Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.W. de Bakker, J.-J.Ch. Meyer

Metric semantics for concurrency

Computer Science/Department of Software Technology Report CS-R8803 January

Bibliotheek
Centrum voor Wiskunde en Informatica
Amsterdam

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

69D13, 69D41, 69D43, 69F12, 69F32.

Copyright © Stichting Mathematisch Centrum, Amsterdam

METRIC SEMANTICS FOR CONCURRENCY

J.W. de Bakker

Centre for Mathematics and Computer Science
Kruislaan 413, NL-1098 SJ Amsterdam /
Free University of Amsterdam

J.-J.Ch. Meyer

Free University of Amsterdam
De Boelelaan 1081, NL-1081 HV Amsterdam

ABSTRACT

An overview is given of work we have done in recent years on the semantics of concurrency, concentrating on semantic models built on metric structures. Three contrasting themes are discussed, viz. (i) uniform or schematic versus nonuniform or interpreted languages; (ii) operational versus denotational semantics, and (iii) linear time versus branching time models. The operational models are based on Plotkin's transition systems. Language constructs which receive particular attention are recursion and merge, synchronization and global nondeterminacy, process creation, and communication with value passing. Various semantic equivalence results are established. Both in the definitions and in the derivation of these equivalences, essential use is made of Banach's theorem for contracting functions.

Keywords: concurrency, operational semantics, denotational semantics, transition systems, process creation, synchronization, metric spaces, domain equations, contracting functions, global nondeterminacy.

1985 Mathematics Subject Classification: 68Q55, 68Q10

1987 Computing Reviews Categories: D.1.3, D.3.1, D.3.3, F.1.2, F.3.2.

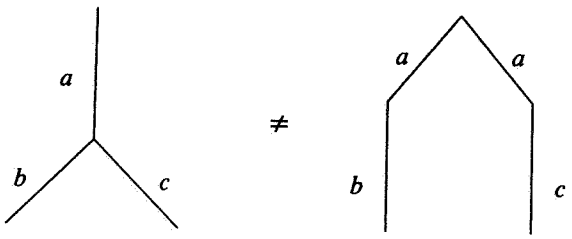
1. Introduction

We present an expository account of work we have been pursuing in recent years on the semantics of concurrency, concentrating on those models which are built on structures from metric topology. We shall exhibit semantic definitions for a variety of programming notions relating to concurrency, viz. recursion with merge (parallel execution in the interleaving sense), synchronization and global nondeterminacy, process creation, and communication with value passing. We hope to demonstrate the power of metric methods, both in the semantic definitions themselves and in the establishment of particularly succinct derivations of equivalence results between

Report CS-R8803
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

operational and denotational semantic models.

Three contrasting themes will recur in our considerations (cf [BKMOZ] for a more elaborate treatment). Firstly, there is the familiar distinction between operational and denotational semantics. The former will always be based on *transition systems* which are variations on the elegant systems of Hennessy and Plotkin ([HP],[Pl2],[Pl3]). The latter will throughout be defined compositionally, with (unique) fixed points to deal with recursion. Such fixed points exist on the basis of Banach's theorem for contracting functions. In fact, this theorem is absolutely pervasive in our technical considerations: a good deal of our definitions and theorems ultimately rely on it. Secondly, we shall contrast *uniform* and *nonuniform* languages. The former are *schematic* in the sense that their elementary actions are uninterpreted, and the meanings rendered by our definitions involve entities with a strong flavour of formal language theory. More specifically, sets of (possibly infinite) words or tree-like objects are delivered. Nonuniform languages have interpreted elementary actions. They include notions such as (individual) variables, assignments, states and state transforming functions. As we shall demonstrate, it requires additional tools to set up a framework in which one may merge such functions. Thirdly, we shall be concerned with both linear time (LT) and branching time (BT) models. Typical examples are sets of words versus trees (with some further properties not stated here) over some alphabet A. In the former, moments of choice are abstracted away which are present in the latter. We recall the classical example of the LT set $\{ab, ac\}$ versus the two different trees in BT:



The genealogy of the work described in the present paper is as follows: Ancestors are Nivat's work on metric techniques in semantics ([Ni]) and Plotkin's work on resumptions in power domains ([Pl1]). In [BZ1] we described a general method to solve domain equations using metric techniques. [BZ2] is an example of a specific semantic application. A substantial improvement on [BZ1] is given in [AR] where the scope of the method in [BZ1] was clarified and, even better, considerably generalized. A comparison of LT and BT models for recursion with merge was first made in [BBKM]. In [BMOZ], [BKMOZ] a systematic comparison of operational and denotational models was developed, both for recursion and merge, for synchronization with (forms of) nondeterminacy, and for nonuniform languages. Somewhat simultaneously we have devoted a number of papers to the design of semantic models for the parallel object oriented language POOL ([ABKR1, ABKR2, AB]), dealing, besides with various other notions, with process creation. An essential step on the way to substantial simplification of the sometimes quite elaborate arguments in [BMOZ], [AB] was performed in [KR]. Here the full

power of the unique fixed point argument, not only in defining but also in comparing semantic models, was first exploited.

In parallel to the metrically based semantic studies, we have also continued to work with models based on partial orders, were it only to relate order-theoretic models to metric ones. In addition, for the metric models as we use them, the requirement that all sets considered be *closed* is vital, and the metric theory fails when phenomena inducing nonclosed sets are encountered. Examples of comparative studies, in particular relating to the 'elemental' combination of recursion with merge, are [BM], [BMO]. An extensive application of order-theoretic tools, specifically to deal with fair merge (the result of which is in general nonclosed) is described in [M]. Another language notion which is not directly amenable to metric techniques is that of *hiding* (cf. [MO]). Finally, we mention [MV] where an order-theoretic counterpart of the topological notion of compactness is studied. In fact, we might have paid some attention to (consequences of) compactness requirements below as well, but we have decided not to do so for reasons of space.

We are at present investigating further applications of the metric method in semantics. Two prime examples are uniform (or 'logicless') versions of logic programming, and more advanced concepts in object-oriented programming.

Acknowledgements

The work described below would not have been possible without the essential contributions of the Amsterdam concurrency group and its affiliates. We acknowledge in particular the work of Joost Kok and Jan Rutten who first realized the crucial role of contractivity arguments in comparing concurrency semantics. Pierre America, Ernst-Rüdiger Olderog and Jeff Zucker have worked together with us on the papers [BMOZ], [AB], both of which were instrumental for the present paper.

2. Mathematical Preliminaries

2.1 Notation

The phrase 'let $(x \in)X$ be such that ...' introduces a set X with variable x ranging over X such that .... For X a set, $\mathcal{P}(X)$ denotes the collection of all subsets of X, and $\mathcal{P}_\pi(X)$ is the collection of all subsets of X which have property π. The notation $f:X\to Y$ expresses that f is a function with domain X and range Y. We use the notation $f\{y/x\}$, with $x \in X$ and $y \in Y$, for a *variant* of f, i.e. for the function which is defined by

$$f\{y/x\}(x') = \begin{cases} y & \text{if } x = x' \\ f(x') & \text{otherwise} \end{cases}$$

If $f:X\to X$ and $f(x)=x$, we call x a *fixed point* of f.

2.2 Metric Spaces

From standard topology (e.g. [Du], [En]) we assume known the notion of *(ultra)metric space* $(\mathbf{M},\mathbf{d})$ with distance or *(ultra)metric* $\mathbf{d}$. We use the notions of *closed* subset X of $(\mathbf{M},\mathbf{d})$, of *continuous* mapping $(\mathbf{M}_1,\mathbf{d}_1)\to(\mathbf{M}_2,\mathbf{d}_2)$, of *completeness* of a metric space, and of *isometry* $(\cong)$ between metric spaces $(\mathbf{M}_1,\mathbf{d}_1)$ and $(\mathbf{M}_2,\mathbf{d}_2)$. A mapping $f:(\mathbf{M}_1,\mathbf{d}_1)\to(\mathbf{M}_2,\mathbf{d}_2)$ is called *contracting* whenever, for all $x,y\in\mathbf{M}_1$, we have $\mathbf{d}_2(f(x),f(y))\leqq\alpha\cdot\mathbf{d}_1(x,y)$, with $0\leqq\alpha<1$. If the same condition holds with $\alpha=1$, we call f *non distance increasing* (ndi). Clearly, a contracting or ndi mapping is continuous. A central role is played below by

Proposition 2.1 (Banach). *Let $f:(\mathbf{M},\mathbf{d})\to(\mathbf{M},\mathbf{d})$ be contracting, and let $(\mathbf{M},\mathbf{d})$ be complete. Then f has a unique fixed point x_0 and, for any y, $x_0=\lim_i f^i(y)$, where $f^0=\lambda x\cdot x$, $f^{i+1}=f\circ f^i$.*

2.3 Metric Spaces of (Sets of) Words

Let A be a (finite or infinite) alphabet, let $A^*(A^\omega)$ denote the collection of all finite (infinite) words over A, and let $A^\infty=^{df.}A^*\cup A^\omega$. Let ϵ denote the empty word. For each $u\in A^\infty$, $u(n)$ is the prefix of u of length n, if this exists, and $u(n)=u$, otherwise. We define a metric $\mathbf{d}$ on A^∞ by putting $\mathbf{d}(u,v)=2^{-n}$, where $n=sup\{k\mid u(k)=v(k)\}$. Thus, $\mathbf{d}(u,v)=2^{-\infty}=0$ if $u=v$. We have

Proposition 2.2. *$(A^\infty,\mathbf{d})$ is a complete ultrametric space.*

Let $\mathscr{S}=\mathscr{P}_{nc}(A^\infty)$ denote the collection of all nonempty closed subsets of A^∞. Let, for $X\in\mathscr{S}$, $X(n)=\{u(n)\mid u\in X\}$. We define a metric $\hat{\mathbf{d}}$ on $\mathscr{S}$ by putting $\hat{\mathbf{d}}(X,Y)=2^{-n}$, where $n=sup\{k\mid X(k)=Y(k)\}$. For example, $\hat{\mathbf{d}}(\{abc,ef\},\{abcd,efg\})=2^{-2}$. We have

Proposition 2.3. *$(\mathscr{S},\hat{\mathbf{d}})$ is a complete ultrametric space.*

On A^∞ and $\mathscr{S}$ we have the usual concatenation operator '$\cdot$'. For subsequent purposes, we are interested in the subset Q of $\mathscr{S}$ consisting of either $\{\epsilon\}$ or of elements X in $\mathscr{S}$ which do not contain ϵ.

