CONTRASTING THEMES

IN THE

SEMANTICS OF IMPERATIVE CONCURRENCY

J.W. de Bakker [1)]
Centre for Mathematics and Computer Science
Kruislaan 413
1098 SJ Amsterdam
The Netherlands

J.N. Kok [2)]
Centre for Mathematics and Computer Science
Kruislaan 413
1098 SJ Amsterdam
The Netherlands

J.-J.Ch. Meyer
Subfaculteit Wiskunde en Informatica
Vrije Universiteit
De Boelelaan 1081
1081 HV Amsterdam
The Netherlands

E.-R. Olderog
Institut für Informatik
Christian-Albrechts-Universität
Olshausenstrasse 40-60
2300 Kiel 1
Federal Republic of Germany

J.I. Zucker [3)]
Department of Computer Science
University of Buffalo (SUNY)
226 Bell Hall
Buffalo, New York 14260
USA

1)The research of J.W. de Bakker was partially supported by ESPRIT Project 415: Parallel Architectures and Languages.

2)The research of J.N. Kok was supported by the Netherlands Organization for the Advancement of Pure Research (Z.W.O.), grant 125-20-04.

3)The research of J.I. Zucker was supported by the National Science Foundation under grant no. MCS-8010728.

ABSTRACT

A survey is given of work performed by the authors in recent years concerning the semantics of imperative concurrency. Four sample languages are presented for which a number of operational and denotational semantic models are developed. All languages have parallel execution through interleaving, and the last three have as well a form of synchronization. Three languages are uniform, i.e., they have uninterpreted elementary actions; the fourth is nonuniform and has assignment, tests and value-passing communication. The operational models build on Hennessy-Plotkin transition systems; as denotational structures both metric spaces and cpo domains are employed. Two forms of nondeterminacy are distinguished, viz. the local and global variety. As associated model-theoretic distinction that of linear time versus branching time is investigated. In the former we use streams, i.e. finite or infinite sequences of actions; in the latter the (metrically based) notion of process is introduced. We furthermore study a model with only finite observations. Ready sets also appear, used as technical tool to compare various semantics. Altogether, ten models for the four languages are described, and precise statements on (the majority of) their interrelationships are made . The paper supplies no proofs; for these references to technical papers by the authors are provided.

Contents:

1980 Mathematics Subject Classification: 68B10, 68C01.
1982 CR Categories: D.3.1, F.3.2, F.3.3.

1. INTRODUCTION

We present a study of a number of contrasting themes in the semantics of imperative concurrency. Special attention will be paid to the mutual connections between on the one hand fundamental notions in concurrency, on the other hand various mathematical structures and associated tools used in building semantic models for these notions. Altogether, a large assortment of such models will be displayed, and precise statements about their relationships will be made. The paper surveys earlier work of the authors on these topics and discusses which sources have been instrumental in its development.

Our paper concentrates on issues in *imperative* concurrency. Specifically, we shall discuss *parallel* execution through interleaving (shuffle or merge) of elementary actions, *synchronization* and *communication*, and (an elementary form of) *message passing*. These notions fit into the tradition of concurrency concepts as initiated in the sixties by Dijkstra [Dij] with his *cobegin-coend* statements, and continued in the seventies with the influential contributions by Hoare on CSP [Ho] and Milner on CCS [Mi2].

Our first contrast is that between imperative and applicative concurrency. Imperative concurrency is characterized by an explicit operator for parallel composition on top of the usual imperative constructs such as elementary action and sequential composition. In applicative concurrency the phenomenon of parallel execution usually appears within a functional context where concurrency is implicit in the way in which arguments of functions are evaluated. (Blends of these two styles in concurrency can be found as well, see e.g. ref. [ABKR1,2].) In order to keep the size of our paper within reasonable bounds, we concentrate solely on imperative notions. Of course, many of the structures and tools described below have more or less direct bearing upon modelling of applicative concepts as well. A few references exemplifying this will be given throughout the paper.

