpo} \beta'$ since both are generated by the same bijection.

Now, we show that (3) implies (1), assume that forall $\beta \in \boldsymbol{\beta}$ there exists an $\alpha \in \boldsymbol{\alpha}$ such that $\alpha \sqsubseteq_{lpo} \beta$. Then, take any $\alpha' \in \boldsymbol{\alpha}$; by definition, there exists a bijection $f : N_\alpha \to N_{\alpha'}$ that satisfies the properties in Definition 9. Now, consider the LPO β' obtained from β by replacing every $x \in N_\alpha$ with $f(x)$ and every $x \in N_\beta \setminus N_\alpha$ with a node that does not occur in $N_{\alpha'}$. Clearly, this induces an extension of f that is a bijection between N_β and $N_{\beta'}$ that satisfies the properties in Definition 9; hence, $\beta' \in \boldsymbol{\beta}$ and, trivially, $\alpha' \sqsubseteq_{lpo} \beta'$. □

A. Truncation

As mentioned in Appendix B-B, we extend the truncation operation to pomsets in the expected way: $\lfloor \boldsymbol{\alpha} \rfloor_n \triangleq \{\lfloor \alpha \rfloor_n \mid \alpha \in \boldsymbol{\alpha}\}$.

Lemma C.2 (Monotonicity of Truncation)**.** *For every $n \in \mathbb{N}$, if $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\beta}$, then $\lfloor \boldsymbol{\alpha} \rfloor_n \sqsubseteq_{pom} \lfloor \boldsymbol{\beta} \rfloor_n$.*

Proof. By Lemma C.1 we know that there exist $\alpha \in \boldsymbol{\alpha}$ and $\beta \in \boldsymbol{\beta}$ such that $\alpha \sqsubseteq_{lpo} \beta$; by Lemma B.6, this entails that $\lfloor \alpha \rfloor_n \sqsubseteq_{lpo} \lfloor \beta \rfloor_n$ and so, again by Lemma C.1, that $\lfloor \boldsymbol{\alpha} \rfloor_n \sqsubseteq_{pom} \lfloor \boldsymbol{\beta} \rfloor_n$. □

Lemma C.3. *If $\lfloor \boldsymbol{\alpha} \rfloor_n \sqsubseteq_{pom} \boldsymbol{\beta}$ for all $n \in \mathbb{N}$, then $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\beta}$.*

Proof. (adapted from [41, Proposition 3.1.10]) Fix any $\alpha \in \boldsymbol{\alpha}$ and $\beta \in \boldsymbol{\beta}$. We will show that there exists $N \subseteq_\downarrow N_\beta$ and a bijection $f \colon N_\alpha \to N$ such that $f(\alpha) \sqsubseteq_{lpo} \beta$; therefore, by Lemma C.1, $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\beta}$.

We construct a tree of bijections as follows. Each node in the tree is a triple (f, X, n) where $n \in \mathbb{N}$, $X \subseteq_\downarrow N_\beta$, and $f \colon N_{\lfloor \alpha \rfloor_n} \to X$ is a bijection such that $f(\lfloor \alpha \rfloor_n) \sqsubseteq_{lpo} \beta$. We add an edge from (f, X, n) to $(g, Y, n+1)$ if $X \subseteq Y$ and $g(x) = f(x)$ for all $x \in N_{\lfloor \alpha \rfloor_n}$ (*i.e.,* f is g restricted to $N_{\lfloor \alpha \rfloor_n}$ or, equivalently, g is an extension of f). Let x_α and x_β be the roots of α and β respectively, and $f_0 \colon \{x_\alpha\} \to \{x_\beta\}$ be the obvious bijection. We will now show that the structure we described above defines an infinite tree with $(f_0, \{x_\beta\}, 0)$ as the root.

We first show that, for each $n \in \mathbb{N}$, there exists a possible node with n as the third element; hence, there are infinitely many nodes (but we do not yet know that they can be all reached from the root). Fix any $n \in \mathbb{N}$. Since $\lfloor \boldsymbol{\alpha} \rfloor_n \sqsubseteq_{pom} \boldsymbol{\beta} = [\beta]$, then there exists an $\alpha' \in \boldsymbol{\alpha}$ such that $\lfloor \alpha' \rfloor_n \sqsubseteq_{lpo} \beta$. Since $\alpha \equiv \alpha'$, then there must be a bijection $g \colon N_\alpha \to N_{\alpha'}$. Now, let $X = \{g(x) \mid x \in N_{\lfloor \alpha \rfloor_n}\}$ and $f \colon N_{\lfloor \alpha \rfloor_n} \to X$ be defined as $f(x) = g(x)$, which is clearly a bijection since g is. Clearly

$$f(\lfloor \alpha \rfloor_n) = \lfloor g(\alpha) \rfloor_n = \lfloor \alpha' \rfloor_n \sqsubseteq_{lpo} \beta$$

and so (f, X, n) is a node.

We now show that any node (f, X, n) as described above can be reached from the root. The proof is by induction on n. If $n = 0$, then the node must be $(f_0, \{x_\beta\}, 0)$, hence it is the root. If $(f, X, n+1)$ is a node, then $f \colon N_{\lfloor \alpha \rfloor_{n+1}} \to X$ is a bijection. Now, let $Y = \{f(x) \mid x \in N_{\lfloor \alpha \rfloor_n}\}$. So, $g \colon N_{\lfloor \alpha \rfloor_n} \to Y$ defined as $g(x) = f(x)$ is also a bijection. We also know that $f(\lfloor \alpha \rfloor_{n+1}) \sqsubseteq_{lpo} \beta$; since $g(\lfloor \alpha \rfloor_n) \sqsubseteq_{lpo} f(\lfloor \alpha \rfloor_{n+1})$, we have that $g(\lfloor \alpha \rfloor_n) \sqsubseteq_{lpo} \beta$ by transitivity. Therefore, (g, Y, n) is a valid node and there is an edge from (g, Y, n) to $(f, X, n+1)$. By the induction hypothesis, (g, Y, n) is reachable from the root, therefore so is $(f, X, n+1)$.

Now, let $k = |N_{\lfloor \alpha \rfloor_n}|$, which must be finite since only finitely many nodes can occur at each level. Therefore, there can only be finitely many elements in each level of the tree, since there are only only finitely many downward closed subsets of N_β of size k; so, the tree must be finitely branching and, by König's Lemma, there exists an infinite path:

$$(f_n, X_n, n)_{n \in \mathbb{N}} \quad \text{where} \quad X_n \subseteq X_{n+1} \quad \text{and} \quad \forall x \in X_n.\ f_n(x) = f_{n+1}(x) \quad \text{for all } n \in \mathbb{N}$$

Now let $X = \bigcup_{n \in \mathbb{N}} X_n$ and let $f \colon N_\alpha \to X$ be defined as $f(x) = f_{\mathsf{lev}_\alpha(x)}(x)$. We now show that $f(\alpha) \sqsubseteq_{lpo} \beta$. First, clearly $N_{f(\alpha)} = X \subseteq_\downarrow N_\beta$ since it is the union of downward closed sets. Now, for the order, we have:

$$\begin{aligned}
<_{f(\alpha)} &= \bigcup_{n \in \mathbb{N}} <_{f_n(\lfloor \alpha \rfloor_n)} \\
&= \bigcup_{n \in \mathbb{N}} \left(<_\beta \cap \left(N_{f_n(\lfloor \alpha \rfloor_n)} \times N_{f_n(\lfloor \alpha \rfloor_n)}\right)\right) \\
&= <_\beta \cap \bigcup_{n \in \mathbb{N}} \left(N_{f_n(\lfloor \alpha \rfloor_n)} \times N_{f_n(\lfloor \alpha \rfloor_n)}\right) \\
&= <_\beta \cap \left(N_{f(\alpha)} \times N_{f(\alpha)}\right)
\end{aligned}$$

Now, for each $x \in N_{f(\alpha)}$, let $n = \mathsf{lev}_\alpha(x)$. We have that:

$$\lambda_{f(\alpha)}(x) = \lambda_\alpha\left(f^{-1}(x)\right) = \lambda_\alpha\left(f_n^{-1}(x)\right) = \lambda_\alpha\left(f_{n+1}^{-1}(x)\right) = \lambda_{\lfloor \alpha \rfloor_{n+1}}\left(f_{n+1}^{-1}(x)\right) = \lambda_{f_{n+1}(\lfloor \alpha \rfloor_{n+1})}(x) \leq \lambda_\beta(x)$$

By an analogous argument, $\varphi_{f(\alpha)}(x) = \varphi_\beta(x)$. Now, by Lemma B.2, we know that $\mathsf{succ}_{f(\alpha)}(x) \subseteq \mathsf{succ}_\beta(x) \setminus \mathsf{Bot}_{f(\alpha)}{\uparrow_\beta}$, it just remains to show the reverse inclusion. Take any $y \in \mathsf{succ}_\beta(x)$ such that $y \notin \mathsf{Bot}_{f(\alpha)}{\uparrow_\beta}$. Let $m = \mathsf{lev}_\beta(y)$ and note that $y \notin \mathsf{Bot}_{f_m(\lfloor \alpha \rfloor_m)}$ since $f(\alpha)$ and $f_m(\lfloor \alpha \rfloor_m)$ are identical up to level m. Since $f_m(\lfloor \alpha \rfloor_m) \sqsubseteq_{lpo} \beta$, then $y \in \mathsf{succ}_{f_m(\lfloor \alpha \rfloor_m)}(x)$, and therefore $y \in \mathsf{succ}_{f(\alpha)}(x)$. □

B. The Pomset Dcpo Structure

Lemma C.4. *Let $D = \{\alpha_i \mid i \in I\} \subseteq \mathit{lpo}(L)$ be a directed set and let $\boldsymbol{D} = \{[\alpha_i] \mid i \in I\} \subseteq \mathit{pom}(L)$; then, $\sup \boldsymbol{D}$ exists and $\sup \boldsymbol{D} = [\sup D]$.*

Proof. By Lemma 3, we know that D admits a sup, let us call it α and let $\boldsymbol{\alpha} = [\alpha]$; we have to prove that $\boldsymbol{\alpha} = \sup \boldsymbol{D}$, *i.e.,*

1) for all $i \in I$ it holds that $[\alpha_i] \sqsubseteq_{\mathit{pom}} [\alpha]$, and
2) for every $\boldsymbol{\beta}$, if $[\alpha_i] \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta}$ for all $i \in I$, then $\boldsymbol{\alpha} \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta}$.

The first property is trivial, thanks to Lemma C.1(2).

For the second property, take some $\boldsymbol{\beta}$ such that $[\alpha_i] \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta}$ for all $i \in I$ and let $A_n = \{\beta \in \boldsymbol{\beta} \mid \lfloor \alpha \rfloor_n \sqsubseteq_{\mathit{lpo}} \beta\}$.

We first prove that $A_n \neq \emptyset$ for all $n \in \mathbb{N}$. Let $N = \{x \in N_\alpha \mid \mathsf{lev}_\alpha(x) \leq n\}$, which by definition is a finite set. By the definition of supremum for lpos, for each $x \in N$ there exists an $i_x \in I$ such that $x \in N_{\alpha_{i_x}}$ and $\lambda_{\alpha_{i_x}}(x) = \lambda_\alpha(x)$. Since $\{\alpha_{i_x} \mid x \in N\}$ is a finite subset of D, which is directed, there must be a $\alpha_j \in D$ such that $\alpha_{i_x} \sqsubseteq_{\mathit{lpo}} \alpha_j$ for all $x \in N$. By construction, $\lfloor \alpha_j \rfloor_n = \lfloor \alpha \rfloor_n$. Since $\alpha_j \in D$, we also know that $[\alpha_j] \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta}$, so there must be a $\beta \in \boldsymbol{\beta}$ such that $\alpha_j \sqsubseteq_{\mathit{lpo}} \beta$. This gives us $\lfloor \alpha \rfloor_n = \lfloor \alpha_j \rfloor_n \sqsubseteq_{\mathit{lpo}} \alpha_j \sqsubseteq_{\mathit{lpo}} \beta$, so $\beta \in A_n$ and therefore $A_n \neq \emptyset$.

Now, for each $n \in \mathbb{N}$, since $A_n \neq \emptyset$, we know that there exists a $\beta_n \in \boldsymbol{\beta}$ such $\lfloor \alpha \rfloor_n \sqsubseteq_{\mathit{lpo}} \beta_n$; therefore, by Lemma C.1, we get that $\lfloor \boldsymbol{\alpha} \rfloor_n = [\lfloor \alpha \rfloor_n] \sqsubseteq_{\mathit{pom}} [\beta_n] = \boldsymbol{\beta}$ so by Lemma C.3, we get that $\boldsymbol{\alpha} \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta}$. □

Lemma C.5. *For every transfinite chain of pomsets $(\boldsymbol{\alpha}_\delta)_{\delta<\zeta}$ (such that $\boldsymbol{\alpha}_\delta \sqsubseteq_{\mathit{pom}} \boldsymbol{\alpha}_{\delta'}$ if $\delta < \delta'$), there exists a corresponding chain of* LPOFs *$(\alpha_\delta)_{\delta<\zeta}$ such that $\alpha_\delta \in \boldsymbol{\alpha}_\delta$, for all $\delta < \zeta$.*

Proof. We construct the chain $(\alpha_\delta)_{\delta<\zeta}$ by transfinite induction as follows. First, fix an arbitrary $\alpha_0 \in \boldsymbol{\alpha}_0$. Second, for any ordinal δ such that α_δ is fixed, we know that there exists an $\alpha_{\delta+1} \in \boldsymbol{\alpha}_{\delta+1}$ such that $\alpha_\delta \sqsubseteq_{\mathit{lpo}} \alpha_{\delta+1}$; so, fix this element. Finally, for any limit ordinal ξ, consider $\{\alpha_\delta \mid \delta < \xi\}$ that is a chain (thus, also a directed set); we have to choose an $\alpha_\xi \in \boldsymbol{\alpha}_\xi$ that is bigger (w.r.t. $\sqsubseteq_{\mathit{lpo}}$) than any α_δ. By Lemma C.4, we know that $\sup_{\delta<\xi} \boldsymbol{\alpha}_\delta = [\sup_{\delta<\xi} \alpha_\delta]$. By definition, $\boldsymbol{\alpha}_\xi$ is an upper bound of $\{\boldsymbol{\alpha}_\delta \mid \delta < \xi\}$, therefore $\sup_{\delta<\xi} \boldsymbol{\alpha}_\delta \sqsubseteq_{\mathit{pom}} \boldsymbol{\alpha}_\xi$; hence, there exists an $\alpha_\xi \in \boldsymbol{\alpha}_\xi$ such that $\sup_{\delta<\xi} \alpha_\delta \sqsubseteq_{\mathit{lpo}} \alpha_\xi$, which also means that $\alpha_\delta \sqsubseteq_{\mathit{lpo}} \alpha_\xi$ for all $\delta < \xi$. □

Lemma 4. *$\langle \mathit{pom}(L), \sqsubseteq_{\mathit{pom}} \rangle$ is a pointed* DCPO.

Proof. By [30, Proposition 2.1.15], $\langle \mathit{pom}(L), \sqsubseteq_{\mathit{pom}} \rangle$ is a dcpo if any transfinite chain $(\boldsymbol{\alpha}_\delta)_{\delta<\zeta}$ has a supremum. By Lemma C.5, there is a corresponding chain of lpos $(\alpha_\delta)_{\delta<\zeta}$ such that $\alpha_\delta \in \boldsymbol{\alpha}_\delta$ for each $\delta < \zeta$. Therefore, by Lemma C.4, we get that $\sup_{\delta<\zeta} \boldsymbol{\alpha}_\delta = [\sup_{\delta<\zeta} \alpha_\delta]$, so the chain has a supremum.

Finally, pointedness follows from the fact that $\langle \bot \rangle \sqsubseteq_{\mathit{pom}} \boldsymbol{\alpha}$, for every $\boldsymbol{\alpha}$, where we let $\langle \bot \rangle$ to denote the isomorphism class of the singleton LPOs whose node is labelled with $\bot$. □

APPENDIX D
APPROXIMATION AND EXTENSION

We start by recalling the notions of approximation and compactness taken from [30].

Definition 14 (Approximation and Compactness)**.** *Given a* DCPO *$\langle D, \leq \rangle$, the* approximation order *$\ll \subseteq D \times D$ is defined as follows: $d_1 \ll d_2$ iff, for any directed set $D' \subseteq D$ such that $d_2 \leq \sup D'$, then $d_1 \leq d$, for some $d \in D'$.*

An element $d \in D$ is compact *if it approximates itself ($d \ll d$). We denote the set of compact elements of D as $\mathsf{K}(D) \triangleq \{d \in D \mid d \ll d\}$.*

The approximation order is sometimes called the *way-below* relation or the *order of definite refinement* [73]. Intuitively, $d_1 \ll d_2$ means that d_1 is a "simpler representation" of d_2. We now want to show that such an order coincides with the notion of finite approximations (see Lemma 5); this requires some preliminary results for LPOFs.

Lemma D.1. *For any $\alpha \in \mathit{lpo}(L)$, the set $\lfloor \alpha \rfloor_{\mathsf{fin}}$ is directed.*

Proof. Take any $\beta, \gamma \in \lfloor \alpha \rfloor_{\mathsf{fin}}$, so $\beta, \gamma \in \mathit{lpo}_{\mathsf{fin}}(L)$ and $\beta, \gamma \sqsubseteq_{\mathit{lpo}} \alpha$. Let n be the maximum level of any node in β or γ, and $\alpha' = \lfloor \alpha \rfloor_{n+1}$. Clearly, $\alpha' \in \lfloor \alpha \rfloor_{\mathsf{fin}}$ by Lemma B.5.

We conclude the proof by showing that α' is an upper bound for β and γ; we will only show $\beta \sqsubseteq_{\mathit{lpo}} \alpha'$, as the case for γ is identical. Clearly $N_\beta \subseteq N_{\alpha'}$ since β contains a subset of the nodes from α at level at most n and α' contains all nodes from α up to level $n+1$. We will now show that it is downward closed. Take any $x \in N_\beta$ and $y <_{\alpha'} x$; this means that $y <_\alpha x$. Being $N_\beta \subseteq_\downarrow N_\alpha$, we have that $y \in N_\beta$.

Next, since $\alpha' \sqsubseteq_{\mathit{lpo}} \alpha$ and $N_\beta \subseteq N_{\alpha'}$, we have that:

$$\begin{aligned} <_{\alpha'} \cap (N_\beta \times N_\beta) &= (<_\alpha \cap (N_{\alpha'} \times N_{\alpha'})) \cap (N_\beta \times N_\beta) \\ &= <_\alpha \cap ((N_{\alpha'} \cap N_\beta) \times (N_{\alpha'} \cap N_\beta)) \end{aligned}$$

$$= <_\alpha \cap (N_\beta \times N_\beta)$$
$$= <_\beta$$

Finally, let $x \in N_\beta$. By construction, $\varphi_\beta(x) = \varphi_\alpha(x) = \varphi_{\alpha'}(x)$ and $\lambda_\beta(x) \leq \lambda_\alpha(x) = \lambda_{\alpha'}(x)$, since $\mathsf{lev}_\alpha(x) \leq n$. For the last requirement, by Lemma B.2(3) we have that $\mathsf{succ}_\beta(x) \subseteq \mathsf{succ}_{\alpha'}(x) \setminus \mathsf{Bot}_\beta{\uparrow}_{\alpha'}$; we need to prove the reverse inclusion. Let $y \in \mathsf{succ}_{\alpha'}(x)$ and $y \notin \mathsf{Bot}_\beta{\uparrow}_{\alpha'}$. We have two cases:

- $y \in N_\beta$: in this case, $y \in \mathsf{succ}_\beta(x)$ by Lemma B.2(1).
- $y \notin N_\beta$: Because y is finitely proceeded in α', there exist $y_1, \ldots, y_n$ such that $y_1 = \min_{\alpha'}$, $y_n = y$ and $y_{i+1} \in \mathsf{succ}_{\alpha'}(y_i)$. Because of Lemma B.2(2), $y_{i+1} \in \mathsf{succ}_\alpha(y_i)$, for all i. Being $y_1 = \min_\beta$, there exists an i $(1 < i < n)$ such that $y_i \in N_\beta$ and $y_{i+1} \notin N_\beta$. Since $\beta \sqsubseteq_{lpo} \alpha$, we have that $\mathsf{succ}_\beta(y_i) \subseteq \mathsf{succ}_\alpha(y_i) \setminus \mathsf{Bot}_\beta{\uparrow}_\alpha$; so, $y_{i+1} \in \mathsf{Bot}_\beta{\uparrow}_\alpha$ and, consequently, $y \in \mathsf{Bot}_\beta{\uparrow}_\alpha$. This latter fact, together with $y \in N_{\alpha'}$, implies that $y \in \mathsf{Bot}_\beta{\uparrow}_{\alpha'}$: contradiction. □

Lemma D.2. *For any $\boldsymbol{\alpha} \in pom(L)$, the set $\lfloor \boldsymbol{\alpha} \rfloor_{\mathsf{fin}}$ is directed.*

Proof. Take any $\boldsymbol{\beta}, \boldsymbol{\beta}' \in \lfloor \boldsymbol{\alpha} \rfloor_{\mathsf{fin}}$, so $\boldsymbol{\beta}, \boldsymbol{\beta}' \in pom_{\mathsf{fin}}(L)$ and $\boldsymbol{\beta} \sqsubseteq_{pom} \boldsymbol{\alpha}$ and $\boldsymbol{\beta}' \sqsubseteq_{pom} \boldsymbol{\alpha}$. Now take any $\alpha \in \boldsymbol{\alpha}$, by Lemma C.1 we know that there is a $\beta \in \boldsymbol{\beta}$ and $\beta' \in \boldsymbol{\beta}'$ such that $\beta \sqsubseteq_{lpo} \alpha$ and $\beta' \sqsubseteq_{lpo} \alpha$. Therefore, $\beta, \beta' \in \lfloor \alpha \rfloor_{\mathsf{fin}}$, so by Lemma D.1, there exists a $\gamma \in \lfloor \alpha \rfloor_{\mathsf{fin}}$ such that $\beta \sqsubseteq_{lpo} \gamma$ and $\beta' \sqsubseteq_{lpo} \gamma$. Since $\gamma \in \lfloor \alpha \rfloor_{\mathsf{fin}}$, then $[\gamma] \in \lfloor \boldsymbol{\alpha} \rfloor_{\mathsf{fin}}$, and clearly $\boldsymbol{\beta} \sqsubseteq_{pom} [\gamma]$ and $\boldsymbol{\beta}' \sqsubseteq_{pom} [\gamma]$. □

