Pomsets with Boxes: Protection, Separation, and Locality in Concurrent Kleene Algebra

Paul Brunet
University College London, UK
paul.brunet-zamansky.fr
paul@brunet-zamansky.fr

David Pym
University College London, UK
www.cantab.net/users/david.pym/
d.pym@ucl.ac.uk

Abstract

Concurrent Kleene Algebra is an elegant tool for equational reasoning about concurrent programs. An important feature of concurrent programs that is missing from CKA is the ability to restrict legal interleavings. To remedy this we extend the standard model of CKA, namely pomsets, with a new feature, called boxes, which can specify that part of the system is protected from outside interference. We study the algebraic properties of this new model. Another drawback of CKA is that the language used for expressing properties of programs is the same as that which is used to express programs themselves. This is often too restrictive for practical purposes. We provide a logic, 'pomset logic', that is an assertion language for specifying such properties, and which is interpreted on pomsets with boxes. In contrast with other approaches, this logic is not state-based, but rather characterizes the runtime behaviour of a program. We develop the basic metatheory for the relationship between pomset logic and CKA, including frame rules to support local reasoning, and illustrate this relationship with simple examples.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Semantics and reasoning; Theory of computation → Separation logic

Keywords and phrases Concurrent Kleene Algebra, Pomsets, Atomicity, Semantics, Separation, Local reasoning, Bunched logic, Frame rules.

Related Version This preprint is based on a paper to appear in FSCD 2020.

Supplement Material A formalization of Section 2 in Coq is available on github: `AtomicCKA`.

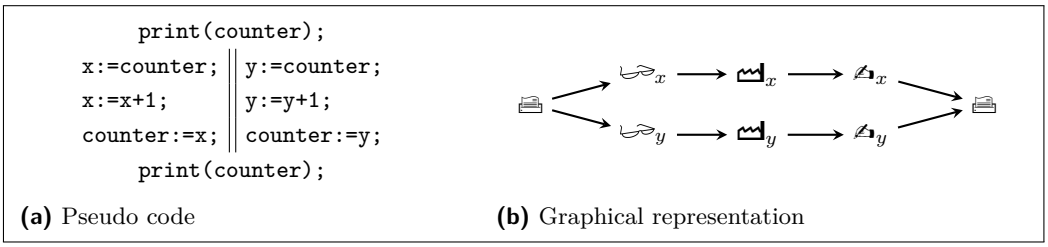
Funding This work has been supported by UK EPSRC Research Grant EP/R006865/1: Interface Reasoning for Interacting Systems (IRIS).

Acknowledgements The authors are grateful to their colleagues at UCL and within IRIS project for their interest. We also thank the anonymous referees for their comments and suggestions.

1 Introduction

Concurrent Kleene Algebra (CKA) [11, 14, 15, 4] is an elegant tool for equational reasoning about concurrent programs. Its semantics is given in terms of pomsets languages; that is, sets of pomsets. Pomsets [8], also known as partial words [9], are a well-known model of concurrent behaviour, traditionally associated with runs in Petri nets [13, 4].

However, in CKA the language used for expressing properties of programs is the same as that which is used to express programs themselves. It is clear that this situation is not ideal for specifying and reasoning about properties of programs. Any language specifiable in CKA terms has bounded width (i.e., the number of processes in parallel; the size of a maximal independent set) and bounded depth (i.e., the number of alternations of parallel



(a) Pseudo code **(b)** Graphical representation

Figure 1 Distributed counter

and sequential compositions)[17]. However, many properties of interest — for example, safety properties — are satisfied by sets of pomsets with both unbounded width and depth.

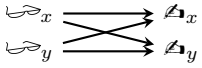
In this paper, we provide a logic, 'pomset logic', that is an assertion language for specifying such properties. We develop the basic metatheory for the relationship between pomset logic and CKA and illustrate this relationship with simple examples. In addition, to the usual classical or intuitionistic connectives — both are possible — the logic includes connectives that characterize both sequential and parallel composition.

In addition, we note that CKA allows programs with every possible interleaving of parallel threads. However, to prove the correctness of such programs, some restrictions must be imposed on what are the legal interleavings. We provide a mechanism of 'boxes' for this purpose. Boxes identify protected parts of the system, so restricting the possible interleavings. From the outside, one may interact with the box as a whole, as if the program inside was atomic. On the other hand, it is not possible to interact with its individual components, as that would intuitively require opening the box. However, boxes can be nested, with this atomicity observation holding at each level. Pomset logic has context and box modalities that characterize this situation.

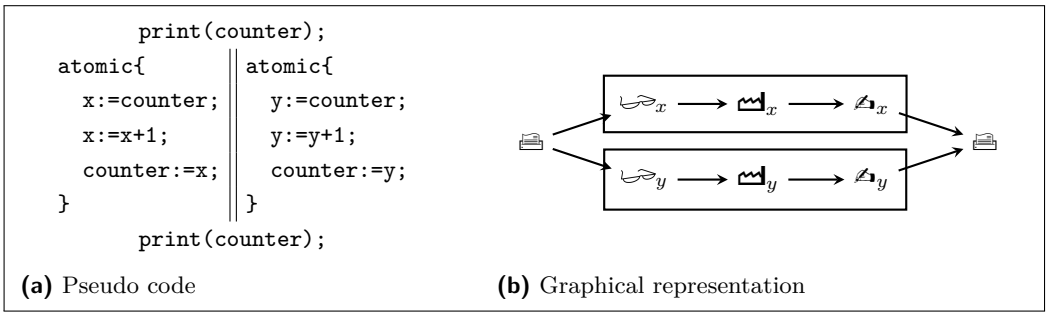
▶ Note. The term 'Pomset logic' has already been used in work by Retoré [25]. We feel that reusing it does not introduce ambiguity, since the two frameworks arise in different contexts.

▶ **Example 1** (Running example: a distributed counter)**.** We consider here a program where a counter is incremented in parallel by two processes. The intention is that the counter should be incremented twice, once by each process. However, to do so each process has to first load the contents of the counter, then compute the increment, and finally commit the result to memory. A naive implementation is presented in Figure 1a. Graphically, we represent the print instruction `print(counter)` by 🖶, the read instruction `x:=counter` by $\hookrightarrow_x$, the increment instruction `x:=x+1` by $\blacksquare_x$, and finally the write instruction `counter:=x` by $\blacktriangleleft_x$. We thus represent the previous program as displayed in Figure 1b.
This program does not comply with our intended semantics, since the following run is possible:

The result is that the counter has been incremented by one. We can identify a subset of instructions that indicate there is a fault: the problem is that both read instructions happened before both write instructions; i.e.,



To preclude this problematic behaviour, a simple solution is to make the sequence 'read;compute;write' *atomic*. This yields the program in Figure 2a. Diagrammatically, this



(a) Pseudo code **(b)** Graphical representation

Figure 2 Distributed counter with atomic increment

can be represented by drawing solid boxes around the `atomic{}` blocks, as shown in Figure 2b. This paper shows how to make these ideas formal.

In Section 2, we extend pomsets with a new construct for protection, namely boxes. We provide a syntax for specifying such pomsets and characterize precisely its expressivity. This enables us, for example, to correctly represent the program from Example 1. We present a sound and complete axiomatization of these terms, with operators for boxing, sequential and parallel composition, and non-deterministic choice, as well as the constants *abort* and *skip*.

In Section 3, we introduce pomset logic. This logic comes in both classical and intuitionistic variants. In addition to the usual classical or intuitionistic connectives, this logic includes connectives corresponding to each of sequential and parallel composition. These two classes of connectives are combined to give the overall logics, in the same way as the additives and multiplicatives of BI (bunched implications logic) [21, 1, 23]. Just as in BI and its associated separation logics [21, 12, 26], pomset logic has both classical and intuitionistic variants. It also includes modalities that characterize, respectively, protection, and locality. These correspondences are made precise by van Benthem–Hennessy–Milner-type theorems asserting that two programs are (operationally) equivalent iff they satisfy the same formulae. We obtain such correspondences for several variants of our framework. In contrast to Hennessy–Milner logic, however, pomset logic is a logic of *behaviours* rather than of states and transitions.

In Section 4, we investigate local reasoning principles for our logic of program behaviours. We showcase the possibilities of our framework on an example. We conclude by briefly discussing future work in Section 5.

2 Algebra of Pomsets with Boxes

In this section, we define our semantic model, and the corresponding syntax. We characterize the expressivity of the syntax, and axiomatize its equational theory.

Throughout this paper, we will use Σ to denote a given set of atomic actions.

2.1 Pomsets with boxes

2.1.1 Definitions and elementary properties

▶ **Definition 2** (Poset with boxes). *A* poset with boxes *is a tuple* $P := \langle \mathcal{E}_P, \leq_P, \lambda_P, \mathcal{B}_P \rangle$, *where* $\mathcal{E}_P$ *is a finite set of* events*;* $\leq_P \subseteq \mathcal{E}_P \times \mathcal{E}_P$ *is a partial order;* $\lambda_P : \mathcal{E}_P \to \Sigma$ *is a labelling function;* $\mathcal{B}_P \subseteq \mathcal{P}(\mathcal{E}_P)$ *is a set of* boxes*, such that* $\emptyset \notin \mathcal{B}_P$.

The partial order should be viewed as a set of necessary dependencies: in any legal scheduling of the pomset, these dependencies have to be satisfied. We therefore consider that a stronger

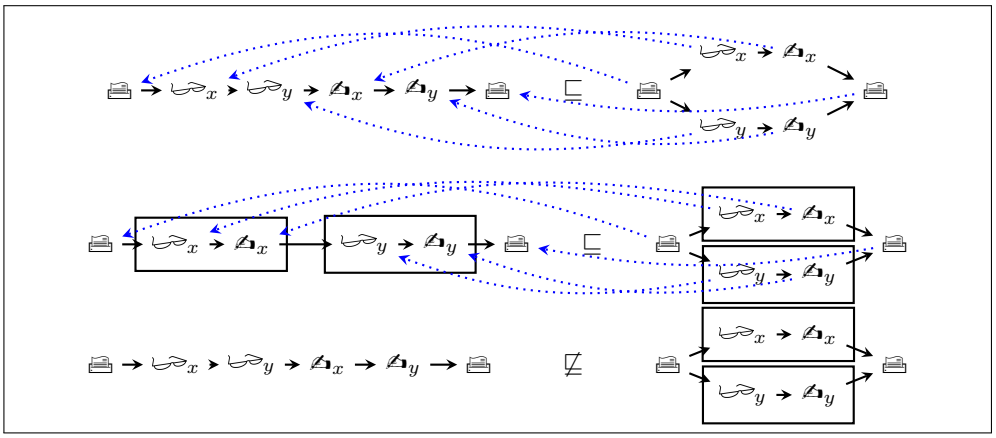


Figure 3 Poset subsumption

ordering — that is, one containing more pairs — yields a smaller pomset. The intuition is that the set of legal schedulings of the smaller pomset is contained in that of the larger one. The boxes are meant to further restrict the legal schedulings: no event from outside a box may be interleaved between the events inside the box. Subsequently, a pomset with more boxes is smaller than one with less boxes. This ordering between pomsets with boxes is formalized by the notion of homomorphism:

▶ **Definition 3** (Poset morphisms). *A* (poset with boxes) homomorphism *is a map between event-sets that is bijective, label respecting, order preserving, and box preserving. In other words, a map* $\phi : \mathcal{E}_P \to \mathcal{E}_Q$ *such that (i)* ϕ *is a bijection; (ii)* $\lambda_Q \circ \phi = \lambda_P$*; (iii)* $\phi(\leq_P) \subseteq \leq_Q$*; (iv)* $\phi(\mathcal{B}_P) \subseteq \mathcal{B}_Q$*. If in addition (iii) holds as an equality,* ϕ *is called* order-reflecting*. If on the other hand (iv) holds as an equality* ϕ *is* box-reflecting*. A homomorphism that is both order- and box-reflecting is a* (poset with boxes) isomorphism*.*

In Figure 3 are some examples and a non-example of subsumption between posets. We introduce some notations. $\mathbb{P}_\Sigma$ is the set of posets with boxes. If ϕ is a homomorphism from P to Q, we write $\phi : P \to Q$. If there exists such a homomorphism (respectively an isomorphism) from P to Q, we write $Q \sqsubseteq P$ (resp. $Q \cong P$).

▶ **Lemma 4.** *$\cong$ is an equivalence relation. $\sqsubseteq$ is a partial order with respect to $\cong$.*

▶ Remark 5. Note that the fact that $\sqsubseteq$ is antisymmetric with respect to $\cong$ relies on the finiteness of the posets considered here. For instance, consider the example is depicted in Figure 4. Formally, we fix some fixed symbol $a \in \Sigma$, and let P and Q be defined as follows:

$$P := \langle \mathbb{N} \times \{0,1,2\}, \leq_P, [_ \mapsto a], \emptyset \rangle \qquad \text{with } \langle n,i\rangle \leq_P \langle m,j\rangle := (n \leqslant m \wedge i = j < 2)$$
$$Q := \langle \mathbb{N} \times \{0,1\}, \leq_Q, [_ \mapsto a], \emptyset \rangle \qquad \text{with } \langle n,i\rangle \leq_Q \langle m,j\rangle := (n \leqslant m \wedge i = j = 0)\,.$$

One can plainly see that P and Q are not isomorphic, but there are indeed homomorphisms in both directions. Let us define the following functions:

$$\begin{aligned} &\phi : \mathcal{E}_P \to \mathcal{E}_Q \\ &\quad \langle n,0\rangle \mapsto \langle 2n,0\rangle \\ &\quad \langle n,1\rangle \mapsto \langle 2n+1,0\rangle \\ &\quad \langle n,2\rangle \mapsto \langle n,1\rangle \end{aligned} \qquad \begin{aligned} &\psi : \mathcal{E}_Q \to \mathcal{E}_P \\ &\quad \langle n,0\rangle \mapsto n,0 \\ &\quad \langle 2n,1\rangle \mapsto \langle n,1\rangle \\ &\quad \langle 2n+1,1\rangle \mapsto \langle n,2\rangle\,. \end{aligned}$$

One may easily check that ϕ and ψ are both poset-homomorphisms.

