1991

J.W. de Bakker, E.P. de Vink

CCS for OO and LP

Computer Science/Department of Software Technology Report CS-R9136 July

CWI, nationaal instituut voor onderzoek op het gebied van wiskunde en informatica

CWI is the research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a non-profit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands organization for scientific research (NWO).

CCS for OO and LP

J.W. de Bakker

CWI, Postbus 4079, NL-1009 AB Amsterdam
& Vrije Universiteit Amsterdam

E.P. de Vink

Department of Mathematics and Computer Science,
Vrije Universiteit Amsterdam, De Boelelaan 1081a, NL-1081 HV Amsterdam

ABSTRACT We illustrate the design of comparative continuation semantics for object-oriented and logic programming languages by three case studies dealing with process creation, backtracking and rendez-vous. Operational and denotational semantics involving syntactic and semantic continuations are proposed, and their equivalence is shown. For the rendez-vous concept, we present a somewhat streamlined version of our earlier work on the semantics of the parallel object-oriented language POOL. Throughout, the metric framework is exploited, and (unique fixed points of) contracting functions are used pervasively.

1985 Mathematics Subject Classification: 68Q10, 68Q55.
1991 Computing Reviews Categories: D.3.1, D.3.3, F.3.2, F.3.3.
Key Words and Phrases: Continuation semantics, concurrency, object-oriented programming, backtracking, rendez-vous, process creation, metric semantics.
Note: This report contains the slightly revised text of an invited address presented at TAPSOFT'91.
Note: The research of J.W. de Bakker was partially supported by ESPRIT Basic Research Action 3020: Integration.

1. INTRODUCTION

We shall present a selection of the work we have performed in recent years on the semantics of object-oriented (OO) and logic programming (LP) languages, in particular focusing on their control flow. As a unifying theme, we have singled out the use of *continuation semantics*. Moreover, we systematically *compare* operational and denotational models. Altogether, we shall be concerned with Comparative Continuation Semantics for OO and LP.

To position the present paper with respect to our earlier work, we start with a bit of history on the general framework we have developed. Since 1981, the Amsterdam Concurrency Group (ACG) has been investigating control flow semantics, with special emphasis on concurrency notions, and employing metric topology as main tool. 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 ($=_{df}$ 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.

Report CS-R9136
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

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 $f: (M_1, d_1) \to (M_2, d_2)$ bring points closer together: it is required that, for some real $\alpha \in [0, 1)$, $d_2(f(x), f(y)) \le \alpha \cdot d_1(x, y)$. Uniqueness of fixed points may conveniently be exploited in a variety of situations.

In the paper [BZ82] we showed how to apply metric techniques to solve domain equations

$$\mathbb{P} = \mathcal{F}(\mathbb{P}) \tag{1.1}$$

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

In a series of papers, starting with [BZ82, BBKM84, BKMOZ86, BM88, BMOZ88], we developed denotational ($\mathcal{D}$) and operational ($\mathcal{O}$) semantics for a number of simple languages with concurrency. Here a denotational semantics $\mathcal{D}$ for a language $\mathcal{L}$ is given as a mapping : $\mathcal{L} \to \mathbb{P}_1$ (for some $\mathbb{P}_1$ solving (1.1) for a suitable $\mathcal{F}$), which is *compositional* and treats recursion through fixed points. $\mathcal{O}$ is a mapping : $\mathcal{L} \to \mathbb{P}_2$, which is derived from some Plotkin-style transition system ([Pl83]), 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 ([ABKR86, AB88, ABKR89, AR89a, B89, R90a]). Throughout these studies, fruitful use was made of the metric formalism. Two further papers deserve special mention. In [AR89b], the technique from [BZ82] for solving domain equations (1.1) was generalised and phrased in the *category* of cms's. In [KR90], a powerful method was proposed to establish equivalences such as $\mathcal{O} = \mathcal{D}$, by (i) *defining* $\mathcal{O}$ as 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 [BM88], where several more examples of the KR-method are treated).

Logic programming and some of its parallel variations were first studied by ACG in [B88, K88]. The paper [B88] proposed to investigate control flow in LP abstracting from the logical complexities (no substitutions, refutations, etc.), and shows how the basic metric techniques apply as well to this, at first sight rather remote, problem area. Related work includes [BK90, BoKPR90].

Since 1989, we have been pursuing the research directions as outlined above as part of the ESPRIT Basic Research Action *Integrating the Foundations of Functional, Logic and Object-*

Oriented Programming. One of the tasks of this action is in particular devoted to the semantics of parallel OO and LP. Representative papers produced by it so far are [AR90, BoKPR91, JaMo90].

Now back to the aims of the present paper. We shall demonstrate the machinery of metric semantics by the investigation of two case studies. From parallel OO we take the notions of process creation and rendez-vous between processes. From (sequential) LP we consider the backtracking notion of *PROLOG*. In both cases we consider only the uniform or schematic version: the elementary actions remain atomic and are not supplied with some form of interpretation as state (or substitution) transformation. Also, both case studies serve as illustrations of more elaborate work reported elsewhere. The OO notions are based on our study of POOL as mentioned earlier; the LP part is an introduction to the paper [B88].

We conclude this introduction with an outline of the paper. In Section 2, we provide a brief summary of our metric tools, including a short discussion of the definition of suitable cms's as solution of metric domain equations (1.1). In Section 3 we illustrate our techniques by means of the discussion of a very simple language with as only notions elementary actions, sequential composition, nondeterministic choice, and recursion. (The reader may recognise here the control structure of context free grammars.) Operational and denotational semantics - both of the LT and BT variety - are developed for this language, and their equivalence is established. By way of preparation for the subsequent sections, the treatment is based on (syntactic and semantic) continuations. In the next section we deal with process creation. Compared with [BM88], some details missing there in the main equivalence proof have been added. Section 5 is devoted to backtracking. Originating with [DeBr86], this notion has also been studied extensively by De Bruin and De Vink, e.g. [BrVi89]. In this TAPSOFT 89 paper they also included a study of *PROLOG*'s cut operator (using cpo rather than metric techniques). Our paper culminates in Section 6 with the treatment of the rendez-vous construct. This is an abstracted (and considerably streamlined) version of the analysis of this notion in [ABKR89, R90a]. Firstly we propose a more convincing operational semantics. Next, in the design of the denotational semantics (which avoids some of the intricacies of [ABKR89] in the definition of the semantic parallel composition) and the ensuing equivalence proof ($\mathcal{O} = \mathcal{D}$) we exploit and advance the technique of using higher-order functions. Firstly, we provide a *simultaneous* definition of $\mathcal{D}$ and of the semantic operator(s) concerned. Secondly, we give an equivalence proof based on the principle of [KR90] combined with a refined complexity measure.

Acknowledgement. We are much indebted to the members of the ACG for fruitful cooperation over the years, especially to Pierre America, Arie de Bruin, Joost Kok, and Jan Rutten, co-authors of the papers to which the present one serves as an introduction. We thank Franck van Breugel for suggesting several corrections.

2. MATHEMATICAL PRELIMINARIES

This section is mainly devoted to a summary of the basic facts from metric topology which we need in the sequel.

2.1 Notations

We use the phrase: let $(x \in)X$ be a set such that $\cdots$ to introduce a set X with variable x ranging over X such that $\cdots$. With $\mathcal{P}(X)$ we denote the collection of all subsets of X, and with $\mathcal{P}_\pi(X)$ the collection of all subset of X which have property π. The notation $f: X \to Y$ expresses that f is a function with domain X and range Y. If $f: X \to X$ and $f(x) = x$ we call x a *fixed point* of f. If f has a unique fixed point, we denote it by $fix(f)$.

2.2 Metric spaces

DEFINITION 2.1 A *metric space* is a pair (M,d) with M a nonempty set and d a mapping $d: M \times M \to [0,1]$ (a metric or distance) that satisfies the following properties:

(i) $\forall x,y \in M$, : $d(x,y) = 0 \Leftrightarrow x = y$

(ii) $\forall x,y \in M$, : $d(x,y) = d(y,x)$

(iii) $\forall x,y,z \in M$: $d(x,z) \leq d(x,y) + d(y,z)$.

We call (M,d) an *ultrametric* space if the following stronger version of property (iii) is satisfied:

$(iii)'$ $\forall x,y,z \in M$: $d(x,z) \leq max\{\, d(x,y), d(y,z) \,\}$.

DEFINITION 2.2 Let (M,d) be a metric space, let $(x_i)_{i=0}^{\infty}$ (or $(x_i)_i$ for short) be a sequence in M.

a. We say that $(x_i)_i$ is a *Cauchy sequence* whenever we have $\forall \varepsilon > 0 \exists N \in \mathbb{N}\, \forall n, m > N$: $d(x_n, x_m) < \varepsilon$.

b. Let $x \in M$. We say that $(x_i)_i$ *converges* to x and call x the *limit* of $(x_i)_i$ whenever we have $\forall \varepsilon > 0 \exists N \in \mathbb{N}\, \forall n \geq N$: $d(x, x_n) < \varepsilon$. Such a sequence we call *convergent*. Notation: $lim_{i \to \infty}\, x_i = x$.

c. The metric space (M,d) is called *complete* whenever each Cauchy sequence converges to an element of M.

DEFINITION 2.3 Let (M_1,d_1), (M_2,d_2) be metric spaces.

a. We say that (M_1,d_1) and (M_2,d_2) are *isometric* if there exists a *bijection* $f: M_1 \to M_2$ such that $\forall x,y \in M_1$: $d_2(f(x), f(y)) = d_1(x,y)$. We then write $M_1 \cong M_2$.

b. Let $\alpha \geq 0$. With $M_1 \to^{\alpha} M_2$ we denote the set of all functions f from M_1 to M_2 that satisfy the following property: $\forall x,y \in M_1$: $d_2(f(x), f(y)) \leq \alpha \cdot d_1(x,y)$. Functions in $M_1 \to^{1} M_2$ we call *non distance increasing* (*ndi*), functions in $M_1 \to^{\alpha} M_2$ with $0 \leq \alpha < 1$ we call *contracting*.

THEOREM 2.4 *(Banach's fixed point theorem) Let (M,d) be a complete metric space (cms, for short). Then there exists $x \in M$ such that*

(i) $f(x) = x$ (x is a fixed point of f),

(ii) $\forall y \in M : f(y) = y \Rightarrow x = y$ (x is unique),

(iii) $\forall x_0 \in M : lim_{n \to \infty} f^n(x_0) = x$ where $f^{n+1}(x_0) = f(f^n(x_0)), f^0(x_0) = x_0$.

DEFINITION 2.5 A subset X of a metric space (M,d) is called *compact* whenever each sequence in X has a subsequence that converges to an element of X.

Each compact set X is *closed* (i.e., each Cauchy sequence in X converges to an element of X). The main role of the compactness property for our purposes is based on the theorems of Kuratowski ([Ku56]) and Michael ([Mic51]). The former states that the space of compact subsets (equipped with a suitable metric) of a complete space is itself complete. The latter is useful for showing the well-definedness of certain semantic operators (such as Definition 4.10; for more details on these issues which are somewhat glossed over in our paper cf. [Br91]).

DEFINITION 2.6 Let (M,d), (M_1,d_1), (M_2,d_2) be metric spaces.

a. We define a metric d_F on $M_1 \to M_2$ as follows: for every $f_1,f_2 \in M_1 \to M_2$, $d_F(f_1,f_2) = sup\{ d_2(f_1(x),f_2(x)) \mid x \in M_1 \}$. For $\alpha \geq 0$ the set $M_1 \to^{\alpha} M_2$ is a subset of $M_1 \to M_2$, and the metric on $M_1 \to^{\alpha} M_2$ can be obtained by taking the restriction of the corresponding d_F.

b. With $M_1 \bar{\cup} M_2$ we denote the disjoint union of M_1 and M_2, which can be defined as $\{ 1 \} \times M_1 \cup \{ 2 \} \times M_2$. We define a metric on $M_1 \bar{\cup} M_2$ as follows: for every $x,y \in M_1 \bar{\cup} M_2$, $d_U(x,y) = d_i(x,y)$ if $x,y \in \{ i \} \times M_i$, i=1 or 2, $d_i(x,y) = 1$ otherwise.

c. We define a metric d_P on $M_1 \times M_2$ by the following clause $d_P((x_1,x_2),(y_1,y_2)) = max\{ d_1(x_1,y_1), d_2(x_2,y_2) \}$.

d. Let $\mathcal{P}_{nc}(M) = \{ X \subseteq M \mid X \textit{ is compact and nonempty} \}$. We define a metric d_H on $\mathcal{P}_{nc}(M)$, called the Hausdorff distance, as follows: For every $X,Y \in \mathcal{P}_{nc}(M)$, $d_H(X,Y) = max\{ sup_{x \in X}\{ d(x,Y) \}, sup_{y \in Y}\{ d(y,X) \} \}$ where $d(w,Z) = inf_{z \in Z}\{ d(w,z) \}$, for every $Z \subseteq M$, $w \in M$.

   In $\mathcal{P}_{co}(M) = \{ X \subseteq M \mid X \textit{ compact} \}$ we also have the empty set as an element. We define d_H as above but extended with the following case: If $X \neq \varnothing$ then $d_H(X, \varnothing) = d_H(\varnothing, X) = 1$.

e. Let $\alpha \geq 0$. We define $id_{\alpha}(M,d) = (M,\alpha \cdot d)$.

THEOREM 2.7 *Let* (M,d), (M_1,d_1) (M_2,d_2), d_F, d_H, d_P *and* d_H *be as in Definition 2.6, and suppose that* (M,d), (M_1,d_1) (M_2,d_2) *are complete. We have that*

a. $(M_1 \to M_2, d_F)$, $(M_1 \to^{\alpha} M_2, d_F)$,

b. $(M_1 \bar{\cup} M_2, d_U)$,

c. $(M_1 \times M_2, d_P)$,

d. $(\mathcal{P}_{nc}(M), d_H)$, $(\mathcal{P}_{co}(M), d_H)$

are complete metric spaces. If (M,d) *and* (M_i,d_i), $i=1,2$, *are ultrametric spaces then these composed spaces are again ultrametric.*

The proof of Theorem 2.7, parts a, b, c are straightforward. Part d is more involved. It can be proved with the help of the following characterisation:

THEOREM 2.8 *Let* $(\mathcal{P}_{co}(M),d_H)$ *be as in Definition 2.6. Let* $(X_i)_i$ *be a Cauchy sequence in* $\mathcal{P}_{co}(M)$. *We have* $lim_i\, X_i = \{ lim_i\, x_i \mid x_i \in X_i \wedge (x_i)_i \textit{ a Cauchy sequence in } M \}$.