2.4. Domain Equations and Resumptions

We briefly recall the notion of a (metric) *domain equation*. The general form of such an equation is

$$P\cong\mathscr{F}(P) \tag{2.1}$$

or, more precisely, $(P,\mathbf{d}_P) = \mathscr{F}((P,\mathbf{d}_P))$, where the mapping $\mathscr{F}$ (technically a functor from the category of complete metric spaces to itself, but we do not have to be aware of this) is built up as follows: $\mathscr{F}$ is either a *constant* (delivering some complete $(A,\mathbf{d}_A)$), a transformation id_α which maps $(\mathbf{M},\mathbf{d})$ to $(\mathbf{M},\alpha\cdot\mathbf{d})$ for some real α, or composed from already given components by operations such as cartesian product, disjoint union, (restricted) function spaces, or the '*closed subset of*' mapping. We have no room to discuss details which are described at length in [BZ1] or [AR] (see also [BK] for the connection between such P and spaces obtained through bisimulation from synchronization trees as, e.g., in [Mi]). It is sufficient to know that isometries such as

$$P \cong \{p_0\} \cup (A \times P) \tag{2.2}$$

$$P \cong \{p_0\} \cup \mathscr{P}_{closed}(A \times id_{1/2}(P)) \tag{2.3}$$

$$P \cong \{p_0\} \cup (A \to \mathscr{P}_{closed}(B \times id_{1/2}(P))) \tag{2.4}$$

all have well-defined solutions as complete metric spaces. (On later occasions, the mappings $id_{1/2}$ will, for simplicity, be assumed implicitly). Elements of such P are either finite (and then equal to p_0 or in some $P_{n+1} = \mathscr{F}(P_n)$), or infinite and then satisfy $p = \lim_n p_n$, with $p_n \in P_n$. Occurrences of P in terms $\ldots \times P$ on the right-hand side of these equations justify the terminology of *resumptions*: For example, for $p \in P$ with P as in (2.4), $p(\neq p_0)$ is a function which, when supplied with argument $a \in A$ turns itself into, among other things, some $<b,p'>$. In later applications we shall read this with the connotation: *process* p maps a to b and then turns itself into process p' as resumption.

For subsequent purposes, we note that, if the constant spaces $(A,\mathbf{d}_A)$, $(B,\mathbf{d}_B)$, ... are assumed to be ultrametric, then the solutions P (as in (2.2) to (2.4)) are also ultrametric.

Example: Elements from P as in (2.3) are, e.g., $\{<a,\{<b,p_0>,<c,p_0>\}>\}$ and $\{<a,\{<b,p_0>\}>,<a,\{<c,p_0>\}>\}$. These may be pictorially represented by the trees from section 1. No such distinction is present in the set Q, where both objects are represented by the set $\{ab,ac\}$.

3. Recursion and Merge

The first language we consider is a simple extension of the traditional (uniform) sequential languages, obtained by adding the programming construct of parallel execution or *merge* $s_1 \| s_2$ of the two statements s_1 and s_2. By a traditional (uniform) language we mean here a language which has (uninterpreted) elementary actions taken from some alphabet A, sequential composition, nondeterministic choice and recursion. It is well-known that these four concepts put together in the customary way - the exact syntax follows in a moments - yield the expressive power of context-free languages, here taken in the general sense of languages over finite and infinite words over A. Thus, we may rephrase the object of study in the present section as

infinitary context-free languages extended with merge or shuffle, where the latter notion is the standard operation of language theory. This combination of (basic notions with) recursion and merge was first studied in [BBKM] (denotational LT and BT models) and [BMOZ, BKMOZ] (operational vs. denotational LT models). The presentation below essentially follows [KR], though our returning here to the format of *simultaneous recursion* - rather than employing possibly nested μ-constructs - allows a considerably more concise treatment.

We build the syntax starting from

- a (not necessarily finite) alphabet A, with elements $a, b, c, \ldots$
- a set $\mathscr{Var}$ of procedure variables $x_1, x_2, \ldots$. It will be convenient to assume that each program uses exactly the procedure variables in the initial fragment $\mathscr{X} = \{x_1, \ldots, x_n\}$ of $\mathscr{Var}$, for some $n \geqq 0$.

We start with

Definition 3.1 (Syntax).

a (statements). The class $(s \in)\mathscr{L}_1$ of *statements* is given by

$$s ::= a \mid x \mid s_1;s_2 \mid s_1 \cup s_2 \mid s_1 \| s_2, \text{ with } x \in \mathscr{X}$$

b (guarded statements). The class $(g \in)\mathscr{L}_1^g$ of *guarded statements* is given by

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

c (declarations). The class $(D \in)\mathscr{Decl}_1$ of *declarations* consists of n-tuples $D \equiv x_1 \Leftarrow g_1, \ldots, x_n \Leftarrow g_n$ or $\langle x_i \Leftarrow g_i \rangle_i$, for short, with $x_i \in \mathscr{X}$ and $g_i \in \mathscr{L}_1^g$.

d (programs). The class $(t \in)\mathscr{Prog}_1$ of *programs* consists of pairs $t \equiv \langle D \mid s \rangle$, with $D \in \mathscr{Decl}_1$ and $s \in \mathscr{L}_1$.

Examples

1. $\langle$
   $x_1 \Leftarrow a;x_2 \cup b;x_3,$
   $x_2 \Leftarrow b \cup b;x_1 \cup a;x_3;x_3,$
   $x_3 \Leftarrow a \cup a;x_1 \cup b;x_2;x_2,$
   $\mid x_1 \rangle$
2. $\langle x \Leftarrow a;(b \| x) \mid (c \| x) \rangle$

Remarks

1. All g_i occurring in a declaration $D \equiv \langle x_i \Leftarrow g_i \rangle_i$ are required to be guarded, i.e. occurrences of $x \in \mathscr{X}$ in g_i are to be preceded by some g (which, by clause b, has to

start with an elementary action). This requirement corresponds to the usual Greibach condition in formal language theory.

2. We have adopted the simultaneous declaration format for recursion rather than the μ-formalism which features constructs such as, for example, $c \| \mu x [a ; (\mu y [b_1 ; y ; b_2 \cup b_3] \| x)]$. As remarked already, the avoidance of (nested) μ-constructs allows for a simpler derivation of the main semantic equivalence result to follow.

We proceed with the definitions leading up to the *operational semantics* for $s \in \mathscr{L}_1$ and $t \in \mathscr{P}rog_1$. It is convenient to extend $\mathscr{L}_1$ with a special 'empty statement' E which performs no action (it will obtain $\{\epsilon\}$ as its meaning). We put $\mathscr{L}_1' = \mathscr{L}_1 \cup \{E\}$. The operational semantics is based on *transitions* (following the operational semantics techniques of Structured Operational Semantics, cf. [HP, Pl2, Pl3]). Here, transitions are four-tuples in $\mathscr{L}_1 \times A \times \mathscr{D}ecl_1 \times \mathscr{L}_1'$, written in the notation

$$s \xrightarrow{a}_D s'$$

with $s \in \mathscr{L}_1$, $a \in A$, $D \in \mathscr{D}ecl_1$, $s' \in \mathscr{L}_1'$. We present a *formal transition system* T_1 which consists of *axioms* and *rules*. Transitions which are given as axioms *hold* by definition. Moreover, a transition which is the consequence of a rule holds in T_1 whenever it can be established that, according to T_1, its premise holds (or, in later sections, its premises hold). We shall employ below self-explanatory notational variants of the format for the rules. T_1 is given in

Definition 3.2 (transition system T_1). Let $s, s', \tilde{s} \in \mathscr{L}_1$, $a \in A$, $D \in \mathscr{D}ecl_1$.

$$a \xrightarrow{a}_D E \qquad (Elem)$$

$$\frac{s \xrightarrow{a}_D s' \mid E}{s ; \tilde{s} \xrightarrow{a}_D s' ; \tilde{s} \mid \tilde{s}} \qquad (SeqComp)$$

$$\frac{s \xrightarrow{a}_D s' \mid E}{\begin{array}{l} s \cup \tilde{s} \xrightarrow{a}_D s' \mid E \\ \tilde{s} \cup s \xrightarrow{a}_D s' \mid E \end{array}} \qquad (Choice)$$

$$\frac{g \xrightarrow{a}_D s' \mid E}{x \xrightarrow{a}_D s' \mid E} \quad \text{,with } x \Leftarrow g \text{ in } D \qquad (Rec)$$

$$\frac{s \xrightarrow{a}_D s' \mid E}{\begin{array}{l} s \| \tilde{s} \xrightarrow{a}_D s' \| \tilde{s} \mid \tilde{s} \\ \tilde{s} \| s \xrightarrow{a}_D \tilde{s} \| s' \mid \tilde{s} \end{array}} \qquad (ParComp)$$

We next define how to collect the successive transitions $s \xrightarrow{a}_D s'$, $s' \xrightarrow{b}_D s''$,..., starting from some $t \equiv \langle D \mid s \rangle$, into its operational meaning $\mathcal{O}[\![t]\!]$. We use Q as introduced in section 2.3.