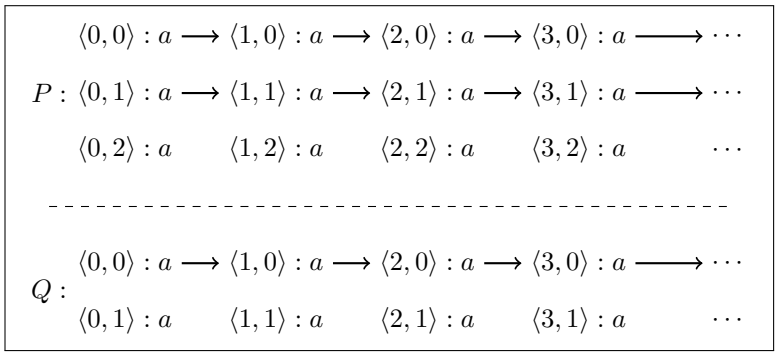


Figure 4 Example of mutual homomorphic posets that are not isomorphic

▶ **Definition 6** (Pomsets with boxes)**.** Pomsets with boxes *are equivalence classes of* $\cong$*. The set* $\mathbf{Pom}_\Sigma$ *of pomsets with boxes is defined as* $\mathbb{P}_\Sigma/_\cong$*.*

We now define some elementary poset-building operations.

▶ **Definition 7** (Constants)**.** *Given a symbol* $a \in \Sigma$*, the* atomic poset *associated with* a *is defined as* $\mathbb{a} := \langle \{0\}, [0 \mapsto a], Id_{\{0\}}, \emptyset \rangle \in \mathbb{P}_\Sigma$*. The empty poset is defined as* $\mathbb{e} := \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle \in \mathbb{P}_\Sigma$*.*

▶ Remark 8. For any poset $P \in \mathbb{P}_\Sigma$, $P \sqsubseteq \mathbb{e} \Leftrightarrow P \sqsupseteq \mathbb{e} \Leftrightarrow P \cong \mathbb{e}$. This is because each of those relations imply there is a bijection between the events of P and $\mathcal{E}_\mathbb{e} = \emptyset$. So we know that P has no events, and since boxes cannot be empty, P has no boxes either. Hence $P \cong \mathbb{e}$.

▶ **Definition 9** (Compositions)**.** *Let* P, Q *be two posets with boxes. The sequential composition* $P \otimes Q$ *and parallel composition* $P \oplus Q$ *are defined by:*

$$P \otimes Q := \langle \mathcal{E}_P \uplus \mathcal{E}_Q, \leq_P \cup \leq_Q \cup (\mathcal{E}_P \times \mathcal{E}_Q), \lambda_P \sqcup \lambda_Q, \mathcal{B}_P \cup \mathcal{B}_Q \rangle$$
$$P \oplus Q := \langle \mathcal{E}_P \uplus \mathcal{E}_Q, \leq_P \cup \leq_Q, \lambda_P \sqcup \lambda_Q, \mathcal{B}_P \cup \mathcal{B}_Q \rangle,$$

where the symbol $\sqcup$ *denotes the union of two functions; that is, given* $f : A \to C$ *and* $g : B \to C$*, the function* $f \sqcup g : A \uplus B \to C$ *associates* $f(a)$ *to* $a \in A$ *and* $g(b)$ *to* $b \in B$*.*

Intuitively, $P \oplus Q$ consists of disjoint copies of P and Q side by side. $P \otimes Q$ also contains disjoint copies of P and Q, but also orders every event in P before any event in Q.

▶ **Definition 10** (Boxing)**.** *Given a poset* P *its* boxing *is denoted by* $[P]$ *and is defined by:* $[P] := \langle \mathcal{E}_P, \leq_P, \lambda_P, \mathcal{B}_P \cup \{\mathcal{E}_P\} \rangle$*.*

Boxing a pomset simply amounts to drawing a box around it.

In our running example, the pattern of interest is a subset of the events of the whole run. To capture this, we define the restriction of a poset to a subset of its events.

▶ **Definition 11** (Restriction, sub-poset)**.** *For a given set of events* $A \subseteq \mathcal{E}_P$*, we define the* restriction of P to A *as* $P\lfloor_A := \langle A, \leq_P \cap (A \times A), \lambda_P \lfloor_A, \mathcal{B}_P \cap \mathcal{P}(A) \rangle$*. We say that* P *is a* sub-poset *of* Q*, and write* $P \trianglelefteq Q$*, if there is a set* $A \subseteq \mathcal{E}_Q$ *such that* $P \cong Q\lfloor_A$*.*

Given a poset P, a set of events $A \subseteq \mathcal{E}_P$ is called:

- **non-trivial** if $A \notin \{\emptyset, \mathcal{E}\}$.
- **nested** if for any box $\beta \in \mathcal{B}_P$ either $\beta \subseteq A$ or $A \cap \beta = \emptyset$;
- **prefix** if for any $e \in A$ and $f \notin A$ we have $e \leq_P f$; and

- **isolated** if for any $e \in A$ and $f \notin A$ we have $e \not\leq_P f$ and $f \not\leq_P e$.

These properties characterize sub-posets of particular interest to P. This is made explicit in the following observation:

▶ **Fact 12.** *Given a poset P and a set of events $A \subseteq \mathcal{E}_P$:*
(i) *A is prefix and nested iff $P \cong P\restriction_A \otimes P\restriction_{\overline{A}}$;*
(ii) *A is isolated and nested iff $P \cong P\restriction_A \oplus P\restriction_{\overline{A}}$.*
(Here $\overline{A}$ denotes the complement of A relative to $\mathcal{E}_P$; that is, $\overline{A} := \mathcal{E}_P \setminus A$.)

This fact is very useful as a way to 'reverse-engineer' how a poset was built.

2.1.2 Series–parallel pomsets

In the sequel, we will often restrict our attention to series–parallel pomsets. These are of particular interest since they are defined as those pomsets that can be generated from constants using the operators we have defined.

▶ **Definition 13** (Pomset terms, SP-Pomsets). *A* (pomset) term *is a syntactic expression generated from the following grammar: $s, t \in \mathtt{SP}_\Sigma ::= 1 \mid a \mid s\,;t \mid s \parallel t \mid [s]$. By convention $;$ binds tighter than $\parallel$. A term is interpreted as a poset as follows:*

$$[\![a]\!] := \mathfrak{a} \qquad [\![1]\!] := \mathfrak{e} \qquad [\![\,[s]\,]\!] := [\,[\![s]\!]\,]$$
$$[\![s\,;t]\!] := [\![s]\!] \otimes [\![t]\!] \qquad [\![s \parallel t]\!] := [\![s]\!] \oplus [\![t]\!]\,.$$

A pomset $[P]_\cong$ is called series–parallel *(or SP for short) if it is the interpretation of some term; that is, $\exists s \in \mathtt{SP}_\Sigma : [\![s]\!] \cong P$.*

▶ **Example 14.** The program in Figure 1 of the running example corresponds to

$$[\![\,;(\,;\,;\,\parallel\,;\,;\,);]\!]\,.$$

The corrected program, from Figure 2, corresponds to

$$[\![\,;([\,;\,;]\parallel[\,;\,;])\,;]\!]\,.$$

Finally, the problematic pattern we identified may be represented as $[\![(\,\parallel\,)\,;(\,\parallel\,)]\!]$.

Series–parallel pomsets with boxes may also be defined by excluded patterns, in the same style as the characterization of series–parallel pomsets [27, 9, 8]. More precisely, one can prove that a pomset $[P]_\cong$ is series–parallel iff and only if it does not contain any of the patterns in Figure 5. Formally,

▶ **Theorem 15.** *A pomset $[P]_\cong$ is series-parallel iff and only if it none of the following properties are satisfied:*
$\mathbf{P_1}$ **:** $\exists e_1, e_2, e_3, e_4 \in \mathcal{E}_P : e_1 \leq_P e_3 \wedge e_2 \leq_P e_3 \wedge e_2 \leq_P e_4 \wedge e_1 \not\leq_P e_4 \wedge e_2 \not\leq_P e_1 \wedge e_4 \not\leq_P e_3$
$\mathbf{P_2}$ **:** $\exists e_1, e_2, e_3 \in \mathcal{E}_P, \exists A, B \in \mathcal{B}_P : e_1 \in A \setminus B \wedge e_2 \in A \cap B \wedge e_3 \in B \setminus A$
$\mathbf{P_3}$ **:** $\exists e_1, e_2, e_3 \in \mathcal{E}_P, \exists A \in \mathcal{B}_P : e_1 \notin A \wedge e_2, e_3 \in A \wedge e_1 \leq_P e_2 \wedge e_1 \not\leq_P e_3$
$\mathbf{P_4}$ **:** $\exists e_1, e_2, e_3 \in \mathcal{E}_P, \exists A \in \mathcal{B}_P : e_1 \notin A \wedge e_2, e_3 \in A \wedge e_2 \leq_P e_1 \wedge e_3 \not\leq_P e_1$.

Before we discuss the proof of this result, we make a number of comments.

The four properties in Theorem 15 are invariant under isomorphism, since they only use the ordering between events and the membership of events to boxes. This is consistent with SP being a property of pomsets, not posets.

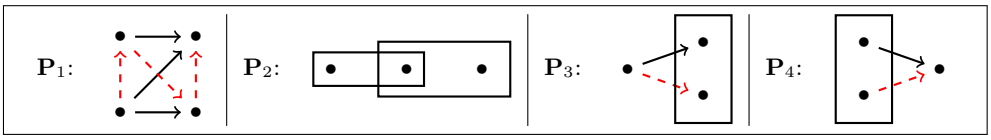


Figure 5 Forbidden patterns of SP-pomsets: dashed arrows (in red) are negated.

Pattern $\mathbf{P}_1$ is known as N, and is the forbidden pattern of series-parallel pomsets (without boxes), as proved by Gischer [8]. Pattern $\mathbf{P}_2$ indicates that the boxes in an SP-pomset are well nested: two boxes are either disjoint, or one is contained in the other. Patterns $\mathbf{P}_3$ and $\mathbf{P}_4$ reflect that an event is outside of a box cannot distinguish the events inside by the order: it is either smaller than all of them, larger than all of them, or incomparable with them.

Together, $\mathbf{P}_2$, $\mathbf{P}_3$, and $\mathbf{P}_4$ provide an alternative view of pomsets with boxes: one may see them as *hyper-pomsets*; that is, pomsets in which some events (the boxes) can be labelled with non-empty pomsets (the contents of the boxes). However, it seems that for our purposes the definition we provide is more convenient. In particular, the definition of hyper-pomset homomorphism is more involved.

Two auxiliary results on sub-posets, which we collect in the following lemma, will be useful in this proof.

▶ **Lemma 16.** *Let P be a poset with at least two events, such that $\mathcal{E}_P \notin \mathcal{B}_P$. Then:*

(i) *if P contains a non-trivial prefix set, it contains one that is nested;*

(ii) *if P contains a non-trivial isolated set, it contains one that is nested.*

Proof. **(i)** Assume there exists non-trivial prefix set. We pick a minimal one; that is, a non-trivial prefix set A such that for any non-trivial prefix B, if $B \subseteq A$, then $B = A$ (this is always possible since $\subseteq$ is a well-founded partial order on finite sets). If A is nested, then A satisfies our requirements. Otherwise, there is a box that is not contained in A while intersecting A. Since P does not contain the pattern $\mathbf{P}_2$, we know that we may pick a maximal such box β. This means that we know the following:

$$\beta \in \mathcal{B}_P \qquad (\forall \alpha \in \mathcal{B}_P,\ \beta \subseteq \alpha \Rightarrow \beta = \alpha) \qquad \exists e_1 \in \beta \cap A \qquad \exists e_2 \in \beta \setminus A.$$

First, we show that $A \subseteq \beta$. Consider the set $A' := A \setminus \beta$. Clearly $A' \subsetneq A$ (since $e_1 \in A \setminus A'$). We may also show that A' is prefix. Let $e \in A'$ and $f \notin A'$. There are two cases:

- either $f \notin A$, then since A is prefix and $e \in A' \subseteq A$ we have $e \leq_P f$;
- or $f \in A \cap \beta$. In this case, we use the fact that P does not have pattern $\mathbf{P}_3$: we know that $e_2 \in \beta \setminus A$, so since $e \in A' \subseteq A$ we have $e \leq_P e_2$, and since $e \notin \beta$ and $e_2, f \in \beta$ we may conclude that $e \leq_P f$.

Therefore, A' is prefix and strictly contained in A. By minimality of A, A' has to be trivial. Since A is non-trivial this means that $A' = \emptyset$, hence that $A \subseteq \beta$.

Now we know that $A \subseteq \beta$. Because we know that β is not empty, and that $\mathcal{E}_P \notin \mathcal{B}_P$, we deduce that β is non-trivial. Since P does not contain pattern $\mathbf{P}_2$, and by maximality of β, we know that β is nested. We now conclude by showing that β is in fact prefix. Let $e \in \beta$, and $f \notin \beta$. Since $A \subseteq \beta$ we get that $f \notin A$. By the prefix property of A we get $e_1 \leq_P f$, and since $e_1, e \in \beta$ and $f \notin \beta$, by the absence of pattern $\mathbf{P}_4$ we get that $e \leq_P f$.

(ii) This proceeds in a similar manner. We pick a minimal non-trivial isolated set A, and try to find a maximal box β such that $\beta \cap A \neq \emptyset$ and $\beta \setminus A \neq \emptyset$. If no such box exists, A

is already nested. If we do find such a box, we first show that A has to be contained in β. Then we use this to show that β is a non-trivial nested isolated set.

◀

Proof of Theorem 15. By a simple induction on terms, we can easily show that SP-posets avoid all four forbidden patterns. The more challenging direction is the converse: given a poset P that does not contain any of the forbidden patterns, can we build a term $s \in \mathtt{SP}_\Sigma$ such that $P \cong [\![s]\!]$. We construct this by induction on the size of P, defined as number of boxes plus the number of events. Notice that if a poset does not contain a pattern, then nor does any of its sub-posets.

If P has at most one event, then the following property holds:

- if P has no events, then $P \cong [\![1]\!]$;
- if P has a single event e, let $a = \lambda_P(e)$; we know that

$$\mathcal{B}_P \subseteq \{\beta \subseteq \{e\} \mid \beta \neq \emptyset\} = \{\{e\}\}\,;$$

  - if $\mathcal{B}_P = \emptyset$ then $P \cong \mathbb{o} = [\![a]\!]$;
  - if $\mathcal{B}_P = \{\{e\}\}$ then $P \cong [\mathbb{o}] = [\![[a]]\!]$.

If $\mathcal{E}_P \in \mathcal{B}_P$, then let P' be the poset obtained by removing the box $\mathcal{E}_P$. The size of P' is strictly smaller than that of P, and $P \cong [P']$. By induction, we get a term s such that $[\![s]\!] \cong P'$, so $P \cong [\![[s]]\!]$.