The proofs of Theorem 2.7d and Theorem 2.8 are due to Kuratowski ([Ku56]), as a generalisation of a similar result for *closed* subsets, see e.g. [Ha48].

The following alternative definition of the Hausdorff distance is sometimes convenient:

LEMMA 2.9 *Let* $\tilde{d}_H(X,Y) = inf\{ \varepsilon \mid \forall x \in x \exists y \in Y : d(x,y)<\varepsilon,\ \forall y \in Y \exists x \in X : d(y,x)<\varepsilon \}$. *Then* $\tilde{d}_H = d_H$.

We conclude this subsection with the important

THEOREM 2.10 *(Michael) Let* $X \in \mathcal{P}_{co}(\mathcal{P}_{co}(M))$. *Then* $\cup X \in \mathcal{P}_{co}(M)$.

2.3 Sets of sequences

Let A be a finite alphabet, and let $A^\infty = A^* \cup A^\omega$ consist of the set of all finite and infinite words over A. We define metrics on A and A^∞ in

DEFINITION 2.11

a. On A we define the *discrete* metric d_A: for all $x,y \in A$, $d_A(x,y) = 0$ if $x = y$, $d_A(x,y) = 1$ otherwise.

b. Let, for $x \in A^\infty$, $x(n)$ denote the prefix of x of length n, if *length*$(x) \geq n$, and x otherwise. We put $d(x,y) = 2^{-sup\{\, n \mid x(n) = y(n) \,\}}$ with the convention that $2^{-\infty} = 0$.

We have

LEMMA 2.12

a. (A^∞, d) *is a complete ultrametric space.*

b. $(\mathcal{P}_{nc}(A^\infty), d_H)$ *is a complete ultrametric space.*

c. *Let, for* $X \in \mathcal{P}_{nc}(A^\infty)$, $X(n) = \{\, x(n) \mid x \in X \,\}$. *Then* $d_H(X,Y) = 2^{-sup\{\, n \mid X(n) = Y(n) \,\}}$.

The space $\mathcal{P}_{nc}(A^\infty)$ will be used extensively in the sequel.

2.4 Domain equations

In [BZ82], [AR89b], a method has been developed to determine complete (ultra)metric spaces as solutions of domain equations of the form

$$\mathbb{P} = \mathcal{F}(\mathbb{P}), \tag{2.1}$$

where $\mathcal{F}$ is a functor (on a category of cms's) satisfying certain conditions. Natural examples of $\mathcal{F}$ are obtained by building it in terms of the operations on metric spaces encountered in Definition 2.6. We shall restrict ourselves here to the discussion of only one example of (2.1). For the general theory we refer to [AR89b]. Several more intricate examples may be encountered in [ABKR89], [AR90].

We shall be concerned with $(\mathbb{P},d)$ - or $\mathbb{P}$, for short, - solving the domain equation

$$\mathbb{P} = \{\, p_0 \,\} \,\bar{\cup}\, \mathcal{P}_{co}(A \times id_{1/2}(\mathbb{P})). \tag{2.2}$$

Elements p in $\mathbb{P}$ are usually called *processes*. The equation (2.2) assumes the discrete metric on $\{p_0\}$ (consisting of the *nil* process p_0 only) and on A, and, moreover, the various metrics (for $\bar{\cup}$, $\times$, $\mathcal{P}_{co}$, $id_{1/2}$) as defined earlier. As a consequence, the metric d on (the non-nil processes in) $\mathbb{P}$ equals the Hausdorff metric d_H induced by the following metric $\bar{d}$ on $A \times \mathbb{P}$:

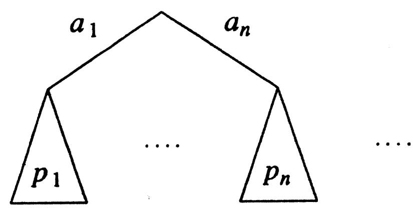
$$\begin{aligned} \bar{d}(\langle a_1,p_1 \rangle, \langle a_2,p_2 \rangle) &= \tfrac{1}{2} d(p_1,p_2) \quad \text{if } a_1 = a_2 \\ &= 1 \quad\quad\quad\quad\; \text{otherwise}. \end{aligned}$$

The metric d_H may, alternatively, be characterised by

$$d_H(p_1,p_2) = 2^{-sup\{\, n \mid p_1(n) = p_2(n) \,\}}$$

where $p(n)$, the *truncation* of process p at depth n, is defined by $p_0(n) = p_0$, and, for $p \neq p_0$, $p(0) = \varnothing$, $p(n+1) = \{\, \langle a,p'(n) \rangle \mid \langle a,p' \rangle \in p \,\}$.

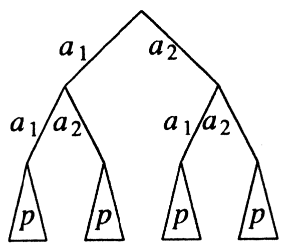
A process $p \in \mathbb{P}$ can be viewed as a tree-like object. It is either the nil process (which terminates normally) or the empty set (which models abnormal termination or *deadlock*), or it consists of a nonempty set of pairs $\langle a,p' \rangle$ which represent all possible steps a that a process p can take (each followed by its *resumption* p', itself another process). In a picture, a process $p(\neq p_0, \varnothing)$ may be drawn in a tree-like fashion:



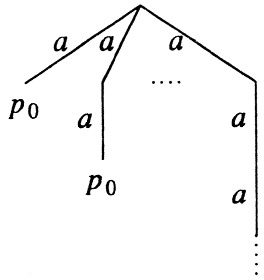
, where each p_i is either p_0, $\varnothing$ or another such 'tree'. Each 'tree' is *commutative, absorptive* (the successors of any node form a set rather than a multiset) and *compact.*

EXAMPLES

1. p_0, $\varnothing$, $\{\langle a_1,\{\langle a_2,p_0\rangle,\langle a_3,p_0\rangle\}\rangle\}$, $\{\langle a_1,\{\langle a_2,p_0\rangle\}\rangle,\langle a_1,\{\langle a_3,p_0\rangle\}\rangle\}$.
2. The process p determined by $p = lim_i\, p_i$, $p_{i+1} = \{\langle a_1,p_i\rangle,\langle a_2,p_i\rangle\}$ (note that p satisfies $p = \{\langle a_1,p\rangle,\langle a_2,p\rangle\}$. This p may be depicted as



3. Let us, informally, define the operation of sequential composition $\circ : \mathbb{P} \times \mathbb{P} \to \mathbb{P}$ by putting: $p_1 \circ p_2$ is the process obtained by replacing, in p_1, all 'leaves' p_0 by p_2. Now let p be defined as the process satisfying $p=\{\langle a,p_0\rangle\} \cup (p \circ \{\langle a,p_0\rangle\})$. Since p is compact (hence closed), it must include the infinite branch $\{\langle a,\{\langle a, \cdots\rangle\}\rangle\}$.



(Warmerdam has shown (personal communication) that the operation of sequential composition sketched above is not well-defined if processes are only required to be closed and infinite alphabets are allowed.)

We conclude with two more remarks on processes.

REMARK Let us call two processes p_1, p_2 *bisimilar* if there exists a *bisimulation* R such that $p_1 R p_2$. Here a bisimulation is a relation R on $\mathbb{P} \times \mathbb{P}$ satisfying

(i) If $p_0 Rp$ or pRp_0 then $p = p_0$

(ii) If $p_1 R p_2$ and $\langle a,p' \rangle \in p_1$, then there exists $\langle a,p'' \rangle \in p_2$ such that $p'Rp''$.

(iii) If $p_1 R p_2$ and $\langle a,p'' \rangle \in p_2$ then there exists $\langle a,p' \rangle \in p_2$ such that $p'Rp''$.

Now an important property of the domain $\mathbb{P}$ is the fact that two processes are bisimilar iff they are equal. For more information about processes and bisimulation cf. [BeK87], [Ru90b].

REMARK LT (sets of sequences) - domains may as well be obtained as solution of (systems of) domain equations. Let $\mathbb{P}_1 = \{ \{\varepsilon\} \} \cup \mathcal{P}_{nc}(A^+ \cup A^\omega)$. This domain (which is almost as $\mathcal{P}_{nc}(A^\infty)$ of Subsection 2.3, the only difference being that, for $p \in \mathbb{P}_1$, if $\varepsilon \in p$, then $p = \{\varepsilon\}$) is isometric to $\mathbb{P}_2$ which is (the first component of) the solution of the system of equations

$$
\begin{aligned}
\mathbb{P} &= \{p_0\} \,\bar{\cup}\, \mathcal{P}_{nc}(\mathbb{Q}) \\
\mathbb{Q} &= p_0 \,\bar{\cup}\, (A \times id_{1/2}(\mathbb{Q})) \\
p_0 &= \{\varepsilon\}
\end{aligned}
\tag{2.3}
$$

This way of defining $\mathbb{P}_1$ in terms of the isometric $\mathbb{P}_2$ may bring out the (dis)similarities between the BT-domain $\mathbb{P}$ (solving equation (2.2)) and the LT-domain $\mathbb{P}_1$ (or $\mathbb{P}_2$).

3. BASIC CONTROL FLOW

As a means to introduce our techniques in an elementary setting, we use a very simple language featuring elementary actions, sequential composition, nondeterministic choice, and recursion. We baptize this language $\mathcal{L}_{cf}$: a program in $\mathcal{L}_{cf}$ has the same expressive power as a context free grammar (generating languages with finite *and infinite* words). For $\mathcal{L}_{cf}$ we shall introduce operational ($\mathcal{O}$) and denotational ($\mathcal{D}$) semantics, both of the *linear time* (LT) and *branching time* (BT) variety (for the latter one also uses the name *bisimulation* semantics). Throughout, we shall use (syntactic and semantic) *continuations*. For the present language this is convenient but not essential. Our reason for employing these techniques here is to prepare the way for their use in the three remaining sections, where continuations are indeed crucial. *Recursion* will be handled by fixed point techniques, in particular through fixed points of contracting functions. We present two alternatives, one based on fixed points of environment transformations, the other defining the denotational meaning function $\mathcal{D}$ itself as fixed point of a contracting higher-order mapping Ψ. The operational semantics $\mathcal{O}$ will - both for the LT and BT case - be derived from a Plotkin-style transition system $\mathcal{T}$. A contracting higher-order mapping Φ will be associated with $\mathcal{T}$, and the operational semantics $\mathcal{O}$ - which may be viewed as a means to collect all steps determined by $\mathcal{T}$ for a given program - is obtained as fixed point of this Φ. Moreover, we shall prove - following the approach of [KR90] - that $\mathcal{D}$ (or, technically, a related function involving $\mathcal{D}$) is as well a fixed point of Φ, thus obtaining $\mathcal{O} = \mathcal{D}$ as a corollary. (Incidentally, for the LT-setting this yields a new proof of Nivat's equivalence result described in [Ni77,78], which in turn generalises the classical Chomsky-Schutzenberger theorem for (finitary) context free languages.)

Altogether $\mathcal{L}_{cf}$ - though itself a language without advanced control flow notions - will be used as a tool to illustrate the convenience and power of the metric framework in control flow semantics.

3.1 Syntax

Throughout the paper we use a self-explanatory BNF-like notation for syntactic definitions. We start with the introduction of two basic syntactic sets:

- $(a \in) A$, the alphabet of *elementary actions*,
- $(x \in) PVar$, the alphabet of *procedure variables*.

DEFINITION 3.1

a. The class $(s \in) \mathcal{L}_{cf}$ of *statements* is given by $s ::= a \mid x \mid s_1 ; s_2 \mid s_1 + s_2$.

b. The class $(g \in) \mathcal{L}_{cf}^{g}$ of *guarded statements* is given by $g ::= a \mid g;s \mid g_1 + g_2$.

c. The class $(d \in) Decl_{cf}$ of *declarations* consists of mappings $d : PVar \rightarrow \mathcal{L}_{cf}^{g}$.

d. A *program* is a pair (d,s).

REMARK The guardedness (or 'Greibach') condition ensures that the execution of each procedure body (each $d(x)$, for $x \in PVar$) starts with the execution of an elementary action (rather than of another procedure variable). Technically this condition yields *contractivity* of an associated function (Lemma 3.15e or 3.17f).

EXAMPLES Take $PVar = \{ x \}$, $A = \{ a,b,c, \cdots \}$, and write $x \Leftarrow g$ for $d(x) = g$. Possible programs are: $(x \Leftarrow a;x+b\, , c;x)$ (with intended meaning $ca^{\omega} + ca^{*}b$), or $(x \Leftarrow a;x;b+c\, , x)$ (with intended meaning $\{ a^n cb^n \mid n \geq 0 \} \cup \{ a^{\omega} \}$). The construct $(x \Leftarrow x;b+a\, , x)$ is not a program, since $x;b+a \notin \mathcal{L}_{cf}^{g}$.

In this and subsequent sections we shall extensively use both syntactic and semantic continuations. The former - to be introduced here - are to play a role in the operational semantics definitions, and the latter in the denotational ones.

DEFINITION 3.2 Let E be a new symbol (standing for *termination*). The class $(r \in) R$ of syntactic continuations is given by $r ::= \mathrm{E} \mid (s;r)$ where $s \in \mathcal{L}_{cf}$.

Parentheses in $(s;r)$ will often be dropped when no ambiguity arises.

3.2 LT-operational semantics

We first introduce the complete metric space which will be used as range for the operational semantics:

DEFINITION 3.3

a. Let $(u,v \in) A^{\infty} =_{df} A^{*} \cup A^{\omega}$, where A is the alphabet from Subsection 3.1. (A^{∞},d) is the cms as introduced in Section 2. Let $\cdot$ be the operation of *prefixing* on $A \times A^{\infty}$, defined by $a \cdot u =_{df} au$.

b. Let $(p \in)\ \mathbb{P} = \mathcal{P}_{nc}(A^{\infty})$ be the family of all *nonempty compact* subsets of A^{∞}, equipped with the Hausdorff metric d_H with respect to the metric d of part a. Let $a \cdot p = \{ a \cdot u \mid u \in p \}$.

The operational semantics $\mathcal{O}$ mapping programs to elements from $\mathbb{P}$ will here and subsequently be given based on a *labeled transition system* $\mathcal{T}$. $\mathcal{T}$ determines a transition relation $\mathcal{R}$ which is given as the least relation satisfying (in the natural way) its axioms and rules.