Definition 3.3.

a. The mapping $\mathcal{O} : \mathcal{P}rog_1 \to Q$ is given by
$\mathcal{O}[\![\langle D \mid s \rangle]\!] = \mathcal{O}_D[\![s]\!]$.

b. The mapping $\mathcal{O}_D : \mathcal{L}_1' \to Q$ is given by: $\mathcal{O}_D[\![E]\!] = \{\epsilon\}$, and for $s \neq E$,

$$\mathcal{O}_D[\![s]\!] = \bigcup \{a \cdot \mathcal{O}_D[\![s']\!] \mid s \xrightarrow{a}_D s'\},$$

where the transitions are with respect to T_1.

It may not be obvious that the function $\mathcal{O}_D$ is well-defined. This is in fact a consequence of the following

Lemma 3.4. *Let the operator* $\Phi : (\mathcal{L}_1' \to Q) \to (\mathcal{L}_1' \to Q)$ *be defined as follows: For any* $\mathcal{F}_D : \mathcal{L}_1' \to Q$ *we put* $\Phi(\mathcal{F}_D)(E) = \{\epsilon\}$, *and, for* $s \neq E$,

$$\Phi(\mathcal{F}_D)(s) = \bigcup \{a \cdot \mathcal{F}_D[\![s']\!] \mid s \xrightarrow{a}_D s'\}.$$

Then Φ *is a contracting mapping with* $\mathcal{O}_D$ *as its fixed point.*

Proof. Clear from the definitions and Banach's theorem. □

Example. $\mathcal{O}[\![\langle x \Leftarrow a; x \cup b \mid x \rangle]\!] = \{a^\omega\} \cup a^* \cdot b$.

Remark. As explained in [BMOZ], if we were to drop the guardedness restriction for the g_i in D, the operational meaning of $\langle D \mid s \rangle$ (based on the definitions in [BMOZ]) is not necessarily a closed set, and definition 3.3 would not, in general, yield the desired result. (Definition 3.3 always gives closed sets as results.)

The next step is the development of the denotational model. We use Q as before, and now also define various semantic operators: $Q \times Q \to Q$, viz. the operators of union ('$\cup$'), composition ('$\circ$') and merge ('$\|$').

Definition 3.5. For each $X \in Q$ we write $X_a = \{u \in A^\infty \mid a \cdot u \in X\}$.

a. $X \cup Y = X$, if $Y = \{\epsilon\}$.
$X \cup Y = Y$, if $X = \{\epsilon\}$.
Otherwise, $X \cup Y$ equals the set-theoretic union of X and Y.

b. Let *op* stand for $\circ$ or $\|$. Let ϕ be any ndi mapping: $Q \times Q \to Q$. Let $\Phi_{op} : (Q \times Q \to Q) \to (Q \times Q \to Q)$ be defined as follows:

$$\Phi_{\circ}(\phi)(X)(Y) = \begin{cases} Y, \text{ if } X = \{\epsilon\} \\ \bigcup_{a \in A} \{a \cdot \phi(X_a)(Y) \mid X_a \neq \varnothing\}, \text{ otherwise} \end{cases}$$

$$\Phi_{\|}(\phi)(X)(Y) = \Phi_{\circ}(\phi)(X)(Y) \cup \Phi_{\circ}(\phi)(Y)(X)$$

c. We now put $\circ = \textit{fixed point}(\Phi_{\circ})$, $\| = \textit{fixed point}(\Phi_{\|})$.

We have

Lemma 3.6. *The operators* $\cup$, $\circ$, $\|$ *are well-defined and ndi (and, hence, continuous).*

Proof. Clear for $\cup$. For the other operators, another appeal to Banach's theorem suffices. □

The denotational semantic definitions employ the usual notion of *environment*. Let $(\gamma \in)\Gamma = \mathscr{Var} \rightarrow Q$ be the set of environments, i.e. of mappings from procedure variables to their meanings. We define

Definition 3.7 (denotational semantics for $\mathscr{L}_1, \mathscr{Prog}_1$). Below we often suppress parentheses around arguments of functions.

a. $\mathscr{M} : \mathscr{Prog}_1 \rightarrow Q$ is given by $\mathscr{M}[\![<D \mid s>]\!] = \mathscr{D}[\![s]\!]\gamma_D$.

b. $\gamma_D = \gamma\{X_i/x_i\}_{i=1}^n$, where, for $D \equiv <x_i \Leftarrow g_i>_i$, we put

$$<X_1,\ldots,X_n> = \textit{fixed point} <\Phi_1,\ldots,\Phi_n>$$

where $\Phi_j : Q^n \rightarrow Q$ is given by $\Phi_j(Y_1)\ldots(Y_n) = \mathscr{D}[\![g_j]\!]\gamma\{Y_i/x_i\}_i$.

c. $\mathscr{D}[\![a]\!]\gamma = \{a\}$, $\mathscr{D}[\![x]\!]\gamma = \gamma(x)$, $\mathscr{D}[\![s_1 \underline{op}\ s_2]\!]\gamma = \mathscr{D}[\![s_1]\!]\gamma\ op\ \mathscr{D}[\![s_2]\!]\gamma$, for $\underline{op} \in \{;, \cup, \|\}$ and $op \in \{\circ, \cup, \|\}$, respectively.

Examples. $\mathscr{D}[\![a;(b \cup c)]\!]\gamma = \mathscr{D}[\![(a;b) \cup (a;c)]\!]\gamma = \{ab, ac\}$.
$\mathscr{M}[\![<x \Leftarrow a;(b \| x) \mid x>]\!] = \lim_i X_i$, where $X_{i+1} = a \cdot (b \| X_i)$, and $X_0 \in Q$ is arbitrary.

Remark. The (unique) fixed point in clause b exists by the guardedness requirement which ensures the contractivity of the Φ_j.

The above definitions of the operational and denotational semantics have been tuned such that the proof of $\mathscr{O} = \mathscr{M}$ is now no longer a major undertaking (as it was in [BMOZ]). We follow the approach as in [KR] (cf. [HP], [AP] for a similar approach in an order-theoretic framework) with the additional simplifications due to our replacing μ-constructs by simultaneous

recursion. We prove

Theorem 3.8. *For all* $t \in \mathcal{P}rog_1$, $\mathcal{O}[\![t]\!] = \mathcal{M}[\![t]\!]$.

Proof. Let us put $\mathcal{D}_D[\![s]\!] =^{df.} \mathcal{D}[\![s]\!]\gamma_D$. By the definition of $\mathcal{O}$ and lemma 3.4, it is sufficient to show that, for $s \in \mathcal{L}_1$, (*):$\mathcal{D}_D[\![s]\!] = \Phi(\mathcal{D}_D)(s)$. The proof proceeds in two stages; first for $g \in \mathcal{L}_1^g$ and next for any $s \in \mathcal{L}_1$.

Stage 1. Take $g \in \mathcal{L}_1^g$. We prove (*) by induction on the complexity of g. We only treat the case that $g = g_1 \| g_2$, the other cases being simpler. We have:

$\Phi(\mathcal{D}_D)(g_1 \| g_2) =$ (def.Φ)

$\cup \{ a \cdot \mathcal{D}_D[\![\tilde{s}]\!] \mid g_1 \| g_2 \xrightarrow{a}_D \tilde{s} \} =$ (def.T_1)

$\cup \{ a \cdot \mathcal{D}_D[\![s' \| g_2]\!] \mid g_1 \xrightarrow{a}_D s' \} \cup \cup \{ a \cdot \mathcal{D}_D[\![g_1 \| s'']\!] \mid g_2 \xrightarrow{a}_D s'' \} =$ (def.$\mathcal{D}$)

$\cup \{ a \cdot (\mathcal{D}_D[\![s']\!] \| \mathcal{D}_D[\![g_2]\!]) \mid g_1 \xrightarrow{a}_D s' \} \cup$

$\cup \{ a \cdot (\mathcal{D}_D[\![g_1]\!] \| \mathcal{D}_D[\![s'']\!]) \mid g_2 \xrightarrow{a}_D s'' \} =$ (ind. hyp.)

$\cup \{ a \cdot (\mathcal{D}_D[\![s']\!] \| \Phi(\mathcal{D}_D)(g_2)) \mid g_1 \xrightarrow{a}_D s' \} \cup$

$\cup \{ a \cdot (\Phi(\mathcal{D}_D)(g_1) \| \mathcal{D}_D[\![s'']\!]) \mid g_2 \xrightarrow{a}_D s'' \} =$ (def.$\|, T_1$)

$\Phi(\mathcal{D}_D)(g_1) \| \Phi(\mathcal{D}_D)(g_2) =$ (ind. hyp.)

$\mathcal{D}_D[\![g_1]\!] \| \mathcal{D}_D[\![g_2]\!] =$ (def.$\mathcal{D}$)

$\mathcal{D}_D[\![g_1 \| g_2]\!]$.