Consider now a pomset P with at least two events, and such that $\mathcal{E}_P \notin \mathcal{B}_P$. As a corollary of Gischer's characterization theorem [8, Theorem 3.1], we know that since P is N-free (i.e., does not contain $\mathbf{P}_1$) and contains at least two events, it contains either a non-trivial prefix set or a non-trivial isolated set.

If P contains a non-trivial, prefix set A, then by Lemma 16 P contains a non-trivial, *nested*, prefix set A'. By Fact 12, this means that $P \cong P{\restriction_A} \otimes P{\restriction_{\overline{A}}}$. We may thus conclude by induction.

The case in which P contains a non-trivial, isolated set A is handled similarly. ◀

2.2 Sets of posets

We now lift our operations and relations to sets of posets. This allows us to enrich our syntax with a non-deterministic choice operator.

▶ **Definition 17** (Orderings on sets of posets). *Let $A, B \subseteq \mathbb{P}_\Sigma$, we define the following:*
Isomorphic inclusion *: $A \lesssim B$ iff $\forall P \in A$, $\exists Q \in B$ such that $P \cong Q$*
Isomorphic equivalence *: $A \cong B$ iff $A \lesssim B \wedge B \lesssim A$*
Subsumption *: $A \sqsubseteq B$ iff $\forall P \in A$, $\exists Q \in B$ such that $P \sqsubseteq Q$.*

▶ Remark 18. Isomorphic inclusion and subsumption are partial orders with respect to isomorphic equivalence, which is an equivalence relation.

▶ **Definition 19** (Operations on sets of posets). *We will use the set-theoretic union of sets of posets, as well as the pointwise liftings of the two products of posets and the boxing operators:*

$$A \otimes B := \{P \otimes Q \mid \langle P, Q\rangle \in A \times B\} \qquad [A] := \{[P] \mid P \in A\}$$
$$A \oplus B := \{P \oplus Q \mid \langle P, Q\rangle \in A \times B\}.$$

▶ **Definition 20** (Closure of a set of posets). *The* (downwards) closure *of a set of posets S is the smallest set containing S that is downwards closed with respect to the subsumption order; that is, $S{\downarrow} := \{P \in \mathbb{P}_\Sigma \mid \exists Q \in S :\ P \sqsubseteq Q\}$. Similarly, the* upwards closure *of S is defined as: $S{\uparrow} := \{P \in \mathbb{P}_\Sigma \mid \exists Q \in S :\ P \sqsupseteq Q\}$.*

Table 1 Equational and inequational logic

$$\frac{e = f \in A}{A \vdash e = f} \qquad A \vdash e = e \qquad \frac{A \vdash e = f}{A \vdash f = e} \qquad \frac{A \vdash e = f \qquad A \vdash f = g}{A \vdash e = g}$$

$$\sigma, \tau : \Sigma \to \mathtt{T}_\Sigma, \ \frac{\forall a \in \Sigma,\ A \vdash \sigma(a) = \tau(a)}{A \vdash \hat{\sigma}(e) = \hat{\tau}(e)}$$

$$\frac{e = f \in A}{A \vdash e \leq f} \qquad \frac{f = e \in A}{A \vdash e \leq f} \qquad \frac{e \leq f \in A}{A \vdash e \leq f} \qquad A \vdash e \leq e \qquad \frac{A \vdash e \leq f \qquad A \vdash f \leq g}{A \vdash e \leq g}$$

$$\sigma, \tau : \Sigma \to \mathtt{T}_\Sigma, \ \frac{\forall a \in \Sigma,\ A \vdash \sigma(a) \leq \tau(a)}{A \vdash \hat{\sigma}(e) \leq \hat{\tau}(e)}$$

▶ Remark 21. $(_)\downarrow$ and $(_)\uparrow$ are Kuratowski closure operators [16]; i.e., they satisfy the following properties:

$$\emptyset\downarrow = \emptyset \qquad A \subseteq A\downarrow \qquad A\downarrow\downarrow = A\downarrow \qquad (A \cup B)\downarrow = A\downarrow \cup B\downarrow.$$

(And, similarly, for the upwards closure.) Using downwards-closures, we can express subsumption in terms of isomorphic inclusion:

$$A \sqsubseteq B \qquad \Leftrightarrow \qquad A \lesssim B\downarrow \qquad \Leftrightarrow \qquad A\downarrow \lesssim B\downarrow.$$

Similarly, the equivalence relation associated with $\sqsubseteq$, defined as the intersection of the relation and its converse, corresponds to the predicate $A\downarrow \cong B\downarrow$.

▶ **Definition 22.** *Terms are defined by the following grammar:*

$$e, f \in \mathtt{T}_\Sigma ::= 0 \mid 1 \mid a \mid e\,;f \mid e \parallel f \mid e + f \mid [e]\,.$$

Terms can be interpreted as finite sets of posets with boxes as follows:

$$[\![0]\!] := \emptyset \qquad [\![1]\!] := \{\mathfrak{e}\} \qquad [\![a]\!] := \{\mathfrak{a}\}$$

$$[\![\,[e]\,]\!] := [\,[\![e]\!]\,] \qquad [\![e\,;f]\!] := [\![e]\!] \otimes [\![f]\!] \qquad [\![e + f]\!] := [\![e]\!] \cup [\![f]\!] \qquad [\![e \parallel f]\!] := [\![e]\!] \oplus [\![f]\!]\,.$$

▶ Remark 23. Interpreted as a program, 0 represents failure: this is a program that aborts the whole execution. $+$, on the other hand, represents non-deterministic choice. It can be used to model conditional branching.

2.3 Axiomatic presentations of pomset algebra

We now introduce axioms to capture the various order and equivalence relations we introduced over posets and sets of posets. Given a set of axioms A (i.e., universally quantified identities), we write $A \vdash e = f$ to denote that the pair $\langle e, f \rangle$ belongs to the smallest congruence containing every axiom in A. Equivalently, $A \vdash e = f$ holds iff this statement is derivable in the equational logic described in Table 1. Similarly, $A \vdash e \leq f$ is the smallest precongruence

Table 2 Axioms

$s\,;(t\,;u) = (s\,;t)\,;u$	(A1)	$e+(f+g) = (e+f)+g$	(C1)
$s \parallel (t \parallel u) = (s \parallel t) \parallel u$	(A2)	$e+f = f+e$	(C2)
$s \parallel t = t \parallel s$	(A3)	$e+e = e$	(C3)
$1\,;s = s$	(A4)	$0+e = e$	(C4)
$s\,;1 = s$	(A5)	$0\,;e = e\,;0 = 0$	(C5)
$1 \parallel s = s$	(A6)	$0 \parallel e = 0$	(C6)
$[[s]] = [s]$	(A7)	$e\,;(f+g) = (e\,;f)+(e\,;g)$	(C7)
$[1] = 1$	(A8)	$(e+f)\,;g = (e\,;g)+(f\,;g)$	(C8)
		$e \parallel (f+g) = (e \parallel f)+(e \parallel g)$	(C9)
$(s \parallel t)\,;(u \parallel v) \leq (s\,;u) \parallel (t\,;v)$	(B1)	$[0] = 0$	(C10)
$[s] \leq s$	(B2)	$[e+f] = [e]+[f]$	(C11)

containing A, where equality axioms are understood as pairs of inequational axioms. An inference system is also provided in Table 1. We will consider the following sets of axioms:

$$\begin{aligned}
\mathrm{BiMon}_\Box &:= (\mathrm{A1}) - (\mathrm{A8}) && \text{(Bimonoid with boxes)}\\
\mathrm{CMon}_\Box &:= \mathrm{BiMon}_\Box, (\mathrm{B1})(\mathrm{B2}) && \text{(Concurrent monoid with boxes)}\\
\mathrm{SR}_\Box &:= \mathrm{BiMon}_\Box, (\mathrm{C1}) - (\mathrm{C11}) && \text{(Bisemiring with boxes)}\\
\mathrm{CSR}_\Box &:= \mathrm{SR}_\Box, (\mathrm{B1}), (\mathrm{B2}) && \text{(Concurrent semiring with boxes)}
\end{aligned}$$

In the last theory, inequational axioms $e \leq f$ should be read as $e + f = f$. Indeed one can show that for $A \in \{\mathrm{SR}_\Box, \mathrm{CSR}_\Box\}$, we have

$$A \vdash e \leq f \Leftrightarrow A \vdash e + f = f \qquad A \vdash e = f \Leftrightarrow A \vdash e \leq f \wedge A \vdash f \leq e.$$

2.3.1 Posets up to isomorphism: the free bimonoid with boxes

In this section we show that the axioms $\mathrm{BiMon}_\Box$ provide a sound and complete axiomatisation of isomorphisms between posets with boxes; that is,

$$[\![s]\!] \cong [\![t]\!] \Leftrightarrow \mathrm{BiMon}_\Box \vdash s = t.$$

Before we move to the prove of this statement, we need to make some remarks and set up some auxiliary definitions.

First, notice that the constant 1 can be handled easily:

▶ **Lemma 24.** *For any term $t \in \mathtt{SP}_\Sigma$, the set $\mathcal{E}_{[\![t]\!]}$ is empty iff* $\mathrm{BiMon}_\Box \vdash t = 1$.

Proof. Since $\mathcal{E}_{[\![t]\!]} = \emptyset$, t does not feature any symbol from Σ, meaning $t \in \mathtt{SP}_\emptyset$. We may then show that for every term $t \in \mathtt{SP}_\emptyset$ we have $\mathrm{BiMon}_\Box \vdash t = 1$. ◀

We can also remove boxes from terms, in the following sense.

▶ **Lemma 25.** *Given a term $s \in \mathtt{SP}_\Sigma$ such that $\mathcal{E}_{[\![s]\!]} \in \mathcal{B}_{[\![s]\!]}$, there exists a term t such that* $\mathrm{BiMon}_\Box \vdash s = [t]$ *and $\mathcal{E}_{[\![t]\!]} \notin \mathcal{B}_{[\![t]\!]}$.*

Proof. We proceed by induction on s:

- $s = 1$, $s = a$: contradicts the premiss;
- $s = [s']$: we have to consider two cases:
  - $\mathcal{E}_{[\![s']\!]} \in \mathcal{B}_{[\![s']\!]}$: in this case, by induction we get t such that $\mathcal{E}_{[\![t]\!]} \notin \mathcal{B}_{[\![t]\!]}$, and
$$\mathrm{BiMon}_{\square} \vdash s = [s'] = [[t]] = [t]$$
  - $\mathcal{E}_{[\![s']\!]} \notin \mathcal{B}_{[\![s']\!]}$: pick $t = s'$;
- $s = s_1 \,;\, s_2$: we know the following facts:
$$\mathcal{E}_{[\![s_1]\!]} \cup \mathcal{E}_{[\![s_2]\!]} = \mathcal{E}_{[\![s]\!]} \in \mathcal{B}_{[\![s]\!]} = \mathcal{B}_{[\![s_1]\!]} \cup \mathcal{B}_{[\![s_2]\!]}$$
$$\forall \beta \in \mathcal{B}_{[\![s_i]\!]},\ \beta \subseteq \mathcal{E}_{[\![s_i]\!]}$$
$$\mathcal{E}_{[\![s_1]\!]} \cap \mathcal{E}_{[\![s_2]\!]} = \emptyset.$$
  From them, we deduce that either $\mathcal{E}_{[\![s_1]\!]} = \emptyset$ or $\mathcal{E}_{[\![s_2]\!]} = \emptyset$. We conclude by applying the induction hypothesis to the appropriate sub-term, and use Lemma 24 to conclude;
- $s = s_1 \parallel s_2$: same as $s_1 \,;\, s_2$. ◀

We now introduce a syntactic variant of the $P{\downarrow_A}$ operator we introduced earlier, which extracts a sub-poset out of a poset, guided by a subset of events.

▶ **Definition 26** (Syntactic restriction). *Let $s \in \mathtt{SP}_\Sigma$ be a term and $A \subseteq \mathcal{E}_{[\![s]\!]}$ a set of events. The* syntactic restriction *of s to A, written $\pi_A(s)$ is defined by induction on terms as follows:*

$$\pi_A(1) := 1 \qquad \pi_A(a) := \begin{cases} 1 & if\ A = \emptyset \\ a & otherwise \end{cases} \qquad \pi_A(s\,;t) := \pi_{A \cap \mathcal{E}_{[\![s]\!]}}(s)\,;\pi_{A \cap \mathcal{E}_{[\![s]\!]}}(t)$$

$$\pi_A([s]) := \begin{cases} [s] & if\ A = \mathcal{E}_{[\![e]\!]} \\ \pi_A(s) & otherwise \end{cases} \qquad \pi_A(s \parallel t) := \pi_{A \cap \mathcal{E}_{[\![s]\!]}}(s) \parallel \pi_{A \cap \mathcal{E}_{[\![s]\!]}}(t).$$

The main properties of this operator are stated in the following observation:

▶ **Fact 27.** *For any $s \in \mathtt{SP}_\Sigma$ and $A \subseteq \mathcal{E}_{[\![s]\!]}$ the following hold:*

(i) $[\![\pi_A(s)]\!] \cong [\![s]\!]{\downarrow_A}$*;*

(ii) *if A is prefix and nested, then* $\mathrm{BiMon}_{\square} \vdash \pi_A(s)\,;\pi_{\overline{A}}(s) = s$*;*

(iii) *if A is isolated and nested, then* $\mathrm{BiMon}_{\square} \vdash \pi_A(s) \parallel \pi_{\overline{A}}(s) = s$.

We may now establish the main result of this section:

▶ **Theorem 28.** *For any pair of terms $s, t \in \mathtt{SP}_\Sigma$, the following holds:*

$$[\![s]\!] \cong [\![t]\!] \Leftrightarrow \mathrm{BiMon}_{\square} \vdash s = t$$

Proof. As often for this kind of result, the right-to-left implication; that is, soundness, is very simple to check, by a simple induction on the derivation.

