Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 286 (2012) 307–321

www.elsevier.com/locate/entcs

A Truly Concurrent Process Semantics over Multi-Pomsets of Consumable Resources

Dan Teodosiu [1]

Paris Centre Mathematical Sciences
University Paris Diderot (Paris 7)
75205 PARIS CEDEX 13, France

Abstract

This paper develops a truly concurrent semantical approach, whereby concurrency is notionally independent of nondeterminism, that allows describing the deterministically concurrent behaviour of recursive processes accessing consumable resources. The process semantics is based on the new coherently complete and prime algebraic domains of real and complex multi-pomsets. The process language that we study contains several deterministic quantitative process operators, namely a renaming, a hiding, a restriction, a serial and a parallel operator, as well as a recursion operator. The displayed deterministic structural operational machine engenders a linear and a complex operational semantics. A compositional denotational semantics is constructed, which uses a functional domain over environments of complex multi-pomsets. The robustness of the presented semantical work is established by proving that the denotational semantics is fully abstract with respect to both linear and complex operational semantics.

Keywords: process calculus, true concurrency, resource, consumption, quantification, pomset, denotational semantics, structural operational semantics, full abstraction.

1 Introduction

The seminal work of Hennessy & Plotkin [6] has shown how *power domains* could be employed to build semantic models that support parallel composition of processes. Supposing that a choice operator is part of the language, parallelism is simply reduced to interleaving and choice, thus introducing nondeterminism into the semantic models.

In this presentation, we follow a genuine approach to parallelism and concurrency, which avoids using choice and nondeterminism. We, thereby, rely on *true concurrency*, which has inspired a wealth of domain-theoretic approaches. These are mainly due to the insight that the prime *event structures* of Winskel [12] are domains that provide suitable models to express truly concurrent process combinations.

[1] Email: dan.teodosiu@wanadoo.fr

1571-0661
doi:10.1016/j.entcs.2012.08.020

However, to our knowledge, the only truly concurrent semantical approach that has actually developed an operational and a matching denotational programming semantics, allowing for finitary process combinators, as well as recursion, originates in the work of Diekert & Gastin [2], Gastin & Teodosiu [5] and Gastin & Mislove [4]. The underlying denotational models are the *pomsets* advocated by Pratt [9] which, although being particular event structures, allow to simply express truly concurrent process combinations. Its intuition is based on the appealing interaction between processes and *resources* in any environment, a paradigm also driving the applied work in computer science of the last decades. One should also note the related work of Pym & Tofts [10] presenting a truly concurrent algebra and logic of processes and resources for a number of process combinators including choice.

A new stream in automata theory, as emphasized by the recent monograph on *weighted automata* [3], consists in attaching weights to transitions which express their *cost* or *consumption* in terms of some available resources and to extend these notions to recognized words and languages. Classical automata theory can in particular be recovered by considering unit weights, which is why the weighted view is more versatile, opening the way to new applications in engineering and economy.

The present process language combines the above truly concurrent approach on the denotational side with a weighted automata view on the operational side. To this end, the labeling of the denotational models and of the operational transition rules is quantified with the bounded or unbounded *time* or *amount* of resources being *consumed*. Quantification requires to replace the previously employed *algebra of sets* of resources, seen as vectors over the booleans, by an *algebra of multi-sets* of resources, seen as vectors over the extended positive reals, while taking care that relevant algebraic properties remain valid in this new setting. Furthermore, it leads to defining operators whose quantitative semantics enriches that of previously considered operators (serial, parallel), together with operators having a new, quantitative semantics (renaming, hiding, restriction, recursion).

Our semantics differs from previous approaches in several technical respects. Firstly, the representation of processes as complex multi-pomsets relies on a real part and a *consumption part*, which allows to trivially check the consumption commutation and continuity. Secondly, the new *prefix partial order* on complex multi-pomsets substantially simplifies the proofs, allowing to check the complex continuity solely for the real part. Finally, the *semantics of recursion* relies on the continuity with respect to two orders instead of one.

The paper is structured as follows. Section 2 recalls basic facts about *partial orders*. Section 3 introduces the domains of *real and complex multi-pomsets*. Section 4 presents the *process language* and defines the consumption semantics. Section 5 displays a deterministic structural operational machine, which engenders the linear and complex *operational semantics*. Section 6 is devoted to the compositional *denotational semantics* using a functional domain over environments of complex multi-pomsets. Section 7 presents the main results of *congruence and full abstraction* of the denotational semantics with respect to both linear and complex operational semantics.

The interested reader may find a complete version of our results in [11].

2 Partial Orders

For a detailed exposition we refer to the presentation of Abramsky & Jung [1].

A *partial order* (PO) is a pair $(X, \leq)$ where $\leq$ is a reflexive, antisymmetric and transitive binary relation on X. A subset $Y \subseteq X$ is *directed* (*coherent*) iff it is non-empty and for all $x, y \in Y$ there exists $z \in Y$ ($z \in X$) such that $x \leq z$ and $y \leq z$. Any directed set is coherent. $(X, \leq)$ is a *coherently complete PO (CCPO)* (*directed complete PO (DCPO)*) iff every coherent subset (directed subset) has a least upper bound. Any CCPO is a DCPO.

An element $x \in X$ is *prime* (*compact*) iff for all (directed) subsets $Y \subseteq X$ having a least upper bound, $x \leq \bigvee Y$ implies $x \leq y$ for some $y \in Y$. The set of all prime (compact) elements of X below $y \in X$ is denoted by $\mathrm{Prm}(y)$ ($\mathrm{Kmp}(y)$). A partial order $(X, \leq)$ is *p-algebraic* iff $x = \bigvee \mathrm{Prm}(x)$ for all $x \in X$. It is *k-algebraic* iff $\mathrm{Kmp}(x)$ is directed and $x = \bigvee \mathrm{Kmp}(x)$ for all $x \in X$. In the case of CCPOs, p-algebraicity also implies k-algebraicity.

A k-algebraic DCPO is called a *(Scott-)domain*. A mapping $F : (X, \leq) \to (X', \leq')$ is *(Scott-)continuous* iff for all directed sets $Y \subseteq X$, such that $\bigvee Y$ exists, $\bigvee' F(Y)$ exists, and $F(\bigvee Y) = \bigvee' F(Y)$.

The set of finite ordinals (i.e. the least infinite ordinal) is denoted by ω.

3 Real and Complex Multi-Pomsets

We denote by $\mathbb{R}_+$ the set of *positive reals* and by $\overline{\mathbb{R}}_+ = \mathbb{R}_+ \cup \{\infty\}$ the set of *positive extended reals*. The notations $\mathbb{R}$ and $\mathbb{C}$ are reserved for later purposes.

We fix in this section a countable *set of resources* $\mathcal{R}$. The *alphabet* is the set of multi-sets of resources $\mathbb{A} = \mathcal{R} \to \overline{\mathbb{R}}_+$. We define $0, \infty \in \mathbb{A}$ by $0(\alpha) = 0$ and $\infty(\alpha) = \infty$ for all $\alpha \in \mathcal{R}$. The *set of actions* is $\mathbb{A}_{\mathcal{R}} = \mathbb{A} \setminus \{0\}$.

The multiplicity attached by an action to a resource measures for instance the time consumed. The notion of *time* may just as well be replaced by *amount*, while the term *consumed* may be replaced by *produced*. If, for example, $\mathcal{R} = \{A, B, C\}$ then an action consuming respectively 3.5, 5.7 and 7.3 time units of A, B and C is denoted by the multi-set $3.5A + 5.7B + 7.3C$. Concrete examples might come from computer science (an action consuming 5 processor, 10 channel and 2 memory time units) or workflow management (an action consuming 100 man, 5 tool and 10 object time units).