Stage 2. Take $s \in \mathcal{L}_1$. We prove (*) by induction on the complexity of s. All cases are as in stage 1, but for the case $s \equiv x$, with $x \equiv x_i \in \mathcal{X}$. We have

$\Phi(\mathcal{D}_D)(x_i) = \cup \{ a \cdot \mathcal{D}_D[\![\tilde{s}]\!] \mid x_i \xrightarrow{a}_D \tilde{s} \} = \cup \{ a \cdot \mathcal{D}_D[\![\tilde{s}]\!] \mid g_i \xrightarrow{a}_D \tilde{s} \}$ (with $x_i \Leftarrow g_i$ in $\mathcal{D}$) = (by stage 1, the desired result holds for $g_i \in \mathcal{L}_1^g$) $\mathcal{D}_D[\![g_i]\!]$ = (by the fixed point property) $\mathcal{D}_D[\![x_i]\!]$. □

4. Synchronization and Global Nondeterminacy

We discuss an extension of $\mathcal{L}_1$ with two new features. Firstly, we add a form of synchronization in the tradition of CCS [Mi] or CSP [Ho]. Secondly, we replace the nondeterministic choice ($s_1 \cup s_2$) of section 3 by a new form of nondeterminism, written as $s_1 + s_2$. The latter is called *global* (sometimes also external) nondeterminism. In the presence of synchronization, the former variety is then called *local*. For an extensive discussion of these two notions we refer to [BMOZ] and the papers cited there. The interesting point with the notion of global nondeterminacy is that it needs some form of non-LT denotational semantics to make sufficient distinctions. For example, assuming that a, b are normal actions and c is a communication action

(which requires a corresponding $\bar{c}$ in a parallel component to establish sysnchronization), we want to assign different denotational meanings to $s_1 \equiv a;(b+c)$ and $s_2 \equiv (a;b)+(a;c)$. A simple LT model would not capture the operational intuition (which treats s_1, s_2 differently, details follow), since it would deliver the outcome $\{ab, ac\}$ in both cases (cf. the example following definition 3.7).

We shall present below a *branching time* (BT) denotational model for $\mathscr{L}_2$ which indeed provides the desired refinement to distinguish between $\mathscr{D}[\![s_1]\!]$ and $\mathscr{D}[\![s_2]\!]$.

The syntax and operational semantics for $\mathscr{L}_2$ exhibit only minor differences with those for $\mathscr{L}_1$. Firstly, we assume a subset $(c \in) C \subseteq A$ of *communications*, and assume moreover a mapping $\bar{}: C \to C$, such that (writing $\bar{c}$ for $\bar{}(c)$) we have $\bar{\bar{c}} = c$. Finally, we postulate a special element $\tau \in A \setminus C$ which will be used as outcome for a successful synchronization between an action c and its counterpart $\bar{c}$. For this we refer to the rule(s) *Synch* in definition 4.2.

Definition 4.1 (Syntax). Let A be as just described, and $\mathscr{X}$ as in section 3.

a $(s \in \mathscr{L}_2)$. $s ::= a \mid x \mid s_1;s_2 \mid s_1+s_2 \mid s_1 \| s_2$, with $x \in \mathscr{X}$

b $(g \in \mathscr{L}_2{}^g)$. $g ::= a \mid g;s \mid g_1+g_2 \mid g_1 \| g_2$

c $(D \in \mathscr{D}ecl_2)$. $D \equiv \langle x_i \Leftarrow g_i \rangle_i, x_i \in \mathscr{X}, g_i \in \mathscr{L}_2{}^g$.

d $(t \in \mathscr{P}rog_2)$. $t \equiv \langle D \mid s \rangle, D \in \mathscr{D}ecl_2, s \in \mathscr{L}_2$.

e. $\mathscr{L}_2{}' = \mathscr{L}_2 \cup \{E\}$

The transition system T_2 is given in

Definition 4.2. The transition system T_2 contains *Elem*, *SeqComp*, *Rec* and *ParComp* from T_1. Moreover, it contains the rules $(s, s', \tilde{s}, s_1, s_2, s'' \in \mathscr{L}_2, a \in A, c \in C, D \in \mathscr{D}ecl_2)$

$$\frac{s \xrightarrow{a}_D s' \mid E}{\begin{array}{c} s+\tilde{s} \xrightarrow{a}_D s' \mid E \\ \tilde{s}+s \xrightarrow{a}_D s' \mid E \end{array}} \qquad (GloCho)$$

$$\frac{s_1 \xrightarrow{c}_D s', \; s_2 \xrightarrow{\bar{c}}_D s''}{s_1 \| s_2 \xrightarrow{\tau}_D s' \| s''} \qquad (Synch_1)$$

$$\frac{s_1 \xrightarrow{c}_D s', \; s_2 \xrightarrow{\bar{c}}_D E}{s_1 \| s_2 \xrightarrow{\tau}_D s'}, \text{ and a symmetric rule} \qquad (Synch_2)$$

$$\frac{s_1 \xrightarrow{c}_D E, \; s_2 \xrightarrow{\bar{c}}_D E}{s_1 \| s_2 \xrightarrow{\tau}_D E} \qquad (Synch_3)$$

Remark. By *Elem*, we now also have that $c \xrightarrow{c}_D E$.

In order to define $\mathcal{O}$ for $\mathcal{P}rog_2$, we provide a slight variation on the set Q used in section 3. We introduce a new symbol $\delta \notin A$, modelling *failure*, and we put $A_\delta^\infty = A^* \cup A^\omega \cup A^* \cdot \delta$. Thus, A_δ^∞ extends A^∞ by adding all finite sequences over A to which δ is appended. Furthermore, we put $\mathcal{S}_\delta = \mathcal{P}_{nc}(A_\delta^\infty)$, and we take R to be the subset of $\mathcal{S}_\delta$ consisting of either $\{\epsilon\}$ or of $\{\delta\}$, or of elements X in $\mathcal{S}_\delta$ that do not contain ϵ or δ. We shall again use X, Y to range over R, and use the notation X_a as before. (Note, however, that elements in X_a now may end with δ.) We give

Definition 4.3 (operational semantics for $\mathcal{P}rog_2$, $\mathcal{L}_2$).

a. $\mathcal{O} : \mathcal{P}rog_2 \to R$ is given by $\mathcal{O}[\![<D \mid s>]\!] = \mathcal{O}_D[\![s]\!]$.

b. $\mathcal{O}_D : \mathcal{L}_2' \to R$ is given by: $\mathcal{O}_D[\![E]\!] = \{\epsilon\}$, and for $s \neq E$,

$$\mathcal{O}_D[\![s]\!] = \begin{cases} \{\delta\}, \text{ if } \{a \mid s \xrightarrow{a}_D s', a \notin C\} = \emptyset \\ \bigcup \{a \cdot \mathcal{O}_D[\![s']\!] \mid s \xrightarrow{a}_D s', a \notin C\}, \text{ otherwise,} \end{cases}$$

where the transitions are with respect to T_2.

As in section 3, $\mathcal{O}_D$ may be shown to be well-defined by a contractivity argument.

Example. $\mathcal{O}_D[\![a;(b+c)]\!] = \{ab\}$, $\mathcal{O}_D[\![(a;b)+(a;c)]\!] = \{ab, a\delta\}$.

The denotational model for $\mathcal{L}_2$ assumes a domain $(p \in) P$ of branching time *processes* (cf. section 2.4) satisfying the isometry

$$P \cong \{p_0\} \cup \mathcal{P}_{closed}(A \times P) \tag{4.1}$$

Here we assume the discrete metric on A. Typical processes are

- the 'nil process' p_0 and the empty process $\emptyset$ (the empty set), corresponding to the LT objects $\{\epsilon\}$ and $\{\delta\}$, respectively,
- $\{<a, \{<b, p_0>\}>, <a, \{<c, p_0>\}>\}$, which is different from $\{<a, \{<b, p_0>, <c, p_0>\}>\}$,
- the infinite process $p = \lim_n p_n$, where $p_{n+1} = \{<a, p_n>, <b, p_n>\}$.

We recall from section 2.4 that P is a complete ultrametric space, and that elements of P are either finite or satisfy $p = \lim_n p_n$, for p_n finite. We draw attention to the difference between

$\{\delta\} \in R$ and $\emptyset \in P$. There is no problem in incorporating $\emptyset$ into P. In particular, we have that $\mathbf{d}(\{<a,p_1>\},\{<a,p_2>\}) = \frac{1}{2}\cdot\mathbf{d}(p_1,p_2)$ holds, even for p_1 or p_2 equal to $\emptyset$. (This follows from the implicit use of $id_{\frac{1}{2}}(P)$ on the right-hand side of (4.1).) On the other hand, since $a\cdot\emptyset = \emptyset$, including $\emptyset$ into R would invalidate the contractivity property $\mathbf{d}(a\cdot X_1, a\cdot X_2) = \frac{1}{2}\cdot\mathbf{d}(X_1,X_2)$.

We next define the semantic operators $op : P\times P\to P$, for $op \in \{\cup,\circ,\|\}$, as natural variations on those of definition 3.5.

Definition 4.4.

a. $p \cup q = p$, if $q = p_0$,
$p \cup q = q$, if $p = p_0$.
Otherwise, $p \cup q$ equals the set-theoretic union of the sets p and q.

b. Let op stand for $\circ$ or $\|$. Let ϕ be any ndi mapping: $P\times P\to P$. Let $\Phi_{op} : (P\times P\to P)\to(P\times P\to P)$ be defined as follows:

$$\Phi_\circ(\phi)(p)(q) = \begin{cases} q, \text{ if } p = p_0 \\ \{<a,\phi(p')(q)> \mid <a,p'> \in p\}, \text{ otherwise} \end{cases}$$

$$\Phi_\|(\phi)(p)(q) = \Phi_\circ(\phi)(p)(q) \cup \Phi_\circ(\phi)(q)(p) \cup \hat{\phi}_|(p)(q)$$

where $\hat{\phi}_| : P\times P\to P$ is given by

$$\hat{\phi}_|(p)(q) = \{<\tau,\phi(p')(q')> \mid <c,p'> \in p, <\bar{c},q'> \in q\}$$

c. We now put $\circ = \textit{fixed point}(\Phi_\circ)$, $\| = \textit{fixed point}(\Phi_\|)$.