DEFINITION 3.4 The transition system $\mathcal{T}_{cf}$ and associated relation $\mathcal{R}_{cf}$ are given as follows:

a. A *transition* is a four-tuple (r_1, a, d, r_2) in $R \times A \times Decl_{cf} \times R$; we usually write it as $r_1 \to_d^a r_2$.

b. The axiom and rules of $\mathcal{T}_{cf}$ are as follows:

- $a;r \to_d^a r$ (el.action)

- $$\frac{g;r \to_d^a \tilde{r}}{x;r \to_d^a \tilde{r}},\ d(x) = g$$ (recursion)

- $$\frac{s_1;(s_2;r) \to_d^a \tilde{r}}{(s_1;s_2);r \to_d^a \tilde{r}}$$ (seq.comp)

- $$\frac{s;r \to_d^a \tilde{r}}{\begin{array}{l}(s+\bar{s});r \to_d^a \tilde{r}\\ (\bar{s}+s);r \to_d^a \tilde{r}\end{array}}$$ (choice)

CONVENTION 1. Rules with the same premise and different conclusions are combined in a self-explanatory notation. Cf. the choice rule. 2. In the notation $\to_d^a$, the subscript d will sometimes be suppressed. 3. Instead of $r_1 \to_d^a r_2 \in \mathcal{R}_{cf}$, we simply write $r_1 \to_d^a r_2$ when $\mathcal{R}_{cf}$ (or its successors in subsequent sections) is understood.

LEMMA 3.5 *$\mathcal{T}_{cf}$ is finitely branching, i.e., for each r, the set* $\{ (a, r') \mid r \to^a r' \}$ *is finite.*
PROOF Direct from the definition of $\mathcal{T}_{cf}$. □

REMARK Note that, without the guardedness condition, Lemma 3.5 does not hold.

In the technical arguments in this and subsequent sections (in particular in establishing $\mathcal{O} = \mathcal{D}$) we shall often use (i) an auxiliary relation '$\twoheadrightarrow$' between syntactic continuations and (ii) the *complexity* $c_r(r)$, where $c_r : R \to \mathbb{N}$.

DEFINITION 3.6 We define the relation $\twoheadrightarrow$ to hold between r_1, r_2 if there is a rule (in the corresponding transition system) of the form

$$\frac{r_2 \to^a \tilde{r}}{r_1 \to^a \tilde{r}}.$$

The relation $r_1 \twoheadrightarrow r_2$ may be read as: in order to execute r_1, find out how to execute r_2. Next, we introduce the complexity of the elements in R and $\mathcal{L}_{cf}$:

DEFINITION 3.7

a. $c_r : R \to \mathbb{N}$ is given by $c_r(\mathrm{E}) = 0$, $c_r(s;r) = c_s(s)$.

b. $c_s : \mathcal{L}_{cf} \to \mathbb{N}$ is given by $c_s(a) = 1$, $c_s(x) = c_s(d(x)) + 1$, $c_s(s_1;s_2) = c_s(s_1) + 1$, $c_s(s_1 + s_2) = c_s(s_1) + c_s(s_2) + 1$.

We have

LEMMA 3.8

a. c_r, c_s *are well-defined.*

b. *If* $r_1 \twoheadrightarrow r_2$ *then* $c_r(r_1) > c_r(r_2)$.

PROOF Well-definedness of c_s is proved by induction on the syntactic complexity of first g then s. Part b is clear form the definitions. □

We now define the mapping $\mathcal{O}_d : R \to \mathbb{P}$ as fixed point of a higher-order function Φ_d which maps meanings to meanings:

DEFINITION 3.9 Let $F \in R \to \mathbb{P}$. The mapping $\Phi_d : (R \to \mathbb{P}) \to (R \to \mathbb{P})$ is given by

$$\Phi_d(F)(\mathrm{E}) = \{\varepsilon\},$$

$$\Phi_d(F)(r) = \cup\{ a \cdot F(r') \mid r \to_d^a r' \}, \quad \text{if } r \neq \mathrm{E}.$$

LEMMA 3.10

a. $\Phi_d(F)(r)$ *is nonempty and compact for each* F, r.

b. Φ_d *is contracting in* F.

PROOF Part a follows from the fact that $\mathcal{T}_{cf}$ is finitely branching (Lemma 3.5); part b is direct from the definition of Φ_d and elementary properties of the Hausdorff metric. □

At last, we are ready to define

DEFINITION 3.11

a. $\mathcal{O}_d = \mathit{fix}(\Phi_d)$.

b. $\mathcal{O}(d,s) = \mathcal{O}_d(s;\mathrm{E})$.

3.3 BT-operational semantics

Only minor changes have to be made in the definitions of the previous subsection to obtain the BT-operational semantics. Let us use the superscript b to indicate the BT-variant of the various definitions. Thus, we shall define $\mathcal{O}_d^b : \mathcal{L}_{cf} \to \mathbb{P}^b$, etc. The main step is the change in the range of the operational semantics (now $\mathbb{P}^b$ rather than $\mathbb{P}$):

DEFINITION 3.12 Let $\mathbb{P}^b$ be the cms which solves the domain equation (2.2):

$$\mathbb{P} = \{p_0\} \,\bar{\cup}\, \mathcal{P}_{co}(A \times id_{1/2}(\mathbb{P})).$$

For more information on $\mathbb{P}^b$ we refer to Subsection 2.4. We proceed with the definition of $\mathcal{O}_d^b$. There are no changes in $\mathcal{T}_{cf}$ (or $\mathcal{R}_{cf}$). The only change we adopt is in the definition of (the new) $\mathcal{O}_d^b$:

DEFINITION 3.13 Let $F \in R \to \mathbb{P}^b$. The function $\Phi_d^b : (R \to \mathbb{P}^b) \to (R \to \mathbb{P}^b)$ is given by

$$\Phi_d^b(F)(\mathrm{E}) = p_0,$$

$$\Phi_d^b(F)(r) = \{ \langle a, F(r') \rangle \mid r \to_d^a r' \}, \quad \text{if } r \neq \mathrm{E}.$$

Note the crucial difference between the second clause in this definition, and that of Definition 3.9, where $\cup\{ a \cdot F(r') \mid \cdots \}$ is used. In the latter, outcomes $a \cdot p_1, a \cdot p_2, \cdots$ are set-theoretically united to yield the result $a \cdot (p_1 \cup p_2 \cup \cdots)$, whereas in the present domain outcomes $\langle a, p_1 \rangle, \langle a, p_2 \rangle, \cdots$ are collected into the (compact) set $\{ \langle a, p_1 \rangle, \langle a, p_2 \rangle, \cdots \}$, rather than united in the form

$\{\langle a,p_1\cup p_2\cup\cdots\rangle\}$! A simple example may clarify the situation: $\mathcal{O}_d(a_1;(a_2+a_3);E)$ = $\mathcal{O}_d(((a_1;a_2)+(a_1;a_3));E)$ = $\{a_1a_2,a_1a_3\}$, whereas $\mathcal{O}_d^b(a_1;(a_2+a_3);E)$ = $\{\langle a_1,\{\langle a_2,p_0\rangle,\langle a_3,p_0\rangle\}\rangle\}$ and $\mathcal{O}_d^b(((a_1;a_2)+(a_1;a_3));E)$ = $\{\langle a_1,\{\langle a_2,p_0\rangle\}\rangle,\langle a_1,\{\langle a_3,p_0\rangle\}\rangle\}$.

3.4 Denotational semantics

We devote most of this subsection to the development of the LT-denotational semantics for $\mathcal{L}_{cf}$; at the end of it, we discuss what variations are required to obtain a BT-denotational model. We shall employ semantic continuations as counterpart of the earlier syntactic ones. Also, we shall provide two ways of handling recursion, one through (fixed points of) environment transformations, the second one using another (besides Φ_d) higher-order mapping from meanings to meanings.

We start with the introduction of the set of *environments* $(\eta\in)\ Env = PVar \to \mathbb{P} \to^1 \mathbb{P}$. In the following (and many subsequent) definitions we suppress most of the parentheses. If deemed necessary, they may be restored on the basis of the types of the mappings involved.

DEFINITION 3.14 (denotational semantics for $\mathcal{L}_{cf}$, first definition)

a. The mapping $\mathcal{S}:\mathcal{L}_{cf}\to Env\to\mathbb{P}\to\mathbb{P}$ is given by

$$\begin{aligned}\mathcal{S}\,a\,\eta p &= a\cdot p\\ \mathcal{S}\,x\,\eta p &= \eta x p\\ \mathcal{S}(s_1;s_2)\eta p &= \mathcal{S}s_1\eta(\mathcal{S}s_2\eta p)\\ \mathcal{S}(s_1+s_2)\eta p &= (\mathcal{S}s_1\eta p)\cup(\mathcal{S}s_2\eta p).\end{aligned}$$

b. H_d: $Env\to Env$ is given by $H_d\eta=\lambda x.\mathcal{S}\,d(x)\,\eta$.

c. $\eta_d = fix(H_d)$, $\mathcal{D}(d,s)=\mathcal{S}\,s\,\eta_d\,\{\varepsilon\}$.

The above definitions are justified in

LEMMA 3.15

a. $\mathcal{S}s\eta\in\mathbb{P}\to^1\mathbb{P}$.

b. $\mathcal{S}s\in Env\to^1\mathbb{P}\to^1\mathbb{P}$.

c. $\mathcal{S}g\eta\in\mathbb{P}\to^{1/2}\mathbb{P}$

d. $\mathcal{S}g\in Env\to^{1/2}\mathbb{P}\to^{1/2}\mathbb{P}$.

e. $H_d\in Env\to^{1/2} Env$.

f. $\mathcal{D}(d,x)=\mathcal{D}(d,d(x))$.

PROOF Simpler than that of lemma 4.13 and therefore omitted. □

We next turn to the definition of $\mathcal{D}$ as fixed point of a higher-order mapping:

DEFINITION 3.16 (denotational semantics for $\mathcal{L}_{cf}$, second definition)

a. Let $F\in\mathcal{L}_{cf}\to\mathbb{P}\to^1\mathbb{P}$. The function $\Psi_d:(\mathcal{L}_{cf}\to\mathbb{P}\to^1\mathbb{P})\to(\mathcal{L}_{cf}\to\mathbb{P}\to^1\mathbb{P})$ is defined as follows:

$$\begin{aligned}\Psi_d F\,a\,p &= a\cdot p\\ \Psi_d F\,x\,p &= \Psi_d F\,d(x)p\\ \Psi_d F\,(s_1;s_2)p &= \Psi_d Fs_1(F\,s_2p)\\ \Psi_d F\,(s_1+s_2)p &= (\Psi_d Fs_1p)\cup(\Psi_d Fs_2p).\end{aligned}$$

b. $\mathcal{S}_d = fix(\Psi_d)$; $\mathcal{D}(d,s) = \mathcal{S}_d\, s\, \{\varepsilon\}$.

The above definition is justified in

LEMMA 3.17 *Let* $F, F_1, F_2 \in \mathcal{L}_{cf} \to \mathbb{P} \to^1 \mathbb{P}, p, p_1, p_2 \in \mathbb{P}$.

a. $\Psi_d F s$ *is well-defined for each* F, s.

b. *For all* s, $d(\Psi_d F s p_1, \Psi_d F s p_2) \le \frac{1}{2} d(p_1, p_2)$.

c. *For all* s, $d(\Psi_d F_1 s, \Psi_d F_2 s) \le \frac{1}{2} d(F_1, F_2)$.

d. $\Psi_d \in (\mathcal{L}_{cf} \to \mathbb{P} \to^1 \mathbb{P}) \to^{\frac{1}{2}} (\mathcal{L}_{cf} \to \mathbb{P} \to^1 \mathbb{P})$.

PROOF Simpler than that of Lemma 5.12 and therefore omitted. □

Comparing Definitions 3.14 and 3.16, and using the uniqueness of the fixed point $\mathcal{S}_d$, we easily obtain that, for all s, $\mathcal{S}_d s = \mathcal{S} s\, \eta_d$. Thus, we see that there are (at least) two ways of defining the denotational semantics $\mathcal{D}$ via fixed point techniques. This will allow us in subsequent sections to adopt the most appropriate definition technique.

No more than a small adjustment is necessary to obtain the BT-denotational meaning: introduce $\mathbb{P}^b$ as before (Subsection 3.3), replace in the definitions of (the types of) $\mathcal{S}$ or Ψ_d the domain $\mathbb{P}$ by $\mathbb{P}^b$, and keep all clauses in the definitions, apart from the first ones (in Definition 3.14.a and Definition 3.16.a) where $a \cdot p$ is replaced by $\{ \langle a,p \rangle \}$. This seemingly small variation is sufficient to handle the new range for $\mathcal{D}^b$: Instead of sets of sequences now 'trees' are delivered, and no further measures are required to handle the semantic operators corresponding to the respective syntactic constructs.

3.5 $\mathcal{O}$ and $\mathcal{D}$ are equivalent

The stated equivalence result holds for $\mathcal{O}$ and $\mathcal{D}$ as well as for $\mathcal{O}^b$ and $\mathcal{D}^b$. We shall present the former, leaving the negligible variations to obtain the latter to the reader.

We first introduce the mapping $\mathcal{E}_d : R \to \mathbb{P}$ relating syntactic and semantic continuations:

DEFINITION 3.18

$$\mathcal{E}_d(\mathrm{E}) = \{\varepsilon\},$$
$$\mathcal{E}_d(s;r) = \mathcal{S} s\, \eta_d\, \mathcal{E}_d(r) \quad (= \mathcal{S}_d\, s\, \mathcal{E}_d(r)).$$

LEMMA 3.19 *If* $r_1 \twoheadrightarrow r_2$ *by (seq.comp.) or (recursion) then* $\mathcal{E}_d(r_1) = \mathcal{E}_d(r_2)$. *If* $r_1 \twoheadrightarrow r_2$ *and* $r_1 \twoheadrightarrow r_3$ *(by (choice)) then* $\mathcal{E}_d(r_1) = \mathcal{E}_d(r_2) \cup \mathcal{E}_d(r_3)$.

PROOF Clear from the definitions. □

The key idea as to how to relate $\mathcal{O}$ and $\mathcal{D}$ is contained in the next lemma (a simple example of the technique first introduced in [KR90]):

LEMMA 3.20 $\Phi_d(\mathcal{E}_d) = \mathcal{E}_d$.

PROOF We show that, for all r, $\Phi_d(\mathcal{E}_d)(r) = \mathcal{E}_d(r)$ using induction on $c_r(r)$. A typical case is $r = x;r'$.

$$\Phi_d(\mathcal{E}_d)(x;r')$$
$$= \quad (\text{def. } \Phi_d) \bigcup \{ a \cdot \mathcal{E}_d(\bar{r}) \mid x;r' \to_d^a \bar{r} \}$$
$$= \quad (\text{def. } \mathcal{I}_{cf}) \bigcup \{ a \cdot \mathcal{E}_d(\bar{r}) \mid g;r' \to \bar{r} \}$$

$=$ (ind.) $\mathcal{E}_d(g;r')$

$=$ (Lemma 3.19) $\mathcal{E}_d(x;r')$. □

COROLLARY 3.21 $\mathcal{O}_d = \mathcal{E}_d$.

