Draft (17 September 2001)
55 pages

A Truly Concurrent Semantics for a Process Algebra Using Resource Pomsets

Paul Gastin

LIAFA, Université Paris 7, 2 Place Jussieu, F-75251 Paris Cedex 05

Michael Mislove [1]

Department of Mathematics, Tulane University, New Orleans, LA 70118

Abstract

In this paper we study a process algebra whose semantics is based on true concurrency. In our model, actions are defined in terms of the resources they need to execute, which allows a simple definition of a weak sequential composition operator. This operator allows actions which do not share any resources to execute concurrently, while dependent actions have to occur sequentially. This weak sequential composition operator may be used to automatically parallelize a sequential process. We add the customary (strict) sequential composition and a parallel composition operator allowing synchronization on specified actions. Our language also supports a hiding operator that allows the hiding of actions and even of individual resources used by actions. Strict sequential composition and hiding require that we generalize from the realm of Mazurkiewicz traces to that of pomsets, since these operations introduce "over-synchronized" traces – ones for which a pair of independent actions may occur sequentially. Our language also supports recursion and our semantics makes the unwinding of recursion visible by the use of special resources used to label unwindings. This is done on purpose in order to make divergence observable, but the usual semantics that does not observe unwindings can be obtained by using the hiding operator to abstract away from these special resources. We give both an SOS-style operational semantics for our language, as well as a denotational semantics based on resource pomsets. Generalizing results from our earlier work in this area, we derive a congruence theorem for our language which shows that the SOS-style operational rules induce the same equivalence relation on the language as the denotational semantic map does. A corollary is that our denotational model is both adequate and fully abstract relative to the behavior function defined from our operational semantics. This behavior consists naturally of the strings of actions the process can perform. This work continues our study into modelling concurrency in the absence of nondeterminism. In particular, our language is deterministic.

---

1 Introduction

The standard process algebraic approach to concurrent computation relies on two ingredients: sequential composition and nondeterminism. These are used to give the semantics of (asynchronous) parallel composition using the interleaving of all actions of the component processes. Synchronous parallel composition then can be derived by pruning some of the asynchronous interleavings (as in the case of CSP [2]), or by adding specific synchronous parallel compositions (as in CCS [11]). There are many successes in this approach, including perhaps the two most notable ones in process algebra, CSP and CCS.

An alternative approach to modelling concurrent computation is to use partially ordered actions instead of the usual streams of actions. In this paper, we explore such an approach, which relies on using resources to define when actions can execute concurrently. A basic operator is one we call *weak sequential composition* that allows actions which share no resources to execute concurrently, while requiring those that have some resources in common to execute in the order in which they were written. The language we study also includes a number of other operators, including the more standard (strict) sequential composition operator, a hiding operator that allows us to hide resources and the actions they define from the environment, and a parallel composition operator that requires synchronization on certain prescribed actions. Our language also supports recursion and the unwinding of recursion is made visible by the use of special resources that label unwindings. This is done on purpose in order to be able to observe divergence, but the usual semantics that does not observe unwindings can be retrieved by using the hiding operator to abstract away from these special resources. We do not include nondeterminism in our language – all processes are deterministic. This is due in part to our desire to explore concurrency without nondeterminism to understand better the relationship between parallel composition and nondeterminism.

From a mathematical viewpoint, the results we derive extend our previous work [5,6] to include what we call *resource pomsets*. These are used to give a denotational model for our language, and they require an extension to pomsets of many of the results in [7]. The earlier approach relied on Mazurkiewicz traces in which all causal orderings between actions could be accounted for simply by whether the actions share some resources. But with a hiding operator present, this is no longer the case: by hiding an action which executes between two independent actions, but on which both of them are dependent, we can find execution sequences in which the order cannot be accounted for by resource sharing. This is accommodated by using pomsets instead of Mazurkiewicz traces to give the semantics – such pomsets prevent dependent actions from occurring concurrently, but still allow independent

[1] The support of the National Science Foundation and of the US Office of Naval Research is gratefully acknowledged.

actions to be sequentially ordered.

We include in our study both an operational semantics for our language in the now-customary SOS style, and a denotational model based on resource pomsets. One of our main results is a congruence theorem which shows that the mapping that assigns to each process its behavior according to the operational rules gives the same result as the denotational mapping which assigns to each process its resource pomset. In particular, this theorem shows we can recover the complex pomset that is the denotational meaning of a process only knowing the set of strings of actions the process can execute, and the resulting process after the execution of each such string (Definition 5.15 and Theorem 5.16). This immediately implies the denotational model is both adequate and fully abstract with respect to the operational semantics. We also extend this by showing that our denotational model is in fact fully abstract with respect to the operational behavior that extracts only the set of strings a process executes, instead of also observing the resources still claimed by the process after executing each string of actions. These results hold for a very large class of closed processes that we call *nice*. We also include several examples of interesting processes, the most notable of which is an n-place buffer.

This work is part of an on-going program the authors initiated to study the notion of "true concurrency" and to better understand the relationship between concurrency and nondeterminism. In an earlier paper, we considered a language which was similar to the one presented here [5,6]. What is most notable about the current language is the inclusion of a hiding operator. And, while the earlier language considered a restriction operator that allowed the restriction of processes to specific sets of resources, we have not included that operator here because it can be derived using parallel composition and hiding (see the discussion in Section 6). What remains after this work is to add choice operators to the language, and, as we comment in Section 6, it is our intention next to pursue incorporating such operators into the language considered here. But we also feel that the subtleties of a proper treatment of hiding merit separate presentation.

Other authors have considered structures that are identified with "true concurrency" – notably, event structures [19] and pomsets [14,15] – as models of concurrency, but none that we know of has actually considered a specific language and attempted the sort of analysis – operational and denotational – presented here. Indeed, the other presentations with which we are familiar are operational in nature, and do not present or consider a specific process algebra, let alone define for it a compositional semantics and then show it is related to the operational semantics being studied. In any case, all this work – ours included – has its origins in Mazurkiewicz trace theory [10] and Petri nets [17]. We believe it was the lack of a continuous interpretation of the concatenation operator from trace theory that prevented such an approach as we present here. That barrier was lifted only with the work in [3,7]. In any case, we believe that, along with [5,6], this represents the first presentation of a truly concurrent denotational semantics for a language including parallel composition and weak sequential composition.

The outline of the paper is as follows. In the next section, we present some background material on domain theory that we need, and then we lay out the theory of resource pomsets which is used to define the denotational model for our language. Following this is the language, as well as its operational semantics and some examples showing what can be expressed with this language. After that the denotational model is described, and the following section contains a proof of the congruence theorem relating the operational and denotational models. The proof relies on two fundamental results: for a process p,

- any sequence u of actions that p can execute, perhaps interspersed with invisible transitions (denoted $p \stackrel{u}{\Longrightarrow}$), is in fact a linearization of a prefix of the denotational meaning $[\![p]\!]$ of p, and conversely
- if u is a linearization of some finite prefix of $[\![p]\!]$, then $p \stackrel{u}{\Longrightarrow}$.

The proof of the last result utilizes a rank function whose main point is to differentiate hiding from the other operators of the language. With these results in hand, the congruence theorem, and the corollaries of adequacy and full abstraction are easily derived. The final section contains a summary and some comments about future work.

2 Some Preliminary Results

2.1 Domain Theory

A *poset* $(P, \leq)$ is a non-empty set P endowed with a partial order $\leq$. The least element of P (if it exists) is denoted $\perp$, and a subset $D \subseteq P$ is *directed* if each finite subset $F \subseteq D$ has an upper bound in D. Note that since $F = \emptyset$ is a possibility, a directed subset must be non-empty. A *(directed) complete partial order (dcpo)* is a poset P in which each directed set $D \subseteq P$ has a least upper bound, denoted $\sqcup D$. If P also has a least element, then it is called a *cpo*. A poset is *consistently complete* if every non-empty subset X for which each pair of elements has an upper bound in P has a least upper bound, $\sqcup X$, and P is *bounded complete* if every non-empty subset having an upper bound has a least upper bound.

If P is a poset, the element $k \in P$ is *compact* if, for each directed subset $D \subseteq P$, if $\sqcup D$ exists and $k \leq \sqcup D$, then $(\exists d \in D)\ k \leq d$. The set of compact elements of P is denoted $K(P)$, and for an element $x \in P$, $K(x) = K(P) \cap {\downarrow} x$, where ${\downarrow} x = \{y \in P \mid y \leq x\}$. P is *algebraic* if $K(x)$ is directed and $x = \sqcup K(x)$ for each $x \in P$.

If P and Q are posets and $f : P \to Q$ is a monotone map, then f is *(Scott) continuous* if f preserves least upper bounds of directed sets: if $D \subseteq P$ is directed and $\sqcup D \in P$ exists, then $\sqcup f(D) \in Q$ exists and $f(\sqcup D) = \sqcup f(D)$. The following result is a simple application of the Tarski-Knaster-Scott Fixed Point Theorem, applied to the sub-dcpo $\uparrow x = \{y \in P \mid x \leq y\}$ of the dcpo P.

Proposition 2.1 *Let $f : P \to P$ be a continuous function defined on a dcpo P. If $x \in P$ satisfies $x \leq f(x)$, then f has a least fixed point $\mathrm{fix}_{\uparrow x} f$ in $\uparrow x$ defined by $\mathrm{fix}_{\uparrow x}(f) = \sqcup_{n \geq 0} f^n(x)$.* □

2.2 Resource Pomsets

We now develop a theory of *resource pomsets* which generalizes many of the results of [7] to the setting of pomsets. We restrict our presentation to the special situation in which the pomsets are labelled with sets of resources. Many of the same results hold in a more general setting, but we are more interested in presenting a coherent theory restricted to our current needs than in presenting the most general results.

We begin by assuming $\mathcal{R}$ is a non-empty, countable set of *resources*. We let $\mathcal{P}_f(\mathcal{R}) = \{F \subseteq \mathcal{R} \mid F \text{ is finite}\}$. We then fix a non-empty set Σ of *actions*: $\emptyset \neq \Sigma \subseteq \{a \in \mathcal{P}_f(\mathcal{R}) \mid a \neq \emptyset\}$. We say two elements $a, b \in \Sigma$ are *dependent* if $a \cap b \neq \emptyset$; otherwise, a and b are *independent*, which we denote by $a\, I\, b$. Note that, since an action is a non-empty set of resources, it is not independent of itself. Therefore, there is no auto-concurrency.

We start with the definition of *real pomsets* and the presentation of their main properties. Then, we turn to the more general notion of *complex pomsets*.

Real pomsets

In all generality, a *pomset* is a finite or infinite labelled partial order. In this paper, we are only interested in special kinds of pomsets that we call *real pomsets*.

Definition 2.2 We define a *real pomset* (or simply *pomset*) over Σ to be the isomorphism class of a labelled partial order $t = (V, \leq, \lambda)$, where

- V is a countable set of *events*,
- $\leq$ is a partial order called the *causality relation* on V satisfying $\downarrow p$ finite for each $p \in V$, and
- $\lambda : V \to \Sigma$ is a node-labelling satisfying the *over-synchronization condition*

$$\forall p, q \in V, \qquad \lambda(p) \cap \lambda(q) \neq \emptyset \;\Rightarrow\; p \leq q \text{ or } q \leq p.$$

The *length* of a real pomset t is $|t| = |V|$. A real pomset is *finite* if its length is finite. The set of real pomsets over Σ is denoted by $\mathbb{R}(\Sigma)$, or simply $\mathbb{R}$, and we let $\mathbb{R}_f$ denote the subfamily of finite pomsets over Σ. We define the *resource mapping* $\mathrm{res} : \mathbb{R} \to \mathcal{P}(\mathcal{R})$ by $\mathrm{res}((V, \leq, \lambda)) = \bigcup \lambda(V)$ and we generalize the independence relation to real pomsets $s, t \in \mathbb{R}$ by $s\, I\, t$ if $\mathrm{res}(s) \cap \mathrm{res}(t) = \emptyset$.

For instance, if $a, b, c, d \in \Sigma$ with b, c independent, then the following finite real pomset represents the semantics of $a \,;\, (b \parallel c) \,;\, d$ where ; and $\parallel$ stand for *strict* composition and *parallel* composition.

$$\left(\begin{tikzcd}[row sep=tiny, column sep=small] & b \arrow[dr] & \\ a \arrow[ur] \arrow[dr] & & d \\ & c \arrow[ur] & \end{tikzcd}\right)$$

The two linearizations of this pomset are *abcd* and *acbd* and correspond to the maximal sequential executions of $a \,;\, (b \parallel c) \,;\, d$.

We give another example. Let $a, b \in \Sigma$ be independent actions, then

$$s = \left(\begin{tikzcd}[row sep=small, column sep=tiny] a \arrow[rr] \arrow[dr] & & a \arrow[rr] \arrow[dr] & & a \arrow[rr] \arrow[dr] & & a \arrow[rr] \arrow[dr] & & a \arrow[dr] & & \cdots & \\ & b \arrow[rr] \arrow[urrr] & & b \arrow[rr] \arrow[urrr] & & b \arrow[rr] \arrow[urrr] & & b \arrow[rr] & & b & & \cdots \end{tikzcd}\right)$$

is an infinite real pomset. Intuitively, it depicts the behavior of a 2-place buffer where a stands for putting a message in the buffer and b stands for getting a message from the buffer.

For an action $a \in \Sigma$, the number of occurrences of a in $t = (V, \leq, \lambda)$ is $|t|_a = |\lambda^{-1}(a)|$, and, for a resource $\rho \in \mathcal{R}$, the number of occurrences of ρ in t is $|t|_\rho = |\{p \in V \mid \rho \in \lambda(p)\}|$. The *alphabet* of t is $\mathrm{alph}(t) = \lambda(V)$.

For several results, it is very useful to use the *canonical representation* of a real pomset that we define now. For $t = (V, \leq, \lambda) \in \mathbb{R}$ and $p \in V$, we let $\nu(p) = |\{q \in V \mid q < p \text{ and } \lambda(q) = \lambda(p)\}|$ and $\gamma(p) = (\lambda(p), \nu(p)) \in \Sigma \times \mathbb{N}$. Thanks to the over-synchronization condition, the mapping γ is injective. The canonical representation of t is $(V_t, \leq_t)$ where

- $V_t = \gamma(V) = \{(a, i) \mid 0 \leq i < |t|_a\} \subseteq \Sigma \times \mathbb{N}$, and
- $\leq_t = \gamma(\leq)$, that is, $(a, i) \leq_t (b, j)$ if $p \leq q$, where $\gamma(p) = (a, i)$ and $\gamma(q) = (b, j)$.

For instance, the canonical representation of the pomset s presented above is

$$s = \left(\begin{tikzcd}[row sep=small, column sep=tiny] (a,0) \arrow[rr] \arrow[dr] & & (a,1) \arrow[rr] \arrow[dr] & & (a,2) \arrow[rr] \arrow[dr] & & (a,3) \arrow[rr] \arrow[dr] & & (a,4) \arrow[dr] & & \cdots & \\ & (b,0) \arrow[rr] \arrow[urrr] & & (b,1) \arrow[rr] \arrow[urrr] & & (b,2) \arrow[rr] \arrow[urrr] & & (b,3) \arrow[rr] & & (b,4) & & \cdots \end{tikzcd}\right)$$

Assuming real pomsets are in their canonical representation allows us to elide the mention of the labelling function λ, or, equivalently, to use the universal labelling $\lambda : \Sigma \times \mathbb{N} \to \Sigma$ defined by $\lambda(a, i) = a$.

Definition 2.3 Let $t_i = (V_i, \leq_i)$, for $i = 1, 2$ be the canonical representations of two real pomsets. We define the prefix relation by

$$t_1 \leq t_2 \quad \text{if and only if} \quad V_1 \subseteq V_2, V_1 = {\downarrow_{\leq_2}} V_1 \text{ and } \leq_1 = \leq_2|_{V_1 \times V_1},$$

where ${\downarrow_{\leq_2}} V_1 = \{p \in V_2 \mid (\exists q \in V_1)\; p \leq_2 q\}$.

The empty pomset, denoted by 1, is the least element of $\mathbb{R}$ for the prefix order. As a result of the canonical representation, it is clear that, if $t_1 \leq t_2$ are real pomsets, then we can define

$$t_1^{-1}t_2 = (V_2 \setminus V_1, \leq_2|_{(V_2 \setminus V_1)\times(V_2 \setminus V_1)})$$

which is again a real pomset.

For instance, $r = \left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & & a \\ & b \arrow[ur] & \\ c \arrow[ur] & & \end{tikzcd}\right)$ is a prefix of $t = \left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & & a \arrow[r] & a \\ & b \arrow[ur] \arrow[dr] & & \\ c \arrow[ur] & & c & \end{tikzcd}\right)$ and the corresponding suffix is $r^{-1}t = \left(\begin{smallmatrix} a \\ c \end{smallmatrix}\right)$. On the other hand, $(a \to b \to a)$ is not a prefix of t. Moreover, t is the least upper bound of $\left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & & a \arrow[r] & a \\ & b \arrow[ur] & & \\ c \arrow[ur] & & & \end{tikzcd}\right)$ and $\left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & & \\ & b \arrow[dr] & \\ c \arrow[ur] & & c \end{tikzcd}\right)$. Also, the infinite real pomset s that corresponds to a 2-place buffer is the least upper bound of the sequence $(s_n)_{n\geq 0}$ where for instance, $s_5 = \left(\begin{tikzcd}[column sep=small] a \arrow[r] \arrow[d] & a \arrow[r] \arrow[d] & a \arrow[r] \arrow[d] & a \arrow[r] \arrow[d] & a \arrow[d] \\ b \arrow[r] \arrow[urr] & b \arrow[r] \arrow[urr] & b \arrow[r] \arrow[urr] & b \arrow[r] & b \end{tikzcd}\right)$. The formal definition of s_n should be clear from this example. The pomset s is also the least upper bound of the sequence $(r_n)_{n\geq 0}$ where for instance, $r_3 = \left(\begin{tikzcd}[column sep=small] a \arrow[r] \arrow[d] & a \arrow[r] \arrow[d] & a \arrow[r] \arrow[d] & a \arrow[r] & a \\ b \arrow[r] \arrow[urr] & b \arrow[r] \arrow[urr] & b \arrow[urr] & & \end{tikzcd}\right)$.

Lemma 2.4 *Let $r, s, t \in \mathbb{R}$ be real pomsets with $r \leq t$ and $s \leq t$. Then, $r \leq s$ if and only if $V_r \subseteq V_s$.*

Proof. One direction is clear. Conversely, assume that $V_r \subseteq V_s$. If $p, q \in V_r \subseteq V_s \subseteq V_t$ are such that $p \leq_r q$, then $r \leq t$ implies $p \leq_t q$ and $s \leq t$ implies $p \leq_s q$. Therefore, $\leq_r \subseteq \leq_s$. Now, let $p \in V_s$ and $q \in V_r$ be such that $p \leq_s q$. From $s \leq t$ we deduce that $p, q \in V_t$ and $p \leq_t q$. Now, from $r \leq t$ we get $p, q \in V_r$ and $p \leq_r q$. Therefore, $V_r = \downarrow_{\leq_s} V_r$ and $\leq_r = \leq_s|_{V_r \times V_r}$. □

Proposition 2.5 $(\mathbb{R}, \leq)$ *is a consistently complete partial order. In fact, if $T \subseteq \mathbb{R}$ is consistent, then $\sqcup T = (\bigcup_{t\in T} V_t, \bigcup_{t\in T} \leq_t)$.*

Proof. It is easy to show that $\leq$ is a partial order on $\mathbb{R}$.

Now, let T be a non-empty, pairwise consistent family of real pomsets. For each $t \in T$, we let $t = (V_t, \leq_t)$ denote the canonical representation for t. We then define $V = \bigcup_{t\in T} V_t$ and $\leq = \bigcup_{t\in T} \leq_t$. We first establish the following

Claim: If $p \in V$, $q \in V_t$ and $p \leq q$, then $p \in V_t$ and $p \leq_t q$.

Proof of Claim: By definition of $\leq$, $p \leq q$ implies $p, q \in V_{t'}$ and $p \leq_{t'} q$ for some $t' \in T$. Since T is consistent, there is some $s \in \mathbb{R}$ with $t, t' \leq s$. Then $p \leq_{t'} q$ implies $p \leq_s q$. Next, $t \leq s$ and $q \in V_t$ implies $p \in V_t$ and $p \leq_t q$. □

Now we show $(V, \leq)$ is a real pomset:

- $\leq$ clearly is reflexive.

- Suppose that $p \leq q \leq p$ with $p, q \in V$. Let $t \in T$ with $p \in V_t$. Then the Claim implies $p, q \in V_t$ and $p \leq_t q \leq_t p$, so $p = q$.
- To show transitivity, let $p, q, r \in V$ with $p \leq q \leq r$. Again, choosing $t \in T$ with $r \in V_t$ and applying the Claim implies $p, q, r \in V_t$ and $p \leq_t q \leq_t r$, so $p \leq_t r$. But then $p \leq r$ by definition of $\leq$.
- For each $p \in V$, there is some $t \in T$ with $p \in V_t$, and then the Claim implies $\downarrow_{\leq} p = \downarrow_{\leq_t} p$, which is finite since t is a real pomset.
- Suppose that $\lambda(p) \cap \lambda(q) \neq \emptyset$ for some $p, q \in V$. Let $t, t' \in T$ with $p \in V_t$ and $q \in V_{t'}$. Again we choose $s \in \mathbb{R}$ with $t, t' \leq s$, and then $p, q \in V_s$. Then $\lambda(p) \cap \lambda(q) \neq \emptyset$ implies that either $p \leq_s q$ or $q \leq_s p$. Since $t, t' \leq s$, it follows $p \leq_{t'} q$ or $q \leq_t p$, which implies $p \leq q$ or $q \leq p$. Hence the over-synchronization condition is fulfilled.

Now if $t \in T$, then $V_t \subseteq V$ and $\leq_t \subseteq \leq$. Using the Claim, we get $V_t = \downarrow_{\leq} V_t$ and $\leq_t = \leq|_{V_t \times V_t}$. Hence, t is a prefix of $(V, \leq)$ for all $t \in T$ and $(V, \leq)$ is an upper bound for T.

Finally, suppose that $t' = (V', \leq') \in \mathbb{R}$ is an upper bound of T. Then $V_t \subseteq V'$ and $\leq_t \subseteq \leq'$ for all $t \in T$. Therefore, $V \subseteq V'$ and $\leq \subseteq \leq'$. If $p \in V'$, $q \in V$ with $p \leq' q$, then there is some $t \in T$ with $q \in V_t$, and then $p \leq_t q$ since $t \leq t'$. Then $p \in V$ and $p \leq q$, so $V = \downarrow_{\leq'} V \subseteq V'$ and $\leq = \leq'|_{V \times V}$. This means $(V, \leq) = \sqcup T$. $\square$

Remark 2.6 As a trivial corollary of Proposition 2.5, we see that every non-empty set T of real pomsets has a greatest lower bound, namely, the least upper bound of the (obviously) consistent set of lower bounds of T. For example, the greatest lower bound of the two real pomsets $(a, 1) \to (b, 1)$ and $(b, 1) \to (a, 1)$ is the empty pomset. On the other hand, the greatest lower bound of $\left(\begin{array}{cccc} a \searrow & & & \\ & b \nearrow & a \to a & \\ c \nearrow & & & \end{array}\right)$ and $\left(\begin{array}{ccc} a \searrow & & \\ & b \searrow & \\ c \nearrow & & c \end{array}\right)$ is the intersection of the two pomsets: $\left(\begin{array}{cc} a \searrow & \\ & b \\ c \nearrow & \end{array}\right)$. The following result shows that this special form holds for the greatest lower bound of T when T is a non-empty set with an upper bound.

Proposition 2.7 *If T is a non-empty family of real pomsets with an upper bound, then $\sqcap T = (\bigcap_{t \in T} V_t, \bigcap_{t \in T} \leq_t)$ is the infimum of T in $\mathbb{R}$.*

Proof. Let $s = (V_s, \leq_s) \in \mathbb{R}$ be an upper bound of T. Let $V = \bigcap_{t \in T} V_t$ and $\leq = \bigcap_{t \in T} \leq_t$. Clearly, $V \subseteq V_s$ and $\leq \subseteq \leq_s$. Now, let $p \in V_s$ and $q \in V$ with $p \leq_s q$. For all $t \in T$, since $t \leq s$, we get $p, q \in V_t$ and $p \leq_t q$. Hence, $p, q \in V$ and $p \leq q$. Therefore, $V = \downarrow_{\leq_s} V$ and $\leq = \leq_s|_{V \times V}$. It follows immediately that $r = (V, \leq)$ is a real pomset and a prefix of s.

Since s is an upper bound of $T \cup \{r\}$, we get immediately from Lemma 2.4 that r is a lower bound of T.

Finally, let $r' = (V', \leq') \in \mathbb{R}$ be a lower bound of T. For all $t \in T$, we have $V' \subseteq V_t$. Hence, $V' \subseteq V$ and since $r \leq s$ and $r' \leq s$, we get $r \leq r'$ by

Lemma 2.4. □

Proposition 2.8

(i) *A real pomset $t \in \mathbb{R}$ is compact if and only if it is finite.*

(ii) $(\mathbb{R}, \leq)$ *is algebraic.*

Proof. Let $r = (V_r, \leq_r)$ be a finite pomset, and let T be a directed family of pomsets with $r \leq \bigsqcup T = (\bigcup_{t\in T} V_t, \bigcup_{t\in T} \leq_t)$. Since V_r is finite and T is directed, there is some $t \in T$ with $V_r \subseteq V_t$. Now, since $t \leq \sqcup T$ and $r \leq \sqcup T$, we deduce from Lemma 2.4 that $r \leq t$. Therefore, any finite real pomset is compact.

Next, let $t \in \mathbb{R}$ and $T(t) = \{s \in \mathbb{R}_f \mid s \leq t\}$. We claim that $T(t)$ is directed and $t = \sqcup T(t)$. Indeed, let $r, s \in T$. Using Proposition 2.5, the vertex set of $r \sqcup s$ is $V_r \cup V_s$ which is finite. Hence, $r \sqcup s \in T(t)$ and $T(t)$ is directed. It is clear that $V_t = \cup_{s\in T(t)} V_s$ and we obtain $t = \sqcup T(t)$ by Lemma 2.4 and Proposition 2.5.

Now, if $t \in \mathbb{R} \setminus \mathbb{R}_f$ is not finite then, $T(t)$ is directed and $t = \sqcup T(t)$ implies that t is not compact since $t \notin T(t)$ which is a lower set. Therefore, the compact real pomsets are exactly the finite ones which proves (i).

(ii) now follows directly since $T(t)$ is the set of compact elements below t. □

Though we do not use it in this paper, we note that a real pomset $(V, \leq)$ is prime if and only if $V = \downarrow p$ for some $p \in V$, which shows that $(\mathbb{R}, \leq)$ is also prime-algebraic.