We have again that the operators $\cup,\circ,\|$ are well-defined and ndi (and, hence, continuous).

The denotational definitions are now easy variations on the ones in section 3. Let $(\gamma \in)\,\Gamma_2 = \mathscr{X}\to P$. Now $\mathscr{M} : \mathscr{P}rog_2\to P$ and $\mathscr{D} : \mathscr{L}_2\to(\Gamma_2\to P)$ are defined in

Definition 4.5 (denotational semantics for $\mathscr{L}_2, \mathscr{P}rog_2$).

a. $\mathscr{M}[\![<D \mid s>]\!] = \mathscr{D}[\![s]\!]\gamma_D$.

b. $\gamma_D = \gamma\{p_i/x_i\}_{i=1}^n$, where, for $D \equiv <x_i \Leftarrow g_i>_i$, we put

$$<p_1,\ldots,p_n> = \textit{fixed point } <\Phi_1,\ldots,\Phi_n>$$

with $\Phi_j : P^n \to P$ is given by, for $j = 1,\ldots,n$, $\Phi_j(q_1)\ldots(q_n) = \mathcal{D}[\![g_j]\!]\gamma\{q_i/x_i\}_i$.

c. $\mathcal{D}[\![a]\!]\gamma = \{<a,p_0>\}$, $\mathcal{D}[\![x]\!]\gamma = \gamma(x)$,
$\mathcal{D}[\![s_1 \underline{op}\ s_2]\!]\gamma = \mathcal{D}[\![s_1]\!]\gamma\ op\ \mathcal{D}[\![s_2]\!]\gamma$, for $\underline{op} \in \{;,\cup,\|\}$ and $op \in \{\circ,\cup,\|\}$, respectively.

Examples. $\mathcal{D}[\![a;(b+c)]\!]\gamma = \{<a,\{<b,p_0>,<c,p_0>\}>\}$,
$\mathcal{D}[\![(a;b)+(a;c)]\!]\gamma = \{<a,\{<b,p_0>\}>,<a,\{<c,p_0>\}>\}$,
$\mathcal{M}[\![<x \Leftarrow a;(b\|x) \mid x>]\!] = \lim_i p_i$, where $p_{i+1} = \{<a,\{<b,p_0>\}\|p_i>\}$.

We see that $\mathcal{D}[\![s]\!]\gamma$ contains traces of unsuccessful communications which are not present in $\mathcal{O}_D[\![s]\!]$. For example, $\mathcal{D}[\![c]\!]\gamma = \{<c,p_0>\}$, $\mathcal{O}_D[\![c]\!] = \delta$. Moreover, the elements delivered by $\mathcal{D}[\![s]\!]\gamma$ are branching time objects (in P) and the elements delivered by $\mathcal{O}_D[\![s]\!]$ are linear time objects (in R). We therefore define an *abstraction operator* $abs : P \to R$ which links the two meanings: given an argument p, abs deletes $<c,\ldots>$ branches from p, and collapses the branching time structure into the set of all 'paths' in the process p.

Definition 4.6 (abstraction). We define abs as fixed point of the contracting mapping $\Psi_{abs} : (P \to R) \to (P \to R)$ given as follows: Let $\psi \in P \to R$. Writing $\tilde{\psi}_{abs}$ as shorthand for $\Psi_{abs}(\psi)$, we put

$$\tilde{\psi}_{abs}(p_0) = \{\epsilon\},$$

and, for $p \neq p_0$,

$$\tilde{\psi}_{abs}(p) = \begin{cases} \{\delta\}, \text{ if } \{a \mid <a,p'> \in p,\ a \notin C\} = \emptyset \\ \bigcup\{a \cdot \psi(p') \mid <a,p'> \in p,\ a \notin C\}, \text{ otherwise.} \end{cases}$$

It can now be shown that

Theorem 4.7. *For each* $t \in \mathcal{P}rog_2$, $\mathcal{O}[\![t]\!] = (abs \circ \mathcal{M})[\![t]\!]$.

We omit the proof which is an extension of that of theorem 3.8. Details are given in [KR].

5. Process Creation

We now turn to the study of a simple uniform language with *process*† *creation* as central feature. We couch the notion of process creation in the framework of the language $\mathcal{L}_3$ (with

†The programming notion of 'process' as studied in section 5 has nothing to do with the mathematical notion of 'process' appearing in section 4.

induced $\mathscr{Prog}_3$). This language is like $\mathscr{L}_1$ (or $\mathscr{Prog}_1$), but with the construct of merge *replaced* by the construct $\underline{new}(s)$: execution of $\underline{new}(s)$ creates a new process with body s, to be executed in parallel with the already existing processes. (A more precise definition follows in a moment.)

We first encountered the notion of process creation during our study of the semantics of POOL, a parallel object-oriented language. In [ABKR1,2] we have designed operational and denotational semantics for POOL, and in [AB] we massaged these definitions such that the equivalence of the two semantics for process creation could be shown. What follows below is a new presentation, which could be simplified considerably thanks to another application of a contractivity argument.

We assume A and $\mathscr{X}$ as in section 3. (For simplicity, this section has no $(c \in)\, C \subseteq A$, and '$\cup$' again replaces '$+$'.)

Definition 5.1 (Syntax).

a $(s \in \mathscr{L}_3).\ s ::= a \mid x \mid s_1;s_2 \mid s_1 \cup s_2 \mid \underline{new}(s)$, with $x \in \mathscr{X}$

b $(s \in \mathscr{L}_3{}^g).\ g ::= h \mid g_1;g_2 \mid g_1 \cup g_2 \mid \underline{new}(g)$
$(h \in H).\ h ::= a \mid h;s \mid h_1 \cup h_2$

c $(D \in \mathscr{Decl}_3).\ D \equiv \langle x_i \Leftarrow g_i \rangle_i, x_i \in \mathscr{X}, g_i \in \mathscr{L}_3{}^g, i=1,\ldots,n.$

d $(t \in \mathscr{Prog}_3).\ t \equiv \langle D \mid s \rangle, D \in \mathscr{Decl}_3, s \in \mathscr{L}_3.$

Remark. The complications in the definition of $(g \in)\mathscr{L}_3{}^g$ are caused by the following phenomenon: We want to make sure that occurrences of x in g are guarded by some statement which starts with an elementary action a. Without the precaution as taken in clause b (i.e., adopting a syntax for $\mathscr{L}_3{}^g$, analogous to $\mathscr{L}_1{}^g$, of the form $g ::= a \mid g;s \mid g_1 \cup g_2 \mid \underline{new}(g)$), a statement $\underline{new}(a);x$ would qualify as guarded. As we shall see later, the intended meaning of $\underline{new}(a);x$ is the same as that of the unguarded ($\mathscr{L}_1$-)statement $a \| x$, allowing execution of x before a. This would violate the desired contractivity of the function(s) associated with the declarations; hence, the need for the more involved definition.

Before providing the formal semantic definitions, we first present an informal explanation of process creation. The execution of s is described in terms of a dynamically growing number of processes which execute statements in parallel in the following manner (all steps are with respect to some given D):

1. Set an auxiliary variable i to 1 and set s_1 to s, the statement to be executed. A process, numbered 1, is created to execute s_1.
2. Processes 1 to i execute in parallel. Process j executes s_j $(1 \leq j \leq i)$ in the usual way in case s_j does not begin with some $\underline{new}(s')$ statement.

3. If some process j $(1 \leqq j \leqq i)$ has to execute a statement of the form $\underline{new}(s')$, then the variable i is set to $i+1$, s_i is set to s', and a new process with number i is created to execute s_i. Process j will continue to execute the part after the $\underline{new}(s')$ statement. Go to step 2.
4. Execution terminates if all processes have terminated their execution.

We proceed with the formal semantic definitions. We use a somewhat extended transition formalism which involves constructs defined in

Definition 5.2.

a. The set $(r \in)\mathscr{Seq}$ of *sequents* is defined by $r ::= E \mid s;r$, with $s \in \mathscr{L}_3$.

b. The set $(\varrho \in)\mathscr{Par}$ of *parallel* constructs is defined by $\varrho ::= r_1,\ldots,r_n$, $n \geqq 1$.

Transitions in T_3 are elements of $\mathscr{Par} \times A \times \mathscr{Decl}_3 \times \mathscr{Par}$, written in the notation

$$\varrho \xrightarrow{a}_D \varrho'.$$

We shall often encounter instances of transitions written as $\ldots,r,\ldots \xrightarrow{a}_D \ldots,r',\ldots$. Here r (r') is a component of ϱ (ϱ'), and the notation implies that all terms at the dots (...) are unaffected by the transition. *Mutatis mutandis*, such notation also applies to transition rules.

Definition 5.3 (transition system T_3).

$$\ldots,a;r,\ldots \xrightarrow{a}_D \ldots,r,\ldots \qquad (Elem)$$

$$\frac{\ldots,s_1;(s_2;r),\ldots \xrightarrow{a}_D \varrho}{\ldots,(s_1;s_2);r,\ldots \xrightarrow{a}_D \varrho} \qquad (SeqComp)$$

$$\frac{\ldots,s;r,\ldots \xrightarrow{a}_D \varrho}{\begin{array}{l}\ldots,(s \cup \tilde{s});r,\ldots \xrightarrow{a}_D \varrho \\ \ldots,(\tilde{s} \cup s);r,\ldots \xrightarrow{a}_D \varrho\end{array}} \qquad (Choice)$$

$$\frac{\ldots,g;r,\ldots \xrightarrow{a}_D \varrho}{\ldots,x;r,\ldots \xrightarrow{a}_D \varrho}, \text{ with } x \Leftarrow g \text{ in } D \qquad (Rec)$$

$$\frac{\ldots,r,\ldots,s;E \xrightarrow{a}_D \varrho}{\ldots,\underline{new}(s);r,\ldots \xrightarrow{a}_D \varrho} \qquad (New)$$

Note that in the rule (*New*), if the transition in the consequence has n components on its left-hand side, then the transition in the premise has $n+1$ components on its left-hand side.

