5

Concurrency semantics based on metric domain equations

J. W. DE BAKKER AND J. J. M. M. RUTTEN

Abstract

We show how domain equations may be solved in the category of complete metric spaces. For five example languages we demonstrate how to exploit domain equations in the design of their operational and denotational semantics. Two languages are schematic or uniform. Three have interpreted elementary actions involving individual variables and inducing state transformations. For the latter group we discuss three denotational models reflecting a variety of language notions considered. A central theme is the distinction, within the non-uniform setting, of linear time versus branching time models. Throughout, fruitful use is made of the technique of obtaining semantic mappings, operators, etc. as fixed points of higher-order functions. A brief discussion of the relationship between bisimulation and one of the domains considered concludes the paper.

5.1 Introduction

Concurrency semantics is concerned with the mathematical modelling of parallel behaviour. A parallel computation induces some form of simultaneous or interleaved execution of the elementary actions from the constituent (parallel) components. Accordingly, it is to be expected that the mathematical description of such a computation involves a detailed modelling of its intermediate steps — rather than just its input–output behaviour, as is mostly sufficient in a sequential setting. The collection of intermediate steps may be said to constitute the *history* of the computation. Two histories p_1, p_2 are close together if their first difference is exhibited only after many steps. This observation is at the basis of the metric approach to concurrency semantics. We introduce distances d such that

$$d(p_1,p_2) \;=\; 2^{-n} \tag{1}$$

where $n = \sup\{k \mid p_1[k] = p_2[k]\}$, with $p[k]$ a truncation of p after k steps. It is our aim in this chapter to make this idea precise, and to illustrate how it may be exploited in the design of semantic models for a variety of concurrency phenomena.

Section 5.2 introduces a rigorous setting for the metric space techniques to be applied subsequently. The category $\mathcal{C}$ of *complete metric spaces* is introduced, and it is shown how metric spaces (P, d), or P for short, can be specified as solutions of *domain equations* $P = F(P)$ for a variety of functors $F : \mathcal{C} \to \mathcal{C}$. In the formation of these F, several composition operators such as $\times$ (cartesian product), $\cup$ (disjoint union), $\to$ (function space), $\mathbb{P}$ (powerset of), etc. are used. The main result of this section is the following. Provided a rather natural condition is satisfied for the recursive occurrences of P in the expression $F(P)$ (which condition ensures a kind of *contractivity* of F in P), the equation $P = F(P)$ can be solved and its solution is unique. The first application of metric spaces in order to obtain domains as solutions of such equations was described in (de Bakker and Zucker 1982), a paper in turn inspired by Nivat's general metric approach to semantics (for example, (Nivat 1979)). The ideas of (de Bakker and Zucker 1982) were generalized (to cover equations of the form $P = \cdots(P \to F_1(P))\cdots$ also, a case missing in (de Bakker and Zucker 1982)) and put in a category-theoretic framework in (America and Rutten 1989a). Since the latter reference provides full mathematical details, including complete proofs, we restrict the treatment in Section 5.2 to a more concise one, not repeating these proofs, but with sufficient information to make the present chapter self-contained. Independently of (America and Rutten 1989a), the question of how to extend the ideas of (de Bakker and Zucker 1982) was also investigated by Majster-Cederbaum (1988, 1989, 199?); in these references the issues of the existence and uniqueness of solutions of the equation $P = F(P)$ are also investigated in a category-theoretic framework.

Section 5.3 constitutes the main body of our chapter. For five example languages L_i, $i = 0, \ldots, 4$, we introduce operational ($\mathcal{O}_i$) and denotational ($\mathcal{D}_i$) semantic models, where O_i is a mapping $L_i \to R_i$, and D_i a mapping $L_i \to P_i$ (here we neglect one refinement to be discussed later), $i = 0, \ldots, 4$. Determined by the range of programming concepts in the language L_i, we shall design a corresponding range of operational domains R_i and denotational domains P_i, $i = 0, \ldots, 4$, each time as the solution of a (pair of) domain equation(s) geared to the construction of an appropriate model capturing the notions concerned. Of the languages L_0 to L_4, two are what we like to call *uniform* (the elementary actions are just symbols) (de Bakker *et al.* 1986, 1987, 1988). The other three are *non-uniform*: the elementary actions refer to individual variables, and we encounter states, assignments, etc. The models for L_2 to L_4 mention states and state transformations, or, put in mathematical terms, the corresponding functor F

now has occurrences of the function space constructor. There are somewhat subtle (and not yet fully understood) differences between P_2, P_3, and P_4. Using a terminology mostly reserved for the uniform case, that is, the contrast between *linear time* (models with sets of sequences) versus *branching time* (models with trees or tree-like entities) (de Bakker *et al.* 1984), we might say that the domains P_2 and P_3 are (non-uniform and) linear time, whereas P_4 is (non-uniform and) branching time. Understanding the difference between P_2 and P_3 requires further study. The introduction and associated analysis of P_2 to P_4 appears here for the first time. In earlier work, we always used P_4 (or trivial variants), and for some time we did not see how to design a satisfactory non-uniform model with the linear time flavour. The domain P_2 was then proposed as a candidate to enable us to design a *fully abstract* $\mathcal{D}_2$ (with respect to the $\mathcal{O}_2$ to be given in Section 5.3). In the meantime it has been shown by Horita *et al.* (1990) that a certain extension P_2' of P_2 (P_2' ignores details present in P_2) indeed allows us to define a fully abstract denotational $\mathcal{D}_2'$ (with respect to $\mathcal{O}_2$ as to be given). For L_3, we do not know whether a similar result holds. For L_4, we do know that $\mathcal{D}_4$ is not fully abstract with respect to $\mathcal{O}_4$.

In general, the material in Section 5.3 is organized in such a way that it brings out the unifying effect of the metric approach. At least the following definitions and proof techniques all follow the same pattern (for $i = 0, \ldots, 4$):

- introduction of the transition system T_i (as in Plotkin's structured operational semantics) and the definition of the associated $\mathcal{O}_i$ as the fixed point of a contracting Ψ_i;
- introduction of the domains R_i, P_i, and definition of the various semantic operators (such as $\circ$, $\|$), for the P_i setting, in terms of fixed points of contracting $\Omega_\circ$, $\Omega_\|$;
- introducing the denotational semantics $\mathcal{D}_i$ as the fixed point of a contracting Φ_i;
- relating $\mathcal{O}_i$ and $\mathcal{D}_i$ through abstraction mappings abs_i, themselves obtained as fixed points of contracting Δ_i;
- establishing that $\mathcal{O}_i = abs_i \circ \mathcal{D}_i$, by introducing an intermediate semantics $\mathcal{I}_i : L_i \to P_i$ (with denotational codomain P_i, but obtained from the transition system T_i), deriving that $\mathcal{I}_i = \mathcal{D}_i$ (as in (Kok and Rutten 1988, de Bakker and Meyer 1988) and then proving that $abs_i \circ \mathcal{I}_i = \mathcal{O}_i$, once more by a fixed point argument.

In case the reader is not satisfied by the elementary character of L_0 to L_4, we emphasize that these languages have been selected for didactic reasons. Elsewhere we have demonstrated how the metric techniques described in the present chapter may be exploited in the treatment of substantially more complicated language notions. For the case of object-oriented programming languages, we refer to (America *et al.* 1989, America and de Bakker 1988, America and Rutten 1989b, Rutten 1990a); for a treatment of parallel logic programming semantics, we mention (de Bakker 1988, de Bakker and Kok 1988, 1990). Earlier introductory or overview presentations of metric concurrency semantics were given in (de Bakker and Meyer 1988, de Bakker 1989).

The last section of the chapter is devoted to a slightly more special topic. It is well known that the notion of *bisimulation* (Park 1981) is a central tool in concurrency semantics, and the question arises whether it may be related to results about domains in the style of P_0 to P_4. For a simple case (P_0 only), we prove the following theorem. Let s_1, s_2 be two states (here used as abstractions of the statements as introduced in Section 5.3) from a set S. We have that s_1 is bisimilar to s_2 (with respect to a given labelled transition system T) if and only if $\mathcal{M}[\![s_1]\!] = \mathcal{M}[\![s_2]\!]$, where $\mathcal{M} : S \rightarrow P_0$ is obtained from T in a manner which is the same as the way in which $\mathcal{I}$ (from Section 5.3) is obtained from T_0. Let us also draw attention to the fact that this result depends critically on the branching structure for P_0.

We conclude this introduction with two remarks about possible extensions of the reported results. In (Rutten 1989), a beginning has been made with the exploration of a technique which 'automatically' infers a denotational semantics $\mathcal{D}$ from a given transition system T (of course obeying the compositionality requirement on $\mathcal{D}$). A bonus of this automatic inference is, in particular, the possibility of avoiding *ad hoc* equivalence proofs for $\mathcal{O} = abs \circ \mathcal{D}$. A second important topic which we want to address in future work is the design of a fully abstract model for a language with process creation.

5.2 Metric spaces and domain equations

As mathematical domains for our operational and denotational semantics we shall use complete metric spaces satisfying a so-called *reflexive domain equation* of the following form:

$$P \cong F(P)$$

(The symbol $\cong$ should be read 'is isometric to' and is defined below.) Here $F(P)$ is an expression built from P and a number of standard constructions on metric spaces (also to be formally introduced shortly). A few examples are

$$P \cong A \cup (B \times P) \tag{2}$$
$$P \cong A \cup \mathbb{P}_{co}(B \times P) \tag{3}$$
$$P \cong A \cup (B \to P) \tag{4}$$

where A and B are given fixed complete metric spaces. De Bakker and Zucker (1982) have first described how to solve these equations in a metric setting. Roughly, their approach amounts to the following. In order to solve $P \cong F(P)$ they define a sequence of complete metric spaces $(P_n)_n$ by $P_0 = A$ and $P_{n+1} = F(P_n)$, for $n > 0$, such that $P_0 \subseteq P_1 \subseteq \cdots$. Then they take the *metric completion* of the union of these spaces P_n, say $\bar{P}$, and show $\bar{P} \cong F(\bar{P})$. In this way they are able to solve equations (2), (3) and (4).

There is one type of equation for which this approach does not work, namely

$$P \cong A \cup (P \to^1 G(P)) \tag{5}$$

in which P occurs at the *left* side of a function space arrow and $G(P)$ is an expression possibly containing P. This is due to the fact that it is not always the case that $P_n \subseteq F(P_n)$.

In (America and Rutten 1989a) the above approach is generalized in order to overcome this problem. The family of complete metric spaces is made into a *category* $\mathcal{C}$ by providing some additional structure. (For an extensive introduction to category theory we refer the reader to (Mac Lane 1971).) Then the expression F is interpreted as a *functor* $F : \mathcal{C} \to \mathcal{C}$ which is (in a sense) *contracting*. It is proved that a generalized version of Banach's theorem (see below) holds, that is, that contracting functors have a fixed point (up to isometry). Such a fixed point, satisfying $P \cong F(P)$, is a solution of the domain equation.

We shall now give a quick overview of these results, omitting many details and all proofs. For a full treatment we refer the reader to (America and Rutten 1989a). We start by listing the basic definitions and facts of metric topology that we shall need.

We assume the following notions to be known (the reader might consult (Dugundji 1966) or (Enkelking 1977)): metric space, ultra-metric space, complete (ultra-) metric space, continuous function, closed set, compact set. (In our definition the distance between two elements of a metric space is always bounded by 1.)

An arbitrary set A can be supplied with a metric d_A, called the *discrete* metric, defined by

$$d_A(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$$

Now (A, d_A) is a metric, even an ultra-metric, space.

Let (M_1, d_1) and (M_2, d_2) be two complete metric spaces. A function $f : M_1 \to M_2$ is called *non-expansive* if for all $x, y \in M_1$

$$d_2(f(x), f(y)) \leqslant d_1(x, y)$$

The set of all non-expansive functions from M_1 to M_2 is denoted by $M_1 \to^1 M_2$. A function $f : M_1 \to M_2$ is called *contracting* (or a *contraction*) if there exists $\epsilon \in [0, 1)$ such that for all $x, y \in M_1$

$$d_2(f(x), f(y)) \leqslant \epsilon \cdot d_1(x, y)$$

(Non-expansive functions and contractions are continuous.)

The following fact is known as Banach's theorem. Let (M, d) be a complete metric space and $f : M \to M$ a contraction. Then f has a unique fixed point, that is, there exists a unique solution $x \in M$ such that $f(x) = x$.

We call M_1 and M_2 *isometric* (notation: $M_1 \cong M_2$) if there exists a bijective mapping $f : M_1 \to M_2$ such that, for all $x, y \in M_1$,

$$d_2(f(x), f(y)) = d_1(x, y)$$

Definition 1. *Let (M, d), $(M_1, d_1), \ldots, (M_n, d_n)$ be metric spaces.*

1. *We define a metric d_F on the set $M_1 \to M_2$ of all functions from M_1 to M_2 as follows. For every $f_1, f_2 \in M_1 \to M_2$ we put*

$$d_F(f_1, f_2) = \sup_{x \in M_1} \{d_2(f_1(x), f_2(x))\}$$

   This supremum always exists since the codomain of our metrics is always $[0, 1]$. The set $M_1 \to^1 M_2$ is a subset of $M_1 \to M_2$, and a metric on $M_1 \to^1 M_2$ can be obtained by taking the restriction of the corresponding d_F.

2. *With $M_1 \bar{\cup} \cdots \bar{\cup} M_n$ we denote the* disjoint union *of $M_1, \ldots, M_n$, which can be defined as $\{1\} \times M_1 \cup \cdots \cup \{n\} \times M_n$. We define a metric d_U on $M_1 \bar{\cup} \cdots \bar{\cup} M_n$ as follows. For every $x, y \in M_1 \bar{\cup} \cdots \bar{\cup} M_n$,*

$$d_U(x, y) = \begin{cases} d_j(x, y) & \text{if } x, y \in \{j\} \times M_j, 1 \leqslant j \leqslant n \\ 1 & \text{otherwise} \end{cases}$$

   If no confusion is possible we shall often write $\cup$ rather than $\bar{\cup}$.