For the converse direction, we prove the following statement by induction on the term s:

$$\forall t \in \mathtt{SP}_\Sigma,\ [\![s]\!] \cong [\![t]\!] \Rightarrow \mathrm{BiMon}_{\square} \vdash s = t.$$

- $s = 1$: follows from Lemma 24.
- $s = a$: we prove the result by induction on t:
  - $t = 1$, $t = [t']$: impossible since $[\![t]\!] \cong \mathbb{o}$;
  - $t = b$: since $[\![t]\!] \cong \mathbb{o}$ this means $a = b$; i.e., by reflexivity $\mathrm{BiMon}_{\square} \vdash t = a$;

- $t = t_1 \,;\, t_2$: since $\mathbb{1}$ has a single event, and $\mathcal{E}_{[\![t_1 \,;\, t_2]\!]} = \mathcal{E}_{[\![t_1]\!]} \uplus \mathcal{E}_{[\![t_2]\!]}$, that event must be either in $\mathcal{E}_{[\![t_1]\!]}$ or in $\mathcal{E}_{[\![t_2]\!]}$, and the other term has no event. The term that has no event, by Lemma 24, is provably equal to 1. The term containing an event is isomorphic to $\mathbb{1}$, so by induction it is provably equal to a. Hence we get that t is provably equal to either $a \,;\, 1$ or $1 \,;\, a$, both of which are provably equal to a;
- $t = t_1 \parallel t_2$: same as $t_1 \,;\, t_2$.

- $s = s_1 \,;\, s_2$: let $A := \mathcal{E}_{[\![s_1]\!]} \subseteq \mathcal{E}_{[\![s]\!]}$. Notice that $\overline{A} = \mathcal{E}_{[\![s]\!]} \setminus \mathcal{E}_{[\![s_1]\!]} = \mathcal{E}_{[\![s_2]\!]}$. Let ϕ be the isomorphism from $[\![s]\!]$ to $[\![t]\!]$ and let $t_1 := \pi_{\phi(A)}(t)$ and $t_2 := \pi_{\overline{\phi(A)}}(t)$. Since A is nested and prefix, so is its image by the isomorphism ϕ. Therefore by Fact 27 we get that: $\mathrm{BiMon}_{\square} \vdash t = t_1 \,;\, t_2$. By properties of isomorphisms, we can also see that for any set $X \subseteq \mathcal{E}_{[\![s]\!]}$ we have $[\![s]\!]\!\downarrow_X \cong [\![t]\!]\!\downarrow_{\phi(X)}$. Hence we have $[\![t_1]\!] \cong [\![t]\!]\!\downarrow_{\phi(A)} \cong [\![s]\!]\!\downarrow_A \cong [\![s_1]\!]$. Similarly, and because by bijectivity of ϕ we have $\overline{\phi(A)} = \phi\left(\overline{A}\right)$, we get $[\![t_2]\!] \cong [\![s_2]\!]$. We may thus conclude by induction that $\mathrm{BiMon}_{\square} \vdash s_i = t_i$ $(i \in \{1, 2\})$; that is,

  $$\mathrm{BiMon}_{\square} \vdash s = s_1 \,;\, s_2 = t_1 \,;\, t_2 = t.$$

- $s = s_1 \parallel s_2$: same as $s_1 \,;\, s_2$. ◀

2.3.2 Posets up to subsumption: the free concurrent monoid

In this section we show a similar relation between the axioms of $\mathrm{CMon}_{\square}$ and poset homomorphisms, namely that for any pair of terms s, t we have $[\![s]\!] \sqsubseteq [\![t]\!] \Leftrightarrow \mathrm{CMon}_{\square} \vdash s \leq t$.

To prove this result, we will deal with the two extra axioms (B1) and (B2) separately. In order to do so, we define the following sub-orders of $\sqsubseteq$:

- we write $P \sqsubseteq_b Q$ if there is an order-reflecting poset homomorphism $\phi : Q \to P$;
- we write $P \sqsubseteq_o Q$ if there is an box-reflecting poset homomorphism $\phi : Q \to P$.

▶ **Lemma 29** (Factorization of subsumption). *If $P \sqsubseteq Q$, then there are R_1, R_2 such that: $P \sqsubseteq_o R_1 \sqsubseteq_b Q$ and $P \sqsubseteq_b R_2 \sqsubseteq_o Q$. In other words, $\sqsubseteq = \sqsubseteq_o \circ \sqsubseteq_b$ and $\sqsubseteq = \sqsubseteq_b \circ \sqsubseteq_o$.*

Proof. Let $\phi : Q \to P$ be a poset homomorphism witnessing $P \sqsubseteq Q$. We may define $R_1 := \langle \mathcal{E}_Q, \leq_Q, \lambda_Q, \{B \mid \phi(B) \in \mathcal{B}_P\} \rangle$ and $R_2 := \langle \mathcal{E}_P, \leq_P, \lambda_P, \phi(\mathcal{B}_Q) \rangle$. Checking that $P \sqsubseteq_o R_1 \sqsubseteq_b Q$ and $P \sqsubseteq_b R_2 \sqsubseteq_o Q$ hold is a simple matter of unfolding definitions. ◀

We also notice the following properties of $\sqsubseteq_b$ and $\sqsubseteq_o$ with respect to the forbidden patterns of series–parallel posets:

▶ **Lemma 30.** *Let P, Q be two posets:*

(i) *if $P \sqsubseteq_b Q$, then P contains $\boldsymbol{P}_1$ iff Q contains $\boldsymbol{P}_1$;*
(ii) *if $P \sqsubseteq_o Q$, then P contains $\boldsymbol{P}_2$ iff Q contains $\boldsymbol{P}_2$;*
(iii) *if $P \sqsubseteq_b Q$ and Q contains $\boldsymbol{P}_3$, then P contains $\boldsymbol{P}_3$;*
(iv) *if $P \sqsubseteq_b Q$ and Q contains $\boldsymbol{P}_4$, then P contains $\boldsymbol{P}_4$.*

Proof. **(i)** if $\phi : Q \to P$ is order reflecting, then by definition we have

$$x \leq_Q y \Leftrightarrow \phi(x) \leq_P \phi(y).$$

The result follows immediately.

(ii) if $\phi : Q \to P$ is order reflecting, then by definition we have

$$B \in \mathcal{B}_Q \Leftrightarrow \phi(B) \in \mathcal{B}_P$$

$$e \in B \setminus C \Leftrightarrow \phi(e) \in \phi(B) \setminus \phi(C) \qquad e \in B \cap C \Leftrightarrow \phi(e) \in \phi(B) \cap \phi(C).$$

The result follows immediately.

(iii) if $\phi : Q \to P$ is order reflecting and Q contains $\mathbf{P}_3$ then by definition of the pattern we have $e_1, e_2, e_3 \in \mathcal{E}_Q$ and $B \in \mathcal{B}_Q$ such that

$$e_1 \notin B \qquad e_2 \in B \qquad e_3 \in B \qquad e_1 \leq_Q e_2 \qquad e_1 \not\leq_Q e_3.$$

Since ϕ is a poset homomorphism, we know that $\phi(B) \in \mathcal{B}_P$ and $\phi(e_1) \leq_P \phi(e_2)$. By definition of the direct image, we also know that $\phi(e_1) \notin \phi(B)$ and $\phi(e_2), \phi(e_3) \in \phi(B)$. Finally, since ϕ is order reflecting $\phi(e_1) \not\leq_P \phi(e_3)$.

(iv) similar to the proof for (iii). ◀

We now prove that $\mathrm{CMon}_\square$ completely axiomatizes $\sqsubseteq_b$:

▶ **Lemma 31.** *If* $[\![s]\!] \sqsubseteq_b [\![t]\!]$*, then* $\mathrm{CMon}_\square \vdash s \leq t$.

Proof. First, we define the following operator $\mathcal{H}(_) : \mathbb{P}_\Sigma \to \mathcal{P}_f(\mathbb{P}_\Sigma)$:

$$\mathcal{H}(P) := \{\langle \mathcal{E}_P, \leq_P, \lambda_P, B\rangle \mid B \subseteq \mathcal{B}_P\}.$$

From the definitions it is straightforward to show that $P \sqsubseteq_b Q$ iff Q is isomorphic to some $P' \in \mathcal{H}(P)$. This operator $\mathcal{H}(P)$ can be mirrored on terms; i.e., we can associate inductively to each term s a finite set of terms $\mathrm{H}(s)$ such that $\forall P \in \mathcal{H}([\![s]\!]), \exists t \in \mathrm{H}(s) : [\![t]\!] \cong P$ and $\forall t \in \mathrm{H}(s), \mathrm{CMon}_\square \vdash s \leq t$. We may therefore conclude:

$$\begin{aligned}
[\![s]\!] \sqsubseteq_b [\![t]\!] &\Rightarrow \exists P \in \mathcal{H}([\![s]\!]) : [\![t]\!] \cong P \\
&\Rightarrow \exists t' \in \mathrm{H}(s) : [\![t']\!] \cong [\![t]\!] \\
&\Rightarrow \exists t' \in \mathrm{H}(s) : \mathrm{BiMon}_\square \vdash t' = t \wedge \mathrm{CMon}_\square \vdash s \leq t' \\
&\Rightarrow \mathrm{CMon}_\square \vdash s \leq t' = t.
\end{aligned}$$ ◀

We now prove the same result for $\sqsubseteq_o$.

▶ **Lemma 32.** *If* $[\![s]\!] \sqsubseteq_o [\![t]\!]$ *then* $\mathrm{CMon}_\square \vdash s \leq t$.

▶ Remark 33. The proof we give below relies on Gischer's completeness theorem [8]. In the Coq proof however, we make no assumptions, and we do not have access to Gischer's result. Therefore we perform a different, more technically involved proof there, with Gischer's theorem as a corollary.

Proof. We will perform this proof by induction on the number of boxes in s. Let $\phi : [\![t]\!] \to [\![s]\!]$ be the box-reflecting homomorphism witnessing $[\![s]\!] \sqsubseteq_o [\![t]\!]$.

If s contains no boxes, then by Gischer's completeness theorem we know that

$$[\![s]\!] \sqsubseteq_o [\![t]\!] \Rightarrow \mathrm{BiMon}_\square, (\mathrm{B}_1) \vdash s \leq t.$$

Hence, as we have $\mathrm{BiMon}_\square, (\mathrm{B}_1) \subseteq \mathrm{CMon}_\square$, we get $\mathrm{CMon}_\square \vdash s \leq t$.

If on the other hand s has boxes, consider the following set of boxes:

$$\mathcal{B} := \left\{B \in \mathcal{B}_{[\![s]\!]} \;\middle|\; \forall C \in \mathcal{B}_{[\![s]\!]}, B \subseteq C \Rightarrow B = C\right\} \left(= \max \mathcal{B}_{[\![s]\!]}\right).$$

Notice that since ϕ is box-reflecting, we have that $\mathcal{B} = \phi\left(\max \mathcal{B}_{[\![t]\!]}\right)$. Furthermore, we have the following property: for any box $B \in \max \mathcal{B}_{[\![t]\!]}$, the map $\phi\downarrow_B$ is a box-reflecting homomorphism from $[\![t]\!]\downarrow_B$ to $[\![s]\!]\downarrow_{\phi(B)}$. We pick new symbols for the elements of $\mathcal{B}$; that is, we

find a set Σ' disjoint from Σ and a bijection $\ell(_) : \mathcal{B} \to \Sigma'$. Using the observation Lemma 25 we made earlier, we find two maps $s(_), t(_) : \Sigma'$ such that:

$$\forall B \in \mathcal{B}, \qquad \mathrm{BiMon}_\Box \vdash \pi_B(s) = [s(\ell(B))]. \tag{2.1}$$

$$\forall x \in \Sigma', \qquad \mathcal{E}_{[\![s(x)]\!]} \notin \mathcal{B}_{[\![s(x)]\!]} \tag{2.2}$$

$$\forall B \in \max \mathcal{B}_{[\![t]\!]}, \qquad \mathrm{BiMon}_\Box \vdash \pi_B(t) = [t(\ell(\phi(B)))]. \tag{2.3}$$

$$\forall x \in \Sigma', \qquad \mathcal{E}_{[\![t(x)]\!]} \notin \mathcal{B}_{[\![t(x)]\!]} \tag{2.4}$$

Clearly, for every $x \in \Sigma'$, $s(x)$ has strictly less boxes than s. Our previous observations imply that $[\![s(x)]\!] \sqsubseteq_o [\![t(x)]\!]$. Therefore, by induction hypothesis, we get that $\forall x \in \Sigma'$, $\mathrm{CMon}_\Box \vdash s(x) \leq t(x)$, hence:

$$\forall x \in \Sigma', \mathrm{CMon}_\Box \vdash [s(x)] \leq [t(x)].$$

Combined with (2.1) and (2.3) this yields that $\forall B \in \max \mathcal{B}_{[\![t]\!]}$, $\mathrm{CMon}_\Box \vdash \pi_{\phi(B)}(s) \leq \pi_B(t)$. We define two substitutions $\sigma, \tau : \Sigma \cup \Sigma' \to \mathtt{SP}_\Sigma$ as follows:

$$\sigma(x) := \begin{cases} \pi_B(s) & \text{if } x = \ell(B) \\ x & \text{if } x \in \Sigma \end{cases} \qquad \tau(x) := \begin{cases} \pi_B(t) & \text{if } x = \ell(\phi(B)) \\ x & \text{if } x \in \Sigma \end{cases}$$

Finally, we syntactically substitute maximal boxed sub-terms with letters from Σ' in s, t, yielding terms $s', t' \in \mathtt{SP}_{\Sigma \cup \Sigma'}$ such that $\hat{\sigma}(s') = s$, $\hat{\tau}(t') = t$, and neither s' nor t' has any box. By unfolding the definitions, we can check that since $[\![s]\!] \sqsubseteq_o [\![t]\!]$ we have $[\![s]\!]' \sqsubseteq_o [\![t]\!]'$. Furthermore, since s' and t' do not contain any box, we may use Gischer's theorem to prove that $\mathrm{CMon}_\Box \vdash s' \leq t'$. We may now conclude: by applying σ everywhere in the proof of $\mathrm{CMon}_\Box \vdash s' \leq t'$, we get that $\mathrm{CMon}_\Box \vdash s = \hat{\sigma}(s') \leq \hat{\sigma}(t')$. Since $\forall a \in \Sigma \cup \Sigma'$, we have $\mathrm{CMon}_\Box \vdash \sigma(a) \leq \tau(a)$, we get $\mathrm{CMon}_\Box \vdash \hat{\sigma}(t') \leq \hat{\tau}(t') = t$. ◀

▶ **Theorem 34.** *For any pair of terms $s, t \in \mathtt{SP}_\Sigma$, the following holds:*

$$[\![s]\!] \sqsubseteq [\![t]\!] \Leftrightarrow \mathrm{CMon}_\Box \vdash s \leq t.$$

Proof. Soundness is straightforward, and completeness arises from Lemma 29,Lemma 31, and Lemma 32. ◀

2.3.3 Completeness results for sets of posets

The following lemma allows us to extend seamlessly our completeness theorem from $\mathrm{BiMon}_\Box$ to $\mathrm{SR}_\Box$ and from $\mathrm{CMon}_\Box$ to $\mathrm{CSR}_\Box$.