Example. $\underline{new}(a;\underline{new}(b;c));d;E \xrightarrow{a}_D d;E\ ,\ \underline{new}(b;c);E \xrightarrow{b}_D d;E\ ,\ c;E \xrightarrow{d}_D E\ ,\ E\ ,\ c;E \xrightarrow{c}_D E\ ,\ E\ ,\ E.$

The operational semantics associated with T_3 is described in

Definition 5.4.

a. $\mathcal{O}: \mathcal{P}rog_3 \to Q$ is given by $\mathcal{O}[\![<D \mid s>]\!] = \mathcal{O}_D[\![s;E]\!]$.

b. $\mathcal{O}_D: \mathcal{P}ar \to Q$ is given by:

$$\mathcal{O}_D[\![E]\!] = \begin{cases} \{\epsilon\}, \text{ if } \varrho = E,E,\ldots,E \\ \bigcup \{a \cdot \mathcal{O}_D[\![\varrho']\!] \mid \varrho \xrightarrow{a}_D \varrho'\}, \text{ otherwise,} \end{cases}$$

where the transitions are with respect to T_3.

Remark. Well-definedness of O_D follows as usual.

We continue with the denotational definitions. Let Q and the operators $\cup, \|$ be as in section 3 ('$\circ$' plays no role here). Besides the usual environments, we also introduce the set of so-called *continuations* $\mathcal{C}ont$ which, in the present setting, coincides with Q. We have, altogether, the following domains and functions:

$(X \in)\ Q,\ (X \in)\ \mathcal{C}ont = Q$

$(\gamma \in)\ \Gamma_3 = \mathcal{X} \to (\mathcal{C}ont \to Q),\ \xi \in \mathcal{C}ont \to Q$

$\mathcal{M}: \mathcal{P}rog_3 \to Q$

$\mathcal{D}: \mathcal{L}_3 \to (\Gamma_3 \to (\mathcal{C}ont \to Q))$

with $\mathcal{M}$ and $\mathcal{D}$ defined in

Definition 5.5.

a. $\mathcal{M}[\![<D \mid s>]\!] = \mathcal{D}[\![s]\!]\gamma_D\{\epsilon\}$.

b. $\gamma_D = \gamma\{\xi_i/x_i\}_{i=1}^n$, where, for $D \equiv <x_i \Leftarrow g_i>_i$, we put

$$<\xi_1,\ldots,\xi_n> = \textit{fixed point } <\Phi_1,\ldots,\Phi_n>$$

with $\Phi_j: (\mathcal{C}ont \to Q)^n \to (\mathcal{C}ont \to Q)$ is given by $\Phi_j(\xi_1')\ldots(\xi_n') = \mathcal{D}[\![g_j]\!]\gamma\{\xi_i'/x_i\}_i$, for $j = 1,\ldots,n$.

c. $\mathcal{D}[\![a]\!]\gamma X = a \cdot X$, $\mathcal{D}[\![x]\!]\gamma X = \gamma(x)X$, $\mathcal{D}[\![s_1;s_2]\!]\gamma X = \mathcal{D}[\![s_1]\!]\gamma(\mathcal{D}[\![s_2]\!]\gamma X)$, $\mathcal{D}[\![s_1 \cup s_2]\!]\gamma X = (\mathcal{D}[\![s_1]\!]\gamma X) \cup (\mathcal{D}[\![s_2]\!]\gamma X)$.

d. $\mathcal{D}[\![\underline{new}(s)]\!]\gamma X = (\mathcal{D}[\![s]\!]\gamma\{\epsilon\}) \| X$.

As usual, our main task is to relate $\mathcal{O}$ and $\mathcal{M}$. We shall prove

Theorem 5.6. *For all* $t \in \mathcal{P}rog_3$, $\mathcal{O}[\![t]\!] = \mathcal{M}[\![t]\!]$.

The proof uses an auxiliary function $\mathcal{E}_D : \mathcal{P}ar \to Q$ defined by

- $\mathcal{E}_D[\![r_1,\ldots,r_n]\!] = \mathcal{E}_D[\![r_1]\!] \| \ldots \| \mathcal{E}_D[\![r_n]\!]$,
- $\mathcal{E}_D[\![E]\!] = \{\epsilon\}$, $\mathcal{E}_D[\![s;r]\!] = \mathcal{D}[\![s]\!]\gamma_D(\mathcal{E}_D[\![r]\!])$.

We shall show that

Claim.

$$\mathcal{E}_D[\![\varrho]\!] = \begin{cases} \{\epsilon\}, \textit{ if } \varrho = E,E,\ldots,E \\ \bigcup \{a \cdot \mathcal{E}_D[\![\varrho']\!] \mid \varrho \xrightarrow{a}_D \varrho'\}, \textit{ otherwise.} \end{cases}$$

Once this claim has been established, we are done: By the usual argument, it implies that $\mathcal{E}_D[\![\varrho]\!] = \mathcal{O}_D[\![\varrho]\!]$; hence, in particular,
$\mathcal{O}[\![<D \mid s>]\!] = \mathcal{O}_D[\![s;E]\!] = \mathcal{E}_D[\![s;E]\!] = \mathcal{D}[\![s]\!]\gamma_D\{\epsilon\} = \mathcal{M}[\![<D \mid s>]\!]$.
The claim is proved by showing that $\mathcal{E}_D$ satisfies (*): $\Psi(\mathcal{E}_D) = \mathcal{E}_D$, where Ψ is defined, for each $\mathcal{F}_D \in \mathcal{P}ar \to Q$, by

$$\Psi(\mathcal{F}_D)(\varrho) = \begin{cases} \{\epsilon\}, \text{ if } \varrho = E,\ldots,E \\ \bigcup \{a \cdot \mathcal{F}_D[\![\varrho']\!] \mid \varrho \xrightarrow{a}_D \varrho'\}, \text{ otherwise.} \end{cases}$$

We prove (*) by induction on the complexity of $\varrho = r_1,\ldots,r_m$, which we define as the entity $<k,c(\varrho)>$, where $k \geqq 0$ is the number of unguarded occurrences of some x_j $(1 \leqq j \leqq n)$ in some r_i $(1 \leqq i \leqq m)$. Moreover, $c(\varrho)$ is defined as $c(r_1)+\ldots+c(r_m)$, where $c(E)=0$, $c(s;r)=c(s)+c(r)$, and $c(a)=c(x)=1$, $c(s_1;s_2)=c(s_1 \cup s_2)=1+c(s_1)+c(s_2)$, $c(\underline{new}(s))=1+c(s)$. (We recall here that x does occur unguarded in, e.g., $\underline{new}(a);x;E$.) We order the entities $<k,c>$ by putting $<k,c><<k',c'>$ whenever $k<k'$ or $k=k'$ and $c<c'$.

Stage 1. We first consider the case that complexity(ϱ) = $<0,\ldots>$. If $\varrho = r,\varrho'$ we show that $\Phi(\mathcal{E}_D)(r,\varrho') = \mathcal{E}_D[\![r,\varrho']\!]$ by an argument similar to that in section 3, stage 1 of the proof of

theorem 3.8. Here we use, in addition, that, if

$$\frac{\varrho_1 \xrightarrow{a}_D \tilde{\varrho}}{\varrho_2 \xrightarrow{a}_D \tilde{\varrho}}$$

then complexity(ϱ_1) < complexity(ϱ_2). If $\varrho = r$, we distinguish various subcases. If $r = E$, the claim is obvious. If $r = s;r'$, we argue by case analysis on the structure of s. We discuss two typical subcases:

- $s \equiv s_1;s_2$. Then

  $\Phi(\mathscr{E}_D)((s_1;s_2);r) =$

  $\bigcup \{a \cdot \mathscr{E}_D[\![\tilde{\varrho}]\!] \mid (s_1;s_2);r \xrightarrow{a}_D \tilde{\varrho}\} =$ (def. T_3)

  $\bigcup \{a \cdot \mathscr{E}_D[\![\tilde{\varrho}]\!] \mid s_1;(s_2;r) \xrightarrow{a}_D \tilde{\varrho}\} =$

  (since $c(s_1;(s_2;r)) < c((s_1;s_2);r)$, we may apply the ind. hyp.)

  $\mathscr{E}_D[\![s_1;(s_2;r)]\!] =$ (def. $\mathscr{E}_D$, $\mathscr{D}$)

  $\mathscr{E}_D[\![(s_1;s_2);r]\!]$.

- $s \equiv \underline{new}(s')$.

  $\Phi(\mathscr{E}_D)(\underline{new}(s);r) =$

  $\bigcup \{a \cdot \mathscr{E}_D[\![\tilde{\varrho}]\!] \mid (\underline{new}(s);r \xrightarrow{a}_D \tilde{\varrho}\} =$ (def. T_3)

  $\bigcup \{a \cdot \mathscr{E}_D[\![\tilde{\varrho}]\!] \mid r,s;E \xrightarrow{a}_D \tilde{\varrho}\} =$

  (since $c(r,s;E) < c(\underline{new}(s);r)$, we may apply the ind. hyp.)