The second contrast concerns *uniform* versus *non-uniform* languages. Characteristic for the former is that the elementary actions of the language are left atomic: no specification for these actions in concrete terms of e.g., assignment or tests, or, more abstractly, in terms of state transforming functions is provided. In other words, 'uniform' refers to an approach at the *schematic* level. As a consequence, the semantic models for this case have much of the flavor of the objects studied in formal language theory. (Here we take formal languages in a wide sense: finite and infinite words *and* tree-like structures are included. The essential common element is the consideration of structured objects over a given alphabet of uninterpreted symbols.)

The nonuniform case extends the uniform one in that a specification of the elementary actions is now supplied. The specific variety we discuss in our paper (section 7) is in fact quite simple: only assignments, tests and send/receive actions are introduced. Further examples which we shall not deal with below are test-and-set, (remote) procedure declarations and calls, critical sections, the ADA rendez-vous, to mention only a few of the more familiar ones. The important difference with the uniform situation is that meanings of programs are no longer reminiscent of formal languages, but are instead primarily of a functional nature, transforming in some way states to (sets of) states. It should be emphasized, however, that on closer scrutiny the objects are more complicated than the simple state transforming functions as abounding in sequential

semantics. In particular, a key role is now played by structures preserving in some way the *history* of the computation.

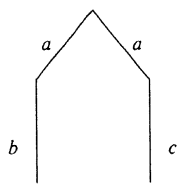
The third theme -which may be seen as the dominant one throughout the present investigation- is the familiar distinction between operational and denotational semantics. Rather than go into a prolonged discussion at this stage of the respective characteristics and merits of these approaches, we shall let the methods as exhibited in the treatment below speak for themselves. Altogether, we shall provide ten semantic definitions for four languages, viz. four operational and six denotational ones. Furthermore, we shall supply detailed information on (the majority of) the respective relationships which hold between these semantic definitions.

Our operational definition method is based on the transition systems technique of Hennessy and Plotkin [HP] and Plotkin [Pl3,Pl4]. (For applications of these in soundness/completeness studies in proof theory see [Ap1,Ap2].) However, we have introduced some important variations: (i) the inclusion throughout of infinite computations; (ii) the inclusion of recursion rather than iteration -roughly in the sense in which context-free extends regular in formal language theory-, and (iii) the coverage of both uniform and nonuniform languages (the papers cited all address nonuniform concepts). Our operational treatment of the uniform case may in fact also be seen as an extension, with shuffle and synchronization, of the algebraic grammars generating languages with finite and infinite words as studied by Nivat (e.g. [Ni1]); later work by Nivat (e.g. [Ni2]) introduces an approach to synchronization which is different from the one studied in our paper.

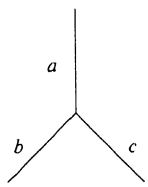
Our fourth contrast pertains to the diversity of mathematical structures which underly the denotational models. We shall primarily use *metric* structures; *order-theoretic* domains play (an important but) secondary role. This is somewhat different from the general situation in denotational semantics, where order-theoretic domains are the preponderant tools. However, the metric approach is particularly felicitous for problems where histories and computational traces of some kind are essential, since for (finite and infinite) sequences a very natural *distance* and associated metric(s) can be defined and exploited. Our interest in metric techniques was kindled by Nivat's lectures on infinite words, infinite trees and infinite computations reported in [NI1]. Detailed references to subsequent applications we have made of his work follow below.

The next contrast concerns a subdivision within the order-theoretic models. In a setting where nondeterminacy is present -either implicity by choices in interleavings or explicitly by language constructs to be discussed in a moment-, it is natural to deal with *sets* of possible outcomes of executing program constructs. Following the general denotational semantic methodology, we have an interest in sequences of such sets as improving approximations to certain *limits*. (Recall that in denotational semantics one has to make precise notions like limits, *continuity* of operators and *fixed points* to deal with recursion.) These approximations require a definition of an ordering on sets. Here again, we have a number of possibilities. Sometimes it suffices to work with what may be called a naive order, viz. that of (reverse) set inclusion. On other occasions, one has already an order available on the elements of the relevant domain, and now wants to define an order on sets in terms of the order on elements. Traditionally, three such definitions are distinguished. We shall concentrate on one of them (the so-called Smyth order), but make a few comparisons with other approaches as well.