▶ **Lemma 35.** *There exists a function $T__ : \mathtt{T}_\Sigma \to \mathcal{P}_f(\mathtt{SP}_\Sigma)$ such that:* $\mathrm{SR}_\Box \vdash e = \sum_{s \in T_e} s$ *and* $[\![e]\!] \cong \{[\![s]\!] \mid s \in T_e\}$.

Proof. T_e is defined by induction on e:

$$T_0 := \emptyset \qquad T_1 := \{1\}$$
$$T_a := \{a\} \qquad T_{[e]} := \{[s] \mid s \in T_e\}$$
$$T_{e;f} := \{s;t \mid \langle s, t\rangle \in T_e \times T_f\} \qquad T_{e \parallel f} := \{s \parallel t \mid \langle s, t\rangle \in T_e \times T_f\}$$
$$T_{e+f} := T_e \cup T_f.$$

Checking the lemma is done by a simple induction. ◀

From there, we can easily establish two completeness results.

▶ **Theorem 36.** *For any pair of terms $e, f \in \mathtt{T}_\Sigma$, the following hold:*

$$[\![e]\!] \cong [\![f]\!] \Leftrightarrow \mathrm{SR}_\Box \vdash e = f \tag{2.5}$$

$$[\![e]\!]{\downarrow} \cong [\![f]\!]{\downarrow} \Leftrightarrow \mathrm{CSR}_\Box \vdash e = f \tag{2.6}$$

Proof. **(2.5)** Soundness is easy to check. By a simple induction on the derivation tree, we can ensure that $\mathrm{SR}_\Box \vdash e = f \Rightarrow [\![e]\!] \cong [\![f]\!]$.

Using Lemma 35, we may rewrite any term e as a finite union of series–parallel terms. Let $s \in T_e$. By soundness, there is $P \in [\![e]\!]$ such that $P \cong [\![s]\!]$. Since $[\![e]\!] \cong [\![f]\!]$, there is $Q \in [\![f]\!]$ such that $P \cong Q$. Since $\mathrm{SR}_\Box \vdash f = \sum_{s \in T_f} s$, by soundness there is $t \in T_f$ such that $Q \cong t$. Therefore $[\![s]\!] \cong [\![t]\!]$, so by Theorem 28 we have $\mathrm{BiMon}_\Box \vdash s = t$. Since $\mathrm{BiMon}_\Box \subseteq \mathrm{SR}_\Box$, we also have $\mathrm{SR}_\Box \vdash s = t$, and because $t \in T_f$ we get $\mathrm{SR}_\Box \vdash s \leq f$. Since this holds for every $s \in T_e$, this means that

$$\mathrm{SR}_\Box \vdash e = \sum_{s \in T_e} s \leq f.$$

By a symmetric argument, we obtain $\mathrm{SR}_\Box \vdash f \leq e$, allowing us to conclude by antisymmetry that $\mathrm{SR}_\Box \vdash e = f$.

(2.6) Again, soundness is straightforward. For completeness, it is sufficient to show if $[\![e]\!] \sqsubseteq [\![f]\!]$ then $\mathrm{CSR}_\Box \vdash e \leq f$. Assume $[\![e]\!] \sqsubseteq [\![f]\!]$; i.e., every poset in $[\![e]\!]$ is subsumed by some poset in $[\![f]\!]$. Since $\mathrm{SR}_\Box \subseteq \mathrm{CSR}_\Box$, we get, by Lemma 35, that

$$\mathrm{CSR}_\Box \vdash e = \sum_{s \in T_e} s \qquad\qquad \mathrm{CSR}_\Box \vdash f = \sum_{t \in T_f} t.$$

Let $s \in T_e$. Since $[\![e]\!] \sqsubseteq [\![f]\!]$, and because of Lemma 35, we know that there is a term $t \in T_f$ such that $[\![s]\!] \sqsubseteq [\![t]\!]$. By Theorem 34, that means $\mathrm{CMon}_\Box \vdash s \leq t$. Since $\mathrm{CMon}_\Box \subseteq \mathrm{CSR}_\Box$, we get $\mathrm{CSR}_\Box \vdash s \leq t \leq f$. This means that for every $s \in T_e$, we have $\mathrm{CSR}_\Box \vdash s \leq f$, hence that $\mathrm{CSR}_\Box \vdash e = \sum_{s \in T_e} s \leq f$. ◀

3 Logic for pomsets with boxes

We introduce a logic for reasoning about pomsets with boxes, in the form of a bunched modal logic, in the sense of [21, 7, 5, 1, 23], with substructural connectives corresponding to each of sequential and concurrent composition. Modalities characterize boxes and locality. The logic is also conceptually related to Concurrent Separation Logic [19, 3].

In contrast with other work, pomset logic is a logic of *behaviours*. A behaviour is a run of some program, represented as a pomset. The logic describes such behaviours in terms of the order in which instructions are, or can be, executed, and the separation properties of sub-runs. Note, in particular, that we do not define any notion of *state*. On the contrary, existing approaches, such as dynamic logic and Hennessy–Milner logic for example, put the emphasis on the state of the machine before and after running the program. Typically, the assertion language describes the memory-states, and some accessibility relations between them. The semantics then relies on labelled transition systems to interpret action modalities.

Here, the satisfaction relation (given in Definition 38) directly defines a relation between sets of behaviours and formulas. An intuitionistic version of the semantics given in Definition 38 might be set up — cf. Tarski's semantics and the semantics of relevant logic — in terms of (ternary) relations on behaviours.

3.1 Pomset logic: definitions

We generate the set of formulas $\mathtt{F}_\Sigma$ and the set of positive formulas $\mathtt{F}^+_\Sigma$ as follows:

$$\phi, \psi \in \mathtt{F}^+_\Sigma ::= \bot \mid a \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \blacktriangleright \psi \mid \phi \star \psi \mid [\phi] \mid (\!|\phi|\!)$$
$$\phi, \psi \in \mathtt{F}_\Sigma ::= \bot \mid a \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \blacktriangleright \psi \mid \phi \star \psi \mid [\phi] \mid (\!|\phi|\!) \mid \neg\phi$$

▶ Remark 37. Here the atomic predicates are chosen to be exactly Σ. Another natural choice would be a separate set $Prop$ of atomic predicates, together with a valuation $v : Prop \to \mathcal{P}(\Sigma)$ to indicate which actions satisfy which predicate. Both definitions are equivalent:

- to encode a formula over $Prop$ as a formula over Σ, simply replace every predicate $p \in Prop$ with the formula $\bigvee_{a \in v(p)} a$
- to encode a formula over Σ as one over $Prop$, we need to make the customary assumption that $\forall a \in \Sigma, \exists p \in Prop : v(p) = \{a\}$.

These formulas are interpreted over posets. We define a satisfaction relation $\models_R$ that is parametrized by a relation $R \subseteq \mathbb{P}_\Sigma \times \mathbb{P}_\Sigma$ (to be instantiated later on with $\cong$, $\sqsubseteq$, and $\sqsupseteq$).

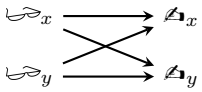
▶ **Definition 38.** *$P \models_R \phi$ is defined by induction on $\phi \in \mathtt{F}_\Sigma$:*

- $P \models_R \bot$ *iff* $R(P, \mathbb{c})$
- $P \models_R a$ *iff* $R(P, \mathbb{o})$
- $P \models_R \neg\phi$ *iff* $P \not\models_R \phi$
- $P \models_R \phi \vee \psi$ *iff* $P \models_R \phi$ *or* $P \models_R \psi$
- $P \models_R \phi \wedge \psi$ *iff* $P \models_R \phi$ *and* $P \models_R \psi$
- $P \models_R \phi \blacktriangleright \psi$ *iff* $\exists P_1, P_2$ *such that* $R(P, P_1 \otimes P_2)$ *and* $P_1 \models_R \phi$ *and* $P_2 \models_R \psi$
- $P \models_R \phi \star \psi$ *iff* $\exists P_1, P_2$ *such that* $R(P, P_1 \oplus P_2)$ *and* $P_1 \models_R \phi$ *and* $P_2 \models_R \psi$
- $P \models_R [\phi]$ *iff* $\exists Q$ *such that* $R(P, [Q])$ *and* $Q \models_R \phi$
- $P \models_R (\!|\phi|\!)$ *iff* $\exists P', Q$ *such that* $R(P, P')$ *and* $P' \Rsh Q$ *and* $Q \models_R \phi$.

The operator $[-]$ describes the (encapsulated) properties of boxed terms. The operator $(\!|-|\!)$ identifies a property of a term that is obtained by removing parts, including boxes and events, of its satisfying term (i.e., its world) such that remainder satisfies the formula that it guards. The meanings of these operators are discussed more fully in Section 4.2.

Note that $\models_\sqsupseteq$ and $\models_\sqsubseteq$ will only be used with positive formulas. Given a formula ϕ and a relation R, we may define the R-semantics of ϕ as $[\![\phi]\!]_R := \{P \in \mathbb{P}_\Sigma \mid P \models_R \phi\}$.

▶ **Example 39.** Recall the problematic pattern we saw in the running example; that is,



This pattern can be represented by the formula $\mathbf{conflict} := (\!|(\curvearrowright_x \star \curvearrowright_y) \blacktriangleright (\blacktriangle_x \star \blacktriangle_y)|\!)$.

We may also interpret these formulas over sets of posets. We consider here two ways a set of posets X may satisfy a formula:

- X satisfies ϕ *universally* if every poset in X satisfies ϕ;
- X satisfies ϕ *existentially* if some poset in X satisfies ϕ.

Combined with our three satisfaction relations for pomsets, this yields six definitions:

$$X \models^{\forall}_{\cong} \phi \text{ iff } \forall P \in X, P \models_\cong \phi \qquad X \models^{\exists}_{\cong} \phi \text{ iff } \exists P \in X, P \models_\cong \phi$$
$$X \models^{\forall}_{\sqsupseteq} \phi \text{ iff } \forall P \in X, P \models_\sqsupseteq \phi \qquad X \models^{\exists}_{\sqsupseteq} \phi \text{ iff } \exists P \in X, P \models_\sqsupseteq \phi$$
$$X \models^{\forall}_{\sqsubseteq} \phi \text{ iff } \forall P \in X, P \models_\sqsubseteq \phi \qquad X \models^{\exists}_{\sqsubseteq} \phi \text{ iff } \exists P \in X, P \models_\sqsubseteq \phi.$$

For a term $e \in \mathtt{T}_\Sigma$, we write $e \models^y_R \phi$ to mean $[\![e]\!] \models^y_R \phi$. In terms of R-semantics, these definitions may be formalized as

$$e \models^\exists_R \phi \Leftrightarrow [\![e]\!] \cap [\![\phi]\!]_R \neq \emptyset \quad \text{and} \qquad e \models^\forall_R \phi \Leftrightarrow [\![e]\!] \subseteq [\![\phi]\!]_R \,. \tag{3.1}$$

3.2 Properties of pomset logic

We now discuss some of the properties of pomset logic. First, notice that if the relation R is transitive, then for any posets P, Q and any formula $\phi \in \mathtt{F}^+_\Sigma$, we have that

$$R(P,Q) \text{ and } Q \models_R \phi \Rightarrow P \models_R \phi. \tag{3.2}$$

Proof. Let R be a transitive relation, we show by induction on $\Phi \in \mathtt{F}^+_\Sigma$ that $\forall P, Q \in \mathbb{P}_\Sigma$, if $R(P,Q)$ and $Q \models_R \Phi$ then $P \models_R \Phi$.

- If $\Phi = \phi \vee \psi$ or $\Phi = \phi \wedge \psi$, we use the induction hypothesis to show that $Q \models_R \Phi$ implies $P \models_R \Phi$.
- If $\Phi = \bot$, $\Phi = a$, $\Phi = \phi \blacktriangleright \psi$, $\Phi = \phi \star \psi$, $\Phi = [\phi]$, or $\Phi = (\!|\phi|\!)$, then the satisfaction relation says $Q \models_R \Phi$ iff $R(Q,Q')$ and $h(Q')$. Since $R(P,Q)$ and $R(Q,Q')$, by transitivity of R we get that $R(P,Q')$ and thus we conclude that $P \models_R \Phi$ without using the induction hypothesis. ◀

If, additionally, R is symmetric, this property may be strengthened to

$$\forall P, Q \in \mathbb{P}_\Sigma, \forall \phi \in \mathtt{F}_\Sigma, \text{ if } R(P,Q), \text{ then } P \models_R \phi \Leftrightarrow Q \models_R \phi. \tag{3.3}$$

Proof. Let R be a symmetric and transitive relation, we show by induction on $\Phi \in \mathtt{F}_\Sigma$ that $\forall P, Q \in \mathbb{P}_\Sigma$, if $R(P,Q)$ then $P \models_R \Phi$ iff $Q \models_R \Phi$.