3. *We define a metric d_P on the cartesian product $M_1 \times \cdots \times M_n$ by the following clause. For every $(x_1, \ldots, x_n), (y_1, \ldots, y_n) \in M_1 \times \cdots \times M_n$,*

$$d_P((x_1, \ldots, x_n), (y_1, \ldots, y_n)) = \max_i \{d_i(x_i, y_i)\}$$

4. *Let* $\mathbb{P}_{cl}(M) = \{X \mid X \subseteq M \wedge X \text{ is closed}\}$. *We define a metric* d_H *on* $\mathbb{P}_{cl}(M)$, *called the* Hausdorff distance, *as follows. For every* $X, Y \in \mathbb{P}_{cl}(M)$,

$$d_H(X, Y) = \max\{\sup_{x \in X}\{d(x, Y)\}, \sup_{y \in Y}\{d(y, X)\}\}$$

where $d(x, Z) = \inf_{z \in Z}\{d(x, z)\}$ *for every* $Z \subseteq M$, $x \in M$. *(We use the convention that* $\sup \emptyset = 0$ *and* $\inf \emptyset = 1$.*) The spaces*

$$\begin{aligned} \mathbb{P}_{co}(M) &= \{X \mid X \subseteq M \wedge X \text{ is compact}\} \\ \mathbb{P}_{nc}(M) &= \{X \mid X \subseteq M \wedge X \text{ is non-empty and compact}\} \end{aligned}$$

are supplied with a metric by taking the restriction of d_H.

5. *For any real number* ϵ *with* $\epsilon \in [0, 1]$ *we define*

$$id_\epsilon((M, d)) = (M, d')$$

where $d'(x, y) = \epsilon \cdot d(x, y)$, *for every* x *and* y *in* M.

Proposition 2. *Let* (M, d), $(M_1, d_1), \ldots, (M_n, d_n)$, d_F, d_U, d_P, *and* d_H *be as in Definition 1 and suppose that* (M, d), $(M_1, d_1), \ldots, (M_n, d_n)$ *are complete. We have that*

$$(M_1 \to M_2, d_F) \quad (M_1 \to^1 M_2, d_F) \tag{a}$$

$$(M_1 \bar{\cup} \cdots \bar{\cup} M_n, d_U) \tag{b}$$

$$(M_1 \times \cdots \times M_n, d_P) \tag{c}$$

$$(\mathbb{P}_{cl}(M), d_H) \quad (\mathbb{P}_{co}(M), d_H) \quad (\mathbb{P}_{nc}(M), d_H) \tag{d}$$

$$id_\epsilon((M, d)) \tag{e}$$

are complete metric spaces. If (M, d) *and* (M_i, d_i) *are all ultra-metric spaces, then so are these composed spaces. (Strictly speaking, for the completeness of* $M_1 \to M_2$ *and* $M_1 \to^1 M_2$ *we do not need the completeness of* M_1. *The same holds for the ultra-metric property.)*

Whenever in the sequel we write $M_1 \to M_2$, $M_1 \to^1 M_2$, $M_1 \bar{\cup} \cdots \bar{\cup} M_n$, $M_1 \times \cdots \times M_n$, $\mathbb{P}_{cl}(M)$, $\mathbb{P}_{co}(M)$, $\mathbb{P}_{nc}(M)$, or $id_\epsilon(M)$, we mean the metric space with the metric defined above.

The proofs of Proposition 2(a), (b), (c), and (e) are straightforward. Part (d) is more involved. It can be proved with the help of the following characterization of the completeness of $(\mathbb{P}_{cl}(M), d_H)$.

Proposition 3. *Let $(\mathbb{P}_{cl}(M), d_H)$ be as in Definition 1. Let $(X_i)_i$ be a Cauchy sequence in $\mathbb{P}_{cl}(M)$. We have*

$$\lim_{i\to\infty} X_i = \{\lim_{i\to\infty} x_i \mid x_i \in X_i, (x_i)_i \text{ a Cauchy sequence in } M\}$$

Proofs of Propositions 2(d) and 3 can be found in, for instance, (Dugundji 1966) and (Enkelking 1977). The proofs are also repeated in (de Bakker and Zucker 1982). The completeness of the Hausdorff space containing compact sets is proved in (Michael 1951).

We proceed by introducing a category of complete metric spaces and some basic definitions, after which a categorical fixed point theorem will be formulated.

Definition 4. (Category of complete metric spaces) *Let $\mathcal{C}$ denote the category that has complete metric spaces for its objects. The arrows ι in $\mathcal{C}$ are defined as follows. Let M_1, M_2 be complete metric spaces. Then $M_1 \to^{\iota} M_2$ denotes a pair of maps $M_1 \rightleftarrows^{i}_{j} M_2$, satisfying the following properties:*

1. *i is an isometric embedding;*
2. *j is non-distance-increasing (NDI);*
3. *$j \circ i = id_{M_1}$.*

(We sometimes write $\langle i, j\rangle$ for ι.) Composition of the arrows is defined in the obvious way.

We can consider M_1 as an approximation to M_2. In a sense, the set M_2 contains more information than M_1, because M_1 can be isometrically embedded into M_2. Elements in M_2 are approximated by elements in M_1. For an element $m_2 \in M_2$ its (best) approximation in M_1 is given by $j(m_2)$. Clause 3 states that M_2 is a consistent extension of M_1.

Definition 5. *For every arrow $M_1 \to^{\iota} M_2$ in $\mathcal{C}$ with $\iota = \langle i, j\rangle$ we define*

$$\delta(\iota) = d_{M_2 \to M_1}(i \circ j, id_{M_2}) \quad (= \sup_{m_2 \in M_2} \{d_{M_2}(i \circ j(m_2), m_2)\})$$

This number can be regarded as a measure of the quality with which M_2 is approximated by M_1: the smaller $\delta(\iota)$, the better M_2 is approximated by M_1.

Increasing sequences of metric spaces are generalized in the following definition.

Definition 6. (Converging tower)

1. *We call a sequence* $(D_n, \iota_n)_n$ *of complete metric spaces and arrows a* tower *whenever we have that* $\forall n \in \mathbb{N} \cdot D_n \rightarrow^{\iota_n} D_{n+1} \in \mathcal{C}$.
2. *The sequence* $(D_n, \iota_n)_n$ *is called a* converging tower *when furthermore the following condition is satisfied:*

$$\forall \epsilon > 0 \cdot \exists N \in \mathbb{N} \cdot \forall m > n \geqslant N \cdot \delta(\iota_{nm}) < \epsilon$$

where $\iota_{nm} = \iota_{m-1} \circ \cdots \circ \iota_n : D_n \rightarrow D_m$.

A special case of a converging tower is a tower $(D_n, \iota_n)_n$ satisfying, for some ϵ with $0 \leqslant \epsilon < 1$,

$$\forall n \in \mathbb{N} \cdot \delta(\iota_{n+1}) \leqslant \epsilon \cdot \delta(\iota_n)$$

Note that

$$\begin{aligned} \delta(\iota_{nm}) &\leqslant \delta(\iota_n) + \cdots + \delta(\iota_{m-1}) \\ &\leqslant \epsilon^n \cdot \delta(\iota_0) + \cdots + \epsilon^{m-1} \cdot \delta(\iota_0) \\ &\leqslant \frac{\epsilon^n}{1-\epsilon} \cdot \delta(\iota_0) \end{aligned}$$

We shall now generalize the technique of forming the metric *completion* of the union of an increasing sequence of metric spaces by proving that, in $\mathcal{C}$, every converging tower has an *initial cone.* The construction of such an initial cone for a given tower is called the *direct limit* construction. Before we treat this direct limit construction, we first give the definition of a cone and an initial cone.

Definition 7. (Cone) *Let* $(D_n, \iota_n)_n$ *be a tower. Let* D *be a complete metric space and* $(\gamma_n)_n$ *a sequence of arrows. We call* $(D, (\gamma_n)_n)$ *a* cone *for* $(D_n, \iota_n)_n$ *whenever the following condition holds:*

$$\forall n \in \mathbb{N} \cdot D_n \rightarrow^{\gamma_n} D \in \mathcal{C} \ \wedge \ \gamma_n = \gamma_{n+1} \circ \iota_n$$

Definition 8. (Initial cone) *A cone* $(D, (\gamma_n)_n)$ *for a tower* $(D_n, \iota_n)_n$ *is called* initial *whenever for every other cone* $(D', (\gamma'_n)_n)$ *for* $(D_n, \iota_n)_n$ *there exists a unique arrow* $\iota : D \rightarrow D'$ *in* $\mathcal{C}$ *such that*

$$\forall n \in \mathbb{N} \cdot \iota \circ \gamma_n = \gamma'_n$$

Definition 9. (Direct limit construction) *Let $(D_n, \iota_n)_n$, with $\iota_n = \langle i_n, j_n \rangle$, be a converging tower.* The direct limit *of $(D_n, \iota_n)_n$ is a cone $(D, (\gamma_n)_n)$, with $\gamma_n = \langle g_n, h_n \rangle$, that is defined as follows:*

$$D = \{(x_n)_n \mid \forall n \geqslant 0 \cdot x_n \in D_n \wedge j_n(x_{n+1}) = x_n\}$$

is equipped with a metric $d : D \times D \to [0,1]$ defined by

$$d((x_n)_n, (y_n)_n) = \sup\{d_{D_n}(x_n, y_n)\}$$

for all $(x_n)_n$ and $(y_n)_n \in D$. The function $g_n : D_n \to D$ is defined by $g_n(x) = (x_k)_k$, where

$$x_k = \begin{cases} j_{kn}(x) & \text{if } k < n \\ x & \text{if } k = n \\ i_{nk}(x) & \text{if } k > n \end{cases}$$

$h_n : D \to D_n$ is defined by $h_n((x_k)_k) = x_n$.

Lemma 10. *The direct limit of a converging tower (as defined in Definition 9) is an initial cone for that tower.*

As a category-theoretic equivalent of a contracting function on a metric space, we have the following notion of a *contracting functor* on $\mathcal{C}$.

Definition 11. (Contracting functor) *We call a functor $F : \mathcal{C} \to \mathcal{C}$ contracting whenever the following holds. There exists an ϵ, with $0 \leqslant \epsilon < 1$, such that, for all $D \to^{\iota} E \in \mathcal{C}$,*

$$\delta(F(\iota)) \leqslant \epsilon \cdot \delta(\iota)$$

A contracting function on a complete metric space is continuous, so it preserves Cauchy sequences and their limits. Similarly, a contracting functor preserves converging towers and their initial cones.

Lemma 12. *Let $F : \mathcal{C} \to \mathcal{C}$ be a contracting functor, and let $(D_n, \iota_n)_n$ be a converging tower with an initial cone $(D, (\gamma_n)_n)$. Then $(F(D_n), F(\iota_n))_n$ is again a converging tower with $(F(D), (F(\gamma_n))_n)$ as an initial cone.*

Theorem 13. (Fixed point theorem) *Let F be a contracting functor $F : \mathcal{C} \to \mathcal{C}$ and let $D_0 \to^{\iota_0} F(D_0) \in \mathcal{C}$. Let the tower $(D_n, \iota_n)_n$ be defined by $D_{n+1} = F(D_n)$ and $\iota_{n+1} = F(\iota_n)$ for all $n \geqslant 0$. This tower is converging, so it has a direct limit $(D, (\gamma_n)_n)$. We have $D \cong F(D)$.*

In (America and Rutten 1989a) it is shown that contracting functors that are moreover contracting on all *hom-sets* (the sets of arrows in $\mathcal{C}$ between any two given complete metric spaces) have *unique* fixed points (up to isometry). It is also possible to impose certain restrictions upon the category $\mathcal{C}$ such that every contracting functor on $\mathcal{C}$ has a unique fixed point.

Let us now indicate how this theorem can be used to solve Equations (2)–(5) above. We define

$$F_1(P) = A \cup id_{1/2}(B \times P) \tag{6}$$

$$F_2(P) = A \cup \mathbb{P}_{co}(B \times id_{1/2}(P)) \tag{7}$$

$$F_3(P) = A \cup (B \to id_{1/2}(P)) \tag{8}$$

If the expression $G(P)$ in Equation (5) is equal to P, for example, then we define F_4 by

$$F_4(P) = A \cup id_{1/2}(P \to^1 P) \tag{9}$$

Note that the definitions of these functors specify, for each metric space (P, d_P), the metric on $F(P)$ *implicitly* (see Definition 1). These metrics all satisfy Equation (1) given in the introduction (Section 5.1) for a suitably defined truncation function.

Now it is easily verified that F_1, F_2, F_3, and F_4 are contracting functors on $\mathcal{C}$. Intuitively, this is a consequence of the fact that in the definitions above each occurrence of P is preceded by a factor $id_{1/2}$. Thus these functors have a fixed point, according to Theorem 13, which is a solution for the corresponding equation. (In the sequel we shall usually omit the factor $id_{1/2}$ in the reflexive domain equations, assuming that the reader will be able to fill in the details.)

In (America and Rutten 1989a) it is shown that functors like F_1 through F_4 are also contracting on hom-sets, which guarantees that they have *unique* fixed points (up to isometry).

The results above hold for complete *ultra-metric* spaces too, which can easily be verified.

In the next section, we shall encounter pairs of reflexive equations of the form

$$P \cong F(P, Q) \qquad Q \cong G(P, Q)$$

where F and G are functors on $\mathcal{C} \times \mathcal{C}$. Equations like this can be solved by a straightforward generalization of the above theory.

5.3 Concurrency semantics

Introduction

In this section we demonstrate how (solutions of) metric domain equations can be exploited in the design of semantics for languages with some form of concurrency. Altogether we shall be concerned with five languages, and for each of them we shall develop operational ($\mathcal{O}$) and denotational ($\mathcal{D}$) semantics, and discuss the relationships between $\mathcal{O}$ and $\mathcal{D}$. The first two languages (L_0, L_1) are what may be called schematic or *uniform*: the elementary actions are uninterpreted symbols from some alphabet, and the meanings assigned to the language constructs concerned will have the flavour of formal (tree) languages. Next, we shall discuss three *non-uniform* languages (L_2, L_3, L_4), where the elementary actions are (primarily) assignments. These have state transformations as meanings, and the domains needed to handle them involve state-transforming functions in a variety of ways.