PROOF Both $\mathcal{O}_d$ and $\mathcal{E}_d$ are fixed points of the contraction Φ_d. □

Finally we have

THEOREM 3.22 $\mathcal{O} = \mathcal{D}$.

PROOF $\mathcal{O}(d,s) = \mathcal{O}_d(s;\mathrm{E}) = \mathcal{E}_d(s;\mathrm{E}) = \mathcal{S}_d\, s\ \{\varepsilon\} = \mathcal{D}(d,s)$. □

4. PROCESS CREATION

Process† creation occurs in parallel languages such as, e.g. the parallel object-oriented language POOL ([A89, AR89a]). A dynamically evolving configuration of processes which may refer to each other through (pointer) variables results from execution of such a program, and the creation of a new process is a central programming concept in this setting. We study here (as everywhere in our paper) a schematic (i.e. variableless) version, abstracting from the pointer structure. What remains is, at each moment, a set of $n \geq 1$ processes executing in parallel. Process creation here amounts to the addition of an $n+1$-st process to this set, together with the initiation of its execution. For some more details on this notion at this abstract level we refer to [AB88]; further details are supplied in [ABKR89, AR89a, AR90, BV91]. (In Sections 4 and 5 we shall only be concerned with LT-semantics; BT returns in Section 6.)

4.1 Syntax

Let $(a,b,c,d \in) A$ and $(x,y \in) PVar$ be as in Section 3. We introduce the language $\mathcal{L}_{pc}$ which extends $\mathcal{L}_{cf}$ with the **new**(s) construct for process creation.

DEFINITION 4.1

a. The class $(s \in) \mathcal{L}_{pc}$ of statements is given by

$$s ::= a \mid x \mid s_1 ; s_2 \mid s_1 + s_2 \mid \mathbf{new}(s)\,.$$

b. The classes $(g \in) \mathcal{L}^g_{pc}$, $(h \in) \mathcal{L}^h_{pc}$ are given by

$$g ::= h \mid g_1 ; g_2 \mid g_1 + g_2 \mid \mathbf{new}(g)\,.$$
$$h ::= a \mid h ; s \mid h_1 + h_2\,.$$

c. The class $(d \in) Decl_{pc}$ has elements $d : PVar \to \mathcal{L}^g_{pc}$.

d. A program is a pair (d,s).

REMARKS

a. The **new**(s) construct serves to create a new process with body s. For example, executing **new**$(a;$**new**$(b);c);d$ will result in parallel execution of $a;$**new**$(b);c$ and of d, to be denoted (for the purposes of this explanation only) by $(a;$**new**$(b);c) \parallel d$. Performing an a-step results in the

† The programming concept of 'process' has nothing to do with the mathematical notion of a process p in a domain $\mathbb{P}$.

remainder program $(\mathbf{new}(b);c) \parallel d$ which may evolve, in turn, to the program $b \parallel c \parallel d$. Note that the parallel operator $\parallel$ is not itself in the syntax of the language (see also the remark at the end of Section 4), but used here only to sketch its intended semantics in familiar terms. Precise definitions will follow.

2. In a procedure declaration such as $d(x) = \mathbf{new}(a);x$, execution of the body $\mathbf{new}(a);x$ may start with execution of x (since $\mathbf{new}(a);x$ has the same effect as $a \parallel x$). In order to avoid such unguarded behaviour, the auxiliary h is employed.

The syntactic continuations $(r \in) R$ are now given in

DEFINITION 4.2 $r ::= \mathrm{E} \mid (s;r) \mid (r_1,r_2)$.

Execution of (r_1,r_2) will be defined in such a way that it amounts to the parallel (here taken in the interleaved sense) execution of r_1 and r_2. It will be convenient to adopt, throughout this section, the following

CONVENTION We shall always identify (E,r) and (r,E) with r.

4.2 Operational semantics

$(u,v \in) A^\infty$, $(p \in) \mathbb{P}$, $a{\cdot}u$, $a{\cdot}p$ are as in Section 3. Transitions are again fourtuples in $R \times A \times Decl_{pc} \times R$, with R and $Decl_{pc}$ as given in Subsection 4.1. The transition system $\mathcal{T}_{pc}$ (and associated relation $\mathcal{R}_{pc}$) is given in

DEFINITION 4.3

- (el. action), (recursion), (seq. comp.) and (choice) are as in Definition 3.4.
- $$\frac{(s;\mathrm{E},\, r) \to_d^a \tilde{r}}{\mathbf{new}(s);r \to_d^a \tilde{r}} \qquad \text{(new)}$$
- $$\frac{r_1 \to_d^a r_2}{\begin{array}{c}(r_1,r) \to_d^a (r_2,r)\\ (r,r_1) \to_d^a (r,r_2)\end{array}} \qquad \text{(par.comp.)}$$

The definition of $\mathcal{O}_d$ and $\mathcal{O}$ proceeds in the same way as in Section 3:

DEFINITION 4.4 Let $F \in R \to \mathbb{P}$, and let Φ_d: $(R \to \mathbb{P}) \to (R \to \mathbb{P})$ be given by

$$\Phi_d(F)(\mathrm{E}) = \{\varepsilon\},$$
$$\Phi_d(F)(r) = \cup\{\, a{\cdot}F(r') : r \to_d^a r' \,\}, \quad \text{if } r \neq \mathrm{E},$$

where $r \to_d^a r' \in \mathcal{R}_{pc}$.

LEMMA 4.5

a. *$\Phi_d(F)(r)$ is nonempty and compact for each F, r.*

b. *Φ_d is contracting in F.*

PROOF As usual. □

DEFINITION 4.6

a. $\mathcal{O}_d = fix(\Phi_d)$.

b. $\mathcal{O}(d,s) = \mathcal{O}_d(s;\mathrm{E})$.

DEFINITION 4.7 $r_1 \twoheadrightarrow r_2$ is as in Definition 3.6 (but now with respect to $\mathcal{T}_{pc}$).

The definition of the complexity l: $R \to \mathbb{N}$ is now more involved. l is given as a *pair* $l = \langle k,c \rangle$, where $k(r)$ gives account to the number of unguarded occurrences in r of a procedure variable, and $c(r)$ gives a certain form of syntactic complexity of its argument r. (Note that the definition here differs from that of Definition 3.7!) We order the l-complexity by putting $\langle k_1,c_1 \rangle < \langle k_2,c_2 \rangle$ if either $k_1 < k_2$ or $k_1 = k_2$ and $c_1 < c_2$.

DEFINITION 4.8

a. k: $R \to \mathbb{N}$ is given by $k(\mathrm{E}) = 0$, $k(r_1,r_2) = k(r_1) + k(r_2)$, $k(a;r) = 0$, $k(x;r) = 1 + k(r)$, $k((s_1;s_2);r) = k(s_1;(s_2;r))$, $k((s_1+s_2);r) = max\{ k(s_1;r), k(s_2;r) \}$, $k(\mathbf{new}(s);r) = k(s;\mathrm{E}) + k(r)$.

b. c: $R \to \mathbb{N}$ is given by $c(\mathrm{E})=0$, $c(r_1,r_2) = c(r_1) + c(r_2)$, $c(a;r) = 1 + c(r)$, $c(x;r) = 1 + c(r)$, $c((s_1;s_2);r) = 1 + c(s_1;(s_2;r))$, $c((s_1+s_2);r) = 1 + c(s_1;r) + c(s_2;r)$, $c(\mathbf{new}(s);r) = 1 + c(s;\mathrm{E}) + c(r)$.

LEMMA 4.9

a. $k(h;r) = 0$, $k(g;r) \le k(r)$.

b. *If* $r_1 \twoheadrightarrow r_2$ *then* $l(r_1) > l(r_2)$.

PROOF Part a is shown by induction on the syntactic complexity of first h, then g. Part b is direct from the definitions. □

4.3 Denotational semantics

Before proceeding with the definitions of the various meaning functions, we first define the operator $\|: \mathbb{P} \times \mathbb{P} \to \mathbb{P}$, which shuffles the elementary actions in its (possibly infinite) arguments p_1, p_2 yielding the result $p_1 \| p_2$. Note that $\|$ only occurs in the *semantics* of $\mathcal{L}_{pc}$. We shall define $\|$ as fixed point of a higher-order mapping. This technique, which may seem somewhat overdone in the present setting, is applied firstly to handle finite *and infinite* arguments in one go, and secondly to prepare the way for the definitions in Section 6, where a higher-order definition for (a more involved version of) $\|$ seems essential.

DEFINITION 4.10

a. Let $\phi \in \mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}$. We define the mappings

$$\Omega_{\circ} : (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}) \to (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P})$$

$$\omega_{\circ} : (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}) \to (A^\infty \times A^\infty \to^1 \mathbb{P})$$

$$\Omega_{\|} : (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}) \to (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P})$$

as follows:

$$\Omega_{\circ}(\phi)(p_1,p_2) = \cup\{\,\omega_{\circ}(\phi)(u,v) \mid u\in p_1, v\in p_2\,\}$$
$$\omega_{\circ}(\phi)(\varepsilon,v) = \{\,v\,\}$$
$$\omega_{\circ}(\phi)(au,v) = a\cdot\phi(\{u\},\{v\})$$
$$\Omega_{\parallel}(\phi)(p_1,p_2) = \Omega_{\circ}(\phi)(p_1,p_2)\cup\Omega_{\circ}(\phi)(p_2,p_1)$$

b. $\parallel = fix(\Omega_{\parallel})$, $⫡ = \Omega_{\circ}(\parallel)$.

EXAMPLE Let a^{ω} be the infinite sequence of a's. Then $a^{\omega}\parallel b = \{\,a^{\omega}\,\}\cup(a^{*}ba^{\omega})$. Note that the 'unfair' outcome a^{ω} (b never got its turn) is included in the result.

LEMMA 4.11

a. All operators in Definition 4.10 are well-defined. $\Omega_{\circ}$, $\omega_{\circ}$, $\Omega_{\parallel}$ are contracting in ϕ.

b. $p_1\parallel p_2 = (p_1 ⫡ p_2)\cup(p_2 ⫡ p_1)$.

PROOF Part a follows by Michael's theorem; part b is direct from the definitions. □

The denotational mappings are collected in the next definition. We draw attention to the clause dealing with **new**(s). Also, the meaning of a procedure variable is handled in the customary way through environments.

DEFINITION 4.12

a. $\mathcal{S}\colon \mathcal{L}_{pc}\to Env\to \mathbb{P}\to\mathbb{P}$ is given by

$$\mathcal{S}a\eta p = a\cdot p$$
$$\mathcal{S}x\eta p = \eta x p$$
$$\mathcal{S}(s_1;s_2)\eta p = \mathcal{S}s_1\eta(\mathcal{S}s_2\eta p)$$
$$\mathcal{S}(s_1+s_2)\eta p = (\mathcal{S}s_1\eta p)\cup(\mathcal{S}s_2\eta p)$$
$$\mathcal{S}\,\mathbf{new}(s)\eta p = (\mathcal{S}s\eta\{\varepsilon\})\parallel p$$

b. $\mathrm{H}_d\colon Env\to Env$ is given by

$$\mathrm{H}_d\eta x = \mathcal{S}(d(x))\eta$$

c. $\eta_d = fix(\mathrm{H}_d)$, $\mathcal{D}(d,s) = \mathcal{S}s\eta_d\{\varepsilon\}$.

The justification of this definition follows in

LEMMA 4.13

a. $\mathcal{S}s\eta\in\mathbb{P}\to^{1}\mathbb{P}$.

b. $\mathcal{S}s\in Env\to^{1}\mathbb{P}\to^{1}\mathbb{P}$.

c. $\mathcal{S}h\eta\in\mathbb{P}\to^{1/2}\mathbb{P}$.

d. $\mathcal{S}h\in Env\to^{1/2}\mathbb{P}\to^{1/2}\mathbb{P}$.

e. $\mathcal{S}g\eta\in\mathbb{P}\to^{1}\mathbb{P}$.

f. $\mathcal{S}g\in Env\to^{1/2}\mathbb{P}\to^{1}\mathbb{P}$.

g. $\mathrm{H}_d\in Env\to^{1/2} Env$.

PROOF We exhibit a few selected subcases. Throughout, we argue by induction on the syntactic complexity of the statements concerned.

c. Case $h\equiv h';s$.

$$d(\mathcal{S}(h';s)\eta p_1, \mathcal{S}(h';s)\eta p_2)$$

$= \quad d(\mathcal{S}h'\eta(\mathcal{S}s\eta p_1), \mathcal{S}h'\eta(\mathcal{S}s\eta p_2))$

$\leq \quad$ (ind.) $\frac{1}{2}d(\mathcal{S}s\eta p_1, \mathcal{S}s\eta p_2)$

$\leq \quad$ (part a) $\frac{1}{2}d(p_1, p_2)$.

d. Take p arbitrary, case $h \equiv h';s$.

$d(\mathcal{S}(h';s)\eta_1 p, \mathcal{S}(h';s)\eta_2 p)$

$\leq \quad$ (def. $\mathcal{S}$, d an ultrametric)

$max\{ d(\mathcal{S}h'\eta_1(\mathcal{S}s\eta_1 p), \mathcal{S}h'\eta_1(\mathcal{S}s\eta_2 p))$ (*),

$d(\mathcal{S}h'\eta_1(\mathcal{S}s\eta_2 p), \mathcal{S}h'\eta_2(\mathcal{S}s\eta_2 p)$ (**) $\}$

$\leq \quad \frac{1}{2}d(\eta_1, \eta_2)$, since

(*)

$\leq \quad$ (part c) $\frac{1}{2}d(\mathcal{S}s\eta_1 p, \mathcal{S}s\eta_2 p)$

$\leq \quad$ (part b) $\frac{1}{2}d(\eta_1, \eta_2)$, and

(**)

$\leq \quad$ (ind.) $\frac{1}{2}d(\eta_1, \eta_2)$.

e. Case $g \equiv \mathbf{new}(g')$.

$d(\mathcal{S}\mathbf{new}(g')\eta p_1, \mathcal{S}\mathbf{new}(g')\eta p_2)$

$= \quad d((\mathcal{S}g'\eta\{\varepsilon\}) \| p_1, (\mathcal{S}g'\eta\{\varepsilon\}) \| p_2)$

$\leq \quad$ ($\|$ ndi) $d(p_1, p_2)$. □

REMARK It is also possible to define $\mathcal{D}$ as fixed point of some higher-order Ψ. Note, however, that some care has to be applied in Ψ's definition: simply following Definition 3.16 would not work. Instead, a definition which is organized following the s, g, and h syntax is required.