Complex pomsets

We now define the class of complex pomsets, which will be the basis for the denotational model for the language we study. We first need a preliminary definition. If $t \in \mathbb{R}$ is a real pomset over Σ, then we define the set of resources that occur infinitely often in t by

$$\mathrm{resinf}(t) = \bigcap\{\mathrm{res}(r^{-1}t) \mid r \in \mathbb{R}_f \text{ and } r \leq t\}.$$

Note that, if $s, t \in \mathbb{R}$ with $s \leq t$, then we have

$$\mathrm{resinf}(s) \subseteq \mathrm{resinf}(t) \subseteq \mathrm{resinf}(s) \cup \mathrm{res}(s^{-1}t).$$

Definition 2.9 A *complex pomset* is a pair $x = (r, R)$ where

- $r \in \mathbb{R}$ is a real pomset over Σ,
- $R \subseteq \mathcal{R}$ satisfies $\mathrm{resinf}(r) \subseteq R$, and
- x satisfies the *resource constraint:* $\mathrm{res}(r) \cup R$ is finite.

For a complex pomset $x = (r, R)$, we define its *real part* $\mathrm{Re}(x) = r$, its imaginary part $\mathrm{Im}(x) = R$, and its resource set $\mathrm{res}(x) = \mathrm{res}(r) \cup R$. We denote the family of complex pomsets over Σ by $\mathbb{C}(\Sigma)$, or simply by $\mathbb{C}$. A complex pomset is finite if its real part is finite. The family of finite complex pomsets is denoted by $\mathbb{C}_f$. Again, we generalize the independence relation to complex pomsets $x, y \in \mathbb{C}$ by $x \mathrel{I} y$ if $\mathrm{res}(x) \cap \mathrm{res}(y) = \emptyset$.

A complex pomset is meant to denote an approximation of a process behavior. The real part depicts what we have already observed while the imaginary part contains the resources that may be used to complete the process execution. Since every process has a finite description, it may only refer to finitely many resources. This explains our restriction that a complex pomset uses finitely many resources only. This restriction was not imposed for real pomsets but is automatically enforced for the real part of a complex pomset. This restriction is crucial for Lemma 2.12 which will be used throughout the paper.

We now define the approximation ordering corresponding to the intuition given above.

Definition 2.10 For complex pomsets (r, R) and (s, S), we define the *approximation relation* by

$$(r, R) \sqsubseteq (s, S) \quad \Leftrightarrow \quad r \leq s \ \wedge \ R \supseteq S \cup \mathrm{res}(r^{-1}s).$$

Note that $x \sqsubseteq y$ implies $\mathrm{res}(x) \supseteq \mathrm{res}(y)$.

For instance, $\left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & \\ & b \\ c \arrow[ur] & \end{tikzcd} , \ a \cup b \cup c\right) \sqsubseteq \left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & & a \\ & b \arrow[ur] \arrow[dr] & \\ c \arrow[ur] & & c \end{tikzcd} , \ b \cup c\right)$ but if $b \not\subseteq a \cup c$, then $\left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & \\ & b \\ c \arrow[ur] & \end{tikzcd} , \ a \cup c\right) \not\sqsubseteq \left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & & & a \\ & b \arrow[r] & b \arrow[ur] \arrow[dr] & \\ c \arrow[ur] & & & c \end{tikzcd} , \ a \cup c\right)$.

Note that, if the imaginary part of a complex pomset x is empty, then x must be finite and x is maximal for the approximation ordering. Actually, a complex pomset x is maximal if and only if $\mathrm{Im}(x) = \mathrm{resinf}(\mathrm{Re}(x))$.

We establish now the domain properties of $(\mathbb{C}, \sqsubseteq)$ that will be useful in this paper.

Proposition 2.11 $(\mathbb{C}, \sqsubseteq)$ *is a consistently complete partial order. In fact, if* $X \subseteq \mathbb{C}$ *is non-empty and pairwise consistent, then*

$$\bigsqcup X = \left(\bigsqcup_{x \in X} \mathrm{Re}(x), \bigcap_{x \in X} \mathrm{Im}(x) \right).$$

Proof. Reflexivity of $\sqsubseteq$ is clear from the fact that $\leq$ is a partial order on pomsets. Now suppose $(r, R) \sqsubseteq (s, S) \sqsubseteq (r, R)$. Then $r \leq s \leq r$ implies $r = s$. Moreover, $R \supseteq S \cup \mathrm{res}(r^{-1}s)$ and $S \supseteq R \cup \mathrm{res}(s^{-1}r)$ imply $R = S$. Finally, if $(r, R) \sqsubseteq (s, S) \sqsubseteq (t, T)$, then $r \leq s \leq t$ implies $r \leq t$. Also,

$$R \supseteq S \cup \mathrm{res}(r^{-1}s) \supseteq T \cup \mathrm{res}(s^{-1}t) \cup \mathrm{res}(r^{-1}s) = T \cup \mathrm{res}(r^{-1}t).$$

The last equality holds since $r \leq s \leq t$ implies that the vertex set of $r^{-1}t$ is $V_t \setminus V_r = (V_t \setminus V_s) \cup (V_s \setminus V_r)$.

For consistent completeness, let $X \subseteq \mathbb{C}$ be a pairwise consistent set of complex pomsets. Then, $\mathrm{Re}(X)$ is a pairwise consistent set of real pomsets and Proposition 2.5 implies $s = \bigsqcup_{x \in X} \mathrm{Re}(x)$ is a real pomset. We let $S = \bigcap_{x \in X} \mathrm{Im}(x)$.

We claim that $r \leq s$ and $R \supseteq S \cup \mathrm{res}(r^{-1}s)$ for all $x = (r, R) \in X$. From the definition of s and S, we already know that $r \leq s$ and $R \supseteq S$. Now, let $\rho \in \mathrm{res}(r^{-1}s)$. Note that the over-synchronization condition implies $|r|_\rho < \infty$. Then, since $s = \sqcup \mathrm{Re}(X)$, we deduce from Proposition 2.5 that there exists $x' = (r', R') \in X$ with $|r'|_\rho > |r|_\rho$. Now, X is consistent and we find $z = (t, T) \in \mathbb{C}$ such that $x, x' \sqsubseteq z$. It follows $|t|_\rho \geq |r'|_\rho > |r|_\rho$ and therefore, $\rho \in \mathrm{res}(r^{-1}t) \subseteq \mathrm{Im}(x) = R$, which proves the claim.

From this, we deduce that $y = (s, S)$ is a complex pomset and an upper bound of X. Indeed, for all $x = (r, R) \in X$, we have

$$S \cup \mathrm{res}(s) = S \cup \mathrm{res}(r) \cup \mathrm{res}(r^{-1}s) \subseteq R \cup \mathrm{res}(r) = \mathrm{res}(x)$$

which is finite, and

$$\mathrm{resinf}(s) \subseteq \mathrm{resinf}(r) \cup \mathrm{res}(r^{-1}s) \subseteq R = \mathrm{Im}(x)$$

and, since $x \in X$ is arbitrary, we get $\mathrm{resinf}(s) \subseteq S$. Hence, $y = (s, S)$ is a complex pomset and the claim above shows that y is an upper bound of X.

Finally, assume that $z \in \mathbb{C}$ is an upper bound for X. Then, $\mathrm{Re}(x) \leq \mathrm{Re}(z)$ and $\mathrm{Im}(z) \subseteq \mathrm{Im}(x)$ for all $x \in X$. It follows that $s \leq \mathrm{Re}(z)$ and $\mathrm{Im}(z) \subseteq S$. Further, for any $x \in X$, we have $\mathrm{Re}(x) \leq s \leq \mathrm{Re}(z)$, which implies $\mathrm{res}(s^{-1}\mathrm{Re}(z)) \subseteq \mathrm{res}(\mathrm{Re}(x)^{-1}\mathrm{Re}(z)) \subseteq \mathrm{Im}(x)$. Therefore, $\mathrm{res}(s^{-1}\mathrm{Re}(z)) \subseteq S$. It follows that $y = (s, S) \sqsubseteq z$, and so $y = \bigsqcup X$. □

Using the restriction that a complex pomset uses only finitely many resources we can prove:

Lemma 2.12 *Let $X \subseteq \mathbb{C}$ be directed. There exists $x \in X$ such that $\mathrm{Im}(x) = \mathrm{Im}(\sqcup X)$, and $\mathrm{res}(\mathrm{Re}(x)) = \mathrm{res}(\mathrm{Re}(\sqcup X))$; hence also $\mathrm{res}(x) = \mathrm{res}(\sqcup X)$.*

Proof. We note that $\mathrm{res}(\sqcup X) = \mathrm{res}(\mathrm{Re}(\sqcup X)) \cup \mathrm{Im}(\sqcup X)$ is finite. For all $x \in X$, we have $\mathrm{res}(\mathrm{Re}(x)) \subseteq \mathrm{res}(\mathrm{Re}(\sqcup X))$. Since $\mathrm{Re}(\sqcup X) = \sqcup_{x \in X} \mathrm{Re}(x)$, for all $\rho \in \mathrm{res}(\mathrm{Re}(\sqcup X))$ we find $x_\rho \in X$ with $\rho \in \mathrm{res}(\mathrm{Re}(x_\rho))$. Since $\mathrm{res}(\mathrm{Re}(\sqcup X))$ is finite and X is directed, we find $x_1 \in X$ with $x_\rho \sqsubseteq x_1$ for all $\rho \in \mathrm{res}(\mathrm{Re}(\sqcup X))$. We deduce that $\mathrm{res}(\mathrm{Re}(x_1)) = \mathrm{res}(\mathrm{Re}(\sqcup X))$.

We have $\mathrm{Im}(\sqcup X) = \bigcap_{x \in X} \mathrm{Im}(x)$. We fix some $y \in X$. For all $\rho \in \mathrm{Im}(y) \setminus \mathrm{Im}(\sqcup X)$ we find $y_\rho \in X$ such that $\rho \notin \mathrm{Im}(y_\rho)$. Since X is directed and $\mathrm{Im}(y)$ is finite, we find $x_2 \in X$ such that $y \sqsubseteq x_2$ and $y_\rho \sqsubseteq x_2$ for all $\rho \in \mathrm{Im}(y) \setminus \mathrm{Im}(\sqcup X)$.

We deduce that

$$\mathrm{Im}(x_2) \subseteq \mathrm{Im}(y) \cap \left(\bigcap_{\rho \in \mathrm{Im}(y) \setminus \mathrm{Im}(\sqcup X)} \mathrm{Im}(y_\rho) \right) \subseteq \mathrm{Im}(\sqcup X).$$

Using the fact that X is directed, we can find $x \in X$ with $x_1, x_2 \sqsubseteq x$. It follows that

$$\mathrm{Im}(\sqcup X) \subseteq \mathrm{Im}(x) \subseteq \mathrm{Im}(x_2) \subseteq \mathrm{Im}(\sqcup X)$$

and

$$\mathrm{res}(\mathrm{Re}(\sqcup X)) = \mathrm{res}(\mathrm{Re}(x_1)) \subseteq \mathrm{res}(\mathrm{Re}(x)) \subseteq \mathrm{res}(\mathrm{Re}(\sqcup X)).$$

□

From this lemma, we derive immediately

Corollary 2.13 *The mapping* $\mathrm{res} : (\mathbb{C}, \sqsubseteq) \to (\mathcal{P}_f(\mathcal{R}), \supseteq)$ *is continuous.* □

We say that a real pomset $r \in \mathbb{R}$ is a *prefix* of a complex pomset $x = (s, S) \in \mathbb{C}$ when $r \leq s$. In this case we write $r \leq x$ and we also let $r^{-1}x = (r^{-1}s, S)$. Note that $\mathrm{res}(r^{-1}x) = S \cup \mathrm{res}(r^{-1}s)$.

Definition 2.14 For $x \in \mathbb{C}$, we define

$$\chi(x) = \{(r, \mathrm{res}(r^{-1}x)) \mid r \leq x \text{ and } r \in \mathbb{R}_f\}.$$

Lemma 2.15 *If* $x \in \mathbb{C}$, *then* $\chi(x)$ *is directed and* $\sqcup\chi(x) = x$.

Proof. Let $r, s \in \mathbb{R}_f$ be such that $r, s \leq x$ and let $t = r \sqcup s$. Clearly, t is finite and $t \leq x$. Now, $r \leq t \leq x$ implies $\mathrm{res}(r^{-1}x) = \mathrm{res}(r^{-1}t) \cup \mathrm{res}(t^{-1}x)$. Therefore, $(r, \mathrm{res}(r^{-1}x)) \sqsubseteq (t, \mathrm{res}(t^{-1}x))$. Similarly, $(s, \mathrm{res}(s^{-1}x)) \sqsubseteq (t, \mathrm{res}(t^{-1}x))$ and we deduce that $\chi(x)$ is directed.

By Propositions 2.8 and 2.11 we have

$$\mathrm{Re}(\sqcup\chi(x)) = \sqcup\{r \in \mathbb{R}_f \mid r \leq \mathrm{Re}(x)\} = \mathrm{Re}(x)$$

and

$$\begin{aligned} \mathrm{Im}(\sqcup\chi(x)) &= \bigcap\{\mathrm{Im}(y) \mid y \in \chi(x)\} \\ &= \bigcap\{\mathrm{res}(r^{-1}x) \mid r \leq x \text{ and } r \in \mathbb{R}_f\} \\ &= \bigcap\{\mathrm{res}(r^{-1}\mathrm{Re}(x)) \cup \mathrm{Im}(x) \mid r \leq x \text{ and } r \in \mathbb{R}_f\} \\ &= \mathrm{resinf}(\mathrm{Re}(x)) \cup \mathrm{Im}(x) = \mathrm{Im}(x), \end{aligned}$$

and so $\sqcup\chi(x) = x$. □

Proposition 2.16

(i) *A complex pomset* $x \in \mathbb{C}$ *is compact if and only if it is finite.*

(ii) $(\mathbb{C}, \sqsubseteq)$ *is algebraic.*

Proof. Assume $y \in \mathbb{C}$ is finite and let X be a directed set of complex pomsets with $y \sqsubseteq \bigsqcup X$. By Proposition 2.8, $\mathrm{Re}(y)$ is a compact real pomset and from Proposition 2.11 we get $\mathrm{Re}(y) \leq \mathrm{Re}(\bigsqcup X) = \bigsqcup \mathrm{Re}(X)$. Since $\mathrm{Re}(X)$ is also directed, we have $\mathrm{Re}(y) \leq \mathrm{Re}(x)$ for some $x \in X$. Using Lemma 2.12 we may assume in addition that $\mathrm{Im}(x) = \mathrm{Im}(\sqcup X)$. Therefore,

$$\mathrm{Im}(x) \cup \mathrm{res}(\mathrm{Re}(y)^{-1}\mathrm{Re}(x)) \subseteq \mathrm{Im}(\sqcup X) \cup \mathrm{res}(\mathrm{Re}(y)^{-1}\mathrm{Re}(\sqcup X)) \subseteq \mathrm{Im}(y)$$

which implies $y \sqsubseteq x$. Therefore, any finite complex pomset is compact. Conversely, if $y \in \mathbb{C}$ is not finite, then Lemma 2.15 implies that y is not compact. Indeed, there is no $x \in \chi(y)$ such that $y \sqsubseteq x$. Hence, (i) is proved.

(ii) Now, let $y \in \mathbb{C}$ and $K(y) = \{x \in \mathbb{C}_f \mid x \sqsubseteq y\}$. Let $x, x' \in K(y)$. By Proposition 2.11 we know that $x \sqcup x'$ is finite, hence $x \sqcup x' \in K(y)$. Therefore, $K(y)$ is directed. Now, $\chi(y) \subseteq K(y)$ hence we deduce from Lemma 2.15 that $y = \sqcup K(y)$ which concludes the proof. □

3 The Language, Its Resources and Its Operational Semantics

In this section we give the syntax for the language we study, and also define the resource mapping that plays a crucial role for us. We then define an SOS-style operational semantics for the language and give a few illustrative examples.

To begin, we assume we have a countable set $\mathcal{V}$ of variables and a countable set $\mathcal{R}_0$ of *resources*. For each variable $x \in \mathcal{V}$, we also define a new resource ρ_x, and we let $\mathcal{R}_\mathcal{V} = \{\rho_x \mid x \in \mathcal{V}\}$. The resources in $\mathcal{R}_\mathcal{V}$ are called *variable resources*. We assume that $\mathcal{R}_0 \cap \mathcal{R}_\mathcal{V} = \emptyset$. We let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_\mathcal{V}$ denote the set of all resources. Also, we let $\mathrm{Act} = \{a \in \mathcal{P}_f(\mathcal{R}_0) \mid a \neq \emptyset\}$ be the family of finite, nonempty subsets of $\mathcal{R}_0$. For all $x \in \mathcal{V}$, we identify $\{\rho_x\}$ with ρ_x and we let $\Sigma = \mathrm{Act} \cup \mathcal{R}_\mathcal{V} \subseteq \{a \in \mathcal{P}_f(\mathcal{R}) \mid a \neq \emptyset\}$.

The BNF-style syntax for our language $\mathcal{L}$ is given by the following:

$$p ::= \mathrm{SKIP} \mid a \mid p\,;p \mid p \circ p \mid p \underset{C}{\|} p \mid p\backslash R \mid x \mid \mathrm{rec}\, x.p,$$

where

- SKIP denotes normal termination;
- $a \in \mathrm{Act}$ is an action;
- ; denotes (strict) sequential composition;
- $\circ$ denotes weak sequential composition;
- $p \underset{C}{\|} q$ is the parallel composition of p and q, synchronizing on all actions $a \in C \subseteq \Sigma$;
- $p\backslash R$ is p with all resources in $R \subseteq \mathcal{R}$ hidden, this means that if an action is contained in R then it is completely hidden and if an action intersects

R then only the resources outside R remain visible when performing this action;

- $x \in \mathcal{V}$ is a variable; and
- $\operatorname{rec} x.p$ is recursion.

An occurrence of a variable x is *free* in a process p if it is not in the scope of some $\operatorname{rec} x$. We denote by $\operatorname{Free}(p)$ the set of variables that have at least one free occurrence in p. A term p is closed if $\operatorname{Free}(p) = \emptyset$ and the set of closed terms is denoted by $\mathcal{L}_c$. The operational semantics is given only for closed terms.

3.1 The Resource Mapping

The definition of our language relies on the set $\mathcal{R}$ of resources. We now define the resources of a process by first showing how to endow $\mathcal{P}_f(\mathcal{R})$ with the same algebraic structure as that of our language $\mathcal{L}$. The resource map is then the unique algebra homomorphism $\operatorname{res} : \mathcal{L} \to \mathcal{P}_f(\mathcal{R})$ which is guaranteed because $\mathcal{L}$ is taken to be the initial algebra with this signature.

Definition 3.1 Define $\operatorname{res} : \mathcal{L} \to \mathcal{P}_f(\mathcal{R})$ by:

- $\operatorname{res}(\mathrm{SKIP}) = \emptyset$,
- $\operatorname{res}(a) = a$ for all $a \in \mathrm{Act}$,
- $\operatorname{res}(x) = \emptyset$ for all $x \in \mathcal{V}$,
- $\operatorname{res}(p\,;q) = \operatorname{res}(p \circ q) = \operatorname{res}(p \mathbin{\underset{C}{\|}} q) = \operatorname{res}(p) \cup \operatorname{res}(q)$ for processes $p, q \in \mathcal{L}$,
- $\operatorname{res}(p\backslash R) = \operatorname{res}(p) \setminus R$ for any process $p \in \mathcal{L}$ and any $R \subseteq \mathcal{R}$,
- $\operatorname{res}(\operatorname{rec} x.p) = \operatorname{res}(p) \cup \{\rho_x\}$ for any process $p \in \mathcal{L}$.

3.2 Operational Semantics

In this section, we give a set of SOS-style rules that define the operational semantics of our language. Notable here is that we use the variable resources from $\mathcal{R}_{\mathcal{V}}$ to observe unwindings of recursion with rule (6) of Table 1. This approach to modelling recursion has the advantage of allowing us to observe divergence: a process that can perform an infinite sequence of ρ_x transitions without performing any visible transition is a divergent process. Also, processes can synchronize not only on actions from Act, but also on variable resources from $\mathcal{R}_{\mathcal{V}}$. This allows us to define the parallel composition of recursive processes which synchronizes on the unwinding of the recursions of each branch. This is an interesting feature to describe processes. Note that, if we do not want to observe the unwinding of recursion, we may abstract away from this by hiding the variable resources with rule (5).

For $a \in \Sigma$, we use the notation $a\,I\,C$ to denote the fact that $a \cap c = \emptyset$ for all $c \in C \subseteq \Sigma$, and $a\,I\,p$ to denote that $a \cap \operatorname{res}(p) = \emptyset$ for a process $p \in \mathcal{L}$. Finally, $a\,I\,C, p$ stands for $a\,I\,C$ and $a\,I\,p$.

(1) $$\frac{}{a \xrightarrow{a} \mathrm{SKIP}}$$

(2a) $$\frac{p \xrightarrow{a} p'}{p \,;\, q \xrightarrow{a} p' \,;\, q}$$

(2b) $$\frac{q \xrightarrow{a} q', \quad \mathrm{res}(p) = \emptyset}{p \,;\, q \xrightarrow{a} p \,;\, q'}$$

(3a) $$\frac{p \xrightarrow{a} p'}{p \circ q \xrightarrow{a} p' \circ q}$$

(3b) $$\frac{q \xrightarrow{a} q', \quad a \mathrel{I} p}{p \circ q \xrightarrow{a} p \circ q'}$$

(4a) $$\frac{p \xrightarrow{a} p', \quad a \mathrel{I} C, q}{p \underset{C}{\|} q \xrightarrow{a} p' \underset{C}{\|} q}$$

(4b) $$\frac{q \xrightarrow{a} q', \quad a \mathrel{I} C, p}{p \underset{C}{\|} q \xrightarrow{a} p \underset{C}{\|} q'}$$

(4c) $$\frac{p \xrightarrow{a} p', q \xrightarrow{a} q', \quad a \in C}{p \underset{C}{\|} q \xrightarrow{a} p' \underset{C}{\|} q'}$$

(5) $$\frac{p \xrightarrow{a} p'}{p \backslash R \xrightarrow{a \backslash R} p' \backslash R}$$

(6) $$\frac{}{\mathrm{rec}\, x.p \xrightarrow{\rho_x} p[\mathrm{rec}\, x.p/x]}$$

Table 1
The Transition Rules for $\mathcal{L}_c$

Note that, if $a \subseteq R$, then hiding R when performing the action a yields a transition labelled with $a \setminus R = \emptyset$ which will be denoted by 1 subsequently. The labels of our transition system are thus from $\Sigma \cup \{1\}$.

For $p, q \in \mathcal{L}$ and $x \in \mathcal{V}$, we denote by $p[q/x]$ the process obtained from p by substituting q for all free occurrences of x. This is standard and we do not recall the formal definition here. Substitutions are used in the operational rule of recursion.

3.3 Examples

We close this section by giving some examples of processes from our language together with their operational semantics.

(i) If $p = \mathrm{rec}\, x.(a \,;\, x)$, then $\mathrm{res}(p) = a \cup \{\rho_x\}$, and we see that applying rules (6), (2a) and (2b) we obtain

$$p \xrightarrow{\rho_x} a \,;\, p \xrightarrow{a} \mathrm{SKIP} \,;\, p \xrightarrow{\rho_x} \mathrm{SKIP} \,;\, (a \,;\, p) \xrightarrow{a} \cdots$$

(ii) On the other hand, if $p = \mathrm{rec}\, x.(a \circ x)$, then we apply rule (6) and

$$p \xrightarrow{\rho_x} a \circ p.$$

But, since $a\, I\, \rho_x$, now there are two possibilities using rules (3a) or (3b):

$$a \circ p \xrightarrow{a} \mathrm{SKIP} \circ p \qquad \text{and} \qquad a \circ p \xrightarrow{\rho_x} a \circ (a \circ p).$$

The first of these has one possible transition, but the second has two:

$$\begin{aligned} \mathrm{SKIP} \circ p &\xrightarrow{\rho_x} \mathrm{SKIP} \circ (a \circ p), \quad \text{while} \\ a \circ (a \circ p) &\xrightarrow{a} \mathrm{SKIP} \circ (a \circ p) \qquad \text{and} \\ a \circ (a \circ p) &\xrightarrow{\rho_x} a \circ (a \circ (a \circ p)). \end{aligned}$$

It is difficult to see what the future transitions are without drawing a diagram. Here is one that illustrates the possible behaviors of $p = \mathrm{rec}\, x.(a \circ x)$:

$$\begin{tikzcd} p \arrow[r, "\rho_x"] & a \circ p \arrow[r, "\rho_x"] \arrow[dr, "a"'] & a \circ (a \circ p) \arrow[r, "\rho_x"] \arrow[dr, "a"'] & a \circ (a \circ (a \circ p)) \arrow[r, "\rho_x"] \arrow[dr, "a"'] & \cdots & \\ & & \mathrm{SKIP} \circ p \arrow[r, "\rho_x"] & \mathrm{SKIP} \circ (a \circ p) \arrow[r, "\rho_x"] \arrow[dr, "a"] & \mathrm{SKIP} \circ (a \circ (a \circ p)) \arrow[r, "\rho_x"] \arrow[dr, "a"] & \cdots \\ & & & & {} & {} \end{tikzcd}$$

We see that it is possible to do n a's in a row by going out the top row n steps and then following a downward path.

From this picture, two things emerge. The first is that p can diverge at any point – there is a horizontal path of ρ_x's emanating from each vertex, and this represents an infinite unwinding of the recursion without performing any action from Act. This occurs because weak sequential composition allows the ρ_x to occur independently of actions from Act, hence, recursion is unguarded in this process.

The second thing that becomes apparent is p's behavior. Indeed, what now should be clear is that p is capable of any linearization of the following real pomset

$$\begin{tikzcd} \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[r] \arrow[dr] & \rho_x & \cdots \\ & a \arrow[r] & a \arrow[r] & a & \cdots \end{tikzcd}$$

Finally, if we identify a process $\mathrm{SKIP} \circ q$ with q, then the diagram above can be greatly simplified:

$$\begin{tikzcd} p \arrow[r, shift left, "\rho_x"] & a \circ p \arrow[l, shift left, "a"] \arrow[r, shift left, "\rho_x"] & a \circ (a \circ p) \arrow[l, shift left, "a"] \arrow[r, shift left, "\rho_x"] & a \circ (a \circ (a \circ p)) \arrow[l, shift left, "a"] \arrow[r, shift left, "\rho_x"] & \cdots \arrow[l, shift left, "a"] \end{tikzcd}$$

This diagram makes the last claim about doing any finite number of a's in a row even clearer.

(iii) As a third example, we consider the process $p \backslash \rho_x$, where $p = \mathrm{rec}\, x.(a \circ x)$

from the previous example. Applying Rule (5), we obtain the transition system of $p\backslash\rho_x$ from the diagram above by hiding the ρ_x's:

$$\begin{tikzcd} p\backslash\rho_x \arrow[r, shift left, "1"] & (a\circ p)\backslash\rho_x \arrow[l, shift left, "a"] \arrow[r, shift left, "1"] & (a\circ(a\circ p))\backslash\rho_x \arrow[l, shift left, "a"] \arrow[r, shift left, "1"] & (a\circ(a\circ(a\circ p)))\backslash\rho_x \arrow[l, shift left, "a"] \arrow[r, shift left, "1"] & \cdots \arrow[l, shift left, "a"] \end{tikzcd}$$

The behavior of this process also is now clear: $p\backslash\rho_x$ is capable of doing any finite number of a's (preceded by at least as many 1's), and then diverging (doing infinitely many 1's). It also is capable of infinitely many a's – by interspersing finite runs of a's with sufficiently long runs of 1's.