The domains employed to define the *operational semantics* for L_0 to L_4 are comparatively easy. For L_0, L_1 we introduce the domain of *streams*, that is, of finite or infinite sequences over the relevant alphabets. Finite sequences end in ϵ (δ) signalling proper (improper or deadlock) termination. Meanings of statements in L_0, L_1 will be (non-empty compact) sets of such streams, and the corresponding domains will be denoted by R_0, R_1. In order to bring out the (dis)similarities between the operational and denotational models, the stream domains R_0, R_1 are defined here as well, through domain equations. (At this stage, the reader may want to refer to the table in Section 5.3, surveying all domain equations.) For L_2 to L_4, the operational semantics domains (R_2 to R_4) are functions from states to sets of streams of states. Altogether, all operational models have streams as their basic constituents, and they may be collectively called *linear time* (LT) models.

The situation is rather different for the various denotational models. For L_0, L_1 we use (purely) *branching time* (BT) models, that is, we use the domain of 'trees' over some alphabets. 'Trees' are not just ordinary trees: they are *commutative* (no order on the successors of any node), what may be called *absorptive* (nodes have sets rather than multisets as successors), and *compact* (for this we omit a precise definition, since we use the technical framework of Section 5.2 anyhow). These properties taken together ensure that the domain of 'trees' does indeed fit into the general domain theory of Section 5.2. From now on, we use the term 'processes' (elements of a domain P solving $P \cong F(P)$) rather than 'trees'. (For a discussion concerning the relationship between the process domains and the class of process graphs modulo bisimulation we refer to (Bergstra and Klop 1989), where, under some mild conditions, isomorphism of the two structures is established.) The processes in P_0 and P_1, serving as models for L_0 and L_1, have as special elements the nil process $\{\epsilon\}$ and the empty process $\emptyset$. Again, these model proper and improper termination. For the languages L_2 to L_4, we introduce domains of processes (P_2 to P_4) which in some manner involve function spaces. Domain P_2 is the simplest of these: it consists of all non-empty compact subsets of a domain Q_2, where Q_2 is built recursively from itself and constant domains using the operators $\rightarrow$, $\times$, and $\cup$, but without the use of the power domain operator. Though slightly different from P_2, P_3 shares with P_2 the property that the power domain operator does not appear in a recursive way. Only when we define P_4 do we have that the power domain operator occurs combined with recursion. Since this kind of combination constitutes the essence of a domain being branching time, we are justified in calling P_4 a non-uniform BT model, whereas P_2, P_3 are, though non-uniform, more of the LT variety.

(In previous papers such as (de Bakker and Zucker 1982, de Bakker *et al.* 1988, de Bakker and Meyer 1988, de Bakker 1989) we have always considered, for the non-uniform case, only domains which are fully BT (such as P_4). The present models P_2, P_3 are new for us. A major motive for their introduction is our desire to understand full abstractness issues better. Domains which are fully branching time are likely to provide too much information to qualify as fully abstract. We shall return to these matters below.)

We use five languages to illustrate the use of domains as outlined above. For our present purposes, the languages themselves are not our primary concern. Our first aim is to present a representative sample of the variety of domains one may employ in semantic design. Secondly, we want to emphasize the resemblance between the definitional tools. Throughout, (unique) fixed points of (contracting) higher-order mappings play a central role. For f a contracting mapping on a complete metric space, let $\text{fix}\, f$ denote its unique fixed point (which exists by Banach's theorem, cf. Section 5.2). For the operational semantics definitions we shall, for $i = 0, \ldots, 4$, define $\mathcal{O}_i = \text{fix}\, \Psi_i$, for suitable operators Ψ_i. In the definitions of the Ψ_i, we shall make fruitful use of transition systems in the sense of Plotkin's structured operational semantics (SOS), from (Hennessy and Plotkin 1979, Plotkin 1981, 1983). In the denotational case, we put $\mathcal{D}_i = \text{fix}\, \Phi_i$, $i = 0, \ldots, 4$. Here Φ_i is defined (on appropriate domains) using semantic operators such as sequential ($\circ$) and parallel ($\|$) composition. In the definition of those operators as well, use is made of the definitional technique in terms of higher-order mappings. In four out of the five cases considered, $\mathcal{O}_i$ is not *compositional*. That is, in these cases we do not have that, for each syntactic operator $\mathbf{op}_{\text{syn}}$, there exists a corresponding semantic operator $\mathbf{op}_{\text{sem}}$ such that, for all s_1, s_2, $\mathcal{O}[\![s_1\ \mathbf{op}_{\text{syn}}\ s_2]\!] = \mathcal{O}[\![s_1]\!]\ \mathbf{op}_{\text{sem}}\ \mathcal{O}[\![s_2]\!]$. (For example, for L_2 and L_4, $\|$ violates this condition.) In order to obtain compositionality, we have to add information to the codomains concerned: in going from $\mathcal{O}_i$ to $\mathcal{D}_i$, we replace R_i by P_i, and P_i is more complex than R_i. In this way we manage to define $\mathcal{D}_i$ in a compositional way, but we have lost the equivalence $\mathcal{O}_i = \mathcal{D}_i$, $i = 0, \ldots, 4$. Rather, we shall apply *abstraction* mappings $abs_i : P_i \rightarrow R_i$, $i = 0, \ldots, 4$. These mappings delete information from the P_i, and they enable us to establish that

$$\mathcal{O}_i = abs_i \circ \mathcal{D}_i \qquad i = 0, \ldots, 4 \tag{10}$$

The question concerning full abstractness asks whether these $\langle \mathcal{D}_i, abs_i \rangle$ are the best possible (in a sense to be defined precisely below). Not much is known on this question. Apart from a few negative results ($\mathcal{D}_i$ is not fully abstract on the basis of known facts), essentially all we have to report here is a few open problems.

We conclude this introduction with a listing of the programming notions appearing in languages L_0 to L_4.

L_0, L_1 **(the uniform case).** Both have elementary actions, sequential composition, non-deterministic choice, and guarded recursion. Guardedness is a syntactic restriction reminiscent of Greibach normal form for context-free grammars. It is imposed to ensure contractivity (of an operator corresponding to (the declarations of) the program). Moreover:

- L_0 has parallel composition;
- L_1 has process creation and (CCS-like) synchronization.

L_2, L_3, L_4 **(the non-uniform case).** Each language has assignment, sequential composition, the conditional statement, and (arbitrary) recursion. In addition:

- L_2 has parallel composition;
- L_3 has process creation and (a form of) local variables;
- L_4 has parallel composition and (CSP-like) communication.

In each of L_0 to L_4, a program consists of a (main) statement s and a set D of declarations. This set 'declares' procedure variables x with corresponding bodies g (the guarded case) or s (the general case). These declarations are (therefore) *simultaneous* and they may involve mutually recursive constructs. Note that we do not utilize some form of μ-notation (in the form of $\mu x[s]$, say) to introduce recursion syntactically. The simultaneous format has technical advantages here (the interested reader may want to compare the technicalitites of (Kok and Rutten 1988) with those of (de Bakker and Meyer 1988)).

L_0: a uniform language with parallel composition

Our first language, L_0, is quite simple. It is introduced for the purpose of illustrating the definitional techniques on an elementary case. We shall design LT operational and BT denotational models for L_0. The motivation for using a BT model for L_0 is solely didactic: we want to explain the somewhat complicated machinery of BT models first for a very simple language (for which even the operational semantics $\mathcal{O}_0$ is already compositional, thus obviating the need for a more complex domain $\mathcal{D}_0$).

(From now on we employ the terminology 'let $(x\in)\,M$ be ...' to introduce a set M with a variable x ranging over M.) Let $(a\in)\,A$ be an alphabet of *elementary actions*, and let $(x\in)\,Pvar$ be an alphabet of *procedure variables.* We introduce the language L_0 and its guarded version L_0^g in the following.

Definition 14. *$(s\in)\,L_0$, $(g\in)\,L_0^g$ and $(D\in)\,Decl_0$ are given by*

1. $s ::= a \mid x \mid s_1 \,;\, s_2 \mid s_1 + s_2 \mid s_1 \,\|\, s_2$
2. $g ::= a \mid g \,;\, s \mid g_1 + g_2 \mid g_1 \,\|\, g_2$
3. *A declaration D consists of a set of pairs (x, g) and a program consists of a pair (D, s).*

Remarks.

1. We find it convenient not to worry about the ambiguity in the syntax for L_0 (L_0^g) — and the other languages we shall define in the sequel. If required, the reader may add parentheses around the composite constructs, or assign priorities to the operators.
2. In a guarded g, each occurrence of a procedure variable x is 'guarded' by a sequentially preceding occurrence of some $a \in A$.

We proceed with the definitions leading up to the operational semantics $\mathcal{O}_0$ for L_0. Let E be a new symbol (not in A or $Pvar$) with as connotation 'the terminated statement', and let $(r\in)\,L_0^+ = L_0 \cup \{E\}$. *Transitions* are four-tuples of the form $\langle s, a, D, r\rangle$, with $s \in L_0$, $a \in A$, $D \in Decl_0$, $r \in L_0^+$. A transition relation $\rightarrow$ is any subset of $L_0 \times A \times Decl_0 \times L_0^+$. Instead of $\langle s, a, D, r\rangle \in \rightarrow$ we write $s \xrightarrow{a}_D r$. From now on, we shall suppress explicit mention of D in our notation. For example, we shall use $s \xrightarrow{a} r$ rather than $s \xrightarrow{a}_D r$, and, at later stages, we use $\mathcal{O}[\![s]\!]$ rather than $\mathcal{O}[\![(D, s)]\!]$, etc. We feel free to do so since D is in no way manipulated in our considerations. Each time, where relevant, some fixed D may be assumed.

As the next step, we introduce a specific transition relation $\rightarrow_0$ in terms of what may be called a formal *transition system* T_0 (consisting of some *axioms* and some *rules*).

Definition 15. $\rightarrow_0$ *is the least relation satisfying the following system* T_0*:*

1. $a \xrightarrow{a}_0 E$
2. *If* $s \xrightarrow{a}_0 r$ *then*

$$\begin{aligned} s\,;\bar{s} &\xrightarrow{a}_0 r\,;\bar{s} \\ s \parallel \bar{s} &\xrightarrow{a}_0 r \parallel \bar{s} \\ \bar{s} \parallel s &\xrightarrow{a}_0 \bar{s} \parallel r \\ s + \bar{s} &\xrightarrow{a}_0 r \\ \bar{s} + s &\xrightarrow{a}_0 r \end{aligned}$$

3. *If* $g \xrightarrow{a}_0 r$ *then* $x \xrightarrow{a}_0 r$*, where* $(x, g) \in D$*.*

Remark. In Clause 2 we use the convention that (in the case $r = E$) $E\,;\bar{s} = E \parallel \bar{s} = \bar{s} \parallel E = \bar{s}$.

We now introduce the operational domains $(r\in)\,R_0$, $(u\in)\,S_0$, and show how to define $\mathcal{O}_0 : L_0 \rightarrow R_0$.

Definition 16.

1. $R_0 = \mathbb{P}_{nc}(S_0)$, $S_0 = (A \times S_0) \cup \{\delta, \epsilon\}$
2. *Let* $(F\in)\,M_0 = L_0^+ \rightarrow R_0$*, and let* $\Psi_0 : M_0 \rightarrow M_0$ *be defined as follows:*

$$\begin{aligned} \Psi_0(F)(E) &= \{\epsilon\} \\ \Psi_0(F)(s) &= \begin{cases} \{\langle a, u\rangle \mid s \xrightarrow{a}_0 r \ \wedge\ u \in F(r)\} & \text{if this set is non-empty} \\ \{\delta\} & \text{otherwise} \end{cases} \end{aligned}$$

3. $\mathcal{O}_0 = \text{fix}\,\Psi_0$

Remarks.

1. In Clause 1, ϵ and δ are new symbols which denote proper and improper termination respectively.
2. By the definition of $\rightarrow_0$, $\{\delta\}$ will never be delivered in Clause 2. We have included this case for consistency with later definitions, where the set $\{\langle a, u\rangle \mid \cdots\}$ may well be empty.
3. For each F and s, $\Psi_0(F)(s)$ is a non-empty compact set (this follows from the definition of T_0). Moreover, Ψ_0 is a contracting operator (on the complete metric space M_0). This depends essentially on our convention (see the remark following Theorem 13) that in a domain equation such as that for S_0, recursive occurrences are implicitly proceeded by the $id_{1/2}$ operator.

Examples 17.

1. $\mathcal{O}[\![(a_1\,;a_2)+a_3]\!] = \{\langle a_1, \langle a_2, \epsilon\rangle\rangle, \langle a_3, \epsilon\rangle\}$
2. $\mathcal{O}[\![((x,(a\,;x)+b),x)]\!]$
$$= \{\underbrace{\langle a, \langle a, \ldots\rangle\rangle}_{\omega \text{ times } a}\} \cup \{\underbrace{\langle a, \langle a, \ldots, \langle b, \epsilon\rangle \ldots\rangle\rangle}_{i \text{ times } a} | i = 0, 1, \ldots\}$$
(In a less cumbersome notation, we would write $\{a^\omega\} \cup a^*b$.)

We continue with the denotational definition for L_0. We shall, here and subsequently, follow a fixed pattern, in that we first introduce the denotational domains, then define the necessary semantic operators, and finally define a higher-order mapping Φ_i which has the desired $\mathcal{D}_i$ as fixed point.

Definition 18.

1. $P_0 = \mathbb{P}_{co}(Q_0) \cup \{\{\epsilon\}\}$, $Q_0 = A \times P_0$
2. *Let* $(\phi\in)\,\mathbb{P}_0 = P_0 \times P_0 \to P_0$. *The operator* $+ \in \mathbb{P}_0$ *is defined by* $p + \{\epsilon\} = \{\epsilon\} + p = p$, *and, for* p_1, $p_2 \neq \{\epsilon\}$, $p_1 + p_2$ *is the set-theoretic union of* p_1 *and* p_2. *Also, the operators* $\circ$ *and* $\|$ *are defined by* $\circ = \text{fix}\,\Omega_\circ$, $\| = \text{fix}\,\Omega_\|$, *where* $\Omega_\circ$, $\Omega_\| : \mathbb{P}_0 \to \mathbb{P}_0$ *are given by*
$$\begin{aligned} \Omega_\circ(\phi)(p_1,p_2) &= \begin{cases} p_2 & \text{if } p_1 = \{\epsilon\} \\ \{\langle a, \phi(p')(p_2)\rangle \mid \langle a,p'\rangle \in p_1\} & \text{if } p_1 \neq \{\epsilon\} \end{cases} \\ \Omega_\|(\phi)(p_1,p_2) &= \Omega_\circ(\phi)(p_1,p_2) + \Omega_\circ(\phi)(p_2,p_1) \end{aligned}$$
3. *Let* $(F\in)\,N_0 = L_0 \to P_0$, *and let* $\Phi_0 : N_0 \to N_0$ *be given by*

   (for $g \in L_0^g$**)**
$$\begin{aligned} \Phi_0(F)(a) &= \{\langle a, \{\epsilon\}\rangle\} \\ \Phi_0(F)(g\,;s) &= \Phi_0(F)(g) \circ F(s) \\ \Phi_0(F)(g_1+g_2) &= \Phi_0(F)(g_1) + \Phi_0(F)(g_2) \\ \Phi_0(F)(g_1 \| g_2) &= \Phi_0(F)(g_1) \| \Phi_0(F)(g_2) \end{aligned}$$

   (for $s \in L_0$**)**
$$\begin{aligned} \Phi_0(F)(a) &= \{\langle a, \{\epsilon\}\rangle\} \\ \Phi_0(F)(x) &= \Phi_0(F)(g) \quad \text{with } (x,g) \in D \\ \Phi_0(F)(s_1\,;s_2) &= \Phi_0(F)(s_1) \circ \Phi_0(F)(s_2) \end{aligned}$$

   and similarly for $s_1 + s_2$, $s_1 \| s_2$.