4.4 $\mathcal{O}$ and $\mathcal{D}$ are equivalent

Let $\mathcal{E}_d: R \to \mathbb{P}$ be given in

DEFINITION 4.14

$$\mathcal{E}_d(E) = \{\varepsilon\},$$
$$\mathcal{E}_d(s;r) = \mathcal{S}s\eta_d\mathcal{E}_d(r),$$
$$\mathcal{E}_d(r_1,r_2) = \mathcal{E}_d(r_1) \| \mathcal{E}_d(r_2).$$

LEMMA 4.15 *If $r_1 \twoheadrightarrow r_2$ by (recursion), (seq.comp.) or (new) then $\mathcal{E}_d(r_1) = \mathcal{E}_d(r_2)$. If $r_1 \twoheadrightarrow r_2$ and $r_1 \twoheadrightarrow r_3$ (by (choice)) then $\mathcal{E}_d(r_1) = \mathcal{E}_d(r_2) \cup \mathcal{E}_d(r_3)$.*

PROOF Clear by the definitions. Observe that the (par.comp.) rule does not contribute to the $\twoheadrightarrow$ relation. □

LEMMA 4.16 $\Phi_d(\mathcal{E}_d) = \mathcal{E}_d$.

PROOF We prove that, for all r, $\Phi_d(\mathcal{E}_d)(r) = \mathcal{E}_d(r)$ by induction on $l(r)$. We exhibit two subcases:

Case $r \equiv x;r'$.

$\Phi_d(\mathcal{E}_d)(x;r')$

$= \quad \cup\{ a.\mathcal{E}_d(\bar{r}) \mid x;r' \to_d^a \bar{r} \}$

$= \quad$ (definition $\mathcal{I}_{pc}$) $\cup\{ a.\mathcal{E}_d(\bar{r}) \mid g;r' \to_d^a \bar{r} \}$

$= \quad \Phi_d(\mathcal{E}_d)(g;r')$

$=$ (ind.) $\mathcal{E}_d(g;r')$

$=$ (Lemma 4.15) $\mathcal{E}_d(x;r')$.

Case $r \equiv (r_1,r_2)$.

$\Phi_d(\mathcal{E}_d)(r_1,r_2)$

$= \bigcup\{ a.\mathcal{E}_d(\bar{r}) \mid (r_1,r_2) \rightarrow_d^a \bar{r} \}$

$=$ (def. $\mathcal{T}_{pc}$) $\bigcup\{ a \cdot \mathcal{E}_d(r',r_2) \mid r_1 \rightarrow_d^a r' \} \cup$ (symm.)

$=$ (def. $\mathcal{E}_d$) $\bigcup\{ a \cdot \mathcal{E}_d(r') \parallel \mathcal{E}_d(r_2) : r_1 \rightarrow_d^a r' \cup$ (symm.)

$=$ (def. $\mathbin{⫫}$) $\bigcup(\{ a.\mathcal{E}_d(r') : r_1 \rightarrow_d^a r' \} \mathbin{⫫} \mathcal{E}_d(r_2)) \cup$ (symm.)

$=$ (prop. $\mathbin{⫫}$) $(\bigcup\{ a.\mathcal{E}_d(r') : r_1 \rightarrow_d^a r' \}) \mathbin{⫫} \mathcal{E}_d(r_2) \cup$ (symm.)

$=$ (def. Φ_d) $(\Phi_d(\mathcal{E}_d)(r_1) \mathbin{⫫} \mathcal{E}_d(r_2)) \cup$ (symm.)

$=$ (ind.) $(\mathcal{E}_d(r_1) \mathbin{⫫} \mathcal{E}_d(r_2)) \cup$ (symm.)

$=$ (def. $\parallel$) $\mathcal{E}_d(r_1) \parallel \mathcal{E}_d(r_2)$

$=$ (def. $\mathcal{E}_d$) $\mathcal{E}_d(r_1,r_2)$. □

It is now immediate that

THEOREM 4.17 $\mathcal{O} = \mathcal{D}$.

PROOF By Lemma 4.16 and Banach's fixed point theorem 2.4, $\mathcal{O}_d = \mathcal{E}_d$. $\mathcal{O} = \mathcal{D}$ now follows as in the proof of Theorem 3.22. □

We conclude this Section 4 with a

REMARK The programming concept of process creation has been modeled in terms of the semantic $\parallel$-operator (from Definition 4.10). As a natural consequence of this, one may want to compare $\mathcal{L}_{cf}$ with the language $\mathcal{L}_{sh}$ which extends $\mathcal{L}_{cf}$ with the *syntactic* merge operator (i.e. which has syntaxis s ($\in \mathcal{L}_{sh}$) $::= a \mid x \mid s_1 ; s_2 \mid s_1 + s_2 \mid s_1 \parallel s_2$, and derived definitions). Now a somewhat surprising result of Aalbersberg and America (personal communication) is that $\mathcal{L}_{pc}$ and $\mathcal{L}_{sh}$ are incomparable: Assuming the natural semantics for $\mathcal{L}_{sh}$ (relating the syntactic $\parallel$ to the semantic $\parallel$), we have that there exists a program in $\mathcal{L}_{pc}$ without an equivalent program in $\mathcal{L}_{sh}$, and vice versa. We have thus falsified a conjecture stating that the **new**-construct may as well be expressed in terms of the merge operator. The nontrivial counter examples as mentioned involve combined use of recursion and process creation or merge. One final point: continuation semantics does not fit well with merge. We do not know how to provide a clause for $\mathcal{S}(s_1 \parallel s_2)\eta p$ in terms of $\mathcal{S}s_1\eta p_1$ and $\mathcal{S}s_2\eta p_2$ for some suitable continuations p_1, p_2.

5. BACKTRACKING

Our next language, $\mathcal{L}_{bt}$, has as characteristic feature *failure* (in the form of the atomic **fail** statement) and *backtracking* (expressed by $s_1 \square s_2$). The nondeterministic choice $s_1 + s_2$ has disappeared; recursion and sequential composition remain. In order to execute $s_1 \square s_2$, we assume two kinds of syntactic continuations, viz. the *success continuation* r and the *failure continuation* t. Execution of $((s_1 \square s_2);r):t$ is performed by executing s_1 with success continuation r and failure continuation $(s_2;r):t$. If somewhere in the execution of s_1 we encounter failure, we continue with the execution of $(s_2;r):t$. If not, we continue with execution of $r:((s_2;r):t)$.

In the papers [B88, BrVi89, Vi90] we have shown how to apply this construct to model the backtracking feature of *PROLOG*. The present model being logicless, in the papers just cited (cf. also [BK90]) we also discuss how to interpret the atomic actions and how to instantiate the procedure variables in such a way that the usual *PROLOG* semantics in terms of computed answer substitutions is obtained. Moreover, in [B88, BrVi89] it was also shown how the simple backtracking formalism to be presented below may be extended with a continuation semantics for the cut operator.

5.1 Syntax

Let $(a \in) A$, $(x \in) PVar$ be as usual. We shall from now on be somewhat more succinct in the various definitions and lemmas.

DEFINITION 5.1

a. $s\ (\in \mathcal{L}_{bt}) ::= a \mid x \mid \mathbf{fail} \mid s_1;s_2 \mid s_1 \square s_2$.

b. $g\ (\in \mathcal{L}_{bt}^{g}) ::= a \mid \mathbf{fail} \mid g;s \mid g_1 \square g_2$.

c. $d\ (\in Decl_{bt})$ is a mapping d: $PVar \to \mathcal{L}_{bt}^{g}$; a program is a pair (d,s).

DEFINITION 5.2

a. $r\ (\in R) ::= \mathrm{E} \mid (s;r)$

b. $t\ (\in T) ::= \mathbf{f} \mid (r:t)$

f is short for **fail**. Parentheses will be omitted when convenient. We do *not* identify t and $\mathrm{E}:t$!

5.2 Operational semantics

Since the behaviour of an $\mathcal{L}_{bt}$-program is deterministic, single sequences rather than sets of those are now delivered. For consistency in notation, we use in this section p to range over $\mathbb{P} =_{df} A_{\delta}^{\infty} =_{df} A^* \cup A^*\cdot\delta \cup A^{\omega}$. Here $A^*\cdot\delta$ denotes the set of all finite sequences over A, with δ postfixed. We define the operator $\circ$ of concatenation on $\mathbb{P}$ as follows:

DEFINITION 5.3 Let ϕ range over $\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}$.

a. $\Omega_\circ : (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}) \to (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P})$ is given by

$$\begin{aligned} \Omega_\circ \phi \varepsilon p &= p \\ \Omega_\circ \phi \delta p &= \delta \\ \Omega_\circ \phi a p' p &= a \cdot \phi p' p \end{aligned}$$

b. $\circ = fix(\Omega_\circ)$.

Thus, '$\circ$' is the usual concatenation with, in addition, the property that $\delta \circ p = \delta$.

Transitions are four-tuples in $T \times A \times Decl_{bt} \times T$, written as $t \to_d^a t'$. The transition system $\mathcal{T}_{bt}$ (and associated transition relation $\mathcal{R}_{bt}$) is given in

DEFINITION 5.4

- $(a;r):t \to_d^a r:t$ (el.action)
- $\dfrac{(g;r):t \to_d^a \tilde{t}}{(x;r):t \to_d^a \tilde{t}}$, $d(x) = g$ (recursion)

- $$\frac{t \rightarrow_d^a \tilde{t}}{(\mathbf{fail};r):t \rightarrow_d^a \tilde{t}}$$ (failure)
- $$\frac{(s_1;(s_2;r)):t \rightarrow_d^a \tilde{t}}{((s_1;s_2);r):t \rightarrow_d^a \tilde{t}}$$ (seq.comp.)
- $$\frac{(s_1;r):((s_2;r):t) \rightarrow_d^a \tilde{t}}{((s_1 \square s_2);r):t \rightarrow_d^a \tilde{t}}$$ (backtrack)

With $\mathcal{T}_{bt}$ we associate the usual $\twoheadrightarrow$ relation:

DEFINITION 5.5

a. $t_1 \twoheadrightarrow t_2$ if there is a rule in $\mathcal{T}_{bt}$ of the form

$$\frac{t_2 \rightarrow_d^a \tilde{t}}{t_1 \rightarrow_d^a \tilde{t}}.$$

$\twoheadrightarrow^*$ is the reflexive and transitive closure of $\twoheadrightarrow$.

b. We say that t *terminates* if, for some t', $t \twoheadrightarrow^* \mathrm{E}{:}t'$. Also, t *fails* if $t \twoheadrightarrow^* \mathbf{f}$.

For the syntactic constructs from $\mathcal{L}_{bt}$, R, T we define a complexity measure which is a slight extension of that introduced in Section 3:

DEFINITION 5.6 The mappings $c_t : T \rightarrow \mathbb{N}$, $c_r : R \rightarrow \mathbb{N}$ are given by

a. $c_t(\mathbf{f}) = 0$, $c_t(r{:}t) = c_r(r) + c_t(t)$.

b. $c_r(\mathrm{E}) = 0$, $c_r(s;r) = c_s(s)$.

c. $c_s(a) = c_s(\mathbf{fail}) = 1$, $c_s(x) = c_s(d(x)) + 1$, $c_s(s_1\,;s_2) = c_s(s_1) + 1$, $c_s(s_1 \square s_2) = c_s(s_1) + c_s(s_2) + 1$.

LEMMA 5.7

a. *c_t, c_r, c_s are well-defined.*

b. *If $t_1 \twoheadrightarrow t_2$ then $c_t(t_1) > c_t(t_2)$.*

c. *For each t, either t terminates, or t fails, or $t \rightarrow^a t'$ for some a, t'.*

PROOF Well-definedness of c_t, c_r is clear. Well-definedness of c_s follows by induction on the syntactic complexity of first g, then arbitrary s. Part b is clear from the definitions, part c uses induction on $c_t(t)$. □

We are now ready for

DEFINITION 5.8 Let F range over $T \rightarrow \mathbb{P}$. The mapping $\Phi_d : (T \rightarrow \mathbb{P}) \rightarrow (T \rightarrow \mathbb{P})$ is given by

$$\begin{array}{lll} \Phi_d(F)(t) & = \varepsilon, & \text{if } t \text{ terminates}, \\ \Phi_d(F)(t) & = \delta, & \text{if } t \text{ fails}, \\ \Phi_d(F)(t) & = a \cdot F(t'), & \text{if } t \rightarrow_d^a t'. \end{array}$$

We have the usual

LEMMA 5.9 *$\Phi_d(F)(t)$ is well-defined for each F, t. Also, Φ_d is contracting in F.*

PROOF Easy. □

The operational semantics for $\mathcal{L}_{bt}$ is given in

DEFINITION 5.10

a. $\mathcal{O}_d = fix(\Phi_d)$.

b. $\mathcal{O}(d,s) = \mathcal{O}_d((s;\mathrm{E}):\mathbf{f})$.

5.3 Denotational semantics

To prepare the way for a related definition in Section 6, we now vary the denotational definition format by replacing the use of (fixed points of) environments by the use of ($\mathcal{S}_d$ as fixed point of) a higher-order mapping Ψ_d. (Recall that both approaches were already used for the simple language $\mathcal{L}_{cf}$ of Section 3.) We use as semantic success continuations functions ϕ in $\mathbb{P} \to^1 \mathbb{P}$, and as semantic failure continuations elements p in $\mathbb{P}$. Moreover, we shall use F to range over $\mathcal{L}_{bt} \to (\mathbb{P} \to^1 \mathbb{P}) \to (\mathbb{P} \to^1 \mathbb{P})$.

DEFINITION 5.11

a. The function Ψ_d: $(\mathcal{L}_{bt} \to (\mathbb{P} \to^1 \mathbb{P}) \to (\mathbb{P} \to^1 \mathbb{P})) \to (\mathcal{L}_{bt} \to (\mathbb{P} \to^1 \mathbb{P}) \to (\mathbb{P} \to^1 \mathbb{P}))$ is given by

$$\begin{aligned}
\Psi_d F\, a\, \phi p &= a \cdot \phi p \\
\Psi_d F\, x\, \phi p &= \Psi_d F\, d(x)\, \phi p \\
\Psi_d F\, \mathbf{fail}\, \phi p &= p \\
\Psi_d F\, (s_1 ; s_2)\, \phi p &= \Psi_d F\, s_1\, (F\, s_2\, \phi) p \\
\Psi_d F\, (s_1 \square s_2)\, \phi p &= \Psi_d F\, s_1\, \phi (\Psi_d F\, s_2\, \phi p)\,.
\end{aligned}$$

b. $\mathcal{S}_d = fix(\Psi_d)$; $\mathcal{D}(d,s) = \mathcal{S}_d\, s\, (\lambda p.\varepsilon)\, \delta$.