(iv) Our final example in this section is a little more practical. We show how we can achieve a 2-place buffer using our process algebra and hiding. We begin by defining a one-place buffer $\mathcal{B}^1(a,b)$ which receives messages of type a and send messages of type b. This is formed from the process $p = \operatorname{rec} x.(a\,;b\,;x)$ by hiding the unwinding on the recursion – i.e., $\mathcal{B}^1(a,b) = p\backslash\rho_x$. Indeed, using reasoning like that of the previous examples, we see that p behaves like:

$$\begin{tikzcd} p \arrow[r, "\rho_x"] & a\,;b\,;p \arrow[r, "a"] & b\,;p \arrow[ll, bend left, "b"] \end{tikzcd}$$

Now, hiding ρ_x gives us:

$$\begin{tikzcd} \mathcal{B}^1(a,b) \arrow[r, "1"] & (a\,;b\,;p)\backslash\rho_x \arrow[r, "a"] & (b\,;p)\backslash\rho_x \arrow[ll, bend left, "b"] \end{tikzcd}$$

The observable behavior of this process is then $a \to b \to a \to b \to \cdots$.

We can form an n-place buffer from $\mathcal{B}^1(a,b)$ simply by placing n-copies of $\mathcal{B}^1$ in parallel with each other – each with distinct inputs and outputs, and then hiding the internal communications. That is,

$$\mathcal{B}^n(a,b) := (\mathcal{B}^1(a,c_1) \mathbin{\|_C} \mathcal{B}^1(c_1,c_2) \mathbin{\|_C} \cdots \mathbin{\|_C} \mathcal{B}^1(c_{n-1},b))\backslash D,$$

where $a, c_1, \ldots, c_{n-1}, b$ are pairwise independent, $C = \{c_1, \ldots, c_{n-1}\}$ and $D = \operatorname{res}(C) = \cup_{1\le i<n} c_i$.

For $n = 2$, we take $\mathcal{B}^2(a,b) = (\mathcal{B}^1(a,c) \mathbin{\|_c} \mathcal{B}^1(c,b))\backslash c$. We let $p_1 = \operatorname{rec} x.(a\,;c\,;x)$ and $p_2 = \operatorname{rec} x.(c\,;b\,;x)$. Figure 1 gives the transition system for the process $\mathcal{B}^2(a,b)$. We see from this transition system that the observable behaviors of the process $\mathcal{B}^2(a,b)$ are the linearizations of the following real pomset.

$$\begin{tikzcd}[column sep=small] a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a \arrow[dr] & \cdots \\ & b \arrow[r] \arrow[ur] & b \arrow[r] \arrow[ur] & b \arrow[r] \arrow[ur] & b \arrow[r] & b & \cdots \end{tikzcd}$$

$$\begin{tikzcd}
 & (p_1\backslash\rho_x \underset{c}{\|} (b\,;p_2)\backslash\rho_x)\backslash c \arrow[r, "1"] \arrow[d, "b"] & ((a\,;c\,;p_1)\backslash\rho_x \underset{c}{\|} (b\,;p_2)\backslash\rho_x)\backslash c \arrow[r, "a"] \arrow[d, "b"] & ((c\,;p_1)\backslash\rho_x \underset{c}{\|} (b\,;p_2)\backslash\rho_x)\backslash c \arrow[d, "b"] \\
 \arrow[r] & \mathcal{B}^2(a,b) \arrow[r, "1"] \arrow[d, "1"] & ((a\,;c\,;p_1)\backslash\rho_x \underset{c}{\|} p_2\backslash\rho_x)\backslash c \arrow[r, "a"] \arrow[d, "1"] & ((c\,;p_1)\backslash\rho_x \underset{c}{\|} p_2\backslash\rho_x)\backslash c \arrow[d, "1"] \\
 & (p_1\backslash\rho_x \underset{c}{\|} (c\,;b\,;p_2)\backslash\rho_x)\backslash c \arrow[r, "1"] & ((a\,;c\,;p_1)\backslash\rho_x \underset{c}{\|} (c\,;b\,;p_2)\backslash\rho_x)\backslash c \arrow[r, "a"] & ((c\,;p_1)\backslash\rho_x \underset{c}{\|} (c\,;b\,;p_2)\backslash\rho_x)\backslash c \arrow[uull, bend left, "1"]
\end{tikzcd}$$

Fig. 1. Operational semantics for the process $\mathcal{B}^2(a,b)$.

4 Denotational Semantics

In this section, we define a denotational semantics for our language $\mathcal{L}$. Later, we show this semantics is adequate and fully abstract with respect to the operational semantics given by the transition system we presented in the previous section. The semantics takes its values in the family $[\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$ of continuous maps from $\mathbb{C}^{\mathcal{V}}$ to the underlying domain $\mathbb{C}$ of complex pomsets. The semantics of a closed process p is a constant map, which means it is simply a complex pomset. But, in order to give the semantics of recursion, we also have to consider terms with free variables.

We begin by defining the family of *semantic environments* to be the mappings $\sigma : \mathcal{V} \to \mathbb{C}$, and we endow this with the domain structure from the target domain $\mathbb{C}$, regarding $\mathbb{C}^{\mathcal{V}}$ as a product on $\mathcal{V}$-copies of $\mathbb{C}$. For $\sigma \in \mathbb{C}^{\mathcal{V}}$, $x \in \mathcal{V}$ and $y \in \mathbb{C}$, we denote by $\sigma[x \mapsto y]$ the environment that coincides with σ on all variables but x, and which assigns y to x.

The semantics of an arbitrary process $p \in \mathcal{L}$ is a continuous map from $\mathbb{C}^{\mathcal{V}}$ to $\mathbb{C}$ satisfying a condition that will be useful in order to deal with recursion.

Definition 4.1 Our semantic domain $\mathcal{D}$ is the set of continuous maps $f : \mathbb{C}^{\mathcal{V}} \to \mathbb{C}$ satisfying for all $\sigma \in \mathbb{C}^{\mathcal{V}}$ and $z \in \mathcal{V}$,

$$(1) \qquad \mathrm{res}(f(\sigma[z \mapsto (1,\emptyset)])) \subseteq \mathrm{res}(f(\sigma)) \subseteq \mathrm{res}(f(\sigma[z \mapsto (1,\emptyset)])) \cup \mathrm{res}(\sigma(z)).$$

As usual in domain theory, we can lift the ordering on $\mathbb{C}$ to $\mathcal{D}$ in pointwise fashion by setting $f \sqsubseteq g$ if $f(\sigma) \sqsubseteq g(\sigma)$ for all $\sigma \in \mathbb{C}^{\mathcal{V}}$.

Proposition 4.2 *The set* $(\mathcal{D}, \sqsubseteq)$ *is a dcpo.*

Proof. It is a standard fact from domain theory that since $(\mathbb{C}, \sqsubseteq)$ is a dcpo (Proposition 2.11), so are $\mathbb{C}^{\mathcal{V}}$ and $[\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$ when endowed with the pointwise order (cf. [1], Theorem 2.1.18). Moreover, if $\mathcal{E} \subseteq [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$ is directed then $g = \sqcup\mathcal{E}$ is the continuous map defined by $g(\sigma) = \sqcup_{f\in\mathcal{E}} f(\sigma)$. Using these facts, we only have to show that if $\mathcal{E} \subseteq \mathcal{D}$ then the map g satisfies Equation (1).

Let $\sigma \in \mathbb{C}^{\mathcal{V}}$, $z \in \mathcal{V}$ and $\sigma' = \sigma[z \mapsto (1, \emptyset)]$. The sets $X = \{f(\sigma) \mid f \in \mathcal{E}\}$ and $Y = \{f(\sigma') \mid f \in \mathcal{E}\}$ are directed. Using Lemma 2.12, we find $f_1, f_2 \in \mathcal{E}$ with $\mathrm{res}(f_1(\sigma)) = \mathrm{res}(g(\sigma))$ and $\mathrm{res}(f_2(\sigma')) = \mathrm{res}(g(\sigma'))$. Since $\mathcal{E}$ is directed, we find $f \in \mathcal{E}$ with $f_1, f_2 \sqsubseteq f$. Then, we get $\mathrm{res}(f(\sigma)) = \mathrm{res}(g(\sigma))$ and $\mathrm{res}(f(\sigma')) = \mathrm{res}(g(\sigma'))$. Now, it is easy to see that g satisfies Equation (1) using the fact that f does. □

We obtain a compositional semantics by defining the structure of an Ω-algebra on $\mathcal{D}$, where Ω is the signature of our language $\mathcal{L}$. We begin with the simplest operations – the nullary operators.

The denotational semantics of constants and of variables are defined by the maps:

$$\begin{aligned} [\![\mathrm{SKIP}]\!] &\in [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}] \quad \text{by} \quad [\![\mathrm{SKIP}]\!](\sigma) = (1, \emptyset) \\ [\![a]\!] &\in [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}] \quad \text{by} \quad [\![a]\!](\sigma) = (a, \emptyset) \\ [\![x]\!] &\in [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}] \quad \text{by} \quad [\![x]\!](\sigma) = \sigma(x) \end{aligned}$$

The first two clearly are continuous, since they are constant maps. As for the last, this mapping amounts to projection of the element $\sigma \in \mathbb{C}^{\mathcal{V}}$ onto its x-component, and since we endow $\mathbb{C}^{\mathcal{V}}$ with the product topology, this mapping also is continuous. Obviously, these three maps satisfy the condition of Definition 4.1, hence they are in $\mathcal{D}$.

Next we define the semantics of strict and weak sequential composition, of parallel composition and of hiding. Rather than define the interpretations of these operators directly at the level of $\mathcal{D} \subseteq [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$, we instead define continuous interpretations on $\mathbb{C}$, and then extend to $[\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$ in a pointwise fashion. Proposition 4.3 is the link that shows this approach induces continuous interpretations on $[\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$. It will then remain to show that these extensions still satisfy the condition of Definition 4.1.

Proposition 4.3 (cf. [1]) *Let* $\mathrm{op} \in \Omega_n$ *be an n-ary operator of our language and assume that we have defined a corresponding* continuous *operation* $\overline{\mathrm{op}} : \mathbb{C}^n \to \mathbb{C}$. *We define the interpretation of the operation* op *on* $[\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$ *in a pointwise fashion:*

$$\widetilde{\mathrm{op}} : [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]^n \to [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}] \ \textit{by} \ \widetilde{\mathrm{op}}(f_1, \ldots, f_n)(\sigma) = \overline{\mathrm{op}}(f_1(\sigma), \ldots, f_n(\sigma)).$$

Then

(i) $\widetilde{\mathrm{op}}(f_1, \ldots, f_n) : \mathbb{C}^{\mathcal{V}} \to \mathbb{C}$ *is a continuous map, and*

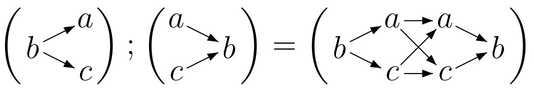
(ii) $\widetilde{\mathrm{op}} : [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]^n \to [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$ *also is continuous.* □

In the following we may use the same notation for op, $\overline{\mathrm{op}}$ and $\widetilde{\mathrm{op}}$. The actual operation should always be clear from the context.

4.1 Strict Sequential Composition

We now give the definition of strict sequential composition. We start with the definition on real pomsets, where it is only defined when the first pomset is finite.

Definition 4.4 Let $s_1 = (V_1, \leq_1, \lambda_1)$ and $s_2 = (V_2, \leq_2, \lambda_2)$ be two real pomsets. If $s_1 \in \mathbb{R}_f$ is finite, then the strict sequential composition of s_1 and s_2 is defined by $s_1 \,;\, s_2 = (V_1 \dot\cup V_2, \leq_1 \dot\cup \leq_2 \dot\cup V_1 \times V_2, \lambda_1 \dot\cup \lambda_2)$.

For instance, we have  .

Note that $s_1 \,;\, s_2$ satisfies the conditions of Definition 2.2, hence ; is a well defined operation from $\mathbb{R}_f \times \mathbb{R}$ to $\mathbb{R}$. Now, we extend the definition of strict sequential composition to complex pomsets.

Definition 4.5 We define the strict sequential composition of $x = (r, R) \in \mathbb{C}$ and $y = (s, S) \in \mathbb{C}$ by

$$x \,;\, y = \begin{cases} (r \,;\, s, S) & \text{if } R = \emptyset, \\ (r, R \cup \operatorname{res}(y)) & \text{otherwise.} \end{cases}$$

Note that if $R = \emptyset$, then r must be finite. Hence, $r \,;\, s \in \mathbb{R}$ is well-defined and $\operatorname{resinf}(r \,;\, s) = \operatorname{resinf}(s) \subseteq S$. It follows directly that $x \,;\, y$ is a complex pomset, hence ; is a well-defined operation from $\mathbb{C}^2$ to $\mathbb{C}$.

Proposition 4.6

(i) $; : \mathbb{C}^2 \to \mathbb{C}$ *is continuous.*

(ii) *For all* $x, y \in \mathbb{C}$, *we have* $\operatorname{res}(x \,;\, y) = \operatorname{res}(x) \cup \operatorname{res}(y)$.

Proof. (i) We start by showing that ; is monotone in the second argument. Let $x = (r, R)$, $y = (s, S)$ and $y' = (s', S')$ be complex pomsets with $y \sqsubseteq y'$. If $R = \emptyset$ then $x \,;\, y = (r \,;\, s, S)$ and $x \,;\, y' = (r \,;\, s', S')$. Clearly, we have $r \,;\, s \leq r \,;\, s'$ and $(r \,;\, s)^{-1}(r \,;\, s') = s^{-1}s'$. Hence, we have $x \,;\, y \sqsubseteq x \,;\, y'$. If $R \neq \emptyset$ then $x \,;\, y = (r, R \cup \operatorname{res}(y)) \sqsubseteq (r, R \cup \operatorname{res}(y')) = x \,;\, y'$ since $y \sqsubseteq y'$ implies $\operatorname{res}(y) \supseteq \operatorname{res}(y')$.

Next, we show that ; is continuous in the second argument. Again, consider $x = (r, R) \in \mathbb{C}$ and let $Y \subseteq \mathbb{C}$ be directed. First, assume that $R = \emptyset$; then we have

$$\bigsqcup (x \,;\, Y) = \left(\bigsqcup (r \,;\, \operatorname{Re}(Y)), \bigsqcup \operatorname{Im}(Y) \right) = \left(r \,;\, \bigsqcup \operatorname{Re}(Y), \bigsqcup \operatorname{Im}(Y) \right) = x \,;\, \bigsqcup Y.$$

Second, if $R \neq \emptyset$, then

$$\bigsqcup (x \,;\, Y) = \left(r, \bigcap_{y \in Y} (R \cup \operatorname{res}(y)) \right) = \left(r, R \cup \bigcap_{y \in Y} \operatorname{res}(y) \right) = x \,;\, \bigsqcup Y.$$

Now, we show that ; is monotone in the first argument. Let $x = (r, R)$, $x' = (r', R')$ and $y = (s, S)$ be complex pomsets with $x \sqsubseteq x'$. If $R = \emptyset$ then $x = x'$ and the result is clear. Otherwise, $x \,;\, y = (r, R \cup \mathrm{res}(y))$ and since $x \sqsubseteq x'$ we deduce easily that $x\,;y \sqsubseteq (r', R' \cup \mathrm{res}(y)) \sqsubseteq x'\,;y$ (note that R' may be empty).

It remains to show that ; is continuous in the first argument. Let $X \subseteq \mathbb{C}$ be directed and let $y \in \mathbb{C}$. If $\mathrm{Im}(\sqcup X) = \emptyset$, then $\sqcup X$ is compact and we deduce that $\sqcup X \in X$. It follows easily that $\sqcup(X\,;y) = (\sqcup X)\,;y$. Now, if $\mathrm{Im}(\sqcup X) \neq \emptyset$, then for all $x \in X$ we also have $\mathrm{Im}(x) \neq \emptyset$. Therefore,

$$\begin{aligned}\bigsqcup(X\,;y) &= \bigsqcup_{x\in X}(\mathrm{Re}(x), \mathrm{Im}(x) \cup \mathrm{res}(y)) = \left(\bigsqcup \mathrm{Re}(X), \bigcap_{x\in X}(\mathrm{Im}(x) \cup \mathrm{res}(y))\right) \\ &= \left(\bigsqcup \mathrm{Re}(X), \left(\bigcap_{x\in X} \mathrm{Im}(x)\right) \cup \mathrm{res}(y)\right) = \left(\bigsqcup X\right)\,; y.\end{aligned}$$

(ii) is clear. □

The interpretation of strict sequential composition on $\mathcal{D}$ is then

$$;: \mathcal{D}^2 \to \mathcal{D} \quad \text{defined by} \quad (f_1\,;f_2)(\sigma) = f_1(\sigma)\,;f_2(\sigma).$$

By Proposition 4.3, we know that $f_1;f_2$ is continuous and by Proposition 4.6(ii), we deduce immediately that it satisfies the condition of Definition 4.1.

We conclude this section with a lemma that will be useful in order to relate the operational semantics and the denotational semantics.

Lemma 4.7 *Let $x, y \in \mathbb{C}$ and $a \in \Sigma$. Then,*

(i) *$a \leq x\,;y$ implies $a \leq x$ or $a \leq y$ and $x = (1, \emptyset)$;*

(ii) *$a \leq x$ implies $a \leq x\,;y$ and $a^{-1}(x\,;y) = (a^{-1}x)\,;y$;*

(iii) *$a \leq y$ and $x = (1, \emptyset)$ implies $a \leq x\,;y$ and $a^{-1}(x\,;y) = a^{-1}y = x\,;(a^{-1}y)$.*

Proof. Let $x = (r, R) \in \mathbb{C}$, $y = (s, S) \in \mathbb{C}$ and $a \in \Sigma$.

(i) Assume that $a \leq x\,;y$. If $x = (1, \emptyset)$ then $x\,;y = y$ and we get $a \leq y$. Otherwise, we must have $a \leq r$ and we get $a \leq x$.

(ii) If $a \leq x$, then $a \leq r$ and $a^{-1}x = (a^{-1}r, R)$. It is then easy to check that $a \leq x\,;y$ and $a^{-1}(x\,;y) = (a^{-1}x)\,;y$.

(iii) If $a \leq y$ and $x = (1, \emptyset)$ then $x\,;y = y$ and the result is clear. □

4.2 Weak Sequential Composition

We start with the definition of weak sequential composition for real pomsets. As for the previous operator, this one also is only partially defined.

Definition 4.8 Let $s_1 = (V_1, \leq_1, \lambda_1)$ and $s_2 = (V_2, \leq_2, \lambda_2)$ be two real pomsets. If $\mathrm{resinf}(s_1) \cap \mathrm{res}(s_2) = \emptyset$, then the weak sequential composition of s_1 and s_2 is defined by $s_1 \circ s_2 = (V, \leq, \lambda)$ where

- $V = V_1 \dot\cup V_2$,
- $\leq$ is the transitive closure of $\leq_1 \dot\cup \leq_2 \dot\cup \{(p,q) \in V_1 \times V_2 \mid \lambda(p) \cap \lambda(q) \neq \emptyset\}$, and
- $\lambda = \lambda_1 \dot\cup \lambda_2$.

For instance, if $a \cap c = \emptyset$ then
$$\left(\begin{tikzcd}[row sep=tiny, column sep=small] & a \\ b \arrow[ur] \arrow[dr] & \\ & c \end{tikzcd}\right) \circ \left(\begin{tikzcd}[row sep=tiny, column sep=small] a \arrow[dr] & \\ & b \\ c \arrow[ur] & \end{tikzcd}\right) = \left(\begin{tikzcd}[row sep=tiny, column sep=small] & a \arrow[r] & a \arrow[dr] & \\ b \arrow[ur] \arrow[dr] & & & b \\ & c \arrow[r] & c \arrow[ur] & \end{tikzcd}\right).$$
Also, if $a \cap c = \emptyset$ then $a^\omega \circ c^\omega = \left(\begin{tikzcd}[row sep=small, column sep=small] a \arrow[r] & a \arrow[r] & a & \cdots \\ c \arrow[r] & c \arrow[r] & c & \cdots \end{tikzcd}\right)$. But if $a \cap c \neq \emptyset$ then $a^\omega \circ c^\omega$ is not defined.

Note that, by definition, $s_1 \circ s_2$ satisfies the over-synchronization condition and $\mathrm{res}(s_1 \circ s_2) = \mathrm{res}(s_1) \cup \mathrm{res}(s_2)$. Also, for all $p \in V_2$, $\downarrow_{\leq} p = \downarrow_{\leq_2} p \cup \downarrow_{\leq_1} \{q \in V_1 \mid \lambda(p) \cap \lambda(q) \neq \emptyset\}$. Since $\mathrm{resinf}(s_1) \cap \mathrm{res}(s_2) = \emptyset$, the set $\{q \in V_1 \mid \lambda(p) \cap \lambda(q) \neq \emptyset\}$ must be finite and therefore $\downarrow_{\leq} p$ is finite. It follows that $s_1 \circ s_2$ satisfies the conditions of Definition 2.2, hence $\circ$ is well defined on its domain $\mathbb{R}_\circ = \{(s_1, s_2) \in \mathbb{R}^2 \mid \mathrm{resinf}(s_1) \cap \mathrm{res}(s_2) = \emptyset\}$.

In order to extend the definition to complex pomsets, we need to restrict the second complex pomset so that the weak sequential composition of the real parts is possible. For this we introduce the following map.

Definition 4.9 We define the mapping $f : \mathcal{P}(\mathcal{R}) \times \mathbb{C} \to \mathbb{C}$ by

$$(R, y) \mapsto f_R(y) = \sqcup \{x \in \mathbb{C} \mid x \sqsubseteq y \text{ and } \mathrm{res}(\mathrm{Re}(x)) \cap R = \emptyset\}.$$

We also use the notation $f_R(y) = (\mu_R(y), \sigma_R(y))$.

For instance, if $R \cap c \neq \emptyset$ and $R \cap (a \cup b) = \emptyset$ then,

$$f_R\left(\left(\begin{tikzcd}[row sep=tiny, column sep=small] & a \arrow[r] & a \arrow[dr] & \\ b \arrow[ur] \arrow[dr] & & & b \\ & c \arrow[r] & c \arrow[ur] & \end{tikzcd}, \ \emptyset\right)\right) = \left(\begin{tikzcd}[column sep=small] b \arrow[r] & a \arrow[r] & a \end{tikzcd}, \ b \cup c\right).$$

Proposition 4.10

(i) *The mapping $f : (\mathcal{P}(\mathcal{R}), \supseteq) \times (\mathbb{C}, \sqsubseteq) \to (\mathbb{C}, \sqsubseteq)$ is continuous.*

(ii) *For all $R \subseteq \mathcal{R}$ and $y \in \mathbb{C}$, we have*
- $\mu_R(y) = \sqcup \{r \in \mathbb{R} \mid r \leq \mathrm{Re}(y) \text{ and } \mathrm{res}(r) \cap R = \emptyset\}$,
- $\sigma_R(y) = \mathrm{Im}(y) \cup \mathrm{res}(\mu_R(y)^{-1}\mathrm{Re}(y))$, *and*
- $\mathrm{res}(f_R(y)) = \mathrm{res}(y)$.

Proof. (i) Since $\mathcal{P}(\mathcal{R})$ is ordered by reverse containment, it is clear that f is monotone.

We show that f is continuous in its first argument. Let $X \subseteq \mathcal{P}(\mathcal{R})$ be directed and $y \in \mathbb{C}$. We have $\sqcup X = \cap_{R \in X} R$. Since $\mathrm{res}(y)$ is finite and X is directed, there exists $R \in X$ such that $R \cap \mathrm{res}(y) = (\sqcup X) \cap \mathrm{res}(y)$. Since $\mathrm{res}(\mathrm{Re}(x)) \subseteq \mathrm{res}(y)$ for all $x \sqsubseteq y$, we deduce easily that $f_R(y) = f_{\sqcup X}(y)$ which implies continuity.

We show that f is continuous in its second argument. Let $R \in \mathcal{P}(\mathcal{R})$ and $Y \subseteq \mathbb{C}$ be directed. We know that $f_R(Y)$ is directed and $\sqcup f_R(Y) \sqsubseteq f_R(\sqcup Y)$.

Conversely, let $z \in \mathbb{C}$ be compact and such that $z \sqsubseteq f_R(\sqcup Y)$. The set $\{x \in \mathbb{C} \mid x \sqsubseteq \sqcup Y \text{ and } \mathrm{res}(\mathrm{Re}(x)) \cap R = \emptyset\}$ is directed, hence there exists $x \sqsubseteq \sqcup Y$ such that $z \sqsubseteq x$ and $\mathrm{res}(\mathrm{Re}(x)) \cap R = \emptyset$. It follows that $\mathrm{res}(\mathrm{Re}(z)) \cap R = \emptyset$ and there exists $y \in Y$ such that $z \sqsubseteq y$. We deduce $z \sqsubseteq f_R(y) \sqsubseteq \sqcup f_R(Y)$. Since $\mathbb{C}$ is algebraic (Proposition 2.16), we conclude that $f_R(\sqcup Y) \sqsubseteq \sqcup f_R(Y)$.

(ii) For $R \subseteq \mathcal{R}$ and $y \in \mathbb{C}$, we let $X = \{x \in \mathbb{C} \mid x \sqsubseteq y \text{ and } \mathrm{res}(\mathrm{Re}(x)) \cap R = \emptyset\}$. Then, $\mathrm{Re}(X) = \{r \in \mathbb{R} \mid r \leq \mathrm{Re}(y) \text{ and } \mathrm{res}(r) \cap R = \emptyset\}$. Indeed, if $r \in \mathbb{R}$ satisfies $r \leq \mathrm{Re}(y)$ and $\mathrm{res}(r) \cap R = \emptyset$, then $x = (r, \mathrm{Im}(y) \cup \mathrm{res}(r^{-1}\mathrm{Re}(y))) \in X$ which shows one inclusion. The other inclusion is clear. We deduce that $\mu_R(y) = \mathrm{Re}(\sqcup X) = \sqcup \mathrm{Re}(X)$ which is the desired equality.

Now, $f_R(y) \sqsubseteq y$, hence $\mathrm{Im}(y) \cup \mathrm{res}(\mu_R(y)^{-1}\mathrm{Re}(y)) \subseteq \sigma_R(y)$. Conversely, $x = (\mu_R(y), \mathrm{Im}(y) \cup \mathrm{res}(\mu_R(y)^{-1}\mathrm{Re}(y))) \sqsubseteq y$ and $\mathrm{res}(\mu_R(y)) \cap R = \emptyset$. Hence $x \in X$ and from $x \sqsubseteq \sqcup X = f_R(y)$ we deduce that $\mathrm{Im}(x) \supseteq \sigma_R(y)$.

