Theoretical Computer Science 90 (1991) 127–149
Elsevier

Four domains for concurrency

J.W. de Bakker and J.H.A. Warmerdam*

Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Abstract

Bakker, J.W. de, and J.H.A. Warmerdam, Four domains for concurrency, Theoretical Computer Science 90 (1991) 127–149.

We give four domains for concurrency in a uniform way by means of domain equations. The domains are intended for modelling the four possible combinations of linear time versus branching time, and of interleaving versus noninterleaving concurrency. We use the linear time, noninterleaved domain to give operational and denotational semantics for a simple concurrent language with recursion, and prove that $\mathcal{O} = \mathcal{D}$.

Prologue

Among the reasons to fondly remember my first IFIP Congress (New York, 1965), I recall a meeting with the late Professors Andrei Ershov and Aad van Wijngaarden, both then already famous scholars, who strongly encouraged me to continue my incipient work on programming language semantics.

Among the reasons to somewhat embarrassedly remember the 6th MFCS meeting (Tatranska Lomnica, 1977), I recall a discussion with Andrei Ershov on my unsatisfactory first steps towards an understanding of concurrency semantics and infinite behaviour (cf. [6]). The paper to follow reports on how we spent the 1980s in Amsterdam working to remedy this.

Among the reasons to sadly remember my otherwise so enjoyable visit to Akademgorodok in the fall of 1988, I recall in sorrow the news about the mortal illness and death of Academician Andrei Ershov, eminent computer scientist and world specialist in programming.

Jaco de Bakker, Amsterdam, May 1990

* Supported by the Netherlands Organization for the Advancement of Research (N.W.O.), project N.F.I.-REX.

1. Introduction

Since 1981, the Amsterdam Concurrency Group (ACG) has been investigating concurrency semantics employing the tools of metric topology. The key observation explaining the relevance of the metric approach is the following: Consider two computations p_1, p_2. A natural *distance* $d(p_1, p_2)$ may be defined by putting

$$d(p_1, p_2) = 2^{-n}$$

where n ($\triangleq sup\{k: p_1[k] = p_2[k]\}$) is the length of the longest common initial segment of p_1 and p_2. Details vary with the form of the p_1, p_2. If computations are given as words (finite or infinite sequences of atomic actions), we take the standard notion of prefix; if p_1, p_2 are trees, we use truncation at depth k for $p[k]$. Other kinds of computations, e.g. involving function application, may be accommodated as well.

Complete metric spaces (cms's) have the characteristic property that Cauchy sequences always have limits; this motivates their use for smooth handling of infinite behaviour. In addition, each *contracting* function $f:(M, d) \to (M, d)$, for (M, d) a cms, has a *unique* fixed point (by Banach's theorem). Contracting functions $f:(M_1, d_2) \to (M_2, d_2)$ bring points closer together: it is required that, for some real $\alpha \in [0, 1)$, $d_2(f(x), f(y)) \leq \alpha \cdot d_1(x, y)$. Uniqueness of fixed points may conveniently be exploited in a variety of situations.

In the paper [17] we showed how to apply metric techniques to solve domain equations

$$P \cong \mathscr{F}(P) \tag{1.1}$$

or, rather, $(P, d) \cong \mathscr{F}((P, d))$, with (P, d) the cms to be determined, $\cong$ isometry, and $\mathscr{F}$ a mapping built from given cms's $(A, d_A), \ldots$, the unknown (P, d), and composition rules such as $\dot\cup$ (disjoint union), $\times$ (Cartesian product), and $\mathscr{P}_{closed}(\cdot)$ (closed subsets of $\cdot$). Section 2 will provide more information on this method.

In a series of papers, starting with [17, 10, 12, 13, 14], we developed denotational ($\mathscr{D}$) and operational ($\mathscr{O}$) semantics for a number of simple languages with concurrency. Here a denotational semantics $\mathscr{D}$ for a language $\mathscr{L}$ is given as a mapping: $\mathscr{L} \to P_1$ (for some P_1 solving (1.1) for a suitable $\mathscr{F}_1$), which is *compositional* and treats recursion through fixed points. $\mathscr{O}$ is a mapping: $\mathscr{L} \to P_2$, which is derived from some Plotkin-style transition system [27], and which handles recursion through syntactic substitution. Also, in the papers referred to, we encounter the contrasting themes of *linear time* (LT, sets of sequences) versus *branching time* (BT, tree-like structures) semantic domains, and of *uniform* (uninterpreted atomic actions) versus *nonuniform* (interpreted actions) concurrency.

After an initial phase in which ACG developed the basic machinery of metric semantics, the group directed its efforts towards concurrency in the setting of object-oriented and, subsequently, of logic programming. In a collaborative effort with Philips Research Eindhoven, within the framework of a project with substantial support from the ESPRIT programme, we designed operational and denotational semantics for the parallel object-oriented language POOL, and investigated the relationship between the respective models [2, 1, 3, 4, 9, 29]. Throughout these studies, fruitful use was made of the metric formalism. Two further papers deserve special mention. In [5], the technique from [17] for solving domain equations (1.1) was generalized and phrased in the *category* of cms's. In [24], a powerful method was proposed to establish equivalences such as $\mathcal{O} = \mathcal{D}$, by (i) *defining* $\mathcal{O}$ as a fixed point of a contracting higher-order mapping Φ (obtained from an appropriate transition system), and (ii) proving that $\mathcal{D} = \Phi(\mathcal{D})$. By Banach's theorem, $\mathcal{O} = \mathcal{D}$ is then immediate (cf. also [13], where several more examples of the Kok-Rutten-method are treated).

Parallelism in the setting of logic programming (LP) was first studied in [7, 23]. The paper [7] proposed to investigate control flow in LP abstracting from the logical intricacies (no substitutions, refutations etc.), and shows how the basic metric techniques apply as well to this, at first sight rather remote, territory. Related work includes [11, 19].

In all of the papers mentioned so far, parallel composition has been handled by the so-called *interleaving* model: typically, the meaning of the statement $s \equiv a \| b$ is given as $\{ab, ba\}$ in an LT, or as shown in Fig. 1 in a BT-style model. Accordingly, the equivalence (*): $a \| b = (a;b) + (b;a)$ is valid in all such models. In recent years, increased attention has been paid to models of the so-called *true concurrency* (or noninterleaving) kind. A variety of domains has been developed where concurrency is modelled through *simultaneity*; thus, in these models, (*) is not satisfied. Well-known examples are Pratt's pomsets [28], and the event structures of [25]. (cf. [15] for extensive references).

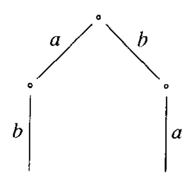

Fig. 1.

At last, we are in a position to formulate the goal of the present paper. We shall discuss a case study in metric semantics, by designing *four domains for concurrency*. These four domains will be employed to model the four possible combinations of linear time versus branching time, and of interleaving versus noninterleaving concurrency. Contrary to the way these or related models have been presented elsewhere in the literature, we shall pay special attention to their development in such a way as to bring out their similarities rather than their differences. We shall give four systems of domain equations with seemingly small differences. Putting it somewhat differently, we want to demonstrate the power of the domain equations approach, by showing how four ways of looking at concurrency, all of which have been advocated or attacked in vivid debates, may be seen as relatively mild variations on the same theme.

Section 2 will be devoted to the four (systems of) equations. The techniques applied here are partly general (as in [17, 5]), partly more ad-hoc, and then follow [11]. Section 3 illustrates the use of domains in semantic design. We select one of the four domains (LT, noninterleaving). For a simple concurrent language with recursion, we design operational and denotational semantics based on this domain, and prove that $\mathcal{O} = \mathcal{D}$. In order to establish this, we apply an extension of the Kok-Rutten-method which may have some interest of its own (and which is close to a method from [7, Section 9]). Technically, this proof constitutes the main contribution of the present paper. For the two interleaving models, such an equivalence proof was already presented earlier [24, 13]; for the BT-noninterleaving model it requires further study whether the argument of Section 3 may be appropriately modified.

We conclude this introduction with a few words on related work. In [8], we also presented four domains for concurrency, but restricted to true concurrency in the form of *synchronous step semantics* only. In [16], we developed a metric pomset semantics for the same language as treated here. Compared to the semantics of Section 3, the transition system of [16] is less convincing. Only transitions of the form $s \xrightarrow{p} E$ are used (s finishes in one step with pomset p as result), rather than also transitions with intermediate steps $s \xrightarrow{p} s'$. On the other hand, the present paper utilizes the same technique for handling recursion, in particular the infinitary proof rule, as in [16]. The pomset model may be fruitfully combined with the domain equations approach to cope with certain problems the methodology of the present paper cannot deal with. Some comments on this follow in the concluding section of our paper.

2. Introduction of the domains by means of domain equations

We assume the reader is acquainted with the notion of (complete) (ultra-) metric space, converging sequence, closed set, as well as the constructors $\dot{\cup}$ (disjoint union), $\times$ (Cartesian product) and $\mathcal{P}_{closed}(\cdot)$ (closed subsets of $\cdot$). In this paper we only consider distance mappings that are bounded by 1. The reader may consult [21, 22] for (metric) topology and, for instance, [5] for the notions we use in metric semantics. Before we can give the domain equations in the second subsection, we need to introduce two new notions, a length function l and a constructor $\rhd$.