A further major theme in our study is the distinction between *local* and *global* nondeterminacy (sometimes also called internal versus external nondeterminism). For s_1, s_2 two statements, we shall distinguish between the choices $s_1 \bigcup s_2$ and $s_1 + s_2$, the first denoting local and the second global nondeterminacy. In the first variety, if one of the s_i is an action which wants to communicate it may be chosen irrespectively of the presence of a communication partner for this action in some parallel component. In the second (global) approach, an action which wants to communicate is chosen only if it has been settled (by inspection of the global or external environment) that a suitable partner ready to communicate is indeed available. (This brief explanation is elaborated in particular in section 6.1.) The choice for one of the two varieties of nondeterminism has far-reaching consequences for the semantic models. Operationally, the transition system dealing with global nondeterminacy is substantially more complicated than that for local nondeterminacy. Denotationally, an even more drastic step is taken: For local nondeterminacy, a semantic model based on sets of (finite or infinite) sequences of actions, also called *streams*, suffices. For global nondeterminacy, a more elaborated semantic model is needed. We shall employ here a model where the complete history of possible choices at any time during the computation is recorded in certain tree-like structures, also called *processes*. Following the terminology of temporal logic (see e.g. [Pn]), the stream semantics is also called a 'linear time' model, and the process semantics a '(full) branching time' model. For example, in a linear time model, we shall encounter the set $\{ab,ac\}$ as meaning for both the statements $(a;b) \bigcup (a;c)$ and $a;(b \bigcup c)$. In the branching time models $(a;b) + (a;c)$ and $a;(b + c)$ will obtain as meanings the trees



and



respectively. A rigorous development of the branching time model requires the solving of domain equations in the style of Plotkin [Pl1] or Scott [Sc]. We shall apply metric tools for this purpose as first described in [BZ2].

The full branching time model has advantages in its generality and extensibility to nonumiform languages. For a proper treatment of global nondeterminacy, models which are 'in between' the

linear time and branching time framework can also be used. This has been investigated in detail by many authors. We mention in particular [BHR], [OH2], [BKO], and the survey [Pn] (which also contains further references). In fact, a model with so-called *ready sets*, one of the variants encountered there, plays a technical role in establishing the relationship between operational and denotational semantics for our language with global nondeterminacy.
The next theme on our list refers to the contrast between models incorporating infinite objects versus models which involve only finite objects. Specifically, we shall contrast -for a language with local nondeterminacy only- a model which uses finite and infinite streams, and a model which uses only finite sequences of so-called 'observations'. For both models, we impose certain restrictions on the sets which are possible outcomes of a computation. Both are then endowed with a suitable cpo structure, and a theorem establishing their isomorphism is presented.
The final theme on our list concerns a syntactic notion: We distinguish between so-called *guarded* and *unguarded* recursion. In the former, inner recursive calls are always preceded by some action, in the latter this is not necessarily the case. The guardedness requirement has important technical consequences for the denotational models. (Transition systems take both varieties in their stride.) In the metric models, guardedness is essential to obtain *contractivity* of the mappings associated with recursive constructs, a corner stone of the fixed point approach in this setting. In the order-theoretic approaches, guardedness is not a formal requirement, but we have to live with a collapse of information in the outcome of an unguarded recursive term.

Our paper is organized in seven sections. Of these, the first one is the introduction you are now reading, and section 2 collects some basic information on metric spaces, complete partially ordered sets, and languages with finite and infinite words. Sections 3 to 7 constitute the main body of the paper. There we introduce four languages:

L_0:shuffle and local nondeterminacy
(uniform, as yet no synchronization or communication)
L_1:synchronization merge and local nondeterminacy
(uniform)
L_2:synchronization merge and global nondeterminacy
(uniform)
L_3:synchronization with value passing and global nondeterminacy
(nonuniform)

Operational and metric denotational models for these languages are developed in sections 3,4,6 and 7. Section 5 describes two order-theoretic models for the language L_1. The denotational models in sections 3,4 and 5 are all linear time; in sections 6 and 7 branching time models are introduced. In each section, precise statements are made on the relationship between the semantics studied in that section. The only exception is section 7, where no more than a conjecture concerning the relationship between operational and denotational semantics for L_3 is made.
Our paper gives a unified presentation of results we have obtained in recent years. Full details of

definitions and statements of theorems are provided, but (almost) no proofs are given. Instead we supply pointers to the literature where these proofs can be found.
The primary sources for our paper are