Lemma D.3. *For any $\alpha \in lpo(L)$, $\alpha = \sup \lfloor \alpha \rfloor_{\mathsf{fin}}$.*

Proof. By Lemma D.1, $\lfloor \alpha \rfloor_{\mathsf{fin}}$ is a directed set, therefore its supremum is given by the construction in Lemma 3. We now show that this gives us exactly α. First, we show this for the nodes. We know that $N_{\sup \lfloor \alpha \rfloor_{\mathsf{fin}}} = \bigcup_{\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}}} N_\beta$. Clearly, $\bigcup_{\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}}} N_\beta \subseteq N_\alpha$ since each $N_\beta \subseteq N_\alpha$ by definition. For the reverse inclusion, take any $x \in N_\alpha$ and let $n = \mathsf{lev}_\alpha(x)$. Obviously, $x \in N_{\lfloor \alpha \rfloor_n}$, and by Lemma B.5, we get that $\lfloor \alpha \rfloor_n \in \lfloor \alpha \rfloor_{\mathsf{fin}}$. Therefore, $x \in \bigcup_{\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}}} N_\beta$, so $N_\alpha \subseteq \bigcup_{\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}}} N_\beta$ and thus $\bigcup_{\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}}} N_\beta = N_\alpha$

By definition, $\beta \sqsubseteq_{lpo} \alpha$, for each $\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}}$; so clearly α is an upper bound for $\lfloor \alpha \rfloor_{\mathsf{fin}}$ and therefore $\sup \lfloor \alpha \rfloor_{\mathsf{fin}} \sqsubseteq_{lpo} \alpha$ (since the supremum is the least upper bound). Hence, $<_{\sup \lfloor \alpha \rfloor_{\mathsf{fin}}} \subseteq <_\alpha$; we show the reverse inclusion as follows. Take any $x <_\alpha y$ and let $\gamma = \lfloor \alpha \rfloor_{\mathsf{lev}_\alpha(y)}$. Clearly $y \in N_\gamma$ and by Lemma B.5, $\gamma \in \lfloor \alpha \rfloor_{\mathsf{fin}}$. Since $N_\gamma \subseteq_\downarrow N_\alpha$, then $x \in N_\gamma$ too, which means that $x <_\gamma y$. This gives us:

$$(x, y) \quad \in \quad <_\gamma \quad \subseteq \quad \bigcup_{\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}}} <_\beta \quad = \quad <_{\sup \lfloor \alpha \rfloor_{\mathsf{fin}}}$$

Finally, for any $x \in N_\alpha$, we have:

- $\varphi_{\sup \lfloor \alpha \rfloor_{\mathsf{fin}}}(x) = \varphi_\alpha(x)$ since $\varphi_\beta(x) = \varphi_\alpha(x)$ for every $\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}}$ such that $x \in N_\beta$ (see Remark 1).
- $\lambda_{\sup \lfloor \alpha \rfloor_{\mathsf{fin}}}(x) = \sup_{\beta \in \lfloor \alpha \rfloor_{\mathsf{fin}} : x \in N_\beta} \lambda_\beta(x)$
  $= \sup\{\lambda_\beta(x) \mid \beta \in lpo_{\mathsf{fin}}(L), \beta \sqsubseteq_{lpo} \alpha, x \in N_\beta\}$
  $\overset{1}{=} \sup\{\ell \mid \ell \leq \lambda_\alpha(x)\}$
  $\overset{2}{=} \lambda_\alpha(x)$

  where (1) holds by the definition of $\sqsubseteq_{lpo}$, that implies that the set above must include all the labels that are less than $\lambda_\alpha(x)$, and (2) holds since the set contains $\lambda_\alpha(x)$ itself. □

Lemma 5. *For any $\boldsymbol{\alpha} \in pom(L)$, $\boldsymbol{\alpha} = \sup \lfloor \boldsymbol{\alpha} \rfloor_{\mathsf{fin}}$.*

Proof. Fix any $\alpha \in \boldsymbol{\alpha}$ and consider the set $\lfloor \alpha \rfloor_{\mathsf{fin}}$: by Lemma D.1 it is directed and by Lemma D.3 its sup is α. Then, by Lemma C.4, $\sup \lfloor \boldsymbol{\alpha} \rfloor_{\mathsf{fin}}$ exists and it is $[\alpha]$, i.e. $\boldsymbol{\alpha}$. □

Lemma D.4. *If $\alpha \in lpo_{\mathsf{fin}}(L)$, then there are only finitely many β such that $\beta \sqsubseteq_{lpo} \alpha$.*

Proof. Any $\beta \sqsubseteq_{lpo} \alpha$ is such that $N_\beta \subseteq N_\alpha$; in other words, $N_\beta \in \mathcal{P}(N_\alpha)$. If $|N_\alpha| = n$. then $|\mathcal{P}(N_\alpha)| = 2^n$ (actually, there are fewer possibilities, since N_β is downward closed, but 2^n is a sufficient upper bound). The order and formula of β are deterministically determined by N_β, since $<_\beta = <_\alpha \cap (N_\beta \times N_\beta)$ and $\varphi_\beta(x) = \varphi_\alpha(x)$. Let $k = \max_{x \in N_\alpha} |\lambda_\alpha(x){\downarrow_L}|$, which must be a finite integer since it is the maximum of a finite set of integers. So, there are at most 2^n subsets of N_α and for each of those subsets, there are at most k choices for the labels of each node. Hence, there are at most $2^n \cdot n \cdot k$ possible $\beta \sqsubseteq_{lpo} \alpha$. □

Lemma D.5. *If $\boldsymbol{D} \subseteq pom(L)$ is a directed set, then $\lfloor \boldsymbol{D} \rfloor_n$ is directed and finite, for any $n \in \mathbb{N}$.*

Proof. We first show that $\lfloor \boldsymbol{D} \rfloor_n$ is directed. Take any $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \lfloor \boldsymbol{D} \rfloor_n$. This means that there is $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2 \in \boldsymbol{D}$ such that $\boldsymbol{\beta}_i = \lfloor \boldsymbol{\gamma}_i \rfloor_n$ for each $i \in \{1, 2\}$, and since $\boldsymbol{D}$ is directed, then there is a $\boldsymbol{\gamma} \in \boldsymbol{D}$ such that each $\boldsymbol{\gamma}_i \sqsubseteq_{pom} \boldsymbol{\gamma}$. By monotonicity of truncation, we have that $\boldsymbol{\beta}_i = \lfloor \boldsymbol{\gamma}_i \rfloor_n \sqsubseteq_{pom} \lfloor \boldsymbol{\gamma} \rfloor_n$, and clearly $\lfloor \boldsymbol{\gamma} \rfloor_n \in \lfloor \boldsymbol{D} \rfloor_n$, therefore $\lfloor \boldsymbol{D} \rfloor_n$ is directed.

Because of Lemma C.2, we know that $\boldsymbol{\beta} \sqsubseteq_{pom} \lfloor \boldsymbol{\alpha} \rfloor_n$ for all $\boldsymbol{\beta} \in \lfloor \boldsymbol{D} \rfloor_n$. Fix any $\alpha \in \lfloor \boldsymbol{\alpha} \rfloor_n$, so clearly α is finite. By Lemma C.1, we know that, for every $\boldsymbol{\beta} \in \lfloor \boldsymbol{D} \rfloor_n$, there is a $\beta \in \boldsymbol{\beta}$ such that $\beta \sqsubseteq_{lpo} \alpha$. By Lemma D.4, there can only be finitely many such elements. Thus we have that $\{\beta \mid \beta \sqsubseteq_{lpo} \alpha\}$ is finite, and also clearly $\lfloor \boldsymbol{D} \rfloor_n \subseteq \{[\beta] \mid \beta \sqsubseteq_{lpo} \alpha\}$; therefore, $\lfloor \boldsymbol{D} \rfloor_n$ is also finite. □

Lemma D.6. *If $\boldsymbol{D}$ is a directed set of pomsets, then $\sup \lfloor \boldsymbol{D} \rfloor_n = \lfloor \sup \boldsymbol{D} \rfloor_n$ for all $n \in \mathbb{N}$.*

Proof. We show that $\lfloor - \rfloor_n$ is chain continuous; indeed, by [71, Corollary 3], any chain continuous function on a DCPO is also Scott continuous, and thus $\sup \lfloor \boldsymbol{D} \rfloor_n = \lfloor \sup \boldsymbol{D} \rfloor_n$.

Take any pomset chain $(\boldsymbol{\alpha}_\delta)_{\delta < \zeta}$; by Lemma C.5, there is a corresponding chain of LPOFs $(\alpha_\delta)_{\delta < \zeta}$ such that $\alpha_\delta \in \boldsymbol{\alpha}_\delta$ for all $\delta < \zeta$. Since truncation is monotonic (Lemma C.2), then $(\lfloor \boldsymbol{\alpha}_\delta \rfloor_n)_{\delta < \zeta}$ is also a chain and, by Lemma B.7, we know that $\sup_{\delta<\zeta} \lfloor \alpha_\delta \rfloor_n = \lfloor \sup_{\delta<\zeta} \alpha_\delta \rfloor_n$. Clearly $\lfloor \alpha_\delta \rfloor_n \in \lfloor \boldsymbol{\alpha}_\delta \rfloor_n$ for all $\delta < \zeta$, therefore by Lemma C.4, we get both $\sup_{\delta<\zeta} \boldsymbol{\alpha}_\delta = [\sup_{\delta<\zeta} \alpha_\delta]$ and $\sup_{\delta<\zeta} \lfloor \boldsymbol{\alpha}_\delta \rfloor_n = [\sup_{\delta<\zeta} \lfloor \alpha_\delta \rfloor_n]$. Putting all these facts together, we get:

$$\sup_{\delta<\zeta} \lfloor \boldsymbol{\alpha}_\delta \rfloor_n = [\sup_{\delta<\zeta} \lfloor \alpha_\delta \rfloor_n] = [\lfloor \sup_{\delta<\zeta} \alpha_\delta \rfloor_n] = \lfloor [\sup_{\delta<\zeta} \alpha_\delta] \rfloor_n = \lfloor \sup_{\delta<\zeta} \boldsymbol{\alpha}_\delta \rfloor_n$$

□

Lemma D.7 (Compactness)**.** $\mathsf{K}(pom(L)) = pom_{\mathsf{fin}}(L)$.

Proof. We first show that any compact pomset is finite, *i.e.,* $\mathsf{K}(pom(L)) \subseteq pom_{\mathsf{fin}}(L)$. Take any $\boldsymbol{\alpha} \in \mathsf{K}(pom(L))$, so for any directed set A, if $\boldsymbol{\alpha} \sqsubseteq_{lpo} \sup A$ then $\boldsymbol{\alpha} \sqsubseteq_{lpo} \boldsymbol{\gamma}$ for some $\boldsymbol{\gamma} \in A$. Now, let $A = \lfloor \boldsymbol{\alpha} \rfloor_{\mathsf{fin}}$, which is directed by Lemma D.2. By Lemma 5, we also know that $\boldsymbol{\alpha} = \sup A$, so $\boldsymbol{\alpha} \sqsubseteq_{pom} \sup A$. Since $\boldsymbol{\alpha}$ is compact, there must be some $\boldsymbol{\gamma} \in A$ such that $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\gamma}$. Since $\boldsymbol{\gamma} \in A$, then $\boldsymbol{\gamma} \in pom_{\mathsf{fin}}(L)$; therefore, since $\boldsymbol{\alpha} \sqsubseteq_{lpo} \boldsymbol{\gamma}$, then $\boldsymbol{\alpha} \in pom_{\mathsf{fin}}(L)$ too.

We now show that any finite pomset is compact, *i.e.,* $pom_{\mathsf{fin}}(L) \subseteq \mathsf{K}(pom(L))$. Taking any $\boldsymbol{\alpha} \in pom_{\mathsf{fin}}(L)$, we must show that $\boldsymbol{\alpha} \ll \boldsymbol{\alpha}$. That is, for any directed set A, if $\boldsymbol{\alpha} \sqsubseteq_{pom} \sup A$, then there exists $\boldsymbol{\gamma} \in A$ such that $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\gamma}$.

To this aim, let A be a directed set such that $\boldsymbol{\alpha} \sqsubseteq_{pom} \sup A$. Since $\boldsymbol{\alpha}$ is finite, let $\alpha \in \boldsymbol{\alpha}$ be an arbitrary representative lpo, then $n = \max_{x \in N_\alpha} \mathsf{lev}(x) + 1$ must be finite. Moreover, $\lfloor \boldsymbol{\alpha} \rfloor_n = \boldsymbol{\alpha}$, since n is one level above the maximum level of $\boldsymbol{\alpha}$ and therefore the truncation only converts labels to $\bot$ above the maximum level of $\boldsymbol{\alpha}$ and removes nodes even above that. Therefore, by montonicity (Lemma C.2) and Scott continuity (Lemma D.6) of truncation, we have:

$$\boldsymbol{\alpha} = \lfloor \boldsymbol{\alpha} \rfloor_n \sqsubseteq_{pom} \lfloor \sup A \rfloor_n = \sup\{\lfloor \boldsymbol{\beta} \rfloor_n \mid \boldsymbol{\beta} \in A\}$$

By Lemma D.5, we know that $\{\lfloor \boldsymbol{\beta} \rfloor_n \mid \boldsymbol{\beta} \in A\}$ is a finite directed set, and therefore it must contain its own supremum. That is, there exists $\boldsymbol{\gamma} \in \{\lfloor \boldsymbol{\beta} \rfloor_n \mid \boldsymbol{\beta} \in A\}$ such that $\boldsymbol{\gamma} = \sup\{\lfloor \boldsymbol{\beta} \rfloor_n \mid \boldsymbol{\beta} \in A\}$. This means that there exists a $\boldsymbol{\gamma}' \in A$ such that $\boldsymbol{\gamma} = \lfloor \boldsymbol{\gamma}' \rfloor_n$, and so clearly $\boldsymbol{\gamma} \sqsubseteq_{pom} \boldsymbol{\gamma}'$, and therefore by transitivity, $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\gamma}'$. □

Corollary 5 (Approximation)**.** $\boldsymbol{\alpha} \ll \boldsymbol{\beta}$ *iff* $\boldsymbol{\alpha} \in \lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}}$.

Proof. Suppose that $\boldsymbol{\alpha} \ll \boldsymbol{\beta}$. Since $\lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}}$ is a directed set and $\sup \lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}} = \boldsymbol{\beta}$ (Lemma D.2 and lemma 5), then there must be some $\boldsymbol{\gamma} \in \lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}}$ such that $\boldsymbol{\alpha} \sqsubseteq_{lpo} \boldsymbol{\gamma}$. Since $\lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}}$ is downward closed by definition, then $\boldsymbol{\alpha} \in \lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}}$ too.

Now suppose that $\boldsymbol{\alpha} \in \lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}}$. By Lemma D.7, $\boldsymbol{\alpha}$ is compact, so $\boldsymbol{\alpha} \ll \boldsymbol{\alpha}$. By the definition of $\lfloor \boldsymbol{\beta} \rfloor_{\mathsf{fin}}$, $\boldsymbol{\alpha} \sqsubseteq_{lpo} \boldsymbol{\beta}$ too. Therefore, by [30, Proposition 2.2.2(2)], we get that $\boldsymbol{\alpha} \ll \boldsymbol{\beta}$. □

For any DCPOs $\langle X, \leq_X \rangle$ and $\langle Y, \leq_Y \rangle$, the product poset $\langle X \times Y, \leq \rangle$ where $(x, y) \leq (x', y')$ iff $x \leq_X x'$ and $y \leq_Y y'$ is also a DCPO [30, Proposition 3.2.2]. We will use this fact in the next lemma. We write $\vec{x} \in X_1 \times \cdots \times X_n$ to denote elements of Cartesian products. Let $\vec{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n)$ and $\vec{\boldsymbol{\beta}} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n)$. We write $\vec{\boldsymbol{\alpha}} \ll \vec{\boldsymbol{\beta}}$ to mean that $\boldsymbol{\alpha}_i \ll \boldsymbol{\beta}_i$ for all $1 \leq i \leq n$.

Lemma D.8. *For any directed set $D \subseteq pom(X)^n$:*

$$\left\{\vec{\boldsymbol{\beta}} \mid \exists \vec{\boldsymbol{\alpha}} \in D. \vec{\boldsymbol{\beta}} \ll \vec{\boldsymbol{\alpha}}\right\} = \left\{\vec{\boldsymbol{\beta}} \mid \vec{\boldsymbol{\beta}} \ll \sup D\right\}$$

Proof. We show the set inclusion in both directions. First, take any element $\vec{\boldsymbol{\beta}} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n)$ from the first set, so we know that $\boldsymbol{\beta}_i \ll \boldsymbol{\alpha}_i$ for all $1 \leq i \leq n$ and some $\vec{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n) \in D$. Clearly $\vec{\boldsymbol{\alpha}} \sqsubseteq_{pom} \sup D$, so $\vec{\boldsymbol{\beta}} \ll \sup D$ [30, Proposition 2.2.2(2)] and therefore the forward inclusion holds.

Now take any element $\vec{\boldsymbol{\beta}} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n)$ from the second set and let $D_1, \ldots, D_n$ be the projections of D. Since the supremum on the product order is coordinatewise [30, Proposition 2.2.2(2)], then $\sup D = (\sup D_1, \ldots, \sup D_n)$. So, $\boldsymbol{\beta}_i \ll \sup D_i$ for all $1 \leq i \leq n$. Therefore, by the definition of $\ll$, there must be some $\boldsymbol{\alpha}_i \in D_i$ such that $\boldsymbol{\beta}_i \sqsubseteq_{pom} \boldsymbol{\alpha}_i$. Now, for each i, let $\vec{\boldsymbol{\alpha}}_i \in D$ be the tuple whose $i-$th projection is $\boldsymbol{\alpha}_i$. Since D is directed, then there is some $\vec{\boldsymbol{\alpha}} \in D$, which is an upper bound of all the $\vec{\boldsymbol{\alpha}}_i$s. From $\boldsymbol{\beta}_i \ll \sup D_i$ and Corollary 5, we know that $\boldsymbol{\beta}_i$ is finite, therefore $\boldsymbol{\beta}_i \in \lfloor \boldsymbol{\alpha}_i \rfloor_{\mathsf{fin}}$. By Corollary 5 again, $\boldsymbol{\beta}_i \ll \boldsymbol{\alpha}_i$, and so $\vec{\boldsymbol{\beta}} \ll \boldsymbol{\alpha}$ and the reverse inclusion holds too. □

Lemma 6 (Extension)**.** *Let* $f\colon pom_{\mathsf{fin}}(L)^n \to T$ *be a monotone function on the* DCPO $\langle T, \leq\rangle$*. Then* $f^*\colon pom(L)^n \to T$*, defined as:*

$$f^*(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n) \triangleq \sup_{\boldsymbol{\alpha}'_1 \ll \boldsymbol{\alpha}_1} \cdots \sup_{\boldsymbol{\alpha}'_n \ll \boldsymbol{\alpha}_n} f(\boldsymbol{\alpha}'_1, \ldots, \boldsymbol{\alpha}'_n)$$

is well-defined and Scott continuous.

Proof. By Corollary 5, $\{(\boldsymbol{\alpha}'_1, \ldots, \boldsymbol{\alpha}'_n) \mid \forall i.\, \boldsymbol{\alpha}'_i \ll \boldsymbol{\alpha}_i\} = \lfloor \boldsymbol{\alpha}_1 \rfloor_{\mathsf{fin}} \times \cdots \times \lfloor \boldsymbol{\alpha}_n \rfloor_{\mathsf{fin}}$, that is a directed set by Lemma D.2. Since f is monotone, then $\{f(\boldsymbol{\alpha}'_1, \ldots, \boldsymbol{\alpha}'_n) \mid \forall i.\, \boldsymbol{\alpha}'_i \ll \boldsymbol{\alpha}_i\}$ is also a directed set, and so clearly the supremum exists.

We now establish that f^* is Scott continuous. Let $D \subseteq pom(L)^n$ be a directed set and:

$$\begin{aligned}
\sup_{\vec{\boldsymbol{\alpha}} \in D} f^*(\vec{\boldsymbol{\alpha}}) &= \sup_{\vec{\boldsymbol{\alpha}} \in D} \sup_{\vec{\boldsymbol{\beta}} \ll \vec{\boldsymbol{\alpha}}} f(\vec{\boldsymbol{\beta}}) \\
&= \sup \left\{ f(\vec{\boldsymbol{\beta}}) \mid \exists \vec{\boldsymbol{\alpha}} \in D.\, \vec{\boldsymbol{\beta}} \ll \vec{\boldsymbol{\alpha}} \right\} && \text{By [30, Proposition 2.1.4(3)].} \\
&= \sup \left\{ f(\vec{\boldsymbol{\beta}}) \mid \vec{\boldsymbol{\beta}} \ll \sup D \right\} && \text{By Lemma D.8.} \\
&= f^*(\sup D)
\end{aligned}$$

□

APPENDIX E
OPERATIONS

A. Sequential Composition

Lemma E.1. *If* $\alpha \sqsubseteq_{lpo} \beta$*, then* $\mathsf{stuck}_\beta \Rightarrow \mathsf{stuck}_\alpha$*.*

Proof. For any $z \in \mathsf{Bot}_\beta$, either $z \in \mathsf{Bot}_\alpha$ or $z \notin N_\alpha$. In the first case, $\varphi_\beta(z) = \varphi_\alpha(z)$, being $\alpha \sqsubseteq_{lpo} \beta$; in the second case, by Lemma 1, there is some $w \in \mathsf{Bot}_\alpha$ such that $w <_\beta z$, which implies that $\varphi_\beta(z) \Rightarrow \varphi_\alpha(w)$. This means that every term in $\bigvee_{z \in \mathsf{Bot}_\beta} \varphi_\beta(z)$ implies some term in $\bigvee_{y \in \mathsf{Bot}_\alpha} \varphi_\alpha(y)$, therefore $\bigvee_{z \in \mathsf{Bot}_\beta} \varphi_\beta(z) \Rightarrow \bigvee_{y \in \mathsf{Bot}_\alpha} \varphi_\alpha(y)$. □

Lemma E.2. *If* $\alpha \sqsubseteq_{lpo} \beta$*, then* $\mathsf{ext}_\alpha = \{x \in \mathsf{ext}_\beta \mid \varphi_\beta(x) \not\Rightarrow \mathsf{stuck}_\alpha\}$*.*

Proof. We show the inclusion in both directions.