2.1. *Introduction of two new notions: l and $\rhd$*

Usually if we write down $A \times P$, or more precisely $A \times id_{1/2}(P)$, where P is a metric space with metric d_p and A is a set of atomic actions (with discrete metric) we assume $A \times P$ is supplied with a metric $d_{A\times P}$ defined by

$$d_{A\times P}(\langle a_1, p_1\rangle, \langle a_2, p_2\rangle) = \begin{cases} 1, & a_1 \neq a_2, \\ \frac{1}{2} \cdot d_P(p_1, p_2), & a_1 = a_2. \end{cases}$$

For the non-interleaving domains we need to generalize this construction to the case where the left-hand side of the Cartesian product contains a nondiscrete metric space. For this purpose we need a notion of length so that we can define a metric on $P_1 \times P_2$ by

$$d_{P_1\times P_2}(\langle p_1, p_2\rangle, \langle p_1', p_2'\rangle) = \begin{cases} d_{P_1}(p_1, p_1'), & p_1 \neq p_1', \\ 2^{-l_{P_1}(p_1)} \cdot d_{P_2}(p_2, p_2'), & p_1 = p_1', \end{cases}$$

where $l_{P_1}(p_1)$ is the length of p_1 in the metric space P_1. This product together with P_1 (i.e. $P_1 \cup (P_1 \times P_2)$) is denoted by $P_1 \rhd P_2$.

From now on we assume that every metric space (X, d_X) is supplied with a length function $l_X : X \to \{1, 2, \ldots\} \cup \{\infty\}$ such that

$$(d_X(x, y) < 2^{-(l-1)} \wedge x \neq y) \Rightarrow (l_X(x) \geq l \wedge l_X(y) > l) \text{ or } (l_X(x) > l \wedge l_X(y) \geq l).$$

This amounts to saying "we cannot have a small distance between short elements (i.e. elements with small length), unless they are equal". If we write a sentence like "the metric space X..." in the sequel, we mean the metric space (X, d_X) with length function l_X.

Definition 2.1.1. We define metric spaces A, $id_{1/2}(X)$, $fin(X)$, $\mathcal{P}_{nc}(X)$, $X_1 \mathbin{\dot\cup} X_2$, $X_1 \rhd X_2$, where A is some fixed set (of atomic actions) and X, X_1, X_2 are metric spaces.

(1) $$d_A(a_1, a_2) = \begin{cases} 1, & a_1 \neq a_2, \\ 0, & a_1 = a_2, \end{cases} \qquad l_A(a) = 1.$$

(2) $$id_{1/2}(X) = X,$$
$$d_{id_{1/2}(X)}(x, y) = \tfrac{1}{2} \cdot d_X(x, y),$$
$$l_{id_{1/2}(X)}(x) = l_X(x) + 1.$$

(3) $$fin(X) = \{x \in X \mid l_X(x) < \infty\},$$
$$d_{fin(X)} = d_X \restriction (fin(X) \times fin(X)),$$
$$l_{fin(X)} = l_X \restriction fin(X).$$

(4) $$\mathcal{P}_{nc}(X) = \{A \subseteq X \mid A \text{ is a nonempty } d_X\text{-closed subset of } X\},$$
$$d_{\mathcal{P}_{nc}(X)}(A, B) = \max\left\{\sup_{a \in A} d_X(a, B), \sup_{b \in B} d_X(b, A)\right\},$$
$$l_{\mathcal{P}_{nc}(X)}(A) = \sup_{a \in A} l_X(a).$$

(5) $$X_1 \mathbin{\dot\cup} X_2 = (\{1\} \times X_1) \cup (\{2\} \times X_2),$$
$$d_{X_1 \dot\cup X_2}(\langle i, z_1\rangle, \langle j, z_2\rangle) = \begin{cases} 1, & i \neq j, \\ d_{X_i}(z_1, z_2), & i = j, \end{cases} \qquad l_{X_1 \dot\cup X_2}(\langle i, z\rangle) = l_{X_i}(z).$$

From now on we will informally use $X_1 \mathbin{\dot\cup} X_2$ as if it were $X_1 \cup X_2$ with disjoint X_1 and X_2.

(6) Let $x_1, x_1' \in X_1$ and $x_2, x_2' \in X_2$.

$$X_1 \rhd X_2 = X_1 \mathbin{\dot\cup} (fin(X_1) \times X_2),$$

$$d_{X_1 \rhd X_2}(x_1, x_1') = d_{X_1}(x_1, x_1');$$

$$d_{X_1 \rhd X_2}(x_1, \langle x_1', x_2' \rangle) = d_{X_1 \rhd X_2}(\langle x_1, x_2 \rangle, x_1') = \begin{cases} d_{X_1}(x_1, x_1'), & x_1 \neq x_1', \\ 2^{-l_{X_1}(x_1)}, & x_1 = x_1'; \end{cases}$$

$$d_{X_1 \rhd X_2}(\langle x_1, x_2 \rangle, \langle x_1', x_2' \rangle) = \begin{cases} d_{X_1}(x_1, x_1'), & x_1 \neq x_1', \\ 2^{-l_{X_1}(x_1)} \cdot d_{X_2}(x_2, x_2'), & x_1 = x_1', \end{cases}$$

$$l_{X_1 \rhd X_2}(x_1) = l_{X_1}(x_1); \; l_{X_1 \rhd X_2}(\langle x_1, x_2 \rangle) = l_{X_1}(x_1) + l_{X_2}(x_2).$$

Note that there is a slight difference between $A \mathbin{\dot\cup} (A \times id_{1/2}(P))$ and $A \rhd P$, namely $d_{A \dot\cup (A \times id_{1/2}(P))}(a, \langle a, p \rangle) = 1$ and $d_{A \rhd P}(a, \langle a, p \rangle) = \frac{1}{2}$, the latter being a little more intuitive. In the general case, it is important that distances in $P_1 \rhd P_2$ between a $p_1 \in P_1$ and a $\langle p_1', p_2' \rangle \in fin(P_1) \times P_2$ may be small (not just 1). In $P_1 \rhd P_2$ there exist sequences $(\langle p_1^i, p_2^i \rangle)_i$ with limit $p \in P_1$. In this case $l(p_1^i) \to \infty$.

Proposition 2.1.2. (1) *If the metric spaces X, X_1 and X_2 are ultra metric, then A, $id_{1/2}(X)$, $fin(X)$, $\mathcal{P}_{nc}(X)$, $X_1 \mathbin{\dot\cup} X_2$, $X_1 \rhd X_2$ are ultra metric spaces.*

(2) *If the metric spaces X, X_1 and X_2 are complete, then A, $id_{1/2}(X)$, $\mathcal{P}_{nc}(X)$, $X_1 \mathbin{\dot\cup} X_2$, $X_1 \rhd X_2$ are complete metric spaces.*

2.2. *Four systems of domain equations*

We are now able to give four sets of domain equations for the four possible combinations of linear time versus branching time, and of interleaving versus noninterleaving concurrency (Table 1). Let us explain in words what a $p \in P$ in the most difficult domain (noninterleaved/branching time) stands for. A process ($p \in P$) is a set of branches ($q \in Q$), standing for a set of choices. Each branch is either a final action (r) or a pair ($\langle r, p \rangle$) consisting of a finite action and a resumption. An action ($r \in R$) is either an atomic action ($a \in A$) or a set of processes, standing for the parallel execution of these processes.

Table 1

	Linear Time	*Branching Time*
Interleaved	$P \cong \mathcal{P}_{nc}(Q)$ $Q \cong A \rhd Q$	$P \cong \mathcal{P}_{nc}(Q)$ $Q \cong A \rhd P$
Non-interleaved	$P \cong \mathcal{P}_{nc}(Q)$ $Q \cong R \rhd Q$ $R \cong A \mathbin{\dot\cup} \mathcal{P}_{nc}(id_{1/2}(Q))$	$P \cong \mathcal{P}_{nc}(Q)$ $Q \cong R \rhd P$ $R \cong A \mathbin{\dot\cup} \mathcal{P}_{nc}(id_{1/2}(P))$

In the next section we give an operational and denotational semantics for a simple language, based on the linear time/noninterleaved domain. Now we illustrate the four domain equations by giving four different semantics for a simple statement (in the language to be introduced in Section 3.1). Consider the statement $a;((b\|c);e+d)$. We give, besides the formal processes denoting this statement in the four models, also drawings of these processes. In these pictures an open node indicates choice (of possibly one alternative) and nodes are closed in other cases. The "and" in a picture denotes (noninterleaved) parallel execution. The pictures are drawn in such a way that the length of the pictures (the number of node-to-node intervals) coincides with the length in the domains.

Linear time/interleaved (Fig. 2)

$$\mathcal{D}_{Lt,In}(a;((b\|c);e+d))=\{\langle a,\langle b,\langle c,e\rangle\rangle\rangle,\langle a,\langle c,\langle b,e\rangle\rangle\rangle,\langle a,d\rangle\}.$$

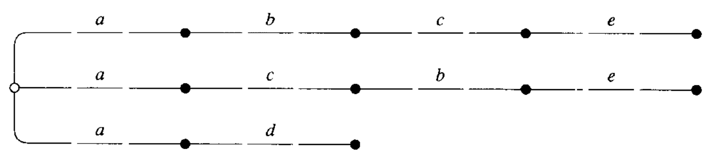

Fig. 2.

Branching time/interleaved (Fig. 3)

$$\mathcal{D}_{Bt,In}(a;((b\|c);e+d))=\{\langle a,\{\langle b,\{\langle c,\{e\}\rangle\}\rangle,\langle c,\{\langle b,\{e\}\rangle\}\rangle,d\}\rangle\}.$$

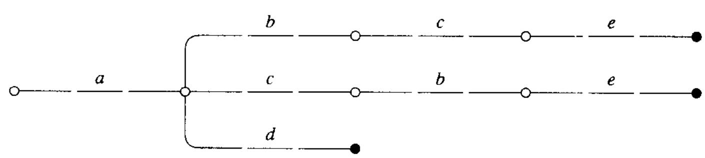

Fig. 3.