- basic papers on the transition systems method
  [HP1, Pl3,4]
- metric denotational model, linear time
  [BBKM, BZ3]
- metric denotational model, branching time
  [BZ1,2,3,4]
- order-theoretic models, observations model
  [Me1,2, OH1,2]
- relationship between operational and metric models
  [BMOZ1,2]
- relationship between order-theoretic models, both mutual and with a metric model
  [BMO1,2,3]

The major theme of our paper being the various relationships between the models, our presentation owes most to the papers [BMOZ2] and [BMO2].

It will be clear from the above what we see as the central topics of our paper. Naturally, there are further interesting issues in the semantics of concurrency not explored in our paper. For example, we do not treat the complex of notions around hiding, abstraction and observational equivalence(s). Neither do we touch upon any of the extensive algebraic or category-theoretic approaches to concurrency. Also we only address *interleaving* models: we do not deal with models based on partial orders, trace theory in the sense of Mazurkiewicz [Ma], or Petri nets, let alone with the interconnections between interleaving and partial order methods. As starting point for the vast literature on these and related topics in concurrency we have neglected in our presentation, the reader may consult the two recent collections of papers [Ap3] and [BRW].
Connections to be explored in future work are (i) the relationship between our metric approaches and the various metric models developed by Degano and Montanari (e.g. [DM] where one finds a treatment of various versions of *fairness*, a notion outside the scope of our paper); (ii) the relationship between the metric and order-theoretic/categorical domain theory; (iii) non-trivial order-theoretic models for unguarded recursion (cf. [Bro1]); (iv) models for applicative languages.

Acknowledgement. We are grateful to Jan Rutten, who has been of great help in preparing the text of this paper.

2. Mathematical Preliminaries

In this section we collect some basic definitions and properties concerning (i) metric spaces and (ii) complete partially ordered sets. Both structures will play a role in the variety of denotational models to be presented in sections 3 to 7. In addition, a number of notations and definitions concerning languages with finite and infinite words are provided.

2.1. Elementary definitions.

Let X be any set. $\mathcal{P}(X)$ denotes the powerset of X, i.e., the set of all subsets of X. $\mathcal{P}_{\cdots}(X)$ denotes the set of all subsets of X which have property $\cdots$. A sequence $x_0, x_1, \cdots$ of elements of X is usually denoted by $<x_i>_{i=0}^{\infty}$ or, briefly, $<x_i>_i$. Often, we shall have occasion to use the *limit*, *supremum* (sup), *least upper bound* (lub), etc. of a sequence $<x_i>_i$. We then use the notations $\lim_{i\to\infty} x_i$, or, briefly, $lim_i\ x_i$, $sup_i\ x_i$, $lub_i\ x_i$, etc. The notation $f : X \to Y$ expresses that f is a function with domain X and range Y. If $X = Y$ and , for $x \in X$, $f(x)=x$, we call x a *fixed point* of f.

2.2. Metric spaces.

Definition 2.1. A metric space is a pair (M,d) with M a set and d (for *distance*) a mapping $d: M \times M \to [0,1]$ which satisfies the following properties:

a. $d(x,y)=0$ iff $x=y$
b. $d(x,y)=d(y,x)$
c. $d(x,y) \leq d(x,z)+d(z,y)$

If clause a. is replaced by the weaker a': $d(x,y)=0$ if $x=y$, we call (M,d) a pseudo-metric space.

Definition 2.2. Let (M,d) be a metric space.

a. Let $<x_i>_i$ be a sequence in M. We say that $<x_i>_i$ *converges to* an element x in M called its limit, whenever we have:

$$\forall \epsilon > 0\ \exists N\ \forall n > N\ [\ d(x,x_n) < \epsilon\ ]$$

A sequence $<x_i>_i$ in M is a convergent sequence if it converges to x for some $x \in X$

b. A sequence $<x_i>_i$ is called a *Cauchy sequence* whenever we have

$$\forall \epsilon > 0\ \exists N\ \forall n,m > N\ [\ d(x_n,x_m) < \epsilon\ ]$$

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

d. A subset X of a complete space (M,d) is called *closed* whenever each Cauchy sequence in X converges to an element of X.