On $\mathbb{A}$ we define componentwise the order $\leq \subseteq \mathbb{A} \times \mathbb{A}$, the complement $^- : \mathbb{A} \to \mathbb{A}$, the sum $+ : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$, the infimum $\wedge : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$, the supremum $\vee : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ and the skew difference $\setminus : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$, whereby for all $n, m \in \overline{\mathbb{R}}_+$ we set $\overline{n} = \infty$ if $n = 0$, $\overline{n} = 0$ if $n \neq 0$, $n \setminus m = \infty$ if $n = \infty$, $n \setminus m = 0$ if $n \neq \infty$ and $m = \infty$, $n \setminus m = (n - m) \vee 0$ if $n, m \neq \infty$.

For $a, b \in \mathbb{A}$ such that $b \leq a$ let $a - b = a \setminus b$. For $a, b \in \mathbb{A}$ we define the *independence* $a \perp b$ iff $a \wedge b = 0$.

For $a \in \mathbb{A} = \mathcal{R} \to \overline{\mathbb{R}}_+$ and $S \in \mathcal{R} \times \mathcal{R} \to \overline{\mathbb{R}}_+$ we set $S(a) = aS$ as a vector-matrix multiplication. The *set of renamings* $\mathbb{S}_\mathcal{R} \subseteq \mathcal{R} \times \mathcal{R} \to \overline{\mathbb{R}}_+$ is defined by $S \in \mathbb{S}_\mathcal{R}$ iff $(a \neq 0 \implies S(a) \neq 0$ for all $a \in \mathbb{A})$ and $(a \perp b \implies S(a) \perp S(b)$ for all $a, b \in \mathbb{A})$.

3.1 The Domain of Real Multi-Pomsets $\mathbb{R}$

A *real multi-labelled partial order* is a triple $(E, \preceq, \rho)$, where

(i) the *synchronization relation* $\preceq \subseteq E \times E$ is a partial order on the *set of events* E satisfying the *past finiteness* condition that $\{ f \in E \mid f \preceq e \}$ is finite for each $e \in E$,

(ii) the *event-labelling* $\rho : E \to \mathbb{A}_\mathcal{R}$ satisfies the *over-synchronization* condition that $\rho(e) \wedge \rho(f) \neq 0 \implies e \preceq f$ or $f \preceq e$ for each $e, f \in E$.

A *real multi-pomset* is the isomorphism class $[E, \preceq, \rho]$ of a real multi-labelled partial order $(E, \preceq, \rho)$. The *set of real multi-pomsets* is denoted by $\mathbb{R}$. A *finite multi-pomset* is a multi-pomset whose event set is finite. The *set of finite multi-pomsets* is denoted by $\mathbb{F}$. The *empty multi-pomset* is $0 = [\emptyset, \emptyset, \emptyset] \in \mathbb{R}$. For all $a \in \mathbb{A}_\mathcal{R}$ we define the *action multi-pomset* $a = [(\{\emptyset\}, \{(\emptyset, \emptyset)\}, \{(\emptyset, a)\})] \in \mathbb{R}$.

The synchronization relation reflects the temporal (or causal) order between the events of the multi-pomset. The past finiteness condition is a technical assumption that restricts the definition to real multi-pomsets, but can be relaxed if one wishes to deal with transfinite multi-pomsets.

The over-synchronization condition is equivalent to the fact that for each resource the events consuming it are *sequentialized* (totally ordered). In particular it implies that the multi-pomset has no auto-concurrency, that is, for all $a \in \mathbb{A}_\mathcal{R}$ the set $\rho^{-1}(a) = \{ e \in E \mid \rho(e) = a \}$ is totally ordered by $\preceq$. The *number of occurrences* $|\ |_a : \mathbb{R} \to \omega + 1$ of $a \in \mathbb{A}_\mathcal{R}$ is defined for all $x \in \mathbb{R}$ by $|x|_a = \mathrm{ord}(\rho^{-1}(a), \preceq \cap \rho^{-1}(a) \times \rho^{-1}(a)) \leq \omega$, that is the ordinal associated with the well-order induced by $\preceq$ on $\rho^{-1}(a)$.

In order to avoid cumbersome isomorphism proofs we define the *standard representative* of $x = (E, \preceq, \rho)$ as the unique isomorphic real multi-labelled partial order $\hat{x} = (E_x, \preceq_x, \rho_x)$, such that

(i) $E_x = \phi_x(E) = \{ (a, n) \mid a \in \mathbb{A}_\mathcal{R}$ and $n < |x|_a \} \subseteq \mathbb{A}_\mathcal{R} \times \omega = \mathbb{E}$,

(ii) $(a, n) \preceq_x (a, m)$ for all $a \in \mathbb{A}_\mathcal{R}$ and $n \leq m < |x|_a$

(iii) $\rho_x(a, n) = a$ for all $a \in \mathbb{A}_\mathcal{R}$ and $n < |x|_a$.

The *past* in $x \in \mathbb{R}$ of $F \subseteq \mathbb{E}$ is $\downarrow_x F = \{ e \in E_x \mid \exists f : e \preceq_x f \in F \}$. The *restriction* of $x \in \mathbb{R}$ to $F \subseteq \mathbb{E}$ is $x/F = [E_x \cap F, \preceq_x \cap F \times F, \rho_x \cap F \times \mathbb{A}_\mathcal{R}]$. The *prefix* is defined for all $x, y \in \mathbb{R}$ by $x \leq y$ iff $E_x = \downarrow_y E_x$ and $x = y/E_x$.

The next theorem shows that $(\mathbb{R}, \leq)$ is an interesting semantic domain.

Theorem 3.1 $(\mathbb{R}, \leq)$ *is a p-algebraic CCPO, hence, it is a (Scott-)domain.*

The *alphabet* $\mathrm{alph} : \mathbb{R} \to \mathcal{P}(\mathbb{A})$ is defined for all $x \in \mathbb{R}$ by $\mathrm{alph}(x) = \{\, \rho_x(e) \mid e \in E_x \,\}$. The *consumption* $\mathrm{cons} : \mathbb{R} \to \mathbb{A}$ is defined for all $x \in \mathbb{R}$ by $\mathrm{cons}(x) = \sum_{e \in E_x} \rho_x(e)$. It can be shown that $\mathrm{cons} : (\mathbb{R}, \leq) \to (\mathbb{A}, \leq)$ is continuous. For $x, y \in \mathbb{R}$ we define the *independence* $x \perp y$ iff $\mathrm{cons}(x) \perp \mathrm{cons}(y)$. The *infinite consumption* $\mathrm{consinf} : \mathbb{R} \to \mathbb{A}$ is defined for all $x \in \mathbb{R}$ and $\alpha \in \mathcal{R}$ by $\mathrm{consinf}(x)(\alpha) = \infty$ if $\mathrm{cons}(x)(\alpha) = \infty$ and $\mathrm{consinf}(x)(\alpha) = 0$ otherwise.