Take any $x \in \mathsf{ext}_\alpha$, this means that $x \in N_\alpha$ and $\varphi_\alpha(x) \not\Rightarrow \mathsf{stuck}_\alpha$; so, there is a valuation v such that $v \vDash \varphi_\alpha(x)$ and $v \nvDash \mathsf{stuck}_\alpha$. By Lemma E.1, $\mathsf{stuck}_\beta \Rightarrow \mathsf{stuck}_\alpha$, therefore v cannot satisfy stuck_β either. Since $\varphi_\alpha(x) = \varphi_\beta(x)$ (because $\alpha \sqsubseteq_{lpo} \beta$), we have that $\varphi_\beta(x) \not\Rightarrow \mathsf{stuck}_\alpha$ and that $v \vDash \varphi_\beta(x)$ and $v \nvDash \mathsf{stuck}_\beta$; therefore, $\varphi_\beta(x) \not\Rightarrow \mathsf{stuck}_\beta$ and so $x \in \mathsf{ext}_\beta$.

For the reverse inclusion, take any $x \in \mathsf{ext}_\beta$ such that $\varphi_\beta(x) \not\Rightarrow \mathsf{stuck}_\alpha$. By Lemma 1, we know that either $x \in N_\alpha$ or $x \in \mathsf{Bot}_\alpha{\uparrow_\beta}$. In the former case, we know that $\varphi_\alpha(x) = \varphi_\beta(x) \not\Rightarrow \mathsf{stuck}_\alpha$, therefore $x \in \mathsf{ext}_\alpha$. The latter case is a contradiction, since $x \in \mathsf{Bot}_\alpha{\uparrow_\beta}$ means that there is some $y \in \mathsf{Bot}_\alpha$ such that $y <_\beta x$, which entails that $\varphi_\beta(x) \Rightarrow \varphi_\beta(y)$; however, $\varphi_\beta(y) = \varphi_\alpha(y) \Rightarrow \mathsf{stuck}_\alpha$ and, by combining these facts, we get that $\varphi_\beta(x) \Rightarrow \mathsf{stuck}_\alpha$. □

Lemma 8 (Monotonicity of branches)**.** *If* $\alpha \sqsubseteq_{lpo} \beta$*, then* $\mathsf{br}_\alpha = \{\psi \in \mathsf{br}_\beta \mid \psi \Rightarrow \neg\mathsf{stuck}_\alpha\}$*.*

Proof. We show the inclusion in both directions.

For the forward inclusion, take $\psi \in \mathsf{br}_\alpha$, i.e. $\psi = \varphi_\alpha(S)$, for some $S \subseteq_{\mathsf{fin}} \mathsf{ext}_\alpha$ maximally satisfiable and that does not imply stuck_α; we want to show that $\psi = \varphi_\beta(S) \in \mathsf{br}_\beta$. By Lemma E.2, we know that $\mathsf{ext}_\alpha \subseteq \mathsf{ext}_\beta$, therefore $S \subseteq_{\mathsf{fin}} \mathsf{ext}_\beta$. Further, since $\varphi_\alpha(x) = \varphi_\beta(x)$ for all $x \in N_\alpha$, then $\varphi_\beta(S) = \varphi_\alpha(S) \Rightarrow \neg\mathsf{stuck}_\alpha$; by using the contrapositive of Lemma E.1, we get that $\varphi_\beta(S) \Rightarrow \neg\mathsf{stuck}_\beta$. Last, take any T such that $S \subset T \subseteq \mathsf{ext}_\beta$. If $T \subseteq \mathsf{ext}_\alpha$, we already know that $\varphi_\alpha(T)$ cannot be satisfiable; so suppose that there is some node $x \in T$ such that $x \notin \mathsf{ext}_\alpha$, meaning that either $x \notin N_\alpha$ or $\varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha$. In the first case, by Lemma 1 $x \in \mathsf{Bot}_\alpha{\uparrow_\beta}$, so there is a $z \in \mathsf{Bot}_\alpha$ such that $z <_\beta x$, which also means that $\varphi_\beta(x) \Rightarrow \varphi_\beta(z) \Rightarrow \mathsf{stuck}_\alpha$. So, in either case, we have $\varphi_\beta(x) \Rightarrow \mathsf{stuck}_\alpha$, while, since $\varphi_\alpha(S) \in \mathsf{br}_\alpha$, we know that $\varphi_\alpha(S) \Rightarrow \neg\mathsf{stuck}_\alpha$. These give us:

$$\begin{aligned}
\varphi_\beta(T) &\Rightarrow \varphi_\beta(S) \wedge \varphi_\beta(x) \\
&= \varphi_\alpha(S) \wedge \varphi_\beta(x) \\
&\Rightarrow \neg\mathsf{stuck}_\alpha \wedge \mathsf{stuck}_\alpha \\
&\Rightarrow \mathsf{false}
\end{aligned}$$

So, $\varphi_\beta(T)$ is not satisfiable, and therefore we have shown that $\varphi_\beta(S) \in \mathsf{br}_\beta$ (and, clearly, $\varphi_\beta(S) \Rightarrow \neg\mathsf{stuck}_\alpha$).

For the reverse inclusion, take any $\psi \in \mathsf{br}_\beta$ such that $\psi \Rightarrow \neg\mathsf{stuck}_\alpha$. We know that $\psi = \varphi_\beta(S)$ where $S \subseteq \mathsf{ext}_\beta$. By Lemma E.2, for each $x \in S$, either $x \in \mathsf{ext}_\alpha$ or $\varphi_\beta(x) \Rightarrow \mathsf{stuck}_\alpha$. But the latter case is not possible, otherwise we would be

able to contradict $\varphi_\beta(S) = \psi \Rightarrow \neg\mathsf{stuck}_\alpha$. Therefore, $S \subseteq \mathsf{ext}_\alpha$. Now take any T such that $S \subset T \subseteq_{\mathsf{fin}} \mathsf{ext}_\alpha$. Since $\psi \in \mathsf{br}_\beta$, we know that $\varphi_\beta(T)$ is not satisfiable and, since $\varphi_\alpha(T) = \varphi_\beta(T)$, then $\varphi_\alpha(T)$ is not satisfiable either. Therefore, $\psi \in \mathsf{br}_\alpha$. □

Lemma E.3. *For all $\alpha, \alpha', \beta, \beta' \in \mathit{lpo}_{\mathsf{fin}}(L)$ and $f \in \mathsf{copy}_{\alpha',\beta'}$ such that $\alpha \sqsubseteq_{\mathit{lpo}} \alpha'$ and $\beta \sqsubseteq_{\mathit{lpo}} \beta'$, there exists $g \in \mathsf{copy}_{\alpha,\beta}$ such that $\alpha \fatsemi_g \beta \sqsubseteq_{\mathit{lpo}} \alpha' \fatsemi_f \beta'$.*

Proof. Let $\beta'_\psi = f(\psi)$ for each $\psi \in \mathsf{br}_{\alpha'}$. Since $\beta' \equiv f(\psi)$ for all ψ, then for each $\psi \in \mathsf{br}_{\alpha'}$ there exists a bijection $f_\psi \colon N_{\beta'} \to N_{\beta'_\psi}$. Since $\beta \sqsubseteq_{\mathit{lpo}} \beta'$, then $N_\beta \subseteq N_{\beta'}$; so, we can restrict f_ψ to N_β and obtain an lpo-isomorphism g_ψ (that clearly satisfies $g_\psi(x) = f_\psi(x)$ for all $x \in N_\beta$). Now, let β_ψ be an isomorphic copy of β generated by the bijection g_ψ. Clearly, $N_{\beta_\psi} \cap N_\alpha = \emptyset$, since $N_\alpha \subseteq N_{\alpha'}$ and $N_{\beta_\psi} \subseteq N_{\beta'_\psi}$; similarly, all the β_ψs have pairwise disjoint nodes. By Lemma 8, $\mathsf{br}_\alpha \subseteq \mathsf{br}_{\alpha'}$; so, we have such a β_ψ for all $\psi \in \mathsf{br}_\alpha$, and we can define $g \in \mathsf{copy}_{\alpha,\beta}$ as $g(\psi) = \beta_\psi$ for all $\psi \in \mathsf{br}_\alpha$. Also, $\beta_\psi = g_\psi(\beta) \sqsubseteq_{\mathit{lpo}} f_\psi(\beta') = \beta'_\psi$ for all $\psi \in \mathsf{br}_\alpha$, since $\beta \sqsubseteq_{\mathit{lpo}} \beta'$ and $g_\psi \subseteq f_\psi$.

We start with proving that $N_{\alpha\fatsemi_g\beta} \subseteq_\downarrow N_{\alpha'\fatsemi_f\beta'}$. Take any $x \in N_{\alpha\fatsemi_g\beta}$; so, either $x \in N_\alpha (\subseteq N_{\alpha'})$ or $x \in N_{\beta_\psi} (\subseteq N_{\beta'_\psi})$, for some $\psi \in \mathsf{br}_\alpha$. Clearly $x \in N_{\alpha'\fatsemi_f\beta'}$, so the subset inclusion holds; we now show that it is downward closed. Take any $y <_{\alpha'\fatsemi_f\beta'} x$. If $x \in N_\alpha$, then clearly $y <_{\alpha'} x$, since $\fatsemi$ does not add any causality into elements of $N_{\alpha'}$, and we therefore conclude that $y \in N_\alpha \subseteq N_{\alpha\fatsemi_g\beta}$, since $N_\alpha \subseteq_\downarrow N_{\alpha'}$. If instead $x \in N_{\beta_\psi}$ for some $\psi \in \mathsf{br}_\alpha$, then we know by Lemma 8 that $\psi \in \mathsf{br}_{\alpha'}$ and therefore $x \in N_{\beta'_\psi} \subseteq N_{\alpha'\fatsemi_f\beta'}$. Since $y <_{\alpha'\fatsemi_f\beta'} x$, then either $y <_{\beta'_\psi} x$, or $\psi \Rightarrow \varphi_{\alpha'}(y)$. In the former case, then clearly $y \in N_{\beta_\psi} \subseteq N_{\alpha\fatsemi_g\beta}$ since $N_{\beta_\psi} \subseteq_\downarrow N_{\beta'_\psi}$. In the latter case, suppose for the sake of contradiction that $y \notin N_\alpha$; then, by Lemma 1 there is some $z \in \mathsf{Bot}_\alpha$ such that $z <_{\alpha'} y$, which also means that $\varphi_{\alpha'}(y) \Rightarrow \varphi_{\alpha'}(z)$. Since $\alpha \sqsubseteq_{\mathit{lpo}} \alpha'$ and $z \in \mathsf{Bot}_\alpha$, we have that $\psi \Rightarrow \varphi_{\alpha'}(y) \Rightarrow \varphi_{\alpha'}(z) \Leftrightarrow \varphi_\alpha(z) \Rightarrow \mathsf{stuck}_\alpha$; this contradicts $\psi \Rightarrow \neg\mathsf{stuck}_\alpha$, that holds since $\psi \in \mathsf{br}_\alpha$.

Next, we show the condition on the order:

$$
\begin{aligned}
&<_{\alpha'\fatsemi_f\beta'} \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) \\
&\quad= \left(<_{\alpha'} \cup \bigcup_{\psi\in\mathsf{br}_{\alpha'}} \left(<_{\beta'_\psi} \cup (\{x \mid \psi \Rightarrow \varphi_{\alpha'}(x)\} \times N_{\beta'_\psi}) \right) \right) \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) \\
&\quad= \left(<_{\alpha'} \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta})\right) \cup \bigcup_{\psi\in\mathsf{br}_{\alpha'}} \left(<_{\beta'_\psi} \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) \right) \cup \left((\{x \mid \psi \Rightarrow \varphi_{\alpha'}(x)\} \times N_{\beta'_\psi}) \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) \right) \qquad (\dagger)
\end{aligned}
$$

Now, N_α is a subset of $N_{\alpha'}$ and it is disjoint from all the N_{β_ψ}; moreover, $\alpha \sqsubseteq_{\mathit{lpo}} \alpha'$. So,

$$<_{\alpha'} \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) = <_{\alpha'} \cap (N_\alpha \times N_\alpha) = <_\alpha$$

Then, by Lemma 8 $\mathsf{br}_\alpha \subseteq \mathsf{br}_{\alpha'}$, and, for all $\psi \in \mathsf{br}_{\alpha'} \setminus \mathsf{br}_\alpha$, the nodes of β'_ψ do not appear in $N_{\alpha\fatsemi_g\beta}$; hence, we can reduce the bounds of the union in (†) and obtain:

$$
\begin{aligned}
&\bigcup_{\psi\in\mathsf{br}_{\alpha'}} \left(<_{\beta'_\psi} \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) \right) \cup \left((\{x \mid \psi \Rightarrow \varphi_{\alpha'}(x)\} \times N_{\beta'_\psi}) \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) \right) \\
&\quad= \bigcup_{\psi\in\mathsf{br}_{\alpha}} \left(<_{\beta'_\psi} \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) \right) \cup \left((\{x \mid \psi \Rightarrow \varphi_{\alpha'}(x)\} \times N_{\beta'_\psi}) \cap (N_{\alpha\fatsemi_g\beta} \times N_{\alpha\fatsemi_g\beta}) \right) \\
&\quad= \bigcup_{\psi\in\mathsf{br}_{\alpha}} \left(<_{\beta'_\psi} \cap (N_{\beta_\psi} \times N_{\beta_\psi}) \right) \cup (\{x \mid \psi \Rightarrow \varphi_{\alpha'}(x)\} \times N_{\beta_\psi}) \\
&\quad= \bigcup_{\psi\in\mathsf{br}_{\alpha}} \left(<_{\beta_\psi} \cup (\{x \mid \psi \Rightarrow \varphi_{\alpha}(x)\} \times N_{\beta_\psi}) \right)
\end{aligned}
$$

where the last two equalities hold because $\beta_\psi \sqsubseteq_{\mathit{lpo}} \beta'_\psi$ and because the nodes of β'_ψ are disjoint from those of α and of any other isomorphic copy of β. Thus, (†) is equal to $<_\alpha \cup \bigcup_{\psi\in\mathsf{br}_\alpha} \left(<_{\beta_\psi} \cup (\{x \mid \psi \Rightarrow \varphi_\alpha(x)\} \times N_{\beta_\psi}) \right)$ that, by definition, is $<_{\alpha\fatsemi\beta}$.

Now, take any $x \in N_{\alpha\fatsemi_g\beta}$. Since $\alpha \sqsubseteq_{\mathit{lpo}} \alpha'$ and $\beta_\psi \sqsubseteq_{\mathit{lpo}} \beta'_\psi$, we have that $\lambda_\alpha(x) \leq \lambda_{\alpha'}(x)$, if $x \in N_\alpha$, and $\lambda_{\beta_\psi}(x) \leq \lambda_{\beta'_\psi}(x)$, if $x \in N_{\beta_\psi}$. This gives us:

$$\lambda_{\alpha\fatsemi_g\beta}(x) = \left\{ \begin{array}{ll} \lambda_\alpha(x) & \text{if } x \in N_\alpha \\ \lambda_{\beta_\psi}(x) & \text{if } x \in N_{\beta_\psi} \end{array} \right\} \leq \left\{ \begin{array}{ll} \lambda_{\alpha'}(x) & \text{if } x \in N_\alpha \\ \lambda_{\beta'_\psi}(x) & \text{if } x \in N_{\beta_\psi} \end{array} \right\} = \lambda_{\alpha'\fatsemi_f\beta'}(x)$$

Similarly, we have that $\varphi_\alpha(x) = \varphi_{\alpha'}(x)$, if $x \in N_\alpha$, and $\varphi_{\beta_\psi}(x) = \varphi_{\beta'_\psi}(x)$, if $x \in N_{\beta_p si}$; so

$$\varphi_{\alpha\fatsemi_g\beta}(x) = \left\{ \begin{array}{ll} \varphi_\alpha(x) & \text{if } x \in N_\alpha \\ \varphi_{\beta_\psi}(x) \wedge \psi & \text{if } x \in N_{\beta_\psi} \end{array} \right\} = \left\{ \begin{array}{ll} \varphi_{\alpha'}(x) & \text{if } x \in N_\alpha \\ \varphi_{\beta'_\psi}(x) \wedge \psi & \text{if } x \in N_{\beta_\psi} \end{array} \right\} = \varphi_{\alpha'\fatsemi_f\beta'}(x)$$

For successors, we know by Lemma B.2(3) that $\mathsf{succ}_{\alpha \fatsemi_g \beta}(x) \subseteq \mathsf{succ}_{\alpha' \fatsemi_f \beta'}(x) \setminus \mathsf{Bot}_{\alpha \fatsemi_g \beta}{\uparrow}_{\alpha' \fatsemi_f \beta'}$, we now must show the reverse inclusion. Take any $y \in \mathsf{succ}_{\alpha' \fatsemi_f \beta'}(x)$ such that $y \notin \mathsf{Bot}_{\alpha \fatsemi_g \beta}{\uparrow}_{\alpha' \fatsemi_f \beta'}$. Given the way that $<_{\alpha' \fatsemi_f \beta'}$ is constructed, there are three cases to consider:

1) $y \in \mathsf{succ}_{\alpha'}(x)$. Since $\alpha \sqsubseteq_{lpo} \alpha'$, we know that $\mathsf{succ}_{\alpha}(x) = \mathsf{succ}_{\alpha'}(x) \setminus \mathsf{Bot}_{\alpha}{\uparrow}_{\alpha'}$. However, $y \notin \mathsf{Bot}_{\alpha}{\uparrow}_{\alpha'}$ (since $\mathsf{Bot}_{\alpha}{\uparrow}_{\alpha'} \subseteq \mathsf{Bot}_{\alpha \fatsemi_g \beta}{\uparrow}_{\alpha' \fatsemi_f \beta'}$); therefore, it must be that $y \in \mathsf{succ}_{\alpha}(x)$, and so $y \in \mathsf{succ}_{\alpha \fatsemi_g \beta}(x)$.

2) $y \in \mathsf{succ}_{\beta'_\psi}(x)$ for some $\psi \in \mathsf{br}_{\alpha}$. This means that $x, y \in N_{\beta'_\psi}$. Let $x' = f_\psi^{-1}(x)$ and $y' = f_\psi^{-1}(y)$, so $x', y' \in N_{\beta'}$. Since $\beta' \equiv \beta'_\psi$, we know that $y' \in \mathsf{succ}_{\beta'}(x')$. Since $\beta \sqsubseteq_{lpo} \beta'$, then $\mathsf{succ}_{\beta}(x') = \mathsf{succ}_{\beta'}(x') \setminus \mathsf{Bot}_{\beta}{\uparrow}_{\beta'}$. However, since $y \notin \mathsf{Bot}_{\alpha \fatsemi_g \beta}{\uparrow}_{\alpha' \fatsemi_g \beta'}$, then it must be that $y' \notin \mathsf{Bot}_{\beta}{\uparrow}_{\beta'}$; so we must have that $y' \in \mathsf{succ}_{\beta}(x')$. Since $x', y' \in N_{\beta}$, then $g_\psi(x') = f_\psi(x') = x$ and $g_\psi(y') = f_\psi(y') = y$; so, $y \in \mathsf{succ}_{\beta_\psi}(x)$ and therefore $y \in \mathsf{succ}_{\alpha \fatsemi_g \beta}(x)$.

3) $x \in N_{\alpha}$ and $y \in N_{\beta'_\psi}$ for some $\psi \in \mathsf{br}_{\alpha'}$. By Lemma 8, either $\psi \in \mathsf{br}_{\alpha}$ or $\psi \not\Rightarrow \neg\mathsf{stuck}_{\alpha}$.
For the first case, $x <_{\alpha' \fatsemi_f \beta'} y$, $x \in N_{\alpha'}$ and $y \in N_{\beta'_\psi}$ imply that $\psi \Rightarrow \varphi_{\alpha'}(x) = \varphi_{\alpha}(x)$. Furthermore, y must be the root of β'_ψ, otherwise it could not be a successor of $x \in N_{\alpha'}$. Since β'_ψ is single-rooted and $\beta_\psi \sqsubseteq_{lpo} \beta'_\psi$, then y must also be the root of β_ψ; so, $x <_{\alpha \fatsemi_g \beta} y$ and, by Lemma B.2(1), $y \in \mathsf{succ}_{\alpha \fatsemi_g \beta}(x)$.
To conclude, we show that the second case is impossible. Suppose that $\psi \not\Rightarrow \neg\mathsf{stuck}_{\alpha}$. From $\psi \in \mathsf{br}_{\alpha'}$ we also know that $\psi \Rightarrow \neg\mathsf{stuck}_{\alpha'}$. Therefore, there is some v such that $v \vDash \psi$, $v \vDash \mathsf{stuck}_{\alpha}$ and $v \nvDash \mathsf{stuck}_{\alpha'}$. From $v \vDash \mathsf{stuck}_{\alpha}$, we get that $v \vDash \varphi_{\alpha}(z)$, for some $z \in \mathsf{Bot}_{\alpha}$; so $v \vDash \varphi_{\alpha'}(z)$ too (since $\varphi_{\alpha}(z) = \varphi_{\alpha'}(z)$). Since $v \vDash \varphi_{\alpha'}(z)$ and $v \nvDash \mathsf{stuck}_{\alpha'}$, then $\varphi_{\alpha'}(z) \not\Rightarrow \mathsf{stuck}_{\alpha'}$; therefore $z \in \mathsf{ext}_{\alpha'}$. Moreover, since $v \vDash \psi$ and $v \vDash \varphi_{\alpha'}(z)$, we know that $\psi \wedge \varphi_{\alpha'}(z)$ is satisfiable; since S is maximal, then $z \in S$. Therefore, $\psi \Rightarrow \varphi_{\alpha'}(z)$, and so $z <_{\alpha' \fatsemi_f \beta'} y$ (by the definition of $<_{\alpha' \fatsemi_f \beta'}$), in contradiction with $y \notin \mathsf{Bot}_{\alpha \fatsemi_g \beta}{\uparrow}_{\alpha' \fatsemi_f \beta'}$. □

Lemma 9 (Monotonicity of $\fatsemi$)**.** *If $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\alpha'}$ and $\boldsymbol{\beta} \sqsubseteq_{pom} \boldsymbol{\beta'}$, then $\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \sqsubseteq_{pom} \boldsymbol{\alpha'} \fatsemi \boldsymbol{\beta'}$*

Proof. Take any $\gamma \in \boldsymbol{\alpha'} \fatsemi \boldsymbol{\beta'}$; by construction, $\gamma = \alpha' \fatsemi_f \beta'$, for some $\alpha' \in \boldsymbol{\alpha'}$, $\beta' \in \boldsymbol{\beta'}$, and $f \in \mathsf{copy}_{\alpha', \beta'}$. By Lemma C.1(3), there exist $\alpha \in \boldsymbol{\alpha}$ and $\beta \in \boldsymbol{\beta}$ such that $\alpha \sqsubseteq_{lpo} \alpha'$ and $\beta \sqsubseteq_{lpo} \beta'$. So, by Lemma E.3 there exists a $g \in \mathsf{copy}_{\alpha, \beta}$ such that $\alpha \fatsemi_g \beta \sqsubseteq_{lpo} \alpha' \fatsemi_f \beta'$. So, by Lemma C.1(3) $\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \sqsubseteq_{pom} \boldsymbol{\alpha'} \fatsemi \boldsymbol{\beta'}$. □

B. Guarded Branching

Lemma E.4. *For every $\ell \in L$, $x \in$ nodes and $\alpha, \alpha', \beta, \beta' \in \mathit{lpo}_{\mathsf{fin}}(L)$ such that $\alpha \sqsubseteq_{lpo} \alpha'$, $\beta \sqsubseteq_{lpo} \beta'$, $N_{\alpha'} \cap N_{\beta'} = \emptyset$ and $x \notin N_{\alpha'} \cup N_{\beta'}$, it holds that $\mathsf{guard}(x, \ell, \alpha, \beta) \sqsubseteq_{lpo} \mathsf{guard}(x, \ell, \alpha', \beta')$.*

Proof. To lighten notations, let us fix ℓ, $\alpha, \alpha', \beta, \beta'$ and x, and denote $\gamma \triangleq \mathsf{guard}(x, \ell, \alpha, \beta)$ and $\gamma' \triangleq \mathsf{guard}(x, \ell, \alpha', \beta')$.