4. *Let* $\mathcal{D}_0 = \text{fix}\,\Phi_0$.

Examples 19.

1. *We use an abbreviated notation for processes in* P_0*: we write* $a \cdot p$ *for* $\langle a, p\rangle$*, we omit final* $\cdot \{\epsilon\}$*, and we write* $q_1 + q_2 + \cdots$ *for process* $p(\neq \{\epsilon\})$ *with elements* $q_1, q_2, \ldots$. *Examples of elements in* P_0 *are* $\emptyset$, $\{\epsilon\}$, $(a_1 \cdot a_2) + (a_1 \cdot a_3)$, $a_1 \cdot (a_2 + a_3)$, $a_1 \cdot (a_2 \cdot a_3 + a_3 \cdot a_2) + a_3 \cdot a_1 \cdot a_2$, *and the processes* p', p'', p''' *defined by*

$$\begin{array}{lllllll} p' &=& \lim_i p'_i & p'_0 &=& \{\epsilon\} & p'_{i+1} = a \cdot p'_i \\ p'' &=& \lim_i p''_i & p''_0 &=& \{\epsilon\} & p''_{i+1} = a \cdot p''_i + b \\ p''' &=& \lim_i p'''_i & p'''_0 &=& \{\epsilon\} & p'''_{i+1} = a \cdot p'''_i \end{array}$$

2. *Putting* $\mathbin{⌊\!⌊} = \Omega_0(\|)$*, we have* $p_1 \| p_2 = (p_1 \mathbin{⌊\!⌊} p_2) + (p_2 \mathbin{⌊\!⌊} p_1)$. *Also,* $(a_1 \cdot a_2) \| a_3 = a_1 \cdot (a_2 \cdot a_3 + a_3 \cdot a_2) + a_3 \cdot a_1 \cdot a_2$. *Moreover,* $\emptyset + p = p + \emptyset = p$, $\emptyset \circ p = \emptyset$ *(but* $p \circ \emptyset = \emptyset$ *only if* $p = \{\epsilon\}$ *or* $p = \emptyset$*). Also, for* p', p'', p''' *as in 19(1), we have, for any* p, $p' \circ p = p'$, $p'' \circ p = a \cdot p'' + b \cdot p$, *and* $p''' \circ p = p'''$.

3.

$$\begin{array}{rcl} \mathcal{D}_0[\![a_1\,;(a_2 + a_3)]\!] &=& a_1 \cdot (a_2 + a_3) \\ \mathcal{D}_0[\![(a_1\,;a_2) + (a_1\,;a_3)]\!] &=& (a_1 \cdot a_2) + (a_1 \cdot a_3) \\ \mathcal{D}_0[\![(a_1\,;a_2) \,\|\, a_3]\!] &=& a_1 \cdot ((a_2 \cdot a_3) + (a_3 \cdot a_2)) + a_3 \cdot a_1 \cdot a_2 \\ \mathcal{D}_0[\![((x, a\,;x), x)]\!] &=& p' \quad \text{as in 19(1)} \\ \mathcal{D}_0[\![((x, a\,;x + b), x)]\!] &=& p'' \quad \text{as in 19(1)} \\ \mathcal{D}_0[\![((x, a\,;x + b\,;x), x)]\!] &=& p''' \quad \text{as in 19(1)} \end{array}$$

Remark. Well-definedness of Φ_0 follows by induction on the complexity of first g and then any s. Contractivity follows, essentially, from the way we have defined $\Phi_0(F)(g;s)$, together with the fact that, for d the metric as determined by the definitions in Section 5.2, we have that $d(p \circ p_1, p \circ p_2) \leqslant d(p_1, p_2)/2$, for $p \neq \{\epsilon\}$.

We now discuss how to relate $\mathcal{O}_0$ and $\mathcal{D}_0$, using the *abstraction* mapping $abs_0 : P_0 \to R_0$. We shall define abs_0 in such a way that each process p is mapped onto the set of all its 'paths'. For compact p, we have that $abs_0(p)$ is indeed a non-empty compact set; hence $abs_0(p)$ is a well-defined element of R_0. (We refer to (de Bakker *et al.* 1984) for a discussion including full proofs of these issues.)

Definition 20.

1. *Let* $(\pi\in) PR_0 = P_0 \to R_0$, *and let* $\Delta_0 : PR_0 \to PR_0$ *be given by*

$$\begin{aligned} \Delta_0(\pi)(\emptyset) &= \{\delta\} \\ \Delta_0(\pi)(\{\epsilon\}) &= \{\epsilon\} \\ \Delta_0(\pi)(p) &= \{\langle a, u\rangle \mid \langle a, p'\rangle \in p \wedge u \in \pi(p')\} \quad \text{for } p \neq \emptyset, \{\epsilon\} \end{aligned}$$

2. *Let* $abs_0 = \text{fix}\,\Delta_0$.

Example 21.

$$\begin{aligned} abs_0((a_1 \cdot a_2) + (a_1 \cdot a_3)) &= abs_0(a_1 \cdot (a_2 + a_3)) \\ &= \{\langle a_1, \langle a_2, \epsilon\rangle\rangle, \langle a_1, \langle a_3, \epsilon\rangle\rangle\} \end{aligned}$$

Also, $abs_0(\emptyset) = \{\delta\}$.

We need one slight extension to $\mathcal{D}_0$ before we can relate $\mathcal{D}_0$ and $\mathcal{O}_0$. Let $\hat{\mathcal{D}}_0 : L_0^+ \to P_0$ be given by: $\hat{\mathcal{D}}_0[\![E]\!] = \{\epsilon\}$, $\hat{\mathcal{D}}_0[\![s]\!] = \mathcal{D}_0[\![s]\!]$. We have the following theorem.

Theorem 22. $\mathcal{O}_0 = abs_0 \circ \hat{\mathcal{D}}_0$.

Proof (outline). First we introduce an intermediate operational semantics $\mathcal{I} : L_0^+ \to P_0$, defined as follows. Let $(F\in) N_0^+ = L_0^+ \to P_0$, and let $\Psi_{\mathcal{I}} : N_0^+ \to N_0^+$ be given by

$$\begin{aligned} \Psi_{\mathcal{I}}(F)(E) &= \{\epsilon\} \\ \Psi_{\mathcal{I}}(F)(s) &= \{\langle a, F(r)\rangle \mid s \xrightarrow{a}_0 r\} \end{aligned}$$

Let $\mathcal{I} \stackrel{\text{def}}{=} \text{fix}\,\Psi_{\mathcal{I}}$. Following (de Bakker and Meyer 1988, Kok and Rutten 1988) we may show that $\mathcal{I} = \hat{\mathcal{D}}_0$ by establishing that $\Psi_{\mathcal{I}}(\hat{\mathcal{D}}_0) = \hat{\mathcal{D}}_0$ (followed by an appeal to Banach's theorem). Next, we have, by the various definitions,

$$\Psi_0(abs_0 \circ F)(r) = abs_0(\Psi_{\mathcal{I}}(F)(r))$$

Hence $\Psi_0(abs_0 \circ \mathcal{I})(r) = abs_0(\Psi_{\mathcal{I}}(\mathcal{I})(r)) = (abs_0 \circ \mathcal{I})(r)$. Thus, $abs_0 \circ \mathcal{I} = abs_0 \circ \hat{\mathcal{D}}_0$ is a fixed point of Ψ_0, and $abs_0 \circ \hat{\mathcal{D}}_0 = \mathcal{O}_0$ follows. □

L_1: a uniform language with process creation and synchronization

We next consider the language L_1 embodying two important variations on L_0. Firstly, the construct of parallel composition is replaced by that of process creation (here 'process' refers to a programming concept, and not to a mathematical process p in some domain P). Secondly, we add a notion of (CCS-like) synchronization. We now take the set of elementary actions A to consist of two disjoint subsets $(b\in) B$ and $(c\in) C$, where the actions in B may be taken as *independent*. Moreover, for each c in C we assume a counterpart $\bar{c}$ in C (where $\bar{\bar{c}} = c$), with the understanding that execution of c in some component has to *synchronize* with execution of $\bar{c}$ in a parallel component (and then delivers a special action τ in B as a result). Process creation is expressed through the construct **new**(s): its execution amounts to the creation of a new process which has the task of executing s in parallel with the execution of the already existing processes (each with its already associated task). In addition, we stipulate that termination of a number of parallel processes requires termination of all its components. This brief description of the meaning of **new**(s) (many details are given in (America and de Bakker 1988)) is elaborated in the formal definitions to follow.

Definition 23. *$(s\in) L_1$, $(g\in) L_1^g$ and the auxiliary $(h\in) L_1^h$ are defined by:*

1. $s ::= a \mid x \mid s_1 ; s_2 \mid s_1 + s_2 \mid \mathbf{new}(s)$
2. $g ::= h \mid g_1 ; g_2 \mid g_1 + g_2 \mid \mathbf{new}(g)$
3. $h ::= a \mid h ; s \mid h_1 + h_2$
4. *A program is a pair (D, s), where D consists of pairs (x, g)*

Remark. Using only $g \in L_1^g$ (and no $h \in L_1^h$) would lead us to the definition $g ::= a \mid g ; s \mid g_1 + g_2 \mid \mathbf{new}(g)$. Then $\mathbf{new}(a) ; x$ would qualify as guarded, which is undesirable since this will obtain the same effect as the L_0-statement $a \parallel x$ (which is unguarded since it may start with execution of x).

We proceed with the definitions for the operational semantics $\mathcal{O}_1$.

Definition 24.

1. *$(r\in) L_1^+$ is given by $r ::= E \mid s ; r$ (r may be seen as a syntactic continuation). $(\rho\in) Par_1$ is given by $\rho ::= \langle r_1, r_2, \ldots, r_n\rangle$, $n \geqslant 1$. We shall identify $\langle r\rangle$ and r. Concatenation of tuples ρ_1, ρ_2 will be denoted by $\rho_1 : \rho_2$.*

2. *Transitions are written as* $\rho_1 \xrightarrow{a}_1 \rho_2$, *where* $\rightarrow_1$ *is the smallest relation satisfying the formal system* T_1 *given by*

3. $a\,;r \xrightarrow{a}_1 r$

   If $s\,;r \xrightarrow{a}_1 \rho$ *then* $(s+\bar{s})\,;r \xrightarrow{a}_1 \rho$ *and* $(\bar{s}+s)\,;r \xrightarrow{a}_1 \rho$

   If $g\,;r \xrightarrow{a}_1 \rho$ *then* $x\,;r \xrightarrow{a}_1 \rho$, *where* $(x,g) \in D$

   If $s_1\,;(s_2\,;r) \xrightarrow{a}_1 \rho$ *then* $(s_1\,;s_2)\,;r \xrightarrow{a}_1 \rho$

   If $\langle r, s;E\rangle \xrightarrow{a}_1 \rho$ *then* $\mathbf{new}(s)\,;r \xrightarrow{a}_1 \rho$

   If $\rho_1 \xrightarrow{a}_1 \rho_2$ *then* $\rho : \rho_1 \xrightarrow{a}_1 \rho : \rho_2$ *and* $\rho_1 : \rho \xrightarrow{a}_1 \rho_2 : \rho$

   If $\rho_1 \xrightarrow{c}_1 \rho'$ *and* $\rho_2 \xrightarrow{\bar{c}}_1 \rho''$ *then* $\rho_1 : \rho_2 \xrightarrow{\tau}_1 \rho' : \rho''$

We present the next definition of (the domain for) $\mathcal{O}_1$.

Definition 25.

1. $R_1 = \mathbb{P}_{nc}(S_1)$, $S_1 = (B \times S_1) \cup \{\delta, \epsilon\}$

2. *Let* $(F\in)\, M_1 = Par_1 \rightarrow R_1$, *and let* $\Psi_1 : M_1 \rightarrow M_1$ *be given by*

$$\Psi_1(F)(\rho) \;=\; \{\epsilon\} \quad \text{for } \rho = \langle E, \ldots, E\rangle$$

   Otherwise

$$\Psi_1(F)(\rho) \;=\; \begin{cases} \{\langle a,u\rangle \mid \rho \xrightarrow{a}_1 \rho' \;\wedge\; u \in F(\rho') \;\wedge\; a \in B\}, \\ \qquad \text{if this set is non-empty} \\ \{\delta\} \quad \text{otherwise} \end{cases}$$

3. $\mathcal{O}_1 = \text{fix}\,\Psi_1$

Examples 26.