We need later on the following definitions. For $a \in \mathbb{A}_{\mathcal{R}}$ and $x \in \mathbb{R}$ satisfying the *action prefix* $a \leq x$ let the *action residue* be $a^{-1}x = x/(E_x \setminus \{(a, 0)\})$. Let 0 denote the *empty string* in $\mathbb{A}^*_{\mathcal{R}}$. For $u \in \mathbb{A}^*_{\mathcal{R}}$ and $x \in \mathbb{R}$ we define the *linear prefix* $u \trianglelefteq x$ and the *linear residue* $u^{-1}x$ inductively on u by:

- Let $0 \trianglelefteq x$ and $0^{-1}x = x$,
- For $a \in \mathbb{A}_{\mathcal{R}}$ let $ua \trianglelefteq x$ iff $u \trianglelefteq x$, $a \leq u^{-1}x$, and let $(ua)^{-1}x = a^{-1}(u^{-1}x)$.

The *set of linearizations* of r is $\mathrm{Lin}(x) = \{\, u \in \mathbb{A}^*_{\mathcal{R}} \mid u \trianglelefteq x, u^{-1}x = 0 \,\}$.

We note that for any $a \in \mathbb{A}_{\mathcal{R}} \subseteq \mathbb{R}$ and $x \in \mathbb{R}$ we have $a \trianglelefteq x$ iff $a \leq x$ iff x contains a minimal event labelled with a, so in this case the prefix and linear prefix relation coincide.

The main deficiency of $(\mathbb{R}, \leq)$ is the fact that it is unsuitable to define continuous denotations for the operators of our process language, since for example the sequential composition is not monotone (hence, not continuous). This is indeed not surprising, since already the sequential composition (catenation) of strings is not monotone with respect to the prefix order, as can be easily seen on the example $a \leq a; a'$ and $b \leq b; b'$ but $a; b \not\leq (a; a'); (b; b')$ unless a' is empty.

3.2 The Domain of Complex Multi-Pomsets $\mathbb{C}$

We surmount the above obstacle by introducing the domain of complex multi-pomsets.

A *complex multi-pomset* is a pair $x = (r, R)$, where $r \in \mathbb{R}$ is a real multi-pomset, and $R \in \mathbb{A}$ is a multi-set of resources, such that $\mathrm{cons}(r) \leq R$. The *set of complex multi-pomsets* is denoted by $\mathbb{C}$. The multi-pomset r is denoted by $\mathrm{Re}(x)$ and called the *real part* of x. The multi-set R is denoted by $\mathrm{cons}(x)$ and called the *consumption part* of x.

The *imaginary part* of $x \in \mathbb{C}$ is $\mathrm{Im}(x) = \mathrm{cons}(x) - \mathrm{cons}(\mathrm{Re}(x))$. If its imaginary part $\mathrm{Im}(x)$ is zero, the complex multi-pomset x is called *terminated*. Note that, due to the convention regarding ∞ and the difference operator, we have $\mathrm{consinf}(\mathrm{Re}(x)) \leq \mathrm{Im}(x)$.

The first component of a complex multi-pomset is a real multi-pomset describing the observed part of the process, while the second component is a multi-set of resources representing the quota actually consumed by the process during its execution.

The *prefix* is defined for all $(r, R), (s, S) \in \mathbb{C}$ by $(r, R) \leq (s, S) \Leftrightarrow r \leq s$ and $R = S$. The underlying idea here is that we increase the information about a process by letting grow its observable part $r \leq s$, while preserving the quota $R = S$

that may be consumed during the execution.

The next theorem shows that $(\mathbb{C}, \leq)$ is a suitable semantic domain.

Theorem 3.2 *$(\mathbb{C}, \leq)$ is a p-algebraic CCPO, hence, it is a (Scott-)domain.*

We need later on the following definitions. For $u \in \mathbb{A}^*_{\mathcal{R}}$ and $x \in \mathbb{C}$ we extend the *linear prefix* by $u \trianglelefteq x$ iff $u \trianglelefteq \mathrm{Re}(x)$ and the *linear residue* by $u^{-1}x = (u^{-1}\,\mathrm{Re}(x), \mathrm{cons}(x) - \mathrm{cons}(u))$.

The main virtue of $(\mathbb{C}, \leq)$ is the fact that it allows defining internal and continuous denotations for all process operators of our language.

4 The Process Language

We fix in the following a countable *set of constants* $\mathcal{C}$ and a disjoint countable *set of variables* $\mathcal{V}$. The *set of resources* is $\mathcal{R} = \mathcal{C} \cup \mathcal{V}$. The *set of constant actions* is $\mathbb{A}_{\mathcal{C}} = (\mathcal{C} \to \overline{\mathbb{N}}) \setminus \{0\} \subseteq \mathbb{A}_{\mathcal{R}}$.

The *language of terms* $\mathcal{L}$ is generated by the following BNF-style grammar

$$p \;::=\; \mathrm{SKIP} \mid c \mid S(p) \mid p \circledcirc T \mid p \oslash T \mid p \cdot p \mid p \underset{C}{\parallel} p \mid x \mid \mathrm{rec}\, x.p$$

for all $c \in \mathbb{A}_{\mathcal{C}}$, $S \in \mathbb{S}_{\mathcal{R}}$, $T \in \mathbb{A}$, $C \subseteq \mathbb{A}_{\mathcal{R}}$ and $x \in \mathcal{V}$.

Here, SKIP is the *empty process*, c is a *constant action*, $S()$ is *renaming* through S, $\circledcirc T$ is *hiding* of the consumption T, $\oslash T$ is *restriction* to the consumption T, $\cdot$ is *serial composition*, $\underset{C}{\parallel}$ is *parallel composition* synchronized on the channels in C, $x \in \mathcal{V}$ is a *variable* and $\mathrm{rec}\, x.$ is *recursion* over x.

The *language of closed terms* $\mathcal{L}_c$ is the set of terms without free variables.

The *consumption* $\mathrm{Cons}(p) : \mathbb{A}^{\mathcal{V}} \to \mathbb{A}$ of a process term $p \in \mathcal{L}$ is inductively defined for all $\tau \in \mathbb{A}^{\mathcal{V}}$ by

$$\begin{aligned}
\mathrm{Cons}(\mathrm{SKIP})(\tau) &= 0\\
\mathrm{Cons}(c)(\tau) &= c\\
\mathrm{Cons}(S(p))(\tau) &= S(\mathrm{Cons}(p)(\tau))\\
\mathrm{Cons}(p \circledcirc T)(\tau) &= \mathrm{Cons}(p)(\tau) \setminus T\\
\mathrm{Cons}(p \oslash T)(\tau) &= \mathrm{Cons}(p)(\tau) \wedge T\\
\mathrm{Cons}(p \cdot q)(\tau) &= \mathrm{Cons}(p)(\tau) + \mathrm{Cons}(q)(\tau)\\
\mathrm{Cons}(p \underset{C}{\parallel} q)(\tau) &= \mathrm{Cons}(p)(\tau) \vee \mathrm{Cons}(q)(\tau)\\
\mathrm{Cons}(x)(\tau) &= \tau(x)\\
\mathrm{Cons}(\mathrm{rec}\, x.p)(\tau) &= \mathrm{lfp}_{\leq} R.(\{x\} + \mathrm{Cons}(p)(\tau[x \mapsto R]))
\end{aligned}$$

For $\tau \in \mathbb{A}^{\mathcal{V}}$ we define $\tau[x \mapsto R]$ to be identical to τ on all arguments except x that is assigned the value R. The characteristic mapping $\{x\} \in \mathbb{A}$ is defined for all $\alpha \in \mathcal{R}$ by $\{x\}(\alpha) = 0$ for $\alpha \neq x$ and $\{x\}(\alpha) = 1$ for $\alpha = x$. One may show by structural induction over $p \in \mathcal{L}$ that $\mathrm{Cons}(p) : (\mathbb{A}, \leq)^{\mathcal{V}} \to (\mathbb{A}, \leq)$ is continuous. Thus, the above definition determines a compositional consumption semantics $\mathrm{Cons} : \mathcal{L} \to ((\mathbb{A}, \leq)^{\mathcal{V}} \to (\mathbb{A}, \leq))$.