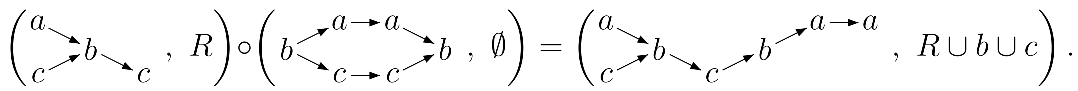
That $\mathrm{res}(f_R(y)) = \mathrm{res}(y)$ is clear from the above results. $\square$

Now, we can extend the definition of weak sequential composition to complex pomsets.

Definition 4.11 We define the weak sequential composition of $x, y \in \mathbb{C}$ by

$$x \circ y = x \circ f_{\mathrm{Im}(x)}(y) = (\mathrm{Re}(x) \circ \mu_{\mathrm{Im}(x)}(y), \mathrm{Im}(x) \cup \sigma_{\mathrm{Im}(x)}(y)).$$

For instance, if $R \cap c \neq \emptyset$, $R \cap (a \cup b) = \emptyset$ and $b \cap c \neq \emptyset$ then,



Note that $\mathrm{resinf}(\mathrm{Re}(x)) \subseteq \mathrm{Im}(x)$ and $\mathrm{res}(\mu_{\mathrm{Im}(x)}(y)) \cap \mathrm{Im}(x) = \emptyset$, so $\mathrm{Re}(x) \circ \mu_{\mathrm{Im}(x)}(y)$ is well-defined. Moreover, $\mathrm{resinf}(\mathrm{Re}(x) \circ \mu_{\mathrm{Im}(x)}(y)) \subseteq \mathrm{Im}(x) \cup \mathrm{Im}(y)$. Finally, it is clear that $\mathrm{res}(x \circ y) \subseteq \mathrm{res}(x) \cup \mathrm{res}(y)$ is finite. Hence the mapping $\circ : \mathbb{C}^2 \to \mathbb{C}$ is well-defined.

Proposition 4.12

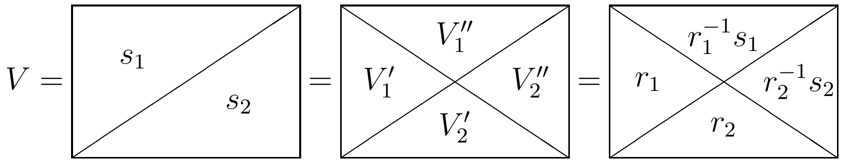
(i) $\circ : \mathbb{C}^2 \to \mathbb{C}$ *is continuous.*

(ii) *For all* $x, y \in \mathbb{C}$, *we have* $\mathrm{res}(x \circ y) = \mathrm{res}(x) \cup \mathrm{res}(y)$.

Proof. (i) We first introduce the mapping $\Phi : \mathbb{C}^2 \to \mathbb{C}^2$ defined by $\Phi(x, y) = (x, f_{\mathrm{Im}(x)}(y))$. Since f is continuous in its second argument, we deduce that the same holds for Φ. Now, $\mathrm{Im} : \mathbb{C} \to (\mathcal{P}(\mathcal{R}), \supseteq)$ is also continuous and since f is continuous in its first argument, we deduce that the same holds for Φ. Hence Φ is continuous. Note that $\Phi(\mathbb{C}^2) = \mathbb{C}_\circ = \{(x, y) \in \mathbb{C}^2 \mid \mathrm{Im}(x) \cap \mathrm{res}(\mathrm{Re}(y)) = \emptyset\}$.

Second, we let Ψ be the restriction of $\circ$ to $\mathbb{C}_\circ$. Note that, for all $(x, y) \in \mathbb{C}_\circ$, we have the following much simpler composition $\Psi(x, y) = x \circ y = (\mathrm{Re}(x) \circ \mathrm{Re}(y), \mathrm{Im}(x) \cup \mathrm{Im}(y))$. We will show that Ψ is continuous. Since the general composition $x \circ y = \Psi(\Phi(x, y))$, this will conclude the proof of (i). Note that $\mathbb{C}_\circ$ is not a product of two subsets of $\mathbb{C}$, hence, it is not enough to show separate continuity.

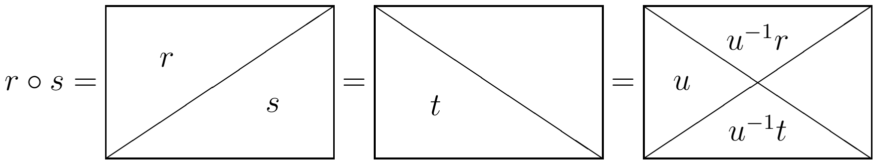
We start by showing the Ψ is monotone. Let $(x_1,x_2),(y_1,y_2) \in \mathbb{C}_\circ$ with $x_i \sqsubseteq y_i$ for $i=1,2$. We let $x_i=(r_i,R_i)$, $y_i=(s_i,S_i)$, $s_i=(V_i,\leq_i,\lambda_i)$ and $s_1 \circ s_2 = (V,\leq,\lambda)$. Since $r_i \leq s_i$ we have $V_i = V_i' \dot\cup V_i''$ with $r_i=(V_i',\leq_i,\lambda_i)$ and $r_i^{-1}s_i = (V_i'',\leq_i,\lambda_i)$:



Note that $\mathrm{res}(V_1'') = \mathrm{res}(r_1^{-1}s_1) \subseteq \mathrm{Im}(x_1)$, hence $r_1^{-1}s_1 I r_2$. Using Definition 4.8 we deduce that $V_1' \cup V_2'$ is a lower set in V. Therefore, $r_1 \circ r_2 \leq s_1 \circ s_2$ and $(r_1 \circ r_2)^{-1}(s_1 \circ s_2) = (r_1^{-1}s_1) \circ (r_2^{-1}s_2)$. We deduce that $\mathrm{res}((r_1 \circ r_2)^{-1}(s_1 \circ s_2)) \subseteq R_1 \cup R_2$. Since we also have $S_1 \cup S_2 \subseteq R_1 \cup R_2$ we have shown $\Psi(x_1,x_2) = x_1 \circ x_2 \sqsubseteq y_1 \circ y_2 = \Psi(y_1,y_2)$.

Next, we show that Ψ is continuous. Let $Z \subseteq \mathbb{C}_\circ$ be directed. We let X and Y be the first and second projections of Z. X and Y are both directed and $\sqcup Z = (\sqcup X, \sqcup Y)$. Let $\sqcup X = (r,R)$ and $\sqcup Y = (s,S)$. By Lemma 2.12, there exist $x_1 \in X$ and $y_1 \in Y$ with $\mathrm{Im}(x_1) = R$, $\mathrm{Im}(y_1) = S$ and $\mathrm{res}(\mathrm{Re}(y_1)) = \mathrm{res}(s)$. Since Z is directed, we find $z_2 = (x_2,y_2) \in Z$ with $x_1 \sqsubseteq x_2 \sqsubseteq \sqcup X$ and $y_1 \sqsubseteq y_2 \sqsubseteq \sqcup Y$. It follows that $\mathrm{Im}(x_2) = R$ and $\mathrm{res}(\mathrm{Re}(y_2)) = \mathrm{res}(s)$. Since $z_2 \in \mathbb{C}_\circ$, we also have $\sqcup Z \in \mathbb{C}_\circ$. Now, from Ψ monotone, we deduce that $\Psi(Z)$ is directed and $\sqcup\Psi(Z) \sqsubseteq \Psi(\sqcup Z)$.

We have $\mathrm{Im}(\Psi(\sqcup Z)) \subseteq \mathrm{Im}(\sqcup\Psi(Z)) \subseteq \mathrm{Im}(\Psi(z_2)) = R \cup S = \mathrm{Im}(\Psi(\sqcup Z))$. Also, $\mathrm{Re}(\sqcup\Psi(Z)) \leq r \circ s = \mathrm{Re}(\Psi(\sqcup Z))$. In order to prove the converse, we start with a compact real pomset $t \in \mathbb{R}_f$ such that $t \leq r \circ s$. Let $(V,\leq)$ be the canonical representation of $r \circ s$. In order to lighten the notations, we simply write $\leq$ instead of $\leq|_{U \times U}$ for all subsets $U \subseteq V$. Let $(V_r,\leq)$ and $(V_t,\leq)$ be the canonical representations of r and t. Since r and t are bounded above by $r \circ s$, Proposition 2.7 implies that $u = r \sqcap t = (V_r \cap V_t, \leq)$. We also have $u^{-1}r = (V_r \setminus V_t, \leq)$ and $u^{-1}t = (V_t \setminus V_r, \leq)$. Since V_r and V_t are lower sets of $(V,\leq)$, there is no ordering between the vertices of $V_r \setminus V_t$ and $V_t \setminus V_r$. Then, the over-synchronization condition implies that $u^{-1}r \; I \; u^{-1}t$. The situation is summarized in the following picture



We have $u \leq r = \sqcup\mathrm{Re}(X)$ and $u^{-1}t \leq s = \sqcup\mathrm{Re}(Y)$. Since t is finite, u and $u^{-1}t$ are compact. Using Z directed, we find $z = (x,y) \in Z$ with $u \leq \mathrm{Re}(x) \leq r$ and $u^{-1}t \leq \mathrm{Re}(y) \leq s$. Now, from $u^{-1}\mathrm{Re}(x) \leq u^{-1}r \; I \; u^{-1}t$ we deduce that $t = u \circ u^{-1}t \leq \mathrm{Re}(x) \circ \mathrm{Re}(y) = \mathrm{Re}(\Psi(x,y)) \leq \mathrm{Re}(\Psi(\sqcup Z))$. We conclude that $r \circ s \leq \mathrm{Re}(\Psi(\sqcup Z))$ since $\mathbb{R}$ is algebraic (Proposition 2.8).

(ii) is clear. □

The interpretation of weak sequential composition on $\mathcal{D}$ is then

$$\circ : \mathcal{D}^2 \to \mathcal{D} \quad \text{defined by} \quad (f_1 \circ f_2)(\sigma) = f_1(\sigma) \circ f_2(\sigma).$$

By Proposition 4.3, we know that $f_1 \circ f_2$ is continuous and from Proposition 4.12(ii), we deduce immediately that it satisfies the condition of Definition 4.1.

We conclude this section with a lemma that will be useful in relating the operational semantics and the denotational semantics.

Lemma 4.13 *Let $x, y \in \mathbb{C}$ and $a \in \Sigma$. Then,*

(i) *$a \le x \circ y$ implies $a \le x$ or $a \le y$ and $a\, I\, x$;*

(ii) *$a \le x$ implies $a \le x \circ y$ and $a^{-1}(x \circ y) = (a^{-1}x) \circ y$;*

(iii) *$a \le y$ and $a\, I\, x$ implies $a \le x \circ y$ and $a^{-1}(x \circ y) = x \circ (a^{-1}y)$.*

Proof. (i) $a \le x \circ y$ means that $a \le \mathrm{Re}(x) \circ \mu_{\mathrm{Im}(x)}(y)$ and we get either $a \le \mathrm{Re}(x) \le x$ or $a I \mathrm{Re}(x)$ and $a \le \mu_{\mathrm{Im}(x)}(y) \le y$. Since $\mathrm{Im}(x) \cap \mathrm{res}(\mu_{\mathrm{Im}(x)}(y)) = \emptyset$, we have $a\, I\, x$ in the last case.

(ii) is easy.

(iii) Assume $a \le y$ and $a\, I\, x$. We have $a\, I\, \mathrm{Im}(x)$ and $a \le \mathrm{Re}(y)$ hence $a \le \mu_{\mathrm{Im}(x)}(y)$. Also, $a I \mathrm{Re}(x)$ so $a \le \mathrm{Re}(x) \circ \mu_{\mathrm{Im}(x)}(y)$ and moreover $a^{-1}(\mathrm{Re}(x) \circ \mu_{\mathrm{Im}(x)}(y)) = \mathrm{Re}(x) \circ (a^{-1}\mu_{\mathrm{Im}(x)}(y))$ which concludes the proof. □

4.3 Parallel Composition

Parallel composition is a bit tricky to deal with because our semantics does not support nondeterminism, and so we must obtain a single complex pomset as the denotational semantics of any process. This is done on purpose to demonstrate clearly that we have a truly concurrent semantics that does not require nondeterminism to deal with parallel composition.

In this section, real pomsets will always be described by their canonical representations. Recall that if $s \in \mathbb{R}$ is a real pomset then its canonical representation is $s = (V, \le)$ where $V = \{(a, i) \in \Sigma \times \mathbb{N} \mid 0 \le i < |s|_a\}$ and the uniform labelling on vertices from $\Sigma \times \mathbb{N}$ is given by $\lambda(a, i) = a$.

We start with the parallel composition of real pomsets. Let $r_1, r_2 \in \mathbb{R}$ be two real pomsets with canonical representations $r_i = (U_i, \le_i)$. We define

$$r = r_1 \parallel r_2 = (U, \le) = (U_1 \cup U_2, (\le_1 \cup \le_2)^*).$$

Note that U_1, U_2 need not be disjoint and, contrary to what was done for strict or weak sequential compositions, we do not take a disjoint union. Due to this, $r_1 \parallel r_2$ is not necessarily a real pomset. First, $\le$ may fail to be antisymmetric, which is the case for instance if $r_1 = a\,;b$ and $r_2 = b\,;a$. Second, $\le$ may fail to be over-synchronized, as for $r_1 = a$, $r_2 = b$ with $a \ne b$ and $a \cap b \ne \emptyset$. The interesting case is when $r_1 \| r_2$ is a real pomset. For instance, if $r_1 = a;b;a;a;b$, $r_2 = b\,;c\,;b\,;c$, and $a \cap c = \emptyset$ then $r_1 \parallel r_2$ is the following real pomset

$$r_1 \parallel r_2 = \left(\begin{tikzcd}[column sep=small, row sep=tiny]
(a,1) \arrow[dr] & & (a,2) \arrow[r] & (a,3) \arrow[dr] & & \\
& (b,1) \arrow[ur] \arrow[dr] & & & (b,2) \arrow[dr] & \\
& & (c,1) \arrow[urr] & & & (c,2)
\end{tikzcd}\right)$$

Recall that for $a \in \Sigma$ and $C \subseteq \Sigma$, we use $a\,I\,C$ to denote the fact $a\,I\,c$ for all $c \in C$. Also, if $x \in \mathbb{C}$ then $a\,I\,C, x$ stands for $a\,I\,C$ and $a\,I\,x$.

Definition 4.14 Let $x_1, x_2 \in \mathbb{C}$ and $C \subseteq \Sigma$. We define $\mathbb{R}_C(x_1, x_2) \subseteq \mathbb{R}^2$ by $(r_1, r_2) \in \mathbb{R}_C(x_1, x_2)$ if the following conditions hold

(i) $r_1 \leq \mathrm{Re}(x_1)$ and $r_2 \leq \mathrm{Re}(x_2)$;

(ii) $|r_1|_c = |r_2|_c$ for all $c \in C$;

(iii) for all $a \in \mathrm{alph}(r_1)$ we have $a \in C$ or $a\,I\,C, x_2$; and for all $a \in \mathrm{alph}(r_2)$ we have $a \in C$ or $a\,I\,C, x_1$;

(iv) $r_1 \parallel r_2 \in \mathbb{R}$ is a real pomset.

We begin by showing some important properties of $\mathbb{R}_C(x_1, x_2)$.

Lemma 4.15 *Let $x_1, x_2 \in \mathbb{C}$ and $C \subseteq \Sigma$.*

(i) *$\mathbb{R}_C(x_1, x_2)$ is a consistent subset of $\mathbb{R}^2$.*

(ii) *We have $(r_1, r_2) = \sqcup \mathbb{R}_C(x_1, x_2) \in \mathbb{R}_C(x_1, x_2)$.*

(iii) *For all $(r_1', r_2') \in \mathbb{R}_C(x_1, x_2)$, we have $r_1' \parallel r_2' \leq r_1 \parallel r_2$.*

Proof. Let $x_i = (s_i, S_i)$ and $s_i = (V_i, \leq_i)$ for $i = 1, 2$. $\mathbb{R}_C(x_1, x_2)$ is bounded above by (s_1, s_2), hence it is consistent, which proves (i).

(ii) Let $(r_1, r_2) = \sqcup \mathbb{R}_C(x_1, x_2)$ with $r_i = (U_i, \leq_i)$ for $i = 1, 2$ and let $r = r_1 \parallel r_2 = (U, \leq)$.

- Clearly, $r_1 \leq s_1$ and $r_2 \leq s_2$.
- Let $c \in C$ and k such that $0 \leq k < |r_1|_c$. There exists $(r_1', r_2') \in \mathbb{R}_C(x_1, x_2)$ with $|r_1'|_c > k$. We deduce that $|r_2|_c \geq |r_2'|_c = |r_1'|_c > k$. Therefore, $|r_1|_c = |r_2|_c$ for all $c \in C$.
- Let $\{i, j\} = \{1, 2\}$ and $a \in \mathrm{alph}(r_i)$. There exists $(r_1', r_2') \in \mathbb{R}_C(x_1, x_2)$ with $a \in \mathrm{alph}(r_i')$. We deduce that $a \in C$ or $a\,I\,C, x_j$.
- It remains to show that $r_1 \parallel r_2 \in \mathbb{R}$ is a real pomset.

  We first show the over-synchronization condition. Let $p, q \in U$ with $\lambda(p) \cap \lambda(q) \neq \emptyset$. Assume for instance that $p \in U_1$. Then $\lambda(q)$ is not independent of x_1. We show that $q \in U_1$. If $q \in U_2$ then using the third point above, we must have $\lambda(q) \in C$. Now, using the second point we get $q \in U_1 \cap U_2$. Since $p, q \in U_1 \subseteq V_1$ and s_1 is a real pomset we deduce that $p \leq_1 q$ or $q \leq_1 p$ and it follows $p \leq q$ or $q \leq p$.

  Now, we prove the following claim: Let $(r_1', r_2') \in \mathbb{R}_C(x_1, x_2)$ and $r' = r_1' \parallel r_2' = (U', \leq')$. Then, U' is a lower set of $(U, \leq)$ and $\leq' = \leq \cap (U' \times U')$.

Since $(r_1, r_2) = \sqcup\mathbb{R}_C(x_1, x_2)$ we know that $U' = U'_1 \cup U'_2 \subseteq U_1 \cup U_2 = U$ and by definition of $\leq'$ and $\leq$ we also get $\leq' \subseteq \leq \cap (U' \times U')$. Now, let $p \in U$, $q \in U'$ be such that $p \leq_1 q$. In this case, we have $p, q \in U_1 \subseteq V_1$ and q depends on x_1. As above, this implies $q \in U'_1$. Now, $p \leq_1 q \in U'_1 = \downarrow_1 U'_1 \subseteq V_1$ implies that $p \in U'_1 \subseteq U'$ and $p \leq' q$. Similarly, if $p \leq_2 q$ we obtain $p \in U'$ and $p \leq' q$. By induction, the same conclusion holds when $p \leq q$ since $\leq = (\leq_1 \cup \leq_2)^*$. We have shown that U' is a lower set of $(U, \leq)$ and also $\leq' \supseteq \leq \cap (U' \times U')$ which concludes the proof of the claim.

Let $q \in U$ and let $(r'_1, r'_2) \in \mathbb{R}_C(x_1, x_2)$ be such that $q \in U'$. Assume that $p \in U$ is such that $p \leq q \leq p$. From the claim above, we get $p \in U'$, $p \leq' q$ and $q \leq' p$. Since $r'_1 \parallel r'_2$ is a real pomset we deduce that $p = q$ showing that $\leq$ is antisymmetric. Finally, using again the claim above we deduce that $\{p \in U \mid p \leq q\} = \{p \in U' \mid p \leq' q\}$ which is finite since $r'_1 \parallel r'_2$ is a real pomset. Therefore, $r_1 \parallel r_2 = (U, \leq)$ satisfies all the conditions of Definition 2.2 and is indeed a real pomset.

(iii) is just the claim above once we know that $r_1 \parallel r_2$ is a real pomset.

□

We are now ready to give the definition of the parallel composition of complex pomsets.

Definition 4.16 Let $x_1, x_2 \in \mathbb{C}$ with $x_i = (s_i, S_i)$ and let $C \subseteq \Sigma$. Then,

$$x_1 \underset{C}{\parallel} x_2 = (r_1 \parallel r_2, S_1 \cup \mathrm{res}(r_1^{-1} s_1) \cup S_2 \cup \mathrm{res}(r_2^{-1} s_2))$$

where $(r_1, r_2) = \sqcup\mathbb{R}_C(x_1, x_2)$.

It is clear that $x_1 \underset{C}{\parallel} x_2$ is a complex pomset, hence $\underset{C}{\parallel} : \mathbb{C}^2 \to \mathbb{C}$ is a well-defined operation.

For instance, if a, b, c are pairwise independent actions and we let

$$x_1 = (a \to c \to a \to c \to a \to c \quad \cdots \quad , a \cup c),$$
$$x_2 = (c \to b \to c \to b \to c \to b \quad \cdots \quad , b \cup c).$$

then we get

$$x_1 \underset{c}{\parallel} x_2 = \left(\begin{tikzcd}[row sep=small, column sep=small] a \arrow[dr] & & a \arrow[dr] & & a \arrow[dr] & & \cdots \\ & c \arrow[ur] \arrow[dr] & & c \arrow[ur] \arrow[dr] & & c & \\ & & b \arrow[ur] & & b \arrow[ur] & & \cdots \end{tikzcd} , a \cup b \cup c \right).$$

On the other hand, parallel composition may introduce (partial) deadlocks when some non-synchronizing actions are not local and therefore may introduce conflicts. For instance, we consider five actions $a, b, c, d, e \in \Sigma$ with a, b, c pairwise independent and also independent from d, e, but we assume that d, e are dependent. We let

$$x_1 = (a \to c \to d \to c \to a \ , \ \emptyset),$$
$$x_2 = (b \to c \to e \to c \to b \ , \ \emptyset).$$

Due to the over-synchronization condition, $\left(\begin{tikzcd}[sep=small] a \arrow[dr] & & d \arrow[dr] & & a \\ & c \arrow[ur] \arrow[dr] & & c \arrow[ur] \arrow[dr] & \\ b \arrow[ur] & & e \arrow[ur] & & b \end{tikzcd}\right)$ is not a real pomset since the vertices labelled d and e would have to be ordered in any real pomset. Therefore, we get

$$x_1 \underset{c}{\|} x_2 = \left(\begin{tikzcd}[sep=small] a \arrow[dr] & \\ & c \\ b \arrow[ur] & \end{tikzcd} ,\ a \cup b \cup c \cup d \cup e\right).$$

Proposition 4.17 *Let $C \subseteq \Sigma$. Then,*

(i) $\underset{C}{\|} : \mathbb{C}^2 \to \mathbb{C}$ *is continuous.*

(ii) *For all $x_1, x_2 \in \mathbb{C}$, we have* $\mathrm{res}(x_1 \underset{C}{\|} x_2) = \mathrm{res}(x_1) \cup \mathrm{res}(x_2)$.

Proof. (ii) is clear. For (i), we start by showing that $\underset{C}{\|}$ is monotone. Let $x_1, x_2, x_1', x_2' \in \mathbb{C}$ be such that $x_i = (s_i, S_i) \sqsubseteq (s_i', S_i') = x_i'$ for $i = 1, 2$. From $\sqcup \mathbb{R}_C(x_1, x_2) = (r_1, r_2) \in \mathbb{R}_C(x_1, x_2)$ and $\mathrm{res}(x_i') \subseteq \mathrm{res}(x_i)$ for $i = 1, 2$, we deduce easily that $(r_1, r_2) \in \mathbb{R}_C(x_1', x_2')$.

Let $(r_1', r_2') = \sqcup \mathbb{R}_C(x_1', x_2')$, $r = r_1 \| r_2$ and $r' = r_1' \| r_2'$. We use the following notation for the canonical representations: $s_i' = (V_i', \leq_i)$, $s_i = (V_i, \leq_i)$, $r_i' = (U_i', \leq_i)$, $r_i = (U_i, \leq_i)$, $r' = (U', \leq')$ and $r = (U, \leq)$.

By Lemma 4.15, we have $r \leq r'$. Moreover, $U' \setminus U = (U_1' \setminus U_1) \cup (U_2' \setminus U_2)$. Indeed, the inclusion $\subseteq$ is trivial. Conversely, if $q \in U_1'$, then q depends on s_1', hence q depends on x_1 and we deduce that $q \in U_2$ implies $\lambda(q) \in C$ and therefore $q \in U_1 \cap U_2$. Therefore, $U_1' \setminus U_1 = U_1' \setminus (U_1 \cup U_2)$ which shows the reverse inclusion.

We deduce that $\mathrm{res}(r^{-1}r') = \mathrm{res}(r_1^{-1}r_1') \cup \mathrm{res}(r_2^{-1}r_2')$. Now, we have

$$S_i' \cup \mathrm{res}(r_i^{-1}r_i') \cup \mathrm{res}(r_i'^{-1}s_i') \subseteq S_i' \cup \mathrm{res}(r_i^{-1}s_i) \cup \mathrm{res}(s_i^{-1}s_i') \subseteq S_i \cup \mathrm{res}(r_i^{-1}s_i).$$

Therefore, $\mathrm{res}(r^{-1}r') \cup \mathrm{Im}(x_1' \underset{C}{\|} x_2') \subseteq \mathrm{Im}(x_1 \underset{C}{\|} x_2)$ and we have shown that $x_1 \underset{C}{\|} x_2 \sqsubseteq x_1' \underset{C}{\|} x_2'$.

In order to show that $\underset{C}{\|}$ is continuous, let $X \subseteq \mathbb{C}^2$ be directed and let $(x_1, x_2) = \sqcup X$. Since $\underset{C}{\|}$ is monotone, we know that $\underset{C}{\|}(X)$ is directed and $\sqcup \underset{C}{\|}(X) \sqsubseteq \underset{C}{\|}(\sqcup X)$. To show the reverse inequality, we fix some more notation: Let $x_i = (s_i, S_i)$ with $s_i = (V_i, \leq_i)$, $(r_1, r_2) = \sqcup \mathbb{R}_C(x_1, x_2)$ with $r_i = (U_i, \leq_i)$, and $x = (r, R) = x_1 \underset{C}{\|} x_2$ with $r = r_1 \| r_2 = (U, \leq)$.

Now, let $z = (t, T) \sqsubseteq (r, R) = x$ be finite with $t = (W, \leq)$. We have $W = {\downarrow} W \subseteq U = U_1 \cup U_2$. We introduce $W_i = W \cap U_i$ and we show that W_i is a lower set of $(U_i, \leq_i)$. If $p \in U_i$, $q \in W_i$ satisfy $p \leq_i q$, then $p, q \in U$ and $p \leq q$. Since $W = {\downarrow} W \subseteq U$ we deduce that $p \in W \cap U_i = W_i$.