DEFINITION 2.3.

a. Let (M_1,d_1) and (M_2,d_2) be two metric spaces. We call the spaces *isometric* if there exists a bijection $f : M_1 \rightarrow M_2$ such that, for all $x,y \in M_1$, $d_2(f(x) , f(y)) = d_1(x , y)$.

b. Let (M_1,d_1) and (M_2,d_2) be two metric spaces. We call the function $f : M_1 \rightarrow M_2$ *continuous*, whenever, for each sequence $<x_i>_i$ with limit x in M_1, we have that $lim_i\, f(x_i) = f(x)$.

c. Let (M,d) be a metric space and $f : M \rightarrow M$. We call f *contracting* if there exists a real constant c, $0 \leqslant c < 1$, such that, for all $x,y \in M$, $d(f(x) , f(y)) \leqslant c \,.\, d(x , y)$.

REMARK In part a it is, in fact, sufficient to require f to be a surjection.

PROPOSITION 2.4.

a. Each contracting function is continuous.

b. (Banach's fixed point theorem). Let (M,d) be complete and $f : M \rightarrow M$ contracting. Then f has a unique fixed point, which can be obtained as the limit of the (Cauchy) sequence $< x_0, f(x_0), f(f(x_0)), \cdots >$, for arbitrary x_0.

For each metric space (M,d) it is possible to define a complete metric space $(\tilde{M},\tilde{d})$ such that (M,d) is isometric to a (dense) subspace of $(\tilde{M},\tilde{d})$. In fact, we may take for $(\tilde{M},\tilde{d})$ the pseudo-metric space of all Cauchy sequences $<x_i>_i$ in M with distance $\tilde{d}(<x_i>_i,<y_i>_i) = lim_i\, d(x_i,y_i)$ which is turned into a metric space by taking equivalence classes with respect to the equivalence relation $<x_i>_i \equiv <y_i>_i$ iff $\tilde{d}(<x_i>_i,<y_i>_i) = 0$. M is embedded into $\tilde{M}$ by identifying each $x \in M$ with the constant Cauchy sequence $<x_i>_i$ with $x_i = x$, $i = 0, 1, \cdots$ in M.

For each metric space (M,d) we can define a metric $\hat{d}$ on the collection of its nonempty closed subsets, denoted by $\mathcal{P}_{nc}(M)$, as follows:

DEFINITION 2.5 (Hausdorff distance).

Let (M,d) be a metric space, and let X, Y be nonempty subsets of M. We put

a. $d'(x,Y) = inf_{y \in Y}\, d(x,y)$.

b. $\hat{d}(X,Y) = max(sup_{x \in X}\, d'(x,Y) \,,\, sup_{y \in Y}\, d'(y,X))$.

We have the following theorem which is quite useful in our metric denotational models:

PROPOSITION 2.6.

Let (M,d) be a metric space and $\hat{d}$ as in definition 2.5.

a. $(\mathcal{P}_{nc}(M),\hat{d})$ is a metric space.

b. If (M,d) is complete then $(\mathcal{P}_{nc}(M),\hat{d})$ is complete. Moreover, for $<X_i>_i$ a Cauchy sequence in $(\mathcal{P}_{nc}(M),\hat{d})$ we have

$$lim_i\ X_i = \{\ lim_i\ x_i : x_i \in X_i,\ <x_i>_i \text{ a Cauchy sequence in } M\ \}.$$

Proofs of proposition 2.6 can be found e.g. in [Du] or [En]. The proposition is due to Hahn [Ha]; the proof is also repeated in [BZ2]. Useful information on topologies on spaces of subsets can be found in [Mic].

We close this subsection with a few definitions and properties relating to *compact* spaces and sets. First some terminology. A subset X of a space (M,d) is *open* if its complement $M \setminus X$ is closed. A subset Y is called *dense* in (M,d) if its closure $\overline{Y}$ (the least set containing Y) equals M. A space (M,d) is called *separable* if it has a countable dense subset. An (open) *cover* of a set X is a family of (open) sets Y_i, $i \in I$, such that $X \subseteq \bigcup_{i \in I} Y_i$.