5 The Operational Semantics

Table 1 presents the transition rules of our deterministic structural operational machine, whereby we let $c \in \mathbb{A}_{\mathcal{C}}$, $T \in \mathbb{A}$, $C \subseteq \mathbb{A}_{\mathcal{R}}$, $x \in \mathcal{V}$, $p, p', q, q' \in \mathcal{L}$, $a \in \mathbb{A}_{\mathcal{R}} \cup \{0\}$, $\tau \in \mathbb{A}^{\mathcal{V}}$. As usual, $p[q/x]$ denotes the term that is obtained from p after substituting all occurrences of the variable x by q. Note that recursion is modelled in an observable way, each unwinding producing as observation the variable being recursed, which may subsequently be renamed or hidden.

For $u \in \mathbb{A}^*_{\mathcal{R}}$ and $p, p' \in \mathcal{L}$, $\tau \in \mathbb{A}^{\mathcal{V}}$, let the *linear transition* $p \overset{u}{\underset{\tau}{\Longrightarrow}} p'$ be inductively defined on the length of u by

- Let $p \overset{0}{\underset{\tau}{\Longrightarrow}} p'$ iff $p = p'$,
- For $a \in \mathbb{A}_{\mathcal{R}}$ let $p \overset{ua}{\underset{\tau}{\Longrightarrow}} p'$ iff there exists $q \in \mathcal{L}$ such that $p \overset{u}{\underset{\tau}{\Longrightarrow}} q$ and $q \overset{a}{\underset{\tau}{\longrightarrow}} p'$

Using the linear transition we next define a linear and a complex operational semantics of closed process terms as follows.

The *linear behaviour* of $p \in \mathcal{L}_c$ is $\mathbb{A}^*_{\mathcal{R}}(p) = \{\, u \in \mathbb{A}^*_{\mathcal{R}} \,|\, p \overset{u}{\Longrightarrow} \,\}$.

For any fixed $E \subseteq \mathbb{E}$ the *intersection* $\bigcap P$ of a set $P \subseteq \mathbb{R}$ such that for all $r \in P$ we have $E_r = E$ is defined by $E_{\bigcap P} = E$ and $\leq_{\bigcap P} = \bigcap \{\, \leq_r \;| r \in P \,\}$. The *restriction* of $p \in \mathcal{L}_c$ to E is $p/E = \bigcap \{\, u \in \mathbb{A}^*_{\mathcal{R}} \,|\, p \overset{u}{\Longrightarrow} \text{ and } E_u = E \,\}$.

The *complex behaviour* of $p \in \mathcal{L}_c$ is $\mathbb{C}(p) = \{\, (p/E_u, \mathrm{Cons}(p)) \,|\, p \overset{u}{\Longrightarrow} \,\}$.

$$[\mathrm{ACT}]\ \frac{}{c \overset{c}{\underset{\tau}{\longrightarrow}} \mathrm{SKIP}} \qquad [\mathrm{REN}]\ \frac{p \overset{a}{\underset{\tau}{\longrightarrow}} p'}{S(p) \overset{S(a)}{\underset{\tau}{\longrightarrow}} S(p')}$$

$$[\mathrm{HID}]\ \frac{p \overset{a}{\underset{\tau}{\longrightarrow}} p'}{p \oslash T \overset{a \setminus T}{\underset{\tau}{\longrightarrow}} p' \oslash (T \setminus a)} \qquad [\mathrm{RES}]\ \frac{p \overset{a}{\underset{\tau}{\longrightarrow}} p',\ a \leq T}{p \oslash T \overset{a}{\underset{\tau}{\longrightarrow}} p' \oslash (T - a)}$$

$$[\mathrm{REC}]\ \frac{\mathrm{rec}\, x.p = q}{q \overset{\{x\}}{\underset{\tau}{\longrightarrow}} p[q/x]}$$

$$[\mathrm{SER}_1]\ \frac{p \overset{a}{\underset{\tau}{\longrightarrow}} p'}{p \cdot q \overset{a}{\underset{\tau}{\longrightarrow}} p' \cdot q} \qquad [\mathrm{SER}_2]\ \frac{\mathrm{Cons}(p)(\tau) \perp a,\ q \overset{a}{\underset{\tau}{\longrightarrow}} q'}{p \cdot q \overset{a}{\underset{\tau}{\longrightarrow}} p \cdot q'}$$

$$[\mathrm{PAR}_0]\ \frac{a \in C,\ p \overset{a}{\underset{\tau}{\longrightarrow}} p',\ q \overset{a}{\underset{\tau}{\longrightarrow}} q'}{p \underset{C}{\parallel} q \overset{a}{\underset{\tau}{\longrightarrow}} p' \underset{C}{\parallel} q'} \qquad [\mathrm{PAR}_1]\ \frac{C, \mathrm{Cons}(q)(\tau) \perp a,\ p \overset{a}{\underset{\tau}{\longrightarrow}} p'}{p \underset{C}{\parallel} q \overset{a}{\underset{\tau}{\longrightarrow}} p' \underset{C}{\parallel} q}$$

$$[\mathrm{PAR}_2]\ \frac{C, \mathrm{Cons}(p)(\tau) \perp a,\ q \overset{a}{\underset{\tau}{\longrightarrow}} q'}{p \underset{C}{\parallel} q \overset{a}{\underset{\tau}{\longrightarrow}} p \underset{C}{\parallel} q'}$$

Table 1

6 The Denotational Semantics

We next construct a denotational semantics for our process language using a functional domain over environments of complex multi-pomsets.

We endow the *set of environments* $\mathbb{C}^{\mathcal{V}}$ with the product order $\leq$, that is, we set $(\mathbb{C}^{\mathcal{V}}, \leq) = (\mathbb{C}, \leq)^{\mathcal{V}}$. $\mathrm{cons}^{\mathcal{V}} : \mathbb{C}^{\mathcal{V}} \to \mathbb{A}^{\mathcal{V}}$ is defined for all environments $\sigma \in \mathbb{C}^{\mathcal{V}}$ and $x \in \mathcal{V}$ by $\mathrm{cons}^{\mathcal{V}}(\sigma)(x) = \mathrm{cons}(\sigma(x))$. We have a canonical embedding $\mathbb{A} \hookrightarrow \mathbb{C}$, $R \mapsto (0, R)$, which allows identifying $\mathbb{A}$ with its image in $\mathbb{C}$ and set $\mathbb{A} \subseteq \mathbb{C}$. Therefore, any function on $\mathbb{C}^{\mathcal{V}}$ is a function on $\mathbb{A}^{\mathcal{V}}$.

The *functional domain* $\mathbb{D}$ is the set of mappings $f : \mathbb{C}^{\mathcal{V}} \to \mathbb{C}$ satisfying the following *functional conditions*

(i) *consumption commutation*: $\mathrm{cons}(f(\sigma)) = f(\mathrm{cons}^{\mathcal{V}}(\sigma))$ for all $\sigma \in \mathbb{C}^{\mathcal{V}}$,

(ii) *consumption continuity*: $f : (\mathbb{A}, \leq)^{\mathcal{V}} \to (\mathbb{A}, \leq)$ is continuous,

(iii) *complex continuity*: $f : (\mathbb{C}, \leq)^{\mathcal{V}} \to (\mathbb{C}, \leq)$ is continuous.