1.

$$\begin{aligned} \mathcal{O}_1[\![b\,;E]\!] &= \mathcal{O}_1[\![\mathbf{new}(b)\,;E]\!] \;=\; \{\langle b,\epsilon\rangle\} \\ \mathcal{O}_1[\![b_1\,;b_2\,;E]\!] &= \{\langle b_1, \langle b_2, \epsilon\rangle\rangle\} \\ \mathcal{O}_1[\![\mathbf{new}(b_1)\,;b_2\,;E]\!] &= \{\langle b_1, \langle b_2, \epsilon\rangle\rangle, \langle b_2, \langle b_1, \epsilon\rangle\rangle\} \end{aligned}$$

2.

$$\begin{aligned} \mathcal{O}_1[\![c\,;E]\!] &= \mathcal{O}_1[\![\bar{c}\,;E]\!] \;=\; \{\delta\} \\ \mathcal{O}_1[\![\langle c\,;E, \bar{c}\,;E\rangle]\!] &= \{\tau\} \end{aligned}$$

3.

$$\mathcal{O}_1[\![\mathbf{new}(c)\,;b_1\,;\mathbf{new}(\bar{c})\,;b_2\,;E]\!] \;=\; \{\langle b_1,\langle\tau,\langle b_2,\epsilon\rangle\rangle\rangle,\langle b_1,\langle b_2,\langle\tau,\epsilon\rangle\rangle\rangle\}$$

4.

$$\mathcal{O}_1[\![(b_1\,;b_2)+(b_1\,;c)]\!] \;=\; \{\langle b_1,\langle b_2,\epsilon\rangle\rangle,\langle b_1,\delta\rangle\}$$

From the examples we see that $\mathcal{O}_1$ is not compositional (Examples 26(1) and 26(2) show this with respect to ; and :). We remedy this as follows: in order to handle : we introduce the BT domain P_1 (refining R_1). P_1 is the same as P_0 from the previous section, but now its branching structure is indeed exploited. Process creation (and the ensuing problems with ;) is dealt with in a different way, namely by using the technique of so-called *semantic continuations*. We shall define $\mathcal{D}_1 : L_1 \to (P_1 \to P_1)$, rather than just $\mathcal{D}_1 : L_1 \to P_1$. Details follow in the next definition.

Definition 27.

1. *$P_1 = P_0$, $Q_1 = Q_0$*
2. *Let $(\phi\in)\,\mathbb{P}_1 = P_1 \times P_1 \to P_1$. We define $+ \in \mathbb{P}_1$, and $\Omega_0 : \mathbb{P}_1 \to \mathbb{P}_1$ as in Definition 18. Also, $\Omega_{\|} : \mathbb{P}_1 \to \mathbb{P}_1$ is given by $\Omega_{\|}(\phi)(p_1,p_2) = \Omega_\circ(\phi)(p_1,p_2) + \Omega_\circ(\phi)(p_2,p_1) + \Omega_|(\phi)(p_1,p_2)$, where*

$$\Omega_|(\phi)(p_1,p_2) \;=\; \{\langle\tau,\phi(p',p'')\rangle \mid \langle c,p'\rangle \in p_1 \;\wedge\; \langle\bar{c},p''\rangle \in p_2\}$$

   Let $\| = \mathrm{fix}\,\Omega_{\|}$.

3. *In the definition of Φ_1 we use an extra argument (from P_1), namely the semantic continuation. Let $(F\in)\,N_1 = L_1 \to (P_1 \to P_1)$, and let $\Phi_1 : N_1 \to N_1$ be given by*

   (for $h \in L_1^h$)

$$\begin{aligned}\Phi_1(F)(a)(p) &= \{\langle a,p\rangle\}\\ \Phi_1(F)(h\,;s)(p) &= \Phi_1(F)(h)(F(s)(p))\\ \Phi_1(F)(h_1+h_2)(p) &= \Phi_1(F)(h_1)(p)+\Phi_1(F)(h_2)(p)\end{aligned}$$

   (for $g \in L_1^g$)

$$\begin{aligned}\Phi_1(F)(h)(p) &= \textit{as above}\\ \Phi_1(F)(g_1\,;g_2)(p) &= \Phi_1(F)(g_1)(\Phi_1(F)(g_2)(p))\\ \Phi_1(F)(g_1+g_2)(p) &= \Phi_1(F)(g_1)(p)+\Phi_1(F)(g_2)(p)\\ \Phi_1(F)(\mathbf{new}(g))(p) &= \Phi_1(F)(g)(\{\epsilon\})\,\|\,p\end{aligned}$$

(**for** $s \in L_1$)

$$\begin{aligned} \Phi_1(F)(x)(p) &= \Phi_1(F)(g)(p) \quad \text{where } (x,g) \in D \\ \Phi_1(F)(\mathbf{new}(s))(p) &= \Phi_1(F)(s)(\{\epsilon\}) \parallel p \end{aligned}$$

The cases $s \equiv a,\ s_1 \,;\, s_2,\ s_1 + s_2$ *are similar to the above.*

4. $\mathcal{D}_1 = \operatorname{fix} \Phi_1$.

Examples 28.

1. $\mathcal{D}_1[\![c]\!](p) = \{\langle c, p\rangle\}$, *and, using the abbreviated notation for processes in* P_0 $(= P_1)$ *from the previous section,* $\mathcal{D}_1[\![\mathbf{new}(c)\,;\,\bar{c}]\!](\{\epsilon\}) = c \cdot \bar{c} + \bar{c} \cdot c + \tau$.

2. $\mathcal{D}_1[\![\mathbf{new}(b_1)\,;\,b_2]\!](p) = \{\langle b_1, \{\langle b_2, p\rangle\}\rangle, \langle b_2, \{\langle b_1, \{\epsilon\}\rangle\} \parallel p\rangle\}$.

We see that $\mathcal{D}_1$ makes more distinctions than does $\mathcal{O}_1$: $\mathcal{O}_1[\![c_1\,;\,E]\!] = \{\delta\} = \mathcal{O}_1[\![c_2\,;\,E]\!]$, whereas $\mathcal{D}_1[\![c_1]\!] = \lambda p \cdot \{\langle c_1, p\rangle\} \neq \lambda p \cdot \{\langle c_2, p\rangle\} = \mathcal{D}_1[\![c_2]\!]$. Also, $\mathcal{O}_1[\![b\,;\,E]\!] = \{\langle b, \{\epsilon\}\rangle\} = \mathcal{O}_1[\![\mathbf{new}(b)\,;\,E]\!]$, whereas $\mathcal{D}_1[\![b]\!] = \lambda p \cdot \{\langle b, p\rangle\} \neq \lambda p \cdot (\{\langle b, \{\epsilon\}\rangle\} \parallel p) = \mathcal{D}_1[\![\mathbf{new}(b)]\!]$.

We next introduce the abstraction mapping $abs_1 : P_1 \to R_1$, which will be used to relate $\mathcal{O}_1$ and $\mathcal{D}_1$.

Definition 29. *Let* $(\pi\in) PR_1 = P_1 \to R_1$, *and let* $\Delta_1 : PR_1 \to PR_1$ *be given by*

$$\Delta_1(\pi)(\{\epsilon\}) = \{\epsilon\}$$

and, for $p \neq \{\epsilon\}$,

$$\Delta_1(\pi)(p) = \begin{cases} \{\langle a, u\rangle \mid \langle a, p'\rangle \in p \ \wedge\ u \in \pi(p') \ \wedge\ a \in B\}, \\ \qquad \text{if this set is non-empty} \\ \{\delta\} \quad \text{otherwise} \end{cases}$$

Let $abs_1 = \operatorname{fix} \Delta_1$.

Remark. $abs_1(p)$ yields the set of all paths from p which involve no c-steps.

Since not only the codomains, but also the domains of $\mathcal{O}_1$ and $\mathcal{D}_1$ differ, we first introduce an auxiliary semantic mapping $\mathcal{E}_1$, and then relate $\mathcal{O}_1$ and $\mathcal{E}_1$. We define $\mathcal{E}_1 : Par_1 \to P_1$ by putting $\mathcal{E}_1[\![E]\!] = \{\epsilon\}$, $\mathcal{E}_1[\![s\,;\,r]\!] = \mathcal{D}_1[\![s]\!](\mathcal{E}_1[\![r]\!])$, and $\mathcal{E}_1[\![\langle r_1, \ldots, r_n\rangle]\!] = \mathcal{E}_1[\![r_1]\!] \parallel \cdots \parallel \mathcal{E}_1[\![r_n]\!]$. We have the following theorem.

Theorem 30. $\mathcal{O}_1 = abs_1 \circ \mathcal{E}_1$

Proof (Sketch). First introduce an intermediate operational semantics $\mathcal{I}_1$ (in the style of the $\mathcal{I}$ of the previous section), and show that $\mathcal{I}_1 = \mathcal{E}_1$ (the reader may consult (de Bakker and Meyer 1988) for this). Then prove that $\mathcal{D}_1 = abs_1 \circ \mathcal{I}_1$ by an argument as in the proof of Theorem 22. □

We conclude this section with a few remarks concerning the question whether $\mathcal{D}_1$ is the 'best possible' with respect to $\mathcal{O}_1$. In technical terms, we ask whether $\mathcal{D}_1$ is *fully abstract* with respect to $\mathcal{O}_1$. Recall that we added information in the denotational domain P_1 (as compared with R_1) in order to make $\mathcal{D}_1$ compositional. In principal, it may be envisaged that more information has been added than is necessary to achieve this purpose. For a language with parallel composition (rather than process creation) and synchronization this is indeed the case. A so-called failure set model (which preserves less information than the full BT model) suffices. See (Brookes *et al.* 1984) for the notion of failure set model; (Rutten 1989) gives a theorem from (Bergstra *et al.* 1988) stating that this model is fully abstract is translated into a metric setting. This result makes it likely that, for L_1 as well, we do not have that $\mathcal{D}_1$ is fully abstract with respect to $\mathcal{O}_1$. A rigorous formulation of this fact (see (Rutten 1989) for alternative formulations and further discussion) is the following. We expect that it is *not* true that, for each $s_1, s_2 \in L_1$, the following two facts are equivalent:

1. $\mathcal{D}_1[\![s_1]\!] = \mathcal{D}_1[\![s_2]\!]$;
2. for each 'context' $C[\bullet]$ we have that $\mathcal{O}_1[\![C[s_1]]\!] = \mathcal{O}_1[\![C[s_2]]\!]$.

Here a context $C[\bullet]$ is a text with a 'hole' such that $C[s]$, the result of filling the hole with s, is a well-formed element of Par_1.

Clearly, it would already be of some interest to investigate these questions for a language L_1' with only process creation (and no synchronization).

L_2: a non-uniform language with parallel composition

We now engage upon the discussion of a number of languages of the non-uniform variety. In the first (L_2) elementary actions are replaced by assignments $v := e$, where $(v\in)\, Ivar$ is the set of *individual variables*, and $(e\in)\, Exp$ is the set of *expressions*. We also introduce the set $(b\in)\ Test$, which is the set of logical expressions. We assume a simple syntax (not specified here) for e, b. 'Simple' ensures at least that no side effects or non-termination occurs in their evaluation. Furthermore, we introduce a set of *states* $(\sigma\in)\, \Sigma = Ivar \to V$, where $(\alpha\in)\, V$ is some set of *values*. It is convenient (for later purposes) to postulate that $V \subseteq Exp$. The notation $\sigma[\alpha/v]$

denotes a state such that $\sigma[\alpha/v](v') = \mathbf{if}\ v = v'\ \mathbf{then}\ \alpha\ \mathbf{else}\ \sigma(v')\ \mathbf{fi}$. Finally, note that for non-uniform languages we shall not distinguish guarded recursion from the general case. (Contractivity of the operator corresponding to the program will be ensured by (semantically) proceeding each call of a procedure by the equivalent of a skip statement.)

The syntax for L_2 is given in the following definition.

Definition 31.

1. *$(s\in)\, L_2$ is given by*

$$s \ ::= \ v := e \mid x \mid s_1 \,;\, s_2 \mid \mathbf{if}\ b\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2\ \mathbf{fi} \mid s_1 \parallel s_2$$

2. *Declarations D are sets of pairs (x, s), and a program is a pair (D, s).*

The operational semantics $\mathcal{O}_2$ is given in terms of a relation $\rightarrow_2$: transitions are now of the form $\langle s, \sigma\rangle \rightarrow_2 \langle r, \sigma'\rangle$, with $\sigma, \sigma' \in \Sigma$, $s \in L_2$, $r \in L_2^+ = L_2 \cup \{E\}$, and $\rightarrow_2$ the smallest relation satisfying the transition system T_2 given in the following definition.

Definition 32. *$\langle v := e, \sigma\rangle \rightarrow_2 \langle E, \sigma[\alpha/v]\rangle$, where $\alpha = [\![e]\!](\sigma)$*
$\langle x, \sigma\rangle \rightarrow_2 \langle s, \sigma\rangle$, where $(x, s) \in D$
If $\langle s, \sigma\rangle \rightarrow_2 \langle r, \sigma'\rangle$ then

$$\langle s \,;\, \bar{s}, \sigma\rangle \rightarrow_2 \langle r \,;\, \bar{s}, \sigma'\rangle$$
$$\langle s \parallel \bar{s}, \sigma\rangle \rightarrow_2 \langle r \parallel \bar{s}, \sigma'\rangle$$
$$\langle \bar{s} \parallel s, \sigma\rangle \rightarrow_2 \langle \bar{s} \parallel r, \sigma'\rangle$$

with the convention that $E \,;\, \bar{s} = E \parallel \bar{s} = \bar{s} \parallel E = \bar{s}$.
If $\langle s, \sigma\rangle \rightarrow_2 \langle r, \sigma'\rangle$, then

$$\text{if } [\![b]\!](\sigma) = tt \text{ then } \langle \mathbf{if}\ b\ \mathbf{then}\ s\ \mathbf{else}\ s_2\ \mathbf{fi}, \sigma\rangle \rightarrow_2 \langle r, \sigma'\rangle$$
$$\text{if } [\![b]\!](\sigma) = f\!f \text{ then } \langle \mathbf{if}\ b\ \mathbf{then}\ s_1\ \mathbf{else}\ s\ \mathbf{fi}, \sigma\rangle \rightarrow_2 \langle r, \sigma'\rangle$$

The operational domains and semantics are given in the following definition.

Definition 33.