DEFINITION 2.7. Let (M,d) be a metric space.

a. (M,d) is called compact whenever each open cover of M has a finite subcover.

b. A subset X of M is called compact whenever each open cover of X has a finite subcover.

PROPOSITION 2.8.

a. Each closed subset of a compact space is compact.

b. If X is compact and f is continuous then $f(X)$ is compact.

c. X is compact iff there is a Cauchy sequence $<X_i>_i$ (with respect to the metric of definition 2.5) of *finite* sets such that $X = lim_i\ X_i$.

d. If (M,d) is separable then (M,d) is compact whenever each infinite sequence $<x_i>_i$ has a convergent subsequence.

e. A subset X of a separable space (M,d) is compact whenever each infinite sequence $<x_i>_i$, $x_i \in X$, has a sub sequence converging to an element of X.

Remark. All metric spaces considered in sections 3 to 7 are in fact separable. Therefore, we may take the properties of proposition 2.8d,e as characteristic for compactness.

In the final definition and proposition of this subsection we suppress explicit mentioning of the metrics involved. For f a function $: M_1 \to M_2$ we define $\hat{f}: \mathcal{P}_{nc}(M_1) \to \mathcal{P}_{nc}(M_2)$ by $\hat{f}(X) = \{f(x) : x \in X\}$. We have the following result from Rounds ([Ro]):

PROPOSITION 2.9.

Let f be a function from a compact metric space M_1 to a compact metric space M_2. The following three statements are equivalent:

a. f is continuous.

b. $\hat{f}:\mathcal{P}_{nc}(M_1) \to \mathcal{P}_{nc}(M_2)$ is continuous with respect to the Hausdorff metric(s).

c. For $X \in \mathcal{P}_{nc}(M_1)$, $\hat{f}(X) \in \mathcal{P}_{nc}(M_2)$ and, for $<X_i>_i$ a decreasing $(X_i \supseteq X_{i+1},\ i=0,1,2,\ \cdots)$ chain of elements in $\mathcal{P}_{nc}(M_1)$ we have

$$\hat{f}(\cap_i X_i) = \cap_i \hat{f}(X_i).$$

2.3. Complete partially ordered sets.

DEFINITION 2.10.

a. A partial order (po) is a pair $(C, \sqsubseteq)$ where C is a set and $\sqsubseteq$ a relation on C (subset of $C \times C$) satisfying

1 $x \sqsubseteq x$

2 if $x \sqsubseteq y$ and $y \sqsubseteq x$ then $x = y$

3 if $x \sqsubseteq y$ and $y \sqsubseteq z$ then $x \sqsubseteq z$

If $\sqsubseteq$ satisfies only 1 and 3 it is called a *preorder*.

b. An (ascending) chain in $(C, \sqsubseteq)$ is a sequence $<x_i>_i$ such that $x_i \sqsubseteq x_{i+1}$, $i=0,1,\cdots$. The chain is called infinitely often increasing if $x_i \neq x_{i+1}$ for infinitely many i.

c. For $X \subseteq C$ we call $y \in C$ the *least upperbound* (lub) of X if

1 $\forall x \in X[x \sqsubseteq y]$

2 $\forall z \in C[\forall x \in X[x \sqsubseteq z] \Rightarrow y \sqsubseteq z]$

DEFINITION 2.11. A *complete partially ordered* set (cpo) is a triple $(C, \sqsubseteq, \perp)$ with $(C, \sqsubseteq)$ a po and $\perp \in C$ such that

a. $\forall x \in C[\perp \sqsubseteq x]$

b. Each chain $<x_i>_i$ in C has a lub in C.

For "the cpo $(C, \sqsubseteq, \perp)$" we often simply write "the cpo C". For $\text{lub}_i\ x_i$ we also write $\sqcup_i x_i$.

DEFINITION 2.12 (continuity). Let C_1 and C_2 be cpo's.

a. A function $f : C_1 \to C_2$ is called monotonic whenever for all $x_1, x_2 \in C_2$, if $x_1 \sqsubseteq x_2$ then $f(x_1) \sqsubseteq f(x_2)$.

b. A function $f : C_1 \to C_2$ is called *continuous* whenever it is monotonic and, for each chain $<x_i>_i$ in C_1 we have $f(\sqcup_i x_i) = \sqcup_i f(x_i)$.