This definition is justified by

LEMMA 5.12 *Let $\phi, \phi_1, \phi_2 \in \mathbb{P} \to^1 \mathbb{P}$, $p, p_1, p_2 \in \mathbb{P}$.*

a. $\forall s$: $d(\Psi_d F\, s\, \phi p_1, \Psi_d F\, s\, \phi p_2) \le d(p_1, p_2)$.

b. $\forall s$: $d(\Psi_d F\, s\, \phi_1, d(\Psi_d F\, s\, \phi_2) \le \frac{1}{2} d(\phi_1, \phi_2)$.

c. $\forall s$: $d(\Psi_d F_1\, s, \Psi_d F_2\, s) \le \frac{1}{2} d(F_1, F_2)$.

d. $\Psi_d \in (\mathcal{L}_{bt} \to (\mathbb{P} \to^1 \mathbb{P}) \to (\mathbb{P} \to^1 \mathbb{P})) \to^{1/2} (\mathcal{L}_{bt} \to (\mathbb{P} \to^1 \mathbb{P}) \to (\mathbb{P} \to^1 \mathbb{P}))$.

PROOF We present a few typical subcases.

a.

$$\begin{aligned}
& d(\Psi_d F\, \mathbf{fail}\, \phi p_1, \Psi_d F\, \mathbf{fail}\, \phi p_2) \\
= \;& d(p_1, p_2); \\
& d(\Psi_d F\, (s_1;s_2)\, \phi p_1, \Psi_d F\, (s_1;s_2)\, \phi p_2) \\
= \;& d(\Psi_d F\, s_1\, (F\, s_2\, \phi) p_1, \Psi_d F\, s_1\, (F\, s_2\, \phi) p_2) \\
\le \;& \text{(the ind. hyp. applies since } F\, s_2\, \phi \in \mathbb{P} \to^1 \mathbb{P})\; d(p_1, p_2).
\end{aligned}$$

c. Choose some ϕ, p. We consider the case $s_1;s_2$:

$$
\begin{aligned}
& d(\Psi_d F_1 (s_1;s_2)\phi p\,,\ \Psi_d F_2 (s_1;s_2)\phi p\,) \\
\le\ & \text{(def., } d \text{ an ultrametric)} \\
& max\{\ d(\Psi_d F_1 s_1 (F_1 s_2 \phi)p\,, \Psi_d F_1 s_1 (F_2 s_2 \phi)p\,)\ (*), \\
& \qquad d(\Psi_d F_1 s_1 (F_2 s_2 \phi)p\,, \Psi_d F_2 s_1 (F_2 s_2 \phi)p\,)\ (**)\ \}, \\
& (*) \\
& \le \text{(part b) } \tfrac{1}{2} d(F_1 s_2 \phi, F_2 s_2 \phi) \\
& \le \tfrac{1}{2} d(F_1, F_2), \\
& (**) \\
& \le \text{(ind.) } \tfrac{1}{2} d(F_1, F_2).\ \square
\end{aligned}
$$

5.4 $\mathcal{O}$ and $\mathcal{D}$ are equivalent

We define functions $\mathcal{E}_d$ and $\mathcal{F}_d$ relating syntactic and semantic success and failure continuations, respectively.

DEFINITION 5.13

a. The function $\mathcal{E}_d\colon R \to (\mathbb{P} \to^1 \mathbb{P})$ is given by

$$\mathcal{E}_d(\mathrm{E}) = \lambda p.\varepsilon,$$

$$\mathcal{E}_d(s;r) = \mathcal{S}_d(s)\mathcal{E}_d(r).$$

b. The function $\mathcal{F}_d\colon T \to \mathbb{P}$ is given by

$$\mathcal{F}_d(\mathrm{f}) = \delta,$$

$$\mathcal{F}_d(r{:}t) = \mathcal{E}_d(r)\mathcal{F}_d(t).$$

LEMMA 5.14

a. *If* $t_1 \twoheadrightarrow t_2$ *then* $\mathcal{F}_d(t_1) = \mathcal{F}_d(t_2)$.

b. $\mathcal{F}_d((a;r):t) = a\cdot\mathcal{F}_d(r:t)$; $\mathcal{F}_d(\mathrm{E}:t) = \varepsilon$.

PROOF We exhibit one typical case for part a:

$$
\begin{aligned}
& \mathcal{F}_d((s_1 \square s_2);r{:}t) \\
=\ & \mathcal{E}_d((s_1 \square s_2);r)\mathcal{F}_d(t) \\
=\ & \mathcal{S}(s_1 \square s_2)\mathcal{E}_d(r)\mathcal{F}_d(t) \\
=\ & \mathcal{S}_d s_1 \mathcal{E}_d(r)(\mathcal{S} s_2 \mathcal{E}_d(r)\mathcal{F}_d(t)) \\
=\ & \dots \\
=\ & \mathcal{F}_d((s_1;r):((s_2;r):t)).\ \square
\end{aligned}
$$

Next we have the usual

LEMMA 5.15 $\Phi_d(\mathcal{F}_d) = \mathcal{F}_d$.

PROOF We show, employing induction on $c_t(t)$, that $\Phi_d(\mathcal{F}_d)(t) = \mathcal{F}_d(t)$, for all t. $\square$

THEOREM 5.16

a. $\mathcal{O}_d = \mathcal{F}_d$.

b. $\mathcal{O} = \mathcal{D}$.

PROOF As usual. □

6. RENDEZ-VOUS

In this section we investigate (a schematic kind of) the rendez-vous programming construct as occurring in ADA [ANS830] or POOL. The version studied here extends the communication mechanism of CCS [Mi80] in the following way: Whereas in CCS synchronised execution of the actions c, $\bar{c}$ in two parallel components results in the execution of a τ-step (as expressed by the equation $c \mid \bar{c} = \tau$), in our language $\mathcal{L}_{rv}$ we extend the class of elementary actions with *methods* m, $\bar{m}$ (which thus occur in pairs as well), together with an extension of the declaration map d which now also maps each m (and $\bar{m}$) to an associated statement $d(m)$ $(= d(\bar{m}))$ as body. The intended execution of this construct is as follows: Imagine two parallel components r_1, r_2, the first ready to execute $m;r'$, and the second ready to execute $\bar{m};r''$. A successful communication will then result in the execution of $d(m);(r',r'')$. Thus, the procedure body $d(m)$ associated with m is executed first; after its completion, the parallel execution of r' and r'' is resumed.

Following the plan to discuss key features of the language POOL, we embed the rendez-vous notion in a language with process creation. Since the denotational meaning of an element in $\mathcal{L}_{rv}$ now involves (in the **new**(s) case) the semantic operator $\|$ which in turn - by the argument as just given - involves the communication operator $\mid$ calling for the denotational meaning of $d(m)$, it may become apparent to the reader that we are confronted with a more complex situation than that encountered earlier: We shall have to design a *simultaneous* higher-order definition for the denotational meaning function and for the semantics $\|$-operator.

One further point to mention in this introduction is that we shall employ a branching time semantic domain (elsewhere often called a *bisimulation*-model): The need for a BT-domain arises - just as for CCS - from the possible deadlock behaviour of an $\mathcal{L}_{rv}$-program: We want to distinguish between the meaning of $a_1;(a_2 + m)$ and $(a_1;a_2) + (a_1;m)$ since, in the presence of a parallel $\bar{m}$, their deadlock behaviour differs.

A final word on the relationship with [R90a]: We have designed here a BT-operational model which is self-contained (expressed only in terms of the familiar transition system formalism). In [R90a], the (intermediate) operational BT-semantics involves as well an application of the *denotational* meaning function. Compared with [ABKR89], the approach adopted here is more demanding since continuations are passed as arguments of functions (necessitating the solution of a domain equation of the form

$$\mathbb{P} = \cdots (\mathbb{P} \to \cdots) \cdots). \tag{6.1}$$

In the present setting dealing with a skeleton-version of the rendez-vous construct we have managed to avoid these complexities. In [BV91] we show how the method to be described below works as well in a setting for the rendez-vous with individual variables, parameters, and a resulting value to be returned.

6.1 Syntax

Let $(a \in) A$ and $(x \in) PVar$ be as usual, and let $(m \in) M$ be a set of *method* names. Let $\bar{\cdot} : M \to M$ be a mapping such that $\bar{\bar{m}} = m$. Let e range over $A \cup M$.

DEFINITION 6.1

a. $s\ (\in \mathcal{L}_{rv}) ::= e \mid x \mid s_1 ; s_2 \mid s_1 + s_2 \mid \mathbf{new}(s)$,
$g\ (\in \mathcal{L}^g_{rv}) ::= h \mid g_1 ; g_2 \mid g_1 + g_2 \mid \mathbf{new}(g)$,
$h\ (\in \mathcal{L}^h_{rv}) ::= a \mid h;s \mid h_1 + h_2$.

b. $(d \in) Decl_{rv}$ consists of mappings $d = (d_1, d_2)$, where
$$d_1 : PVar \to \mathcal{L}^g_{rv},\ d_2 : M \to \mathcal{L}^h_{rv}$$
such that $d_2(m) = d_2(\bar{m})$. For simplicity, we drop indices on d when no confusion is expected.

c. Programs are as usual.

REMARKS

1. Note that the syntax for h involves a, not e. For guarding purposes, method names have the same role as procedure variables.
2. The codomain for d_2 is $\mathcal{L}^h_{rv}$ rather than $\mathcal{L}^g_{rv}$ (or $\mathcal{L}_{rv}$). This is motivated by our wish to have contracting functions in the semantic definitions (cf. Subsection 6.3).

Syntactic continuations are as in Section 4:

DEFINITION 6.2 $r\ (\in R) ::= \mathrm{E} \mid (s;r) \mid (r_1, r_2)$.

Again, we identify (E, r) and (r, E) with r.

6.2 Operational semantics

Transitions are four-tuples in $R \times (A \cup M) \times Decl_{rv} \times R$, written as $r_1 \to^e_d r_2$.

DEFINITION 6.3 $\mathcal{T}_{rv}$ and associated $\mathcal{R}_{rv}$ are given by

a. All axioms and rules as in $\mathcal{T}_{pc}$ of Section 4, with e replacing a.

b. In addition, the rule

- $$\frac{r_1 \to^m r',\ r_2 \to^{\bar{m}} r'',\ h;(r',r'') \to^e r}{(r_1, r_2) \to^e r},\ d(m) = h \qquad \text{(rendez-vous)}$$

The relation $\twoheadrightarrow$ is as in Section 4 (the rendez-vous case will obtain special treatment below). The complexity definition is slightly amended:

DEFINITION 6.4 The complexity $l = \langle k, c \rangle$ for $r \in R$ is as in Section 4, with the addition that $k(m;r) = 1 + k(r)$, and $c(m;r) = 1 + c(r)$.

Again we have

LEMMA 6.5

a. $k(h;r) = 0,\ k(g;r) \leq k(r)$.

b. *If* $r_1 \twoheadrightarrow r_2$ *then* $l(r_1) > l(r_2)$. □

As semantic domain we use here the complete metric space $\mathbb{P}$ which satisfies the domain equation

$$\mathbb{P} = \{p_0\} \,\bar{\cup}\, \mathcal{P}_{co}((A \,\bar{\cup}\, M) \times id_{1/2}(\mathbb{P})) \tag{6.2}$$

Here p_0 is the nil-process - modeling the nil action. Also, from Section 2 we recall that (6.2) is actually an equation in (complete) metric spaces.

Let F range over $R \to \mathbb{P}$. We give the usual

DEFINITION 6.6 $\Phi_d : (R \to \mathbb{P}) \to (R \to \mathbb{P})$ is given by

$\Phi_d(F)(\mathrm{E}) \;=\; p_0$,

$\Phi_d(F)(r) \;=\; \{\langle e, F(r')\rangle \mid r \to_d^e r'\}$, if $r \neq \mathrm{E}$.

REMARK See also Definition 3.13 and the comments following it.

DEFINITION 6.7 $\mathcal{O}_d = fix(\Phi_d)$, $\mathcal{O}(d,s) = \mathcal{O}_d(s;\mathrm{E})$.

The $\mathcal{O}$ as just given yields branching time (BT) results; moreover, it preserves m-steps which have not synchronised with a corresponding $\bar{m}$. For example, $\mathcal{O}(d, (a+m);\mathrm{E}) = \{\langle a,p_0\rangle, \langle m,p_0\rangle\}$. The main advantage of this $\mathcal{O}$ is that it equals the denotational $\mathcal{D}$. On the other hand, it is possible to define a *linear time* $\mathcal{O}'$ which, in addition, suppresses m-steps in the result. The details are as follows: Let $\mathbb{P}' =_{df} \mathcal{P}_{nc}(A_\delta^\infty)$. The mapping $\mathcal{O}_d'$: $R \to \mathbb{P}'$ satisfies

$\mathcal{O}_d'(\mathrm{E}) \;=\; \{\varepsilon\}$,

$\mathcal{O}_d'(r) \;=\; \cup\{e\cdot\mathcal{O}_d(r') \mid r \to_d^e r', e \in A\}$ if $r \neq \mathrm{E}$ and $\{e \mid r \to_d^e r', e \in A\} \neq \varnothing$.

$\mathcal{O}_d'(r) \;=\; \{\delta\}$, otherwise

Here $r \to_d^e r'$ is from $\mathcal{R}_{rv}$ (but note that e may not be from M). Well-definedness of $\mathcal{O}_d'$ may be shown in the usual manner. Examples are

$\mathcal{O}_d'(a_1;(a_2+m);\mathrm{E}) = \{a_1a_2\}$.

$\mathcal{O}_d'(((a_1;a_2)+(a_1;m));\mathrm{E}) = \{a_1a_2, a_1\delta\}$.

Furthermore, we may show that $\mathcal{O}_d' = abs \circ \mathcal{O}_d$, where the abstraction mapping abs: $\mathbb{P} \to \mathbb{P}'$ firstly replaces a tree by the set of all its paths (thus collapsing the branching structure), and secondly omits all $m \cdot \cdots$ paths. We define abs to satisfy

$abs(p_0) \;=\; \{\varepsilon\}$,

$abs(p) \;=\; \cup\{e\cdot abs(p') \mid \langle e,p'\rangle \in p, e \in A\}$ if $\{e \mid \langle e,p'\rangle \in p, e \in A\} \neq \varnothing$,

$abs(p) \;=\; \{\delta\}$, otherwise.

This definition may be justified by the familiar higher-order argument.