Linear time/noninterleaved (Fig. 4)

$$\mathcal{D}_{Lt,Ni}(a;((b\|c);e+d))=\{\langle a,\langle\{b,c\},e\rangle\rangle,\langle a,d\rangle\}.$$

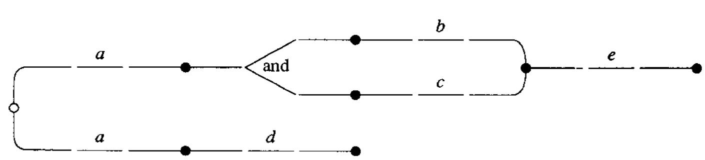

Fig. 4.

Branching time/noninterleaved (Fig. 5)

$$\mathcal{D}_{Bt,Ni}(a;((b\|c);e+d))=\{\langle a,\{\langle\{\{b\},\{c\}\},\{e\}\rangle,d\}\rangle\}.$$

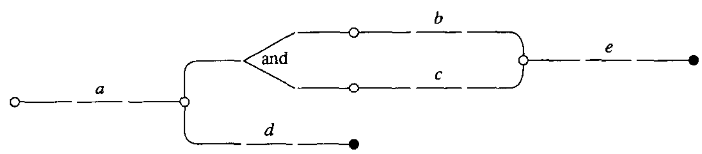

Fig. 5.

Consider the following statements to see the difference between linear time and branching time semantics, and between interleaved and noninterleaved semantics.

$$s_1\equiv a\|(b+c),\qquad s_2\equiv a\|b+a\|c,\qquad s_3\equiv a;(b+c)+(b+c);a.$$

Linear time/interleaved

$$\mathcal{D}_{Lt,In}(s_1)=\mathcal{D}_{Lt,In}(s_2)=\mathcal{D}_{Lt,In}(s_3)=\{\langle a,b\rangle,\langle a,c\rangle,\langle b,a\rangle,\langle c,a\rangle\}.$$

Branching time/interleaved

$$\mathcal{D}_{Bt,In}(s_1)=\mathcal{D}_{Bt,In}(s_3)=\{\langle a,\{b,c\}\rangle,\langle b,\{a\}\rangle,\langle c,\{a\}\rangle\},$$

$$\mathcal{D}_{Bt,In}(s_2)=\{\langle a,\{b\}\rangle,\langle a,\{c\}\rangle,\langle b,\{a\}\rangle,\langle c,\{a\}\rangle\}.$$

Linear time/noninterleaved

$$\mathcal{D}_{Lt,Ni}(s_1)=\mathcal{D}_{Lt,Ni}(s_2)=\{\{a,b\},\{a,c\}\},$$

$$\mathcal{D}_{Lt,Ni}(s_3)=\mathcal{D}_{Lt,In}(s_3).$$

Branching time/noninterleaved

$$\mathcal{D}_{Bt,Ni}(s_1)=\{\{\{a\},\{b,c\}\}\},$$

$$\mathcal{D}_{Bt,Ni}(s_2)=\{\{\{a\},\{b\}\},\{\{a\},\{c\}\}\},$$

$$\mathcal{D}_{Bt,Ni}(s_3)=\mathcal{D}_{Bt,In}(s_3).$$

The branching time models distinguish between s_1 and s_2 whereas the linear time models do not. The noninterleaving models distinguish between s_1 and s_3 whereas the interleaving models do not.

The interleaving domain equations can be solved in the category of complete metric spaces as is shown in [17] and in a more general setting in [5]. The America–Rutten-theory cannot be applied to the noninterleaving case immediately, since there does not exist a notion of length in a general complete metric space, which is essential for our definition of the metric on a product space. We are convinced, however, that an adjustment of the category of complete metric spaces is possible, without affecting the theorem, in order to solve the above equations.

We will briefly discuss the construction of a solution for the noninterleaved linear time equation in a De Bakker–Zucker-like way.

Definition 2.2.1. Define

$$\begin{cases} R_0 = A \\ Q_0 = R_0 \end{cases} \text{and} \quad \begin{cases} R_{n+1} = A \cup \mathcal{P}_{nc}(id_{1/2}(Q_n)) \\ Q_{n+1} = R_n \rhd Q_n. \end{cases}$$

Note that $R_n \subseteq R_{n+1}$ and $Q_n \subseteq Q_{n+1}$. Let

$$\begin{cases} Q_\omega = \bigcup Q_n, & d_{Q_\omega} = \bigcup d_{Q_n}, & l_{Q_\omega} = \bigcup l_{Q_n}, \\ R_\omega = \bigcup R_n, & d_{R_\omega} = \bigcup d_{R_n}, & l_{R_\omega} = \bigcup l_{R_n}. \end{cases}$$

Let

$$\begin{cases} Q = \overline{Q_\omega}: & \text{the completion of } Q_\omega, \quad l_Q(\lim_i q_i) = \lim_i l_{Q_\omega}(q_i), \\ R = \overline{R_\omega}: & \text{the completion of } R_\omega, \quad l_R(\lim_i r_i) = \lim_i l_{R_\omega}(r_i), \\ P = \mathcal{P}_{nc}(Q). \end{cases}$$

These P, Q and R satisfy the linear time/noninterleaved system of domain equations, which is stated in the following.

Theorem 2.2.2. (1) $\mathit{fin}(R) \cong R_\omega$,
(2) $Q \cong R \rhd Q$,
(3) $R \cong A \cup \mathcal{P}_{nc}(id_{1/2}(Q))$.

3. Linear time/noninterleaved semantics for a concurrent language

In this section we show how to use the linear time/noninterleaved domain to give operational and denotational semantics for a simple concurrent language ($\mathcal{L}$). In the first subsection we introduce the language. In the second subsection we give a transition system and derive some properties of this transition system. The third subsection contains the definition of an operational semantics $\mathcal{O}$, based on this transition system. The fourth subsection contains semantical operators which are the counter-parts of the syntactical operators in the language. With the aid of these operators we give a denotational semantics $\mathcal{D}$ for the language $\mathcal{L}$. The fifth and concluding subsection will contain the proof of the equivalence of the operational and denotational semantics.

3.1. The language

First we introduce the language. For this we need two basic sets. Let $(a, b, c, \ldots \in) A$ be a (finite or infinite) set of atomic actions and let $(x \in) \mathcal{P}var$ be a set of procedure variables.

Definition 3.1.1. (a) The class $(s \in) \mathcal{L}$ of *statements* is given by

$$s ::= a \mid x \mid s_1; s_2 \mid s_1 + s_2 \mid s_1 \| s_2.$$

(b) The class $(g \in) \mathcal{L}_g$ of *guarded statements* is given by

$$g ::= a \mid g; s \mid g_1 + g_2 \mid g_1 \| g_2.$$

(c) The class $(d \in)\mathcal{D}ecl$ of *declarations* consists of mappings from $\mathcal{P}var$ to $\mathcal{L}_g$.

(d) The class $(\pi \in)\mathcal{P}rog$ of *programs* consists of pairs $\pi \equiv \langle d \mid s\rangle$ with $d \in \mathcal{D}ecl$ and $s \in \mathcal{L}$.

3.2. Transition system for $\mathcal{L}$

In this subsection we give a Plotkin-style transition system and derive some properties about the system.

Usually, transitions are of the form $s \xrightarrow{a} s'$, where s and s' are statements and a is an action (or a set of actions, in step semantics). The intuition is that the statement s can be executed by doing the action a. After this we have to proceed with statement s'. In true concurrency semantics, this can not be applied immediately. Consider the following situation: $s_1 \xrightarrow{a_1} s_1'$ and $s_2 \xrightarrow{a_2} s_2'$. If we derive something like $s_1 \| s_2 \xrightarrow{\{a_1, a_2\}} s_1' \| s_2'$, then the information is lost that a_1 stems from s_1 and a_2 stems from s_2. This information is essential, for if $s_1' \xrightarrow{b} s_1''$, we want to combine, in the operational semantics, the b with only the a_1, not with $\{a_1, a_2\}$.

Some people proposed to use placeholders [20] in order to be able to determine which actions belong to some statements in a parallel construct. We will use another approach here. Firstly, we add transitions of the form $s \xrightarrow{q} E$ to our transition system, where q is a *sequence* of actions and E is the terminated statement. Secondly, instead of combining $s_1 \xrightarrow{a_1} s_1'$ and $s_2 \xrightarrow{a_2} s_2'$ at this stage, there will be a rule to combine $s_1 \xrightarrow{q_1} E$ and $s_2 \xrightarrow{q_2} E$ into $s_1 \| s_2 \xrightarrow{\{q_1, q_2\}} E$. ($\{q_1, q_2\}$ is now considered as one (composed) action.)

Since the only way to produce $s_1 \| s_2 \xrightarrow{q} E$ is by combining the steps $s_1 \xrightarrow{q_1} E$ and $s_2 \xrightarrow{q_2} E$, it should hold that $\forall s : \exists q : s \xrightarrow{q} E$ even if s is a nonterminating statement. In order to deal with this last case we even include transitions $s \xrightarrow{q} E$ where q is an *infinite* sequence of actions. Such infinite behaviour arises in particular when recursion is present in s. To handle this situation we have added a special action "e" to the action set and an axiom $x \xrightarrow{e} E$ to the transition system. This allows us to terminate a (recursive) procedure prematurely. If we now derive $x \xrightarrow{q_n} E$ for $n = 1, 2, 3 \ldots$ by terminating each time in a later stage, we get a Cauchy sequence $(q_n)_n$, and a Cauchy-rule in our transition system allows us to derive $x \xrightarrow{q} E$, where q is the infinite sequence of actions (without "e"), obtained by taking $\lim_n q_n$. Example 3.2.2 should help the reader to understand this method.