We have seen that $(W_i, \leq_i) = t_i \leq r_i \leq s_i$. Since t_1 and t_2 are finite and X is directed, we find $(x_1', x_2') \in X$ with $t_i \leq \mathrm{Re}(x_i') \leq s_i$, $\mathrm{Im}(x_i') = \mathrm{Im}(x_i)$

and $\mathrm{res}(x_i') = \mathrm{res}(x_i)$ (Lemma 2.12). We will show that $z \sqsubseteq x_1' \parallel_C x_2' \sqsubseteq \sqcup \parallel_C (X)$ which will conclude the proof since $\mathbb{C}$ is algebraic (Proposition 2.16).

First, we show that $(t_1, t_2) \in \mathbb{R}_C(x_1', x_2')$. We know already that $t_i \leq \mathrm{Re}(x_i')$. Now, let $p \in W_1 \subseteq U$ with $\lambda(p) \in C$. Since $(r_1, r_2) \in \mathbb{R}_C(x_1, x_2)$, we deduce that $p \in U_1 \cap U_2$ and $p \in W_1 \cap W_2$. Therefore, $|t_1|_c = |t_2|_c$ for all $c \in C$. Next, assume that $p \in W_1 \subseteq U_1$ and $\lambda(p) \notin C$. Then, $\lambda(p)\, I\, C, x_2$ and since $\mathrm{res}(x_2) = \mathrm{res}(x_2')$, we deduce that $\lambda(p)\, I\, C, x_2'$. Finally, it is easy to show that $t_1 \parallel t_2 = (W_1 \cup W_2, (\leq_1 \cup \leq_2)^*) = (W, \leq) = t$ is indeed a real pomset, since the same is true of $r = r_1 \parallel r_2$ and $W_i = \downarrow W_i \subseteq U_i$.

We fix some more notation: $x_i' = (s_i', S_i)$ with $s_i' = (V_i', \leq_i)$, $(r_1', r_2') = \sqcup \mathbb{R}_C(x_1', x_2')$ with $r_i' = (U_i', \leq_i)$, and $x' = (r', R') = x_1' \parallel_C x_2'$ with $r' = r_1' \parallel r_2' = (U', \leq)$. From Lemma 4.15, we have $t \leq r'$, and, as above when we proved that $\parallel_C$ is monotone, we can show that $\mathrm{res}(t^{-1}r') = \mathrm{res}(t_1^{-1}r_1') \cup \mathrm{res}(t_2^{-1}r_2')$. Similarly, we also get $\mathrm{res}(t^{-1}r) = \mathrm{res}(t_1^{-1}r_1) \cup \mathrm{res}(t_2^{-1}r_2)$.

Now, since $t_i \leq r_i' \leq s_i' \leq s_i$ and $t_i \leq r_i \leq s_i$, we deduce that

$$\begin{aligned}
\mathrm{Im}(x_1' \parallel_C x_2') \cup \mathrm{res}(t^{-1}r') &= \bigcup_{i=1,2} S_i \cup \mathrm{res}(r_i'^{-1}s_i') \cup \mathrm{res}(t_i^{-1}r_i') = \bigcup_{i=1,2} S_i \cup \mathrm{res}(t_i^{-1}s_i') \\
&\subseteq \bigcup_{i=1,2} S_i \cup \mathrm{res}(t_i^{-1}s_i) = \bigcup_{i=1,2} S_i \cup \mathrm{res}(r_i^{-1}s_i) \cup \mathrm{res}(t_i^{-1}r_i) \\
&= \mathrm{Im}(x_1 \parallel_C x_2) \cup \mathrm{res}(t^{-1}r) \subseteq \mathrm{Im}(z),
\end{aligned}$$

where the last inclusion holds since $z \sqsubseteq x_1 \parallel_C x_2$. Thus, we have shown that $z \sqsubseteq x_1' \parallel_C x_2'$ which concludes the proof. □

The interpretation of parallel composition on $\mathcal{D}$ is then

$$\parallel_C : \mathcal{D}^2 \to \mathcal{D} \quad \text{defined by} \quad (f_1 \parallel_C f_2)(\sigma) = f_1(\sigma) \parallel_C f_2(\sigma).$$

By Proposition 4.3, we know that $f_1 \parallel_C f_2$ is continuous and by Proposition 4.17(ii), we deduce immediately that it satisfies the condition of Definition 4.1.

We conclude this section with a lemma that will be useful in relating the operational and denotational semantics.

Lemma 4.18 *Let $x_1, x_2 \in \mathbb{C}$ and $a \in \Sigma$. Then,*

(i) *$a \leq x_1 \parallel_C x_2$ implies $a \leq x_1$ and $a\, I\, C, x_2$, or $a \leq x_2$ and $a\, I\, C, x_1$, or $a \leq x_1$, $a \leq x_2$ and $a \in C$;*

(ii) *$a \leq x_1$ and $a\, I\, C, x_2$ imply $a \leq x_1 \parallel_C x_2$ and $a^{-1}(x_1 \parallel_C x_2) = (a^{-1}x_1) \parallel_C x_2$;*

(iii) *$a \leq x_2$ and $a\, I\, C, x_1$ imply $a \leq x_1 \parallel_C x_2$ and $a^{-1}(x_1 \parallel_C x_2) = x_1 \parallel_C (a^{-1}x_2)$;*

(iv) *$a \leq x_1$, $a \leq x_2$ and $a \in C$ imply $a \leq x_1 \parallel_C x_2$ and $a^{-1}(x_1 \parallel_C x_2) =$*

$(a^{-1}x_1) \underset{C}{\|} (a^{-1}x_2)$.

Proof. (i) Let $(r_1, r_2) = \sqcup \mathbb{R}_C(x_1, x_2)$ with $r_i = (U_i, \leq_i)$ and $r = r_1 \| r_2 = (U, \leq)$. If $a \leq r$, then $(a, 0)$ is minimal in $(U, \leq)$. If $(a, 0) \in U_1 \cap U_2$, then $a \leq r_1 \leq x_1$, $a \leq r_2 \leq x_2$ and $a \in C$ using Definition 4.14. If $(a, 0) \in U_1 \setminus U_2$, we get $a \leq r_1 \leq x_1$, and using Definition 4.14 we must have $a \; I \; C, x_2$. The last case is similar.

The proofs of (ii) and (iii) are similar to that of (iv) and are left to the reader. In order to prove (iv), we start with two claims.

Claim 1: Let $r_1, r_2 \in \mathbb{R}$ with $a \leq r_i$ and let $r'_i = a^{-1}r_i$. Then, $r_1 \| r_2 \in \mathbb{R}$ if and only if $r'_1 \| r'_2 \in \mathbb{R}$ and in this case, $a \leq r_1 \| r_2$ and $a^{-1}(r_1 \| r_2) = r'_1 \| r'_2$.

Let $r_i = (U_i, \leq_i)$ and $U'_i = U_i \setminus \{(a, 0)\}$. We have $r'_i = (U'_i, \leq_i)$ and even though this is not the canonical representation of r'_i, it is easy to check that $r'_1 \| r'_2 = (U', \leq') = (U'_1 \cup U'_2, (\leq_1 \cup \leq_2)^*)$. We also have $r_1 \| r_2 = (U, \leq) = (U_1 \cup U_2, (\leq_1 \cup \leq_2)^*)$.

Since $U' = U \setminus \{(a, 0)\}$ and $(a, 0)$ is minimal both in $(U_1, \leq_1)$ and in $(U_2, \leq_2)$, we can easily show that for all $p, q \in U'$, $p \leq q$ if and only if $p \leq' q$. Therefore, if $r_1 \| r_2 = (U, \leq) \in \mathbb{R}$, we immediately get $r'_1 \| r'_2 = (U', \leq') \in \mathbb{R}$, $a \leq r_1 \| r_2$ and $a^{-1}(r_1 \| r_2) = r'_1 \| r'_2$.

Conversely, assume that $r'_1 \| r'_2 = (U', \leq') \in \mathbb{R}$. We have shown above that $\leq' = \leq \cap (U' \times U')$. Since $\leq'$ is antisymmetric and $(a, 0)$ is minimal in $(U, \leq)$ we deduce that $\leq$ is antisymmetric. It is also clear that for all $q \in U'$, $\{p \in U \mid p \leq q\} \subseteq \{p \in U' \mid p \leq' q\} \cup \{(a, 0)\}$ is finite. Finally, let $p \in U'$ be such that $a \cap \lambda(p) \neq \emptyset$. If $p \in U_1$, we must have $(a, 0) \leq_1 p$ since r_1 is a real pomset with a as minimal letter. The case $p \in U_2$ is similar. Therefore, $(a, 0) \leq p$ and $(U, \leq)$ satisfies the over-synchronization condition. We have shown that $r_1 \| r_2 = (U, \leq) \in \mathbb{R}$. This concludes the proof of Claim 1.

Claim 2: Let $x_1, x_2 \in \mathbb{C}$ and $a \in C$ be such that $a \leq x_1$, $a \leq x_2$. Then,

$$\mathbb{R}_C(a^{-1}x_1, a^{-1}x_2) = \{(a^{-1}r_1, a^{-1}r_2) \mid (r_1, r_2) \in \mathbb{R}_C(x_1, x_2), a \leq r_1, a \leq r_2\}.$$

We let $x_i = (s_i, S_i)$ and $a^{-1}x_i = x'_i = (s'_i, S_i)$ with $s'_i = a^{-1}s_i$.

Let $(r_1, r_2) \in \mathbb{R}_C(x_1, x_2)$ with $a \leq r_i$ and let $r'_i = a^{-1}r_i$. Using Claim 1, we have $r'_1 \| r'_2 \in \mathbb{R}$. Now, from $a \leq r_i \leq s_i$, we deduce that $r'_i = a^{-1}r_i \leq a^{-1}s_i = s'_i$. Next, $|r'_1|_a = |r_1|_a - 1 = |r_2|_a - 1 = |r'_2|_a$ and if $a \neq c \in C$, we get $|r'_1|_c = |r_1|_c = |r_2|_c = |r'_2|_c$. Finally, if p is a vertex of r'_1 then p is a vertex of r_1 and we deduce that $\lambda(p) \in C$ or $\lambda(p) \; I \; C, x_2$. Since $\mathrm{res}(x'_2) \subseteq \mathrm{res}(x_2)$ we get $\lambda(p) \in C$ or $\lambda(p) \; I \; C, x'_2$. We have shown that $(r'_1, r'_2) \in \mathbb{R}_C(x'_1, x'_2)$.

Conversely, let $(r'_1, r'_2) \in \mathbb{R}_C(x'_1, x'_2)$. We have $r'_i \leq s'_i = a^{-1}s_i$ hence there is a unique $r_i \leq s_i$ with $a \leq r_i$ and $a^{-1}r_i = r'_i$. We show that $(r_1, r_2) \in \mathbb{R}_C(x_1, x_2)$. From Claim 1 we already know that $r_1 \| r_2 \in \mathbb{R}$ and by definition we have $r_i \leq s_i$. Next, $|r_1|_a = |r'_1|_a + 1 = |r'_2|_a + 1 = |r_2|_a$ and if $a \neq c \in C$, we get $|r_1|_c = |r'_1|_c = |r'_2|_c = |r_2|_c$. Finally, if p is a vertex of r_1 then, either $\lambda(p) = a \in C$ or p is a vertex of r'_1 and we get $\lambda(p) \in C$ or $\lambda(p) \; I \; C, x'_2$.

Since $x'_2 = a^{-1}x_2$ and $a \in C$ we deduce in the last case that $\lambda(p)\, I\, C, x_2$. This concludes the proof of Claim 2.

We return to the proof of (iv). Let $(r_1, r_2) = \sqcup \mathbb{R}_C(x_1, x_2)$. Since $(a, a) \in \mathbb{R}_C(x_1, x_2)$, we get $a \leq r_i$ and we let $r'_i = a^{-1}r_i$. We show that $(r'_1, r'_2) = \sqcup \mathbb{R}_C(x'_1, x'_2)$. By Claim 2, we know that $(r'_1, r'_2) \in \mathbb{R}_C(x'_1, x'_2)$. Conversely, if $(t'_1, t'_2) \in \mathbb{R}_C(x'_1, x'_2)$ then, using Claim 2 again, we find $(t_1, t_2) \in \mathbb{R}_C(x_1, x_2)$ with $t'_i = a^{-1}t_i$. Then, $t_i \leq r_i$ and we deduce $t'_i = a^{-1}t_i \leq a^{-1}r_i = r'_i$. Therefore, $(r'_1, r'_2) = \sqcup \mathbb{R}_C(x'_1, x'_2)$.

By Claim 1, we have $a \leq r_1 \parallel r_2 = \mathrm{Re}(x_1 \underset{C}{\parallel} x_2)$ and $a^{-1}(r_1 \parallel r_2) = r'_1 \parallel r'_2 = \mathrm{Re}(x'_1 \underset{C}{\parallel} x'_2)$. Since $\mathrm{res}(r'^{-1}_i s'_i) = \mathrm{res}(r^{-1}_i s_i)$, we have $\mathrm{Im}(x_1 \underset{C}{\parallel} x_2) = \mathrm{Im}(x'_1 \underset{C}{\parallel} x'_2)$ and we have proved (iv). $\square$

4.4 Hiding

As with the previous operators, we start with the definition of hiding on real pomsets. Hiding a set of resources R from a real pomset s amounts to removing from the labelling functions all resources from R and removing also those vertices that end up with an empty resource label.

Definition 4.19 Let $s = (V, \leq, \lambda) \in \mathbb{R}$ be a real pomset and let $R \subseteq \mathcal{R}$. We define $s \backslash R = (V', \leq', \lambda')$ by $V' = \{p \in V \mid \lambda(p) \setminus R \neq \emptyset\}$, $\leq' = \leq \cap (V' \times V')$, and $\lambda'(p) = \lambda(p) \setminus R$ for all $p \in V'$.

It is easy to see that $s\backslash R$ satisfies all the conditions of Definition 2.2, hence we have a well defined mapping $\backslash R : \mathbb{R} \to \mathbb{R}$. We extend the definition to complex pomsets simply by removing from the imaginary part the resources in R.

Definition 4.20 Let $x = (s, S) \in \mathbb{C}$ be a complex pomset and let $R \subseteq \mathcal{R}$. We define the hiding operator by $x \backslash R = (s \backslash R, S \setminus R)$.

Again, we have a well-defined operation $\backslash R : \mathbb{C} \to \mathbb{C}$ since for $x = (s, S) \in \mathbb{C}$ we clearly have $\mathrm{res}(x \backslash R) = \mathrm{res}(x) \setminus R$, and $\mathrm{resinf}(s \backslash R) = \mathrm{resinf}(s) \setminus R \subseteq S \setminus R$.

For instance, if $\alpha, \beta, \gamma \in \mathcal{R}$ are resources and we let

$$x = \left(\begin{tikzcd}[column sep=small, row sep=tiny] \{\alpha,\beta\} \arrow[dr] & & \{\alpha,\beta\} \arrow[dr] & & \{\alpha,\beta\} \arrow[dr] & & \{\alpha,\beta\} & \cdots \\ & \{\beta\} \arrow[ur] \arrow[dr] & & \{\beta\} \arrow[ur] \arrow[dr] & & \{\beta\} \arrow[ur] \arrow[dr] & & \\ & & \{\gamma\} \arrow[ur] & & \{\gamma\} \arrow[ur] & & \{\gamma\} & \cdots \end{tikzcd} , \{\alpha,\beta,\gamma\} \right)$$

then we get

$$x \backslash \beta = \left(\begin{tikzcd} \{\alpha\} \arrow[r] \arrow[dr] & \{\alpha\} \arrow[r] \arrow[dr] & \{\alpha\} \arrow[r] \arrow[dr] & \{\alpha\} & \cdots \\ & \{\gamma\} \arrow[ur] \arrow[r] & \{\gamma\} \arrow[ur] \arrow[r] & \{\gamma\} & \cdots \end{tikzcd} , \{\alpha,\gamma\} \right).$$

Proposition 4.21 *Let $R \subseteq \mathcal{R}$. We have*

(i) $\backslash R : \mathbb{C} \to \mathbb{C}$ *is continuous.*

(ii) *For all* $x \in \mathbb{C}$, *we have* $\mathrm{res}(x\backslash R) = \mathrm{res}(x) \setminus R$.

Proof. (ii) is clear from the definition. For (i), we start by showing that $\backslash R$ is monotone. Let $x = (s, S) \sqsubseteq (t, T) = y$ and let $t = (V, \leq, \lambda)$, $s = (U, \leq, \lambda)$ with $U = {\downarrow_{\leq}} U \subseteq V$. Now, let $t\backslash R = (V', \leq', \lambda')$. It is easy to check that $U' = \{p \in U \mid \lambda(p) \setminus R \neq \emptyset\} = {\downarrow_{\leq'}} U' \subseteq V'$ is a downward closed subset of $(V', \leq')$. Therefore, $s\backslash R = (U', \leq', \lambda') \leq t\backslash R$. Moreover, we have $s^{-1}t = (V \setminus U, \leq, \lambda)$ and $(s\backslash R)^{-1}(t\backslash R) = (V' \setminus U', \leq', \lambda')$. It is easy to see that $V' \setminus U' = \{p \in V \setminus U \mid \lambda(p) \setminus R \neq \emptyset\}$, and we deduce that $(s^{-1}t)\backslash R = (s\backslash R)^{-1}(t\backslash R)$. From this, we obtain $\mathrm{res}((s\backslash R)^{-1}(t\backslash R)) = \mathrm{res}((s^{-1}t)\backslash R) = \mathrm{res}(s^{-1}t) \setminus R \subseteq S \setminus R$. Since we also have $T \setminus R \subseteq S \setminus R$ we have shown that $x\backslash R \sqsubseteq y\backslash R$.

Now, we show that $\backslash R$ is continuous. Let $X \subseteq \mathbb{C}$ be directed. Since $\backslash R$ is monotone, we know that $X\backslash R$ is directed and $\sqcup(X\backslash R) \sqsubseteq (\sqcup X)\backslash R$. Conversely, let $\sqcup X = (t, T)$ with $t = (V, \leq, \lambda)$ and $(\sqcup X)\backslash R = (t\backslash R, T \setminus R)$ with $t\backslash R = (V', \leq', \lambda')$. Let $y' = (s', S') \sqsubseteq (\sqcup X)\backslash R$ be a compact complex pomset. We have $s' = (U', \leq', \lambda')$ for some finite set $U' = {\downarrow_{\leq'}} U' \subseteq V'$. Since $U' \subseteq V$ is finite and the lower set in $(V, \leq)$ of any vertex is finite, we deduce that the set $U = {\downarrow_{\leq}} U' \subseteq V$ is also finite. Let $s = (U, \leq, \lambda)$ and $y = (s, S)$ with $S = \mathrm{res}(s^{-1}t) \cup T$. Clearly, $y \in \mathbb{C}_f$ is a finite complex pomset with $y \sqsubseteq \sqcup X = (t, T)$. Hence, we find $x \in X$ such that $y \sqsubseteq x$. Since $\backslash R$ is monotone, it follows $y\backslash R \sqsubseteq x\backslash R \sqsubseteq \sqcup(X\backslash R)$. We claim that $y' \sqsubseteq y\backslash R$. Clearly, we have $s\backslash R = s'$. Now, we have seen above that $\mathrm{res}((s\backslash R)^{-1}(t\backslash R)) = \mathrm{res}(s^{-1}t) \setminus R$. Therefore, $S' \supseteq \mathrm{res}(s'^{-1}(t\backslash R)) \cup T \setminus R = (\mathrm{res}(s^{-1}t) \cup T) \setminus R = S \setminus R$, which proves the claim.

We have shown that for all finite complex pomsets $y' \sqsubseteq (\sqcup X)\backslash R$, we have $y' \sqsubseteq \sqcup(X\backslash R)$. This concludes the proof since $\mathbb{C}$ is algebraic and the compact complex pomsets are the finite ones by Proposition 2.16. $\square$

The interpretation of the hiding operator on $\mathcal{D}$ is then

$$\backslash R : \mathcal{D} \to \mathcal{D} \quad \text{defined by} \quad (f\backslash R)(\sigma) = f(\sigma)\backslash R.$$

By Proposition 4.3, we know that $f \setminus R$ is continuous and by Proposition 4.21(ii), we deduce that it satisfies the condition of Definition 4.1. Indeed, for all $\sigma \in \mathbb{C}^{\mathcal{V}}$ and $z \in \mathcal{V}$ we have

$$\begin{aligned}
\mathrm{res}((f\backslash R)(\sigma[z \mapsto (1, \emptyset)])) &= \mathrm{res}(f(\sigma[z \mapsto (1, \emptyset)])) \setminus R \\
&\subseteq \mathrm{res}(f(\sigma)) \setminus R = \mathrm{res}((f\backslash R)(\sigma)) \\
&\subseteq (\mathrm{res}(f(\sigma[z \mapsto (1, \emptyset)])) \setminus R) \cup (\mathrm{res}(\sigma(z)) \setminus R) \\
&\subseteq \mathrm{res}((f\backslash R)(\sigma[z \mapsto (1, \emptyset)])) \cup \mathrm{res}(\sigma(z)).
\end{aligned}$$

We conclude this section with a lemma that will be useful in relating the operational and denotational semantics.

Lemma 4.22 *Let* $x \in \mathbb{C}$, $R \subseteq \mathcal{R}$, *and* $a \in \Sigma$. *Then,*

(i) *if* $a \le x\backslash R$, *then there are some* $r, s \in \mathbb{R}_f$ *and* $b \in \Sigma$ *with* $r \le s \le x$, $r^{-1}s = b$, $r\backslash R = 1$ *and* $b\backslash R = a$.

(ii) $a \le x$ *implies* $a\backslash R \le x\backslash R$ *and* $(a^{-1}x)\backslash R = (a\backslash R)^{-1}(x\backslash R)$.

Proof. We let $x = (t, T) \in \mathbb{C}$ with $t = (V, \le, \lambda)$ and $t\backslash R = (V', \le', \lambda')$.

(i) If $a \le x\backslash R$, then $a \le t\backslash R$. Hence there exists a minimal vertex $p \in V'$ with $\lambda'(p) = a$. Let $s \in \mathbb{R}_f$ be the finite prefix of t defined by the downward closed subset of vertices $\downarrow p = \{q \in V \mid q \le p\}$. The subset $U = \{q \in V \mid q < p\} = \downarrow p \setminus \{p\}$ is also downward closed and defines a finite prefix r of t. Since p is minimal in V' then we have $\lambda(q) \subseteq R$ for all $q \in U$, which means $r\backslash R = 1$. Now, the pomset $r^{-1}s$ is reduced to the single vertex p which is labelled in t by some $b = \lambda(p)$. Finally, we have $\lambda'(p) = b \setminus R = a$.

(ii) We have shown in the proof of Proposition 4.21 that for all $s, t \in \mathbb{R}$ with $s \le t$, we have $s\backslash R \le t\backslash R$ and $(s^{-1}t)\backslash R = (s\backslash R)^{-1}(t\backslash R)$. The generalization to $s \in \mathbb{R}$ and $x = (t, T) \in \mathbb{C}$ with $s \le x$ is immediate. □

4.5 Recursion

In order to have a compositional semantics for recursion, for each variable $x \in \mathcal{V}$, we need to define a continuous map $\mathrm{rec}\, x : \mathcal{D} \to \mathcal{D}$, and then we can set $[\![\mathrm{rec}\, x.p]\!] = \mathrm{rec}\, x.[\![p]\!]$ for all processes $p \in \mathcal{L}$. We use a fixed point of a continuous selfmap from $\mathbb{C}$ to $\mathbb{C}$, but contrary to the classical approach, we cannot use the least fixed point semantics since our domain $\mathbb{C}$ does not have a least element.

Example 4.23 We begin with an example. Consider the process $\mathrm{rec}\, x.p$ with $p = a \circ x$. From the previous sections we know that the semantics of p is the continuous map $[\![p]\!] : \mathbb{C}^{\mathcal{V}} \to \mathbb{C}$ defined by $[\![p]\!](\sigma) = (a, \emptyset) \circ \sigma(x)$. The semantics of $\mathrm{rec}\, x.p$ will be a fixed point of the continuous selfmap from $\mathbb{C}$ to $\mathbb{C}$ defined by $y \mapsto \rho_x ; [\![p]\!](\sigma[x \mapsto y])$, where ρ_x stands for the finite complex pomset $(\{\rho_x\}, \emptyset)$ in order to lighten the notation. This fixed point is obtained as follows. Let $x_0 = (1, \mathrm{res}(\mathrm{rec}\, x.p)) = (1, a \cup \{\rho_x\})$ and $x_{n+1} = \rho_x \ ; [\![p]\!](\sigma[x \mapsto x_n])$. For instance, we have

$$x_3 = \left(\begin{tikzcd}[column sep=small, row sep=small] \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[dr] & \\ & a \arrow[r] & a \arrow[r] & a \end{tikzcd} , \ a \cup \{\rho_x\} \right).$$

The sequence (x_n) is increasing and its least upper bound is

$$x_\omega = \left(\begin{tikzcd}[column sep=small, row sep=small] \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[dr] & \cdots & \\ & a \arrow[r] & a \arrow[r] & a \arrow[r] & a & \cdots \end{tikzcd} , \ a \cup \{\rho_x\} \right)$$

which is the semantics of the process $\mathrm{rec}\, x.p$. Since $\mathrm{rec}\, x.p$ has no free variable, its semantics must be a constant map, assigning, in this case, x_ω to any environment $\sigma \in \mathbb{C}^{\mathcal{V}}$.

Note that the resource set claimed by $\mathrm{rec}\, x.p$ (Section 3.1) is exactly the resource set used by its semantics: $\mathrm{res}([\![\mathrm{rec}\, x.p]\!]) = a \cup \{\rho_x\} = \mathrm{res}(\mathrm{rec}\, x.p)$. It is very important to start the iteration for the fixed point by claiming as few resources as possible, that is $\mathrm{res}(\mathrm{rec}\, x.p)$. If we claimed more resources to start the iteration, these resources would stay in the imaginary part of the fixed point and our denotational semantics would not behave correctly when composing this process with other processes. For instance, if b is an action with $b \cap a = \emptyset$, then the semantics of $(\mathrm{rec}\, x.p) \circ b$ should have b as a minimal element. This would not be the case if some resources from b were unnecessarily claimed by the recursion.