First, being $N_{\alpha} \subseteq N_{\beta}$ and $N_{\alpha'} \subseteq N_{\beta'}$, we also have that $N_{\gamma} = \{x\} \cup N_{\alpha} \cup N_{\beta} \subseteq \{x\} \cup N_{\alpha'} \cup N_{\beta'} = N_{\gamma'}$; we have to prove that N_{γ} is downward closed. Let $y \in N_{\gamma}$ and $z <_{\gamma'} y$; clearly, $y \neq x$, because $\mathsf{pred}_{\gamma'}(x) = \emptyset$. Assume that $y \in N_{\alpha}$ ($\subseteq N_{\alpha'}$); the case for $y \in N_{\beta}$ is identical. If $z = x$, we trivially conclude. If $z \in N_{\alpha'}$, we have $z <_{\alpha'} y$, which leads to $z \in N_{\alpha}$ (because $N_{\alpha} \subseteq_{\downarrow} N_{\alpha'}$); hence, $z \in N_{\gamma}$.

Second,

$$
\begin{aligned}
&<_{\gamma'} \cap (N_{\gamma} \times N_{\gamma}) \\
&\quad = \quad (<_{\alpha'} \cup <_{\beta'} \cup (\{x\} \times (N_{\alpha'} \cup N_{\beta'}))) \ \cap \ (N_{\gamma} \times N_{\gamma}) \\
&\quad = \quad (<_{\alpha'} \cap ((\{x\} \cup N_{\alpha} \cup N_{\beta}) \times (\{x\} \cup N_{\alpha} \cup N_{\beta}))) \\
&\qquad \cup (<_{\beta'} \cap ((\{x\} \cup N_{\alpha} \cup N_{\beta}) \times (\{x\} \cup N_{\alpha} \cup N_{\beta}))) \\
&\qquad \cup ((\{x\} \times (N_{\alpha'} \cup N_{\beta'})) \cap ((\{x\} \cup N_{\alpha} \cup N_{\beta}) \times (\{x\} \cup N_{\alpha} \cup N_{\beta}))) \\
&\quad = \quad (<_{\alpha'} \cap (N_{\alpha} \times N_{\alpha})) \ \cup \ (<_{\beta'} \cap (N_{\beta} \times N_{\beta})) \ \cup \ (\{x\} \times (N_{\alpha} \cup N_{\beta})) \\
&\quad = \quad <_{\alpha} \cup <_{\beta} \cup (\{x\} \times (N_{\alpha} \cup N_{\beta})) \quad = \quad <_{\gamma}
\end{aligned}
$$

Fix $y \in N_{\gamma}$; we have three subcases to consider:

1) $y = x$: by construction, $\lambda_{\gamma}(y) = \lambda_{\gamma'}(y) = \ell$ and $\varphi_{\gamma}(y) = \varphi_{\gamma'} = \mathsf{true}$.
2) $y \in N_{\alpha}$: then, $\lambda_{\gamma}(y) = \lambda_{\alpha}(y) \leq \lambda_{\alpha'}(y) = \lambda_{\gamma'}(y)$ and $\varphi_{\gamma}(y) = \varphi_{\alpha}(y) \wedge x = \varphi_{\alpha'}(y) \wedge x = \lambda_{\gamma'}(y)$.
3) $y \in N_{\beta}$: this case is similar to previous one, with $\neg x$ in place of x for the φ case.

To conclude, we need to prove that $\mathsf{succ}_{\gamma}(y) \supseteq \mathsf{succ}_{\gamma'}(y) \setminus \mathsf{Bot}_{\gamma}{\uparrow}_{\gamma'}$ (since the other inclusion holds for Lemma B.2). Thus, take $z \in \mathsf{succ}_{\gamma'}(y)$; we have three cases to consider:

1) $y = x$: by construction, $z \in \{\min_{\alpha'}, \min_{\beta'}\}$ and so we immediately conclude $z \in \mathsf{succ}_{\gamma}(y)$, being $\min_{\alpha} = \min_{\alpha'}$ and $\min_{\beta} = \min_{\beta'}$.
2) $y \in N_{\alpha}$: by construction, $z \in \mathsf{succ}_{\alpha'}(y)$. Since $\alpha \sqsubseteq_{lpo} \alpha'$, we know that $\mathsf{succ}_{\alpha}(y) = \mathsf{succ}_{\alpha'}(y) \setminus \mathsf{Bot}_{\alpha}{\uparrow}_{\alpha'}$. So, either $z \in \mathsf{succ}_{\alpha}(y) = \mathsf{succ}_{\gamma}(y)$ or $z \in \mathsf{Bot}_{\alpha}{\uparrow}_{\alpha'} \subseteq \mathsf{Bot}_{\gamma}{\uparrow}_{\gamma'}$.

3) $y \in N_\beta$: this case is similar to previous one. □

Lemma 7 (Monotonicity of guard)**.** *If* $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\alpha}'$ *and* $\boldsymbol{\beta} \sqsubseteq_{pom} \boldsymbol{\beta}'$*, then* $\mathsf{guard}(\ell, \boldsymbol{\alpha}, \boldsymbol{\beta}) \sqsubseteq_{pom} \mathsf{guard}(\ell, \boldsymbol{\alpha}', \boldsymbol{\beta}')$.

Proof. Take any $\gamma \in \mathsf{guard}(\ell, \boldsymbol{\alpha}', \boldsymbol{\beta}')$; by construction, $\gamma = \mathsf{guard}(x, \ell, \alpha', \beta')$, for some $\alpha' \in \boldsymbol{\alpha}'$ and $\beta' \in \boldsymbol{\beta}'$ such that $N_{\alpha'} \cap N_{\beta'} = \emptyset$ and $x \notin N_{\alpha'} \cup N_{\beta'}$. By Lemma C.1(3), there exist $\alpha \in \boldsymbol{\alpha}$ and $\beta \in \boldsymbol{\beta}$ such that $\alpha \sqsubseteq_{lpo} \alpha'$ and $\beta \sqsubseteq_{lpo} \beta'$. So, by Lemma E.4 $\mathsf{guard}(x, \ell, \alpha, \beta) \sqsubseteq_{lpo} \mathsf{guard}(x, \ell, \alpha', \beta')$. So, by Lemma C.1(3) $\mathsf{guard}(\ell, \boldsymbol{\alpha}, \boldsymbol{\beta}) \sqsubseteq_{pom} \mathsf{guard}(\ell, \boldsymbol{\alpha}', \boldsymbol{\beta}')$. □

APPENDIX F
DENOTATIONAL SEMANTICS

A. Linearization

Lemma F.1. *If* $\alpha \sqsubseteq_{lpo} \beta$ *and* $S \cap \mathsf{Bot}_\alpha = \emptyset$*, then* $\mathsf{next}(\alpha, \psi, S) = \mathsf{next}(\beta, \psi, S)$.

Proof. We show the set inclusion in both directions. Take any $y \in \mathsf{next}(\alpha, \psi, S)$, so $y \in N_\alpha$, $y \notin S$, $y{\downarrow_\alpha} \subseteq S$, and $\psi \Rightarrow \varphi_\alpha(y)$. Since $\alpha \sqsubseteq_{lpo} \beta$, we get that $y \in N_\beta$, $y{\downarrow_\alpha} = y{\downarrow_\beta}$, and $\varphi_\alpha(y) = \varphi_\beta(y)$, so $y \in \mathsf{next}(\beta, \psi, S)$.

For the reverse inclusion, take any $y \in \mathsf{next}(\beta, \psi, S)$, so $y \in N_\beta$, $y \notin S$, $y{\downarrow_\beta} \subseteq S$, and $\psi \Rightarrow \varphi_\beta(y)$. If $y \in N_\alpha$, then the proof follows from a similar argument as above. If not, we will show that there is a contradiction. Suppose that $y \notin N_\alpha$, so by Lemma 1, we get that $y \in \mathsf{Bot}_\alpha{\uparrow_\beta}$. So there is a $z \in \mathsf{Bot}_\alpha$ such that $z <_\beta y$. Since $y{\downarrow_\beta} \subseteq S$, then $z \in S$. However, this is a contradiction since we assumed that $S \cap \mathsf{Bot}_\alpha = \emptyset$. □

Lemma F.2 (Monotonicity of $\mathcal{L}_{lpo}$)**.** *For any* $\alpha, \beta \in \mathit{lpo}_{\mathsf{fin}}(L)$*,* $\psi \in \mathit{form}$*,* $S \subseteq N_\alpha$*, and* $s \in \mathcal{S}$ *such that* $\alpha \sqsubseteq_{lpo} \beta$ *and* $S \cap \mathsf{Bot}_\alpha = \emptyset$*:*

$$\mathcal{L}_{lpo}(\alpha, \psi, S) \sqsubseteq^\bullet \mathcal{L}_{lpo}(\beta, \psi, S)$$

Proof. Let $T_{S,\psi} = \{x \in N_\alpha \mid x \notin S, \mathsf{sat}(\psi \wedge \varphi_\alpha(x))\}$ be the set of nodes still to schedule. The proof is by induction on the size of the set $T_{S,\psi}$.

If $T_{S,\psi}$ is empty, then $x \in S$ or $\psi \wedge \varphi_\alpha(x)$ is unsatisfiable for all $x \in N_\alpha$. In either case, clearly $x \notin \mathsf{next}(\alpha, \psi, S)$, therefore $\mathsf{next}(\alpha, \psi, S) = \emptyset$. Moreover, by Lemma F.1, $\mathsf{next}(\beta, \psi, S) = \emptyset$ too and so $\mathcal{L}_{lpo}(\alpha, \psi, S)(s) = \mathcal{L}_{lpo}(\beta, \psi, S)(s) = \eta(s)$.

Now suppose the claim holds for all sets smaller than $T_{S,\psi}$, and let f be defined as follows:

$$f(\gamma, \psi, S, x)(s) = \begin{cases} \mathcal{L}_{lpo}(\gamma, \psi, S \cup \{x\})^\dagger([\![\lambda_\gamma(x)]\!]_{act}(s)) & \text{if } \lambda_\gamma(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\gamma, \psi \wedge (\!|x = [\![\lambda_\gamma(x)]\!](s)|\!), S \cup \{x\})(s) & \text{if } \lambda_\gamma(x) \in \mathit{test} \\ \bot & \text{if } \lambda_\gamma(x) = \bot \\ \mathcal{L}_{lpo}(\gamma, \psi, S \cup \{x\})(s) & \text{if } \lambda_\gamma(x) = \mathsf{fork} \end{cases}$$

So, $\mathcal{L}_{lpo}(\gamma, \psi, S)(s) = \bigotimes_{x \in \mathsf{next}(\gamma, \psi, S)} f(\gamma, \psi, S, x)(s)$. We start by showing that $f(\alpha, \psi, S, x)(s) \sqsubseteq f(\beta, \psi, S, x)(s)$. First, we remark that $S \cap \mathsf{Bot}_\alpha = \emptyset$ and so, if $x \notin \mathsf{Bot}_\alpha$, then $(S \cup \{x\}) \cap \mathsf{Bot}_\alpha = \emptyset$, which will allow us to use the induction hypothesis in cases (1), (2), and (4) below. We proceed by case analysis on $\lambda_\alpha(x)$.

1) If $\lambda_\alpha(x) \in \mathit{act}$, then $\lambda_\beta(x) \in \mathit{act}$ too, and $[\![\lambda_\alpha(x)]\!]_{act}(s) \sqsubseteq [\![\lambda_\beta(x)]\!]_{act}(s)$. Note that $T_{S \cup \{x\},\psi} \subset T_{S,\psi}$, so by the induction hypothesis, we can conclude that:

$$\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\}) \sqsubseteq^\bullet \mathcal{L}_{lpo}(\beta, \psi, S \cup \{x\})$$

Therefore, by monotonicity of Kleisli extension, we get that:

$$f(\alpha, \psi, S, x)(s) \;=\; \mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s)) \;\sqsubseteq\; \mathcal{L}_{lpo}(\beta, \psi, S \cup \{x\})^\dagger([\![\lambda_\beta(x)]\!]_{act}(s)) \;=\; f(\beta, \psi, S, x)(s)$$

2) If $\lambda_\alpha(x) \in \mathit{test}$, then again $\lambda_\beta(x) \in \mathit{test}$ as well and $\lambda_\alpha(x) = \lambda_\beta(x)$ since the order over tests is flat. Since $T_{S \cup \{x\}, \psi \wedge (\!|x = [\![b]\!]_{test}(s)|\!)} \subset T_{S,\psi}$, then we can use the induction hypothesis to conclude that:

$$\begin{aligned} f(\alpha, \psi, S, x)(s) &= \mathcal{L}_{lpo}(\alpha, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!]_{test}(s)|\!), S \cup \{x\})(s) \\ &\sqsubseteq \mathcal{L}_{lpo}(\beta, \psi \wedge (\!|x = [\![\lambda_\beta(x)]\!]_{test}(s)|\!), S \cup \{x\})(s) \\ &= f(\beta, \psi, S, x)(s) \end{aligned}$$

3) If $\lambda_\alpha(x) = \bot$, then we have:

$$f(\alpha, \psi, S, x)(s) \quad = \quad \bot \quad \sqsubseteq \quad f(\beta, \psi, S, x)(s)$$

4) If $\lambda_\alpha(x) = \mathsf{fork}$, then $\lambda_\beta(x) = \mathsf{fork}$, and so since $T_{S \cup \{x\},\psi} \subset T_{S,\psi}$, the induction hypothesis gives us the following:

$$f(\alpha, \psi, S, x)(x) \quad = \quad \mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})(x) \quad \sqsubseteq \quad \mathcal{L}_{lpo}(\beta, \psi, S \cup \{x\})(x) \quad = \quad f(\beta, \psi, S, x)(x)$$

We now complete the proof of the main claim as follows:

$$\mathcal{L}_{lpo}(\alpha, \psi, S)(s) = \bigwith_{x \in \mathsf{next}(\alpha,\psi,S)} f(\alpha, \psi, S, x)(s)$$

Since $S \cap \mathsf{Bot}_\alpha = \emptyset$, by Lemma F.1 we know that $\mathsf{next}(\alpha, \psi, S) = \mathsf{next}(\beta, \psi, S)$.

$$= \bigwith_{x \in \mathsf{next}(\beta,\psi,S)} f(\alpha, \psi, S, x)(s)$$

Since $\&$ is monotone, and given the claim we just proved about f:

$$\sqsubseteq \bigwith_{x \in \mathsf{next}(\beta,\psi,S)} f(\beta, \psi, S, x)(s)$$
$$= \mathcal{L}_{lpo}(\beta, \psi, S)(s) \qquad \square$$

Lemma 10 (Monotonicity of $\mathcal{L}_{\mathsf{fin}}$)**.** *If $\boldsymbol{\alpha} \sqsubseteq_{lpo} \boldsymbol{\beta}$, then $\mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha}) \sqsubseteq^\bullet \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\beta})$.*

Proof. Fix any $\alpha \in \boldsymbol{\alpha}$, then we know there exists a $\beta \in \boldsymbol{\beta}$ such that $\alpha \sqsubseteq_{lpo} \beta$:

$$\begin{aligned} \mathcal{L}_{\mathsf{fin}}([\alpha]) &= \mathcal{L}_{lpo}(\alpha, \mathsf{true}, \emptyset) \\ &\sqsubseteq^\bullet \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset) && \text{By Lemma F.2} \\ &= \mathcal{L}_{\mathsf{fin}}([\beta]) \end{aligned}$$ $\square$

B. Properties of Linearization: Skip and Actions

Lemma 11. *$\mathcal{L}([\![\texttt{skip}]\!]) = \eta$ and $\mathcal{L}([\![a]\!]) = [\![a]\!]_{act}$.*

Proof. We show only the case for **skip**, as the case for actions is nearly identical.

$$\mathcal{L}([\![\texttt{skip}]\!])(s) = \mathcal{L}(\langle \mathsf{fork} \rangle)(s)$$

Since a singleton pomset is already finite, $\mathcal{L} = \mathcal{L}_{\mathsf{fin}}$. Fixing an arbitrary $x \in \mathit{nodes}$, we have:

$$= \mathcal{L}_{\mathsf{fin}}([\langle \mathsf{fork} \rangle_x])(s)$$
$$= \mathcal{L}_{lpo}(\langle \mathsf{fork} \rangle_x, \mathsf{true}, \emptyset)(s)$$

Obviously, $\mathsf{next}(\langle \mathsf{fork} \rangle_x, \mathsf{true}, \emptyset) = \{x\}$, so we get:

$$= \mathcal{L}_{lpo}(\langle \mathsf{fork} \rangle_x, \mathsf{true}, \{x\})(s)$$
$$= \eta(s) \qquad \square$$

C. Properties of Linearization: Sequential Composition

We first introduce a few definitions:

$$\begin{aligned} \mathsf{next}^*(\alpha, \psi, S) &= \{x \in N_\alpha \setminus S \mid \mathsf{sat}(\psi \wedge \varphi_\alpha(x))\} \\ \mathsf{tests}_\alpha(S) &= \{x \in S \mid \lambda_\alpha(x) \in \mathit{test}\} \\ \mathsf{forms}(X) &= \left\{ \bigwedge_{x \in X} (\!|x = b_x|\!) \;\middle|\; \forall x \in X.\ b_x \in \mathbb{B} \right\} \\ \mathsf{valid}_\alpha(S) &= \{\psi \mid \psi \in \mathsf{forms}(\mathsf{tests}_\alpha(S)), \forall x \in S.\ \psi \Rightarrow \varphi_\alpha(x)\} \end{aligned}$$

The set $\mathsf{next}^*(\alpha, \psi, S)$ contains all of the nodes in α that are still able to be scheduled, given that S has already been processed and ψ is the path condition. The set $\mathsf{tests}_\alpha(S)$ contains all the test nodes from S. The set $\mathsf{forms}(X)$ contains all the formulae over the variables X, which are conjunctions of literals, where a literal is either a variable x or its negation $\neg x$. Finally, $\mathsf{valid}_\alpha(S)$ contains all the formulae over $\mathsf{tests}_\alpha(S)$ that imply the formulae of all $x \in S$. As an abuse of notation, we will write $\psi \in S$ to mean that there exists a $\psi' \in S$ such that $\psi \Leftrightarrow \psi'$.

Lemma F.3. *If α is binary branching, $S \subseteq \mathsf{nBot}_\alpha$, $\mathsf{next}^*(\alpha, \psi, S) = \emptyset$ and $\psi \in \mathsf{valid}_\alpha(S)$, then $\psi \in \mathsf{br}_\alpha$ and $\mathsf{next}(\alpha \fatsemi_f \beta) = \min_{f(\psi)}$.*

Proof. Since $\mathsf{next}^*(\alpha, \psi, S) = \emptyset$, we know that for all $x \in N_\alpha$, either $x \in S$ or $\psi \wedge \varphi_\alpha(x)$ is unsatisfiable. In particular, since $S \subseteq \mathsf{nBot}_\alpha$, then $\psi \wedge \varphi_\alpha(x)$ is unsatisfiable for all $x \in \mathsf{Bot}_\alpha$. We will now establish that $\psi \in \mathsf{br}_\alpha$ by proving the following claims:

1) $\psi \Rightarrow \neg\mathsf{stuck}_\alpha \Leftrightarrow \neg \bigvee_{x \in \mathsf{Bot}_\alpha} \varphi_\alpha(x) \Leftrightarrow \bigwedge_{x \in \mathsf{Bot}_\alpha} \neg\varphi_\alpha(x)$. It will suffice to show that $\psi \Rightarrow \neg\varphi_\alpha(x)$ for all $x \in \mathsf{Bot}_\alpha$. Take any $v \vDash \psi$. We know that $\psi \wedge \varphi_\alpha(x)$ is unsatisfiable since $x \in \mathsf{Bot}_\alpha$, therefore $v \vDash \neg\varphi_\alpha(x)$, so we are done.
2) $\psi \Leftrightarrow \bigwedge_{x \in S} \varphi_\alpha(x)$. From $\psi \in \mathsf{valid}_\alpha(S)$, we immediately get that $\psi \Rightarrow \bigwedge_{x \in S} \varphi_\alpha(x)$. We now also show the reverse. Since $\psi \in \mathsf{valid}_\alpha(S)$, then $\psi \in \mathsf{forms}(\mathsf{tests}_\alpha(S))$. We now show that for every $y \in \mathsf{tests}_\alpha(S)$, either $\bigwedge_{x \in S} \varphi_\alpha(x) \Rightarrow y$ or $\bigwedge_{x \in S} \varphi_\alpha(x) \Rightarrow \neg y$. Take any $y \in \mathsf{tests}_\alpha(S)$. By the binary branching property, $\mathsf{succ}_\alpha(y) = \{z_1, z_2\}$ such that $\varphi_\alpha(z_1) \Leftrightarrow \varphi_\alpha(y) \wedge y$ and $\varphi_\alpha(z_2) \Leftrightarrow \varphi_\alpha(y) \wedge \neg y$. We also know that since $z_1, z_2 \in N_\alpha$, then for each $i \in \{1, 2\}$ either $z_i \in S$ or $\psi \wedge \varphi_\alpha(z_i)$ is unsatisfiable. We already know that $\psi \Rightarrow \varphi_\alpha(y)$ since $y \in S$. We also know that $\psi \Rightarrow y$ or $\psi \Rightarrow \neg y$ since $y \in \mathsf{tests}_\alpha(S)$ and $\psi \in \mathsf{valid}_\alpha(S)$. Without loss of generality, suppose that $\psi \Rightarrow y$, then clearly $\psi \wedge \varphi_\alpha(z_1) \Leftrightarrow \psi \wedge \varphi_\alpha(y) \wedge y$ is satisfiable, therefore $z_1 \in S$. This clearly means that $\bigwedge_{x \in S} \varphi_\alpha(x) \Rightarrow \varphi_\alpha(z_1) \Rightarrow y$. If instead we had that $\psi \Rightarrow \neg y$, then we would have $\bigwedge_{x \in S} \varphi_\alpha(x) \Rightarrow \neg y$.
   Therefore we have shown that $\bigwedge_{x \in S} \varphi_\alpha(x)$ assigns a truth value to each $y \in \mathsf{vars}(\psi)$, and so $\bigwedge_{x \in S} \varphi_\alpha(x) \Rightarrow \psi$.
3) We now show that $S \subseteq \mathsf{ext}_\alpha$, that is $\varphi_\alpha(x) \not\Rightarrow \mathsf{stuck}_\alpha$ for all $x \in S$. For the sake of contradiction, suppose that $\varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha$ for some $x \in S$. Since $\psi \in \mathsf{valid}_\alpha(S)$, we also have that $\psi \Rightarrow \varphi_\alpha(x)$, which gives us:
$$\psi \Rightarrow \varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha \Leftrightarrow \bigvee_{y \in \mathsf{Bot}_\alpha} \varphi_\alpha(y)$$
   But this contradicts the fact that $\psi \wedge \varphi_\alpha(y)$ is unsatisfiable for all $y \in \mathsf{Bot}_\alpha$, therefore it cannot be true that $\varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha$, and instead we have $\varphi_\alpha(x) \not\Rightarrow \mathsf{stuck}_\alpha$.
4) $\psi \wedge \varphi_\alpha(x)$ is unsatisfiable for any $x \in \mathsf{ext}_\alpha \setminus S$. This is immediate, since $\psi \wedge \varphi_\alpha(x)$ is unsatisfiable for any $x \notin S$.