We pointwise lift the ordering $\leq$ from $\mathbb{C}$ to $\mathbb{D}$, that is, for $f, g \in \mathbb{D}$ we define $f \leq g$ iff $f(\sigma) \leq g(\sigma)$ for all $\sigma \in \mathbb{C}^{\mathcal{V}}$.

Proposition 6.1 *$\mathbb{D}$ is closed by substitution and $(\mathbb{D}, \leq)$ is a DCPO.*

The denotational semantics $[\![]\!] : \mathcal{L} \to \mathbb{D}$ inductively defined below is uniquely determined by the denotation which will be defined in the next subsections for each *finitary operator* symbol as an operation on $\mathbb{C}$ having the same arity and satisfying the functional conditions of $\mathbb{D}$ (that is, we indeed have $[\![\mathrm{SKIP}]\!]$, $[\![c]\!]$, $[\![S(p)]\!]$, $[\![p \diamond T]\!]$, $[\![p \oslash T]\!]$, $[\![p \cdot q]\!]$, $[\![p \underset{C}{\parallel} q]\!] \in \mathbb{D}$ for $p, q \in \mathcal{L}$) and the denotation which will be defined for the *infinitary recursion operator* symbol as an operation on $\mathbb{D}$ (that is, we indeed have $[\![\mathrm{rec}\, x.p]\!] \in \mathbb{D}$ for $p \in \mathcal{L}$). Hereby, consumption commutation and continuity are straightforward to check whereas complex continuity is increasingly difficult to prove. From the above we may thus in advance state the following

Theorem 6.2 *The denotational semantics $[\![]\!] : \mathcal{L} \to \mathbb{D}$ inductively defined by*

$$
\begin{aligned}
[\![\mathrm{SKIP}]\!](\sigma) &= (0,0) & [\![p \cdot q]\!](\sigma) &= [\![p]\!](\sigma) \cdot [\![q]\!](\sigma) \\
[\![c]\!](\sigma) &= (c,c) & [\![p \underset{C}{\parallel} q]\!](\sigma) &= [\![p]\!](\sigma) \underset{C}{\parallel} [\![q]\!](\sigma) \\
[\![S(p)]\!](\sigma) &= S([\![p]\!](\sigma)) & [\![x]\!](\sigma) &= \sigma(x) \\
[\![p \diamond T]\!](\sigma) &= [\![p]\!](\sigma) \diamond T & [\![\mathrm{rec}\, x.p]\!](\sigma) &= (\mathrm{rec}\, x.[\![p]\!])(\sigma) \\
[\![p \oslash T]\!](\sigma) &= [\![p]\!](\sigma) \oslash T
\end{aligned}
$$

is well-defined, that is $[\![p]\!] \in \mathbb{D}$ for all $p \in \mathcal{L}$.

Directly on the denotation of each of the operators SKIP, c, $S()$, $\diamond T$, $\oslash T$, $\cdot$, $\underset{C}{\parallel}$ and $\mathrm{rec}\, x.$ we will also be able to inductively check the following

Proposition 6.3 *If $p \in \mathcal{L}$, $\sigma \in \mathbb{C}^{\mathcal{V}}$ then* $\mathrm{cons}([\![p]\!](\sigma)) = \mathrm{Cons}(p)(\mathrm{cons}^{\mathcal{V}}(\sigma))$.

We now proceed with the denotation of the operators of our language.

6.1 The Renaming Operator $S(x)$

Renaming amounts to a simple relabeling which preserves the events and their initial ordering. Simple as it may seem it nevertheless allows linear computations to take place on the labels of the events and may, therefore, be used to derive further operators from those of our language.

For every $S \in \mathbb{S}_{\mathcal{R}}$ the *renaming* operator $S() : \mathbb{R} \to \mathbb{R}$ is defined by $S([E, \preceq, \rho]) = [E, \preceq, S(\rho)]$, whereby $S(\rho)(e) = S(\rho(e))$ for all $e \in E$. For every $S \in \mathbb{S}_{\mathcal{R}}$ the *renaming* operator $S() : \mathbb{C} \to \mathbb{C}$ is defined for all $x \in \mathbb{C}$ by $S(x) = (S(\mathrm{Re}(x)), S(\mathrm{cons}(x)))$.

6.2 The Hiding Operator $x \oslash T$

The hiding operator allows to internalize some given quota of resources and prevents other processes from synchronizing on events that make use of them. As usually, this expresses the need for local, as opposed to global, computation and communication.

For every $T \in \mathbb{A}$ the *hiding* operator $\oslash T : \mathbb{R} \to \mathbb{R}$ is defined for all $[E, \preceq, \rho] \in \mathbb{R}$ by $[E, \preceq, \rho] \oslash T = [E', \preceq', \rho']$ where

(i) $\rho'(e) = \rho(e) \setminus (T \setminus \sum_{f \prec e} \rho'(f))$,

(ii) $E' = \{\, e \in E \mid \rho'(e) \neq 0 \,\}$,

(iii) $\preceq' = \preceq \cap E' \times E'$.

For every $T \in \mathbb{A}$ the *hiding* operator $\oslash T : \mathbb{C} \to \mathbb{C}$ is defined for all $x \in \mathbb{C}$ by $x \oslash T = (\mathrm{Re}(x) \oslash T, \mathrm{cons}(x) \setminus T)$.

The hiding $x \oslash T$ erases a given additive consumption T out of the past of each event of x, thereby rendering it unobservable.

6.3 The Restriction Operator $x \oslash T$

The restriction operator blocks a process on all but some given quota of resources. This is used for instance in order to assure confinement of that process to a certain safe environment and may be useful in security protocols.

For every $T \in \mathbb{A}$ the *restriction* operator $\oslash T : \mathbb{R} \to \mathbb{R}$ is defined for all $x \in \mathbb{R}$ by $x \oslash T = \bigvee \{\, y \in \mathbb{R} \mid y \leq x, \mathrm{cons}(y) \leq T \,\}$. For every $T \in \mathbb{A}$ the *restriction* operator $\oslash T : \mathbb{C} \to \mathbb{C}$ is defined for all $x \in \mathbb{C}$ by $x \oslash T = (\mathrm{Re}(x) \oslash T, \mathrm{cons}(x) \wedge T)$.

One can easily see that $x \oslash T$ is the restriction of x to the set of events having a past that additively consumes resources below T, that is, we have $x \oslash T = x/E'_x$ where $E'_x = \{\, e \in E_x \mid \sum_{f \preceq_x e} \rho_x(f) \leq T \,\}$.

6.4 The Serial Composition $x \cdot y$

The following presentation of the serial composition is a generalization of the concatenation treated in [5]. The serial composition enforces synchronizations between the first and the second process only to prevent races for resources. Events at the end of the first and events at the beginning of the second process can thus occur

concurrently if they are independent. This construct may be of interest in automatic code parallelization and in transactional systems.