  $\mathscr{E}_D[\![r,s;E]\!] =$ (def. $\mathscr{E}_D$)

  $\mathscr{E}_D[\![r]\!] \| \mathscr{E}_D[\![s;E]\!] =$ (def.$\mathscr{E}_D$, $\mathscr{D}$)

  $(\mathscr{D}[\![s]\!]\gamma_D\{\epsilon\}) \| \mathscr{E}_D[\![r]\!] =$

  $\mathscr{D}[\![\underline{new}(s)]\!]\gamma_D(\mathscr{E}_D[\![r]\!]) =$

  $\mathscr{E}_D[\![\underline{new}(s);r]\!]$.

Stage k+1. Assume that (*) holds for any ϱ with at most k unguarded occurrences of some x_i. Now consider a ϱ with $k+1$ unguarded occurrences. All cases are as before, but for the case $\varrho = r, r = s;r', s = x_i$, for some $x_i \in \mathscr{X}$. Then

$\Phi(\mathscr{E}_D)(x_i;r') =$

$\bigcup \{a \cdot \mathscr{E}_D[\![\tilde{\varrho}]\!] \mid x_i;r' \xrightarrow{a}_D \tilde{\varrho}\} =$ (def. T_3)

$\bigcup \{a \cdot \mathscr{E}_D[\![\tilde{\varrho}]\!] \mid g_i;r' \xrightarrow{a}_D \tilde{\varrho}\}$ (with $x_i \Leftarrow g_i$ in D) =

(since g_i is guarded, we may apply stage k)

$\mathscr{E}_D[\![g_i; r']\!] =$

$\mathscr{E}_D[\![x_i; r']\!]$, where the last equality holds by the definition of $\mathscr{D}$. □

We conclude this section with two

Remarks.

1. It has been shown by IJ.J. Aalbersberg and P. America that the expressive power of || and of *new*(...) are incomparable: There exists $t_1 \in \mathscr{P}rog_1$ such that for no $t_3 \in \mathscr{P}rog_3$, $\mathscr{O}[\![t_1]\!] = \mathscr{O}[\![t_3]\!]$, and vice versa.
2. In [AB], process creation is also considered in a nonuniform setting, in the sense of, e.g., the language of the next section.

6. Communication with Value Passing

We conclude our list of four specimen languages analyzed with metric tools with a discussion of a *nonuniform* language $\mathscr{L}_4$ which is best seen as an extension of $\mathscr{L}_2$ from section 4. The atomic actions of $\mathscr{L}_4$ are no longer uninterpreted symbols a from some alphabet A, but, instead, assignments $v:=e$, for v an individual variable and e an expression, and communication actions $c?v$ or $c!e$. Also, booleans b are introduced appearing as tests in conditional statements. Accordingly, the semantic models now incorporate *states*, i.e. mappings from individual variables v to elements α in some set V of *values*.

We first collect some syntactic preparations. We introduce the set $(v \in)\mathscr{I}nd_v$ of individual variables and $(c \in)C$ of *channels*. Channel names c appear in the communication actions $c?v$ and $c!e$. Synchronization of two such actions is defined similarly to that of $c, \bar{c}$ in section 4. In addition, however, at the moment of successful synchronization the assignment $v:=e$ takes place. Assuming that $c?v$ occurs in some component s_1, and $c!e$ in a component s_2 of the parallel statement $s_1 \| s_2$, the current value of e is transmitted by the sender s_2 over the channel c to the receiver s_1, where it is (instantaneously) assigned to the variable v. Furthermore, we introduce the syntactic classes $(e \in)\mathscr{E}xp$ of expressions and $(b \in)\mathscr{B}ool$ of booleans. For simplicity, we assume some elementary syntax for $\mathscr{E}xp$ and $\mathscr{B}ool$, and leave this unspecified here. We only postulate that no complications such as side-effects or nontermination arise in the evaluation of some e or b.

We now give

Definition 6.1. Let $\mathscr{X} = \{x_1, \ldots, x_n\}$ be as before.

a $(s \in \mathscr{L}_4)$.

$s ::= v:=e \mid c?v \mid c!e \mid x \mid \underline{if}\ b\ \underline{then}\ s_1\ \underline{else}\ s_2\ \underline{fi} \mid s_1;s_2 \mid s_1 \| s_2$, with $x \in \mathscr{X}$

b $(g \in \mathscr{L}_4^g)$. $g ::= v:=e \mid c?v \mid c!e \mid g;s \mid \underline{if}\ b\ \underline{then}\ g_1\ \underline{else}\ g_2\ \underline{fi} \mid g_1 \| g_2$

c. $D \in \mathscr{Decl}_4$, $t \in \mathscr{Prog}_4$ are formed from $s \in \mathscr{L}_4$ and $g \in \mathscr{L}_4^g$ as usual.

Remark. For simplicity, $\mathscr{L}_4$ has no form of nondeterminism.

Some semantic preparations are contained in

Definition 6.2.

a. $(\alpha \in)V$ is the set of *values*, $\{tt, ff\}$ is the set of *truth-values*.

b. $(\sigma \in)\Sigma = \mathscr{Ind}_{\mathscr{V}} \to V$ is the set of *states*.

c. $(\eta \in)\mathsf{H} = \Sigma \cup \Delta$, where
$(\delta \in)\Delta = \{c?v \mid c \in C, v \in \mathscr{Ind}_{\mathscr{V}}\} \cup \{c!\alpha \mid c \in C, \alpha \in V\}$.

d. For $e \in \mathscr{Exp}$, $[\![e]\!](\sigma)$ denotes its value in state σ; for $b \in \mathscr{Bool}$, $[\![b]\!](\sigma)$ denotes its truth-value in state σ.

Remarks.

1. The reader may always take $\mathbb{Z}$ for V to give some realistic flavour to our considerations.
2. The set H serves technical purposes in the definitions below. For given input σ, computations yield elements $\eta \in \mathsf{H}$ as output. These may be distinguished into 'normal' $\eta \in \Sigma$ and 'abnormal' $\eta \in \Delta$, where the latter results from one-sided (and therefore failing) attempts at synchronization $c?v$ or $c!e$.

We proceed with the definition of the transition system T_4. This time, transitions are five-tuples in $\mathscr{L}_4 \times \Sigma \times \mathscr{Decl}_4 \times \mathscr{L}_4 \times \mathsf{H}$ or four-tuples $\mathscr{L}_4 \times \Sigma \times \mathscr{Decl}_4 \times \mathsf{H}$, written as

$$\langle s, \sigma \rangle \to_D \langle s', \eta \rangle,$$

$$\langle s, \sigma \rangle \to_D \eta,$$

respectively. T_4 is defined, again applying a self-explanatory style of abbreviating rules, in

Definition 6.3.

$$\langle v:=e, \sigma \rangle \to_D \sigma\{\alpha/v\}, \text{ where } \alpha = [\![e]\!](\sigma) \qquad (Ass)$$

$$\langle c?v, \sigma \rangle \to_D c?v \quad \langle c!e, \sigma \rangle \to_D c!\alpha, \text{ where } \alpha = [\![e]\!](\sigma) \qquad (IndCom)$$

$$\frac{\langle s_i, \sigma \rangle \to_D \langle s', \eta \rangle \mid \eta}{\langle \underline{if}\ b\ \underline{then}\ s_1\ \underline{else}\ s_2\ \underline{fi}, \sigma \rangle \to_D \langle s', \eta \rangle \mid \eta} \qquad (Cond)$$

where $s_i = s_1$ (s_2) in case $[\![b]\!](\sigma) = tt$ (ff)

$$\frac{<s,\sigma> \rightarrow_D <s',\eta> \mid \eta}{<s;\tilde{s},\sigma> \rightarrow_D <s';\tilde{s},\eta> \mid <\tilde{s},\eta>} \qquad (SeqComp)$$

$$\frac{<s,\sigma> \rightarrow_D <s',\eta> \mid \eta}{\begin{array}{l}<s \| \tilde{s},\sigma> \rightarrow_D <s' \| \tilde{s},\eta> \mid <\tilde{s},\eta> \\ <\tilde{s} \| s,\sigma> \rightarrow_D <\tilde{s} \| s',\eta> \mid <\tilde{s},\eta>\end{array}} \qquad (ParComp)$$

$$\frac{<s_1,\sigma> \rightarrow_D <s',c?v>,\ <s_2,\sigma> \rightarrow_D <s'',c!\alpha>}{<s_1 \| s_2,\sigma> \rightarrow_D <s' \| s'',\sigma\{\alpha/v\}>} \qquad (Synch)$$

and the three obvious variations in case s', s'' or both are missing

$$\frac{<g,\sigma> \rightarrow_D <s',\eta> \mid \eta}{<x,\sigma> \rightarrow_D <s',\eta> \mid \eta}\ \text{, with } x \Leftarrow g \text{ in } D \qquad (Rec)$$

Before we define $\mathcal{O}[\![t]\!]$ and $\mathcal{O}_D[\![s]\!]$ we first introduce the process domain P as solution of

$$P \cong \{p_0\} \cup (\Sigma \rightarrow \mathcal{P}_{closed}(H \times P)), \qquad (6.1)$$

with the discrete metric on Σ and H.

Remark. We leave for another occasion discussion of the equation

$$P' \cong \{\epsilon\} \cup (\Sigma \rightarrow \mathcal{P}_{closed}(H \cdot P')) \qquad (6.2)$$

determining P' as possible 'linear time' alternative for P. This discussion will in particular have to clarify the role of '$\cdot$' versus '$\times$' in a nonuniform context.