Let us first add the special symbol e to our domain.

$$A_e = A \mathbin{\dot\cup} \{e\}; \qquad P_e, Q_e, R_e \text{ satisfy } P_e = \mathcal{P}_{nc}(Q_e);$$

$$Q_e = R_e \rhd Q_e; \qquad R_e = A_e \mathbin{\dot\cup} \mathcal{P}_{nc}(id_{1/2}(Q_e)).$$

We will define $\rightarrow \subseteq \mathcal{L} \times Q_e \times (\mathcal{L} \mathbin{\dot\cup} \{E\})$ in a moment. Here E is a special symbol denoting the terminated statement. We will use s for real statements, i.e. elements of $\mathcal{L}$, and t for members of $\mathcal{L} \mathbin{\dot\cup} \{E\}$. We use the notation $s \xrightarrow{q} t$ instead of $(s, q, t) \in \rightarrow$. In case $s \xrightarrow{q} t$ with $q \notin R_e$ we will always have $t = E$. So one can only do a composed step q, consisting of a sequence of actions, to the final statement E.

Axioms

$$a \xrightarrow{a} E, \qquad \text{Elem}$$

$$x \xrightarrow{e} E, \qquad \text{Proc Term}$$

Rules

$$\frac{s \xrightarrow{r} s' \mid E}{s;\bar{s} \xrightarrow{r} s';\bar{s} \mid \bar{s}}, \qquad \text{Seq Comp}$$

$$\frac{s \xrightarrow{r} s' \mid E}{\begin{array}{l} s\|\bar{s} \xrightarrow{r} s'\|\bar{s} \mid \bar{s} \\ \bar{s}\|s \xrightarrow{r} \bar{s}\|s' \mid \bar{s} \end{array}}, \qquad \text{Int Par}$$

$$\frac{s \xrightarrow{r} s' \mid E}{\begin{array}{l} s+\bar{s} \xrightarrow{r} s' \mid E \\ \bar{s}+s \xrightarrow{r} s' \mid E \end{array}}, \qquad \text{Choice}$$

$$\frac{d(x) = g \wedge g \xrightarrow{r} s \mid E}{x \xrightarrow{r} s \mid E}, \qquad \text{Proc}$$

$$\frac{s \xrightarrow{r} s' \wedge s' \xrightarrow{q} E \wedge r \text{ is finite}}{s \xrightarrow{\langle r,q\rangle} E}, \qquad \text{Comp}$$

$$\frac{s \xrightarrow{r} s' \wedge r \text{ is infinite}}{s \xrightarrow{r} E}, \qquad \text{Inf-rule}$$

$$\frac{\forall i: \quad s \xrightarrow{q_i} E \wedge \lim_i q_i = q}{s \xrightarrow{q} E}, \qquad \text{Cauchy-rule}$$

$$\frac{s_1 \xrightarrow{q_1} E \wedge s_2 \xrightarrow{q_2} E}{s_1\|s_2 \xrightarrow{\{q_1,q_2\}} E}, \qquad \text{True Par}$$

Remark 3.2.1. Observe that we take a "hybrid" approach to concurrency here. We will have $\mathcal{O}(s_1\|s_2) = \mathcal{O}(s_1\|s_2 + s_1;s_2 + s_2;s_1)$. We warn the reader that we have taken the true concurrency approach (no interleaving at all) in the examples of Section 2.2 for simplicity. Without the presence of the Int Par rules we would obtain a true concurrent operational semantics in Section 3.3. If we also appropriately adapt the denotational semantics (by deleting the two left-merge parts of the semantical operator $\|$ in Definition 3.4.1) then we can obtain $\mathcal{O} = \mathcal{D}$ in a similar but simpler way as we will do here. It reduces the number of subcases in several proofs (in particular in the proof of Lemma 3.2.7). Only the proof of Lemma 3.2.5 is a bit more complicated without Int Par.

Example 3.2.2. Let $d(x) = a;(b\|x)$.

(1) $\quad a \xrightarrow{a} E, \qquad$ Elem

(2)	$a;(b\|x) \xrightarrow{a} b\|x,$	Seq Comp (1)
(3)	$x \xrightarrow{a} b\|x,$	Proc (2)
(4)	$b \xrightarrow{b} E,$	Elem
(5)	$x \xrightarrow{e} E,$	Proc term
(6)	$b\|x \xrightarrow{\{b,e\}} E,$	True Par (4,5)
(7)	$x \xrightarrow{\langle a,\{b,e\}\rangle} E,$	Comp (3,6)
(8)	$b\|x \xrightarrow{\{b,\langle a,\{b,e\}\rangle\}} E,$	True Par (4, 7)
(9)	$x \xrightarrow{\langle a,\{b,\langle a,\{b,e\}\rangle\}\rangle} E,$	Comp (3, 8)
	$\vdots$	
	$x \xrightarrow{\langle a,\{b,\langle a,\{b,\langle a,\{\cdots\}\rangle\}\rangle\}\rangle} E.$	Cauchy-rule (5, 7, 9, 11, ...)

Now we are going to prove a series of five lemmas. The last lemma is essential for the proof of the equivalence of $\mathcal{O}$ and $\mathcal{D}$ in Section 3.5. The first lemma is stating the so-called "image finiteness" property, that will be used to prove that $\mathcal{O}$ is well-defined in Section 3.3.

Lemma 3.2.3. *$\forall s$: $\forall r$: $\{s' \mid s \xrightarrow{r} s'\}$ is finite.*

Proof. Induction on the structure of s; first for guarded statements g.

$g = a$. The only rules and axioms that can be used to produce $a \xrightarrow{r} t$ are Elem, Comp, the Inf-rule and the Cauchy-rule. In all cases $t = E$. So $\forall r$: $\{s' \mid a \xrightarrow{r} s'\} = \emptyset$.

$g = g;s$. Assume $g;s \xrightarrow{r} s'$. The only rules that can be used are

$$\frac{g \xrightarrow{r} s''}{g;s \xrightarrow{r} s'';s} \quad \text{and} \quad \frac{g \xrightarrow{r} E}{g;s \xrightarrow{r} s}.$$

So $\{s' \mid g;s \xrightarrow{r} s'\} \subseteq \{s'';s \mid g \xrightarrow{r} s''\} \cup \{s\}$ is finite.

$g = g_1 \| g_2$. Assume $g_1 \| g_2 \xrightarrow{r} s'$. The only rules that can be used are the following:

$$\frac{g_1 \xrightarrow{r} s''}{g_1\|g_2 \xrightarrow{r} s''\|g_2}, \quad \frac{g_1 \xrightarrow{r} E}{g_1\|g_2 \xrightarrow{r} g_2}, \quad \frac{g_2 \xrightarrow{r} s''}{g_1\|g_2 \xrightarrow{r} g_1\|s''}, \quad \frac{g_2 \xrightarrow{r} E}{g_1\|g_2 \xrightarrow{r} g_1}.$$

So $\{s' \mid g_1\|g_2 \xrightarrow{r} s'\} \subseteq \{s''\|g_2 \mid g_1 \xrightarrow{r} s''\} \cup \{g_1\|s'' \mid g_2 \xrightarrow{r} s''\} \cup \{g_1, g_2\}$ is finite.

$g = g_1 + g_2$. Assume $g_1 + g_2 \xrightarrow{r} s'$. The only rules that can be used are

$$\frac{g_1 \xrightarrow{r} s''}{g_1 + g_2 \xrightarrow{r} s''} \quad \text{and} \quad \frac{g_2 \xrightarrow{r} s''}{g_1 + g_2 \xrightarrow{r} s''}.$$

So $\{s' \mid g_1 + g_2 \xrightarrow{r} s'\} \subseteq \{s'' \mid g_1 \xrightarrow{r} s''\} \cup \{s'' \mid g_2 \xrightarrow{r} s''\}$ is finite.

$s = x$. Since Proc is the only rule that can be used to produce $x \xrightarrow{r} s'$, we have $\{s' \mid x \xrightarrow{r} s'\} \subseteq \{s'' \mid d(x) \xrightarrow{r} s''\}$ is finite since $d(x)$ is guarded.

The remaining cases are similar to previous ones. □

Lemma 3.2.4. *If $s \xrightarrow{\langle r,q\rangle} E$ then $\exists s': s \xrightarrow{r} s' \wedge s' \xrightarrow{q} E$.*

Proof. Induction on the depth of the proof tree of $s \xrightarrow{\langle r,q\rangle} E$. The last rule used is either

$$\frac{s \xrightarrow{r} s' \wedge s' \xrightarrow{q} E \wedge r \text{ is finite}}{s \xrightarrow{\langle r,q\rangle} E} \quad \text{or} \quad \frac{\forall i\colon s \xrightarrow{q_i} E \wedge \lim_i q_i = \langle r, q\rangle}{s \xrightarrow{\langle r,q\rangle} E}.$$

If the first is used we are done. Else $\exists N : \forall i > N : q_i = \langle r, q_i'\rangle \wedge \lim_i q_i' = q$. By induction we have that $\forall i > N : \exists s_i : s \xrightarrow{r} s_i \wedge s_i \xrightarrow{q_i'} E$. Since $\{s_i \mid s \xrightarrow{r} s_i\}$ is finite, there exists a subsequence $(q_{i_j}')_j$ and a statement s' such that $\forall j : s \xrightarrow{r} s' \wedge s' \xrightarrow{q_{i_j}'} E$. Now $s' \xrightarrow{q_{i_j}'} E$ and $\lim_j q_{i_j}' = q$ so (Cauchy-rule) $s' \xrightarrow{q} E$. So we have $s \xrightarrow{r} s' \wedge s' \xrightarrow{q} E$. $\square$

Lemma 3.2.5. $\forall s \in \mathscr{L} : \exists q \in Q_e : s \xrightarrow{q} E.$

Proof. First we show that $\forall s \in \mathscr{L} : \exists a \in A_e$: either $s \xrightarrow{a} E$ or $\exists s'$ with lower complexity than $s : s \xrightarrow{a} s'$. Induction on structure of s: for example $x \xrightarrow{e} E$ and if $s_1 \xrightarrow{a} s'$ with s' lower complexity than s_1 then $s_1;s_2 \xrightarrow{a} s's_2$ with $s';s_2$ lower complexity than $s_1;s_2$.