PROPOSITION 2.13.
Let f be a continuous mapping from a cpo C into itself. f has *a least fixed point* $\mu f =^{df} \bigsqcup_i f^i(\bot)$ satisfying

1 $f(\mu f) = \mu f$

2 if $f(y) \sqsubseteq y$ then $\mu f \sqsubseteq y$.

DEFINITION 2.14.

a. A subset X is called *flat* whenever, for all $x,y \in X$, $x \sqsubseteq y$ implies $x = y$.

b. A subset X of a cpo C is called *closed* whenever, for each infinitely often increasing chain $<x_i>_i$ of elements in C such that, for all $i = 0, 1, \cdots$, we have that $x_i \sqsubseteq y_i$ for some $y_i \in X$, it follows that $\bigsqcup_i x_i \in X$.

This definition of closed appears in [Ba] or [Ku]. We now introduce a number of preorders on $\mathcal{P}(C)$, for $(C, \sqsubseteq, \bot)$ a cpo.

DEFINITION 2.15.

a. The Smyth preorder $\sqsubseteq_S$: $X \sqsubseteq_S Y$ iff $\forall y \in Y \exists x \in X [x \sqsubseteq y]$

b. The Hoare preorder $\sqsubseteq_H$: $X \sqsubseteq_H Y$ iff $\forall x \in X \exists y \in Y [x \sqsubseteq y]$

c. The Egli-Milner preorder $\sqsubseteq_{EM}$: $X \sqsubseteq_{EM} Y$ iff $X \sqsubseteq_S Y$ and $X \sqsubseteq_H Y$.

None of the three preorders is, in general, a partial order. In fact, we may take the two sets $X = \{x,y,z\}$ and $Y = \{x,z\}$ with $x \sqsubseteq y$ and $y \sqsubseteq z$ as a counterexample. In later sections we shall encounter various special cases where the preorder is turned into a (complete) partial order by additional requirements.

2.4. Finite and infinite words.

We introduce some basic definitions and notations for languages consisting of finite and infinite words. Let A be a *finite* alphabet, i.e., a finite nonempty set of symbols with a concatenation operator.

Important remark. We emphasize that throughout the paper A is always assumed finite. At a number of places this is an essential condition. In particular, certain continuity properties stated in later sections do not hold when finiteness of A is not assumed.

DEFINITION 2.16. Let A be an alphabet. We use A^* to denote the collection of all finite words over A and A^ω to denote the collection of all infinite words over A. We put

$$A^\dagger =^{df} A^* \cup A^\omega.$$

We use ϵ for the empty word, a^* for the set of all finite sequences of a's, and a^ω for the infinite sequence of a's. Analogously we use notations such as $(ab)^*$ or $(ab)^\omega$, etc. We shall use

$u,v,w, \cdots$ to range over $A^\dagger$ and $X,Y, \cdots$ for subsets of $A^\dagger$.
We next define the prefix order "$\leqslant$" on $A^\dagger$:

DEFINITION 2.17.
a. For $u \in A^*$, $|u|$ denotes the *length* of u.
b. For $u,v \in A^\dagger$ we put $u \leqslant v$ if there exists w such that $u.w = v$ (see definition 2.18a). We call u a *prefix* of v.
c. For $u,v \in A^\dagger$, $u(n)$ denotes the prefix of u of length n, in case $|u| \geqslant n$. Otherwise, $u(n) = u$.
d. For $X \subseteq A^\dagger$, $X(n) = \{u(n) : u \in X\}$.