The *serial composition* $\cdot : \mathbb{R}^2 \to \mathbb{R}$ is defined for all $x_1 = [E_1, \preceq_1, \rho_1] \in \mathbb{R}$ and $x_2 = [E_2, \preceq_2, \rho_2] \in \mathbb{R}$ if $\mathrm{consinf}(x_1) \wedge \mathrm{cons}(x_2) = 0$ by $x_1 \cdot x_2 = [E, \preceq, \rho]$, where

(i) $E = E_1 \dot\cup E_2$,

(ii) $\preceq = (\preceq_1 \dot\cup \{\, (e_1, e_2) \in E_1 \times E_2 \mid \rho_1(e_1) \wedge \rho_2(e_2) \neq 0 \,\} \dot\cup \preceq_2)^*$,

(iii) $\rho = \rho_1 \dot\cup \rho_2$.

The *serial composition* $\cdot : \mathbb{C}^2 \to \mathbb{C}$ is defined for all $x, y \in \mathbb{C}$ by $x \cdot y = (\mathrm{Re}(x) \cdot (\mathrm{Re}(y) \oslash \overline{\mathrm{Im}(x)}), \mathrm{cons}(x) + \mathrm{cons}(y))$.

It can be shown that using the serial composition together with hidings enables us to denote all compact multi-pomsets by closed terms of the process language, which means that the denotational semantics is optimal.

6.5 The Sequential Composition $x\,;\,y$

The sequential composition should be employed whenever there is a need to temporally completely synchronize two processes. The compound processes are scheduled such that the entire first process occurs before the entire second process, hence they are temporally ordered, even if independent.

The *sequential composition* $; : \mathbb{R}^2 \to \mathbb{R}$ is defined for $x_1 = [E_1, \preceq_1, \rho_1] \in \mathbb{F}$ and $x_2 = [E_2, \preceq_2, \rho_2] \in \mathbb{R}$ by $x_1 \,;\, x_2 = [E_1 \dot\cup E_2, (\preceq_1 \dot\cup E_1 \times E_2 \dot\cup \preceq_2)^*, \rho_1 \dot\cup \rho_2]$. The *sequential composition* $; : \mathbb{C}^2 \to \mathbb{C}$ is defined for all $x, y \in \mathbb{C}$ by

$$x \,;\, y = \begin{cases} (\mathrm{Re}(x) \,;\, \mathrm{Re}(y), \mathrm{cons}(x) + \mathrm{cons}(y)) & \text{if } \mathrm{Im}(x) = 0 \\ (\mathrm{Re}(x), \mathrm{cons}(x) + \mathrm{cons}(y)) & \text{otherwise.} \end{cases}$$

It can be shown that the sequential composition can be expressed as a renaming of a the serial composition of renamings of the compound processes, hence it is a derived operator of our process language. We shall later on essentially use the sequential composition in order to define the denotational semantics of recursion.

6.6 The Parallel Composition $x \underset{C}{\parallel} y$

The following presentation of the parallel composition is a generalization of the one treated in [4]. The parallel composition is indexed by a set of channels that processes are supposed to employ in order to synchronize. Events accessing the channels are commonly processed by the compound processes, while events which make no use of the channels may be independently processed by each compound process. This construct may in particular be used to model data-parallel programs running on PRAMs.

Let $x_1 = [E_1, \preceq_1, \rho_1], x_2 = [E_2, \preceq_2, \rho_2] \in \mathbb{R}$ be in standard representation. We define their *parallel composition* by $x_1 \| x_2 = [E_1 \cup E_2, (\preceq_1 \cup \preceq_2)^*, \rho_1 \cup \rho_2]$. Note that $x_1 \| x_2$ may fail to be a real multi-pomset for two different reasons. First, $\preceq$ may fail to be antisymmetric. This is the case for instance if $x_1 = a\,;b$ and $x_2 = b\,;a$. Second, $\preceq$ may fail to be over-synchronized. This is the case for instance if $x_1 = a$

and $x_2 = b$ with $\neg(a \perp b)$.

Let $x_1, x_2 \in \mathbb{C}$ and $C \subseteq \mathbb{A}_{\mathcal{R}}$. We define $(r_1, r_2) \in \mathbb{R}_C(x_1, x_2) \subseteq \mathbb{R}^2$ iff

(i) for all $i \in \{1, 2\}$ we have $r_i \leq \mathrm{Re}(x_i)$,

(ii) $|r_1|_a = |r_2|_a$ for all $a \in C$,

(iii) for all $\{i, j\} = \{1, 2\}$ and $a \in \mathrm{alph}(r_i)$ we have $a \in C$ or $(a \perp C$ and $a \perp x_j)$,

(iv) $r_1 \| r_2 \in \mathbb{R}$.

The *parallel composition* $\underset{C}{\|} : \mathbb{C}^2 \to \mathbb{C}$ is defined for all $x_1, x_2 \in \mathbb{C}$ and $C \subseteq \mathbb{A}_{\mathcal{R}}$ by $x_1 \underset{C}{\|} x_2 = (r_1 \| r_2, \mathrm{cons}(x_1) \vee \mathrm{cons}(x_2))$ where $(r_1, r_2) = \bigvee \mathbb{R}_C(x_1, x_2)$.

6.7 The Recursion Operator $\mathrm{rec}\, x.f$

The denotation of the recursion operator essentially differs from the least fixed point semantics. Indeed, the recursion operator $\mathrm{rec}\, x. : \mathbb{D} \to \mathbb{D}$, $f \mapsto \mathrm{rec}\, x.f$ is defined by computing $(\mathrm{rec}\, x.f)(\sigma)$ for all $f \in \mathbb{D}$ and $\sigma \in \mathbb{C}^{\mathcal{V}}$ using two chained fixed point computations. Firstly, we compute the least consumption of a fixed point by solving the recursion in the consumption domain $(\mathbb{A}, \leq)$ and, secondly, we compute the least fixed point having the least consumption by solving the recursion in the complex domain $(\mathbb{C}, \leq)$, as follows.

For $\sigma \in \mathbb{A}^{\mathcal{V}}$ we define as usual $\sigma[x \mapsto y]$ to be identical to σ on all arguments except x, which is assigned the value y. For $x \in \mathcal{V}$ we define $\mathbb{C}_x : \mathbb{D} \times \mathbb{C}^{\mathcal{V}} \times \mathbb{C} \to \mathbb{C}$ by $C_x(f, \sigma, y) = \{x\} ; f(\sigma[x \mapsto y])$, and $A_x : \mathbb{D} \times \mathbb{C}^{\mathcal{V}} \times \mathbb{A} \to \mathbb{A}$ by $A_x(f, \sigma, R) = \{x\} + f(\mathrm{cons}^{\mathcal{V}}(\sigma)[x \mapsto R]) = \mathrm{cons}(C_x(f, \sigma, R))$.

Firstly, we start with $R_0 = 0$ as lower bound and iterate the consumption mapping $A_x(f, \sigma) : (\mathbb{A}, \leq) \to (\mathbb{A}, \leq)$, which delivers the fixed point $x_0(f, \sigma) = \bigvee_{n<\omega} R_n(f, \sigma)$ where $R_{n+1}(f, \sigma) = A_x(f, \sigma, R_n(f, \sigma))$ for all $n < \omega$. Since $A_x(f, \sigma) : (\mathbb{A}, \leq) \to (\mathbb{A}, \leq)$ can be easily shown to be continuous for each $x \in \mathcal{V}$ and R_0 is a prefixed point, it follows that the sequence $R_n(f, \sigma)$ is increasing in the DCPO $(\mathbb{A}, \leq)$, hence the last supremum indeed exists.