With this we can prove the lemma immediately with induction on the structural complexity of s. $\square$

Lemma 3.2.6. *If* $\exists (r_i)_i : \exists (t_i)_i : s \xrightarrow{r_i} t_i$ *and* r_i *is finite and* $\lim_i r_i = r$ *with* r *is infinite, then* $s \xrightarrow{r} E$.

Proof. Either $t_i = E$ and then $s \xrightarrow{r_i} E$ or $t_i \neq E$ and then, by the previous lemma $\exists q_i : t_i \xrightarrow{q_i} E$. If we define $(q_i')_i$ by r_i in the first case and by $\langle r_i, q_i\rangle$ in the second case we have $\forall i : s \xrightarrow{q_i'} E$ and $\lim_i q_i' =$ (since r is infinite) $\lim_i r_i = r$ so by the Cauchy-rule $s \xrightarrow{r} E$. $\square$

Lemma 3.2.7. *Let* $r \in R_e$.

(a) $a \xrightarrow{r} E \Leftrightarrow r = a$,

(b) $x \xrightarrow{r} E \Leftrightarrow d(x) \xrightarrow{r} E$ or $r = e$,

(c) $s_1;s_2 \xrightarrow{r} E \Leftrightarrow s_1 \xrightarrow{r} E \wedge r$ *is infinite*,

(d) $s_1 + s_2 \xrightarrow{r} E \Leftrightarrow s_1 \xrightarrow{r} E$ *or* $s_2 \xrightarrow{r} E$,

(e) $s_1 \| s_2 \xrightarrow{r} E \Leftrightarrow s_1 \xrightarrow{r} E \wedge r$ *is infinite or* $s_2 \xrightarrow{r} E \wedge r$ *is infinite or*

$$\exists q_1, q_2 \in Q_e\colon \quad r = \{q_1, q_2\} \wedge s_1 \xrightarrow{q_1} E \wedge s_2 \xrightarrow{q_2} E.$$

Proof. We only prove part (e), the other parts being easier.

($\Leftarrow$) If $s_1 \xrightarrow{r} E \wedge r$ is infinite then $s_1 \| s_2 \xrightarrow{r} s_2 \wedge r$ is infinite so $s_1 \| s_2 \xrightarrow{r} E$ by the Inf-rule. The case $s_2 \xrightarrow{r} E \wedge r$ is infinite is analogous. If $s_1 \xrightarrow{q_1} E \wedge s_2 \xrightarrow{q_2} E$ then $s_1 \| s_2 \xrightarrow{\{q_1,q_2\}} E$ by True Par.

($\Rightarrow$) Induction on the depth of the proof tree for $s_1 \| s_2 \xrightarrow{r} E$. The last rule that is used to produce $s_1 \| s_2 \xrightarrow{r} E$ is either

$$\frac{s_1 \xrightarrow{q_1} E \wedge s_2 \xrightarrow{q_2} E}{s_1 \| s_2 \xrightarrow{\{q_1,q_2\}} E} \quad \text{or} \quad \frac{s_1 \| s_2 \xrightarrow{r} s' \wedge r \text{ is infinite}}{s_1 \| s_2 \xrightarrow{r} E}$$

$$\text{or} \quad \frac{\forall i\colon \quad s_1 \| s_2 \xrightarrow{q_i} E \wedge \lim_i q_i = r}{s_1 \| s_2 \xrightarrow{r} E}.$$

If the first one is used then we are ready.

Assume now that the second one is used. The only way to derive $s_1 \| s_2 \xrightarrow{r} s'$ is by Int Par so by $s_1 \xrightarrow{r} E$ or $s_2 \xrightarrow{r} E$ or $s_1 \xrightarrow{r} \bar{s}$ or $s_2 \xrightarrow{r} \bar{s}$. So we always have $s_1 \xrightarrow{r} E$ or $s_2 \xrightarrow{r} E$, since r is infinite.

The most difficult case is the case where the Cauchy-rule is used. So assume now that $(q_i)_i$ is a sequence such that $\forall i$: $s_1 \| s_2 \xrightarrow{q_i} E$ and $\lim_i q_i = r$. Now $q_i \in R_e$ or $q_i \in R_e \times Q_e$, so there exists a subsequence $(q_{f(i)})_i$ such that either $\forall i$: $q_{f(i)} \in R_e$ or $\forall i$: $q_{f(i)} \in R_e \times Q_e$.

Case I. $\forall i$: $q_{f(i)} \in R_e$. Rename $q_{f(i)}$ by r_i. We have $\forall i$: $s_1 \| s_2 \xrightarrow{r_i} E$ and $\lim_i r_i = r$. By induction, we have for all i:

$$s_1 \xrightarrow{r_i} E \wedge r_i \text{ is infinite} \quad \text{or}$$

$$s_2 \xrightarrow{r_i} E \wedge r_i \text{ is infinite} \quad \text{or}$$

$$\exists q_i^1, q_i^2: \quad r_i = \{q_i^1, q_i^2\} \wedge s_1 \xrightarrow{q_i^1} E \wedge s_2 \xrightarrow{q_i^2} E.$$

So there exists a subsequence $(r_{g(i)})_i$ such that

$$\forall i: \quad s_1 \xrightarrow{r_{g(i)}} E \wedge r_{g(i)} \text{ is infinite} \quad \text{or}$$

$$\forall i: \quad s_2 \xrightarrow{r_{g(i)}} E \wedge r_{g(i)} \text{ is infinite} \quad \text{or}$$

$$\forall i: \quad \exists q_i^1, q_i^2: \quad r_{g(i)} = \{q_i^1, q_i^2\} \wedge s_1 \xrightarrow{q_i^1} E \wedge s_2 \xrightarrow{q_i^2} E.$$

Case Ia. $\forall i$: $s_1 \xrightarrow{r_{g(i)}} E$ and $r_{g(i)}$ is infinite. We have $\lim_i r_{g(i)} = \lim_i r_i = r$, so by the Cauchy-rule $s_1 \xrightarrow{r} E \wedge r$ is infinite.

Case Ib. $\forall i$: $s_2 \xrightarrow{r_{g(i)}} E$ and $r_{g(i)}$ is infinite: analogous.

Case Ic. $\forall i$: $\exists q_i^1, q_i^2$: $r_{g(i)} = \{q_i^1, q_i^2\} \wedge s_1 \xrightarrow{q_i^1} E \wedge s_2 \xrightarrow{q_i^2} E$. There exists a subsequence $(r_{h(g(i))})_i$ such that $(q_{h(i)}^1)_i$ is converging, say to q^1, and $(q_{h(i)}^2)_i$ is converging, say to q^2. By the Cauchy-rule, we have $s_1 \xrightarrow{q^1} E$ and $s_2 \xrightarrow{q^2} E$ and $r = \lim_i r_i = \lim_i r_{h(g(i))} = \{q^1, q^2\}$.

Case II. $\forall i$: $q_{f(i)} \in R_e \times Q_e$. Say $q_{f(i)} = \langle r_i, \bar{q}_i \rangle$. Since $\lim_i \langle r_i, \bar{q}_i \rangle = r$ we know that r is infinite and $\lim_i r_i = r$. By Lemma 3.2.4 we can deduce from $s_1 \| s_2 \xrightarrow{\langle r_i, \bar{q}_i \rangle} E$ that $\forall i$: $\exists \bar{s}_i$: $s_1 \| s_2 \xrightarrow{r_i} \bar{s}_i$ (and $\bar{s}_i \xrightarrow{\bar{q}_i} E$). So for all i either $\exists t_i$: $s_1 \xrightarrow{r_i} t_i$ or $\exists t_i$: $s_2 \xrightarrow{r_i} t_i$. Now take a subsequence $r_{g(i)}$ such that $\forall i$: $s_1 \xrightarrow{r_{g(i)}} t_i$ or $\forall i$: $s_2 \xrightarrow{r_{g(i)}} t_i$. Since $r_{g(i)}$ is finite, r is infinite and $\lim_i r_{g(i)} = r$, lemma 3.2.6 guarantees that $s_1 \xrightarrow{r} E$ or $s_2 \xrightarrow{r} E$. $\square$

3.3. Operational semantics

Let P, Q and R be the solution of the system of domain equations of the linear time/noninterleaved variety given in Section 2.2. From now on, we will no longer encounter the special action "e". So if we write down $s \xrightarrow{r} s'$, $s \xrightarrow{r} E$ or $s \xrightarrow{q} E$ then we mean that $r \in R$ and $q \in Q$. The "e" is still present in our system, but hidden: in order to derive some transition, we sometimes have to use the "e" temporarily.

Definition 3.3.1 (*Operational semantics*). Let $F:\mathscr{L}\to P$. We define the higher-order mapping $\Phi:(\mathscr{L}\to P)\to(\mathscr{L}\to P)$ and $\mathcal{O}:\mathscr{L}\to P$ by

$$\Phi(F)(s)=\{\langle r,q\rangle \mid \exists s'\colon s\xrightarrow{r}s' \text{ with } r\in R \text{ is finite and } q\in F(s')\}$$
$$\cup\{r\in R\mid s\xrightarrow{r}E\},$$
$$\mathcal{O}=\text{fixed-point of } \Phi.$$

We have to show firstly that $\Phi(F)(s)$ is closed and secondly that Φ is a contraction. This last fact is straightforward, so we only prove the following.