So, we have now shown that $\psi \in \mathsf{br}_\alpha$. Next, we show that $\mathsf{next}(\alpha \fatsemi_f \beta, \psi, S) = \min_{f(\psi)}$ by showing the set inclusion in both directions.

Take any $x \in \mathsf{next}(\alpha \fatsemi_f \beta, \psi, S)$, so $x \in N_{\alpha \fatsemi_f \beta}$, $x \notin S$, $x{\downarrow_{\alpha \fatsemi_f \beta}} \subseteq S$, and $\psi \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(x)$. Clearly $x \notin N_\alpha$, otherwise one of $x \notin S$ or $\psi \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(x) = \varphi_\alpha(x)$ would be violated. Now, take any $y <_{\alpha \fatsemi_f \beta} x$, we know from $x{\downarrow_{\alpha \fatsemi_f \beta}} \subseteq S$ that $y \in S \subseteq N_\alpha$. So, since $\psi \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(x)$, then $x \in N_{f(\psi)}$. Furthermore, $\{x\} = \min_{f(\psi)}$ since we already established that all predecessors of x are in α, and so x has no predecessors in $\beta_{f(\psi)}$.

We now show the reverse direction. Take any $\{x\} = \min_{f(\psi)}$. Clearly $x \in N_{\alpha \fatsemi_f \beta}$ and $x \notin S$. By construction, $\varphi_{\alpha \fatsemi_f \beta}(x) = \varphi_{f(\psi)}(x) \wedge \psi \Leftrightarrow \mathsf{true} \wedge \psi$ ($\varphi_{f(\psi)}(x) \Leftrightarrow \mathsf{true}$ since x is minimal and therefore no branches have occurred), so clearly $\psi \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(x)$. Now take any $y <_{\alpha \fatsemi_f \beta} x$, since $\{x\} = \min_{f(\psi)}$, then $y \in N_\alpha$, so $\psi \Rightarrow \varphi_\alpha(y)$. We already know that for all $y \in N_\alpha$ either $y \in S$, or $\psi \wedge \varphi_\alpha(y)$ is unsatisfiable, but clearly the latter case is impossible, so $y \in S$. Therefore $x{\downarrow_{\alpha \fatsemi_f \beta}} \subseteq S$, and we have shown all the conditions to guarantee that $x \in \mathsf{next}(\alpha \fatsemi_f \beta, \psi, S)$. □

Lemma F.4. *If α is binary branching, $S \subseteq \mathsf{nBot}_\alpha$, $\mathsf{next}^*(\alpha, \psi, S) \neq \emptyset$, and $\psi \in \mathsf{valid}_\alpha(S)$, then* $\mathsf{next}(\alpha \fatsemi_f \beta, \psi, S) = \mathsf{next}(\alpha, \psi, S) \neq \emptyset$

Proof. Since $\mathsf{next}^*(\alpha, \psi, S) \neq \emptyset$, then there exists $x \in \mathsf{next}^*(\alpha, \psi, S)$, meaning that $x \in N_\alpha \setminus S$ and $\psi \wedge \varphi_\alpha(x)$ is satisfiable. Take a minimal such x, so that there is no $y \in \mathsf{next}^*(\alpha, \psi, S)$ such that $y <_\alpha x$.

We now show that $x{\downarrow_\alpha} \subseteq S$. Take any $y <_\alpha x$. We know that $y \notin \mathsf{next}^*(\alpha, \psi, S)$, so either $y \in S$ or $\psi \wedge \varphi_\alpha(y)$ is unsatisfiable. However, since $y <_\alpha x$, then $\varphi_\alpha(x) \Rightarrow \varphi_\alpha(y)$, so $\psi \wedge \varphi_\alpha(y)$ must be satisfiable, meaning that the only option is that $y \in S$.

Now, since α is binary branching, there are two options. The first option is that $x \in \mathsf{succ}_\alpha(y)$ where y is a test node and $\varphi_\alpha(x) \Leftrightarrow \varphi_\alpha(y) \wedge y$ or $\varphi_\alpha(x) \Leftrightarrow \varphi_\alpha(y) \wedge \neg y$. Since $\psi \in \mathsf{valid}_\alpha(S)$ and $\psi \wedge \varphi_\alpha(x)$ is satisfiable, then it must be that $\psi \Rightarrow \varphi_\alpha(x)$. If instead x is not the successor of any test nodes, then $\varphi_\alpha(x) \Leftrightarrow \bigwedge_{y \in \mathsf{pred}_\alpha(x)} \varphi_\alpha(y)$, so again since $\psi \wedge \varphi_\alpha(x)$ is satisfiable, then it must be that $\psi \Rightarrow \varphi_\alpha(x)$. In both cases, we have that $x \in \mathsf{next}(\alpha, \psi, S)$.

Finally, we show that $\mathsf{next}(\alpha, \psi, S) = \mathsf{next}(\alpha \fatsemi_f \beta, \psi, S)$ by showing the set inclusion in both directions. Take any $y \in \mathsf{next}(\alpha, \psi, S)$, so $y \in N_\alpha \setminus S$ such that $y{\downarrow_\alpha} \subseteq S$ and $\psi \Rightarrow \varphi_\alpha(y)$. Since $N_\alpha \subseteq_\downarrow N_{\alpha \fatsemi_f \beta}$, and $\varphi_\alpha(y) = \varphi_{\alpha \fatsemi_f \beta}(y)$, this immediately gives us that $y \in N_{\alpha \fatsemi_f \beta} \setminus S$, $y{\downarrow_{\alpha \fatsemi_f \beta}} \subseteq S$, and $\psi \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(x)$ therefore $y \in \mathsf{next}(\alpha \fatsemi_f \beta, \psi, S)$.

Now, for the reverse inclusion, take any $y \in \mathsf{next}(\alpha \fatsemi_f \beta, \psi, S)$, so $y \in N_{\alpha \fatsemi_f \beta} \setminus S$ such that $y{\downarrow_{\alpha \fatsemi_f \beta}} \subseteq S$ and $\psi \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(x)$. From previously, we know that $x \in N_\alpha \setminus S$ and $\psi \wedge \varphi_\alpha(x)$ is satisfiable. Now, suppose for the sake of contradiction that $y \notin N_\alpha$, so it must be that $y \in N_{f(\psi')}$ for some $\psi' \in \mathsf{br}_\alpha$. This implies that $x <_{\alpha \fatsemi_f \beta} y$, as we will now show. First, $\psi \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(y) = \varphi_{f(\psi')}(y) \wedge \psi' \Rightarrow \psi'$, so $\psi' \wedge \varphi_\alpha(x)$ is satisfiable. If $x \notin \mathsf{ext}_\alpha$, then $\varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha$, but we know that $\psi' \Rightarrow \neg\mathsf{stuck}_\alpha$, which contradicts that $\psi' \wedge \varphi_\alpha(x)$ is satisfiable. Therefore, $x \in \mathsf{ext}_\alpha$, and since ψ' is maximal, then it must be the case that $\psi' \Rightarrow \varphi_\alpha(x)$ already, which implies that $x <_{\alpha \fatsemi_f \beta} y$. However, since $x \notin S$, then $y{\downarrow_{\alpha \fatsemi_f \beta}} \not\subseteq S$, which is a contradiction. Therefore, it must be the case that $y \in N_\alpha$. We therefore have that $y \in N_\alpha \setminus S$, $y{\downarrow_\alpha} = y{\downarrow_{\alpha \fatsemi_f \beta}} \subseteq S$, and $\psi \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(y) = \varphi_\alpha(y)$; therefore, $y \in \mathsf{next}(\alpha, \psi, S)$. □

Lemma F.5. *For any binary branching $\alpha, \beta \in \mathit{lpo}_{\mathsf{fin}}(L)$, $S \subseteq \mathsf{nBot}_\alpha$, and $\psi \in \mathsf{valid}_\alpha(S)$:*
$$\mathcal{L}_{\mathit{lpo}}(\alpha \fatsemi \beta, \psi, S) = \mathcal{L}_{\mathit{lpo}}(\beta, \mathsf{true}, \emptyset)^\dagger \circ \mathcal{L}_{\mathit{lpo}}(\alpha, \psi, S)$$

Proof. The proof is by induction on the size of the set $\mathsf{next}^*(\alpha, \psi, S)$.

In the base case, $\mathsf{next}^*(\alpha, \psi, S)$ is empty, so by Lemma F.3, we know that $\psi \in \mathsf{br}_\alpha$ and $\mathsf{next}(\alpha \fatsemi_f \beta, \psi, S) = \min_{f(\psi)}$. Since each lpo is single-rooted, then $\min_{f(\psi)} = \{x\}$ for some $x \in N_{f(\psi)}$. This gives us:

$$\begin{aligned}
\mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S)(s) &= \mathop{\&}_{x \in \mathsf{next}(\alpha \fatsemi \beta, \psi, S)} \begin{cases} \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S \cup \{x\})^\dagger([\![\lambda_{\alpha \fatsemi \beta}(x)]\!]_{act}(s)) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi \wedge (\!|x = [\![\lambda_{\alpha \fatsemi \beta}(x)]\!](s)|\!), S \cup \{x\})(s) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) \in \mathit{test} \\ \bot & \text{if } \lambda_{\alpha \fatsemi \beta}(x) = \bot \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S \cup \{x\})(s) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) = \mathsf{fork} \end{cases} \\
&= \begin{cases} \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S \cup \{x\})^\dagger([\![\lambda_{\alpha \fatsemi \beta}(x)]\!]_{act}(s)) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi \wedge (\!|x = [\![\lambda_{\alpha \fatsemi \beta}(x)]\!](s)|\!), S \cup \{x\})(s) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) \in \mathit{test} \\ \bot & \text{if } \lambda_{\alpha \fatsemi \beta}(x) = \bot \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S \cup \{x\})(s) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) = \mathsf{fork} \end{cases} \\
&= \begin{cases} \mathcal{L}_{lpo}(\beta_\psi, \mathsf{true}, \{x\})^\dagger([\![\lambda_{\beta_\psi}(x)]\!]_{act}(s)) & \text{if } \lambda_{\beta_\psi}(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\beta_\psi, (\!|x = [\![\lambda_{\beta_\psi}(x)]\!](s)|\!), \{x\})(s) & \text{if } \lambda_{\beta_\psi}(x) \in \mathit{test} \\ \bot & \text{if } \lambda_{\beta_\psi}(x) = \bot \\ \mathcal{L}_{lpo}(\beta_\psi, \mathsf{true}, \{x\})(s) & \text{if } \lambda_{\beta_\psi}(x) = \mathsf{fork} \end{cases} \\
&= \mathop{\&}_{x \in \mathsf{next}(\beta, \mathsf{true}, \emptyset)} \begin{cases} \mathcal{L}_{lpo}(\beta, \mathsf{true}, \{x\})^\dagger([\![\lambda_\beta(x)]\!]_{act}(s)) & \text{if } \lambda_\beta(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\beta, (\!|x = [\![\lambda_\beta(x)]\!](s)|\!), \{x\})(s) & \text{if } \lambda_\beta(x) \in \mathit{test} \\ \bot & \text{if } \lambda_\beta(x) = \bot \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \{x\})(s) & \text{if } \lambda_\beta(x) = \mathsf{fork} \end{cases} \\
&= \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)(s) \\
&= \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\eta(s)) \\
&= \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi, S)(s))
\end{aligned}$$

Now suppose $\mathsf{next}^*(\alpha, \psi, S) \neq \emptyset$. By Lemma F.4, we know that $\mathsf{next}(\alpha \fatsemi_f \beta, \psi, S) = \mathsf{next}(\alpha, \psi, S) \neq \emptyset$. Given this, we have:

$$\begin{aligned}
\mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S)(s) &= \mathop{\&}_{x \in \mathsf{next}(\alpha \fatsemi \beta, \psi, S)} \begin{cases} \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S \cup \{x\})^\dagger([\![\lambda_{\alpha \fatsemi \beta}(x)]\!]_{act}(s)) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi \wedge (\!|x = [\![\lambda_{\alpha \fatsemi \beta}(x)]\!](s)|\!), S \cup \{x\})(s) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) \in \mathit{test} \\ \bot & \text{if } \lambda_{\alpha \fatsemi \beta}(x) = \bot \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S \cup \{x\})(s) & \text{if } \lambda_{\alpha \fatsemi \beta}(x) = \mathsf{fork} \end{cases} \\
&= \mathop{\&}_{x \in \mathsf{next}(\alpha, \psi, S)} \begin{cases} \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!](s)|\!), S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) \in \mathit{test} \\ \bot & \text{if } \lambda_\alpha(x) = \bot \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi, S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) = \mathsf{fork} \end{cases}
\end{aligned}$$

Clearly $\mathsf{next}^*(\alpha, \psi, S \cup \{x\}) \subset \mathsf{next}^*(\alpha, \psi, S)$ and $\psi \in \mathsf{valid}_\alpha(S \cup \{x\})$, if x is not a test node and therefore $\mathsf{valid}_\alpha(S \cup \{x\}) = \mathsf{valid}_\alpha(S)$. So, by the induction hypothesis and monad laws, we get:

$$= \mathop{\&}_{x \in \mathsf{next}(\alpha, \psi, S)} \begin{cases} \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s))) & \text{if } \lambda_\alpha(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\alpha \fatsemi \beta, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!](s)|\!), S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) \in \mathit{test} \\ \bot & \text{if } \lambda_\alpha(x) = \bot \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})(s)) & \text{if } \lambda_\alpha(x) = \mathsf{fork} \end{cases}$$

We also have that $\mathsf{next}^*(\alpha, S \cup \{x\}, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!]_{\mathit{test}}(s)|\!)) \subset \mathsf{next}^*(\alpha, S, \psi)$. In addition, $\psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!]_{\mathit{test}}(s)|\!) \in \mathsf{valid}_\alpha(S \cup \{x\})$. So, we can again use the induction hypothesis to get:

$$= \mathop{\&}_{x \in \mathsf{next}(\alpha, \psi, S)} \begin{cases} \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s))) & \text{if } \lambda_\alpha(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!](s)|\!), S \cup \{x\})(s)) & \text{if } \lambda_\alpha(x) \in \mathit{test} \\ \bot & \text{if } \lambda_\alpha(x) = \bot \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})(s)) & \text{if } \lambda_\alpha(x) = \mathsf{fork} \end{cases}$$

Finally since $f^\dagger(\bot) = \bot$ for any f, we get:

$$= \mathop{\&}_{x \in \mathsf{next}(\alpha, \psi, S)} \begin{cases} \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s))) & \text{if } \lambda_\alpha(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!](s)|\!), S \cup \{x\})(s)) & \text{if } \lambda_\alpha(x) \in \mathit{test} \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\bot) & \text{if } \lambda_\alpha(x) = \bot \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger(\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})(s)) & \text{if } \lambda_\alpha(x) = \mathsf{fork} \end{cases}$$

Since $f^\dagger(X \,\&\, Y) = f^\dagger(X) \,\&\, f^\dagger(Y)$:

$$= \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger \left(\mathop{\&}_{x \in \mathsf{next}(\alpha,\psi,S)} \begin{cases} \mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in \mathit{act} \\ \mathcal{L}_{lpo}(\alpha, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!](s)|\!), S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) \in \mathit{test} \\ \bot & \text{if } \lambda_\alpha(x) = \bot \\ \mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) = \mathsf{fork} \end{cases} \right)$$

$$= \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger \left(\mathcal{L}_{lpo}(\alpha, \psi, S)(s) \right)$$

□

Lemma 12. $\mathcal{L}([\![C_1; C_2]\!]) = \mathcal{L}([\![C_2]\!])^\dagger \circ \mathcal{L}([\![C_1]\!])$.

Proof.

$$\mathcal{L}([\![C_1; C_2]\!]) = \mathcal{L}([\![C_1]\!] \fatsemi [\![C_2]\!])$$

By the definition of $\fatsemi$ for infinite pomsets:

$$= \mathcal{L}\left(\sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \right)$$

Since $\mathcal{L}$ is Scott continuous:

$$= \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \mathcal{L}(\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta})$$
$$= \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \sup_{\boldsymbol{\gamma} \ll \boldsymbol{\alpha} \fatsemi \boldsymbol{\beta}} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\gamma})$$

Since $\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta}$ is finite, then $\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \ll \boldsymbol{\alpha} \fatsemi \boldsymbol{\beta}$ and therefore since $\mathcal{L}_{\mathsf{fin}}$ is monotone, then $\sup_{\boldsymbol{\gamma} \ll \boldsymbol{\alpha} \fatsemi \boldsymbol{\beta}} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\gamma}) = \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta})$.

$$= \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta})$$
$$= \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \mathcal{L}_{\mathsf{fin}}(\{\alpha \fatsemi_f \beta \mid \alpha \in \boldsymbol{\alpha}, \beta \in \boldsymbol{\beta}, f \in \mathsf{copy}_{\alpha,\beta}\})$$

The set above is an equivalence class, so fixing an arbitrary $\alpha \in \boldsymbol{\alpha}$, $\beta \in \boldsymbol{\beta}$, and $f \in \mathsf{copy}_{\alpha,\beta}$, we can rewrite the expression as follows:

$$= \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \mathcal{L}_{\mathsf{fin}}([\alpha \fatsemi_f \beta])$$
$$= \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \mathcal{L}_{lpo}(\alpha \fatsemi_f \beta, \mathsf{true}, \emptyset)$$

By Lemma F.5:

$$= \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)^\dagger \circ \mathcal{L}_{lpo}(\alpha, \mathsf{true}, \emptyset)$$
$$= \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \mathcal{L}_{\mathsf{fin}}([\beta])^\dagger \circ \mathcal{L}_{\mathsf{fin}}([\alpha])$$

By continuity of Kleisli extension, and since $\boldsymbol{\alpha} = [\alpha]$ and $\boldsymbol{\beta} = [\beta]$:

$$= \left(\sup_{\boldsymbol{\beta} \ll [\![C_2]\!]} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\beta}) \right)^\dagger \circ \sup_{\boldsymbol{\alpha} \ll [\![C_1]\!]} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha})$$
$$= \mathcal{L}([\![C_2]\!])^\dagger \circ \mathcal{L}([\![C_1]\!])$$

□

D. Properties of Linearization: Guarded Branching

Lemma F.6. *For any $\alpha, \beta \in \mathit{lpo}_{\mathsf{fin}}(\mathit{act})$, $b \in \mathit{test}$, and $x \in \mathit{nodes} \setminus (N_\alpha \cup N_\beta)$:*

$$\mathcal{L}_{lpo}(\mathsf{guard}(x, b, \alpha, \beta), \mathsf{true}, \emptyset)(s) = \begin{cases} \mathcal{L}_{lpo}(\alpha, \mathsf{true}, \emptyset)(s) & \mathit{if}\ [\![b]\!]_{\mathit{test}}(s) = 1 \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)(s) & \mathit{if}\ [\![b]\!]_{\mathit{test}}(s) = 0 \end{cases}$$

Proof.

$$\mathcal{L}_{lpo}(\mathsf{guard}(x, b, \alpha, \beta), \mathsf{true}, \emptyset)(s)$$

$$= \bigotimes_{x \in \mathsf{next}(\alpha,\psi,S)} \begin{cases} \mathcal{L}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta), \mathsf{true}, \{x\})^\dagger(\llbracket \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) \rrbracket_{act}(s)) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) \in act \\ \mathcal{L}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta), (\!|x = \llbracket \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) \rrbracket(s)|\!), \{x\})(s) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) \in test \\ \bot & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) = \bot \\ \mathcal{L}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta), \mathsf{true}, \{x\})(s) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) = \mathsf{fork} \end{cases}$$

Since x is the root of $\mathsf{guard}(x,b,\alpha,\beta)$, then $\mathsf{next}(\alpha,\psi,S) = \{x\}$ and $\lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) \in test$.

$$= \mathcal{L}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta), (\!|x = \llbracket \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) \rrbracket(s)|\!), \{x\})(s)$$
$$= \begin{cases} \mathcal{L}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta), x, \{x\})(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 1 \\ \mathcal{L}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta), \neg x, \{x\})(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 0 \end{cases}$$