The operational semantics are given in

Definition 6.4 (operational semantics for $\mathcal{P}rog_4, \mathcal{L}_4$).

a. $\mathcal{O} : \mathcal{P}rog_4 \rightarrow P$ is given by $\mathcal{O}[\![<D \mid s>]\!] = \mathcal{O}_D[\![s]\!]$.

b. $\mathcal{O}_D : \mathcal{L}_4 \rightarrow P$ is given by:

$$\mathcal{O}_D[\![s]\!] = \lambda\sigma.(\{<\sigma', \mathcal{O}_D[\![s']\!]> \mid <s,\sigma> \rightarrow_D <s',\sigma'>\} \cup \{<\sigma',p_0> \mid <s,\sigma> \rightarrow_D \sigma'\});$$

where the transitions are with respect to T_4.

Remark. Just as in definition 4.3, $\mathcal{O}_D$ does not take into account transitions stemming from failing communications, signalled here by the format $<s,\sigma> \rightarrow_D <s',\delta> \mid \delta$ with $\delta \in \Delta$.

For P as in (6.1), we can define the usual operators $\cup$, $\circ$, $\|$. We restrict ourselves to the definition of $\|$, here involving the auxiliary operators $\mathbb{L}$ and $|$.

Definition 6.5. Let P be as in (6.1), and let $(X \in)\mathscr{P}$ abbreviate $\mathscr{P}_{closed}(H \times P)$. We define the operator $\|$ as fixed point of $\Phi_{\|} : (P \times P \to P) \to (P \times P \to P)$, where, for $\phi \in P \times P \to P$ and ϕ ndi, $\Phi_{\|}(\phi) =^{df.} \tilde{\phi}_{\|}$ is given by

$$\tilde{\phi}_{\|}(p)(q) = \begin{cases} p, \text{ if } q = p_0 \\ q, \text{ if } p = p_0 \\ \lambda\sigma.\,(\hat{\phi}_{\mathbb{L}}(p(\sigma))(q) \cup \hat{\phi}_{\mathbb{L}}(q(\sigma))(p) \cup \hat{\phi}_{|,\sigma}(p(\sigma))(q(\sigma))) \end{cases}$$

where $\hat{\phi}_{\mathbb{L}} : \mathscr{P} \times P \to \mathscr{P}$ is defined by

$$\hat{\phi}_{\mathbb{L}}(X)(q) = \{<\eta, \phi(p')(q)> \mid <\eta, p'> \in X\},$$

and $\hat{\phi}_{|,\sigma} : \mathscr{P} \times \mathscr{P} \to \mathscr{P}$ is defined by

$$\hat{\phi}_{|,\sigma}(X)(Y) = \{<\sigma\{\alpha/v\}, \phi(p')(q')> \mid <c?v, p'> \in X,\ <c!\alpha, q'> \in Y \text{ or vice versa}\}.$$

We are now ready for the definition of $\mathscr{M}[\![t]\!]$ and $\mathscr{D}[\![t]\!]$. Let $(\gamma \in)\Gamma_4 = \mathscr{X} \to P$, let $\mathscr{M} : \mathscr{P}rog_4 \to P$ and $\mathscr{D} : \mathscr{L}_4 \to (\Gamma_4 \to P)$. We give

Definition 6.6 (denotational semantics for $\mathscr{L}_4, \mathscr{P}rog_4$).

a. $\mathscr{M}[\![<D \mid s>]\!] = \mathscr{D}[\![s]\!]\gamma_D$.

b. γ_D is as usual.

c. $\mathscr{D}[\![v := e]\!]\gamma = \lambda\sigma.\,\{<\sigma\{\alpha/v\}, p_0>\}$, with $\alpha = [\![e]\!](\sigma)$, $\mathscr{D}[\![c?v]\!]\gamma = \lambda\sigma.\,\{<c?v, p_0>\}$, $\mathscr{D}[\![c!e]\!]\gamma = \lambda\sigma.\,\{<c!\alpha, p_0>\}$, with $\alpha = [\![e]\!](\sigma)$, $\mathscr{D}[\![\underline{if}\ b\ \underline{then}\ s_1\ \underline{else}\ s_2\ \underline{fi}]\!] = \lambda\sigma.\,\mathit{if}\ [\![b]\!](\sigma) = tt\ \mathit{then}\ \mathscr{D}[\![s_1]\!]\gamma\sigma\ \mathit{else}\ \mathscr{D}[\![s_2]\!]\gamma\sigma\ \mathit{fi}$, and $\mathscr{D}[\![x]\!]\gamma$, $\mathscr{D}[\![s_1\ \underline{op}\ s_2]\!]\gamma$ for $\underline{op} \in \{;, \cup, \|\}$, as usual.

One last step is necessary before we can formulate our final result. We define the abstraction mapping $abs : P \to P''$, where P'' satisfies

$$P'' \cong \{p_0\} \cup (\Sigma \to \mathscr{P}_{closed}(\Sigma \times P'')) \tag{6.3}$$

by putting $abs = \mathit{fixed\ point}(\Psi_{abs})$, with $\Psi_{abs} : (P \to P'') \to (P \to P'')$ defined by: For $\psi \in P \to P''$, $\Psi_{abs}(\psi) =^{df.} \tilde{\psi}_{abs}$ is given by

$$\tilde{\psi}_{abs}(p_0) = p_0,$$

and, for $p \neq p_0$,

$$\tilde{\psi}_{abs}(p) = \lambda\sigma.\,\hat{\psi}(p(\sigma))$$

and

$$\hat{\psi}(X) = \{<\sigma,\psi(p')> \mid <\sigma,p'> \in X\}.$$

Note that the last clause deletes pairs $<\delta,p'>$ from X.

We finally have:

Theorem 6.7. *For each* $t \in \mathscr{P}rog_4$, $\mathscr{O}[\![t]\!] = (abs \circ \mathscr{M})[\![t]\!]$.

Proof. By the usual contractivity argument. □

References

[AB] P. America, J.W. de Bakker, Designing equivalent semantic models for process creation, Report CS-R8732, Centre for Mathematics and Computer Science, Amsterdam, 1987. Also in Proc. Advanced School on Mathematical Models for the Semantics of Parallelism (M. Venturini Zilli, ed.), LNCS 280, Springer, 1987.

[ABKR1] P. 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.

[ABKR2] P. America, J.W. de Bakker, J.N. Kok, J.J.M.M. Rutten, Denotational semantics of a parallel object-oriented language, Report CS-R8626, Centre for Mathematics and Computer Science, Amsterdam, 1986. To appear in Information and Computation.

[AR] P. America, J.J.M.M. Rutten: Solving reflexive domain equations in a category of complete metric spaces, Report CS-R8709, Centre for Mathematics and Computer Science, Amsterdam, the Netherlands, 1987. To appear in Proc. of the Third Workshop on Mathematical Foundations of Programming Language Semantics.

[AP] K. Apt & G. Plotkin, Countable Nondeterminism and Random Assignment, JACM 33(4) (1986) 724-767.

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

[BKMOZ] 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.

[BM] J.W. de Bakker, J.-J.Ch. Meyer, Order and metric in the stream semantics of elemental concurrency, Acta Informatica 24 (1987) 491-511.

[BMO] J.W. de Bakker, J.-J.Ch. Meyer, E.-R. Olderog, Infinite streams and finite observations in the semantics of uniform concurrency, TCS 49 (1987) 87-112.

[BMOZ] 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, Report CS-R8601, Centre for Mathematics and Computer Science, Amsterdam, 1986. To appear in Journal of Comp. Syst. Sci.

[BZ1] J.W. de Bakker, J.I. Zucker, Processes and the denotational semantics of concurrency, Inform. and Control 54 (1982) 70-120.

[BZ2] J.W. de Bakker, J.I. Zucker, Processes and a fair semantics for the ADA rendezvous, in: Proc. 10th ICALP (J. Diaz ed.), LNCS 154, Springer (1983) 52-66.

[BK] J.A. Bergstra, J.W. Klop, A convergence theorem in process algebra, Report CS-R8733, Centre for Mathematics and Computer Science, Amsterdam, 1987.

[Du] J. Dugundji, Topology, Allen and Bacon, Rockleigh, N.J. 1966.

[En] R. Engelking, General topology, Polish Scientific Publishers 1977.

[HP] M. Hennessy, G.D. Plotkin, Full abstraction for a simple parallel programming language, in: Proceedings 8th MFCS (J. Becvar ed.), LNCS 74 Springer (1979) 108-120.

[Ho] C.A.R. Hoare, Communicating sequential processes, Prentice-Hall Int., Englewood

Cliffs, New Jersey, 1985.

[KR] J.N. Kok, J.J.M.M. Rutten, Contractions in comparing concurrency semantics, Report CS-R8755, Centre for Mathematics and Computer Science, Amsterdam, 1987.

[M] J.-J.Ch. Meyer, Merging regular processes by means of fixed point theory, TCS 45 (1986) 193-260.

[MO] J.-J.Ch. Meyer, E.-R. Olderog, Hiding in stream semantics of uniform concurrency, Report IR-125, Free University, Amsterdam, 1987.

[MV] J.-J.Ch. Meyer, E.P. de Vink, Applications of compactness in the Smyth powerdomain of streams, in: Proc. TAPSOFT '87 (H. Ehrig, R. Kowalski, G. Levi, U. Montanari, eds.), LNCS 249, Springer (1987) 241-255.

[Mi] R. Milner, A calculus for communicating systems, LNCS 92, Springer, 1980,

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

[Pl1] G.D. Plotkin, A powerdomain construction, SIAM Journal of Computing, Vol.5, No 3 (1976) 452-487.

[Pl2] G.D. Plotkin, A structural approach to operational semantics, Report DAIMI FN-19, Comp.Sci.Dept., Aarhus Univ., 1981.

[Pl3] G.D. Plotkin, An operational semantics for CSP, in: D. Bjørner (ed.): Formal description of programming concepts II, North-Holland (1983) 199-223.