1. *$R_2 = \Sigma \rightarrow \mathbb{P}_{nc}(S_2)$, $S_2 = (\Sigma \times S_2) \cup \{\delta, \epsilon\}$*
2. *Let $(F\in)\, M_2 = L_2^+ \rightarrow R_2$, and let $\Psi_2 : M_2 \rightarrow M_2$ be given by*

$$\begin{aligned}
\Psi_2(F)(E) &= \lambda\sigma \cdot \{\epsilon\} \\
\Psi_2(F)(s) &= \lambda\sigma \cdot \begin{cases} \{\langle \sigma', u\rangle \mid \langle s, \sigma\rangle \rightarrow_2 \langle r, \sigma'\rangle \ \wedge\ u \in F(r)(\sigma')\}, \\ \qquad \text{if this set is non-empty} \\ \{\delta\} \quad \text{otherwise} \end{cases}
\end{aligned}$$

3. $\mathcal{O}_2 = \text{fix}\,\Psi_2$

Example 34.

$$\begin{aligned}\mathcal{O}_2[\![v := 0\,;\, v := v+1]\!] &= \mathcal{O}_2[\![v := 0\,;\, v := 1]\!]\\ &= \lambda\sigma\cdot\{\langle\sigma[0/v], \langle\sigma[1/v], \epsilon\rangle\rangle\}\end{aligned}$$

but

$$\mathcal{O}_2[\![(v := 0\,;\, v := v+1) \parallel (v := 2)]\!] \neq \mathcal{O}_2[\![(v := 0\,;\, v := 1) \parallel (v := 2)]\!]$$

From this example we see that O_2 is not compositional. We therefore add information to the domains R_2, S_2 obtaining P_2, Q_2 in such a way that $\mathcal{D}_2$ is indeed compositional. The definitions are collected below.

Definition 35.

1. $P_2 = \mathbb{P}_{nc}(Q_2)$, $Q_2 = (\Sigma \to (\Sigma \times Q_2)) \cup \{\epsilon\}$
2. *Let* $(\phi\in)\,\mathbb{P}_2 = P_2 \times P_2 \to P_2$. *The operator* $+ \in \mathbb{P}_2$ *is defined by:* $\{\epsilon\}+p = p+\{\epsilon\} = p$, *and, for* $p_1, p_2 \neq \{\epsilon\}$, p_1+p_2 *is the set-theoretic union of* p_1 *and* p_2. *The mappings* $\Omega_\circ$, $\Omega_\parallel : \mathbb{P}_2 \to \mathbb{P}_2$ *are given by*

$$\begin{aligned}\Omega_\circ(\phi)(p_1,p_2) &= \bigcup\{\tilde{\phi}(q_1)(q_2) \mid q_1 \in p_1 \;\wedge\; q_2 \in p_2\}\\ \tilde{\phi}(\epsilon)(q) &= \{q\}\end{aligned}$$

and, for $q_1 \neq \epsilon$,

$$\begin{aligned}\tilde{\phi}(q_1)(q_2) &= \{q \mid \forall\sigma\cdot q(\sigma) \in \hat{\phi}(q_1(\sigma))(q_2)\}\\ \hat{\phi}(\langle\sigma, q'\rangle)(q) &= \{\langle\sigma, \bar{q}\rangle \mid \bar{q} \in \phi(\{q'\})(\{q\})\}\end{aligned}$$

Also, $\Omega_\parallel(\phi)(p_1,p_2) = \Omega_\circ(\phi)(p_1,p_2) + \Omega_\circ(\phi)(p_2,p_1)$, $\circ = \text{fix}\,\Omega_\circ$, *and* $\parallel = \text{fix}\,\Omega_\parallel$.

3. *Let* $(F\in)\,N_2 = L_2 \to P_2$, *and let* $\Phi_2 : N_2 \to N_2$ *be given by*

$$\begin{aligned}\Phi_2(F)(v := e) &= \{\lambda\sigma\cdot\langle\sigma[\alpha/v], \epsilon\rangle\} && \alpha = [\![e]\!](\sigma)\\ \Phi_2(F)(x) &= \{\lambda\sigma\cdot\langle\sigma, \epsilon\rangle\} \circ F(s) && (x,s) \in D\\ \Phi_2(F)(s_1\,;\,s_2) &= \Phi_2(F)(s_1) \circ \Phi_2(F)(s_2)\end{aligned}$$

and similarly for $\parallel$

$$\begin{aligned}&\Phi_2(F)(\textbf{if } b \textbf{ then } s_1 \textbf{ else } s_2 \textbf{ fi})\\ &\quad = \left\{\begin{array}{l}\lambda\sigma\cdot\textbf{if } [\![b]\!](\sigma) \textbf{ then } q_1(\sigma) \textbf{ else } q_2(\sigma) \textbf{ fi}\\ \quad \mid q_1 \in \Phi_2(F)(s_1) \;\wedge\; q_2 \in \Phi_2(F)(s_2)\end{array}\right\}\end{aligned}$$

4. $\mathcal{D}_2 = \text{fix}\,\Phi_2$

We conclude this section with the introduction of the abstraction operator $abs_2 : P_2 \to R_2$.

Definition 36. *(The structure of this definition slightly deviates from the previous abstraction definitions.)*

1. *Let* $(\pi\in) Q\Sigma S_2 = Q_2 \to (\Sigma \to S_2)$. *We define* $\Delta_2' : Q\Sigma S_2 \to Q\Sigma S_2$ *by putting*

$$\Delta_2'(\pi)(\epsilon) = \lambda\sigma \cdot \epsilon$$

and, for $q \neq \epsilon$,

$$\begin{aligned} \Delta_2'(\pi)(q) &= \lambda\sigma \cdot \hat{\pi}(q(\sigma)) \\ \hat{\pi}(\langle\sigma, q\rangle) &= \langle\sigma, \pi(q)(\sigma)\rangle \end{aligned}$$

2. *Let* $abs_2' = \text{fix}\,\Delta_2'$. *Let* $abs_2 : P_2 \to P_2$ *be given by*

$$abs_2(p) = \lambda\sigma \cdot \begin{cases} \{abs_2'(q)(\sigma) \mid q \in p\} & \text{if this set is non-empty} \\ \{\delta\} & \text{otherwise} \end{cases}$$

We have (putting $\hat{\mathcal{D}}_2[\![E]\!] = \{\lambda\sigma \cdot \epsilon\}$, $\hat{\mathcal{D}}_2[\![s]\!] = \mathcal{D}_2[\![s]\!]$) the following theorem.

Theorem 37. $\mathcal{O}_2 = abs_2 \circ \hat{\mathcal{D}}_2$

The proof is a non-essential variation on previously given proofs (in turn relying on (Kok and Rutten 1988) and (de Bakker and Meyer 1988)). For the intermediate semantics definition we use the clauses

$$\begin{aligned} \Psi_{\mathcal{I}}(F)(E) &= \{\lambda\sigma \cdot \epsilon\} \\ \Psi_{\mathcal{I}}(F)(s) &= \{q \mid \forall\sigma \cdot q(\sigma) \in \{\langle\sigma', \bar{q}\rangle \mid \langle s, \sigma\rangle \to_2 \langle r, \sigma'\rangle \ \wedge\ \bar{q} \in F(r)\}\} \end{aligned}$$

As before, we have the issue of full abstractness. Is it true that, for all s_1, s_2, $\mathcal{D}_2[\![s_1]\!] = \mathcal{D}_2[\![s_2]\!]$ iff, for all contexts $C[\bullet]$, $\mathcal{O}_2[\![C[s_1]]\!] = \mathcal{O}_2[\![C[s_2]]\!]$? It has been shown by E. Horita that the answer to this question is negative.

L_3: a non-uniform language with process creation and locality

We continue with the treatment of the language L_3 which has process creation (as for L_1, but this time without some form of synchronization) and the notion of local declaration of an individual variable. We find it convenient to discuss only *initialized declarations* (cf. (de Bakker 1980 , Chapter 6)). Our first aim in this section is to motivate a type of domain of the form $P_3 = \Sigma \to \mathbb{P}_{nc}(Q_3)$, rather than the previous case $P_2 = \mathbb{P}_{nc}(Q_2)$: the elements of P_3 are (apart from special cases) of the form $\lambda\sigma \cdot \{\cdots, \langle\sigma', q'\rangle, \cdots\}$, where the 'resumptions' q' depend, in general, on the argument σ. With L_3 we intend to illustrate the need for this type of construction.

The syntax of L_3 is given in the next definition.

Definition 38.

1. *$(s\in) L_3$ is given by*

$$s ::= v := e \mid x \mid s_1 ; s_2 \mid \textbf{if } b \textbf{ then } s_1 \textbf{ else } s_2 \textbf{ fi} \mid \textbf{new}(s)$$
$$\mid \textbf{begin int } v := e \, ; s \textbf{ end} \quad \text{where } v \text{ does not occur in } e$$

2. *Declarations and programs are as usual.*

The operational semantics domains for L_3 are the same as those for L_2. We again (cf. the section on L_1) introduce $(\rho\in) Par_3$, where $\rho = \langle r_1, \ldots, r_n\rangle$, $n \geqslant 1$ (and where we identify $\langle r\rangle$ and r). Also, $r(\in L_3^+)$ is given by $r ::= E \mid s\, ; r$.

The transition system T_3 employs transitions of the form $\langle\rho,\sigma\rangle \to_3 \langle\rho',\sigma'\rangle$, where $\to_3$ is the least relation satisfying the following.

Definition 39.

1. $\langle v := e\, ; r, \sigma\rangle \to_3 \langle r, \sigma[\alpha/v]\rangle$, *where* $\alpha = [\![e]\!](\sigma)$
2. $\langle x\, ; r, \sigma\rangle \to_3 \langle s\, ; r, \sigma\rangle$, *where* $(x, s) \in D$
3. *If* $\langle s_1 ; (s_2 ; r), \sigma\rangle \to_3 \langle\rho, \sigma'\rangle$ *then* $\langle (s_1 ; s_2) ; r, \sigma\rangle \to_3 \langle\rho, \sigma'\rangle$
4. *If* $\langle\langle s\, ; E, r\rangle, \sigma\rangle \to_3 \langle\rho, \sigma'\rangle$ *then* $\langle \textbf{new}(s)\, ; r, \sigma\rangle \to_3 \langle\rho, \sigma'\rangle$
5. *If* $\langle v := e\, ; s\, ; v := \sigma(v)\, ; r, \sigma\rangle \to_3 \langle\rho, \sigma'\rangle$
   then $\langle \textbf{begin int } v := e\, ; s \textbf{ end}\, ; r, \sigma\rangle \to_3 \langle\rho, \sigma'\rangle$
6. **if** ... **fi**: *omitted*

7. *If* $\langle\rho_1,\sigma_1\rangle \rightarrow_3 \langle\rho_2,\sigma_2\rangle$ *then*

$$\langle\rho_1 : \rho, \sigma_1\rangle \rightarrow_3 \langle\rho_2 : \rho, \sigma_2\rangle$$
$$\langle\rho : \rho_1, \sigma_1\rangle \rightarrow_3 \langle\rho : \rho_2, \sigma_2\rangle$$

$\mathcal{O}_3$ is obtained from T_3 in the usual manner.

Definition 40.

1. *Let* $(F\in)\, ParR_3 = Par_3 \rightarrow R_3$, *and let* $\Psi_3 : ParR_3 \rightarrow ParR_3$ *be given by*

$$\Psi_3(F)(\langle E,\ldots,E\rangle) = \lambda\sigma\cdot\{\epsilon\}$$

   and, for $\rho \neq \langle E,\ldots,E\rangle$,

$$\Psi_3(F)(\rho) = \lambda\sigma\cdot\begin{cases}\{\langle\sigma',u\rangle \mid \langle\rho,\sigma\rangle \ \wedge\ u \in F(\rho')(\sigma')\}, \\ \quad\text{if this set is non-empty} \\ \{\delta\} \quad \text{otherwise}\end{cases}$$

2. $\mathcal{O}_3 = \text{fix}\,\Psi_3$

Example 41.

$$\begin{aligned}&\mathcal{O}_3[\![\textbf{begin int}\ v := 0\ ;\ \textbf{begin int}\ v := 1\ ;\ v' := v\ \textbf{end}\ ;\ v' := v\ \textbf{end}\ ;\ E]\!] \\ &\quad = \lambda\sigma\cdot\{[\sigma[0/v],[\sigma[1/v],[\sigma[1/v][1/v'],[\sigma[0/v][1/v'],[\sigma[0/v][0/v'], \\ &\qquad [\sigma[0/v][0/v'][\sigma(v)/v],\epsilon]]]]]]\}\end{aligned}$$

$\mathcal{O}_3$ is not compositional (cf. the discussion for $\mathcal{O}_1$), and we resort to a more complex domain for the denotational semantics. In the remainder of this section we shall employ the following notation.

Notation 42. *Let* $f : A \rightarrow \mathbb{P}(B)$ *be a function from* A *to subsets of* B. *We then put*

$$f^{\dagger} = \{g : A \rightarrow B \mid \forall a\cdot g(a) \in f(a)\}$$

The denotational definitions are collected in the next definition.

Definition 43.

1. $P_3 = \Sigma \rightarrow \mathbb{P}_{nc}(Q_3)$, $Q_3 = (\Sigma \times (\Sigma \rightarrow Q_3)) \cup \{\epsilon\}$. *We shall use* X *to range over* $\mathbb{P}_{nc}(Q_3)$, *and* ξ *to range over* $\Sigma \rightarrow Q_3$.

2. *Let* $(\phi\in)\,\mathbb{P}_3 = P_3 \times P_3 \to P_3$, *and let* $\Omega_\circ$, $\Omega_\| : \mathbb{P}_3 \to \mathbb{P}_3$ *be given as follows*

$$\begin{aligned}
\Omega_\circ(\phi)(p_1,p_2) &= \lambda\sigma\cdot\tilde{\phi}(p_1(\sigma))(p_2)\\
\tilde{\phi}(X)(p) &= \bigcup\{\hat{\phi}(q)(p)\mid q\in X\}\\
\hat{\phi}(\epsilon)(p) &= \{\langle\sigma,\xi\rangle\mid \sigma\in\Sigma\ \wedge\ \xi\in p^\dagger\}\\
\hat{\phi}(\langle\sigma,\xi\rangle)(p) &= \{\langle\sigma,\bar{\xi}\rangle\mid \bar{\xi}\in\phi(\lambda\bar{\sigma}\cdot\{\xi(\bar{\sigma})\})(p)^\dagger\}\\
\Omega_\|(\phi)(p_1,p_2)(\sigma) &= \Omega_\circ(\phi)(p_1,p_2)(\sigma)\cup\Omega_\circ(\phi)(p_2,p_1)(\sigma)
\end{aligned}$$

Let $\circ = \mathrm{fix}\,\Omega_\circ$, $\| = \mathrm{fix}\,\Omega_\|$.