The formula x prevents all nodes in β from being scheduled and similarly $\neg x$ prevents the α nodes from being scheduled, so we can remove the guard call to obtain the following:

$$= \begin{cases} \mathcal{L}_{lpo}(\alpha, \mathsf{true}, \emptyset)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = \mathsf{true} \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = \mathsf{false} \end{cases}$$

□

Lemma 13.

$$\mathcal{L}(\llbracket \texttt{if } b \ \{C_1\} \ \texttt{else} \ \{C_2\} \rrbracket)(s) = \begin{cases} \mathcal{L}(\llbracket C_1 \rrbracket)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 1 \\ \mathcal{L}(\llbracket C_2 \rrbracket)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 0 \end{cases}$$

Proof.

$$\begin{aligned} \mathcal{L}\llbracket \texttt{if } b \ \{C_1\} \ \texttt{else} \ \{C_2\} \rrbracket(s) &= \mathcal{L}(\mathsf{guard}(b, \llbracket C_1 \rrbracket, \llbracket C_2 \rrbracket))(s) \\ &= \sup_{\boldsymbol{\alpha} \ll \llbracket C_1 \rrbracket} \sup_{\boldsymbol{\beta} \ll \llbracket C_2 \rrbracket} \mathcal{L}_{\mathsf{fin}}(\mathsf{guard}(b, \boldsymbol{\alpha}, \boldsymbol{\beta}))(s) \\ &= \sup_{\boldsymbol{\alpha} \ll \llbracket C_1 \rrbracket} \sup_{\boldsymbol{\beta} \ll \llbracket C_2 \rrbracket} \mathcal{L}_{\mathsf{fin}}(\{\mathsf{guard}(x,b,\alpha,\beta) \mid x \in nodes, \alpha \in \boldsymbol{\alpha}, \beta \in \boldsymbol{\beta}\})(s) \\ &= \sup_{\boldsymbol{\alpha} \ll \llbracket C_1 \rrbracket} \sup_{\boldsymbol{\beta} \ll \llbracket C_2 \rrbracket} \mathcal{L}_{\mathsf{fin}}([\mathsf{guard}(x,b,\alpha,\beta)])(s) \\ &= \sup_{\boldsymbol{\alpha} \ll \llbracket C_1 \rrbracket} \sup_{\boldsymbol{\beta} \ll \llbracket C_2 \rrbracket} \mathcal{L}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta), \mathsf{true}, \emptyset)(s) \\ &= \sup_{\boldsymbol{\alpha} \ll \llbracket C_1 \rrbracket} \sup_{\boldsymbol{\beta} \ll \llbracket C_2 \rrbracket} \begin{cases} \mathcal{L}_{lpo}(\alpha, \mathsf{true}, \emptyset)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 1 \\ \mathcal{L}_{lpo}(\beta, \mathsf{true}, \emptyset)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 0 \end{cases} \qquad \text{(By Lemma F.6)} \\ &= \begin{cases} \sup_{\boldsymbol{\alpha} \ll \llbracket C_1 \rrbracket} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha})(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 1 \\ \sup_{\boldsymbol{\beta} \ll \llbracket C_2 \rrbracket} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\beta})(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 0 \end{cases} \\ &= \begin{cases} \mathcal{L}(\llbracket C_1 \rrbracket)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 1 \\ \mathcal{L}(\llbracket C_2 \rrbracket)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 0 \end{cases} \end{aligned}$$

□

E. Properties of Linearization: While Loops

Lemma F.7. *Let $D(S)$ be the dcpo such that $\mathcal{L}$: pom $\to S \to D(S)$, and let $\bot_{pom} \in$ pom be the bottom of the pomset order, $\bot_D \in D(S)$ be the bottom of $D(S)$, and $\bot_D^\bullet$: $S \to D(S)$ be the bottom of the pointwise order on $D(S)$. Then*

$$\mathcal{L}\left(\Phi^n_{\langle C,b\rangle}(\bot_{pom})\right) = \Psi^n_{\langle \mathcal{L}(\llbracket C \rrbracket),b\rangle}(\bot_D^\bullet)$$

Proof. The proof is by induction on n. If $n = 0$, then we have:

$$\mathcal{L}\left(\Phi^0_{\langle C,b\rangle}(\bot_{pom})\right)(s) = \mathcal{L}(\bot_{pom})(s) = \bot_D = \bot_D^\bullet(s) = \Psi^0_{\langle \mathcal{L}(\llbracket C \rrbracket),b\rangle}(\bot_D^\bullet)(s)$$

Now, suppose the claim holds for n, then we have:

$$\begin{aligned} \mathcal{L}\left(\Phi^{n+1}_{\langle C,b\rangle}(\bot_{pom})\right)(s) &= \mathcal{L}\left(\Phi_{\langle C,b\rangle}\left(\Phi^n_{\langle C,b\rangle}(\bot_{pom})\right)\right)(s) \\ &= \mathcal{L}\left(\mathsf{guard}\left(b, \llbracket C \rrbracket \fatsemi \Phi^n_{\langle C,b\rangle}(\bot_{pom}), \llbracket \texttt{skip} \rrbracket\right)\right)(s) \end{aligned}$$

By Lemma F.6.

$$= \begin{cases} \mathcal{L}\left(\llbracket C \rrbracket \fatsemi \Phi^n_{\langle C,b\rangle}(\bot_{pom})\right)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 1 \\ \mathcal{L}(\llbracket \texttt{skip} \rrbracket)(s) & \text{if } \llbracket b \rrbracket_{test}(s) = 0 \end{cases}$$

By Lemmas 11 and 12.

$$= \begin{cases} \mathcal{L}\left(\Phi^n_{\langle C,b\rangle}(\bot_{pom})\right)^\dagger(\mathcal{L}([\![C]\!])(s)) & \text{if } [\![b]\!]_{test}(s) = 1 \\ \eta(s) & \text{if } [\![b]\!]_{test}(s) = 0 \end{cases}$$

By the induction hypothesis.

$$= \begin{cases} \left(\Psi^n_{\langle \mathcal{L}[\![C]\!],b\rangle}(\bot^\bullet_D)\right)^\dagger(\mathcal{L}([\![C]\!])(s)) & \text{if } [\![b]\!]_{test}(s) = 1 \\ \eta(s) & \text{if } [\![b]\!]_{test}(s) = 0 \end{cases}$$
$$= \Psi_{\langle \mathcal{L}[\![C]\!],b\rangle}\left(\Psi^n_{\langle \mathcal{L}[\![C]\!],b\rangle}(\bot^\bullet_D)\right)(s)$$
$$= \Psi^{n+1}_{\langle \mathcal{L}[\![C]\!],b\rangle}(\bot^\bullet_D)(s)$$ □

Lemma 14. $\mathcal{L}([\![\texttt{while}\ b\ \{C\}]\!]) = \mathsf{lfp}\left(\Psi_{\langle \mathcal{L}([\![C]\!]),b\rangle}\right)$ *where*

$$\Psi_{\langle f,b\rangle}(g)(s) \triangleq \begin{cases} g^\dagger(f(s)) & \text{if } [\![b]\!]_{test}(\sigma) = 1 \\ \eta(s) & \text{if } [\![b]\!]_{test}(\sigma) = 0 \end{cases}$$

Proof.

$$\begin{aligned} \mathcal{L}([\![\texttt{while}\ b\ \{C\}]\!]) &= \mathcal{L}\left(\mathsf{lfp}\left(\Phi_{\langle C,b\rangle}\right)\right) \\ &= \mathcal{L}\left(\sup_{n\in\mathbb{N}} \Phi^n_{\langle C,b\rangle}(\bot_{pom})\right) && \text{(By the Kleene fixed point theorem)} \\ &= \sup_{n\in\mathbb{N}} \mathcal{L}\left(\Phi^n_{\langle C,b\rangle}(\bot_{pom})\right) && \text{(By Scott continuity of } \mathcal{L}) \\ &= \sup_{n\in\mathbb{N}} \Psi^n_{\langle \mathcal{L}([\![C]\!]),b\rangle}(\bot^\bullet_D) && \text{(By Lemma F.7)} \\ &= \mathsf{lfp}\left(\Psi_{\langle \mathcal{L}([\![C]\!]),b\rangle}\right) && \text{(By the Kleene fixed point theorem)} \end{aligned}$$ □

APPENDIX G
POWERDOMAINS

In this section, we fix our domain of computation to be the Hoare powerdomain $\langle \mathcal{P}(S), \subseteq\rangle$. In particular, this means that suprema are given by set union: $\sup_{i\in I} S_i \triangleq \bigcup_{i\in I} S_i$.

Lemma 15. *If* $\mathsf{Bot}_\alpha = \emptyset$, $\varphi_\alpha(x) = \mathsf{true}$ *for all* $x \in N_\alpha$, *and* $f = [\mathsf{true} \mapsto \beta]$, *then:*

$$N_{\alpha \S_f \beta} = N_\alpha \cup N_\beta \quad \textit{and} \quad <_{\alpha \S_f \beta} = <_\alpha \cup <_\beta \cup (N_\alpha \times N_\beta)$$

Proof. First note that since $\mathsf{Bot}_\alpha = \emptyset$, then $\mathsf{stuck}_\alpha = \mathsf{false}$ and so $\mathsf{ext}_\alpha = N_\alpha$. Therefore, $\mathsf{br}_\alpha = \{\bigwedge_{x\in N_\alpha} \varphi_\alpha(x)\} = \{\mathsf{true}\}$. We now show the condition on nodes.

$$N_{\alpha \S_f \beta} = N_\alpha \cup \bigcup_{\psi\in \mathsf{br}_\alpha} N_{f(\psi)} = N_\alpha \cup N_{f(\mathsf{true})} = N_\alpha \cup N_\beta$$

We now also show the condition on orders:

$$\begin{aligned} <_{\alpha \S_f \beta} &= <_\alpha \cup \bigcup_{\psi\in \mathsf{br}_\alpha} \left(<_{f(\psi)} \cup \{x \in N_\alpha \mid \psi \Rightarrow \varphi_\alpha(x)\} \times N_{f(\psi)}\right) \\ &= <_\alpha \cup <_{f(\mathsf{true})} \cup \{x \in N_\alpha \mid \mathsf{true} \Rightarrow \mathsf{true}\} \times N_{f(\mathsf{true})} \\ &= <_\alpha \cup <_\beta \cup (N_\alpha \times N_\beta) \end{aligned}$$ □

A. Properties of the Translation Function

Lemma G.1 (Monotonicity of tr_{lpo})**.** *If* $\alpha \sqsubseteq_{lpo} \alpha'$ *and* $\psi \in \mathsf{br}_\alpha$, *then* $\mathsf{tr}_{lpo}(\alpha,\psi) = \mathsf{tr}_{lpo}(\alpha',\psi)$

Proof. We first show that $N_{\mathsf{tr}_{lpo}(\alpha,\psi)} = N_{\mathsf{tr}_{lpo}(\alpha',\psi)}$, or in other words, that $\{x \in N_\alpha \mid \psi \Rightarrow \varphi_\alpha(x)\} = \{x \in N_{\alpha'} \mid \psi \Rightarrow \varphi_{\alpha'}(x)\}$ The forward inclusion is immediate since $N_\alpha \subseteq N'_\alpha$ and $\varphi_\alpha(x) = \varphi_{\alpha'}(x)$ for all $x \in N_\alpha$. We now show the reverse inclusion. Take any $x \in N_{\alpha'}$ such that $\psi \Rightarrow \varphi_{\alpha'}(x)$. If $x \notin N_\alpha$, then by Lemma 1, $x \in \mathsf{Bot}_\alpha{\uparrow_{\alpha'}}$. This means that there is some $y \in \mathsf{Bot}_\alpha$ such that $y <_{\alpha'} x$, and so $\varphi_{\alpha'}(x) \Rightarrow \varphi_{\alpha'}(y) \Rightarrow \mathsf{stuck}_\alpha$. But this contradicts the fact that $\psi \Rightarrow \varphi_{\alpha'}(x)$ and $\psi \in \mathsf{br}_\alpha$. Therefore, it must be that $x \in N_\alpha$, and so the claim holds.

Now, for the order, we have:

$$<_{\mathsf{tr}_{lpo}(\alpha,\psi)} = <_\alpha \cap (N_{\mathsf{tr}_{lpo}(\alpha,\psi)} \times N_{\mathsf{tr}_{lpo}(\alpha,\psi)})$$

$$= (<_{\alpha'} \cap (N_\alpha \times N_\alpha)) \cap (N_{\mathsf{tr}_{lpo}(\alpha,\psi)} \times N_{\mathsf{tr}_{lpo}(\alpha,\psi)})$$

Since $N_{\mathsf{tr}_{lpo}(\alpha,\psi)} \subseteq N_\alpha$:

$$\begin{aligned}
&= <_{\alpha'} \cap (N_{\mathsf{tr}_{lpo}(\alpha,\psi)} \times N_{\mathsf{tr}_{lpo}(\alpha,\psi)}) \\
&= <_{\alpha'} \cap (N_{\mathsf{tr}_{lpo}(\alpha',\psi)} \times N_{\mathsf{tr}_{lpo}(\alpha',\psi)}) \\
&= <_{\mathsf{tr}_{lpo}(\alpha',\psi)}
\end{aligned}$$

Finally, for any $x \in N_{\mathsf{tr}_{lpo}(\alpha,\psi)}$, we have:

$$\lambda_{\mathsf{tr}_{lpo}(\alpha,\psi)}(x) = \begin{cases} \lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \in \mathit{test} \text{ and } \psi \Rightarrow x \\ \mathsf{assume}\ \neg\lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \in \mathit{test} \text{ and } \psi \Rightarrow \neg x \end{cases}$$

By virtue of the fact that $x \in N_{\mathsf{tr}_{lpo}(\alpha,\psi)}$, we know that $\psi \Rightarrow \varphi_\alpha(x)$, and since $\psi \in \mathsf{br}_\alpha$, then $\psi \Rightarrow \neg\mathsf{stuck}_\alpha$, therefore $\lambda_\alpha(x)$ cannot be $\bot$, and since *label* is a flat cpo, then it must be that $\lambda_\alpha(x) = \lambda_{\alpha'}(x)$.

$$\begin{aligned}
&= \begin{cases} \lambda_{\alpha'}(x) & \text{if } \lambda_{\alpha'}(x) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_{\alpha'}(x) & \text{if } \lambda_{\alpha'}(x) \in \mathit{test} \text{ and } \psi \Rightarrow x \\ \mathsf{assume}\ \neg\lambda_{\alpha'}(x) & \text{if } \lambda_{\alpha'}(x) \in \mathit{test} \text{ and } \psi \Rightarrow \neg x \end{cases} \\
&= \lambda_{\mathsf{tr}_{lpo}(\alpha',\psi)}(x)
\end{aligned}$$

And finally, clearly we have $\varphi_{\mathsf{tr}_{lpo}(\alpha,\psi)}(x) = \varphi_{\mathsf{tr}_{lpo}(\alpha',\psi)}(x) = \mathsf{true}$. □

Lemma G.2 (Monotonicity of $\mathsf{tr}_{\mathsf{fin}}$)**.** *If* $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\alpha}'$*, then* $\mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}) \subseteq \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}')$.

Proof. Since $\boldsymbol{\alpha} \sqsubseteq_{pom} \boldsymbol{\alpha}'$, then by Lemma C.1 there exists an $\alpha \in \boldsymbol{\alpha}$ and $\alpha' \in \boldsymbol{\alpha}'$ such that $\alpha \sqsubseteq_{lpo} \alpha'$. We now prove the claim as follows:

$$\begin{aligned}
\mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}) &= \mathsf{tr}_{\mathsf{fin}}([\alpha]) \\
&= \{[\beta] \mid \psi \in \mathsf{br}_\alpha, \beta \in \mathsf{tr}_{lpo}(\alpha)\} \\
&\subseteq \{[\beta] \mid \psi \in \mathsf{br}_{\alpha'}, \beta \in \mathsf{tr}_{lpo}(\alpha)\} && \text{By Lemma 8.} \\
&= \{[\beta] \mid \psi \in \mathsf{br}_{\alpha'}, \beta \in \mathsf{tr}_{lpo}(\alpha')\} && \text{By Lemma G.1.} \\
&= \mathsf{tr}_{\mathsf{fin}}([\alpha']) \\
&= \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}')
\end{aligned}$$

□

B. Translation and Pomset Operations

Lemma G.3. *For any* $\ell \in \mathit{act} \cup \{\mathsf{fork}\}$*:*

$$\mathsf{tr}(\langle \ell \rangle) = \{\langle \ell \rangle\}$$

Proof. Since $\langle \ell \rangle$ is a finite pomset:

$$\mathsf{tr}(\langle \ell \rangle) = \mathsf{tr}_{\mathsf{fin}}(\langle \ell \rangle)$$

Fixing any arbitrary $x \in \mathit{nodes}$, we have that $\langle \ell \rangle = [\langle \ell \rangle_x]$.

$$\begin{aligned}
&= \mathsf{tr}_{\mathsf{fin}}([\langle \ell \rangle_x]) \\
&= \{[\mathsf{tr}_{lpo}(\langle \ell \rangle_x, \psi)] \mid \psi \in \mathsf{br}_{\langle \ell \rangle_x}\}
\end{aligned}$$

Since $\mathsf{br}_{\langle \ell \rangle_x} = \{\mathsf{true}\}$ and $\mathsf{tr}_{lpo}(\langle \ell \rangle_x, \mathsf{true}) = \langle \ell \rangle_x$:

$$\begin{aligned}
&= \{[\langle \ell \rangle_x]\} \\
&= \{\langle \ell \rangle\}
\end{aligned}$$

□

For sequential composition of LPOFs, where $\mathsf{br}_\alpha = \{\mathsf{true}\}$, we will write $\alpha \fatsemi \beta$ (omitting the copy function f), to mean $\alpha \fatsemi_f \beta$ where $f \triangleq [\mathsf{true} \mapsto \beta]$, *i.e.,* f does not perform any renaming, since only a single isomorphic copy of β is needed.

Lemma G.4. *For any* $\alpha, \beta \in \mathit{lpo}_{\mathsf{fin}}(\mathit{label})$*,* $f \in \mathsf{copy}_{\alpha,\beta}$*,* $\psi \in \mathsf{br}_\alpha$*, and* $\psi' \in \mathsf{br}_{f(\psi)}$*,*

$$\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi \wedge \psi') = \mathsf{tr}_{lpo}(\alpha, \psi) \fatsemi \mathsf{tr}_{lpo}(f(\psi), \psi')$$

Proof. We show the equality component-wise. Some of the steps use Lemma 15. First, for the nodes, we have:

$$N_{\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi \wedge \psi')} = \{x \in N_{\alpha \fatsemi_f \beta} \mid \psi \wedge \psi' \Rightarrow \varphi_{\alpha \fatsemi_f \beta}(x)\}$$

Since $\psi \in \mathsf{br}_\alpha$, then we will only take nodes from $f(\psi)$ and no other $\psi'' \in \mathsf{br}_\alpha$

$$\begin{aligned}
&= \{x \in N_\alpha \mid \psi \wedge \psi' \Rightarrow \varphi_\alpha(x)\} \cup \{x \in N_{f(\psi)} \mid \psi \wedge \psi' \Rightarrow \varphi_{f(\psi)}(x) \wedge \psi\} \\
&= \{x \in N_\alpha \mid \psi \Rightarrow \varphi_\alpha(x)\} \cup \{x \in N_{f(\psi)} \mid \psi' \Rightarrow \varphi_{f(\psi)}(x)\} \\
&= N_{\mathsf{tr}_{lpo}(\alpha,\psi)} \cup N_{\mathsf{tr}_{lpo}(f(\psi),\psi')} \\
&= N_{\mathsf{tr}_{lpo}(\alpha,\psi) \fatsemi \mathsf{tr}_{lpo}(f(\psi),\psi')}
\end{aligned}$$

Now, we show that the orders are the same:

$$\begin{aligned}
&<_{\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi \wedge \psi')} \\
&= <_{\alpha \fatsemi_f \beta} \cap (N_{\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi \wedge \psi')} \times N_{\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi \wedge \psi')}) \\
&= (<_\alpha \cap (N_{\mathsf{tr}_{lpo}(\alpha,\psi)} \times N_{\mathsf{tr}_{lpo}(\alpha,\psi)}) \cup (<_{f(\psi)} \cap (N_{\mathsf{tr}_{lpo}(f(\psi),\psi')} \times N_{\mathsf{tr}_{lpo}(f(\psi),\psi')}) \cup (N_{\mathsf{tr}_{lpo}(\alpha,\psi)} \times N_{\mathsf{tr}_{lpo}(f(\psi),\psi')}) \\
&= <_{\mathsf{tr}_{lpo}(\alpha,\psi) \fatsemi \mathsf{tr}_{lpo}(f(\psi),\psi')}
\end{aligned}$$

Now, for any x, clearly we have:

$$\varphi_{\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi \wedge \psi')}(x) = \varphi_{\mathsf{tr}_{lpo}(\alpha,\psi) \fatsemi \mathsf{tr}_{lpo}(f(\psi),\psi')}(x) = \mathsf{true}$$

If $x \in N_\alpha$, then:

$$\begin{aligned}
\lambda_{\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi \wedge \psi')}(x) &= \begin{cases} \lambda_{\alpha \fatsemi_f \beta}(x) & \text{if } \lambda_{\alpha \fatsemi_f \beta}(x) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_{\alpha \fatsemi_f \beta}(x) & \text{if } \lambda_{\alpha \fatsemi_f \beta}(x) \in \mathit{test} \text{ and } \psi \wedge \psi' \Rightarrow x \\ \mathsf{assume}\ \neg\lambda_{\alpha \fatsemi_f \beta}(x) & \text{if } \lambda_{\alpha \fatsemi_f \beta}(x) \in \mathit{test} \text{ and } \psi \wedge \psi' \Rightarrow \neg x \end{cases} \\
&= \begin{cases} \lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \in \mathit{test} \text{ and } \psi \Rightarrow x \\ \mathsf{assume}\ \neg\lambda_\alpha(x) & \text{if } \lambda_\alpha(x) \in \mathit{test} \text{ and } \psi \Rightarrow \neg x \end{cases} \\
&= \lambda_{\mathsf{tr}_{lpo}(\alpha,\psi)}(x) \\
&= \lambda_{\mathsf{tr}_{lpo}(\alpha,\psi) \fatsemi \mathsf{tr}_{lpo}(\beta,\psi')}(x)
\end{aligned}$$

The case for $x \in f(\psi)$ is nearly identical. □

Lemma G.5. $\mathsf{tr}(\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta}) = \{\boldsymbol{\alpha}' \fatsemi \boldsymbol{\beta}' \mid \boldsymbol{\alpha}' \in \mathsf{tr}(\boldsymbol{\alpha}), \boldsymbol{\beta}' \in \mathsf{tr}(\boldsymbol{\beta})\}$.