Since we are not using the classical least fixed point semantics, we have to explain in some detail how our semantics works. Recall that our semantic domain $\mathcal{D}$ is the set of continuous maps $f \in [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]$ satisfying for all $\sigma \in \mathbb{C}^{\mathcal{V}}$ and $z \in \mathcal{V}$

$$\text{(1)} \quad \mathrm{res}(f(\sigma[z \mapsto (1,\emptyset)])) \subseteq \mathrm{res}(f(\sigma)) \subseteq \mathrm{res}(f(\sigma[z \mapsto (1,\emptyset)])) \cup \mathrm{res}(\sigma(z)).$$

We fix a variable $x \in \mathcal{V}$ and recall that ρ_x stands for the finite complex pomset $(\{\rho_x\}, \emptyset)$. We define the mapping $\Phi_x : \mathcal{D} \times \mathbb{C}^{\mathcal{V}} \times \mathbb{C} \to \mathbb{C}$ by

$$\Phi_x(f, \sigma, y) = \rho_x \,;\, f(\sigma[x \mapsto y]).$$

Lemma 4.24 *The map Φ_x is continuous.*

Proof. First, the mapping $\mathbb{C}^{\mathcal{V}} \times \mathbb{C} \to \mathbb{C}^{\mathcal{V}}$ defined by $(\sigma, y) \mapsto \sigma[x \mapsto y]$ which overrides the value of σ at x is clearly continuous. Since applying a function is also a continuous operation in both the function and its argument, we conclude that the mapping $(f, \sigma, y) \mapsto f(\sigma[x \mapsto y])$ is continuous, being the composition of continuous maps. We conclude using Proposition 4.6. $\square$

For $f \in \mathcal{D}$ and $\sigma \in \mathbb{C}^{\mathcal{V}}$, we define $(\mathrm{rec}\, x.f)(\sigma)$ to be a fixed point of the continuous selfmap $\Phi_x(f, \sigma, -) : \mathbb{C} \to \mathbb{C}$ defined by $y \mapsto \Phi_x(f, \sigma, y)$. As explained above, we do not use the least fixed point. Instead, we start the iteration yielding the fixed point from a complex pomset $x_0(f, \sigma)$ which depends on f and σ.

Definition 4.25 For $f \in \mathcal{D}$ and $\sigma \in \mathbb{C}^{\mathcal{V}}$, we define

$$\begin{aligned} R_x(f,\sigma) &= \mathrm{res}(\Phi_x(f,\sigma,(1,\emptyset))) = \{\rho_x\} \cup \mathrm{res}(f(\sigma[x \mapsto (1,\emptyset)])) \\ x_0(f,\sigma) &= (1, R_x(f,\sigma)) \\ x_{n+1}(f,\sigma) &= \Phi_x(f,\sigma,x_n(f,\sigma)) = \rho_x \,;\, f(\sigma[x \mapsto x_n(f,\sigma)])). \end{aligned}$$

The next lemma establishes several important properties.

Lemma 4.26 *Let $f \in \mathcal{D}$ and $\sigma \in \mathbb{C}^{\mathcal{V}}$.*

(i) *For all $n \in \mathbb{N}$,* $\mathrm{res}(x_n(f,\sigma)) = R_x(f,\sigma)$.

(ii) *The sequence $(x_n(f,\sigma))_{n \geq 0}$ is increasing.*

(iii) *For all* $n \in \mathbb{N}$*, the map* $(f, \sigma) \mapsto x_n(f, \sigma)$ *is continuous.*

(iv) *For all* $n \in \mathbb{N}$*, the map* $\sigma \mapsto x_n(f, \sigma)$ *satisfies Equation 1.*

Proof. (i) We proceed by induction on n. This is where the property (1) of $f \in \mathcal{D}$ is used. Clearly, we have $\operatorname{res}(x_0(f, \sigma)) = R_x(f, \sigma)$. Now assume the result holds for n. We have $\operatorname{res}(x_{n+1}(f, \sigma)) = \{\rho_x\} \cup \operatorname{res}(f(\sigma[x \mapsto x_n(f, \sigma)]))$. Applying property (1) to f, $\sigma[x \mapsto x_n(f, \sigma)]$ and x, we obtain

$$R_x(f, \sigma) \subseteq \operatorname{res}(x_{n+1}(f, \sigma)) \subseteq R_x(f, \sigma) \cup \operatorname{res}(x_n(f, \sigma))$$

and, using the induction hypothesis, we get $\operatorname{res}(x_{n+1}(f, \sigma)) = R_x(f, \sigma)$.

(ii) We deduce immediately using (i) that $x_0(f, \sigma) \sqsubseteq x_1(f, \sigma)$ and the result follows since the mapping $\Phi_x(f, \sigma, -)$ is monotone.

(iii) is proved by induction on n.

- We obtain the continuity of $(f, \sigma) \mapsto x_0(f, \sigma)$ since it is the composition of three continuous maps: $(f, \sigma) \mapsto \Phi_x(f, \sigma, (1, \emptyset))$, $\operatorname{res} : \mathbb{C} \to (\mathcal{P}_f(\mathcal{R}), \supseteq)$ and the embedding of $(\mathcal{P}_f(\mathcal{R}), \supseteq)$ into $(\mathbb{C}, \sqsubseteq)$ defined by $R \mapsto (1, R)$.
- Assume now that x_n is continuous for some n. Then, x_{n+1} is also continuous, being the composition of continuous maps: $x_{n+1}(f, \sigma) = \Phi_x(f, \sigma, x_n(f, \sigma))$.

(iv) Let $\sigma \in \mathbb{C}^{\mathcal{V}}$, $z \in \mathcal{V}$ and $\sigma' = \sigma[x \mapsto (1, \emptyset)]$. If $z = x$, then we deduce from Definition 4.25 that $x_n(f, \sigma) = x_n(f, \sigma[z \mapsto (1, \emptyset)])$ and property (1) follows trivially.

If $z \neq x$, then using (i) and Property (1) for f, σ' and z, we deduce

$$\begin{aligned}
\operatorname{res}(x_n(f, \sigma[z \mapsto (1, \emptyset)])) &= \{\rho_x\} \cup \operatorname{res}(f(\sigma'[z \mapsto (1, \emptyset)])) \\
&\subseteq \{\rho_x\} \cup \operatorname{res}(f(\sigma')) = \operatorname{res}(x_n(f, \sigma)) \\
&\subseteq \{\rho_x\} \cup \operatorname{res}(f(\sigma'[z \mapsto (1, \emptyset)])) \cup \operatorname{res}(\sigma'(z)) \\
&= \operatorname{res}(x_n(f, \sigma[z \mapsto (1, \emptyset)])) \cup \operatorname{res}(\sigma(z)).
\end{aligned}$$ □

We now are ready to define the interpretation of recursion on $\mathcal{D}$:

$$\operatorname{rec} x : \mathcal{D} \to \mathcal{D} \quad \text{defined by} \quad (\operatorname{rec} x.f)(\sigma) = \bigsqcup_{n \geq 0} x_n(f, \sigma).$$

Proposition 4.27

(i) *For* $f \in \mathcal{D}$ *and* $\sigma \in \mathbb{C}^{\mathcal{V}}$,

$$\operatorname{res}((\operatorname{rec} x.f)(\sigma)) = R_x(f, \sigma) = \{\rho_x\} \cup \operatorname{res}(f(\sigma[x \mapsto (1, \emptyset)])).$$

(ii) *For* $f \in \mathcal{D}$ *and* $\sigma \in \mathbb{C}^{\mathcal{V}}$,

$$(\operatorname{rec} x.f)(\sigma) = \rho_x \,;\, f(\sigma[x \mapsto (\operatorname{rec} x.f)(\sigma)])$$

is the least fixed point of $\Phi_x(f, \sigma, -)$ above $x_0(f, \sigma)$.

(iii) *The mapping* $\operatorname{rec} x : \mathcal{D} \to \mathcal{D}$ *is well defined and continuous.*

Proof. (i) Since res : $\mathbb{C} \to (\mathcal{P}_f(\mathcal{R}), \supseteq)$ is continuous (Corollary 2.13), using Lemma 4.26 (i) we get $\mathrm{res}((\mathrm{rec}\, x.f)(\sigma)) = \bigcap_{n\geq 0} \mathrm{res}(x_n(f,\sigma)) = R_x(f,\sigma)$.

The proof of (ii) follows directly from our definitions using the continuity of f and of the operator ";":

$$\begin{aligned}(\mathrm{rec}\, x.f)(\sigma) = x_\omega(f,\sigma) &= \bigsqcup_{n\geq 0} x_n(f,\sigma) = \bigsqcup_{n\geq 0} \rho_x \,;\, f(\sigma[x \mapsto x_n(f,\sigma)]) \\ &= \rho_x \,;\, f(\sigma[x \mapsto x_\omega(f,\sigma)]) = \rho_x \,;\, f(\sigma[x \mapsto (\mathrm{rec}\, x.f)(\sigma)]).\end{aligned}$$

(iii) By Lemma 4.26, $(x_n(f,-))_{n\geq 0}$ is an increasing sequence of maps in $\mathcal{D}$. Using Proposition 4.2, we deduce that $\mathrm{rec}\, x.f = \bigsqcup_{n\geq 0} x_n(f,-)$ is also in $\mathcal{D}$ and therefore, the mapping $\mathrm{rec}\, x : \mathcal{D} \to \mathcal{D}$ is well defined.

Now, the map $(f,\sigma) \mapsto (\mathrm{rec}\, x.f)(\sigma) = \bigsqcup_{n\geq 0} x_n(f,\sigma)$ from $\mathcal{D} \times \mathbb{C}^{\mathcal{V}}$ to $\mathbb{C}$ is continuous as a directed supremum of continuous maps, a standard fact from domain theory (cf. [1], Theorem 2.1.18). The continuity of $\mathrm{rec}\, x : \mathcal{D} \to \mathcal{D}$ follows since

$$[\mathcal{D} \times \mathbb{C}^{\mathcal{V}} \to \mathbb{C}] \simeq [\mathcal{D} \to [\mathbb{C}^{\mathcal{V}} \to \mathbb{C}]].$$

□

4.6 Summary

We conclude the discussion of the denotational semantics of our language by giving a summary of the semantics for the processes in $\mathcal{L}$. We also prove two results concerning the resource map and substitution.

We have defined our denotational semantics as a compositional mapping $[\![-]\!] : \mathcal{L} \to \mathcal{D}$; the work in this section has validated that such a mapping exists, since $\mathcal{L}$ is the initial Ω-algebra, and we have given a continuous interpretation in $\mathcal{D}$ for each of the operators $\mathrm{op} \in \Omega$ in the signature of our language. To summarize, the semantics of a process $p \in \mathcal{L}$ is the continuous map $[\![p]\!]$ defined inductively by:

$$\begin{aligned}
[\![\mathrm{SKIP}]\!](\sigma) &= (1, \emptyset) \\
[\![a]\!](\sigma) &= (a, \emptyset) \\
[\![x]\!](\sigma) &= \sigma(x) \\
[\![p \,;\, q]\!](\sigma) &= [\![p]\!](\sigma) \,;\, [\![q]\!](\sigma) \\
[\![p \circ q]\!](\sigma) &= [\![p]\!](\sigma) \circ [\![q]\!](\sigma) \\
[\![p \parallel_C q]\!](\sigma) &= [\![p]\!](\sigma) \parallel_C [\![q]\!](\sigma) \\
[\![p \backslash R]\!](\sigma) &= ([\![p]\!](\sigma)) \backslash R \\
[\![\mathrm{rec}\, x.p]\!](\sigma) &= (\mathrm{rec}\, x.[\![p]\!])(\sigma).
\end{aligned}$$

We immediately get that the denotational semantics of a process p only depends on the free variables of p.

Lemma 4.28 *Let $p \in \mathcal{L}$ be a process and $\sigma, \sigma' \in \mathbb{C}^V$ be environments satisfying $\sigma(x) = \sigma'(x)$ for all $x \in \mathrm{Free}(p)$. Then $[\![p]\!](\sigma) = [\![p]\!](\sigma')$.*

Proof. This is an easy structural induction argument based on p. The only non-trivial case is when $p = \mathrm{rec}\, x.p_1$. In this case, we have $\mathrm{Free}(p_1) \subseteq \mathrm{Free}(p) \cup$

$\{x\}$. Using the induction hypothesis on p_1, we get easily by induction on n that $x_n([\![p_1]\!], \sigma) = x_n([\![p_1]\!], \sigma')$ for all $n \geq 0$. Therefore, $[\![p]\!](\sigma) = [\![p]\!](\sigma')$. $\square$

We deduce from Lemma 4.28 that the denotational semantics of a closed term $p \in \mathcal{L}_c$ is a constant map and we simply write $[\![p]\!]$ for its value, hence in this case $[\![p]\!] \in \mathbb{C}$. We now show how the denotational semantics behaves with respect to process substitution.

Lemma 4.29 *Let $p \in \mathcal{L}$, $q \in \mathcal{L}_c$ and $\sigma \in \mathbb{C}^{\mathcal{V}}$. Then,*

$$[\![p[q/x]]\!](\sigma) = [\![p]\!](\sigma[x \mapsto [\![q]\!]]).$$

Proof. Let $\sigma' = \sigma[x \mapsto [\![q]\!]]$. We proceed by structural induction on p. The cases $p = \mathrm{SKIP}$, $p = a$, $p = x$ or $p = y \neq x$ are trivial.

- If $p = p_1 \,\mathrm{op}\, p_2$ with op being either ; or $\circ$ or $\underset{C}{\|}$. Then $p[q/x] = p_1[q/x]$ op $p_2[q/x]$ and we deduce

$$[\![p[q/x]]\!](\sigma) = [\![p_1[q/x]]\!](\sigma) \,\mathrm{op}\, [\![p_2[q/x]]\!](\sigma) = [\![p_1]\!](\sigma') \,\mathrm{op}\, [\![p_2]\!](\sigma') = [\![p]\!](\sigma').$$

- If $p = p_1 \backslash R$, then $p[q/x] = p_1[q/x] \backslash R$ and we deduce

$$[\![p[q/x]]\!](\sigma) = [\![p_1[q/x]]\!](\sigma) \backslash R = [\![p_1]\!](\sigma') \backslash R = [\![p]\!](\sigma').$$

- If $p = \mathrm{rec}\, x.p_1$, then $p[q/x] = p$ and

$$[\![p]\!](\sigma) = (\mathrm{rec}\, x.[\![p_1]\!])(\sigma) = (\mathrm{rec}\, x.[\![p_1]\!])(\sigma') = [\![p]\!](\sigma').$$

- Finally, assume $p = \mathrm{rec}\, y.p_1$ with $y \neq x$. Since q is closed, we have $p[q/x] = \mathrm{rec}\, x.(p_1[q/x])$. We first show by induction on n that $y_n([\![p_1[q/x]]\!], \sigma) = y_n([\![p_1]\!], \sigma')$. We start with $n = 0$. We have,

$$\begin{aligned} y_0([\![p_1[q/x]]\!], \sigma) &= (1, \{\rho_y\} \cup \mathrm{res}([\![p_1[q/x]]\!](\sigma[y \mapsto (1, \emptyset)]))) \\ &= (1, \{\rho_y\} \cup \mathrm{res}([\![p_1]\!](\sigma'[y \mapsto (1, \emptyset)]))) \\ &= y_0([\![p_1]\!], \sigma'). \end{aligned}$$

  Then, we obtain by induction

$$\begin{aligned} y_{n+1}([\![p_1[q/x]]\!], \sigma) &= \rho_y \,;\, [\![p_1[q/x]]\!](\sigma[y \mapsto y_n([\![p_1[q/x]]\!], \sigma)]) \\ &= \rho_y \,;\, [\![p_1]\!](\sigma'[y \mapsto y_n([\![p_1]\!], \sigma')]) \\ &= y_{n+1}([\![p_1]\!], \sigma'). \end{aligned}$$

  Finally, we deduce

$$[\![p[q/x]]\!](\sigma) = (\mathrm{rec}\, y.[\![p_1[q/x]]\!])(\sigma) = (\mathrm{rec}\, y.[\![p_1]\!])(\sigma') = [\![p]\!](\sigma').$$

$\square$

The resource map defined for the terms of our language in Section 3.1 coincides with the resources used by the denotational semantics.

Lemma 4.30 *If* $p \in \mathcal{L}_c$ *is a closed process, then* $\mathrm{res}(p) = \mathrm{res}([\![p]\!])$.

Proof. We proceed by induction on the structural depth of p. The cases $p = \mathrm{SKIP}$ and $p = a$ are trivial, and $p = x$ is not possible since p is closed. The cases $p = p_1 \,\mathrm{op}\, p_2$ and $p = p_1 \backslash R$ follow easily from Definition 3.1 and Propositions 4.6, 4.12, 4.17 and 4.21. It remains to deal with $p = \mathrm{rec}\, x.p_1$. Note that p_1 has at most one free variable which is x and that the closed process $p_1[\mathrm{SKIP}/x]$ has the same structural depth as p_1 and also claims the same resources: $\mathrm{res}(p_1) = \mathrm{res}(p_1[\mathrm{SKIP}/x])$. Hence we can apply the induction hypothesis to $p_1[\mathrm{SKIP}/x]$. We have,

$$\begin{aligned}
\mathrm{res}([\![p]\!]) &= \{\rho_x\} \cup \mathrm{res}([\![p_1]\!](\sigma[x \mapsto (1, \emptyset)])) && \text{by Proposition 4.27}\\
&= \{\rho_x\} \cup \mathrm{res}([\![p_1[\mathrm{SKIP}/x]]\!]) && \text{by Lemma 4.29}\\
&= \{\rho_x\} \cup \mathrm{res}(p_1[\mathrm{SKIP}/x]) && \text{by induction}\\
&= \{\rho_x\} \cup \mathrm{res}(p_1) = \mathrm{res}(p).
\end{aligned}$$

□

4.7 Examples

We now recall the examples from Section 3.3, and give the denotational meaning of each process.

(i) If $p = \mathrm{rec}\, x.(a\,;x)$, then for any $\sigma \in C^{\mathcal{V}}$, we first calculate

$$R_x([\![a\,;x]\!], \sigma) = \{\rho_x\} \cup \mathrm{res}([\![a\,;x]\!](\sigma[x \mapsto (1, \emptyset)])) = \{\rho_x\} \cup a.$$

Then,

$$\begin{aligned}
x_0 &= x_0([\![a\,;x]\!], \sigma) = (1, \{\rho_x\} \cup a),\\
x_1 &= x_1([\![a\,;x]\!], \sigma) = \rho_x\,; [\![a\,;x]\!](\sigma[x \mapsto x_0])\\
&\qquad = \rho_x\,; ((a, \emptyset)\,; (1, \{\rho_x\} \cup a))\\
&\qquad = (\rho_x \to a, \{\rho_x\} \cup a),
\end{aligned}$$

and, in general

$$\begin{aligned}
x_{n+1} = x_{n+1}([\![a\,;x]\!], \sigma) &= \rho_x\,; [\![a\,;x]\!](\sigma[x \mapsto x_n])\\
&= \rho_x\,; ((a, \emptyset)\,; x_n)\\
&= (\rho_x \to a \to \cdots \to \rho_x \to a, \{\rho_x\} \cup a).
\end{aligned}$$

with $n+1$ occurrences of $\rho_x \to a$. Finally, by taking the least upper bound of the sequence $(x_n)_{n \geq 0}$, we get

$$[\![\mathrm{rec}\, x.(a\,;x)]\!] = (\rho_x \to a \to \rho_x \to a \to \rho_x \to a \cdots, \{\rho_x\} \cup a).$$

Note that this process cannot diverge, that is, it cannot continuously unwind the recursion without executing a visible action. Actually, the process has to execute a visible action after each unwinding since the recursion is guarded. Once we have checked that the process cannot diverge, we can safely abstract away from observing the unwindings which is done by hiding ρ_x. Applying the definition for hiding 4.20, we see that,

$$[\![p\backslash\rho_x]\!] = [\![p]\!]\backslash\rho_x = (a \longrightarrow a \longrightarrow a \longrightarrow a \quad \cdots \ , \ a).$$

(ii) On the other hand, if $q = \mathrm{rec}\, x.(a \circ x)$, then for any $\sigma \in C^{\mathcal{V}}$, we have

$$R_x([\![a \circ x]\!], \sigma) = \{\rho_x\} \cup \mathrm{res}([\![a \circ x]\!](\sigma[x \mapsto (1, \emptyset)])) = \{\rho_x\} \cup a.$$

Then,

$$\begin{aligned}
x_0 &= x_0([\![a \circ x]\!], \sigma) = (1, \{\rho_x\} \cup a),\\
x_1 &= x_1([\![a \circ x]\!], \sigma) = \rho_x \, ; [\![a \circ x]\!](\sigma[x \mapsto x_0])\\
&\qquad = \rho_x \, ; ((a, \emptyset) \circ (1, \{\rho_x\} \cup a))\\
&\qquad = (\rho_x \to a, \{\rho_x\} \cup a),\\
x_2 &= x_2([\![a \circ x]\!], \sigma) = \rho_x \, ; [\![a \circ x]\!](\sigma[x \mapsto x_1])\\
&\qquad = \rho_x \, ; ((a, \emptyset) \circ (\rho_x \to a, \{\rho_x\} \cup a))\\
&\qquad = \left(\begin{tikzcd}[row sep=small, column sep=small] \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[dr] & \\ & a \arrow[r] & a \end{tikzcd} , \ \{\rho_x\} \cup a \right),
\end{aligned}$$

$$\begin{aligned}
x_{n+1} &= x_{n+1}([\![a \circ x]\!], \sigma) = \rho_x \, ; [\![a \circ x]\!](\sigma[x \mapsto x_n])\\
&\qquad = \rho_x \, ; ((a, \emptyset) \circ x_n)\\
&\qquad = \left(\begin{tikzcd}[row sep=small, column sep=small] \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[dr] & \cdots & \rho_x \arrow[dr] & \\ & a \arrow[r] & a & \cdots & a \end{tikzcd} , \ \{\rho_x\} \cup a \right).
\end{aligned}$$

Finally, by taking the least upper bound of the sequence $(x_n)_{n \geq 0}$, we get

$$[\![\mathrm{rec}\, x.(a \circ x)]\!] = \left(\begin{tikzcd}[row sep=small, column sep=small] \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[r] \arrow[dr] & \rho_x \arrow[dr] & \cdots & \\ & a \arrow[r] & a \arrow[r] & a \arrow[r] & a & \cdots \end{tikzcd} , \ \{\rho_x\} \cup a \right).$$

This example illustrates how the resource ρ_x reveals the potential divergence of q. This is the case because $\{\rho_x\} \cap a = \emptyset$ and weak sequential composition allows the unwinding of recursion to continue without the action a being executed. In fact, contrary to the previous example, this recursion is unguarded, causing the possible divergence. Note that it is not safe to hide a variable resource before checking the process for divergence. Indeed, if we hide ρ_x we cannot observe the unwindings of recursion and therefore, we cannot observe divergence anymore. Again, applying the definition for hiding 4.20, we see that,

$$[\![q\backslash\rho_x]\!] = [\![q]\!]\backslash\rho_x = (a \longrightarrow a \longrightarrow a \longrightarrow a \quad \cdots \ , \ a) = [\![p\backslash\rho_x]\!].$$

(iii) Our final example is the 2-place buffer introduced in Section 3.3

$$\mathcal{B}^2(a, b) = (\mathcal{B}^1(a, c) \underset{c}{\|} \mathcal{B}^1(c, b))\backslash c$$

with $\mathcal{B}^1(a,c) = (\mathrm{rec}\, x.(a\,;c\,;x))\backslash\rho_x$ and $\mathcal{B}^1(c,b) = (\mathrm{rec}\, x.(c\,;b\,;x))\backslash\rho_x$. As with the previous examples, we get the following semantics

$$\begin{aligned}
[\![\mathcal{B}^1(a,c)]\!] &= (\, a \to c \to a \to c \to a \to c \quad \cdots \quad ,\ a \cup c),\\
[\![\mathcal{B}^1(c,b)]\!] &= (\, c \to b \to c \to b \to c \to b \quad \cdots \quad ,\ b \cup c).
\end{aligned}$$

Now, the semantics of parallel compositions gives for $\mathcal{B}^1(a,c) \underset{c}{\|} \mathcal{B}^1(c,b)$

$$[\![\mathcal{B}^1(a,c) \underset{c}{\|} \mathcal{B}^1(c,b)]\!] = \left(\begin{tikzcd}[row sep=small, column sep=small]
a \arrow[dr] & & a \arrow[dr] & & a \arrow[dr] & & \cdots \\
& c \arrow[ur] \arrow[dr] & & c \arrow[ur] \arrow[dr] & & c & \\
& & b \arrow[ur] & & b \arrow[ur] & & \cdots
\end{tikzcd} \quad , \ a \cup b \cup c \right).$$

Finally, hiding c from this complex pomset gives the denotational semantics of $\mathcal{B}^2(a,b)$:

$$[\![\mathcal{B}^2(a,b)]\!] = \left(\begin{tikzcd}[row sep=small, column sep=small]
a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a & \cdots \\
& b \arrow[r] \arrow[ur] & b \arrow[r] \arrow[ur] & b \arrow[r] \arrow[ur] & b & \cdots
\end{tikzcd} \quad , \ a \cup b \right).$$

We can see that this fulfills the property of being a 2-place buffer. The buffer has to receive a's before sending the corresponding b's. The buffer may also receive two a's (but not three) before sending the first b. The semantics of a 3-place buffer would be

$$[\![\mathcal{B}^3(a,b)]\!] = \left(\begin{tikzcd}[row sep=small, column sep=small]
a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a \arrow[r] \arrow[dr] & a & \cdots \\
& b \arrow[r] \arrow[urr] & b \arrow[r] \arrow[urr] & b \arrow[r] & b & \cdots
\end{tikzcd} \ , a \cup b \right).$$

5 Relating Semantics: The Congruence Theorem

In this section we complete the picture by showing that the operational behavior of a process defined in Section 3 is essentially the same as the Ω-algebra map we defined for the denotational model in the last section. We first show that the denotational semantics is adequate with respect to the behavior function which associates to a process p the set of strings of actions that p can execute. Then, we prove a congruence theorem for our semantics, and we conclude by showing that the denotational semantics is fully abstract with respect to the behavior function defined above.

Our proof relies on two crucial propositions: the first states that any action the process p can execute is a prefix of its denotational semantics $[\![p]\!]$, and the second says that any action that is a prefix of $[\![p]\!]$ is in fact an action that p can execute. This second proposition only holds for processes we call nice.

We begin by presenting these two crucial results in the next section, the first of which we can prove with the results already in hand. The second result cannot be proved using standard structural induction techniques, and so we introduce a special rank function that we use to prove it. Finally comes the adequacy result, the congruence theorem, and the full abstraction results mentioned above.

5.1 Two Crucial Results

We now prove two propositions that together show the intimate relationship between our operational semantics and our denotational semantics. We begin with a definition.

Definition 5.1 If $u \in \Sigma^*$ and $s \in \mathbb{R}$, then we define $u \lhd s$ and $u^{-1}s$ by induction on the length of the string u:

- $1 \lhd s$ and $1^{-1}s = s$.
- If $u \in \Sigma^*$ and $a \in \Sigma$, then $ua \lhd s$ if $u \lhd s$ and $a \leq u^{-1}s$; in this case, $(ua)^{-1}s = a^{-1}(u^{-1}s)$.

We can then extend this definition to $u \in \Sigma^*$ and $x \in \mathbb{C}$ by $u \lhd x$ if and only if $u \lhd \mathrm{Re}(x)$, and in this case, $u^{-1}x = (u^{-1}\mathrm{Re}(x), \mathrm{Im}(x))$.

Remark 5.2

(i) We note that for $s \in \mathbb{R}$ and $u \in \Sigma^*$, we have $u \lhd s$ if and only if there exists a finite prefix $r \in \mathbb{R}_f$ of s such that $u \in \mathrm{Lin}(r)$, where $\mathrm{Lin}r$ denotes the set of linearizations of r. Moreover in this case, $u^{-1}s$ as defined above is equal to $r^{-1}s$ as defined in Definition 2.3.