Secondly, we start with $x_0(f, \sigma)$ as lower bound and iterate the complex mapping $C_x(f, \sigma) : (\mathbb{C}, \leq) \to (\mathbb{C}, \leq)$, which delivers the fixed point $(\mathrm{rec}\, x.f)(\sigma) = \bigvee_{n<\omega} x_n(f, \sigma)$ where $x_{n+1}(f, \sigma) = C_x(f, \sigma, x_n(f, \sigma))$ for all $n < \omega$. Since $C_x(f, \sigma) : (\mathbb{C}, \leq) \to (\mathbb{C}, \leq)$ can be easily shown to be continuous for each $x \in \mathcal{V}$ and $x_0(f, \sigma)$ is a prefixed point, it follows that the sequence $x_n(f, \sigma)$ is increasing in the DCPO $(\mathbb{C}, \leq)$, hence the last supremum indeed exists.

Finally, we show that $\mathrm{rec}\, x. : \mathbb{D} \to \mathbb{D}$ is well-defined, that is, $\mathrm{rec}\, x.f \in \mathbb{D}$ for all $f \in \mathbb{D}$, and moreover that $\mathrm{rec}\, x. : (\mathbb{D}, \leq) \to (\mathbb{D}, \leq)$ is continuous.

7 Congruence and Full Abstraction

We now arrive at the main results of the paper which require generalizing the arguments and constructions presented in [4]. We shall state in this section two results of full abstraction, a linear and a complex one. To this purpose we first

exhibit *linear translations* to and fro between linear transition on the operational side and linear prefix and residue on the denotational side.

The following hard technical lemma concerns the interaction of the operators with the action residue. Note the close resemblance of the denotational Table 2 to the operational Table 1. For all $a \in \mathbb{A}_{\mathcal{R}}$, $x, x' \in \mathbb{C}$ if $a^{-1}x = x'$ then we suppose in particular that $a \trianglelefteq x$.

Lemma 7.1 *For any $a \in \mathbb{A}_{\mathcal{R}} \cup \{0\}$, $x, x', y, y' \in \mathbb{C}$, $c \in \mathbb{A}_{\mathcal{C}}$, $T \in \mathbb{A}$, $C \subseteq \mathbb{A}_{\mathcal{R}}$, $f \in \mathbb{D}$, $\sigma \in \mathbb{C}^{\mathcal{V}}$ we have the properties of Table 2.*

[ACT]	$\dfrac{}{c^{-1}c() = 0}$	[SER$_1$]	$\dfrac{a^{-1}x = x'}{a^{-1}(x \cdot y) = x' \cdot y}$
[REN]	$\dfrac{a^{-1}x = x'}{S(a)^{-1}S(x) = S(x')}$	[SER$_2$]	$\dfrac{\mathrm{cons}(x) \perp a,\ a^{-1}y = y'}{a^{-1}(x \cdot y) = x \cdot y'}$
[HID]	$\dfrac{a^{-1}x = x'}{(a \setminus T)^{-1}(x \oslash T) = x' \oslash (T \setminus a)}$	[PAR$_0$]	$\dfrac{a \in C,\ a^{-1}x = x',\ a^{-1}y = y'}{a^{-1}(x \underset{C}{\parallel} y) = x' \underset{C}{\parallel} y'}$
[RES]	$\dfrac{a^{-1}x = x',\ a \leq T}{a^{-1}(x \oslash T) = x' \oslash (T - a)}$	[PAR$_1$]	$\dfrac{C, \mathrm{cons}(y) \perp a,\ a^{-1}x = x'}{a^{-1}(x \underset{C}{\parallel} y) = x' \underset{C}{\parallel} y}$
[REC]	$\dfrac{(\mathrm{rec}\, x.f)(\sigma) = y}{\{x\}^{-1}y = f(\sigma[x \mapsto y])}$	[PAR$_2$]	$\dfrac{C, \mathrm{cons}(x) \perp a,\ a^{-1}y = y'}{a^{-1}(x \underset{C}{\parallel} y) = x \underset{C}{\parallel} y'}$

Table 2

The next proposition, which is easy to prove relying on proposition 6.3 and the previous lemma, allows us to translate linear transition on the operational side to linear residue on the denotational side. For all $u \in \mathbb{A}^*_{\mathcal{R}}$, $x, x' \in \mathbb{C}$ if $u^{-1}x = x'$ then we suppose in particular $u \trianglelefteq x$.

Proposition 7.2 *Let $u \in \mathbb{A}^*_{\mathcal{R}}$, $p, p' \in \mathcal{L}$, $\sigma \in \mathbb{C}^{\mathcal{V}}$ and $\tau = \mathrm{cons}^{\mathcal{V}}(\sigma)$. Then*

$$p \xrightarrow[\tau]{u} p' \quad \Longrightarrow \quad u^{-1}[\![p]\!](\sigma) = [\![p']\!](\sigma)$$

The following easy technical lemma concerns the interaction of the operators with the action prefix.

Lemma 7.3 *For any $a \in \mathbb{A}_{\mathcal{R}}$, $x, y \in \mathbb{C}$, $c \in \mathbb{A}_{\mathcal{C}}$, $T \in \mathbb{A}$, $C \subseteq \mathbb{A}_{\mathcal{R}}$, $f \in \mathbb{D}$, $\sigma \in \mathbb{C}^{\mathcal{V}}$ we have the properties of Table 3.*

The next proposition, which is difficult to prove relying on proposition 6.3 and the previous lemma, allows us to translate linear prefix on the denotational side to linear transition on the operational side. The main difficulty resides in adequately treating the hiding operator which is the most difficult case. We only obtain the translation for a large subclass of terms that we call nice which do not contain hiding

[ACT]	$a \trianglelefteq c \Longrightarrow a = c.$
[REN]	$a \trianglelefteq S(x) \Longrightarrow b \trianglelefteq x$ for some $b \in \mathbb{A}_{\mathcal{R}}$ such that $a = S(b)$.
[HID]	$a \trianglelefteq x \oslash T \Longrightarrow$ $u \trianglelefteq ub \trianglelefteq x$ for some $u \in \mathbb{A}^*_{\mathcal{R}}$ and $b \in \mathbb{A}_{\mathcal{R}}$ such that $u \oslash T = 0$ and $(ub) \oslash T = a$
[RES]	$a \trianglelefteq x \oslash T \Longrightarrow a \trianglelefteq x$ and $a \leq T$.
[SER]	$a \trianglelefteq x \cdot y \Longrightarrow (a \trianglelefteq x)$ or $(\mathrm{cons}(x) \perp a$ and $a \trianglelefteq y)$.
[PAR]	$a \trianglelefteq x \underset{C}{\parallel} y \Longrightarrow$ $(a \in C$ and $a \trianglelefteq x$ and $a \trianglelefteq y)$ or $(C, \mathrm{cons}(y) \perp a$ and $a \trianglelefteq x)$ or $(C, \mathrm{cons}(x) \perp a$ and $a \trianglelefteq y)$.
[REC]	$a \trianglelefteq (\mathrm{rec}\, x.f)(\sigma) \Longrightarrow a = \{x\}$.

Table 3

terms that hide variables of open subterms, a condition that is rather sensible to assume for any practical purposes.

Let $\preceq$ denote the *subterm ordering* in $\mathcal{L}$. $p \in \mathcal{L}$ is a *nice term* iff for all subterms $p_1 \oslash T \preceq p$ we have $T \wedge \mathcal{V} = 0$ or $p_1 \in \mathcal{L}_c$. The *set of nice terms* is denoted by $\mathcal{L}_n$, the *set of nice and closed terms* is denoted by $\mathcal{L}_{c,n}$.