REMARK Though the rendez-vous rule (and the subsequent denotational definitions) is *sound* (it yields only 'correct' computations), one might argue that it is not *complete*: In certain situations it yields deadlock where it should derive some successful computation. Consider, e.g., the situation that $d(m) = a;m'$, and that $r_1 = (m;\mathrm{E},\bar{m}';\mathrm{E})$, $r_2 = \bar{m};\mathrm{E}$. Since $r_1 \to^m \bar{m}';\mathrm{E} = r'$, $r_2 \to^{\bar{m}} \mathrm{E} = r''$, and

$d(m);(r',r'') \to^a m';(r',r'')$, we may infer that $(r_1,r_2) \to^a m';\bar{m}';\mathrm{E}$. Now in this result (i.e., $m';\bar{m}';\mathrm{E}$), a rendez-vous between m' and $\bar{m}'$ is not possible, due to the occurrence of $\bar{m}'$ *after* m', and a deadlock has been yielded where a successful computation should have been delivered. A way out of this problem is the introduction (taken from [ABKR89]) of a distinction between *dependent* and *independent* resumptions. This works as follows: labels in transitions are now from $A \cup M \times R$, and the r' occurring in a transition $r_1 \to^{\langle m,r' \rangle} r_2$ is the so-called dependent resumption which may resume only after the rendez-vous for m has taken place (and r_2 is the independent resumption). The induced modifications in $\mathcal{T}_{rv}$ are (besides the fact that e now ranges over $A \cup M \times R$):

- $a;r \to^a r$
  $m;r \to^{\langle m,r \rangle} \mathrm{E}$
  (these rules together replace the rule $e;r \to^e r$ of the former $\mathcal{T}_{rv}$).
- the (par.comp.) rule is as before (but note that it now affects only the independent resumption).
- $$\frac{r_1 \to^{\langle m,r_1'' \rangle} r_1',\ r_2 \to^{\langle \bar{m},r_2'' \rangle} r_2',\ (h;(r_1'',r_2''),(r_1',r_2')) \to^e r}{(r_1,r_2) \to^e r},\ d(m) = h$$

(revised rendez-vous)

Next, in the definition of $\mathbb{P}$ we replace M by $M \times \mathbb{P}$. Finally, we change the definition of $\Phi_d(F)(r)$, for $r \neq \mathrm{E}$, to read

$$\Phi_d(F)(r) = \{ \langle a,F(r') \rangle \mid r \to^a r' \} \cup \{ \langle m,F(r''),F(r') \rangle \mid r \to^{\langle m,r'' \rangle} r' \},$$

with $\to$ with respect to the amended $\mathcal{T}_{rv}$. (We shall not address below how the denotational definitions and the proof of $\mathcal{O} = \mathcal{D}$ have to be modified accordingly.)

EXAMPLE Let $d(m) = a;m'$. We have, in the revised system, that

$(m;\mathrm{E},\bar{m}';\mathrm{E}) \to^{\langle m,\mathrm{E} \rangle} \bar{m}';\mathrm{E}$

$\bar{m};\mathrm{E} \to^{\langle \bar{m},\mathrm{E} \rangle} \mathrm{E}$

$(a;m';\mathrm{E},\bar{m}';\mathrm{E}) \to^a (m';\mathrm{E},\bar{m}';\mathrm{E})$

Hence, by the revised rendez-vous rule, we infer that

$((m;\mathrm{E},\bar{m}';\mathrm{E}),\bar{m};\mathrm{E}) \to^a (m';\mathrm{E},\bar{m}';\mathrm{E})$.

6.3 Denotational semantics

We define the mutually dependent $\mathcal{S}_d : \mathcal{L}_{rv} \to \mathbb{P} \to^1 \mathbb{P}$ and $\parallel\ : \mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}$ as simultaneous fixed points of the higher-order Ψ_d, Ω_d. The definition for Ψ_d uses auxiliary Ψ_d', Ψ_d'' which are defined for $g \in \mathcal{L}_{rv}^g$ and $h \in \mathcal{L}_{rv}^h$ (rather than for $s \in \mathcal{L}_{rv}$). The separate introduction of Ψ_d', Ψ_d'' is necessary to obtain contractivity of Ψ_d.

DEFINITION 6.8 Let $F \in \mathcal{L}_{rv} \to \mathbb{P} \to^1 \mathbb{P}$, $\phi \in \mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}$. The mappings

$$\begin{array}{llll}
\Psi_d & : (\mathcal{L}_{rv} \to \mathbb{P} \to^1 \mathbb{P}) \times (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}) & \to & (\mathcal{L}_{rv} \to \mathbb{P} \to^1 \mathbb{P}) \\
\Psi_d' & : (\mathcal{L}_{rv} \to \mathbb{P} \to^1 \mathbb{P}) \times (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}) & \to & (\mathcal{L}_{rv}^g \to \mathbb{P} \to^1 \mathbb{P}) \\
\Psi_d'' & : (\mathcal{L}_{rv} \to \mathbb{P} \to^1 \mathbb{P}) \times (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}) & \to & (\mathcal{L}_{rv}^h \to \mathbb{P} \to^1 \mathbb{P}) \\
\Omega_d & : (\mathcal{L}_{rv} \to \mathbb{P} \to^1 \mathbb{P}) \times (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P}) & \to & (\mathbb{P} \times \mathbb{P} \to^1 \mathbb{P})
\end{array}$$

are defined as follows:

$$\Psi_d F \phi\, e\, p = \Psi_d' F \phi\, e\, p = \Psi_d'' F \phi\, e\, p = \{ \langle e,p \rangle \}$$
$$\Psi_d F \phi\, x\, p = \Psi_d' F \phi\, d(x) p$$
$$\Psi_d F \phi (s_1 ; s_2) p = \Psi_d F \phi\, s_1 (\Psi_d F \phi\, s_2\, p)$$
$$\Psi_d' F \phi (g_1 ; g_2) p = \Psi_d' F \phi\, g_1 (\Psi_d' F \phi\, g_2\, p)$$
$$\Psi_d'' F \phi (h ; s) p = \Psi_d F \phi\, h (Fsp)$$
$$\Psi_d F \phi\, \mathbf{new}(s) p = \Omega_d F \phi (\Psi_d F \phi\, s\, p_0) p$$
$$\Psi_d' F \phi\, \mathbf{new}(g) p = \Omega_d F \phi (\Psi_d' F \phi\, g\, p_0) p$$
$$\Psi_d F \phi (s_1 + s_2) p = (\Psi_d F \phi\, s_1\, p) \cup (\Psi_d F \phi\, s_2\, p) \text{ and similarly for } \Psi_d', \Psi_d''$$
$$\Omega_d F \phi\, p\, p_0 = \Omega_d F \phi\, p_0\, p = p$$
$$\Omega_d F \phi p_1 p_2 = (\mathring{\Omega}_d F \phi p_1 p_2) \cup (\mathring{\Omega}_d F \phi p_2 p_1) \cup (\Omega^{|}_d F \phi p_1 p_2) \quad \text{if } p_1, p_2 \neq p_0$$

where

$$\mathring{\Omega}_d F \phi p_1 p_2 = \{ \langle e, \phi(p', p_2) \rangle \mid \langle e, p' \rangle \in p_1 \}$$
$$\Omega^{|}_d F \phi p_1 p_2 = \cup \{ \Psi_d'' F \phi\, h\, \phi(p', p'') \mid \exists m : \langle m, p' \rangle \in p_1, \langle \bar{m}, p'' \rangle \in p_2, d(m) = h \}.$$

Moreover, we put $(\mathcal{S}_d, \|) = \mathit{fix}(\Psi_d, \Omega_d)$ and $\mathcal{D}(d,s) = \mathcal{S}_d(s)(p_0)$.

EXPLANATION Using $\Psi_d(\mathcal{S}_d, \|) = \mathcal{S}_d$, $\Omega_d(\mathcal{S}_d, \|) = \|$, putting $\mathbin{\|\!\lfloor} = \mathring{\Omega}_d(\mathcal{S}_d, \|)$, $| = \Omega^{|}_d(\mathcal{S}_d, \|)$, and using part j of Lemma 6.9, we obtain the equalities

$$\mathcal{S}_d\, \mathbf{new}(s) p = (\mathcal{S}_d\, s\, p_0) \| p$$
$$p_1 \| p_2 = (p_1 \mathbin{\|\!\lfloor} p_2) \cup (p_2 \mathbin{\|\!\lfloor} p_1) \cup (p_1 | p_2)$$
$$p_1 | p_2 = \cup \{ \mathcal{S}_d(h)(p' \| p'') \mid \exists m : \langle m, p' \rangle \in p_1, \langle \bar{m}, p'' \rangle \in p_2, d(m) = h \}.$$

Note that the last of these equations is the denotational counterpart of the operational rendez-vous rule. The terms $p_1 \mathbin{\|\!\lfloor} p_2$ and $p_2 \mathbin{\|\!\lfloor} p_1$ in the second equation describe individual one-sided steps which do not lead to communication.

A lemma justifying Definition 6.8 follows:

LEMMA 6.9 *For all relevant arguments:*

a. $\Psi_d, \Psi_d', \Psi_d''$ and Ω_d are well-defined.

b. $d(\Psi_d'' F \phi\, h\, p_1, \Psi_d'' F \phi\, h\, p_2) \leq \frac{1}{2} d(p_1, p_2)$

c. $d(\Psi_d'' F_1 \phi_1, \Psi_d'' F_2 \phi_2) \leq \frac{1}{2} d(\langle F_1, \phi_1 \rangle, \langle F_2, \phi_2 \rangle)$

d. $d(\Omega_d F \phi\, p_1\, p_2, \Omega_d F \phi\, p_1'\, p_2') \leq d(p_i, p_i') \quad i = 1,2$

e. $d(\Omega_d F_1 \phi_1, \Omega_d F_2 \phi_2) \leq \frac{1}{2} d(\langle F_1, \phi_1 \rangle, \langle F_2, \phi_2 \rangle)$

f. $d(\Psi_d' F \phi\, g\, p_1, \Psi_d' F \phi\, g\, p_2) \leq d(p_1, p_2)$

g. $d(\Psi_d F \phi\, s\, p_1, \Psi_d F \phi\, s\, p_2) \leq d(p_1, p_2)$

h. $d(\Psi_d' F_1 \phi_1, \Psi_d' F_2 \phi_2) \leq \frac{1}{2} d(\langle F_1, \phi_1 \rangle, \langle F_2, \phi_2 \rangle)$

i. $d(\Psi_d F_1 \phi_1, \Psi_d F_2 \phi_2) \leq \frac{1}{2} d(\langle F_1, \phi_1 \rangle, \langle F_2, \phi_2 \rangle)$

j. $\Psi_d'' \mathcal{S}_d \| h = \mathcal{S}_d h$, $\Psi_d' \mathcal{S}_d \| g = \mathcal{S}_d g$.

PROOF We prove a few selected subcases, organized by the various possibilities for the s, g or h. We use 1 or 2 to abbreviate $\langle F_1, \phi_1 \rangle$ or $\langle F_2, \phi_2 \rangle$.

b. Case a.

$d(\Psi_d''F\phi ap_1, \Psi_d''F\phi ap_2) = d(\{\langle a,p_1\rangle, \{\langle a,p_2\rangle\}) = \frac{1}{2}d(p_1,p_2)$

Case $h;s$.

$d(\Psi_d''F\phi(h;s)p_1, \Psi_d''F\phi(h;s)p_2) =$

$d(\Psi_d''F\phi h(Fsp_1), \Psi_d''F\phi h(Fsp_2)) \leq (ind.)$

$\frac{1}{2}d(Fsp_1, Fsp_2) \leq (Fn.d.i.\ in\ p)\ \frac{1}{2}d(p_1,p_2)$

c. Case $h;s$. Take any p.

$d(\Psi_d''1(h;s)p, \Psi_d''2(h;s)p) \leq$ (def. Ψ_d'', d an ultrametric)

$max\{\ d(\Psi_d''1h(F_1sp), \Psi_d''1h(F_2sp))\ (*), d(\Psi_d''1h(F_2sp), \Psi_d''2h(F_2sp))\ (**)\ \}$

$(*) \leq (part\ b)$

$\frac{1}{2}d(F_1sp, F_2sp) \leq \frac{1}{2}d(F_1,F_2) \leq \frac{1}{2}d(1,2)$

$(**) \leq (ind.)\ \frac{1}{2}d(1,2)$

e. Take arbitrary p_1,p_2. We only consider the terms $\Omega^|_d1p_1p_2$ and $\Omega^|_d2p_1p_2$:

$d(\Omega^|_d1p_1p_2, \Omega^|_d2p_1p_2) =$

$d(\cup\{\ \Psi_d''1h(\phi_1p'p'') \mid \exists m{:}\langle m,p'\rangle \in p_1, \langle \bar{m},p''\rangle \in p_2, d(m)=h\ \},$

$\cup\{\ \Psi_d''2h(\phi_2p'p'') \mid \exists m{:}\langle m,p'\rangle \in p_1, \langle \bar{m},p''\rangle \in p_2, d(m)=h\ \}) \leq$

$sup\{\ d(\Psi_d''1h(\phi_1p'p''), \Psi_d''2h(\phi_2p'p'')) \mid \exists m{:}\langle m,p'\rangle \in p_1, \langle \bar{m},p''\rangle \in p_2, d(m)=h\ \} \leq$

$sup\{\ max\{\ d(\Psi_d''1h(\phi_1p'p''), \Psi_d''1h(\phi_2p'p''))\ (*), d(\Psi_d''1h(\phi_2p'p''), \Psi_d''2h(\phi_2p'p''))\ (**)\ \}\ \},$

$(*) \leq part\ c\ \frac{1}{2}d(\phi_1,\phi_2) \leq d(1,2)$

$(**) \leq (ind.)\ \frac{1}{2}d(1,2).$

g. Case $(s_1;s_2)$.

$d(\Psi_dF\phi(s_1;s_2)p_1, \Psi_dF\phi(s_1;s_2)p_2) =$

$d(\Psi_dF\phi s_1(\Psi_dF\phi s_2p_1), (\Psi_dF\phi s_1(\Psi_dF\phi s_2p_2)) \leq (ind.)$

$d(\Psi_dF\phi s_2p_1, \Psi_dF\phi s_2p_2) \leq (ind.)$

$d(p_1,p_2).$

Case **new**(s).

$d(\Psi_dF\phi\mathbf{new}(s)p_1, \Psi_dF\phi\mathbf{new}(s)p_2) =$

$d(\Omega_dF\phi(\Psi_dF\phi sp_0)p_1, \Omega_dF\phi(\Psi_dF\phi sp_0)p_2) \leq (part\ d.)$

$d(p_1,p_2).$

h. Case **new**(g). Take any p.