(ii) Any word $u \in \Sigma^*$ can be regarded as a real pomset when endowed with the linear order. Note that, if $a \in \Sigma$ and $s \in \mathbb{R}$ then $a \leq s$ if and only if $a \lhd s$. This equivalence does not hold if we replace a by $u \in \Sigma^*$. If $u \leq s$, then $u \lhd s$, but $u \lhd s$ does not imply u is a prefix of s in general. For example,

$$(a \to b) \lhd \begin{pmatrix} a \\ \\ b \end{pmatrix}, \text{ but clearly } (a \to b) \not\leq \begin{pmatrix} a \\ \\ b \end{pmatrix}.$$

(iii) We have extended the operation $r^{-1}s$ given in Definition 2.3 to those pairs $(r, s) \in \Sigma^* \times \mathbb{R}$ with $r \lhd s$. And, if $r \leq s$, then the two definitions coincide. Therefore, we can use the same notation without causing confusion.

(iv) Throughout this section, whenever we write $u^{-1}x = y$, we implicitly mean that $u \lhd x$ also holds and $u^{-1}x$ is the complex pomset defined in Definition 5.1.

Recall that $p \xrightarrow{a} p'$ denotes a transition from our operational semantics. If $u \in \Sigma^*$ then we use $p \overset{u}{\Longrightarrow} p'$ to denote the fact that we can write $u = a_1 \cdots a_n$ with $n \geq 0$, $a_1, \ldots, a_n \in \Sigma \cup \{1\}$ and $p = p_0 \xrightarrow{a_1} p_1 \cdots p_{n-1} \xrightarrow{a_n} p_n = p'$. Note that we always have $p \overset{1}{\Longrightarrow} p$.

Proposition 5.3 *Let $p \in \mathcal{L}_c$ be a closed term.*

(i) *If $p \xrightarrow{a} p'$ with $a \in \Sigma \cup \{1\}$, then $a \lhd [\![p]\!]$ and $a^{-1}[\![p]\!] = [\![p']\!]$.*

(ii) *If $p \overset{u}{\Longrightarrow} p'$ with $u \in \Sigma^*$, then $u \lhd [\![p]\!]$ and $u^{-1}[\![p]\!] = [\![p']\!]$.*

Proof. (ii) follows directly by induction on the length of u using (i) and Definition 5.1. In order to prove (i) we use induction on p.

- $p = \mathrm{SKIP}$ or $p = x$ are not possible.
- If $p = b \in \mathrm{Act}$, then $a = b$ and $p \xrightarrow{a} p' = \mathrm{SKIP}$. Then $[\![p]\!] = (a, \emptyset)$, $[\![\mathrm{SKIP}]\!] = (1, \emptyset)$, and the result is clear.
- If $p = p_1 \,;\, p_2$, then there are two cases:
  · $p_1 \xrightarrow{a} p_1'$. Then $p' = p_1' \,;\, p_2$ and we obtain by induction $a^{-1}[\![p_1]\!] = [\![p_1']\!]$. Then, Lemma 4.7 implies $a^{-1}[\![p]\!] = (a^{-1}[\![p_1]\!]) \,;\, [\![p_2]\!] = [\![p_1']\!] \,;\, [\![p_2]\!] = [\![p']\!]$.
  · $p_2 \xrightarrow{a} p_2'$ and $\mathrm{res}(p_1) = \emptyset$. Then $p' = p_1 \,;\, p_2'$ and we obtain by induction $a^{-1}[\![p_2]\!] = [\![p_2']\!]$. By Lemma 4.30 we deduce that $[\![p_1]\!] = (1, \emptyset)$. Then, Lemma 4.7 implies $a^{-1}[\![p]\!] = a^{-1}[\![p_2]\!] = [\![p_2']\!] = [\![p']\!]$.
- If $p = p_1 \circ p_2$, then there are also two cases:
  · $p_1 \xrightarrow{a} p_1'$. Same as above using Lemma 4.13.
  · $p_2 \xrightarrow{a} p_2'$ and $a \; I \; p_1$. Then $p' = p_1 \circ p_2'$ and we obtain by induction $a^{-1}[\![p_2]\!] = [\![p_2']\!]$. By Lemma 4.30 we deduce that $aI[\![p_1]\!]$. Then, Lemma 4.13 implies $a^{-1}[\![p]\!] = [\![p_1]\!] \circ (a^{-1}[\![p_2]\!]) = [\![p_1]\!] \circ [\![p_2']\!] = [\![p']\!]$.
- If $p = p_1 \underset{C}{\|} p_2$, then there are three cases:
  · $p_1 \xrightarrow{a} p_1'$ and $a \; I \; p_2, C$. Then $p' = p_1' \underset{C}{\|} p_2$ and we obtain by induction $a^{-1}[\![p_1]\!] = [\![p_1']\!]$. By Lemma 4.30 we deduce that $a \; I \; [\![p_2]\!], C$. Then, Lemma 4.18 implies $a^{-1}[\![p]\!] = (a^{-1}[\![p_1]\!]) \underset{C}{\|} [\![p_2]\!] = [\![p_1']\!] \underset{C}{\|} [\![p_2]\!] = [\![p']\!]$.
  · $p_2 \xrightarrow{a} p_2'$ and $a \; I \; p_1, C$ is similar.
  · $p_1 \xrightarrow{a} p_1'$, $p_2 \xrightarrow{a} p_2'$, and $a \in C$. Then, $p' = p_1' \underset{C}{\|} p_2'$ and we obtain by induction $a^{-1}[\![p_1]\!] = [\![p_1']\!]$ and $a^{-1}[\![p_2]\!] = [\![p_2']\!]$. Then, Lemma 4.18 implies $a^{-1}[\![p]\!] = (a^{-1}[\![p_1]\!]) \underset{C}{\|} (a^{-1}[\![p_2]\!]) = [\![p_1']\!] \underset{C}{\|} [\![p_2']\!] = [\![p']\!]$.
- If $p = p_1 \backslash R$, then $p_1 \xrightarrow{b} p_1'$ for some b and p_1' with $b \backslash R = a$ and $p_1' \backslash R = p'$. By induction, $b^{-1}[\![p_1]\!] = [\![p_1']\!]$. There are two cases:
  · If $b = 1$, then $a = 1$ and $a^{-1}[\![p]\!] = [\![p]\!] = [\![p_1]\!] \backslash R = [\![p_1']\!] \backslash R = [\![p']\!]$.
  · If $b \neq 1$, then Lemma 4.22 implies $a^{-1}[\![p]\!] = (b^{-1}[\![p_1]\!]) \backslash R = [\![p_1']\!] \backslash R = [\![p']\!]$.
- If $p = \mathrm{rec}\, x.p_1$, then $p \xrightarrow{a} p'$ implies $a = \rho_x$ and $p' = p_1[p/x]$. Proposition 4.27 and Lemma 4.29 imply $a^{-1}[\![p]\!] = [\![p_1]\!](x \mapsto [\![p]\!]) = [\![p_1[p/x]]\!] = [\![p']\!]$. $\square$

We now turn our attention to the converse of Proposition 5.3. Note first that we did not need to restrict the processes on Proposition 5.3 except that they are closed. However, to gain the converse – in essence, that our denotational semantics is adequate – we must restrict our attention to the processes we call *nice*. We are forced to do this because, in general, there is the possibility of hiding variable resources over a free variable, and no rank function we can devise can properly account for this possibility. So we restrict our attention to those processes for which this problem does not arise. We denote by $\preceq$ the subterm order.

Definition 5.4 Let $p \in \mathcal{L}$ be a term. We call p *nice* when, for all subterms $p_1 \backslash R \preceq p$, if $R \cap \mathcal{R}_\mathcal{V} \neq \emptyset$, then $p_1 \in \mathcal{L}_c$ is closed. We use $\mathcal{L}_n$ and $\mathcal{L}_{c,n}$ to denote the subset of nice terms, respectively, of closed and nice terms from $\mathcal{L}$.

Example 5.5 To illustrate our definition, we present the following examples:

(i) Let $p = (\mathrm{rec}\, x.a\,;x) \backslash \rho_x$. Then p is nice, since the only variable within p – x – is bound *inside* the scope of the hiding operator $\backslash \rho_x$.

(ii) If $p = \mathrm{rec}\, x.((a\,;x) \backslash R)$, where R does not contain variable resources, then p is nice.

(iii) On the other hand, the process $p = \mathrm{rec}\, x.((a\,;x) \backslash \rho_y)$ is not nice, since the subterm $q = (a\,;x) \backslash \rho_y$ has a free variable within the scope of the hiding operator $\backslash \rho_y$. Note that this would still be the case even if $y = x$.

The nice property is a syntactic restriction that is preserved by the operational transitions. This is important because it ensures that if we start with a nice process, we will never leave the class of nice processes.

Lemma 5.6

(i) *If* $p \in \mathcal{L}_{c,n}$ *and* $q \in \mathcal{L}_n$*, then* $q[p/x] \in \mathcal{L}_n$.

(ii) *If* $p \in \mathcal{L}_{c,n}$*,* $a \in \Sigma \cup \{1\}$ *and* $p \xrightarrow{a} p'$*, then* $p' \in \mathcal{L}_{c,n}$.

Proof. For part (i), we first note that there is nothing to prove if x is not free in q. On the other hand, if x is free in q we proceed by induction on q.

- If $q = x$, then $q[p/x] = p \in \mathcal{L}_n$, and the result is clear.
- If $q = q_1 \mathrm{\,op\,} q_2$, then $q[p/x] = q_1[p/x] \mathrm{\,op\,} q_2[p/x]$. Now, if $p' = p_1 \backslash R \preceq q[p/x]$ then $p' \preceq q_1[p/x]$ or $p' \preceq q_2[p/x]$, and so the result is clear by induction.
- If $q = q_1 \backslash R$, then $q \in \mathcal{L}_n$ implies that either $R \cap \mathcal{R}_\mathcal{V} = \emptyset$ or $q_1 \in \mathcal{L}_c$ is closed. But x is free in q, so the first case must hold. Now, $q[p/x] = q_1[p/x] \backslash R$, and so any subterm $p' = p_1 \backslash R \preceq q[p/x]$ is either equal to $q[p/x]$ or it is a subterm of $q_1[p/x]$. The result follows from the inductive hypothesis and $R \cap \mathcal{R}_\mathcal{V} = \emptyset$.
- Assume $q = \mathrm{rec}\, y.q_1$. Since p is closed we have $q[p/x] = \mathrm{rec}\, y.(q_1[p/x])$. Again, if $p' = p_1 \backslash R$ is a subterm of $q[p/x]$ then it is a subterm of $q_1[p/x]$. The result follows by induction.

This completes the proof of part (i). For part (ii), we proceed by structural induction on p.

- The cases $p = \mathrm{SKIP}$ or $p = x$ are not possible.
- If $p = b$, then $p \xrightarrow{a} p'$ implies $a = b$ and $p' = \mathrm{SKIP}$, which is nice.
- If $p = p_1 \mathrm{\,op\,} p_2$, then $p = p_1 \mathrm{\,op\,} p_2 \xrightarrow{a} p_1' \mathrm{\,op\,} p_2' = p'$ with $p_1 \xrightarrow{a} p_1'$ and $p_2 = p_2'$ or $p_1 = p_1'$ and $p_2 \xrightarrow{a} p_2'$ or $p_1 \xrightarrow{a} p_1'$ and $p_2 \xrightarrow{a} p_2'$. By induction, p_1' and p_2' are nice and therefore, p' is nice.
- If $p = p_1 \backslash R$, then $p \xrightarrow{a} p'$ implies $p_1 \xrightarrow{b} p_1'$ for some p_1' and b with $b \backslash R = a$

and $p' = p'_1 \backslash R$. By induction, p'_1 is nice. Since p is closed, so is p_1 and p'_1 and we deduce that $p' = p'_1 \backslash R$ is nice.

- If $p = \mathrm{rec}\, x.p_1$, then $p \xrightarrow{a} p'$ implies $a = \rho_x$ and $p' = p_1[p/x]$. The first part of this lemma then implies p' is nice.

□

We can now state the converse of Proposition 5.3.

Proposition 5.7 *Let $p \in \mathcal{L}_{c,n}$ be a closed, nice process.*

(i) *If $a \in \Sigma$ and $a \lhd [\![p]\!]$, then $p \stackrel{a}{\Longrightarrow}$.*

(ii) *If $u \in \Sigma^*$ and $u \lhd [\![p]\!]$, then $p \stackrel{u}{\Longrightarrow}$.*

Even though it is restricted to nice processes, the proof of this proposition is not possible using standard structural induction on the terms of $\mathcal{L}$. We need a new rank function. That new rank function, and the proof of Proposition 5.7 are the subject of the next section.

5.2 The Rank Function

The rank function we need has two components, the first of which measures the hiding depth of variable resources. The second component essentially measures the structural depth of a term except that it is reset to 0 when we hide some variable resources. This will not be a problem since in this case the first component of the rank function increases, and we endow $\mathbb{N} \times \mathbb{N}$ with the lexicographic order. We denote by $n \vee m$ the maximum of $n, m \in \mathbb{N}$.

Definition 5.8 Define $l : \mathcal{L} \to \mathbb{N} \times \mathbb{N}$ by $l(p) = (l_1(p), l_2(p))$ with

- $l(\mathrm{SKIP}) = l(a) = l(x) = (0, 0)$, for all $a \in \mathrm{Act}$ and all $x \in \mathcal{V}$;
- $l(p_1 \,\mathrm{op}\, p_2) = (l_1(p_1) \vee l_1(p_2), 1 + l_2(p_1) + l_2(p_2))$, for all processes p_1 and p_2, and the operators $\mathrm{op} = \circ$ or $\mathrm{op} = ;$ or $\mathrm{op} = \underset{C}{\|}$;
- $l(\mathrm{rec}\, x.p) = (l_1(p), 1 + l_2(p))$, for all $p \in \mathcal{L}$ and $x \in \mathcal{V}$; and
- $l(p \backslash R) = \begin{cases} (l_1(p) + 1, 0) & \text{if } R \cap \mathcal{R}_{\mathcal{V}} \neq \emptyset, \\ (l_1(p), 1 + l_2(p)) & \text{otherwise,} \end{cases}$ for all $p \in \mathcal{L}$ and $R \subseteq \mathcal{R}$.

A crucial property of this rank function is that it is invariant when performing a transition of the operational semantics whose label is not a variable resource. This fact may seem strange since unwinding a recursion usually increases the structural depth of a process as in

$$\mathrm{rec}\, x.(a \,;\, x) \xrightarrow{\rho_x} a \,;\, \mathrm{rec}\, x.(a \,;\, x).$$

Indeed, the rank function does increase with this transition and this is the reason for the restriction "whose label is not a variable resource". Still, unwinding a recursion whose variable resource is then hidden is admissible, as in

$$(\mathrm{rec}\, x.(a\,;x))\backslash\rho_x \longrightarrow (a\,;\mathrm{rec}\, x.(a\,;x))\backslash\rho_x.$$

But since l_2 is reset to 0 when we hide a variable resource, the result also holds for recursion unwindings that are hidden. We start by proving the result for the second component of the rank function.

Lemma 5.9 *If $p \in \mathcal{L}_c$ is a closed term and $a \in \mathrm{Act} \cup \{1\}$, then $p \xrightarrow{a} p'$ implies $l_2(p) = l_2(p')$.*

Proof. We proceed by structural induction on p.

- $p = \mathrm{SKIP}$ or $p = x$ are not possible.
- If $p = b$, then $b = a$ and $p' = \mathrm{SKIP}$. Then $l_2(p) = l_2(\mathrm{SKIP}) = 0$.
- If $p = p_1 \mathbin{\mathrm{op}} p_2$, then $p = p_1 \mathbin{\mathrm{op}} p_2 \xrightarrow{a} p'_1 \mathbin{\mathrm{op}} p'_2 = p'$ with $p_1 \xrightarrow{a} p'_1$ and $p_2 = p'_2$, or $p_1 = p'_1$ and $p_2 \xrightarrow{a} p'_2$, or $p_1 \xrightarrow{a} p'_1$ and $p_2 \xrightarrow{a} p'_2$. In any case, $l_2(p_i) = l_2(p'_i)$ for each $i = 1, 2$ by induction. We deduce that $l_2(p) = l_2(p')$.
- If $p = p_1 \backslash R$, then $p_1 \xrightarrow{b} p'_1$ for some p'_1 and $b \in \mathrm{Act} \cup \mathcal{R}_\mathcal{V} \cup \{1\}$ with $p'_1 \backslash R = p'$ and $b \backslash R = a$. We have two cases:
  · If $R \cap \mathcal{R}_\mathcal{V} \neq \emptyset$, then $l_2(p) = 0 = l_2(p')$.
  · If $R \cap \mathcal{R}_\mathcal{V} = \emptyset$, then $a \in \mathrm{Act} \cup \{1\}$ implies $b \in \mathrm{Act} \cup \{1\}$ as well. By induction, $l_2(p'_1) = l_2(p_1)$, and we obtain $l_2(p) = l_2(p')$.
- Finally, $p = \mathrm{rec}\, x.p_1$ is not possible, since this term can only make a ρ_x transition, which is not allowed since $a \in \mathrm{Act} \cup \{1\}$.

□

An important property of the rank function l_1 is that it is invariant under all transitions of the operational semantics, even unwindings of recursions that are not hidden. This results holds for nice processes, which is precisely why we introduced this notion.

Lemma 5.10

(i) *If $p \in \mathcal{L}_c$ and $q \in \mathcal{L}_n$, then $l_1(q) \leq l_1(q[p/x]) \leq l_1(q) \vee l_1(p)$.*
(ii) *If $p \in \mathcal{L}_{c,n}$, $a \in \Sigma \cup \{1\}$ and $p \xrightarrow{a} p'$, then $l_1(p) = l_1(p')$.*

Proof. For part (i), we first note that there is nothing to prove if x is not free in q. On the other hand, if x is free in q, then we proceed by induction on q.

- If $q = x$, then $q[p/x] = p$, so $0 = l_1(q) \leq l_1(q[p/x] = l_1(p) = l_1(q) \vee l_1(p)$.
- If $q = q_1 \mathbin{\mathrm{op}} q_2$, then $q[p/x] = q_1[p/x] \mathbin{\mathrm{op}} q_2[p/x]$. By induction we have $l_1(q_i) \leq l_1(q_i[p/x]) \leq l_1(q_i) \vee l_1(p)$ for each i, and so
$$\begin{aligned} l_1(q) &= l_1(q_1 \mathbin{\mathrm{op}} q_2) = l_1(q_1) \vee l_1(q_2) \\ &\leq l_1(q_1[p/x]) \vee l_1(q_2[p/x]) = l_1(q_1[p/x] \mathbin{\mathrm{op}} q_2[p/x]) = l_1(q[p/x]) \\ &\leq (l_1(q_1) \vee l_1(p)) \vee (l_1(q_2) \vee l_1(p)) = l_1(q) \vee l_1(p). \end{aligned}$$
- If $q = q_1 \backslash R$, then $q[p/x] = (q_1[p/x]) \backslash R$. Since x is free in q and q is nice, we must have $R \cap \mathcal{R}_\mathcal{V} = \emptyset$. Using the induction hypothesis and the definition

of l_1 we obtain

$$l_1(q) = l_1(q_1) \leq l_1(q_1[p/x]) = l_1(q[p/x]) \leq l_1(q_1) \vee l_1(p) = l_1(q) \vee l_1(p).$$

- If $q = \mathrm{rec}\, y.q_1$, then $q[p/x] = \mathrm{rec}\, y.(q_1[p/x])$ since p is closed. Using the induction hypothesis and the definition of l_1 we obtain

$$l_1(q) = l_1(q_1) \leq l_1(q_1[p/x]) = l_1(q[p/x]) \leq l_1(q_1) \vee l_1(p) = l_1(q) \vee l_1(p).$$

For part (ii), we proceed by structural induction on p.

- If $p = \mathrm{SKIP}$ or $p = x \in \mathcal{V}$, then there is nothing to prove.
- If $p = b \in \mathrm{Act}$, then $b = a$, $p' = \mathrm{SKIP}$ and we have $l_1(p) = 0 = l_1(p')$.
- If $p = p_1 \mathbin{\mathrm{op}} p_2$, then $p = p_1 \mathbin{\mathrm{op}} p_2 \xrightarrow{a} p'_1 \mathbin{\mathrm{op}} p'_2 = p'$ with $p_1 \xrightarrow{a} p'_1$ and $p_2 = p'_2$, or $p_1 = p'_1$ and $p_2 \xrightarrow{a} p'_2$, or $p_1 \xrightarrow{a} p'_1$ and $p_2 \xrightarrow{a} p'_2$ and we obtain by induction $l_1(p) = l_1(p_1) \vee l_1(p_2) = l_1(p'_1) \vee l_1(p'_2) = l_1(p')$.
- If $p = \mathrm{rec}\, x.p_1$, then $a = \rho_x$ and $p' = p_1[p/x]$. We can apply (i), and conclude that $l_1(p) = l_1(p_1) \leq l_1(p_1[p/x]) = l_1(p') \leq l_1(p_1) \vee l_1(p) = l_1(p)$.
- If $p = p_1 \backslash R$, then $p_1 \xrightarrow{b} p'_1$ for some p'_1 and $b \in \Sigma \cup \{1\}$ with $p'_1 \backslash R = p'$ and $b \backslash R = a$. Then $l_1(p'_1) = l_1(p_1)$ by induction, and we obtain $l_1(p) = l_1(p')$.

□

Now, we summarize in a proposition the results concerning the rank function that will be needed for the proof of Proposition 5.7.

Proposition 5.11

(i) *Let $p \in \mathcal{L}_{c,n}$ and $u \in \Sigma^*$. If $p \overset{u}{\Longrightarrow} p'$, then $p' \in \mathcal{L}_{c,n}$ is closed and nice.*

(ii) *Let $p \in \mathcal{L}_{c,n}$ and $u \in \Sigma^*$. If $p \overset{u}{\Longrightarrow} p'$, then $l_1(p) = l_1(p')$.*

(iii) *Let $p \in \mathcal{L}_c$ and $u \in \mathrm{Act}^*$. If $p \overset{u}{\Longrightarrow} p'$, then $l_2(p) = l_2(p')$.*

(iv) *Let $p, q \in \mathcal{L}$. If q is a proper subterm of p, then $l(q) < l(p)$.*

Proof. (i), (ii) and (iii) follow directly by induction on the derivation length from Lemmas 5.6 and 5.10 and 5.9. The proof of (iv) is by structural induction on p.

- $p = \mathrm{SKIP}$, or $p = a \in \mathrm{Act}$, or $p = x \in \mathcal{V}$ are not possible.
- If $p = p_1 \mathbin{\mathrm{op}} p_2$, then $q \preceq p_i$ for some $i = 1$ or $i = 2$. By induction we get $l(q) \leq l(p_i)$ and we have $l(p_i) < l(p)$.
- If $p = \mathrm{rec}\, x.p_1$ or $p = p_1 \backslash R$, then $q \preceq p_1$. By induction we get $l(q) \leq l(p_1)$ and we have $l(p_1) < l(p)$.

□

We are now ready to prove the converse of Proposition 5.3.

Proof of Proposition 5.7. (ii) follows directly by induction on the length of u using (i). In order to prove (i) we use induction on $l(p)$. Let $p \in \mathcal{L}_{c,n}$

and assume that for all $q \in \mathcal{L}_{c,n}$ with $l(q) < l(p)$ property (i) holds for q. We consider the possible structure for p.

- $p = \text{SKIP}$, or $p = x \in V$ are not possible.
- If $p = b \in \text{Act}$, then $a \lhd [\![p]\!] = (b, \emptyset)$ implies $a = b$ and so $p \xrightarrow{a} \text{SKIP}$.
- If $p = \text{rec}\, x.p_1$, then $a \lhd [\![p]\!]$ implies by Lemma 4.27 that $a = \rho_x$ and then $p \xrightarrow{a} p_1[p/x]$.
- If $p = p_1 \,;\, p_2$, then $a \lhd [\![p]\!] = [\![p_1]\!] \,;\, [\![p_2]\!]$. By Lemma 4.7 there are two cases:
  · $a \lhd [\![p_1]\!]$. By Proposition 5.11 we have $l(p_1) < l(p)$ hence we can apply the induction hypothesis and we get $p_1 \overset{a}{\Longrightarrow}$. Using the definition of the operational semantics we deduce easily that $p \overset{a}{\Longrightarrow}$.
  · $a \lhd [\![p_2]\!]$ and $[\![p_1]\!] = (1, \emptyset)$. By Proposition 5.11 we have $l(p_2) < l(p)$ hence we can apply the induction hypothesis and conclude $p_2 \overset{a}{\Longrightarrow}$. Since $\text{res}(p_1) = \text{res}([\![p_1]\!]) = \emptyset$ by Lemma 4.30, we deduce easily that $p \overset{a}{\Longrightarrow}$.
- If $p = p_1 \circ p_2$, then $a \lhd [\![p]\!] = [\![p_1]\!] \circ [\![p_2]\!]$. By Lemma 4.13 there are two cases:
  · $a \lhd [\![p_1]\!]$ is similar to the case for strict sequential composition.
  · $a \lhd [\![p_2]\!]$ and $a\ I\ [\![p_1]\!]$. By Proposition 5.11 we have $l(p_2) < l(p)$ hence we can apply the induction hypothesis to conclude that $p_2 \overset{a}{\Longrightarrow}$. Since $\text{res}(p_1) = \text{res}([\![p_1]\!])$ by Lemma 4.30, we have $a\ I\ p_1$ and we deduce easily that $p \overset{a}{\Longrightarrow}$.
- If $p = p_1 \underset{C}{\|} p_2$, then $a \lhd [\![p]\!] = [\![p_1]\!] \underset{C}{\|} [\![p_2]\!]$. By Lemma 4.18 there are three cases:
  · $a \lhd [\![p_1]\!]$ and $a\ I\ [\![p_2]\!], C$. By Proposition 5.11 we have $l(p_1) < l(p)$ hence we can apply the induction hypothesis and we get $p_1 \overset{a}{\Longrightarrow}$. Since $\text{res}(p_2) = \text{res}([\![p_2]\!])$ by Lemma 4.30 we also have $a\ I\ p_2, C$. Using the definition of the operational semantics we deduce easily that $p \overset{a}{\Longrightarrow}$.
  · $a \lhd [\![p_2]\!]$ and $a\ I\ [\![p_1]\!], C$ is similar.
  · $a \lhd [\![p_1]\!]$ and $a \lhd [\![p_2]\!]$ and $a \in C$. By Proposition 5.11 we have $l(p_i) < l(p)$ for $i = 1, 2$, hence we can apply the induction hypothesis and we get $p_i \overset{a}{\Longrightarrow}$. Again, using the definition of the operational semantics, we deduce easily that $p \overset{a}{\Longrightarrow}$.
- $p = p_1 \backslash R$ is the most difficult case. This is where a structural induction is not sufficient and why we introduced the rank function l. We start with a technical result: If $p_1 \overset{w}{\Longrightarrow} p_1'$ with $w \in \Sigma^*$ and $w \backslash R = 1$, then $l(p_1') < l(p)$. We have two cases.
  · Either $R \cap \mathcal{R}_\mathcal{V} \neq \emptyset$. Then, using Proposition 5.11 (ii) and the definition of l_1 we get $l_1(p) = l_1(p_1) + 1 > l_1(p_1) = l_1(p_1')$. In this case, we do not use the fact $w \backslash R = 1$.
  · Or $R \cap \mathcal{R}_\mathcal{V} = \emptyset$. Using the definition of the rank function and Proposition 5.11 we get $l_1(p) = l_1(p_1) = l_1(p_1')$ and $l_2(p) = 1 + l_2(p_1) = 1 + l_2(p_1')$. For the last equality, we use the fact that $R \cap \mathcal{R}_\mathcal{V} = \emptyset$ and $w \backslash R = 1$ implies $w \in \text{Act}^*$ which allows to apply Proposition 5.11 (iii).