3. *Let* $(F\in)\,N_3 = L_3 \to (P_3 \to P_3)$, *and let* $\Phi_3 : N_3 \to N_3$ *be given by*

$$\begin{aligned}
\Phi_3(F)(v:=e)(p) &= \lambda\sigma\cdot\{\langle\sigma[\alpha/v],\xi\rangle\mid\xi\in p^\dagger\}\\
&\qquad \text{where } \alpha = [\![e]\!](\sigma)\\
\Phi_3(F)(x)(p) &= \lambda\sigma\cdot\{\langle\sigma,\xi\rangle\mid\xi\in F(s)(p)^\dagger\}\\
&\qquad \text{where } (x,s)\in D\\
\Phi_3(F)(s_1\,;s_2)(p) &= \Phi_3(F)(s_1)(\Phi_3(F)(s_2)(p))\\
\Phi_3(F)(\mathbf{if}\ldots\mathbf{fi})(p) &= \lambda\sigma\cdot\mathbf{if}\ [\![b]\!](\sigma)\\
&\qquad\ \mathbf{then}\ \Phi_3(F)(s_1)(p)(\sigma)\\
&\qquad\ \mathbf{else}\ \Phi_3(F)(s_2)(p)(\sigma)\\
&\qquad \mathbf{fi}\\
\Phi_3(F)(\mathbf{new}(s))(p) &= \Phi_3(F)(s)(\lambda\sigma\cdot\{\epsilon\})\,\|\,p
\end{aligned}$$

and

$$\begin{aligned}
&\Phi_3(F)(\mathbf{begin\ int}\ v:=e\,;s\ \mathbf{end})(p)\\
&\quad = \lambda\sigma\cdot\Phi_3(F)(v:=e\,;s)(\lambda\bar{\sigma}\cdot\{\langle\bar{\sigma}[\sigma(v)/v],\xi\rangle\mid\xi\in p^\dagger\})(\sigma)
\end{aligned}$$

4. *Let* $\mathcal{D}_3 = \mathrm{fix}\,\Phi_3$, *and let* $\mathcal{E}_3 : Par_3 \to P_3$ *be obtained from* $\mathcal{D}_3$ *similar to the definitions of* $\mathcal{E}_1$ *for* L_1 *(where* $\mathcal{E}_3[\![E]\!] = \lambda\sigma\cdot\{\epsilon\}$*).*

We finally relate $\mathcal{O}_3$ and $\mathcal{E}_3$ in the usual manner through the abstraction function abs_3.

Definition 44.

1. *Let* $(\pi\in)\,QS_3 = Q_3 \to S_3$, *and let* $\Delta_3' : QS_3 \to QS_3$ *be given by*

$$\begin{aligned}
\Delta_3'(\pi)(\epsilon) &= \epsilon\\
\Delta_3'(\pi)(\langle\sigma,\xi\rangle) &= \langle\sigma,\pi(\xi(\sigma))\rangle
\end{aligned}$$

2. *Let* $abs_3' = \text{fix}\,\Delta_3'$, *and let* $abs_3 : P_3 \to P_3$ *be given as*

$$abs_3(p) \;=\; \lambda\sigma \cdot \begin{cases} \{abs_3'(q) \mid q \in p(\sigma)\} & \text{if this set is non-empty} \\ \{\delta\} & \text{otherwise} \end{cases}$$

We have the, now familiar, result.

Theorem 45. $\mathcal{O}_3 = abs_3 \circ \mathcal{E}_3$

We do not know whether $\mathcal{E}_3$ is fully abstract with respect to $\mathcal{O}_3$.

L_4: a non-uniform language with parallel composition and communication

The language L_4 is an extension of L_2 in that now (CSP-like) communication over *channels* $c(\in \mathit{Chan})$ is added. A send statement has the form $c\,!\,e$, a receive statement has the form $c\,?\,v$, and synchronized execution of these (in two parallel components) amounts to the execution of the assignment $v := e$.

The syntax for L_4 is given in the next definition.

Definition 46.

1. $(s\in)\,L_4$ *has as syntax*

$$s \;::=\; v := e \mid x \mid s_1 \,;\, s_2 \mid \textbf{if}\; b\; \textbf{then}\; s_1\; \textbf{else}\; s_2\; \textbf{fi} \mid s_1 \parallel s_2 \mid c\,?\,v \mid c\,!\,e$$

2. *Declarations and programs are as usual.*

The operational semantics for L_4 employs the sets

$$\begin{aligned}(\gamma\in)\,\Gamma \;&=\; \{c\,?\,v \mid c \in \mathit{Chan} \,\wedge\, v \in \mathit{Ivar}\} \cup \{c\,!\,\alpha \mid c \in \mathit{Chan} \,\wedge\, \alpha \in V\} \\ (\eta\in)\,H \;&=\; \Sigma \cup \Gamma\end{aligned}$$

Transitions are of the form $\langle s, \sigma\rangle \to_4 \langle r, \eta\rangle$, with $r \in L_4^+ = L_4 \cup \{E\}$. The transition system T_4 is given in the following definition.

Definition 47.

1.

$$\begin{aligned}\langle v := e, \sigma\rangle &\to_4 \langle E, \sigma[\alpha/v]\rangle \quad \alpha \textit{ as usual} \\ \langle c\,?\,v, \sigma\rangle &\to_4 \langle E, c\,?\,v\rangle \\ \langle c\,!\,e, \sigma\rangle &\to_4 \langle E, c\,!\,\alpha\rangle \quad\;\; \alpha \textit{ as usual}\end{aligned}$$

2. *The rules for* x, ;, **if** ... **fi**, $\|$ *are as those in* T_2 *(with* $\rightarrow_4$ *replacing* $\rightarrow_2$*). For* $\|$ *we have in addition the following rule.*

3. *If* $\langle s_1, \sigma\rangle \rightarrow_4 \langle r', c\,?\,v\rangle$ *and* $\langle s_2, \sigma\rangle \rightarrow_4 \langle r'', c\,!\,\alpha\rangle$ *then* $\langle s_1 \parallel s_2, \alpha\rangle \rightarrow_4 \langle r' \parallel r'', \sigma[\alpha/v]\rangle$*. (We assume the usual convention that* $E\|r = r\|E = r$*.)*

The operational domains and semantics are given in the next definition.

Definition 48.

1. $R_4 = \Sigma \rightarrow \mathbb{P}_{nc}(S_4)$, $S_4 = (\Sigma \times S_4) \cup \{\delta, \epsilon\}$
2. *Let* $(F\in)\, M_4 = L_4^+ \rightarrow R_4$*, and let* $\Psi_4 : M_4 \rightarrow M_4$ *be given by*

$$\begin{aligned}\Psi_4(F)(E) &= \lambda\sigma \cdot \{\epsilon\}\\ \Psi_4(F)(s) &= \lambda\sigma \cdot \begin{cases}\{\langle s', u\rangle \mid \langle s, \sigma\rangle \rightarrow_4 \langle r, \sigma'\rangle \ \wedge\ u \in F(r')(s')\},\\ \qquad \text{if this set is non-empty}\\ \{\delta\} \quad \text{otherwise}\end{cases}\end{aligned}$$

3. $\mathcal{O}_4 = \text{fix}\,\Psi_4$.

Remark. Note that, in the definition of $\Psi_4(F)(s)(\sigma)$, no contributions are made by steps $\langle s, \sigma\rangle \rightarrow_4 \langle r, \gamma\rangle$.

Once more $\mathcal{O}_4$ is not compositional. The denotational definitions assume a domain P_4 which combines the BT structure of P_1 with the non-uniform structure of P_3.

Definition 49.

1. $P_4 = (\Sigma \rightarrow \mathbb{P}_{co}(Q_4)) \cup \{\{\epsilon\}\}$, $Q_4 = (\Sigma \cup \Gamma) \times P_4$
2. *Let* X *range over* $\mathbb{P}_{co}(Q_4)$*. Let* $(\phi\in)\, \mathbb{P}_4 = P_4 \times P_4 \rightarrow P_4$*, and let* $\Omega_\circ$, $\Omega_\| : \mathbb{P}_4 \rightarrow \mathbb{P}_4$ *be given by*

$$\begin{aligned}\Omega_\circ(\phi)(p_1, p_2) &= p_2 \quad \text{if } p_1 = \{\epsilon\}\\ &= \lambda\sigma \cdot \hat{\phi}(p_1(\sigma))(p_2) \quad \text{if } p_1 \neq \{\epsilon\}\\ \hat{\phi}(X)(p) &= \{\tilde{\phi}(q)(p) \mid q \in X\}\\ \tilde{\phi}(\langle \eta, p'\rangle)(p) &= \langle \eta, \phi(p')(p)\rangle\\ \Omega_\|(\phi)(p_1, p_2) &= \lambda\sigma \cdot \begin{pmatrix}\Omega_\circ(\phi)(p_1, p_2)(\sigma)\\ \cup\, \Omega_\circ(\phi)(p_2, p_1)(\sigma)\\ \cup\, \Omega_|(\phi)(p_1, p_2)(\sigma)\end{pmatrix}\end{aligned}$$

where

$$\Omega_{|}(\phi)(p_1,p_2)(\sigma) = \lambda\sigma \cdot \left\{ \begin{array}{l} \langle \sigma[\alpha/v], \phi(p')(p'')\rangle \\ \quad \mid \langle c\,?\,v, p'\rangle \in p_1 \ \wedge\ \langle c\,!\,\alpha, p''\rangle \in p_2 \text{ or vice versa} \end{array} \right\}$$

$\circ = \text{fix}\,\Omega_\circ$, $\| = \text{fix}\,\Omega_{\|}$.

3. *Let* $(F\in)\, N_4 = L_4 \to P_4$, *and let* $\Phi_4 : N_4 \to N_4$ *be given by*

$$\begin{array}{rcll} \Phi_4(F)(v := e) &=& \lambda\sigma \cdot \{\langle \sigma[\alpha/v], \{\epsilon\}\rangle\} & \alpha \text{ as usual} \\ \Phi_4(F)(c\,?\,v) &=& \lambda\sigma \cdot \{\langle c\,?\,v, \{\epsilon\}\rangle\} & \\ \Phi_4(F)(c\,!\,e) &=& \lambda\sigma \cdot \{\langle c\,!\,\alpha, \{\epsilon\}\rangle\} & \alpha \text{ as usual} \end{array}$$

$s \equiv s_1\,;\,s_2,\ s_1 \,\|\, s_2,$ **if** ... **fi**: *omitted*

$$\Phi_4(F)(x) \;=\; \lambda\sigma \cdot \{\langle \sigma, F(s)\rangle\} \quad (x,s) \in D$$

4. *Let* $\mathcal{D}_4 = \text{fix}\,\Phi_4$.

We conclude with the abstraction mapping between $\mathcal{O}_4$ and $\mathcal{D}_4$.

Definition 50. *Let* $(\pi\in)\, PR_4 = P_4 \to R_4$, *and let* $\Delta_4 : PR_4 \to PR_4$ *be defined as follows:*

$$\Delta_4(\pi)(\{\epsilon\}) \;=\; \lambda\sigma \cdot \{\epsilon\}$$

and, for $p \neq \{\epsilon\}$,

$$\begin{array}{rcl} \Delta_4(\pi)(p) &=& \lambda\sigma \cdot \left\{ \begin{array}{ll} \bigcup\{\tilde{\pi}(q) \mid q \in p(\sigma)\} & \text{if this set is non-empty} \\ \{\delta\} & \text{otherwise} \end{array} \right. \\ \tilde{\pi}(\langle \sigma, p\rangle) &=& \{\langle \sigma, q\rangle \mid q \in \pi(p)(\sigma)\} \\ \tilde{\pi}(\langle \gamma, p\rangle) &=& \emptyset \end{array}$$

Let $abs_4 = \text{fix}\,\Delta_4$.

We have (for $\hat{\mathcal{D}}_4$ similar to $\hat{\mathcal{D}}_2$) the following theorem.

Theorem 51. $\mathcal{O}_4 = abs_4 \circ \hat{\mathcal{D}}_4$

As to the question of full abstractness, since $\mathcal{D}_1$ is (probably) not fully abstract with respect to $\mathcal{O}_1$ (cf. the discussion for L_1), there is no reason to expect $\mathcal{D}_4$ to be fully abstract with respect to $\mathcal{O}_4$. (In (Horita *et al.* 1990) full abstractness with respect to a non-uniform version of the failure set model is shown.)

Conclusion

We conclude with a table which surveys the domain equations encountered in Section 5.3.

	Operational	Denotational
Uniform		
L_0	$R_0 = \mathbb{P}_{nc}(S_0)$	$P_0 = \mathbb{P}_{co}(Q_0) \cup \{\{\epsilon\}\}$
	$S_0 = (A \times S_0) \cup \{\delta, \epsilon\}$	$Q_0 = A \times P_0$
L_1	$R_1 = \mathbb{P}_{nc}(S_1)$	$P_1 = \mathbb{P}_{co}(Q_1) \cup \{\{\epsilon\}\}$
	$S_1 = (B \times S_1) \cup \{\delta, \epsilon\}$	$Q_1 = (B \cup C) \times P_1$
Non-uniform		
L_2	$R_2 = \Sigma \to \mathbb{P}_{nc}(S_2)$	$P_2 = \mathbb{P}_{nc}(Q_2)$
	$S_2 = (\Sigma \times S_2) \cup \{\delta, \epsilon\}$	$Q_2 = (\Sigma \to (\Sigma \times Q_2)) \cup \{\epsilon\}$
L_3	$R_3 = R_2$	$P_3 = \Sigma \to \mathbb{P}_{nc}(Q_3)$
	$S_3 = S_2$	$Q_3 = (\Sigma \times (\Sigma \to Q_3)) \cup \{\epsilon\}$
L_4	$R_4 = R_2$	$P_4 = (\Sigma \to \mathbb{P}_{co}(Q_4)) \cup \{\{\epsilon\}\}$
	$S_4 = S_2$	$Q_4 = (\Sigma \cup \Gamma) \times P_4$

5.4 Labelled transition systems and bisimulation

In this section we shall use the domain P_0 of the previous section to give a general model for *bisimulation* equivalence (Park 1981), a well-known notion in the theory of concurrency. (The same result holds for P_1. For the domains used for the non-uniform languages some further study is still needed.) It is based on the basic notion of a *labelled transition system* (LTS).