Proposition 3.3.2. *$\Phi(F)(s)$ is closed.*

Proof. Because of the Cauchy rule, we have $\{r\in R\mid s\xrightarrow{r}E\}$ is closed. Assume now that $\lim_i\langle r_i,q_i\rangle=q$ with $s\xrightarrow{r_i}s_i\wedge r_i\in R\wedge q_i\in F(s_i)$. Either $\lim_i r_i=q$ or $\exists N\colon\forall i>N\colon r_i=r_N$ and $\langle r_N,\lim_i q_i\rangle=q$.

Case $\lim_i r_i=q$. By Lemma 3.2.6 we have $s\xrightarrow{\lim_i r_i}E$ so $q=\lim_i r_i\in\{r\in R\mid s\xrightarrow{r}E\}$.

Case $\forall i>N\colon r_i=r_N$ and $\langle r_N,\lim_i q_i\rangle=q$. We have $\forall i>N\colon s\xrightarrow{r_N}s_i$. By image finiteness there exists a subsequence $(s_{i_j})_j$ and an $\bar{s}$ such that $\forall j\colon s_{i_j}=\bar{s}$. So $\forall j\colon q_{i_j}\in F(\bar{s})$ so $\lim_i q_i=\lim_j q_{i_j}\in F(\bar{s})$ so $q=\langle r_N,\lim_i q_i\rangle\in\Phi(F)(s)$. □

3.4. Denotational semantics

First we introduce a number of semantical operators on P.

Definition 3.4.1. We define $\bullet$, $\|$, $\Lsh$ $\colon Q\times Q\to P$ by

$$r\bullet q=\begin{cases}\{r\}, & l_Q(r)=\infty,\\ \{\langle r,q\rangle\}, & \text{otherwise},\end{cases}$$
$$\langle r,q'\rangle\bullet q=\{\langle r,\bar{q}\rangle\mid\bar{q}\in q'\bullet q\},$$
$$q_1\,\|\,q_2=\{\{q_1,q_2\}\}\cup(q_1 \mathbin{|\!\underline{\,\,}} q_2)\cup(q_2 \mathbin{|\!\underline{\,\,}} q_1),$$
$$r \mathbin{|\!\underline{\,\,}} q=\begin{cases}\{r\}, & l_Q(r)=\infty,\\ \{\langle r,q\rangle\}, & \text{otherwise},\end{cases}$$
$$\langle r,q'\rangle \mathbin{|\!\underline{\,\,}} q=\{\langle r,\bar{q}\rangle\mid\bar{q}\in q'\|q\}.$$

For $op=\bullet,\ \|,\ \mathbin{|\!\underline{\,\,}}$ we define $op\colon P\times P\to P$ by

$$p_1\ op\ p_2=\bigcup\{q_1\ op\ q_2\mid q_1\in p_1\wedge q_2\in p_2\}.$$

Notation: $p_1\odot p_2\stackrel{\text{df}}{=}\{\{q_1,q_2\}\mid q_1\in p_1\wedge q_2\in p_2\}$. Then we have $p_1\,\|\,p_2=(p_1\odot p_2)\cup p_1 \mathbin{|\!\underline{\,\,}} p_2\cup p_2 \mathbin{|\!\underline{\,\,}} p_1$

Remark 3.4.2. The above definitions need some justification. First of all the operators are defined in terms of themselves. By the use of contracting higher-order operators one can show that the above definitions make sense. Second we need to show that the result of $p_1\, op\, p_2$ is closed and nonempty. We will skip the proof here. For a comparable proof, see [10] and [16].

Definition 3.4.3 (Denotational semantics). Let $F:\mathscr{L}\rightarrow P$. We define the higher order mapping $\Psi:(\mathscr{L}\rightarrow P)\rightarrow(\mathscr{L}\rightarrow P)$ by

$$\Psi(F)(a)=\{a\},$$

$$\Psi(F)(s_1;s_2)=\Psi(F)(s_1)\bullet F(s_2),$$

$$\Psi(F)(s_1\|s_2)=\Psi(F)(s_1)\|\Psi(F)(s_2),$$

$$\Psi(F)(s_1+s_2)=\Psi(F)(s_1)\cup\Psi(F)(s_2),$$

$$\Psi(F)(x)=\Psi(F)(d(x)),$$

$$\mathscr{D}=\text{fixed-point of }\Psi.$$

This way of defining a denotational semantics is extensively discussed in [24] and in [13]. The well-definedness can be shown by induction on the structure of the statement, first for guarded statements g and then for general statements s. In order to prove that Ψ is a contraction, we need to have some properties of the semantical operators.

Proposition 3.4.4.

(1) $d_p(p_1\bullet p_1', p_2\bullet p_2')\leqslant\max\{d_P(p_1,p_2),\tfrac{1}{2}d_P(p_1',p_2')\}$,

(2) $d_P(p_1 \mathbin{\|\!\_} p_1', p_2 \mathbin{\|\!\_} p_2')\leqslant\max\{d_P(p_1,p_2),\tfrac{1}{2}d_P(p_1',p_2')\}$,

(3) $d_P(p_1\| p_1', p_2\| p_2')\leqslant\max\{d_P(p_1,p_2), d_P(p_1',p_2')\}$,

(4) $d_P(p_1\odot p_1', p_2\odot p_2')\leqslant\max\{\tfrac{1}{2}d_P(p_1,p_2),\tfrac{1}{2}d_P(p_1',p_2')\}$.

We want to ask for special attention for the $\frac{1}{2}$ in the fourth clause of this proposition. These factors are caused by the $id_{1/2}$ in the domain equations.

3.5. Operational semantics = denotational semantics

First we shall introduce an intermediate semantics $\mathscr{I}$ and prove that $\mathscr{I}=\mathscr{O}$. Next we shall give the proof of the equivalence of $\mathscr{O}$ and $\mathscr{D}$.

Definition 3.5.1 (*Intermediate semantics*). $\mathscr{I}(s)=\{q\in Q\mid s\xrightarrow{q}E\}$.

Lemma 3.5.2. $\forall s\in\mathscr{L}$: $\mathscr{I}(s)\neq\emptyset$.

We leave the verification of this lemma to the reader. For a comparable proof, see [16].

Lemma 3.5.3. $\mathcal{I} = \mathcal{O}$.

Proof

$$\begin{aligned}\mathcal{I}(s) &= \{q \in Q \mid s \xrightarrow{q} E\}\\ &= \{r \in R \mid s \xrightarrow{r} E\} \cup \{\langle r, q\rangle \in R \times Q \mid s \xrightarrow{\langle r,q\rangle} E\}\\ &= \{r \in R \mid s \xrightarrow{r} E\} \cup \{\langle r, q\rangle \mid \exists s' : s \xrightarrow{r} s', r \in R \text{ is finite, } s' \xrightarrow{q} E \text{ and } q \in Q\}\\ &= \{r \in R \mid s \xrightarrow{r} E\} \cup \{\langle r, q\rangle \mid \exists s' : s \xrightarrow{r} s', r \in R \text{ is finite and } q \in \mathcal{I}(s')\}\\ &= \Phi(\mathcal{I})(s).\end{aligned}$$

So $\Phi(\mathcal{I}) = \mathcal{I}$. Since also $\Phi(\mathcal{O}) = \mathcal{O}$ and Φ is a contraction, we have $\mathcal{I} = \mathcal{O}$. □

The next lemma almost says that $\Phi(\mathcal{D}) = \mathcal{D}$, which would be sufficient to prove $\mathcal{O} = \mathcal{D}$ immediately.

Lemma 3.5.4.

(1) $\Phi(\mathcal{D})(a) = \mathcal{D}(a)$,

(2) $\Phi(\mathcal{D})(s_1;s_2) = \Phi(\mathcal{D})(s_1) \bullet \mathcal{D}(s_2)$,

(3) $\Phi(\mathcal{D})(s_1 \| s_2) = (\Phi(\mathcal{D})(s_1) \mathbin{⫡} \mathcal{D}(s_2)) \cup (\Phi(\mathcal{D})(s_2) \mathbin{⫡} \mathcal{D}(s_1)) \cup (\mathcal{O}(s_1) \odot \mathcal{O}(s_2))$,

(4) $\Phi(\mathcal{D})(s_1 + s_2) = \Phi(\mathcal{D})(s_1) \cup \Phi(\mathcal{D})(s_2)$,

(5) $\Phi(\mathcal{D})(x) = \Phi(\mathcal{D})(d(x))$.

Proof. In this proof we indicate the use of Lemma 3.2.7 by a mark $*$ on the "=" sign: "$\overset{*}{=}$".