  Now, if $a \lhd [\![p]\!] = [\![p_1]\!] \backslash R$, then Lemma 4.22 implies there are some $r, s \in \mathbb{R}_f$

with $r \leq s \leq [\![p_1]\!]$, $r^{-1}s = b \in \Sigma$, $r \backslash R = 1$ and $b \backslash R = a$. Let $u \in \mathrm{Lin}(r)$ be any linearization of r. By Remark 5.2 (i) we deduce that $ub \lhd [\![p_1]\!]$ and we also have $u \backslash R = 1$.

We claim that $p_1 \stackrel{v}{\Longrightarrow}$ for all $v \leq ub$ and we prove this claim by induction on the length of the word v. If $|v| = 0$, then $v = 1$ and $p_1 \stackrel{1}{\Longrightarrow} p_1$. Now assume that $v = wc$. By induction we know that $p_1 \stackrel{w}{\Longrightarrow} p_1'$. Also, $w \leq u$ and therefore $w \backslash R = 1$. We have seen above that $l(p) > l(p_1')$. Now, $p_1 \stackrel{w}{\Longrightarrow} p_1'$ and Proposition 5.3 implies $[\![p_1']\!] = w^{-1}[\![p_1]\!]$. Since $v = wc \lhd [\![p_1]\!]$ we deduce $c \lhd [\![p_1']\!]$. Finally, since $l(p) > l(p_1')$ we can apply the inductive hypothesis and we get $p_1' \stackrel{c}{\Longrightarrow}$. Therefore $p_1 \stackrel{wc}{\Longrightarrow}$, which concludes the proof of the Claim.

Taking $v = ub$ we obtain $p_1 \stackrel{ub}{\Longrightarrow}$ and $p \stackrel{a}{\Longrightarrow}$ since $(ub) \backslash R = a$.

□

5.3 Adequacy and the Congruence Theorem

In this section we bring the previous results together to show our denotational and operational semantics agree. We first define the observable behaviors of a closed process and show that the denotational semantics is adequate with respect to these observable behaviors. Then, we introduce the complex behaviors of a process by allowing to observe the resources claimed after executing some transitions and by abstracting away from irrelevant orders between transitions. Finally, we prove a congruence theorem, stating that the complex behavior function on processes coincides with the semantic map we defined into the denotational model.

We start by defining the observable behavior map of a closed process. It consists of all possible strings that can be observed of the process.

Definition 5.12 For a closed term $p \in \mathcal{L}_c$, we define

$$X_{\Sigma^*}(p) = \{u \in \Sigma^* \mid p \stackrel{u}{\Longrightarrow}\}.$$

From Propositions 5.3 and 5.7, we deduce immediately the following characterization of the observable behavior of a process.

Proposition 5.13 *If $p \in \mathcal{L}_{c,n}$ is closed and nice, then*

$$X_{\Sigma^*}(p) = \{u \in \Sigma^* \mid u \lhd [\![p]\!]\}.$$

From this, we deduce that our denotational semantics is adequate with respect to the operational rules. As usual, by a *context*, we mean a term $C(-) \in \mathcal{L}$ with one free variable (denoted by $-$). The context is *nice-preserving* if $C(p)$ is nice whenever p is nice and closed.

Theorem 5.14 (Adequacy) *If $p, q \in \mathcal{L}_{c,n}$ are closed, nice processes and $C(-) \in \mathcal{L}$ is a nice-preserving context, then $[\![p]\!] = [\![q]\!]$ implies $X_{\Sigma^*}(C(p)) = X_{\Sigma^*}(C(q))$.*

Proof. We note that, since the semantic map is compositional, the hypothesis that $[\![p]\!] = [\![q]\!]$ implies $[\![C(p)]\!] = [\![C(q)]\!]$. Moreover, since the context is nice-preserving, we deduce that $C(p)$ and $C(q)$ are both nice and closed. The result follows from Proposition 5.13. $\square$

Now, we define a richer behavior map that is still closely related with our denotational semantics. We do this first by observing the resources that are still claimed after executing some transitions in the operational semantics, and second by abstracting away from the unnecessary order between transitions that we observe in the operational semantics. For instance, if a process p may perform ab or ba, then a and b are actually concurrent in p. Our *complex behavior* map precisely reflects this fact.

Any word $u \in \Sigma^*$ can be regarded as a finite pomset when endowed with the linear order and we can use its canonical representative $(V_u, \leq_u)$. If we consider a set $K \subseteq \Sigma^*$ of words, all having the same canonical vertex set V, then we define a finite pomset $\cap K$ by taking the intersection of the total orders induced by these words: $\cap K = (V, \cap_{u \in K} \leq_u)$. For instance, the pomset induced by $K = \{abcdabc, adcbacb\}$ is

$$\cap K \quad = \quad \begin{tikzcd} & b \arrow[dr] & & b \\ a \arrow[ur] \arrow[r] \arrow[dr] & d \arrow[r] & a \arrow[ur] \arrow[dr] & \\ & c \arrow[ur] & & c \end{tikzcd}$$

Definition 5.15 Let $p \in \mathcal{L}_c$ be a closed process. We define

$$X_{\mathbb{C}}(p) = \{(\cap[u]_p, \mathrm{res}(p')) \mid p \overset{u}{\Longrightarrow} p'\},$$

where $[u]_p = \{v \in \Sigma^* \mid p \overset{v}{\Longrightarrow} \text{ and } V_u = V_v\}$.

Recall (Definition 2.14) that $\chi(x) = \{(s, \mathrm{res}(s^{-1}x)) \mid s \leq x \text{ and } s \in \mathbb{R}_f\}$, for each $x \in \mathbb{C}$. We are now able to state the congruence theorem. In particular, the following result implies that $X_{\mathbb{C}}(p)$ consists of complex pomsets, something that is not clear from its definition.

Theorem 5.16 (Congruence) *For all closed, nice processes $p \in \mathcal{L}_{c,n}$, we have*

$$X_{\mathbb{C}}(p) = \chi([\![p]\!]) \qquad \text{and} \qquad \sqcup X_{\mathbb{C}}(p) = [\![p]\!].$$

Proof. We first claim that for $s \in \mathbb{R}_f$, if $s \leq [\![p]\!]$ and $u \in \mathrm{Lin}(s)$ is a linearization of s, then $\mathrm{Lin}(s) = [u]_p$. Indeed, if $v \in \mathrm{Lin}(s)$, then $v \triangleleft [\![p]\!]$ by Remark 5.2, and Proposition 5.7 implies $p \overset{v}{\Longrightarrow}$. Also, $u, v \in \mathrm{Lin}(s)$ implies that $V_u = V_v$ and thus $v \in [u]_p$. Conversely, if $v \in [u]_p$, then Proposition 5.3 implies $v \triangleleft [\![p]\!]$. From Remark 5.2 we get that $v \in \mathrm{Lin}(r)$ for some $r \leq [\![p]\!]$. Using $u \in \mathrm{Lin}(s)$, we deduce that $V_r = V_v = V_u = V_s$. Hence, $r = s$ and $v \in \mathrm{Lin}(s)$. This finishes the proof of the Claim.

Now, to prove the Proposition, we first assume $s \leq [\![p]\!]$ for some $s \in \mathbb{R}_f$ and we fix some $u \in \mathrm{Lin}(s)$. The Claim implies $[u]_p = \mathrm{Lin}(s)$, and it follows that $s = \cap[u]_p$, since any pomset is the intersection of its linear extensions. Now, $s \le [\![p]\!]$ and $u \in \mathrm{Lin}(s)$ implies $u \lhd [\![p]\!]$ and by Proposition 5.7 we obtain $p \stackrel{u}{\Longrightarrow} p'$ for some p'. Using Proposition 5.3, it follows $s^{-1}[\![p]\!] = u^{-1}[\![p]\!] = [\![p']\!]$. Using Lemma 4.30 we obtain $\mathrm{res}(s^{-1}[\![p]\!]) = \mathrm{res}([\![p']\!]) = \mathrm{res}(p')$. This shows $(s, \mathrm{res}(s^{-1}[\![p]\!])) = (\cap[u]_p, \mathrm{res}(p')) \in X_{\mathbb{C}}(p)$.

Conversely, suppose that $p \stackrel{u}{\Longrightarrow} p'$. Then Proposition 5.3 implies $u \lhd [\![p]\!]$ and $u^{-1}[\![p]\!] = [\![p']\!]$. Then there is a (unique!) $s \le [\![p]\!]$ with $u \in \mathrm{Lin}(s)$. The Claim above implies $s = \cap[u]_p$, and again from Lemma 4.30 we obtain $\mathrm{res}(s^{-1}[\![p]\!]) = \mathrm{res}(u^{-1}[\![p]\!]) = \mathrm{res}([\![p']\!]) = \mathrm{res}(p')$. Therefore, $(\cap[u]_p, \mathrm{res}(p')) = (s, \mathrm{res}(s^{-1}[\![p]\!])) \in \chi([\![p]\!])$.

From Lemma 2.15, we deduce immediately $\sqcup X_{\mathbb{C}}(p) = [\![p]\!]$. □

5.4 Full Abstraction

In this section we show that our denotational semantics $[\![-]\!]$ is fully abstract with respect to the weakest notion of behavior X_{Σ^*} – that of observing only the strings that a process can execute.

Theorem 5.17 (Full abstraction) *For all closed and nice processes $p, q \in \mathcal{L}_{c,n}$, the following are equivalent*

(i) $[\![p]\!] = [\![q]\!]$,

(ii) *for all nice-preserving contexts $C(-)$ we have $X_{\Sigma^*}(C(p)) = X_{\Sigma^*}(C(q))$.*

The Congruence Theorem 5.16 immediately implies that the denotational semantics is adequate and fully abstract with respect to the complex behavior map $X_{\mathbb{C}}$ that extracts both the real pomset of actions a process can execute, as well as the imaginary part of resources still claimed after the execution is complete. But in this section we use the weaker behavior map X_{Σ^*}.

We need a lemma before proving Theorem 5.17.

Lemma 5.18 *If $s, t \in \mathbb{R}$ and $s \neq t$, then there is some $u \in \Sigma^*$ with either $u \lhd s$ and $u \not\lhd t$, or vice versa.*

Proof. If $|s|_a > |t|_a$ for some $a \in \Sigma$, then we find a finite prefix $r \le s$ with $|r|_a > |t|_a$ and taking any linearization u of r we get $u \lhd s$ and $u \not\lhd t$. Now, assume $|s|_a = |t|_a$ for all $a \in \Sigma$. The canonical representations for $s = (V_s, \le_s)$ and $t = (V_t, \le_t)$ have the same vertex set $V_s = V_t = \{(a, i) \mid 0 \le i < |s|_a = |t|_a\}$. Since $s \neq t$ we must have $\le_s \neq \le_t$ hence we find $p, q \in V_s = V_t$ with for instance $p \le_t q$ and $p \not\le_s q$. Let $r = (\downarrow_s q, \le_s|_{\downarrow_s q \times \downarrow_s q}) \le s$. Since $p \notin \downarrow_s q$ and $p \le_t q$, $\downarrow_s q$ is not a lower set of $(V_t, \le_t)$ hence r is not a prefix of t. Now, taking any linearization u of r we get $u \lhd s$ and $u \not\lhd t$. □

Proof of Theorem 5.17. (i) implies (ii) is just Theorem 5.14. Conversely, let $p, q \in \mathcal{L}_{c,n}$ with $[\![p]\!] \neq [\![q]\!]$. If $\mathrm{Re}([\![p]\!]) \neq \mathrm{Re}([\![q]\!])$, then Lemma 5.18 implies there is some $u \lhd \mathrm{Re}([\![p]\!])$ with $u \not\lhd \mathrm{Re}([\![q]\!])$, or vice versa. Assuming the former, then Proposition 5.7 implies $p \stackrel{u}{\Longrightarrow}$, while Proposition 5.3 implies $q \stackrel{u}{\not\Longrightarrow}$. Hence, the identity context serves to distinguish p and q in this case.

On the other hand, if $\mathrm{Re}([\![p]\!]) = \mathrm{Re}([\![q]\!])$, then $[\![p]\!] \neq [\![q]\!]$ implies $\mathrm{Im}([\![p]\!]) \neq \mathrm{Im}([\![q]\!])$. We assume then that $\mathrm{Im}([\![q]\!])\setminus\mathrm{Im}([\![p]\!]) \neq \emptyset$. Now, we let $R = \mathrm{Im}([\![p]\!])$, we choose any $a \in \mathrm{Act}$ and define the nice-preserving context $C(-) = (-\setminus R);a$. We have $[\![p \setminus R]\!] = [\![p]\!] \setminus R = (\mathrm{Re}([\![p]\!]) \setminus R, \emptyset)$, and so $\mathrm{Re}([\![p]\!] \setminus R) \in \mathbb{R}_f$ is finite and $\mathrm{Re}([\![C(p)]\!]) = \mathrm{Re}([\![p \setminus R]\!]) \,;\, a$. Moreover, the imaginary part of $[\![q \setminus R]\!] = [\![q]\!] \setminus R$ is the non-empty set $\mathrm{Im}([\![q]\!]) \setminus \mathrm{Im}([\![p]\!])$. Therefore, we have $\mathrm{Re}([\![C(q)]\!]) = \mathrm{Re}([\![q \setminus R]\!])$. Now, since $\mathrm{Re}([\![p]\!]) = \mathrm{Re}([\![q]\!])$, we deduce that $\mathrm{Re}([\![p\setminus R]\!]) = \mathrm{Re}([\![q\setminus R]\!])$ and so $|\mathrm{Re}([\![C(p)]\!])|_a > |\mathrm{Re}([\![C(q)]\!])|_a$. Taking $u \in \Sigma^*$ to be any linearization of $\mathrm{Re}([\![C(p)]\!])$ we deduce as above that $C(p) \stackrel{u}{\Longrightarrow}$, but $C(q) \stackrel{u}{\not\Longrightarrow}$. ◻

We close this section with one further observation about our denotational semantics: every compact complex pomset is the denotation of some closed term from our language, therefore, our denotational model of complex pomsets is optimal.

Proposition 5.19 *If $x = (s, S) \in \mathbb{C}_f$ is finite, then $x = [\![p]\!]$ is denotable by some process $p \in \mathcal{L}_c$.*

Proof. We first prove that for all finite real pomsets $s \in \mathbb{R}_f$, there is a closed term $p \in \mathcal{L}_c$ such that $[\![p]\!] = (s, \emptyset)$. Assume that $s = (V, \leq, \lambda)$ with $V = \{v_1, \ldots, v_n\}$ such that $v_i \leq v_j$ implies $i \leq j$. With $b_i = \lambda(v_i) \in \Sigma$ we obtain that $b_1 \cdots b_n$ is a linearization of s.

Now, for each i, j with $v_i < v_j$ we introduce a new resource $\alpha_{i,j} \in \mathcal{R}_0 \setminus \mathrm{res}(s)$. Moreover, for each i we introduce a new resource $\beta_i \in \mathcal{R}_0\setminus\mathrm{res}(s)$ and we define $a_i = \{\beta_i\} \cup \{\alpha_{j,i} \mid v_j < v_i\} \in \mathrm{Act}$ and $c_i = \{\beta_i\} \cup \{\alpha_{i,j} \mid v_i < v_j\} \in \mathrm{Act}$. The resources β_i are just used to make sure that a_i and c_i are non-empty sets of resources even when v_i is minimal or maximal in $(V, \leq)$.

Next, for each i we consider the process

$$p_i = \begin{cases} a_i \,;\, b_i \,;\, c_i & \text{if } b_i \in \mathrm{Act}, \\ a_i \,;\, (\mathrm{rec}\, x_i.\mathrm{SKIP}) \,;\, c_i & \text{if } b_i = \{\rho_{x_i}\}. \end{cases}$$

It is easy to show that $[\![p_i]\!] = (a_i \to b_i \to c_i, \emptyset)$. Let $p = p_1 \circ \cdots \circ p_n$. The semantics of p is $(t, \emptyset)$ where $t = (a_1 \to b_1 \to c_1) \circ \cdots \circ (a_n \to b_n \to c_n)$. Hence, if we consider two disjoint copies $V' = \{v'_1, \ldots, v'_n\}$ and $V'' = \{v''_1, \ldots, v''_n\}$ of V and we let $V_1 = V' \cup V \cup V''$, then we can represent t by $(V_1, \leq_1, \lambda_1)$ with $\lambda_1(v'_i) = a_i$, $\lambda_1(v_i) = b_i$ and $\lambda_1(v''_i) = c_i$.

We claim that for all i, j, we have $v_i < v_j$ if and only if $v_i <_1 v_j$. One direction is easy. Indeed, assume that $v_i < v_j$. Then, $c_i \cap a_j = \{\alpha_{i,j}\}$ and therefore $v_i <_1 v''_i <_1 v'_j <_1 v_j$. Conversely, we use an induction on $j - i$. If $v_i <_1 v_k <_1 v_j$ for some k, then $i < k < j$ and we can conclude by induction. If $b_i \cap b_j \neq \emptyset$ then, the over-synchronization condition for s gives $v_i < v_j$. Hence, we assume that $b_i \cap b_j = \emptyset$ and that there is no k with $v_i <_1 v_k <_1 v_j$.

Using $a_k, c_k \, I \, b_l$ for all k, l and the definition of weak sequential composition, we deduce from $v_i <_1 v_j$ that $v_i <_1 v_i'' <_1 v_j' <_1 v_j$. Using in addition that for $k < l$ we have $a_k \, I \, a_l$, $a_k \, I \, c_l$ and $c_k \, I \, c_l$, we can deduce that there is no vertex between v_i'' and v_j' in t. Then, the definition of weak sequential composition implies $c_i \cap a_j \neq \emptyset$. We must have $c_i \cap a_j = \{\alpha_{i,j}\}$ which implies $v_i < v_j$ and we have proved the claim.

Finally, let $R = \{\beta_1, \ldots, \beta_n\} \cup \{\alpha_{i,j} \mid v_i < v_j\}$. Using the claim and the definition of hiding, we immediately get that $[\![p \backslash R]\!] = (t \backslash R, \emptyset) = (s, \emptyset)$ and we have proved that every finite and maximal complex pomset $(s, \emptyset)$ is denotable.

Now, let $S \in \mathcal{P}_f(\mathcal{R})$ be a finite set of resources. We prove that $(1, S)$ is denotable. Let $a = R \cap \mathcal{R}_0$ and let $R \cap \mathcal{R}_\mathcal{V} = \{\rho_{x_1}, \ldots, \rho_{x_n}\}$. Let $x_0 \in \mathcal{V} \backslash \{x_1, \ldots, x_n\}$ be a new variable and let $q_a = a$ if $\emptyset \neq a \in \mathrm{Act}$ and $q_a = \mathrm{SKIP}$ if $a = \emptyset$. We consider the process

$$q = (\mathrm{rec}\, x_0.x_0) \, ; (\mathrm{rec}\, x_1.x_1) \, ; \cdots ; (\mathrm{rec}\, x_n.x_n) \, ; q_a.$$

It is easy to see that $[\![\mathrm{rec}\, x_i.x_i]\!] = (\rho_{x_i}^\omega, \{\rho_{x_i}\})$ for all i. Using the definition of the strict sequential composition, we get $[\![q]\!] = (\rho_{x_0}^\omega, \{\rho_{x_0}, \ldots, \rho_{x_n}\} \cup a) = (\rho_{x_0}^\omega, \{\rho_{x_0}\} \cup S)$. Therefore, by the definition of hiding it follows $[\![q \backslash \rho_{x_0}]\!] = (1, S)$.

Finally, to prove the proposition, we just have to consider the process $(p \backslash R) \, ; (q \backslash \rho_{x_0})$ and we get

$$[\![(p \backslash R) \, ; (q \backslash \rho_{x_0})]\!] = [\![p \backslash R]\!] \, ; [\![q \backslash \rho_{x_0}]\!] = (s, \emptyset) \, ; (1, S) = (s, S).$$

$\square$

6 Summary

In this paper we have presented a truly concurrent semantics for a process algebra. We have used resource pomsets to define the denotational semantic model. We have presented an operational semantics for the language, which was shown to be congruent to the denotational semantics, and we also proved that the denotational model is adequate and fully abstract with respect to the behavior that extracts from a process's operational semantics only the set of strings the process can execute. Notable in our semantics is the treatment of hiding – we support the hiding of resources as well as of actions, which generalizes somewhat the original notion of hiding from *CSP* [18].

Our semantics also uses special variable resources that make the unwinding of recursion observable. In particular, this allows us to observe divergence. If we do not want to observe these unwindings of recursion, we can hide the variable resources. It should be noted that our congruence and full abstraction theorems only apply to the processes we called nice (see Definition 5.4), that is processes that do not hide variable resources over a process containing some free variables. We believe this is not a serious restriction to the expressiveness of our language and all natural processes we could think of can be written as nice processes. Still, we have this restriction and we leave as an open problem whether this restriction can be overcome.

The work reported here is part of an on-going program the authors have embarked upon whose goal is to understand better the difference between concurrency and nondeterminism. Our earlier work [5,6] considered a language similar to the one treated here: the hiding operator was not present in that language, nor was strict sequential composition, but, on the other hand, the earlier language included a restriction operator $p|_R$ that confines process p to the resource set R. The operational rule for restriction is simply

$$\frac{p \xrightarrow{a} p', \quad a \subseteq R}{p|_R \xrightarrow{a} p'|_R}$$

We have not considered that operator here because it can be derived from our parallel composition and hiding. One can show that $p|_R = (p \underset{C}{\|} p)\backslash\overline{R}$ where $C = \{a \in \Sigma \mid a \subseteq R\}$ and $\overline{R} = \mathcal{R} \setminus R$.

For the future, we intend to add nondeterminism to our language. It is clear that the language we have studied here has many interesting properties, and that there are interesting processes that can be written in the language. Nonetheless, we believe that a full understanding of the difference between the "true concurrency" approach to modelling concurrency and the more standard interleaving approach will be possible only in a setting where both are supported.

We also are looking at potential applications for the language we are developing. We believe there will be many such, but to name some areas in particular at this point is somewhat premature. The approach we take to defining the semantics – using resources – has obvious appeal, and suggests many possible applications. But until these appealing potential areas of application have actually been explored, we prefer not to predict what utility the language may have. In any case, we believe the results presented here shed light on how a semantics depending on pomsets and defined using resources can be crafted.

Other authors have considered event structures [19] and pomsets [14,15] as models of concurrency, but none that we know of has actually written a specific language and attempted the sort of analysis – operational and denotational – presented here. Most of this work – ours included – has its origins in Mazurkiewicz trace theory [10] and Petri nets [17], and it would be interesting to see how expressive the language we have presented is for modelling these constructs. Nonetheless, we believe that, along with [5,6], this represents the first presentation of a truly concurrent denotational semantics for a language including parallel composition and weak sequential composition.

7 Acknowledgement

The authors would like to take this opportunity to thank Maurice Nivat for providing us with the opportunity to discuss our mutual research interests. This opportunity has led to a fruitful collaboration, part of which is reflected in this work. We also wish to express our gratitude to the members of *LIAFA* and of the math department of Tulane University for their kind hospitality and for the stimulating environment they provided while this work was being carried out. Thanks are also due to the US Office of Naval Research and the National Science Foundation, whose support also helped to allow this work to take place.

References

[1] Abramsky, S. and A. Jung, *Domain Theory*, in: "Handbook of Computer Science and Logic," Volume **3**, Clarendon Press, 1995.

[2] Brookes, S. D, C. A. R. Hoare and A. W. Roscoe, *A theory of communicating sequential processes,* JACM **31** (1984), pp. 560–599.

[3] Diekert, V. and P. Gastin, *Approximating traces*, *Acta Informatica* **35** (1998), pp. 567–593.

[4] Diekert, V. and G. Rozenberg, editors, *The Book of Traces*, World Scientific, Singapore (1995).

[5] Gastin, P. and M. Mislove, *A truly concurrent semantics for a simple parallel programming language*, in: J. Flum and M. Rodríguez-Artalejo, editors, *Proceedings of the Annual Conference of the European Association for Computer Science Logic (CSL'99)*, Lecture Notes in Computer Science **1683** (1999), pp. 515–529.

[6] Gastin, P. and M. Mislove, *A truly concurrent semantics for a simple parallel programming language,* (journal version), submitted to Mathematical Structures for Computer Science, 2001.

[7] Gastin, P. and D. Teodesiu, *Resource traces: a domain for processes sharing exclusive resources*, *Theoretical Computer Science*, to appear.

[8] Gierz, G., K. H. Hofmann, K. Keimel, J. Lawson, M. Mislove and D. Scott, "A Compendium of Continuous Lattices," Springer-Verlag, Berlin, Heidelberg, New York (1980), 376pp.

[9] Hennessy, M. and G. D. Plotkin, *Full abstraction for a simple parallel programming language*, Lecture Notes in Computer Science **74** (1979), Springer-Verlag

[10] Mazurkiewicz, A., *Trace theory*, in: M.P. Chytil et al., editors, *Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science (MFCS'84)*, Lecture Notes in Computer Science **255** (1987), pp. 279–324.

[11] Milner, R., "A Calculus for Communicating Systems," Lecture Notes in Computer Science **92** (1980).

[12] Mislove, M. W. and F. J. Oles, *Full abstraction and recursion, Theoretical Computer Science* **151** (1995), pp. 207–256.

[13] Mislove, M. W., *Algebraic posets, algebraic cpos and models of concurrency,* in: G. M. Reed, A. W. Roscoe and R. F. Wachter, editors, "Topology and Computer Science," Oxford University Press, 1990.

[14] Pratt, V.R., *On the composition of processes*, Proceedings of the 9th ACM POPL (1982), pp. 213–223.

[15] Pratt, V.R., *Modelling concurrency with partial orders*, *J. of Parallel Programming* **15** (1987), pp. 33–71.

[16] Plotkin, G. D., *A structural approach to operational semantics*, Report DAIMI–FN–19, 1981.

[17] Reisig, W., "Petri Nets," EATCS Monographs in Theoretical Computer Science **4** (1985), Springer-Verlag.

[18] Roscoe, A.W., *CSP and determinism in security modelling*, Proceedings of the 1995 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, 1995.

[19] Winskel, G., "Events in Computation," Ph.D. Thesis, University of Cambridge, 1980.