Proof.

$$\begin{aligned}
\mathsf{tr}(\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta}) &= \mathsf{tr}(\sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \boldsymbol{\alpha}' \fatsemi \boldsymbol{\beta}') \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}' \fatsemi \boldsymbol{\beta}') \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}(\{\alpha \fatsemi_f \beta \mid \alpha \in \boldsymbol{\alpha}', \beta \in \boldsymbol{\beta}', f \in \mathsf{copy}_{\alpha,\beta}\})
\end{aligned}$$

Since the above set is an equivalence class, we can fix any $\alpha \in \boldsymbol{\alpha}'$, $\beta \in \boldsymbol{\beta}'$, and $f \in \mathsf{copy}_{\alpha,\beta}$.

$$\begin{aligned}
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}([\alpha \fatsemi_f \beta]) \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \{[\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi)] \mid \psi \in \mathsf{br}_{\alpha \fatsemi_f \beta}\} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \{[\mathsf{tr}_{lpo}(\alpha \fatsemi_f \beta, \psi \wedge \psi')] \mid \psi \in \mathsf{br}_\alpha, \psi' \in \mathsf{br}_{f(\psi)}\} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \{[\mathsf{tr}_{lpo}(\alpha,\psi) \fatsemi \mathsf{tr}_{lpo}(f(\psi),\psi')] \mid \psi \in \mathsf{br}_\alpha, \psi' \in \mathsf{br}_{f(\psi)}\} \qquad \text{(By Lemma G.4)} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}') \fatsemi \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\beta}') \\
&= \left(\bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}')\right) \fatsemi \left(\bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\beta}')\right)
\end{aligned}$$

$$= \mathsf{tr}(\boldsymbol{\alpha}) \fatsemi \mathsf{tr}(\boldsymbol{\beta})$$ □

Lemma G.6. $\mathsf{tr}_{lpo}(\mathsf{guard}(x, b, \alpha, \beta), \psi \wedge x) = \langle \mathsf{assume}\ b \rangle_x \fatsemi \mathsf{tr}_{lpo}(\alpha, \psi)$ *and* $\mathsf{tr}_{lpo}(\mathsf{guard}(x, b, \alpha, \beta), \psi \wedge \neg x) = \langle \mathsf{assume}\ \neg b \rangle_x \fatsemi \mathsf{tr}_{lpo}(\beta, \psi)$.

Proof. We prove the equality for the first case. The second is entirely symmetrical. We first show equality of the node sets:

$$N_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)} = \{y \in N_{\mathsf{guard}(x,b,\alpha,\beta)} \mid \psi \wedge x \Rightarrow \varphi_{\mathsf{guard}(x,b,\alpha,\beta)}(y)\}$$

By the construction of guard, x is true in all nodes in the α branch, and the root x has formula true.

$$\begin{aligned}
&= \{x\} \cup \{y \in N_\alpha \mid \psi \Rightarrow \varphi_\alpha(y)\} \\
&= N_{\langle \mathsf{assume}\ b \rangle_x} \cup N_{\mathsf{tr}(\alpha,\psi)} \\
&= N_{\langle \mathsf{assume}\ b \rangle_x \fatsemi \mathsf{tr}(\alpha,\psi)}
\end{aligned}$$

For the order, we have:

$$\begin{aligned}
<_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)} &= <_{\mathsf{guard}(x,b,\alpha,\beta)} \cap \left(N_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)} \times N_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)}\right) \\
&= \left(<_\alpha \cup <_\beta \cup \left(\{x\} \times (N_\alpha \cup N_\beta)\right)\right) \cap \left(N_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)} \times N_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)}\right)
\end{aligned}$$

Using what we just showed about the node sets:

$$\begin{aligned}
&= \left(<_\alpha \cap \left(N_{\mathsf{tr}(\alpha,\psi)} \times N_{\mathsf{tr}(\alpha,\psi)}\right)\right) \cup \left(\{x\} \times N_{\mathsf{tr}(\alpha,\psi)}\right) \\
&= <_{\mathsf{tr}(\alpha,\psi)} \cup \left(\{x\} \times N_{\mathsf{tr}(\alpha,\psi)}\right) \\
&= <_{\langle \mathsf{assume}\ b \rangle_x \fatsemi \mathsf{tr}(\alpha,\psi)}
\end{aligned}$$

For every y, clearly:

$$\varphi_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)}(y) = \varphi_{\langle \mathsf{assume}\ b \rangle_x \fatsemi \mathsf{tr}(\alpha,\psi)}(y) = \mathsf{true}$$

If $y = x$, then we have:

$$\begin{aligned}
\lambda_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)}(y) &= \begin{cases} \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) \in \mathit{test} \text{ and } \psi \wedge x \Rightarrow y \\ \mathsf{assume}\ \neg\lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) \in \mathit{test} \text{ and } \psi \wedge x \Rightarrow \neg y \end{cases} \\
&= \mathsf{assume}\ \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(x) \\
&= \mathsf{assume}\ b \\
&= \lambda_{\langle \mathsf{assume}\ b \rangle_x}(x) \\
&= \lambda_{\langle \mathsf{assume}\ b \rangle_x \fatsemi \mathsf{tr}(\alpha,\psi)}(x)
\end{aligned}$$

If not, then $y \in \mathsf{tr}(\alpha, \psi) \subseteq N_\alpha$:

$$\begin{aligned}
\lambda_{\mathsf{tr}_{lpo}(\mathsf{guard}(x,b,\alpha,\beta),\psi\wedge x)}(y) &= \begin{cases} \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) \in \mathit{test} \text{ and } \psi \wedge x \Rightarrow y \\ \mathsf{assume}\ \neg\lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) & \text{if } \lambda_{\mathsf{guard}(x,b,\alpha,\beta)}(y) \in \mathit{test} \text{ and } \psi \wedge x \Rightarrow \neg y \end{cases} \\
&= \begin{cases} \lambda_\alpha(y) & \text{if } \lambda_\alpha(y) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_\alpha(y) & \text{if } \lambda_\alpha(y) \in \mathit{test} \text{ and } \psi \Rightarrow y \\ \mathsf{assume}\ \neg\lambda_\alpha(y) & \text{if } \lambda_\alpha(y) \in \mathit{test} \text{ and } \psi \Rightarrow \neg y \end{cases} \\
&= \lambda_{\mathsf{tr}(\alpha,\psi)}(y) \\
&= \lambda_{\langle \mathsf{assume}\ b \rangle_x \fatsemi \mathsf{tr}(\alpha,\psi)}(y)
\end{aligned}$$ □

Lemma G.7. $\mathsf{tr}(\mathsf{guard}(b, \boldsymbol{\alpha}, \boldsymbol{\beta})) = \{\langle \mathsf{assume}\ b \rangle \fatsemi \boldsymbol{\alpha}' \mid \boldsymbol{\alpha}' \in \mathsf{tr}(\boldsymbol{\alpha})\} \cup \{\langle \mathsf{assume}\ \neg b \rangle \fatsemi \boldsymbol{\beta}' \mid \boldsymbol{\beta}' \in \mathsf{tr}(\boldsymbol{\beta})\}$.

Proof.

$$\begin{aligned}
&\mathsf{tr}(\mathsf{guard}(b, \boldsymbol{\alpha}, \boldsymbol{\beta})) \\
&= \mathsf{tr}\left(\sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{guard}(b, \boldsymbol{\alpha}', \boldsymbol{\beta}')\right) \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}(\mathsf{guard}(b, \boldsymbol{\alpha}', \boldsymbol{\beta}'))
\end{aligned}$$

$$= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}(\{\mathsf{guard}(x, b, \alpha, \beta) \mid \alpha \in \boldsymbol{\alpha}', \beta \in \boldsymbol{\beta}', x \in \mathit{nodes}\})$$

Since the set above is an equivalence class, we can fix any x, α, and β and write:

$$\begin{aligned}
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}([\mathsf{guard}(x, b, \alpha, \beta)]) \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \{[\mathsf{tr}_{\mathit{lpo}}(\mathsf{guard}(x, b, \alpha, \beta), \psi)] \mid \psi \in \mathsf{br}_{\mathsf{guard}(x,b,\alpha,\beta)}\} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \{[\mathsf{tr}_{\mathit{lpo}}(\mathsf{guard}(x, b, \alpha, \beta), \psi \wedge x)] \mid \psi \in \mathsf{br}_{\alpha}\} \cup \{[\mathsf{tr}_{\mathit{lpo}}(\mathsf{guard}(x, b, \alpha, \beta), \psi \wedge \neg x)] \mid \psi \in \mathsf{br}_{\beta}\}
\end{aligned}$$

By Lemma G.6:

$$\begin{aligned}
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \{[\langle \mathsf{assume}\ b \rangle_x \fatsemi \mathsf{tr}_{\mathit{lpo}}(\alpha, \psi)] \mid \psi \in \mathsf{br}_{\alpha}\} \cup \{[\langle \mathsf{assume}\ \neg b \rangle_x \fatsemi \mathsf{tr}_{\mathit{lpo}}(\beta, \psi)] \mid \psi \in \mathsf{br}_{\beta}\} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \{\langle \mathsf{assume}\ b \rangle \fatsemi \boldsymbol{\alpha}'' \mid \boldsymbol{\alpha}'' \in \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}')\} \cup \{\langle \mathsf{assume}\ \neg b \rangle \fatsemi \boldsymbol{\beta}'' \mid \boldsymbol{\beta}'' \in \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\beta}')\} \\
&= \{\langle \mathsf{assume}\ b \rangle \fatsemi \boldsymbol{\alpha}'' \mid \boldsymbol{\alpha}'' \in \mathsf{tr}(\boldsymbol{\alpha})\} \cup \{\langle \mathsf{assume}\ \neg b \rangle \fatsemi \boldsymbol{\beta}'' \mid \boldsymbol{\beta}'' \in \mathsf{tr}(\boldsymbol{\beta})\}
\end{aligned}$$

□

Lemma G.8. *For any $\alpha, \beta \in \mathit{lpo}_{\mathsf{fin}}$, $\psi \in \mathsf{br}_{\alpha}$, and $\psi' \in \mathsf{br}_{\beta}$:*

$$\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi') = \mathsf{tr}_{\mathit{lpo}}(\alpha, \psi) \parallel_x \mathsf{tr}_{\mathit{lpo}}(\beta, \psi')$$

Proof. We show the equality for each component. First, for the nodes, we have:

$$\begin{aligned}
N_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')} &= \{y \in N_{\alpha \parallel_x \beta} \mid \psi \wedge \psi' \Rightarrow \varphi_{\alpha \parallel_x \beta}(y)\} \\
&= \{x\} \cup \{y \in \mathsf{nFork}_{\alpha} \mid \psi \Rightarrow \varphi_{\alpha}(y)\} \cup \{z \in \mathsf{nFork}_{\beta} \mid \psi' \Rightarrow \varphi_{\beta}(z)\} \\
&= \{x\} \cup \mathsf{nFork}_{\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi)} \cup \mathsf{nFork}_{\mathsf{tr}_{\mathit{lpo}}(\beta, \psi')} \\
&= N_{\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi) \parallel_x \mathsf{tr}_{\mathit{lpo}}(\beta, \psi')}
\end{aligned}$$

Now, for the orders:

$$\begin{aligned}
<_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')} &= <_{\alpha \parallel_x \beta} \cap \left(N_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')} \times N_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')}\right) \\
&= ((\{x\} \times (\mathsf{nFork}_{\alpha} \cup \mathsf{nFork}_{\beta})) \cup\ (<_{\alpha} \cap (\mathsf{nFork}_{\alpha} \times \mathsf{nFork}_{\alpha})) \cup\ (<_{\beta} \cap (\mathsf{nFork}_{\beta} \times \mathsf{nFork}_{\beta}))) \\
&\quad \cap \left(N_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')} \times N_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')}\right) \\
&= (\{x\} \times (\mathsf{nFork}_{\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi)} \cup \mathsf{nFork}_{\mathsf{tr}_{\mathit{lpo}}(\beta, \psi')})) \\
&\quad \cup\ (<_{\alpha} \cap (\mathsf{nFork}_{\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi)} \times \mathsf{nFork}_{\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi)})) \\
&\quad \cup\ (<_{\beta} \cap (\mathsf{nFork}_{\mathsf{tr}_{\mathit{lpo}}(\beta, \psi')} \times \mathsf{nFork}_{\mathsf{tr}_{\mathit{lpo}}(\beta, \psi')}) \\
&= <_{\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi) \parallel_x \mathsf{tr}_{\mathit{lpo}}(\beta, \psi')}
\end{aligned}$$

Now take any y, clearly we have:

$$\varphi_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')}(y) = \varphi_{\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi) \parallel_x \mathsf{tr}_{\mathit{lpo}}(\beta, \psi')}(y) = \mathsf{true}$$

Now, if $y = x$, then $\lambda_{\alpha \parallel_x \beta}(y) = \mathsf{fork}$, so:

$$\begin{aligned}
\lambda_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')}(y) &= \begin{cases} \lambda_{\alpha \parallel_x \beta}(y) & \text{if } \lambda_{\alpha \parallel_x \beta}(y) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_{\alpha \parallel_x \beta}(y) & \text{if } \lambda_{\alpha \parallel_x \beta}(y) \in \mathit{test} \text{ and } \psi \wedge \psi' \Rightarrow x \\ \mathsf{assume}\ \neg\lambda_{\alpha \parallel_x \beta}(y) & \text{if } \lambda_{\alpha \parallel_x \beta}(y) \in \mathit{test} \text{ and } \psi \wedge \psi' \Rightarrow \neg x \end{cases} \\
&= \lambda_{\alpha \parallel_x \beta}(y) \\
&= \mathsf{fork} \\
&= \lambda_{\mathsf{tr}_{\mathit{lpo}}(\alpha, \psi) \parallel_x \mathsf{tr}_{\mathit{lpo}}(\beta, \psi')}(y)
\end{aligned}$$

Now, if $y \in \mathsf{nFork}_{\alpha}$, then:

$$\lambda_{\mathsf{tr}_{\mathit{lpo}}(\alpha \parallel_x \beta, \psi \wedge \psi')}(y) = \begin{cases} \lambda_{\alpha \parallel_x \beta}(y) & \text{if } \lambda_{\alpha \parallel_x \beta}(y) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_{\alpha \parallel_x \beta}(y) & \text{if } \lambda_{\alpha \parallel_x \beta}(y) \in \mathit{test} \text{ and } \psi \wedge \psi' \Rightarrow x \\ \mathsf{assume}\ \neg\lambda_{\alpha \parallel_x \beta}(y) & \text{if } \lambda_{\alpha \parallel_x \beta}(y) \in \mathit{test} \text{ and } \psi \wedge \psi' \Rightarrow \neg x \end{cases}$$

$$= \begin{cases} \lambda_\alpha(y) & \text{if } \lambda_\alpha(y) \notin \mathit{test} \\ \mathsf{assume}\ \lambda_\alpha(y) & \text{if } \lambda_\alpha(y) \in \mathit{test} \text{ and } \psi \Rightarrow x \\ \mathsf{assume}\ \neg\lambda_\alpha(y) & \text{if } \lambda_\alpha(y) \in \mathit{test} \text{ and } \psi \Rightarrow \neg x \end{cases}$$
$$= \lambda_{\mathsf{tr}(\alpha,\psi)}(y)$$
$$= \lambda_{\mathsf{tr}_{lpo}(\alpha,\psi)\|_x \mathsf{tr}_{lpo}(\beta,\psi')}(y)$$

The case where $y \in \mathsf{nBot}(\beta)$ is nearly identical. □

Since $\|$ is not monotone (see Remark 3), we need a different way to approximate infinite pomsets composed in parallel, which we now describe. Let $\mathsf{root}(\gamma)$ be the label of the root node of the pomset γ and

$$\gamma \ll_1 \gamma' \qquad \text{iff} \qquad \gamma \ll \gamma' \quad \text{and} \quad \mathsf{root}(\gamma) = \mathsf{root}(\gamma') = \mathsf{fork} \text{ or } \mathsf{root}(\gamma') \neq \mathsf{fork}$$

i.e., $\gamma \ll_1 \gamma'$ is similar to the standard approximation order $\ll$ on pomsets, but it prevents a fork node at the root of γ' from being converted to $\bot$, which would cause the root of a parallel composition involving γ' vs γ to have a different number of successors. Coming back to the example of Remark 3, we have that $[\alpha] \sqsubseteq_{\mathit{pom}} [\beta]$ and also $[\alpha] \ll [\beta]$, whereas $[\alpha] \not\ll_1 [\beta]$, since the root of β is fork but not so is the root of α.

We need two properties of $\ll_1$. First:

$$\sup_{\gamma \ll_1 \gamma'} \gamma = \sup_{\gamma \ll \gamma'} \gamma \tag{2}$$

By Lemma 5 and Corollary 5, we have that $\sup_{\gamma \ll \gamma'} \gamma = \gamma'$. If $\mathsf{root}(\gamma') = \mathsf{fork}$, then $\{\gamma \mid \gamma \ll_1 \gamma'\} = \{\gamma \mid \gamma \ll \gamma'\} \setminus \{\bot_{\mathit{pom}}\}$, which clearly has the same supremum, as we have only removed the bottom element from the set. If $\mathsf{root}(\gamma') \neq \mathsf{fork}$, then the orders are equivalent. The second property that we need is

$$\lfloor \boldsymbol{\alpha} \parallel \boldsymbol{\beta} \rfloor_{\mathsf{fin}} \setminus \{\bot_{\mathit{pom}}\} = \{\boldsymbol{\alpha}' \parallel \boldsymbol{\beta}' \mid \boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}, \boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}\} \tag{3}$$

This can be easily seen by observing that $\bot_{\mathit{pom}}$ is not included in the right set, since all parallel compositions have fork at the root therein. Moreover, the successors nodes (but not their labels) of the root are also fixed in both cases: on the left they are fixed because the root is fixed, and on the right they are fixed due to the $\ll_1$ order, since any fork node at the root would be removed by the parallel composition. Above level 2, both sets allow for any approximation of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

Lemma G.9. $\mathsf{tr}(\boldsymbol{\alpha} \parallel \boldsymbol{\beta}) = \{\boldsymbol{\alpha}' \parallel \boldsymbol{\beta}' \mid \boldsymbol{\alpha}' \in \mathsf{tr}(\boldsymbol{\alpha}), \boldsymbol{\beta}' \in \mathsf{tr}(\boldsymbol{\beta})\}$.

Proof.

$$\begin{aligned}
\mathsf{tr}(\boldsymbol{\alpha} \parallel \boldsymbol{\beta}) &= \mathsf{tr}\left(\sup_{\gamma \in \lfloor \boldsymbol{\alpha} \parallel \boldsymbol{\beta} \rfloor_{\mathsf{fin}}} \gamma\right) \\
&= \mathsf{tr}(\langle \bot \rangle) \cup \mathsf{tr}\left(\sup_{\gamma \in \lfloor \boldsymbol{\alpha} \parallel \boldsymbol{\beta} \rfloor_{\mathsf{fin}} \setminus \{\bot_{\mathit{pom}}\}} \gamma\right) \\
&= \emptyset \cup \mathsf{tr}\left(\sup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \boldsymbol{\alpha}' \parallel \boldsymbol{\beta}'\right) && \text{By Equation (3)} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}' \parallel \boldsymbol{\beta}') \\
&= \bigcup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}(\{\alpha \parallel_x \beta \mid \alpha \in \boldsymbol{\alpha}', \beta \in \boldsymbol{\beta}', x \in \mathit{nodes}\}) \\
&= \bigcup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \mathsf{tr}_{\mathsf{fin}}([\alpha \parallel_x \beta]) \\
&= \bigcup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \{[\mathsf{tr}_{lpo}(\alpha \parallel_x \beta, \psi)] \mid \psi \in \mathsf{br}_{\alpha \parallel_x \beta}\} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \{[\mathsf{tr}_{lpo}(\alpha \parallel_x \beta, \psi \wedge \psi')] \mid \psi \in \mathsf{br}_\alpha, \psi' \in \mathsf{br}_\beta\} && \text{By Lemma G.8} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \{[\{\alpha' \parallel_x \beta' \mid \alpha' \in \mathsf{tr}_{lpo}(\alpha, \psi), \beta' \in \mathsf{tr}_{lpo}(\beta, \psi')\}] \mid \psi \in \mathsf{br}_\alpha, \psi' \in \mathsf{br}_\beta\} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \{\boldsymbol{\alpha}'' \parallel \boldsymbol{\beta}'' \mid \boldsymbol{\alpha}'' \in \{[\mathsf{tr}_{lpo}(\alpha, \psi)] \mid \psi \in \mathsf{br}_\alpha\},\ \boldsymbol{\beta}'' \in \{[\mathsf{tr}_{lpo}(\beta, \psi')] \mid \psi' \in \mathsf{br}_\beta\}\}
\end{aligned}$$

$$= \bigcup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \{\boldsymbol{\alpha}'' \parallel \boldsymbol{\beta}'' \mid \boldsymbol{\alpha}'' \in \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}'),\ \boldsymbol{\beta}'' \in \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\beta}')\}$$
$$= \{\boldsymbol{\alpha}'' \parallel \boldsymbol{\beta}'' \mid \boldsymbol{\alpha}'' \in \mathsf{tr}(\boldsymbol{\alpha}),\ \boldsymbol{\beta}'' \in \mathsf{tr}(\boldsymbol{\beta})\} \qquad \text{By Equation (2)} \ \square$$

Lemma G.10. *If* $\mathsf{tr}(\llbracket C \rrbracket) = \llbracket C \rrbracket_{\mathsf{PL}}$*, for all* $n \in \mathbb{N}$*:*

$$\mathsf{tr}(\Phi^n_{\langle C,b\rangle}(\bot_{pom})) = \Xi^n_{\langle C,b\rangle}(\emptyset)$$

where above we use $\bot_{pom} = \langle \bot \rangle$ *to disambiguate.*

Proof. The proof is by induction on n. If $n = 0$, then we have:

$$\mathsf{tr}\left(\Phi^0_{\langle C,b\rangle}(\bot_{pom})\right) = \mathsf{tr}\left(\bot_{pom}\right) = \{[\mathsf{tr}_{lpo}(\langle \bot \rangle_x, \psi)] \mid \psi \in \mathsf{br}_{\langle \bot \rangle_x}\} = \emptyset = \Xi^0_{\langle C,b\rangle}(\emptyset)$$

Where the second to last step follows from the fact that $\mathsf{br}_{\langle \bot \rangle_x} = \emptyset$. Now, suppose the claim holds for n, we have:

$$\begin{aligned}
\mathsf{tr}(\Phi^{n+1}_{\langle C,b\rangle}(\bot_{pom})) &= \mathsf{tr}\left(\Phi_{\langle C,b\rangle}\left(\Phi^n_{\langle C,b\rangle}(\bot_{pom})\right)\right) \\
&= \mathsf{tr}\left(\mathsf{guard}\left(b, \llbracket C \rrbracket \fatsemi \Phi^n_{\langle C,b\rangle}(\emptyset), \llbracket \mathbf{skip} \rrbracket\right)\right)
\end{aligned}$$

By Lemma G.7.

$$= \left\{\langle \mathsf{assume}\ b\rangle \fatsemi \boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in \mathsf{tr}\left(\llbracket C \rrbracket \fatsemi \Phi^n_{\langle C,b\rangle}(\emptyset)\right)\right\} \cup \{\langle \mathsf{assume}\ b\rangle \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\beta} \in \mathsf{tr}(\llbracket \mathbf{skip} \rrbracket)\}$$

By Lemmas G.3 and G.5.

$$= \left\{\langle \mathsf{assume}\ b\rangle \fatsemi \boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in \mathsf{tr}(\llbracket C \rrbracket), \boldsymbol{\beta} \in \mathsf{tr}\left(\Phi^n_{\langle C,b\rangle}(\emptyset)\right)\right\} \cup \{\langle \mathsf{assume}\ b\rangle \fatsemi \llbracket \mathbf{skip} \rrbracket\}$$

By assumption, and the induction hypothesis:

$$\begin{aligned}
&= \left\{\langle \mathsf{assume}\ b\rangle \fatsemi \boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in \llbracket C \rrbracket_{\mathsf{PL}}, \boldsymbol{\beta} \in \Xi^n_{\langle C,b\rangle}(\emptyset)\right\} \cup \{\langle \mathsf{assume}\ b\rangle \fatsemi \llbracket \mathbf{skip} \rrbracket\} \\
&= \Xi_{\langle C,b\rangle}\left(\Xi^n_{\langle C,b\rangle}(\emptyset)\right) \\
&= \Xi^{n+1}_{\langle C,b\rangle}(\emptyset)
\end{aligned}$$

$\square$

C. Translations and Program Semantics

Lemma G.11. *For any program* $C \in \mathit{cmd}$*,* $\llbracket C \rrbracket_{\mathsf{PL}} = \mathsf{tr}(\llbracket C \rrbracket)$.