(1) $$\begin{aligned}\Phi(\mathcal{D})(a) &= \{\langle r, q\rangle \mid \exists s' : a \xrightarrow{r} s', r \in R \text{ is finite and } q \in \mathcal{D}(s')\}\\ &\quad \cup \{r \in R \mid a \xrightarrow{r} E\}\\ &\overset{*}{=} \{a\} = \mathcal{D}(a).\end{aligned}$$

(2) $$\begin{aligned}\Phi(\mathcal{D})(s_1;s_2) &= \{\langle r, q\rangle \mid \exists s' : s_1;s_2 \xrightarrow{r} s', r \in R \text{ is finite and } q \in \mathcal{D}(s')\}\\ &\quad \cup \{r \in R \mid s_1;s_2 \xrightarrow{r} E\}\\ &\overset{*}{=} \{\langle r, q\rangle \mid \exists s' : s_1 \xrightarrow{r} s', r \in R \text{ is finite and } q \in \mathcal{D}(s'; s_2)\}\\ &\quad \cup \{\langle r, q\rangle \mid s_1 \xrightarrow{r} E, r \in R \text{ is finite and } q \in \mathcal{D}(s_2)\}\end{aligned}$$

$$\cup \{r \in R \mid s_1 \xrightarrow{r} E \text{ and } r \text{ is infinite}\}$$

$$= \{\langle r, q\rangle \mid \exists s' : s_1 \xrightarrow{r} s',\ r \in R \text{ is finite and}$$
$$\exists q_1 \in \mathscr{D}(s') : \exists q_2 \in \mathscr{D}(s_2) : q \in q_1 \bullet q_2\}$$

$$\cup \{\langle r, q\rangle \mid s_1 \xrightarrow{r} E,\ r \in R \text{ is finite and } q \in \mathscr{D}(s_2)\}$$

$$\cup \{r \in R \mid s_1 \xrightarrow{r} E \text{ and } r \text{ is infinite}\}$$

$$= \bigcup \{\langle r, q_1\rangle \bullet q_2 \mid \exists s' : s_1 \xrightarrow{r} s',\ r \in R \text{ is finite, } q_1 \in \mathscr{D}(s') \text{ and}$$
$$q_2 \in \mathscr{D}(s_2)\}$$

$$\cup \bigcup \{r \bullet q \mid s_1 \xrightarrow{r} E,\ r \in R \text{ and } q \in \mathscr{D}(s_2)\}$$

$$= \bigcup \{q_1 \bullet q_2 \mid q_1 \in \Phi(\mathscr{D})(s_1) \text{ and } q_2 \in \mathscr{D}(s_2)\}$$

$$= \Phi(\mathscr{D})(s_1) \bullet \mathscr{D}(s_2).$$

(3) $$\Phi(\mathscr{D})(s_1) \mathbin{|\!\!\lfloor} \mathscr{D}(s_2) = \bigcup \{q_1 \mathbin{|\!\!\lfloor} q_2 \mid q_1 \in \Phi(\mathscr{D})(s_1) \text{ and } q_2 \in \mathscr{D}(s_2)\}$$

$$= \bigcup \{\langle r, q\rangle \mathbin{|\!\!\lfloor} q_2 \mid \exists s : s_1 \xrightarrow{r} s,\ r \in R \text{ is finite,}$$
$$q \in \mathscr{D}(s) \text{ and } q_2 \in \mathscr{D}(s_2)\}$$

$$\cup \bigcup \{r \mathbin{|\!\!\lfloor} q_2 \mid s_1 \xrightarrow{r} E,\ r \in R \text{ and } q_2 \in \mathscr{D}(s_2)\}$$

$$= \{\langle r, \bar{q}\rangle \mid \exists s : s_1 \xrightarrow{r} s,\ r \in R \text{ is finite and}$$
$$\exists q \in \mathscr{D}(s) : \exists q_2 \in \mathscr{D}(s_2) : \bar{q} \in q \| q_2\}$$

$$\cup \{\langle r, q_2\rangle \mid s_1 \xrightarrow{r} E,\ r \in R \text{ is finite and } q_2 \in \mathscr{D}(s_2)\}$$

$$\cup \{r \in R \mid s_1 \xrightarrow{r} E,\ r \in R \text{ is infinite}\}$$

$$= \{r, \bar{q}\rangle \mid \exists s : s_1 \xrightarrow{r} s,\ r \in R \text{ is finite and}$$
$$\bar{q} \in \mathscr{D}(s) \| \mathscr{D}(s_2)\}$$

$$\cup \{\langle r, q_2\rangle \mid s_1 \xrightarrow{r} E,\ r \in R \text{ is finite and } q_2 \in \mathscr{D}(s_2)\}$$

$$\cup \{r \in R \mid s_1 \xrightarrow{r} E,\ r \in R \text{ is infinite}\}$$

and

$$\mathscr{O}(s_1) \odot \mathscr{O}(s_2) = \mathscr{I}(s_1) \odot \mathscr{I}(s_2)$$
$$= \{\{q_1, q_2\} \mid s_1 \xrightarrow{q_1} E,\ s_2 \xrightarrow{q_2} E,\ q_1, q_2 \in Q\}$$

so

$$\begin{aligned}
\Phi(\mathscr{D})(s_1\|s_2) &= \{\langle r, q\rangle \mid \exists s' : s_1\|s_2 \xrightarrow{r} s',\ r\in R \text{ is finite and } q\in\mathscr{D}(s')\}\\
&\quad \cup \{r\in R \mid s_1\|s_2 \xrightarrow{r} E\}\\
&\stackrel{*}{=} \{\langle r, q\rangle \mid \exists s' : s_1 \xrightarrow{r} s',\ r\in R \text{ is finite and}\\
&\qquad q\in\mathscr{D}(s'\|s_2)\} \cup \text{symmetric case}\\
&\quad \cup \{\langle r, q\rangle \mid s_1 \xrightarrow{r} E,\ r\in R \text{ is finite and}\\
&\qquad q\in\mathscr{D}(s_2)\} \cup \text{symmetric case}\\
&\quad \cup \{\{q_1, q_2\} \mid s_1 \xrightarrow{q_1} E,\ s_2 \xrightarrow{q_2} E,\ q_1, q_2\in Q\}\\
&\quad \cup \{r\in R \mid s_1 \xrightarrow{r} E \text{ and } r \text{ is infinite}\} \cup \text{symmetric case}\\
&= (\Phi(\mathscr{D})(s_1) \mathbin{|\!\!\lfloor} \mathscr{D}(s_2)) \cup (\Phi(\mathscr{D})(s_2) \mathbin{|\!\!\lfloor} \mathscr{D}(s_1)) \cup (\mathscr{O}(s_1) \odot \mathscr{O}(s_2)).
\end{aligned}$$

(4)
$$\begin{aligned}
\Phi(\mathscr{D})(s_1+s_2) &= \{\langle r, q\rangle \mid \exists s' : s_1+s_2 \xrightarrow{r} s',\ r\in R \text{ is finite and } q\in\mathscr{D}(s')\}\\
&\quad \cup \{r\in R \mid s_1+s_2 \xrightarrow{r} E\}\\
&\stackrel{*}{=} \{\langle r, q\rangle \mid \exists s' : s_1 \xrightarrow{r} s',\ r\in R \text{ is finite and}\\
&\qquad q\in\mathscr{D}(s')\} \cup \text{symmetric case}\\
&\quad \cup \{r\in R \mid s_2 \xrightarrow{r} E\} \cup \text{symmetric case}\\
&= \Phi(\mathscr{D})(s_1) \cup \Phi(\mathscr{D})(s_2).
\end{aligned}$$

(5)
$$\begin{aligned}
\Phi(\mathscr{D})(x) &= \{\langle r, q\rangle \mid \exists s' : x \xrightarrow{r} s',\ r\in R \text{ is finite and } q\in\mathscr{D}(s')\}\\
&\quad \cup \{r\in R \mid x \xrightarrow{r} E\}\\
&\stackrel{*}{=} \{\langle r, q\rangle \mid \exists s' : d(x) \xrightarrow{r} s',\ r\in R \text{ is finite and } q\in\mathscr{D}(s')\}\\
&\quad \cup \{r\in R \mid d(x) \xrightarrow{r} E\}\\
&= \Phi(\mathscr{D})(d(x)). \quad \square
\end{aligned}$$

Because of the occurrences of $\mathscr{O}$, instead of $\mathscr{D}$, at two places of the right-hand side of the previous lemma, clause (3), we are not able to prove $d(\Phi(\mathscr{D}), \mathscr{D})=0$ immediately, but instead of this we are able to prove $d(\Phi(\mathscr{D}), \mathscr{D}) \leq \frac{1}{2}d(\mathscr{D}), \mathscr{O})$ which turns out to be sufficient.

Lemma 3.5.5. $d(\Phi(\mathcal{D}), \mathcal{D}) \leq \frac{1}{2}d(\mathcal{D}, \mathcal{O})$.

Proof. We show with induction on the structure of s, (first g) that $d(\Phi(\mathcal{D})(s), \mathcal{D}(s)) \leq \frac{1}{2}d(\mathcal{D}, \mathcal{O})$. The only cases that we prove here are $g = g;s$ and $g = g_1 \| g_2$; the other cases are easier or similar.

$$g = g;s\colon\ d(\Phi(\mathcal{D})(g;s), \mathcal{D}(g;s)) = d(\Phi(\mathcal{D})(g) \bullet \mathcal{D}(s), \mathcal{D}(g) \bullet \mathcal{D}(s))$$
$$\leq \max\{d(\Phi(\mathcal{D})(g), \mathcal{D}(g)), \tfrac{1}{2}d(\mathcal{D}(s), \mathcal{D}(s))\}$$
$$\leq \text{(by induction)} \tfrac{1}{2}d(\mathcal{D}, \mathcal{O}).$$

$$g = g_1 \| g_2\colon\ d(\Phi(\mathcal{D})(g_1 \| g_2), \mathcal{D}(g_1 \| g_2))$$
$$= d(\Phi(\mathcal{D})(g_1) \mathbin{\|\!\!\_} \mathcal{D}(g_2) \cup \Phi(\mathcal{D})(g_2) \mathbin{\|\!\!\_} \mathcal{D}(g_1) \cup \mathcal{O}(g_1) \odot \mathcal{O}(g_2),$$
$$\mathcal{D}(g_1) \mathbin{\|\!\!\_} \mathcal{D}(g_2) \cup \mathcal{D}(g_2) \mathbin{\|\!\!\_} \mathcal{D}(g_1) \cup \mathcal{D}(g_1) \odot \mathcal{D}(g_2))$$
$$\leq \max\{d(\Phi(\mathcal{D})(g_1), \mathcal{D}(g_1)), d(\Phi(\mathcal{D}(g_2), \mathcal{D}(g_2)),$$
$$\tfrac{1}{2}d(\mathcal{O}(g_1), \mathcal{D}(g_1)), \tfrac{1}{2}d(\mathcal{O}(g_2), \mathcal{D}(g_2))\}$$
$$\leq \tfrac{1}{2}d(\mathcal{D}, \mathcal{O}) \text{ by induction. } \square$$

Theorem 3.5.6. $\mathcal{O} = \mathcal{D}$.