- If $\Phi = \neg\phi$, $\Phi = \phi \vee \psi$, or $\Phi = \phi \wedge \psi$, we use the induction hypothesis to conclude.
- If $\Phi = \bot$, $\Phi = a$, $\Phi = \phi \blacktriangleright \psi$, $\Phi = \phi \star \psi$, $\Phi = [\phi]$, or $\Phi = (\!|\phi|\!)$, then the satisfaction relation says $Q \models_R \Phi$ iff $R(Q,Q')$ and $h(Q')$. Since $R(P,Q)$, by symmetry and transitivity of R we get that $R(P,Q') \Leftrightarrow R(Q,Q')$ and thus we conclude that $P \models_R \Phi \Leftrightarrow Q \models_R \Phi$ without using the induction hypothesis. ◀

Furthermore, increasing the relation R increases the satisfaction relation as well:

$$R \subseteq R' \Rightarrow \forall \phi \in \mathtt{F}^+_\Sigma, \forall P \in \mathbb{P}_\Sigma,\ P \models_R \phi \Rightarrow P \models_{R'} \phi. \tag{3.4}$$

Proof. This follows by a straightforward induction on formulas. ◀

From these observations and (3.1), we obtain the following characterizations of the universal satisfaction relations for $R \in \{\cong, \sqsubseteq, \sqsupseteq\}$:

$$e \models^\forall_\cong \phi \Leftrightarrow [\![e]\!] \lesssim [\![\phi]\!]_\cong \tag{3.5}$$

$$e \models^\forall_\sqsubseteq \phi \Leftrightarrow [\![e]\!] \sqsubseteq [\![\phi]\!]_\sqsubseteq \tag{3.6}$$

$$e \models^\forall_\sqsupseteq \phi \Leftrightarrow \forall P \in [\![e]\!]\,, \exists Q \in [\![\phi]\!]_\sqsupseteq : P \sqsupseteq Q. \tag{3.7}$$

Additionally, the following preservation properties hold for sets of posets:

$$e \lesssim f \Rightarrow \forall \phi \in \mathtt{F}_\Sigma,\ \left(e \models^\exists_\cong \phi \Rightarrow f \models^\exists_\cong \phi\right) \wedge \left(f \models^\forall_\cong \phi \Rightarrow e \models^\forall_\cong \phi\right) \tag{3.8}$$

$$e \sqsubseteq f \Rightarrow \forall \phi \in \mathtt{F}^+_\Sigma,\ \left(e \models^\exists_\sqsupseteq \phi \Rightarrow f \models^\exists_\sqsupseteq \phi\right) \wedge \left(f \models^\forall_\sqsubseteq \phi \Rightarrow e \models^\forall_\sqsubseteq \phi\right) \tag{3.9}$$

Proof. (3.8) Assume $e \lesssim f$.

- If $e \models^{\exists}_{\cong} \phi$, then there exists a poset $P \in [\![e]\!] \cap [\![\phi]\!]_{\cong}$. Because $e \subsetneq f$, we can find a poset $Q \in [\![f]\!]$ such that $P \cong Q$. Since $P \in [\![\phi]\!]_{\cong}$ and $[\![\phi]\!]_{\cong}$ is closed under $\cong$ we get $Q \in [\![f]\!] \cap [\![\phi]\!]_{\cong}$.
- If $f \models^{\forall}_{\cong} \phi$, then $[\![f]\!] \subsetneq [\![\phi]\!]_{\cong}$. By transitivity, we get that $[\![e]\!] \subsetneq f \subsetneq [\![\phi]\!]_{\cong}$; i.e., $e \models^{\forall}_{\cong} \phi$.

(3.9) Assume $e \sqsubseteq f$.

- If $e \models^{\exists}_{\sqsupseteq} \phi$, then there exists a poset $P \in [\![e]\!] \cap [\![\phi]\!]_{\sqsupseteq}$. Because $e \sqsubseteq f$, we can find a poset $Q \in [\![f]\!]$ such that $P \sqsubseteq Q$. Since $P \in [\![\phi]\!]_{\sqsupseteq}$ and $[\![\phi]\!]_{\sqsupseteq}$ is upwards-closed we get $Q \in [\![f]\!] \cap [\![\phi]\!]_{\sqsupseteq}$.
- If $f \models^{\forall}_{\sqsubseteq} \phi$, then $[\![f]\!] \sqsubseteq [\![\phi]\!]_{\sqsubseteq}$. By transitivity, we get that $[\![e]\!] \sqsubseteq [\![f]\!] \sqsubseteq [\![\phi]\!]_{\sqsubseteq}$, that is, $e \models^{\forall}_{\sqsubseteq} \phi$. ◀

We can build formulas from series–parallel terms: $\phi(a) := a$, $\phi(1) := \bot$, $\phi([s]) := [\phi(s)]$, $\phi(s\,;t) := \phi(s) \blacktriangleright \phi(t)$, and $\phi(s \parallel t) := \phi(s) \star \phi(t)$. Using $T_$, we generalize this construction our full syntax: given a term $e \in \mathtt{T}_\Sigma$, we define the formula $\Phi(e) := \bigvee_{s \in T_e} \phi(s)$. These formulas are closely related to terms thanks to the following lemma:

▶ **Lemma 40.** *For any term $s \in \mathtt{SP}_\Sigma$ and any poset P, we have that*

$$P \models_{\cong} \phi(s) \Leftrightarrow P \cong [\![s]\!] \qquad P \models_{\sqsupseteq} \phi(s) \Leftrightarrow P \sqsupseteq [\![s]\!] \qquad P \models_{\sqsubseteq} \phi(s) \Leftrightarrow P \sqsubseteq [\![s]\!]\,.$$

For a term $e \in \mathtt{T}_\Sigma$ and a set of posets $X \subseteq \mathbb{P}_\Sigma$, we have that

$$X \models^{\forall}_{\cong} \Phi(e) \Leftrightarrow X \subsetneq [\![e]\!] \qquad X \models^{\forall}_{\sqsubseteq} \Phi(e) \Leftrightarrow X \sqsubseteq [\![e]\!]\,.$$

The proof of this lemma may be found in the appendix.

As an immediate corollary, for any $e \in \mathtt{T}_\Sigma$ and any $s \in \mathtt{SP}_\Sigma$, we obtain that:

$$e \models^{\exists}_{\cong} \phi(s) \Leftrightarrow [\![s]\!] \in [\![e]\!] \qquad e \models^{\exists}_{\sqsupseteq} \phi(s) \Leftrightarrow [\![s]\!] \in [\![e]\!]{\downarrow} \tag{3.10}$$

Proof.

$$e \models^{\exists}_{\cong} \phi(s) \Leftrightarrow [\![e]\!] \cap [\![\phi(s)]\!]_{\cong} \neq \emptyset \Leftrightarrow [\![e]\!] \cap \{[\![s]\!]\} \neq \emptyset \Leftrightarrow [\![s]\!] \in [\![e]\!]\,.$$
$$e \models^{\exists}_{\sqsupseteq} \phi(s) \Leftrightarrow [\![e]\!] \cap [\![\phi(s)]\!]_{\sqsupseteq} \neq \emptyset \Leftrightarrow [\![e]\!] \cap \{[\![s]\!]\}{\uparrow} \neq \emptyset \Leftrightarrow [\![s]\!] \in [\![e]\!]{\downarrow}. \qquad ◀$$

We can now establish adequacy lemmas. These should be understood as appropriate formulations of the completeness theorems relating operational equivalence and logical equivalence in the sense of van Benthem [2] and Hennessy–Milner [10, 18] for this logic (cf. [1]). From the results we have established so far, we may directly prove the following:

▶ **Proposition 41.** *For a pair of series–parallel terms $s, t \in \mathtt{SP}_\Sigma$,*

$$\mathrm{BiMon}_{\square} \vdash s = t \Leftrightarrow \forall \phi \in \mathtt{F}_\Sigma,\ ([\![s]\!] \models_{\cong} \phi \Leftrightarrow [\![t]\!] \models_{\cong} \phi) \tag{3.11}$$
$$\mathrm{CMon}_{\square} \vdash s \leq t \Leftrightarrow \forall \phi \in \mathtt{F}^{+}_\Sigma,\ ([\![s]\!] \models_{\sqsupseteq} \phi \Rightarrow [\![t]\!] \models_{\sqsupseteq} \phi)\,. \tag{3.12}$$

Proof.

$$\begin{aligned}
\mathrm{BiMon}_{\square} \vdash s = t &\Leftrightarrow [\![s]\!] \cong [\![t]\!] && \text{(Theorem 28)}\\
&\Rightarrow \forall \phi \in \mathtt{F}_\Sigma,\ [\![s]\!] \models_{\cong} \phi \Leftrightarrow [\![t]\!] \models_{\cong} \phi. && \text{(Equation (3.3))}\\
&\Rightarrow [\![s]\!] \models_{\cong} \phi(t) && \text{(Since } [\![t]\!] \models_{\cong} \phi(t) \text{ follows from Lemma 40)}\\
&\Leftrightarrow [\![s]\!] \cong [\![t]\!] && \text{(Lemma 40)}\\
&\Leftrightarrow \mathrm{BiMon}_{\square} \vdash s = t && \text{(Theorem 28)}
\end{aligned}$$

$$
\begin{aligned}
\mathrm{CMon}_{\square} \vdash s \leq t &\Leftrightarrow [\![s]\!] \sqsubseteq [\![t]\!] && \text{(Theorem 34)}\\
&\Rightarrow \forall \phi \in \mathtt{F}^{+}_{\Sigma},\ [\![s]\!] \models_{\sqsupseteq} \phi \Rightarrow [\![t]\!] \models_{\sqsupseteq} \phi. && \text{(Equation (3.2))}\\
&\Rightarrow [\![t]\!] \models_{\sqsupseteq} \phi(s) && \text{(Since } [\![s]\!] \models_{\sqsupseteq} \phi(s) \text{ follows from Lemma 40)}\\
&\Leftrightarrow [\![t]\!] \sqsupseteq [\![s]\!] && \text{(Lemma 40)}\\
&\Leftrightarrow \mathrm{CMon}_{\square} \vdash s \leq t && \text{(Theorem 34)}
\end{aligned}
$$

◀

This extends to sets of pomsets in the following sense:

▶ **Proposition 42.** *Given two terms $e, f \in \mathtt{T}_{\Sigma}$, the following equivalences hold:*

$$\mathrm{SR}_{\square} \vdash e \leq f \Leftrightarrow \left(\forall \phi,\ e \models^{\exists}_{\cong} \phi \Rightarrow f \models^{\exists}_{\cong} \phi\right) \Leftrightarrow \left(\forall \phi,\ f \models^{\forall}_{\cong} \phi \Rightarrow e \models^{\forall}_{\cong} \phi\right) \tag{3.13}$$

$$\mathrm{SR}_{\square} \vdash e = f \Leftrightarrow \left(\forall \phi,\ e \models^{\exists}_{\cong} \phi \Leftrightarrow f \models^{\exists}_{\cong} \phi\right) \Leftrightarrow \left(\forall \phi,\ e \models^{\forall}_{\cong} \phi \Leftrightarrow f \models^{\forall}_{\cong} \phi\right) \tag{3.14}$$

$$\mathrm{CSR}_{\square} \vdash e \leq f \Leftrightarrow \left(\forall \phi,\ e \models^{\exists}_{\sqsupseteq} \phi \Rightarrow f \models^{\exists}_{\sqsupseteq} \phi\right) \Leftrightarrow \left(\forall \phi,\ f \models^{\forall}_{\sqsubseteq} \phi \Rightarrow e \models^{\forall}_{\sqsubseteq} \phi\right) \tag{3.15}$$

$$\mathrm{CSR}_{\square} \vdash e = f \Leftrightarrow \left(\forall \phi,\ e \models^{\exists}_{\sqsupseteq} \phi \Leftrightarrow f \models^{\exists}_{\sqsupseteq} \phi\right) \Leftrightarrow \left(\forall \phi,\ e \models^{\forall}_{\sqsubseteq} \phi \Leftrightarrow f \models^{\forall}_{\sqsubseteq} \phi\right) \tag{3.16}$$

Proof. (3.13) We prove both directions.

($\Rightarrow$) Assume $\mathrm{SR}_{\square} \vdash e \leq f$. By Equation (2.5) this means $[\![e]\!] \lesssim [\![f]\!]$. Therefore, we may conclude by Equation (3.8).

($\Leftarrow$) We show that each LHS implies $[\![e]\!] \lesssim [\![f]\!]$; that is, $\mathrm{SR}_{\square} \vdash e \leq f$:

- Assume $\forall \phi,\ f \models^{\forall}_{\cong} \phi \Rightarrow e \models^{\forall}_{\cong} \phi$. Then in particular, since $[\![f]\!] \lesssim [\![f]\!]$ by Lemma 40 we have $f \models^{\forall}_{\cong} \Phi(f)$, hence $e \models^{\forall}_{\cong} \Phi(f)$ ergo $[\![e]\!] \lesssim [\![f]\!]$;
- Assume $\forall \phi,\ e \models^{\exists}_{\cong} \phi \Rightarrow f \models^{\exists}_{\cong} \phi$, and let $P \in [\![e]\!]$. By Lemma 35 we know that there is $s \in T_e$ such that $P \cong [\![s]\!]$, and by Lemma 40 we get $e \models^{\exists}_{\cong} \phi(s)$. Hence $f \models^{\exists}_{\cong} \phi(s)$. Hence, by Equation (3.10), we get $P \cong [\![s]\!] \in [\![f]\!]$.

(3.14) follows from (3.13), and the fact that $\leq$ is antisymmetric. ◀

4 Local Reasoning

Some of the discussions in this section do not rely on which satisfaction relation we pick. When this is the case, we use the symbol $\models$ to mean any of the relations $\models_{\cong}, \models_{\sqsupseteq}, \models_{\sqsubseteq}$.

4.1 Modularity

Pomset logic enjoys a high level of compositionality, much like algebraic logic. Formally, this comes from the following principle:

$$\text{If } e \models \phi \text{ and } \forall a,\ \sigma a \models \tau a, \text{ then } \hat{\sigma} e \models \hat{\tau} \phi.$$

This makes possible the following verification scenario: Let P be a large program, involving a number of simpler sub-programs $P_1, \ldots, P_n$. We may simplify P by replacing the sub-programs by uninterpreted symbols $x_1, \ldots, x_n$. We then check that this simplified program satisfies a formula Φ, the statement of which might involve the x_i. We then separately determine for each sub-program P_i some specification ϕ_i. Finally, using the principle we just stated, we can show that the full program P satisfies the formula Φ', obtained by replacing the x_i with ϕ_i.

4.2 Frame rule

A key of objective of applied, modelling-oriented, work in logic and semantics is to understand systems — such as complex programs, large-scale distributed systems, and organizations — compositionally. That is, we seek understand how the system is made of components that can be understood independently of one another. A key aspect of this is what has become known as *local reasoning*. That is, that the pertinent logical properties of the components of a system should be independent of their context.

In the world of Separation Logic [26, 12, 19], for reasoning about how computer programs manipulate memory, O'Hearn, Reynolds, and Yang [22] suggest that

> 'To understand how a program works, it should be possible for reasoning and specification to be confined to the cells that the program actually accesses. The value of any other cell will automatically remain unchanged.'

In this context, a key idea is that of the 'footprint' of a program; that is, that part of memory that is, in an appropriate sense, used by the program [24]. If, in an appropriate sense, a program executes correctly, or 'safely', on its footprint, then the so-called 'frame property' ensures that the resources present outside of the footprint and, by implication, their inherent logical properties, are unchanged by the program.