$d(\Psi_d'1\mathbf{new}(g)p, \Psi_d'2\mathbf{new}(g)p) =$

$d(\Omega_d1(\Psi_d'1gp_0)p, \Omega_d2(\Psi_d'2gp_0)p \leq (as\ usual)$

$max\{\ d(\Omega_d1(\Psi_d'1gp_0)p, \Omega_d1(\Psi_d'2gp_0)p)\ (*), d(\Omega_d1(\Psi_d'2gp_0)p, \Omega_d2(\Psi_d2gp_0)p)\ (**)\ \}$

$(*) \leq (part\ d)\ d(\Psi_d'1gp_0, \Psi_d'2gp_0) \leq (ind.)\ \frac{1}{2}d(1,2)$

$(**) \leq (part\ e)\ \frac{1}{2}d(1,2).$

i. Case x. Take any p.

$d(\Psi_d1xp, \Psi_d2xp) =$

$d(\Psi_d'1d(x)p, \Psi_d'2d(x)p \leq (part\ h)\ \frac{1}{2}d(1,2).$

Case $(s_1;s_2)$. Take any p.

$d(\Psi_d1(s_1;s_2)p, \Psi_d2(s_1;s_2)p) =$

$d(\Psi_d1s_1(\Psi_d1s_2p), \Psi_d2s_1(\Psi_d2s_2p)) \leq (as\ usual)$

$max\{\ d(\Psi_d1s_1(\Psi_d1s_2p), \Psi_d1s_1(\Psi_d2s_2p))\ (*), d(\Psi_d1s_1(\Psi_d2s_2p), \Psi_d2s_1(\Psi_d2s_2p))\ (**)\ \}$

$(*) \leq (part\ g)\ d(\Psi_d1s_2p, \Psi_d2s_2p) \leq (ind.)\ \frac{1}{2}d(1,2)$

$(**) \le (ind.)\ \frac{1}{2} d(1,2)$.

j. Case $(h;s)$. Take any p.

$$\mathcal{S}_d(h;s)p = \Psi_d\mathcal{S}_d \,\|\, (h;s)p = \Psi_d\mathcal{S}_d \,\|\, h(\Psi_d\mathcal{S}_d sp) = \mathcal{S}_d h(\mathcal{S}_d sp) = (\text{ind.})\ \Psi_d''\mathcal{S}_d \,\|\, h(\mathcal{S}_d sp) = \Psi_d''\mathcal{S}_d \,\|\, (h;s)p.$$

REMARK Note how the proof of part e builds on part c which is stated (at this moment in the proof) for Ψ_d'' only. This explains the earlier restriction that $d(m) \in \mathcal{L}^h_{rv}$.

6.4 $\mathcal{O}$ and $\mathcal{D}$ are equivalent

We first define $\mathcal{E}_d$ in a similar way as in Section 4 (cf. Definition 4.14):

DEFINITION 6.10 $\mathcal{E}_d : R \to \mathbb{P}$ is given by

$$\begin{aligned} \mathcal{E}_d(\mathrm{E}) &= p_0, \\ \mathcal{E}_d(s;r) &= \mathcal{S}_d(s)\mathcal{E}_d(r), \\ \mathcal{E}_d(r_1,r_2) &= \mathcal{E}_d(r_1) \,\|\, \mathcal{E}_d(r_2). \end{aligned}$$

We have the usual

LEMMA 6.11 *If $r_1 \twoheadrightarrow r_2$ by (recursion), (seq.comp.) or (new) then $\mathcal{E}_d(r_1) = \mathcal{E}_d(r_2)$. If $r_1 \twoheadrightarrow r_2$ and $r_1 \twoheadrightarrow r_3$ (by (choice)) then $\mathcal{E}_d(r_1) = \mathcal{E}_d(r_2) \cup \mathcal{E}_d(r_3)$.*

PROOF Standard. □

In order to be able to obtain the main technical result (viz. $\Phi_d(\mathcal{E}_d) = \mathcal{E}_d$, see Lemma 6.13), we need some auxiliary facts:

LEMMA 6.12

a. *If $h;r_1 \to^e r_2$, then $e \in A$.*

b. *If $r_1 \to^m r_2$ then $k(r_1) > k(r_2)$.*

c. *If $(r_1,r_2) \to^e r$ has been obtained by an application of the rendez-vous rule, then $l(r_1,r_2) > l(h;(r',r''))$.*

PROOF

a. Clear by the definition of $\mathcal{L}^h_{rv}$.

b. Induction on $l(r_1)$. We consider a few typical subcases, depending on how $r_1 \to^m r_2$ was obtained.

(el.action) Then $r_1 = m;r'$, $r_1 = r'$, $k(r_1) = k(m;r') = 1 + k(r') > k(r') = k(r_2)$.

(recursion) Then r_1 is of the form $x;r$ and the rule

$$\frac{g;r \to^m r_2}{x;r \to^m r_2}$$

has been applied. Since $k(x;r) = 1 + k(r) > k(r) \ge k(g;r)$, we have $l(g;r) < l(x;r)$. By induction, $k(g;r) > k(r_2)$, and $k(x;r) > k(r_2)$ follows.

(rendez-vous) By part a, this case cannot occur.

c. By part b, if $r_1 \to^m r'$ and $r_2 \to^{\bar{m}} r''$ then $k(r_1) > k(r') \ge 0$ and $k(r_2) > k(r'') \ge 0$. Hence $k(r_1,r_2) > k(h;(r',r'')) = 0$, and $l(r_1,r_2) > l(h;(r',r''))$ follows.

We are now ready for

LEMMA 6.13 *For all* r, $\Phi_d(\mathcal{E}_d)(r) = \mathcal{E}_d(r)$.

PROOF We use induction on $l(r)$. The interesting case is $r = (r_1, r_2)$. We have

$$\begin{aligned}
&\Phi_d(\mathcal{E}_d)(r_1,r_2)\\
=\ &\{\langle e,\mathcal{E}_d(r')\rangle \mid r_1 \to^e r'\} \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup \{\langle e,\mathcal{E}_d(r'')\rangle \mid r_2 \to^e r''\} \mathbin{\|\!\!_} \mathcal{E}_d(r_1) \cup\\
&\{\langle e,\mathcal{E}_d(\bar{r})\rangle \mid r_1 \to^m r', r_2 \to^{m} r'', h;(r',r'') \to^e \bar{r}, d(m) = h\}\\
=\ &\Phi_d(\mathcal{E}_d)(r_1) \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup \Phi_d(\mathcal{E}_d)(r_2) \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup\\
&\cup\{\Phi_d(\mathcal{E}_d)(h;(r',r'')) \mid r_1 \to^m r', r_2 \to^m r'', d(m) = h\}\\
=\ &(\text{ind.})\ \mathcal{E}_d(r_1) \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup \mathcal{E}_d(r_2) \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup\\
&\cup\{\mathcal{E}_d(h;(r',r'')) \mid r_1 \to^m r', r_2 \to^{\bar{m}} r'', d(m) = h\}\\
=\ &(\text{def. } \mathcal{E}_d, \Phi_d)\ \mathcal{E}_d(r_1) \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup \mathcal{E}_d(r_2) \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup\\
&\cup\{\mathcal{S}_d(h)(p' \parallel p'') \mid \langle m,p'\rangle \in \Phi_d(\mathcal{E}_d)(r_1), \langle \bar{m},p''\rangle \in \Phi_d(\mathcal{E}_d)(r_2), d(m) = h\}\\
=\ &(\text{def.})\ \mathcal{E}_d(r_1) \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup \mathcal{E}_d(r_2) \mathbin{\|\!\!_} \mathcal{E}_d(r_2) \cup \Phi_d(\mathcal{E}_d)(r_1) \mid \Phi_d(\mathcal{E}_d)(r_2)\\
=\ &(\text{ind., def } \parallel)\ \mathcal{E}_d(r_1) \parallel \mathcal{E}_d(r_2)\\
=\ &(\text{def. } \mathcal{E}_d)\ \mathcal{E}_d(r_1,r_2). \ \square
\end{aligned}$$

Finally, we conclude with our main

THEOREM 6.14 *For all* $s \in \mathcal{L}_{rv}$, $d \in Decl_{rv}$, $\mathcal{O}(d,s) = \mathcal{D}(d,s)$.

PROOF Follows from Lemma 6.13 by the familiar argument. □

7. REFERENCES

[A89] P.H.M. America, *Issues in the design of a parallel object-oriented language,* Formal Aspects of Computing 1 (1989), pp. 366-411.

[AB88] P.H.M. America, J.W. de Bakker, *Designing equivalent semantic models for process creation,* Theoretical Computer Science 60 (1988) 109-176.

[ABKR86] P.H.M. America, J.W. de Bakker, J.N. Kok, J.J.M.M. Rutten, *Operational semantics of a parallel object-oriented language,* 13th ACM Symposium on Principles of Programming Languages, St. Petersburg, Florida, January 13-15, 1986, pp. 194-208.

[ABKR89] P.H.M. America, J.W. de Bakker, J.N. Kok, J.J.M.M. Rutten, *Denotational semantics of a parallel object-oriented language,* Information and Computation, Vol. 83, pp. 152-205, 1989.

[AR89a] P.H.M. America, 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) pp. 1-49.

[AR89b] P.H.M. America, J.J.M.M. Rutten, *Solving reflexive domain equations in a category of complete metric spaces,* Journal of Computer and System Sciences 39, (1989) 343-375.

[AR90] P.H.M. America, J.J.M.M. Rutten, *A layered semantics for a parallel object-oriented language,* CS-R9052, CWI, Amsterdam, 1990, to appear in Springer LNCS 489.

[ANS83] ANSI. *The Programming Language ADA Reference Manual,* ANSI/MIL-STD 1815A-1983, Vol. 155, Springer, 1983.

[B88] J.W. de Bakker, *Comparative semantics for flow of control in logic programming without logic,* Report CS-R8840, CWI, Amsterdam (1988), revised version to appear in Information and Computation.

[B89] J.W. de Bakker, *Designing concurrency semantics,* in: Proc. 11th World Computer Congress (G.X. Ritter, ed.), North Holland, 1989, pp. 591-598.

[BBKM84] J.W. de Bakker, J.A. Bergstra, J.W. Klop, J.-J.Ch. Meyer, *Linear time and branching time semantics for recursion with merge,* Theoretical Computer Science 34 (1984) 135-156.

[BK90] J.W. de Bakker, J.N. Kok, *Comparative metric semantics for Concurrent Prolog,* Theoretical Computer Science 75 (1990), 15-44.

[BKMOZ86] J.W. de Bakker, J.N. Kok, J.-J.Ch. Meyer, E.-R. Olderog, J.I. Zucker, *Contrasting themes in the semantics of imperative concurrency,* in Current Trends in Concurrency: Overviews and Tutorials (J.W. de Bakker, W.P. de Roever, G. Rozenberg, eds.), LNCS 224, Springer (1986) 51-121.

[BM88] J.W. de Bakker, J.-J.Ch. Meyer, *Metric semantics for concurrency,* BIT 28, pp. 504-529, 1988.

[BMOZ88] J.W. de Bakker, J.-J.Ch. Meyer, E.-R. Olderog, J.I. Zucker, *Transition systems, metric spaces and ready sets in the semantics of uniform concurrency,* Journal of Computer and System Sciences 36 (1988), 158-224.

[BZ82] J.W. de Bakker, J.I. Zucker, *Processes and the denotational semantics of concurrency,* Information and Control 54 (1982) 70-120.

[BV91] J.W. de Bakker, E.P. de Vink, *Rendez-vous with metric semantics,* Proc. PARLE 91, LNCS, Springer, to appear.

[BeK87] J.A. Bergstra, J.W. Klop, *A convergence theorem in process algebra,* Report CS-8733, CWI , Amsterdam, 1987.

[BoKPR90] F.S. de Boer, J.N. Kok, C. Palamidessi, J.J.M.M. Rutten, *From failure to success: Comparing a denotational and a declarative semantics for Horn Clause Logic,* in Proc. of the international BCS-FACS Workshop on Semantics for Concurrency (M.Z. Kwiatkowska, M.W. Shields, R.M. Thomas, eds.), Workshops in Computing, Springer (1990), 38-60.

[BoKPR91] F.S. de Boer, J.N. Kok, C. Palamidessi, J.J.M.M. Rutten, *The failure of failures: towards a paradigm for asynchronous communication,* Report, CS-R91.., CWI, to appear.

[Br91] F. van Breugel, *Comparative semantics for a real-time programming language with integration,* in Proc. TAPSOFT 91 (S. Abramsky, T.S.E. Maibaum, eds.) vol. 1, LNCS 493, Springer, pp 397-411, 1991.

[DeBr86] A. de Bruin, *Exercises in continuation semantics: jumps, backtracking, dynamic networks,* PhD Thesis, Vrije Universiteit Amsterdam, 1986.

[BrVi89] A. de Bruin, E.P. de Vink, *Continuation semantics for PROLOG with cut,* Proc. TAPSOFT 89, Vol I (J. Diaz, F. Orejas, eds.), LNCS 351, Springer, pp. 178-192, 1989.

[Ha48] H. Hahn, *Reelle Funktionen*, Chelsea 1948.

[JaMo90] J.-M. Jacquet & L. Monteiro, *Comparative Semantics for a Parallel Contextual Programming Language*, in Proc. North-American Logic Programming Conference (S. Debray, M. Hermenegildo, eds.) pp. 195-214, MIT Press, 1990

[K88] J.N. Kok, *A compositional semantics for Concurrent Prolog*, in Proc. 5th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, February 1988 (R. Cori, M. Wirsing, eds.), LNCS 294, pp. 373-388.

[KR90] J.N. Kok, J.J.M.M. Rutten, *Contractions in comparing concurrency semantics*, Theoretical Computer Science 76, pp. 180-222 (1990).

[Ku56] K. Kuratowski, *Sur une méthode de métrisation complète des certains espaces d'ensembles compacts*, Fundamenta Mathematicae 42 (1956), pp. 114-138.

[Mic51] E. Michael, *Topologies on spaces of subsets*, Transactions of the AMS 71, 1951, pp. 152-182.

[Mi80] R. Milner, *A Calculus for Communicating Systems*, LNCS 92, Springer, 1980.

[Ni77] M. Nivat, *Mots infinis engendrés par une grammaire algébrique*, RAIRO Informatique Théorique 11 (1977) pp. 311-327.

[Ni78] M. Nivat, *Sur les ensembles de mots infinis engendrés par une grammaire algabrique*, RAIRO Informatique Théorique 12 (1978), pp. 259-278.

[R90a] J.J.M.M. Rutten, *Semantic correctness for a parallel object-oriented language*, SIAM Journal on Computing 19, 1990, pp. 341-383.

[R90b] J.J.M.M. Rutten, *Deriving metric models for bisimulation from transition system specifications*, in Proc. IFIP TC2 Working Conference on Programming Concepts and Methods, North-Holland, 1990, pp. 148-170.

[Vi90] E.P. de Vink, *Comparative semantics for Prolog with cut*, Science of Computer Programming 13 (1990), pp. 237-264.