Proof.

$$d(\mathcal{O}, \mathcal{D}) \leq d(\Phi(\mathcal{O}), \mathcal{D}) \leq \max\{d(\Phi(\mathcal{O}), \Phi(\mathcal{D})), d(\Phi(\mathcal{D}), \mathcal{D})\}$$
$$\leq \max\{\tfrac{1}{2}d(\mathcal{O}, \mathcal{D}), \tfrac{1}{2}d(\mathcal{O}, \mathcal{D})\} = \tfrac{1}{2}d(\mathcal{O}, \mathcal{D}),$$

so $\mathcal{O} = \mathcal{D}$. $\square$

4. Conclusions

The language considered in Section 3 does not include a notion of synchronization. The noninterleaved domains are not sufficient to handle synchronization. To demonstrate this, look at the following statement.

$$s \equiv (a;c) \| (b;(\bar{c} \| d)).$$

We assume here that a, b and d are internal actions and that c and $\bar{c}$ are communication actions, able to synchronize with each other. The process denoting this statement is shown in Fig. 6 (in pomset notation) where τ denotes successful synchronization.

The pomset is called the N-pomset in the literature (cf. for example [18]). The problem is that such a structure is not present in our domain. In fact we conjecture that it is not possible to define an appropriate domain by means of domain equations

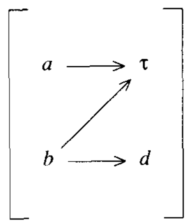

Fig. 6.

built from given sets (without any structure) and the usual constructors (described in the introduction and in Section 2.1). Therefore, we propose to combine the domain equation approach and the pomset approach. Let $\mathcal{POM}[A, P]$ denote the set of pomsets where the labels at level 1 come from the set A and the remaining labels are elements of P. Then the following system of domain equations might be appropriate to handle noninterleaved branching time concurrency with synchronization.

$$P \cong \mathcal{P}_{nc}(Q),$$

$$Q \cong \mathcal{POM}[A, P].$$

Future research is needed to investigate this domain.

The linear time variant $P \cong \mathcal{P}_{nc}(Q)$, $Q \cong \mathcal{POM}[A, Q]$ is isomorphic to $P \cong \mathcal{P}_{nc}(Q)$, $Q \cong \mathcal{POM}[A]$, where $\mathcal{POM}[A]$ denotes the set of all pomsets with labels in A. This domain was used in [16].

Acknowledgment

Vadim Kotov, responder to our paper [8], insisted that we should investigate how the metric approach may handle true concurrency. We acknowledge fruitful collaboration over the years with the members of ACG, in particular with Pierre America, Joost Kok and Jan Rutten, our primary co-authors on metric concurrency semantics. Moreover, we would like to thank Jan Rutten for his scrutinizing this text. We remain in debt to Maurice Nivat and Jeff Zucker for their seminal roles in the early stages of our work. We thank Jan Heering for discussions on the organization of the paper.

References

[1] P. America and J.W. de Bakker, Designing equivalent semantic models for process creation, *Theoret. Comput. Sci.* **60** (1988) 109–176.

[2] P. America, J.W. de Bakker, J.N. Kok and J.J.M.M. Rutten, Operational semantics of a parallel object-oriented language, in: *13th ACM Symp. on Principles of Programming Languages*, St. Petersburg, Florida (1986) 194–208.
[3] P. America, J.W. de Bakker, J.N. Kok and J.J.M.M. Rutten, Denotational semantics of a parallel object-oriented language, *Inform. and Comput.* **83** (1989) 152–205.
[4] P. America and J.J.M.M. Rutten, A parallel object-oriented language: design and semantic foundations, in: J.W. de Bakker, ed., *Languages for Parallel Architectures: Design, Semantics, Implementation Models*, Wiley Series in Parallel Computing (1989) 1–49.
[5] P. America and J.J.M.M. Rutten, Solving reflexive domain equations in a category of complete metric spaces, *J. Comput. System Sci.* **39** (3) (1989) 343–375.
[6] J.W. de Bakker, Semantics of infinite processes using generalized trees, in: J. Gruska, ed., *Proc. 6th Symp. Mathematical Foundations of Computer Science 1977* Lecture Notes in Computer Science **53** (Springer, Berlin, 1977) 240–246.
[7] J.W. de Bakker, Comparative semantics for flow of control in logic programming without logic, Report CS-R8840, Centre for Mathematics and Computer Science, Amsterdam, 1988; revised version to appear in *Inform. and Comput.*
[8] J.W. de Bakker, Designing concurrency semantics, in: G.X. Ritter, ed., *Proc. 11th World Computer Congress* (North-Holland, Amsterdam, 1989) 591–598 (invited address).
[9] J.W. de Bakker, ed., *Languages for Parallel Architectures: Design, Semantics, Implementation Models* (Wiley, New York, 1989).
[10] J.W. de Bakker, J.A. Bergstra, J.W. Klop and J.-J.Ch. Meyer, Linear time and branching time semantics for recursion with merge, *Theoret. Comput. Sci.* **34** (1984) 135–156.
[11] J.W. de Bakker and J.N. Kok, Comparative metric semantics for concurrent prolog, *Theoret. Comput. Sci.* **75** (1, 2) (1990) 15–43.
[12] J.W. de Bakker, J.N. Kok, J.-J.Ch. Meyer, E.-R. Olderog and J.I. Zucker, Contrasting themes in the semantics of imperative concurrency, in: J.W. de Bakker, W.P. de Roever, G. Rozenberg, eds., *Current Trends in Concurrency: Overviews and Tutorials*, Lecture Notes in Computer Science **224** (Springer, Berlin, 1986) 51–121.
[13] J.W. de Bakker and J.-J.Ch. Meyer, Metric semantics for concurrency, *BIT* **28** (1988) 504–529.
[14] J.W. de Bakker, J.-J.Ch. Meyer, E.-R. Olderog and J.I. Zucker, Transition systems, metric spaces and ready sets in the semantics of uniform concurrency, *J. Comput. System Sci.* **36** (1988) 158–224.
[15] J.W. de Bakker, W.P. de Roever and G. Rozenberg, eds., Linear time, branching time and partial order in logics and models for concurrency, in: *Proc. REX School/Workshop, Noordwijkerhout* (1988), Lecture Notes in Computer Science **354** (Springer, Berlin, 1989).
[16] J.W. de Bakker and J.H.A. Warmerdam, Metric pomset semantics for a concurrent language with recursion, in: I. Guessarian, ed., *Proc. 18ème Ecole de Printemps d'Informatique Theorique, Semantique du Parallelisme* Lecture Notes in Computer Science **469** (Springer, Berlin, 1990) 21–49.
[17] J.W. de Bakker and J.I. Zucker, Processes and the denotational semantics of concurrency, *Inform. and Control* **54** (1982) 70–120.
[18] G. Boudol and I. Castellani, Concurrency and atomicity, *Theoret. Comput. Sci.* **59** (1988) 25–84.
[19] F.S. de Boer, J.N. Kok, C. Palamidessi and J.J.M.M. Rutten, From failure to success: Comparing a denotational and a declarative semantics for Horn Clause Logic, in: M.Z. Kwiatkowska, M.W. Shields and R.M. Thomas, eds., *Proc. Internat. BCS-FACS Workshop on Semantics for Concurrency* Leicester (1990) Workshops in Computing (Springer, Berlin, 1990) 38–60.
[20] P. Degano, R. De Nicola and U. Montanari, Observational equivalences for concurrency models, in: M. Wirsing, ed., *Formal Description of Programming Concepts-III* (North-Holland, Amsterdam, 1987) 105–132.
[21] J. Dugundji, *Topology* (Allyn & Bacon, Rockleigh, MA, 1966).
[22] R. Engelking, *General Topology* (PWN, Warsaw, 1977).
[23] J.N. Kok, A compositional semantics for concurrent prolog, in: R. Cori, and M. Wirsing, eds., *Proc. Symp. on Theoretical Aspects of Computer Science* Lecture Notes in Computer Science **294** (Springer, Berlin, 1988) 373–388.
[24] J.N. Kok and J.J.M.M. Rutten, Contractions in comparing concurrency semantics, in: T. Lepistö and A. Salomaa, eds., *Proc. 15th ICALP* Lecture Notes in Computer Science **317** (Springer, Berlin, 1988) 317–332; *Theoret. Comput. Sci.* **76** (2, 3) (1990) 179–222.

[25] M. Nielsen, G.D. Plotkin and G. Winskel, Petri nets, event structures and domains, part I, *Theoret. Comput. Sci.* **13** (1981) 85–108.

[26] M. Nivat, Infinite words, infinite trees, infinite computations, in: J. W. de Bakker and J. van Leeuwen, eds., *Foundations of Computer Science* III.2, Mathematical Centre Tracts **109** (1979) 3–52.

[27] G.D. Plotkin, An operational semantics for CSP, in: D. Bjørner, ed., *Formal Description of Programming Concepts II* (North-Holland, Amsterdam, 1983) 199–223.

[28] V. Pratt, Modelling concurrency with partial orders, *Internat. J. Parallel Programming* **15** (1986) 33–71.

[29] J.J.M.M. Rutten, Semantic correctness for a parallel object-oriented language, *SIAM J. Comput.* **19**(3) (1990) 341–383.