In the setting of Separation Logic, the frame property is usually represented by a Hoare-triple rule of the form

$$\frac{\{\phi\}C\{\psi\}}{\{\phi * \chi\}C\{\psi * \chi\}} \quad C \text{ is independent of } \chi.$$

That is, the formula χ does not include any variables (from the memory) that are modified by the program C.

In order to formulate the frame property in our framework, we first fix the notion of independence between a program and a formula. We say that a pomset P is *R-independent* of a formula ϕ, written $P \#_R \phi$ if $P \not\models_R (\!|[\phi]|\!)$. Since independence is meant to prevent overlap, the use of the $(\!|-|\!)$ modality should come as no surprise.

To explain the need for the $[-]$ modality, first consider a pomset P satisfying $(\!|\phi|\!)$. To extract a witness of this fact, we must remove parts of P, including boxes and events, such that the remainder satisfies ϕ. However, there are no restrictions on the relationship between the remaining events and those we have deleted. In a sequence of three events, we are allowed to keep the two extremities, and delete the middle one. In contrast, to get a witness of $(\!|[\phi]|\!)$, we need to identify a box on P whose contents satisfy ϕ, and remove all events external to that box. The result is that the deleted events, that is, the context of our witness, can only appear outside the box, and must treat all events inside uniformly. In other words, these events can interact with the behaviour encapsulated in the box, but cannot interact with individual components inside. For this reason, the frame properties given in Proposition 43 are expressed using $[\phi]$ — that is, the encapsulation of ϕ — rather than ϕ.

With this definition, we can now state three frame rules, enabling local reasoning with respect to the parallel product, sequential prefixing, and sequential suffixing.

▶ **Proposition 43** (Frame properties). *If $P \#_{\cong} \phi$, and $Q \models_{\cong} [\phi]$, then it holds that:*

(i) $\forall \psi \in \mathtt{F}_\Sigma,\ P \models_{\cong} \psi \Leftrightarrow P \oplus Q \models_{\cong} \psi \star [\phi]$;

(ii) $\forall \psi \in \mathtt{F}_\Sigma,\ P \models_{\cong} \psi \Leftrightarrow P \otimes Q \models_{\cong} \psi \blacktriangleright [\phi]$;

(iii) $\forall \psi \in \mathtt{F}_\Sigma,\ P \models_{\cong} \psi \Leftrightarrow Q \otimes P \models_{\cong} [\phi] \blacktriangleright \psi$.

Proof. Clearly, if $P \models_{\cong} \psi$, since $Q \models_{\cong} [\phi]$ we have immediately that $P \oplus Q \models_{\cong} \psi \star [\phi]$, $P \otimes Q \models_{\cong} \psi \blacktriangleright [\phi]$, and $P \otimes Q \models_{\cong} [\phi] \blacktriangleright \psi$.

Now assume $P \oplus Q \models_{\cong} \psi \star [\phi]$. This means that there exists P', Q' such that $P \oplus Q \cong P' \oplus Q'$, $P' \models_{\cong} \psi$ and $Q' \models_{\cong} [\phi]$. The factorization $P \oplus Q \cong P' \oplus Q'$ may be further decomposed into pomsets P_1, P_2, Q_1, and Q_2 such that $P \cong P_1 \oplus P_2$, $Q \cong Q_1 \oplus Q_2$, $P_1 \oplus Q_1 \cong P'$, and $P_2 \oplus Q_2 \cong Q'$.

Note that ϕ cannot be satisfied by the empty pomset: otherwise, since the empty pomset is contained in any other pomset, we would have $P \models_{\cong} (\!|[\phi]|\!)$. Therefore since both Q' and Q satisfy $[\phi]$, both have to be non-empty boxes. As a result, we get:

- since $Q \cong Q_1 \oplus Q_2$, either Q_1 or Q_2 have to be empty;
- since $Q' \cong P_2 \oplus Q_2$, either P_2 or Q_2 have to be empty.

If Q_2 is empty, we violate again the hypothesis that P is independent from ϕ, since $P \cong P_1 \oplus P_2 \cong P_1 \oplus Q' \Rsh Q'$, and $Q' \models_{\cong} [\phi]$.

Hence we know that Q_1 and P_2 are empty, meaning that $P \cong P_1 \cong P' \models_{\cong} \psi$.

The same argument works for the sequential product. ◀

▶ Remark 44. Note that this lemma does not hold for $\sqsubseteq$ or $\sqsupseteq$ instead of $\simeq$. The left-to-right implications always hold, but the converse may be fail, as we now demonstrate with some examples.

- For $\sqsubseteq$, consider the following:

$$P := \mathbb{a} \qquad Q := [\mathbb{b} \oplus [\mathbb{c}]] \qquad \phi := (\!|\mathbb{c}|\!) \qquad \psi := a \star b.$$

We may check that in this case we have:

$$P \not\models_{\sqsubseteq} (\!|[\phi]|\!) \qquad Q \models_{\sqsubseteq} [\phi] \qquad P \oplus Q \models_{\sqsubseteq} \psi \star [\phi] \qquad P \not\models_{\sqsubseteq} \psi.$$

This happens because in order to satisfy $\psi \star [\phi]$ we may rearrange the pomsets using the ordering. More precisely, we use the fact that

$$P \oplus Q = \mathbb{a} \oplus [\mathbb{b} \oplus [\mathbb{c}]] \sqsubseteq (\mathbb{a} \oplus \mathbb{b}) \oplus [\mathbb{c}]\,.$$

- For $\sqsupseteq$, we have a similar example. The difference is that instead of Q overspiling into P, we have the converse.

$$P := \mathbb{a} \oplus \mathbb{b} \qquad Q := \mathbb{c} \qquad \phi := (\!|\mathbb{c}|\!) \qquad \psi := \mathbb{a}.$$

Since P does not contain any c, but has more than one event, $P \not\models_{\sqsubseteq} (\!|[\phi]|\!) \vee \psi$. On the other hand $Q \models_{\sqsubseteq} [\phi]$ because $Q \sqsupseteq [Q]$ and $Q \Rsh Q \models_{\sqsupseteq} c$. Finally, $P \oplus Q = \mathbb{a} \oplus \mathbb{b} \oplus \mathbb{c} \sqsupseteq \mathbb{a} \oplus [\mathbb{b} \oplus \mathbb{c}]$, and we have both $\mathbb{a} \models_{\sqsubseteq} \psi$ and $[\mathbb{b} \oplus \mathbb{c}] \models_{\sqsubseteq} [\phi]$.

▶ Remark 45. this principle may be extended to sets of pomsets. Indeed, if we define the independence relation for sets of pomsets as $A \#_R \phi := \forall P \in A,\ P \#_R \phi$, then Proposition 43 holds for both $\models^{\forall}_{\cong}$ and $\models^{\exists}_{\cong}$.

4.3 Example

In this section, we present an example program, and showcase reasoning principles of pomset logic. In particular, we will highlight the use of local reasoning when appropriate.

Consider the following voting protocol: a fixed number of voters, $v_1, \ldots, v_n$, are each asked to increment one of the counters $c_1, \ldots, c_k$. The tally is then sent to each of the v_i, to inform them of the result. The increment is implemented similarly to our running example of the distributed counter (Example 1). The implementation of the protocol is displayed in Figure 6, together with the intended semantics of the atomic actions.

$$\begin{aligned}
\texttt{VoteProc} &:= \texttt{Choose}\,;\texttt{Publish}\\
\texttt{Choose} &:= [\texttt{Vote}\,(1)] \parallel \cdots \parallel [\texttt{Vote}\,(n)]\\
\texttt{Vote}\,(i) &:= \sum_{1\leqslant j\leqslant k} \boxtimes_{i,j} ; \text{⌓}_j ; \text{▟} ; \text{✍}_j\\
\texttt{Publish} &:= \text{☎}_1 \parallel \cdots \parallel \text{☎}_n
\end{aligned}$$

☎_i : the contents of counters $c_1, \ldots, c_k$ is sent to voter v_i
$\boxtimes_{i,j}$: voter v_i chooses counter c_j
⌓_j: the content of counter c_j is loaded into a local variable
▟ : the local variable is incremented
✍_j : the content of the local variable is stored in counter c_j

Figure 6 Voting protocol

Conflict

As in Example 1, if we forgo the boxes in `Choose`, we cannot enforce mutual exclusion. Recall that the undesirable behaviour is captured by the following formula:

$$\mathbf{conflict}_j := (\!|(\text{⌓}_j \star \text{⌓}_j) \blacktriangleright (\text{✍}_j \star \text{✍}_j)|\!)$$

We may see this by defining an alternative (faulty) protocol:

$$\texttt{VoteProc}' := (\texttt{Vote}\,(1) \parallel \cdots \parallel \texttt{Vote}\,(n))\,;\texttt{Publish}$$

and then checking that this protocol displays the behaviour we wanted to avoid:

$$\texttt{VoteProc}' \models^{\exists}_{\sqsupseteq} \mathbf{conflict}_j.$$

This statement should be read as 'there is a pomset in $[\![\texttt{VoteProc}']\!]$ that is larger than one containing a conflict'. We can show the existence of this 'bug' by local reasoning. We may first prove that $\texttt{Vote}\,(i) \parallel \texttt{Vote}\,(i') \models^{\exists}_{\sqsupseteq} \mathbf{conflict}_j$ (for some arbitrary $i \neq i'$). The properties of $(\!|-|\!)$ then allow us to deduce that

$$\texttt{VoteProc}' \cong (\texttt{Vote}\,(i) \parallel \texttt{Vote}\,(i') \parallel \cdots)\,;\cdots \models^{\exists}_{\sqsupseteq} (\!|\mathbf{conflict}_j|\!) \equiv \mathbf{conflict}_j.$$

The implementation in Figure 6 avoids this problem, and indeed it holds that:

$$\texttt{VoteProc} \not\models^{\exists}_{\sqsupseteq} \mathbf{conflict}_j.$$

However, showing that this formula is *not* satisfied by the program is less straightforward and, in particular, cannot be done locally: we have to enumerate all possible sub-pomsets, and check that none provide a suitable witness.

Sequential separation

In our protocol, the results of the vote are only communicated after every participant has voted. This is specified by the following statement:

$$\mathbf{SendAfterVote} := \left(\!\left|\neg\left(\bigvee_i \text{☎}_i\right)\right|\!\right) \blacktriangleright \left(\!\left|\neg\left(\bigvee_{i,j} \boxtimes_{i,j}\right)\right|\!\right).$$

This may be checked modularly. Indeed, one may prove by simple syntactic analysis that

$$\texttt{Choose} \models^{\forall}_{\cong} \left(\!\!\left| \neg \left(\bigvee_i \text{☎}_i \right) \right|\!\!\right) \qquad \text{and} \qquad \texttt{Publish} \models^{\forall}_{\cong} \left(\!\!\left| \neg \left(\bigvee_{i,j} \boxtimes_{i,j} \right) \right|\!\!\right).$$

Therefore, we may combine these to get that:

$$\texttt{VoteProc} = \texttt{Choose}\,;\texttt{Publish} \models^{\forall}_{\cong} \left(\!\!\left| \neg \left(\bigvee_i \text{☎}_i \right) \right|\!\!\right) \blacktriangleright \left(\!\!\left| \neg \left(\bigvee_{i,j} \boxtimes_{i,j} \right) \right|\!\!\right) = \mathbf{SendAfterVote}.$$

For voter i, two of the most meaningful steps are $\boxtimes_{i,j}$ and ☎_i, i.e. when the vote is cast and when the result of the vote is forwarded to them. Using the macro $\mathbf{choose}_i := \bigvee_j \boxtimes_{i,j}$, we can specify that during the protocol, each voter first votes, and then gets send the result:

$$\mathbf{VoteThenSend} := (\!|(\mathbf{choose}_1 \blacktriangleright \text{☎}_1) \star \ldots \star (\mathbf{choose}_n \blacktriangleright \text{☎}_n)|\!).$$

Unique votes

Another important feature of this protocol is that each voter may only cast a single vote. Knowing that each voter controls a single box, we express this property with the statement:

$$\texttt{VoteProc} \not\models^{\exists}_{\sqsubseteq} \bigvee_{j,j'} (\!|[(\!|\text{✍}_j \star \text{✍}_{j'}|\!)]|\!).$$

Since we use the relation $\models^{\exists}_{\sqsubseteq}$ with the connective $\star$, we allow any possible ordering of the two write events. The only constraint is that there should be at least two of them in the same box. As for the 'conflict' property, if the 'bad' behaviour were to happen, one could prove it compositionally. However, disproving the existence of such a behaviour is a more global process, involving the exploration of all possible sub-pomsets.

Frame property

As we have seen in previous examples, proving that a formula does not hold can be challenging, because the non-existence of a local pattern is *not* a local property. We may circumvent this problem by adding more boxes in both programs and formulas. This is related to a common pattern in parallel programming: in a multi-threaded program, one may insert fences to 'tame' concurrency. Doing so simplifies program analysis, at the cost of some efficiency. Similarly, since adding boxes restricts behaviours — thus disallowing some possible optimizations — the analysis of a program becomes simpler and more efficient.

We illustrate this with the following statement:

$$[\texttt{Choose}]\,;[\texttt{Publish}] \not\models^{\forall}_{\cong} (\!|\text{✍} \blacktriangleright \text{✍}|\!) \blacktriangleright [\phi] \qquad \text{where } \phi := \neg\,(\bot \vee (\!|[(\!|\text{✍}|\!)]|\!))\,.$$

$(\!|\text{✍} \blacktriangleright \text{✍}|\!)$ indicates that two 'write' instructions can be executed in sequence, while ϕ denotes a non-empty pomset, not containing any boxes with a 'write' event inside. We can first prove properties of the subprograms:

$$[\texttt{Publish}] \models^{\forall}_{\cong} [\phi] \qquad [\texttt{Choose}] \not\models^{\forall}_{\cong} (\!|[\phi]|\!) \qquad [\texttt{Choose}] \not\models^{\forall}_{\cong} (\!|\text{✍} \blacktriangleright \text{✍}|\!).$$

Since $[\texttt{Choose}] \#_{\cong} \phi$ and $[\texttt{Publish}] \models^{\forall}_{\cong} [\phi]$, we obtain from the frame rule that

$$[\texttt{Choose}]\,;[\texttt{Publish}] \models^{\forall}_{\cong} (\!|\text{✍} \blacktriangleright \text{✍}|\!) \blacktriangleright [\phi] \Leftrightarrow [\texttt{Choose}] \models^{\forall}_{\cong} (\!|\text{✍} \blacktriangleright \text{✍}|\!).$$

Since we have locally disproved the latter, we may deduce that the former does not hold.