Proposition 7.4 *Let $u \in \mathbb{A}^*_{\mathcal{R}}$, $p \in \mathcal{L}_n$, $\tau \in \mathbb{A}^{\mathcal{V}}$. Then*

$$u \trianglelefteq [\![p]\!](\tau) \quad \Longrightarrow \quad p \overset{u}{\underset{\tau}{\Longrightarrow}}$$

Using the linear translations we are able to state a denotational characterization of the linear behaviour, which observes only strings of actions.

Theorem 7.5 (Linear Congruence) *For all $p \in \mathcal{L}_{c,n}$ we have*

$$\mathbb{A}^*_{\mathcal{R}}(p) = \{\, u \in \mathbb{A}^*_{\mathcal{R}} \mid u \trianglelefteq [\![p]\!] \,\}$$

As a main result, we next infer a linear full abstraction by probing processes terms in suitable hiding contexts. A *context* $C(\_)$ is a term $C \in \mathcal{L}$ with one distinguished variable denoted by $\_$. A context $C(\_)$ is *nice-preserving* iff $C(p) \in \mathcal{L}_n$ whenever $p \in \mathcal{L}_{c,n}$.

Theorem 7.6 (Linear Full Abstraction) *For all $p, q \in \mathcal{L}_{c,n}$ we have*

$$[\![p]\!] = [\![q]\!] \iff \text{for all nice-preserving } C(\_) \text{ we have } \mathbb{A}^*_{\mathcal{R}}(C(p)) = \mathbb{A}^*_{\mathcal{R}}(C(q))$$

Using the fact that each multi-pomset $x \in \mathbb{R}$ is the intersection of its set of linearizations $\mathrm{Lin}(x) \subseteq \mathbb{A}^*_{\mathcal{R}}$, we derive a denotational characterization of the complex behaviour, which observes multi-pomsets of actions.

Theorem 7.7 (Complex Congruence) *For all $p \in \mathcal{L}_{c,n}$ we have*

$$\mathbb{C}(p) = \mathrm{Kmp}(\llbracket p \rrbracket) \text{ and } \llbracket p \rrbracket = \bigvee \mathbb{C}(p)$$

This allows us to finally infer a complex full abstraction result.

Theorem 7.8 (Complex Full Abstraction) *For all $p, q \in \mathcal{L}_{c,n}$ we have*

$$\llbracket p \rrbracket = \llbracket q \rrbracket \iff \text{for all nice-preserving } C(\_) \text{ we have } \mathbb{C}(C(p)) = \mathbb{C}(C(q))$$

We conclude the presentation with a result relating the denotational semantics to bisimilarity and the consumption semantics.

A relation $S \subseteq \mathcal{L}_{c,n} \times \mathcal{L}_{c,n}$ is a *bisimulation* iff for all $u \in \mathbb{A}^*_{\mathcal{R}}$ we have

- $p \, S \, q$ and $q \stackrel{u}{\Longrightarrow} q'$ then $p \stackrel{u}{\Longrightarrow} p'$ and $p' \, S \, q'$ for some p',
- $p \, S \, q$ and $p \stackrel{u}{\Longrightarrow} p'$ then $q \stackrel{u}{\Longrightarrow} q'$ and $p' \, S \, q'$ for some q'.

The coarsest bisimulation $\approx = \bigcup \{ S \mid S \text{ bisimulation} \}$, called bisimilarity, is known to be an equivalence relation (see for example [8]).

Theorem 7.9 *For all $p, q \in \mathcal{L}_{c,n}$ we have*

$$\llbracket p \rrbracket = \llbracket q \rrbracket \iff \mathrm{Cons}(p) = \mathrm{Cons}(q) \text{ and } p \approx q.$$

8 Conclusion

We developed a truly concurrent semantics that allows describing the concurrent behaviour of recursive processes accessing consumable resources. We first presented the coherently complete and prime algebraic ground domains of real and complex multi-pomsets. The modelled process language contains several deterministic quantitative process operators as well as a recursion operator. Next, we displayed a deterministic structural operational machine that is straightforward to comprehend and allows extracting a linear and a complex behaviour. We then constructed a compositional denotational semantics using a functional domain over environments of complex multi-pomsets. The main results have finally shown that the denotational semantics is fully abstract with respect to both linear and complex operational semantics.

The only operator customary in classical process languages [7,8] that has been intentionally left out in ours is the *non-deterministic choice*. We think that using power-domains over complex multi-pomsets provides a clear way of handling choice at the expense of rendering the domain-theoretic tools more involved. Another possibility to achieve the same result could reside in enriching real multi-pomsets with a further relation on events expressing conflict of choices, thus imposing a modelling view closer to event structures.

Apart from the intended applications in engineering and economy the presented language can be also employed as a powerful formalism to abstractly specify and

handle labelled partial orders which are far more complex than the series-parallel constructions usually considered in the literature.

Acknowledgement

I thank Paul Gastin for several fruitful discussions, Volker Diekert for pointing out interesting extensions, Daniele Varacca for constructive criticism, and Pierre-Louis Curien for various pertinent remarks.

References

[1] S. Abramsky, A. Jung. Domain Theory. In *Handook of Logic in Computer Science*, Vol. III, pages 1-168, Clarendon Press, 1994.

[2] V. Diekert, P. Gastin. A Domain for Concurrent Termination: A generalization of Mazurkiewicz traces. *Proceedings of the 22th International Colloquium on Automata, Languages and Programming, 1995*. Lecture Notes in Computer Science 944, pages 15-26, Springer, 1995.

[3] M. Droste, W. Kuich, H. Vogler. Handbook of Weighted Automata. EATCS Monographs in Theoretical Computer Science, Springer, 2009.

[4] P. Gastin, M. Mislove. A Truly Concurrent Semantics for a Process Algebra using Resource Pomsets. *Theoretical Computer Science*, volume 281, number 1-2, pages 369-421, 2002.

[5] P. Gastin, D. Teodosiu. Resource Traces: A Domain for Processes sharing Exclusive Resources. *Twelfth Conference on the Mathematical Foundations of Programming Semantics, 1996. Theoretical Computer Science*, volume 278, number 1-2, pages 195–221, May 2002.

[6] M. Hennessy, G. D. Plotkin. Full Abstraction for a Simple Parallel Programming Language. Lecture Notes in Computer Science 74, Springer, 1979.

[7] C. A. R. Hoare. Communicating Sequential Processes. Prentice-Hall International Series in Computer Science, Prentice Hall, 1985.

[8] R. Milner. Communication and Concurrency. Prentice Hall International Series in Computer Science, Prentice Hall, 1989.

[9] V. Pratt. Modeling Concurrency with Partial Orders. *International Journal of Parallel Programming*, volume 15, issue 1, pages 33-71, Kluwer Academic Publishers, February 1986.

[10] D. Pym, C. Tofts. A Calculus and Logic of Resources and Processes. *Formal Aspects of Computing*, volume 18, pages 495-517, 2006.

[11] D. Teodosiu. A Truly Concurrent Semantics for Processes sharing Quantified Resources. *PhD Thesis*, Department of Computer Science, University Paris 7, March 2012, at http://teodosiu.pagesperso-orange.fr.

[12] G. Winskel. Event Structures. In W. Brauer, W. Reisig, G. Rozenberg (editors), Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course, Bad Honnef, September 1986, *Lecture Notes in Computer Science* 255, pages 325-392, Springer, 1987