Definition 52. (LTS) *A* labelled transition system *is a triple* $\mathcal{A} = (S, L, \to)$ *consisting of a set of* states S, *a set of* labels L, *and a* transition relation $\to\ \subseteq S \times L \times S$. *We shall write* $s \xrightarrow{a} s'$ *for* $(s, a, s') \in \to$. *Following the approach of the previous section, we assume the presence of a special element* $E \in S$ *that syntactically denotes successful termination. An* LTS *is called* finitely branching *if for all* $s \in S$, $\{(a, s') \mid s \xrightarrow{a} s'\}$ *is finite.*

Every LTS induces a *bisimulation* equivalence.

Definition 53. *Let* $\mathcal{A} = (S, L, \to)$ *be an* LTS. *A relation* $R \subseteq S \times S$ *is called a* (strong) bisimulation *if it satisfies for all* $s, t \in S$ *and* $a \in A$:

$$(s\,R\,t \ \wedge\ s \xrightarrow{a} s') \ \Rightarrow\ \exists t' \in S \cdot t \xrightarrow{a} t' \ \wedge\ s'\,R\,t'$$

and

$$(s\,R\,t \ \wedge\ t \xrightarrow{a} t') \ \Rightarrow\ \exists s' \in S \cdot s \xrightarrow{a} s' \ \wedge\ s'\,R\,t'$$

We require that $E\ R\ s$ or $s\ R\ E$ implies $s = E$. Two states are bisimilar in $\mathcal{A}$, notation $s \leftrightarrow t$, if there exists a bisimulation relation R with $s\ R\ t$. (Note that bisimilarity is an equivalence relation on states.)

Next we define, for every LTS $\mathcal{A}$, a model assigning to every state a process in P_0.

Definition 54. *Let $\mathcal{A} = (S, L, \rightarrow)$ be a finitely branching* LTS. *Here we have taken for the set of labels the alphabet A of elementary actions used in the definition of P_0. We define a model $\mathcal{M}_{\mathcal{A}} : S \rightarrow P_0$ by*

$$\mathcal{M}_{\mathcal{A}}[\![s]\!] \;=\; \{\langle a, \mathcal{M}_{\mathcal{A}}[\![s']\!]\rangle \mid s \xrightarrow{a} s'\}$$

if $s \neq E$, and by $\mathcal{M}_{\mathcal{A}}[\![E]\!] = \{\epsilon\}$.

We can justify this recursive definition by taking $\mathcal{M}_{\mathcal{A}}$ as the unique fixed point (Banach's theorem) of a contraction $\Phi : (S \rightarrow^1 P) \rightarrow (S \rightarrow^1 P)$ defined by

$$\Phi(F)(s) \;=\; \{\langle a, F(s')\rangle \mid s \xrightarrow{a} s'\}$$

if $s \neq E$, and by $\Phi(F)(E) = \{\epsilon\}$. The fact that Φ is a contraction can be easily proved. The compactness of the set $\Phi(F)(s)$ is an immediate consequence of the fact that $\mathcal{A}$ is finitely branching.

As an example we can take in the above definition the LTS of Definition 15. We then obtain the function $\mathcal{I}$ given in the proof of Theorem 22.

This model is of interest because it assigns the same meaning to bisimilar states. This we prove next.

Theorem 55. *Let $\leftrightarrow \subseteq S \times S$ denote the bisimilarity relation induced by the labelled transition system $\mathcal{A} = (S, A, \rightarrow)$. Then*

$$\forall s, t \in S \cdot s \leftrightarrow t \iff \mathcal{M}_{\mathcal{A}}[\![s]\!] = \mathcal{M}_{\mathcal{A}}[\![t]\!]$$

Proof. Let $s, t \in S$.

$\Leftarrow$ Suppose $\mathcal{M}_{\mathcal{A}}[\![s]\!] = \mathcal{M}_{\mathcal{A}}[\![t]\!]$. We define a relation $\equiv : S \times S$ by

$$s' \equiv t' \iff \mathcal{M}_{\mathcal{A}}[\![s']\!] = \mathcal{M}_{\mathcal{A}}[\![t']\!]$$

From the definition of $\mathcal{M}_{\mathcal{A}}$ it is straightforward that $\equiv$ is a bisimulation relation on S. Suppose $s' \equiv t'$ and $s' \xrightarrow{a} s''$. Then $\langle a, \mathcal{M}_{\mathcal{A}}[\![s'']\!]\rangle \in \mathcal{M}_{\mathcal{A}}[\![s']\!] = \mathcal{M}_{\mathcal{A}}[\![t']\!]$; thus there exists $t'' \in S$ with $t' \xrightarrow{a} t''$ and $\mathcal{M}_{\mathcal{A}}[\![s'']\!] = \mathcal{M}_{\mathcal{A}}[\![t'']\!]$, that is, $s'' \equiv t''$. Symmetrically, the second property of a bisimulation relation holds. From the hypothesis we have $s \equiv t$. Thus we have $s \leftrightarrow t$.

$\Rightarrow$ Let $R \subseteq S \times S$ be a bisimulation relation with $s\, R\, t$. We define

$$\epsilon = \sup_{s',t' \in S} \{d(\mathcal{M}_{\mathcal{A}}[\![s']\!], \mathcal{M}_{\mathcal{A}}[\![t']\!]) \mid s'\, R\, t'\}$$

We prove that $\epsilon = 0$, from which $\mathcal{M}_{\mathcal{A}}[\![s]\!] = \mathcal{M}_{\mathcal{A}}[\![t]\!]$ follows, by showing that $\epsilon \leqslant \epsilon/2$. We prove for all s', t' with $s'\, R\, t'$ that $d(\mathcal{M}_{\mathcal{A}}[\![s']\!], \mathcal{M}_{\mathcal{A}}[\![t']\!]) \leqslant \epsilon/2$. Consider $s', t' \in S$ with $s'\, R\, t'$. From the definition of the Hausdorff metric on P it follows that it suffices to show

$$d(x, \mathcal{M}_{\mathcal{A}}[\![t']\!]) \leqslant \epsilon/2 \text{ and } d(y, \mathcal{M}_{\mathcal{A}}[\![s']\!]) \leqslant \epsilon/2$$

for all $x \in \mathcal{M}_{\mathcal{A}}[\![s']\!]$ and $y \in \mathcal{M}_{\mathcal{A}}[\![t']\!]$. We shall only show the first inequality; the second is similar. Consider $\langle a, \mathcal{M}_{\mathcal{A}}[\![s'']\!]\rangle$ in $\mathcal{M}_{\mathcal{A}}[\![s']\!]$ with $s' \xrightarrow{a} s''$. (The case that $\mathcal{M}_{\mathcal{A}}[\![s']\!] = \{\epsilon\}$ is trivial.) Because $s'\, R\, t'$ and $s' \xrightarrow{a} s''$ there exists $t'' \in S$ with $t' \xrightarrow{a} t''$ and $s''\, R\, t''$. Therefore

$$\begin{aligned}
& d(\langle a, \mathcal{M}_{\mathcal{A}}[\![s'']\!]\rangle, \mathcal{M}_{\mathcal{A}}[\![t']\!]) \\
&= d(\langle a, \mathcal{M}_{\mathcal{A}}[\![s'']\!]\rangle, \{\langle \bar{a}, \mathcal{M}_{\mathcal{A}}[\![\bar{t}]\!]\rangle \mid t' \xrightarrow{\bar{a}} \bar{t}\}) \\
&\leqslant [\text{we have: } d(x, Y) = \inf\{d(x,y) \mid y \in Y\}] \\
&\quad\; d(\langle a, \mathcal{M}_{\mathcal{A}}[\![s'']\!]\rangle, \langle a, \mathcal{M}_{\mathcal{A}}[\![t'']\!]\rangle) \\
&= d(\mathcal{M}_{\mathcal{A}}[\![s'']\!], \mathcal{M}_{\mathcal{A}}[\![t'']\!])/2 \\
&\leqslant [\text{because } s''\, R\, t''] \\
&\quad\; \epsilon/2
\end{aligned}$$

□

The proof above makes convenient use of the Hausdorff metric on P. It was first given in (Rutten 1989). An alternative proof, using so-called non-well-founded sets, can be found in (van Glabbeek and Rutten 1989, Rutten 1990b).

References

America, P., and de Bakker, J. W. (1988). Designing equivalent semantic models for process creation, *Theoretical Computer Science*, Vol. 60, pp. 109–176.

America, P., and Rutten, J. J. M. M. (1989a). Solving reflexive domain equations in a category of complete metric spaces, *Journal of Computer and System Sciences*, Vol. 35, No. 3, pp. 343–375.

America, P., and Rutten, J. J. M. M. (1989b). 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, pp. 1–49.

America, P., de Bakker, J. W., Kok, J. N., and Rutten, J. J. M. M. (1989). Denotational semantics of a parallel object-oriented language, *Information and Computation*, Vol. 83, No. 2, pp. 152–205.

de Bakker, J. W. (1980). *Mathematical Theory of Program Correctness*, Prentice Hall International.

de Bakker, J. W. (1988). *Comparative Semantics for Flow of Control in Logic Programming without Logic*, CWI Report CS-R8840, to appear in *Information and Computation*.

de Bakker, J. W. (1989). Designing concurrency semantics, in: G. X. Rytter (Ed.), *Proc. 11th World Computer Congress*, North-Holland, pp. 591–598.

de Bakker, J. W., and Kok, J. N. (1988). Uniform abstraction, atomicity and contractions in the comparative semantics of Concurrent Prolog, in: *Proc. Int. Conference on Fifth Generation Computer Systems, Institute for New Generation Computer Technology*, pp. 347–355.

de Bakker, J. W. and Kok, J. N. (1990). Comparative semantics for Concurrent Prolog, *Theoretical Computer Science*, to appear.

de Bakker, J. W., and Meyer, J.-J. Ch. (1988). Metric semantics for concurrency, *BIT 28*, pp. 504–529.

de Bakker, J. W., and Zucker, J. I. (1982). Processes and the denotational semantics of concurrency, *Information and Control*, Vol. 54, pp. 70–120.

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

de Bakker, J. W., Meyer, J.-J. Ch., and Olderog, E.-R. (1987). Infinite streams and finite observations in the semantics of uniform concurrency, *Theoretical Computer Science*, Vol. 49, pp. 87–112.

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

de Bakker, J. W., Kok, J. N., Meyer, J.-J. Ch., Olderog, E.-R., and Zucker, J. I. (1986). Contrasting themes in the semantics of imperative concurrency, in: J. W. de Bakker *et al.* (Eds.), *Current Trends in Concurrency: Overviews and Tutorials*, Lecture Notes in Computer Science, Vol. 224, Springer-Verlag, pp. 51–121.

Bergstra, J. A., and Klop, J. W. (1989). Bisimulation semantics, in: J. W. de Bakker *et al.* (Eds.), *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Lecture Notes in Computer Science, Vol. 354, Springer-Verlag, pp. 50–122.

Bergstra, J. A., Klop, J. W., and Olderog, E.-R. (1988). Readies and failures in the algebra of communicating processes, *SIAM J. of Computing*, Vol. 17, No. 6, pp. 1134–1177.

Brookes, S. D., Hoare, C. A. R., and Roscoe, A. W. (1984). A theory of communicating sequential processes, *J. ACM*, Vol 31, pp. 499–560.

Dugundji, J. (1966). *Topology*, Allen and Bacon, Rockleigh, N. J.

Enkelking, R. (1977). *General Topology*, Polish Scientific Publishers.

van Glabbeek, R. J., and Rutten, J. J. M. M. (1989). The processes of de Bakker and Zucker represent bisimulation equivalence classes, in: *J. W. de Bakker, 25 jaar semantiek*, Centre for Mathematics and Computer Science, Amsterdam.

Hennessy, M., and Plotkin, G. D. (1979). Full abstraction for a simple parallel programming language, in: J. Bečvář (Ed.), *Proceedings 8th MFCS*, Lecture Notes in Computer Science, Vol. 74, Springer-Verlag, pp. 108–120.

Horita, E., de Bakker, J. W., and Rutten, J. J. M. M. (1990). *Fully abstract denotational models for non-uniform concurrent languages*, Report CS-R90?, Centre for Mathematics and Computer Science, Amsterdam.

Kok, J. N., and Rutten, J. J. M. M. (1988). Contractions in comparing concurrency semantics, in: T. Lepisto and A. Salomaa (Eds.), *Proc. 15th ICALP*, Lecture Notes in Computer Science, Vol. 317, Springer-Verlag, pp. 317–332. (To appear in *Theoretical Computer Science.*)

Mac Lane, S. (1971). *Categories for the Working Mathematician*, Springer-Verlag.

Majster-Cederbaum, M. E. (1988). The contraction property is sufficient to guarantee the uniqueness of fixed points of endofunctors in a category of complete metric spaces, *Information Processing Letters*, Vol. 29, pp. 277–281.

Majster-Cederbaum, M. E. (1989). On the uniqueness of fixed points of endofunctors in a category of complete metric spaces, *Information Processing Letters*, Vol. 33, pp. 15–20.

Majster-Cederbaum, M. E., and Zetzsche, F. (199?). Towards a foundation for semantics in complete metric spaces, *Information and Computation*, to appear.

Michael, E. (1951). Topologies on spaces of subsets, *Trans. AMS*, Vol. 71, pp. 152–182.

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

Park, D. M. R. (1981). Concurrency and automata on infinite sequences, in: *Proc. 5th GI Conference*, Lecture Notes in Computer Science, Vol. 104, Springer-Verlag, pp. 15–32.

Plotkin, G. D. (1981). *A structural approach to operational semantics*, Report DAIMI FN-19, Computer Science Department, Aarhus University.

Plotkin, G. D. (1983). An operational semantics for CSP, in: D. Björner

(Ed.), *Formal Description of Programming Concepts II*, North-Holland, pp. 199-223.

Rutten, J. J. M. M. (1989). Correctness and full abstraction of metric semantics for concurrency, in: J. W. de Bakker *et al.* (Eds.), *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Lecture Notes in Computer Science, Vol. 354, Springer-Verlag, pp. 628–659.

Rutten, J. J. M. M. (1989). *Deriving Denotational Models for Bisimulation from Transition System Specifications*, CWI Report CS-R8955, Amsterdam. (To appear in the proceedings of IFIP TC2 Working Conference, Israel, 1990.)

Rutten, J. J. M. M. (1990a). Semantic correctness for a parallel object-oriented language, *SIAM J. of Computing*, Vol. 19, No. 2, pp. 341–383.

Rutten, J. J. M. M. (1990b). *Nonwellfounded sets and programming language semantics*, CWI Report CS-R90?, Amsterdam.