5 Future work

In this paper, we have not considered the CKA operator $-^\star$. A natural further step would be to do so, with the corresponding need to consider versions of pomset logic with fixed points. Connections with Hoare-style program logics, such as Concurrent Separation Logic [3, 20] with its concrete semantics, should also be considered.

Our satisfaction relation over pomsets is defined inductively. However, the satisfaction relations we define for sets of pomsets is not: we define in terms of the former relation. For practical purposes, such as model-checking, it would be useful to have a similar inductive definition for sets of pomsets.

It is also worth noticing that the definitions and statements in Section 2 are straightforward generalizations of their counterparts in CKA (without boxes); even the proofs of those results follow a similar strategy. However, we could reuse almost no result from CKA: instead we had to reprove everything from scratch. This situation is deeply unsatisfactory, and we plan on investigating techniques to better 'recycle' proofs in this context. Recent work on (C)KA with *hypotheses* [6, 14] seems to be a step towards this goal.

References

1 Gabrielle Anderson and David Pym. A calculus and logic of bunched resources and processes. *Theoretical Computer Science*, 614, 2016. doi:10.1016/j.tcs.2015.11.035.

2 Johan Van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2014.

3 Stephen Brookes and Peter W. O'Hearn. Concurrent Separation Logic. *ACM SIGLOG News*, 3(3), August 2016. doi:10.1145/2984450.2984457.

4 Paul Brunet, Damien Pous, and Georg Struth. On Decidability of Concurrent Kleene Algebra. In *International Conference on Concurrency Theory*, 2017. doi:10.4230/LIPIcs.CONCUR.2017.28.

5 Matthew Collinson and David Pym. Algebra and Logic for Resource-Based Systems Modelling. *Mathematical Structures in Computer Science*, 19(5), 2009. doi:10.1017/S0960129509990077.

6 Amina Doumane, Denis Kuperberg, Damien Pous, and Pierre Pradic. Kleene Algebra with Hypotheses. In *International Conference on Foundations of Software Science and Computational Structures*, 2019. doi:10.1007/978-3-030-17127-8_12.

7 Didier Galmiche, Daniel Méry, and David Pym. The Semantics of BI and Resource Tableaux. *Mathematical Structures in Computer Science*, 15(6), December 2005. doi:10.1017/S0960129505004858.

8 Jay L. Gischer. The Equational Theory of Pomsets. *Theoretical Computer Science*, 61(2-3), 1988. doi:10.1016/0304-3975(88)90124-7.

9 Jan Grabowski. On partial languages. *Fundamenta Mathematicae*, 4(2), 1981.

10 Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In *International Colloquium on Automata, Languages and Programming*, 1980. doi:10.1007/3-540-10003-2_79.

11 C.A.R. Hoare, Bernhard Möller, Georg Struth, and Ian Wehrman. Concurrent Kleene Algebra. In *International Conference on Concurrency Theory*, 2009. doi:10.1007/978-3-642-04081-8_27.

12 Samin S. Ishtiaq and Peter W. O'Hearn. BI as an Assertion Language for Mutable Data Structures. In *Symposium on Principles of Programming Languages*, 2001. doi:10.1145/373243.375719.

13 Lalita Jategaonkar and Albert R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1), 1996. doi:10.1016/0304-3975(95)00132-8.

14 Tobias Kappé, Paul Brunet, Alexandra Silva, Jana Wagemaker, and Fabio Zanasi. Concurrent Kleene Algebra with Observations: From Hypotheses to Completeness. In *International Conference on Foundations of Software Science and Computational Structures*, 2020. doi:"10.1007/978-3-030-45231-5_20".

15 Tobias Kappé, Paul Brunet, Alexandra Silva, and Fabio Zanasi. Concurrent Kleene Algebra: Free Model and Completeness. In *European Symposium on Programming*, 2018. doi:10.1007/978-3-319-89884-1_30.

16 Casimir Kuratowski. Sur l'opération Ā de l'Analysis Situs. *Fundamenta Mathematicae*, 3(1), 1922.

17 Michael R. Laurence and Georg Struth. Completeness Theorems for Bi-Kleene Algebras and Series-Parallel Rational Pomset Languages. In *International Conference on Relational and Algebraic Methods in Computer Science*, 2014. doi:10.1007/978-3-319-06251-8_5.

18 Robin Milner. *Communication and Concurrency*. Prentice-Hall, Inc., 1989.

19 Peter W. O'Hearn. Resources, Concurrency, and Local Reasoning. *Theoretical Computer Science*, 375(1-3), May 2007. doi:10.1016/j.tcs.2006.12.035.

20 Peter W. O'Hearn, Rasmus L. Petersen, Jules Villard, and Akbar Hussain. On the Relation between Concurrent Separation Logic and Concurrent Kleene Algebra. *Journal of Logical and Algebraic Methods in Programming*, 84(3), May 2015. doi:10.1016/j.jlamp.2014.08.002.

21 Peter W. O'Hearn and David J. Pym. The Logic of Bunched Implications. *Bulletin of Symbolic Logic*, 5(2), 1999. doi:10.2307/421090.

22 Peter W. O'Hearn, John C. Reynolds, and Hongseok Yang. Local Reasoning about Programs That Alter Data Structures. In *International Conference on Computer Science Logic*, 2001. doi:10.1007/3-540-44802-0_1.

23 David Pym. Resource Semantics: Logic as a Modelling Technology. *ACM SIGLOG News*, 6(2), April 2019. doi:10.1145/3326938.3326940.

24 Mohammad Raza and Philippa Gardner. Footprints in Local Reasoning. In *International Conference on Foundations of Software Science and Computational Structures*, 2008. doi:10.1007/978-3-540-78499-9_15.

25 Christian Retoré. Pomset logic: A non-commutative extension of classical linear logic. In *Typed Lambda Calculi and Applications*, 1997. doi:10.1007/3-540-62688-3_43.

26 John C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Symposium on Logic in Computer Science*, July 2002. doi:10.1109/LICS.2002.1029817.

27 Jacobo Valdes, Robert E. Tarjan, and Eugene L. Lawler. The Recognition of Series Parallel Digraphs. *SIAM J. Comput.*, 11(2), 1982. doi:10.1137/0211023.

A Proof of Lemma 40

▶ **Lemma 40.** *For any term $s \in \mathtt{SP}_\Sigma$ and any poset P, we have that*

$$P \models_{\cong} \phi(s) \Leftrightarrow P \cong [\![s]\!] \qquad P \models_{\sqsupseteq} \phi(s) \Leftrightarrow P \sqsupseteq [\![s]\!] \qquad P \models_{\sqsubseteq} \phi(s) \Leftrightarrow P \sqsubseteq [\![s]\!]\,.$$

For a term $e \in \mathtt{T}_\Sigma$ and a set of posets $X \subseteq \mathbb{P}_\Sigma$, we have that

$$X \models^{\forall}_{\cong} \Phi(e) \Leftrightarrow X \subsetneq [\![e]\!] \qquad X \models^{\forall}_{\sqsubseteq} \Phi(e) \Leftrightarrow X \sqsubseteq [\![e]\!]\,.$$

Proof. By induction on s:

$s = 1$:
$$\begin{aligned} P \models_{\cong} \phi(1) = \bot &\Leftrightarrow P \cong \mathbb{c} = [\![1]\!]\,. \\ P \models_{\sqsupseteq} \phi(1) = \bot &\Leftrightarrow P \cong \mathbb{c} \Leftrightarrow P \sqsupseteq \mathbb{c} = [\![1]\!]\,. \\ P \models_{\sqsubseteq} \phi(1) = \bot &\Leftrightarrow P \cong \mathbb{c} \Leftrightarrow P \sqsubseteq \mathbb{c} = [\![1]\!]\,. \end{aligned}$$

$s = a$:
$$\begin{aligned} P \models_{\cong} \phi(a) = a &\Leftrightarrow P \cong \mathbb{a} = [\![a]\!]\,. \\ P \models_{\sqsupseteq} \phi(a) = a &\Leftrightarrow P \sqsupseteq \mathbb{a} = [\![a]\!]\,. \\ P \models_{\sqsubseteq} \phi(a) = a &\Leftrightarrow P \sqsubseteq \mathbb{a} = [\![a]\!]\,. \end{aligned}$$

$s = [t]$:
$$\begin{aligned} P \models_{\cong} [\phi(t)] &\Leftrightarrow P \cong [Q] \wedge Q \models_{\cong} \phi(t) \\ &\Leftrightarrow P \cong [Q] \wedge Q \cong [\![t]\!] \\ &\Leftrightarrow P \cong [[\![t]\!]] = [\![[t]]\!]\,. \\ P \models_{\sqsupseteq} [\phi(t)] &\Leftrightarrow P \sqsupseteq [Q] \wedge Q \models_{\sqsupseteq} \phi(t) \\ &\Leftrightarrow P \sqsupseteq [Q] \wedge Q \sqsupseteq [\![t]\!] \\ &\Leftrightarrow P \sqsupseteq [[\![t]\!]] = [\![[t]]\!]\,. \\ P \models_{\sqsubseteq} [\phi(t)] &\Leftrightarrow P \sqsubseteq [Q] \wedge Q \models_{\sqsubseteq} \phi(t) \\ &\Leftrightarrow P \sqsubseteq [Q] \wedge Q \sqsubseteq [\![t]\!] \\ &\Leftrightarrow P \sqsubseteq [[\![t]\!]] = [\![[t]]\!]\,. \end{aligned}$$

$s = s_1 \,;\, s_2$:
$$\begin{aligned} P \models_{\cong} \phi(s_1) \blacktriangleright \phi(s_2) &\Leftrightarrow P \cong P_1 \otimes P_2 \\ &\quad \wedge P_1 \models_{\cong} \phi(s_1) \wedge P_2 \models_{\cong} \phi(s_2) \\ &\Leftrightarrow P \cong P_1 \otimes P_2 \wedge P_1 \cong [\![s_1]\!] \wedge P_2 \cong [\![s_2]\!] \\ &\Leftrightarrow P \cong [\![s_1]\!] \otimes [\![s_2]\!] = [\![s_1 \,;\, s_2]\!]\,. \\ P \models_{\sqsupseteq} \phi(s_1) \blacktriangleright \phi(s_2) &\Leftrightarrow P \sqsupseteq P_1 \otimes P_2 \\ &\quad \wedge P_1 \models_{\sqsupseteq} \phi(s_1) \wedge P_2 \models_{\sqsupseteq} \phi(s_2) \\ &\Leftrightarrow P \sqsupseteq P_1 \otimes P_2 \wedge P_1 \sqsupseteq [\![s_1]\!] \wedge P_2 \sqsupseteq [\![s_2]\!] \\ &\Leftrightarrow P \sqsupseteq [\![s_1]\!] \otimes [\![s_2]\!] = [\![s_1 \,;\, s_2]\!]\,. \\ P \models_{\sqsubseteq} \phi(s_1) \blacktriangleright \phi(s_2) &\Leftrightarrow P \sqsubseteq P_1 \otimes P_2 \\ &\quad \wedge P_1 \models_{\sqsubseteq} \phi(s_1) \wedge P_2 \models_{\sqsubseteq} \phi(s_2) \\ &\Leftrightarrow P \sqsubseteq P_1 \otimes P_2 \wedge P_1 \sqsubseteq [\![s_1]\!] \wedge P_2 \sqsubseteq [\![s_2]\!] \\ &\Leftrightarrow P \sqsubseteq [\![s_1]\!] \otimes [\![s_2]\!] = [\![s_1 \,;\, s_2]\!]\,. \end{aligned}$$

$s = s_1 \parallel s_2$:
$$\begin{aligned} P \models_{\cong} \phi(s_1) \star \phi(s_2) &\Leftrightarrow P \cong P_1 \oplus P_2 \\ &\quad \wedge P_1 \models_{\cong} \phi(s_1) \wedge P_2 \models_{\cong} \phi(s_2) \\ &\Leftrightarrow P \cong P_1 \oplus P_2 \wedge P_1 \cong [\![s_1]\!] \wedge P_2 \cong [\![s_2]\!] \\ &\Leftrightarrow P \cong [\![s_1]\!] \oplus [\![s_2]\!] = [\![s_1 \parallel s_2]\!]\,. \end{aligned}$$

$$
\begin{aligned}
P \models_{\sqsupseteq} \phi(s_1) \star \phi(s_2) &\Leftrightarrow P \sqsupseteq P_1 \oplus P_2 \\
&\quad \wedge P_1 \models_{\sqsupseteq} \phi(s_1) \wedge P_2 \models_{\sqsupseteq} \phi(s_2) \\
&\Leftrightarrow P \sqsupseteq P_1 \oplus P_2 \wedge P_1 \sqsupseteq [\![s_1]\!] \wedge P_2 \sqsupseteq [\![s_2]\!] \\
&\Leftrightarrow P \sqsupseteq [\![s_1]\!] \oplus [\![s_2]\!] = [\![s_1 \parallel s_2]\!] \,. \\
P \models_{\sqsubseteq} \phi(s_1) \star \phi(s_2) &\Leftrightarrow P \sqsubseteq P_1 \oplus P_2 \\
&\quad \wedge P_1 \models_{\sqsubseteq} \phi(s_1) \wedge P_2 \models_{\sqsubseteq} \phi(s_2) \\
&\Leftrightarrow P \sqsubseteq P_1 \oplus P_2 \wedge P_1 \sqsubseteq [\![s_1]\!] \wedge P_2 \sqsubseteq [\![s_2]\!] \\
&\Leftrightarrow P \sqsubseteq [\![s_1]\!] \oplus [\![s_2]\!] = [\![s_1 \parallel s_2]\!] \,.
\end{aligned}
$$

Recall that since $_\uparrow$ and $_\downarrow$ are Kuratowski closure operators, they distribute over unions. We may thus obtain:

$$
\begin{aligned}
[\![\Phi(e)]\!]_{\cong} &= \bigcup_{s \in T_e} [\![\phi(s)]\!]_{\cong} \cong \bigcup_{s \in T_e} \{[\![s]\!]\} \cong [\![e]\!] \,. \\
[\![\Phi(e)]\!]_{\sqsubseteq} &= \bigcup_{s \in T_e} [\![\phi(s)]\!]_{\sqsubseteq} \cong \bigcup_{s \in T_e} \{[\![s]\!]\}\downarrow \cong [\![e]\!]\downarrow .
\end{aligned}
$$

The statements then follow by (3.5) and (3.6). ◀