Proof. The proof is by induction on the structure of the program.

- $C = \mathbf{skip}$ or $C \in \mathit{act}$. Follows immediately from Lemma G.3.
- $C = C_1; C_2$.

$$\begin{aligned}
\llbracket C_1; C_2 \rrbracket_{\mathsf{PL}} &= \{\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in \llbracket C_1 \rrbracket_{\mathsf{PL}}, \boldsymbol{\beta} \in \llbracket C_2 \rrbracket_{\mathsf{PL}}\} \\
&= \{\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in \mathsf{tr}(\llbracket C_1 \rrbracket), \boldsymbol{\beta} \in \mathsf{tr}(\llbracket C_2 \rrbracket)\} && \text{(By the induction hypothesis.)} \\
&= \mathsf{tr}(\llbracket C_1 \rrbracket \fatsemi \llbracket C_2 \rrbracket) && \text{(By Lemma G.5.)} \\
&= \mathsf{tr}(\llbracket C_1; C_2 \rrbracket)
\end{aligned}$$

- $C = C_1 \,|\, C_2$.

$$\begin{aligned}
\llbracket C_1 \,|\, C_2 \rrbracket_{\mathsf{PL}} &= \{\boldsymbol{\alpha} \parallel \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in \llbracket C_1 \rrbracket_{\mathsf{PL}}, \boldsymbol{\beta} \in \llbracket C_2 \rrbracket_{\mathsf{PL}}\} \\
&= \{\boldsymbol{\alpha} \parallel \boldsymbol{\beta} \mid \boldsymbol{\alpha} \in \mathsf{tr}(\llbracket C_1 \rrbracket), \boldsymbol{\beta} \in \mathsf{tr}(\llbracket C_2 \rrbracket)\} && \text{(By the induction hypothesis.)} \\
&= \mathsf{tr}(\llbracket C_1 \rrbracket \parallel \llbracket C_2 \rrbracket) && \text{(By Lemma G.9.)} \\
&= \mathsf{tr}(\llbracket C_1 \,|\, C_2 \rrbracket)
\end{aligned}$$

- $C = \mathbf{if}\ b\ \{C_1\}\ \mathbf{else}\ \{C_2\}$.

$$\llbracket \mathbf{if}\ b\ \{C_1\}\ \mathbf{else}\ \{C_2\} \rrbracket_{\mathsf{PL}} = \{\langle \mathsf{assume}\ b\rangle \fatsemi \boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in \llbracket C_1 \rrbracket_{\mathsf{PL}}\} \cup \{\langle \mathsf{assume}\ \neg b\rangle \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\beta} \in \llbracket C_2 \rrbracket_{\mathsf{PL}}\}$$

By the induction hypothesis:

$$= \{\langle \mathsf{assume}\ b\rangle \fatsemi \boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in \mathsf{tr}(\llbracket C_1 \rrbracket)\} \cup \{\langle \mathsf{assume}\ \neg b\rangle \fatsemi \boldsymbol{\beta} \mid \boldsymbol{\beta} \in \mathsf{tr}(\llbracket C_2 \rrbracket)\}$$

By Lemma G.7.

$$= \mathsf{tr}(\mathsf{guard}(b, [\![C_1]\!], [\![C_2]\!])) \\ = \mathsf{tr}([\![\texttt{if } b \ \{C_1\} \ \texttt{else} \ \{C_2\}]\!])$$

- $C = \texttt{while } b \ \{C'\}$.

$$\begin{aligned} [\![\texttt{while } b \ \{C'\}]\!]_{\mathsf{PL}} &= \mathsf{lfp}\left(\Xi_{\langle C', b\rangle}\right) \\ &= \bigcup_{n\in\mathbb{N}} \Xi^n_{\langle C', b\rangle}(\emptyset) && \text{(By the Kleene fixed point theorem.)} \\ &= \bigcup_{n\in\mathbb{N}} \mathsf{tr}\left(\Phi^n_{\langle C', b\rangle}(\bot_{pom})\right) && \text{(By Lemma G.10, and the induction hypothesis.)} \\ &= \mathsf{tr}\left(\sup_{n\in\mathbb{N}} \Phi^n_{\langle C', b\rangle}(\bot_{pom})\right) && \text{(By Scott continuity of tr.)} \\ &= \mathsf{tr}\left(\mathsf{lfp}\left(\Phi^n_{\langle C', b\rangle}\right)\right) && \text{(By the Kleene fixed point theorem.)} \\ &= \mathsf{tr}\left([\![\texttt{while } b \ \{C'\}]\!]\right) \end{aligned}$$

□

D. Translations and Linearization

Lemma G.12. *For any binary branching $\alpha \in lpo_{\mathsf{fin}}(label)$, $S \subseteq \mathsf{nBot}_\alpha$, and $\psi \in \mathsf{valid}_\alpha(S)$, then:*

$$\bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathsf{next}(\mathsf{tr}_{lpo}(\alpha, \psi'), \mathsf{true}, S) = \mathsf{next}(\alpha, \psi, S) \setminus \{x \in N_\alpha \mid \varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha\}$$

Proof. We show the inclusion in both directions. Take any $x \in \bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathsf{next}(\mathsf{tr}_{lpo}(\alpha, \psi'), \mathsf{true}, S)$, so there is a $\psi' \in \mathsf{br}_\alpha$ such that $\psi' \Rightarrow \psi$ and $x \in N_{\mathsf{tr}_{lpo}(\alpha,\psi')} \setminus S$ such that $x{\downarrow}_{\mathsf{tr}_{lpo}(\alpha,\psi')} \subseteq S$. From $x \in N_{\mathsf{tr}_{lpo}(\alpha,\psi')} \setminus S$, we get that $x \in N_\alpha \setminus S$ and $\psi' \Rightarrow \varphi_\alpha(x)$. We now show that $x{\downarrow}_\alpha \subseteq S$. Take any $y <_\alpha x$, so we know that $\psi' \Rightarrow \varphi_\alpha(x) \Rightarrow \varphi_\alpha(y)$, which means that $y \in N_{\mathsf{tr}_{lpo}(\alpha,\psi')}$, and therefore $y <_{\mathsf{tr}_{lpo}(\alpha,\psi')} x$, so it must be that $y \in S$. From $\psi' \Rightarrow \psi$ and $\psi' \Rightarrow \varphi_\alpha(x)$, we know that $\varphi_\alpha(x) \wedge \psi$ is satisfiable. Since $\psi \in \mathsf{valid}_\alpha(S)$, then ψ must assign a truth value to every test in S, and since we just showed that $x{\downarrow}_\alpha \subseteq S$, then $\mathsf{vars}(\varphi_\alpha(x)) \subseteq S$, and so we must have that $\psi' \Rightarrow \varphi_\alpha(x)$. Therefore, we have established all the conditions of $x \in \mathsf{next}(\alpha, \psi, S)$. Clearly also $\varphi_\alpha(x) \not\Rightarrow \mathsf{stuck}_\alpha$ since $\psi' \Rightarrow \varphi_\alpha(x)$ and $\psi' \Rightarrow \neg\mathsf{stuck}_\alpha$.

Now take any $x \in \mathsf{next}(\alpha, \psi, S) \setminus \{x \in N_\alpha \mid \varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha\}$, so $x \in N_\alpha \setminus S$, $\psi \Rightarrow \varphi_\alpha(x)$, $x{\downarrow}_\alpha \subseteq S$, and $\varphi_\alpha(x) \not\Rightarrow \mathsf{stuck}_\alpha$. Since $\varphi_\alpha(x) \not\Rightarrow \mathsf{stuck}_\alpha$, then $x \in \mathsf{ext}_\alpha$, and so there is clearly a $\psi' \in \mathsf{br}_\alpha$ such that $\psi' \Rightarrow \varphi_\alpha(x)$. This means that $x \in N_{\mathsf{tr}_{lpo}(\alpha,\psi')} \setminus S$. Since by definition $\varphi_{\mathsf{tr}_{lpo}(\alpha,\psi')}(x) = \mathsf{true}$, then clearly $\mathsf{true} \Rightarrow \varphi_{\mathsf{tr}_{lpo}(\alpha,\psi')}(x)$. Finally, we will show that $x{\downarrow}_{\mathsf{tr}_{lpo}(\alpha,\psi')} \subseteq S$. Take any $y <_{\mathsf{tr}_{lpo}(\alpha,\psi')} x$, this means that $y <_\alpha x$, so clearly $y \in S$. Therefore, we have that $x \in \mathsf{next}(\mathsf{tr}_{lpo}(\alpha, \psi'), \mathsf{true}, S) \subseteq \bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathsf{next}(\mathsf{tr}_{lpo}(\alpha, \psi'), \mathsf{true}, S)$. □

Lemma G.13. *If $S \subseteq \mathsf{nBot}_\alpha$, $\psi \in \mathsf{valid}_\alpha(S)$ and $\psi \Rightarrow \mathsf{stuck}_\alpha$, then $\mathcal{L}_{lpo}(\alpha, \psi, S) = \emptyset$.*

Proof. The proof is by induction on the size of the set $\mathsf{next}^*(\alpha, \psi, S)$. If $\mathsf{next}^*(\alpha, \psi, S) = \emptyset$, then by Lemma F.3 we know that $\psi \in \mathsf{br}_\alpha$, but this is impossible since $\psi \Rightarrow \mathsf{stuck}_\alpha$. Therefore, it cannot be the case that $\mathsf{next}^*(\alpha, \psi, S) = \emptyset$.

Now suppose that $\mathsf{next}^*(\alpha, \psi, S) \neq \emptyset$. By Lemma F.4, we know that $\mathsf{next}(\alpha, \psi, S) \neq \emptyset$. So, we have:

$$\mathcal{L}_{lpo}(\alpha, \psi, S) = \bigcup_{x \in \mathsf{next}(\alpha,\psi,S)} \begin{cases} \mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in act \\ \mathcal{L}_{lpo}(\alpha, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!]_{test}(s)|\!), S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) \in test \\ \emptyset & \text{if } \lambda_\alpha(x) = \bot \\ \mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) = \mathsf{fork} \end{cases}$$

By the induction hypothesis in the first, second, and fourth cases, we get:

$$\begin{aligned} &= \bigcup_{x \in \mathsf{next}(\alpha,\psi,S)} \begin{cases} \emptyset & \text{if } \lambda_\alpha(x) \in act \\ \emptyset & \text{if } \lambda_\alpha(x) \in test \\ \emptyset & \text{if } \lambda_\alpha(x) = \bot \\ \emptyset & \text{if } \lambda_\alpha(x) = \mathsf{fork} \end{cases} \\ &= \bigcup_{x \in \mathsf{next}(\alpha,\psi,S)} \emptyset = \emptyset \end{aligned}$$

□

Lemma G.14. *For any binary branching $\alpha \in lpo_{\mathsf{fin}}(label)$, $S \subseteq \mathsf{nBot}_\alpha$, and $\psi \in \mathsf{valid}_\alpha(S)$:*

$$\bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathcal{L}_{lpo}(\mathsf{tr}_{lpo}(\alpha, \psi'), \mathsf{true}, S)(s) = \mathcal{L}_{lpo}(\alpha, \psi, S)(s)$$

Proof. The proof is by induction on the size of the set $\mathsf{next}^*(\alpha, \psi, S)$. If $\mathsf{next}^*(\alpha, \psi, S) = \emptyset$, then clearly $\mathsf{next}(\alpha, \psi, S) = \emptyset$ and by Lemma F.3, we know that $\psi \in \mathsf{br}_\alpha$. That means that the union on the left side is only over a single term, generated from ψ. Now, $\mathsf{tr}_{lpo}(\alpha, \psi)$ contains all those nodes x such that $\psi \Rightarrow \varphi_\alpha(x)$, but we know from $\mathsf{next}^*(\alpha, \psi, S) = \emptyset$ that all such nodes are already in S, therefore by Lemma G.12, $\mathsf{next}(\mathsf{tr}_{lpo}(\alpha, \psi), \mathsf{true}, S) = \emptyset$ too. This gives us

$$\bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathcal{L}_{lpo}(\mathsf{tr}_{lpo}(\alpha, \psi'), \mathsf{true}, S)(s) = \mathcal{L}_{lpo}(\mathsf{tr}_{lpo}(\alpha, \psi), \mathsf{true}, S)(s) = \{s\} = \mathcal{L}_{lpo}(\alpha, \psi, S)(s)$$

Now, assume that $\mathsf{next}^*(\alpha, \psi, S) \neq \emptyset$, so by Lemma F.4, then $\mathsf{next}(\alpha, \psi, S) \neq \emptyset$.

Suppose that $\varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha$ for some $x \in \mathsf{next}(\alpha, \psi, S)$. This means that $\psi \Rightarrow \varphi_\alpha(x) \Rightarrow \mathsf{stuck}_\alpha$, so clearly there can be no $\psi \in \mathsf{br}_\alpha$ such that $\psi' \Rightarrow \psi$, so $\bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathcal{L}_{lpo}(\mathsf{tr}_{lpo}(\alpha, \psi'), \mathsf{true}, S)(s) = \emptyset$. In addition, by Lemma G.13, $\mathcal{L}_{lpo}(\alpha, \psi, S)(s) = \emptyset$ too.

So, we are left with the case where $\varphi_\alpha(x) \not\Rightarrow \mathsf{stuck}_\alpha$ for all $x \in \mathsf{next}(\alpha, \psi, S)$. Let $\alpha_{\psi'} = \mathsf{tr}_{lpo}(\alpha, \psi')$. We now have:

$$\begin{aligned}
&\bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S) \\
&= \bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \bigcup_{x \in \mathsf{next}(\alpha_{\psi'}, \mathsf{true}, S)} \begin{cases}
\mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})^\dagger(\left[\!\left[\lambda_{\alpha_{\psi'}}(x)\right]\!\right]_{act}(s)) & \text{if } \lambda_{\alpha_{\psi'}}(x) \in act_{\mathsf{PL}} \\
\mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true} \wedge (\!|x = \left[\!\left[\lambda_{\alpha_{\psi'}}(x)\right]\!\right]_{test}(s)|\!), S \cup \{x\})(s) & \text{if } \lambda_{\alpha_{\psi'}}(x) \in test_{\mathsf{PL}} \\
\emptyset & \text{if } \lambda_{\alpha_{\psi'}}(x) = \bot \\
\mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})(s) & \text{if } \lambda_{\alpha_{\psi'}}(x) = \mathsf{fork}
\end{cases}
\end{aligned}$$

Since $\alpha_{\psi'}$ has no test or $\bot$ nodes, we can remove those two cases. In addition, we expand the case for actions into cases for whether or not the action is assume.

$$= \bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \bigcup_{x \in \mathsf{next}(\alpha_{\psi'}, \mathsf{true}, S)} \begin{cases}
\mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in act \\
\mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})^\dagger([\![\mathsf{assume}\ \lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in test \text{ and } \psi' \Rightarrow x \\
\mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})^\dagger([\![\mathsf{assume}\ \neg\lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in test \text{ and } \psi' \Rightarrow \neg x \\
\mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) = \mathsf{fork}
\end{cases}$$

By Lemma G.12, and the fact that $\psi \Rightarrow \varphi_\alpha(x)$ for all $x \in \mathsf{next}(\alpha, \psi, S)$, we can rearrange the unions and move the inner one inside of the conditional.

$$= \bigcup_{x \in \mathsf{next}(\alpha, \psi, S)} \begin{cases}
\bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in act \\
\bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi \wedge x} \mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{test}(s)) & \text{if } \lambda_\alpha(x) \in test \\
\quad \cup \bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi \wedge \neg x} \mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})^\dagger([\![\neg\lambda_\alpha(x)]\!]_{test}(s)) & \\
\bigcup_{\psi' \in \mathsf{br}_\alpha : \psi' \Rightarrow \psi} \mathcal{L}_{lpo}(\alpha_{\psi'}, \mathsf{true}, S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) = \mathsf{fork}
\end{cases}$$

By the induction hypothesis.

$$= \bigcup_{x \in \mathsf{next}(\alpha, \psi, S)} \begin{cases}
\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in act \\
\mathcal{L}_{lpo}(\alpha, \psi \wedge x, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{test}(s)) & \text{if } \lambda_\alpha(x) \in test \\
\quad \cup \mathcal{L}_{lpo}(\alpha, \psi \wedge \neg x, S \cup \{x\})^\dagger([\![\neg\lambda_\alpha(x)]\!]_{test}(s)) & \\
\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) = \mathsf{fork}
\end{cases}$$

Exactly one of $[\![\lambda_\alpha(x)]\!]_{test}(s)$ and $[\![\neg\lambda_\alpha(x)]\!]_{test}(s)$ will evaluate to $\{s\}$ and the other will evaluate to $\emptyset$, so we can consolidate the terms.

$$\begin{aligned}
&= \bigcup_{x \in \mathsf{next}(\alpha, \psi, S)} \begin{cases}
\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger([\![\lambda_\alpha(x)]\!]_{act}(s)) & \text{if } \lambda_\alpha(x) \in act \\
\mathcal{L}_{lpo}(\alpha, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!]_{test}(s)|\!), S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) \in test \\
\mathcal{L}_{lpo}(\alpha, \psi, S \cup \{x\})(s) & \text{if } \lambda_\alpha(x) = \mathsf{fork}
\end{cases} \\
&= \mathcal{L}_{lpo}(\alpha, \psi, S)
\end{aligned}$$

□

Lemma G.15. *For any $\boldsymbol{\alpha} \in pom$ and $s \in S$, $\mathcal{L}(\boldsymbol{\alpha})(s) = \mathcal{L}_{\mathsf{PL}}(\mathsf{tr}(\boldsymbol{\alpha}))(s)$.*

Proof.

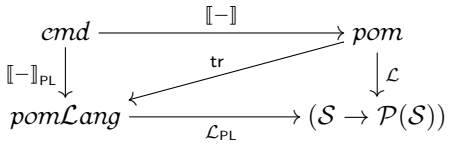
$$\begin{aligned}
\mathcal{L}(\boldsymbol{\alpha})(s) &= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha}')(s) \\
&= \bigcup_{[\alpha] \ll \boldsymbol{\alpha}} \mathcal{L}_{lpo}(\alpha, \mathsf{true}, \emptyset)(s)
\end{aligned}$$

$$
\begin{aligned}
&= \bigcup_{[\alpha] \ll \boldsymbol{\alpha}} \bigcup_{\psi \in \mathsf{br}_\alpha} \mathcal{L}_{lpo}(\mathsf{tr}_{lpo}(\alpha, \psi), \mathsf{true}, \emptyset)(s) && \text{By Lemma G.14} \\
&= \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \bigcup_{\boldsymbol{\beta} \in \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}')} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\beta})(s) \\
&= \bigcup_{\boldsymbol{\beta} \in \bigcup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \mathsf{tr}_{\mathsf{fin}}(\boldsymbol{\alpha}')} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\beta})(s) \\
&= \bigcup_{\boldsymbol{\beta} \in \mathsf{tr}(\boldsymbol{\alpha})} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\beta})(s) \\
&= \mathcal{L}_{\mathsf{PL}}(\mathsf{tr}(\boldsymbol{\alpha}))
\end{aligned}
$$

□

E. Main Theorem

Theorem 1 (Equivalence of Semantics)**.** *The following diagram commutes:*



Proof. It suffices to show that $[\![-]\!]_{\mathsf{PL}} = \mathsf{tr} \circ [\![-]\!]$ and $\mathcal{L}_{\mathsf{PL}} \circ \mathsf{tr} = \mathcal{L}$, which follows immediately from Lemmas G.11 and G.15. We then have:

$$
\begin{aligned}
\mathcal{L}_{\mathsf{PL}} \circ [\![-]\!]_{\mathsf{PL}} &= \mathcal{L}_{\mathsf{PL}} \circ (\mathsf{tr} \circ [\![-]\!]) \\
&= (\mathcal{L}_{\mathsf{PL}} \circ \mathsf{tr}) \circ [\![-]\!] \\
&= \mathcal{L} \circ [\![-]\!]
\end{aligned}
$$

□