Finally, we define the operators of concatenation (.) and shuffle ($\|$) for words and sets of words:

DEFINITION 2.18.
a. $u.v$ is defined as usual for $u \in A^*$, $v \in A^\dagger$. Moreover, $u.v = u$ for $u \in A^\omega$, $v \in A^\dagger$.
b. $X.Y = \{u.v : u \in X, v \in Y\}$.
c. The shuffle $u \| v$ yields a subset of $A^\dagger$ defined in the following way: let $\bar{A}$ and $\bar{\bar{A}}$ be copies of A with $\bar{A} = \{\bar{a} : a \in A\}$ and $\bar{\bar{A}} = \{\bar{\bar{a}} : a \in A\}$. Let h, h_1, h_2 be homomorphisms (with respect to ".") such that, for each $a \in A$, $h(\bar{a}) = h(\bar{\bar{a}}) = a$, $h_1(\bar{a}) = h_2(\bar{\bar{a}}) = a$, $h_2(\bar{a}) = h_1(\bar{\bar{a}}) = \epsilon$. For $u,v \in A^*$ we define (cf. [HU], p.142)

$$u \| v = \{ w \mid \exists w' \in h^{-1}(w) [h_1(w') = u , h_2(w') = v] \}.$$

If u or v belongs to A^ω we define

$$u \| v = \{ w \in A^\omega \mid \exists w' \in h^{-1}(w) [h_1(w') \leqslant u , h_2(w') \leqslant v] \}.$$

d. $X \| Y = \bigcup \{ u \| v : u \in X , v \in Y \}$.

Remark. Note that, by clause c, for u or v in A^ω, we have, since $w \in A^\omega$, that either $h_1(w') = u$ or $h_2(w') = v$.

Examples.
1. $ab \| c = \{abc, acb, cab\}$.
2. $a^\omega \| bc = \{a^\omega\} \cup a^*ba^\omega \cup a^*ba^*ca^\omega$.

Remark. For a reader interested in fairness (a notion not dealt with below): if we replace, in the last formula of clause c, $\leqslant$ by $=$ (twice) we obtain the definition of the fair merge $u \|_f v$. For example, $a^\omega \|_f bc = a^*ba^*ca^\omega$.

3. SHUFFLE AND LOCAL NONDETERMINACY: OPERATIONAL AND METRIC DENOTATIONAL SEMANTICS

3.1. Introduction and syntax.

Section 3 is concerned with the first (out of four) of the languages studied in our paper. We call this language L_0 and let s,t stand for typical elements of L_0. Elements s,t are also called (concurrent) *statements.* For the syntax of L_0 we need two classes of terminal elements:

1 The class A, with typical elements $a,b, \cdots$, of elementary actions. For A we take an arbitrary (but finite!) alphabet.

2 The class *Stmv,* with typical elements $x,y, \cdots$, of statement variables. For *Stmv* we take some infinite set of symbols: it is convenient to have an infinite supply of fresh statement variables. Statement variables play a role in the syntactic construct for *recursion* as we shall see in a moment.

We now give, in a self-explanatory notation,

DEFINITION 3.1 (Syntax for L_0)

$$s ::= a \mid s_1 ; s_2 \mid s_1 \cup s_2 \mid s_1 \| s_2 \mid x \mid \mu x[\, s \,].$$

A statement s is of one of the following six forms:

- an *elementary action a.*
- the *sequential composition* $s_1 ; s_2$ of statements s_1 and s_2.
- the *nondeterministic choice* $s_1 \cup s_2$, also known as *local nondeterminism* [FHLR]: $s_1 \cup s_2$ is executed by executing either s_1 or s_2 chosen nondeterministically.
- the *concurrent execution* $s_1 \| s_2$, modelled by the arbitrary interleaving *(shuffle)* of the elementary actions of s_1 and s_2
- a statement variable x which is (normally) used in
- the recursive construct $\mu x[\, s \,]$: its execution amounts to execution of s where occurrences of x in s are executed by (recursively) executing $\mu x[\, s \,]$. For example, with the definitions to be proposed presently, the intended meaning of $\mu x[\, (a \,;\, x) \cup b \,]$ is the set $a^*.b \cup \{ a^\omega \}$.

L_0 has no synchronization or communication. These will appear only in languages L_1 to L_3. In the absence of communication, nondeterministic choice is just arbitrarily choosing between s_1 and s_2. Only in the subsequent languages the distinction between local and global nondeterminacy becomes meaningful.

L_0 is, indeed, a quite simple language. However, it does extend classical sequential (schematic) languages with the shuffle operator and therefore induces several challenging problems. In fact, we believe that the full operational and denotational semantics together with the proof establishing their equivalence, as described in [BMOZ 1,2], has not been described before in the literature. The remainder of this section is devoted to operational semantics (subsection 3.2), denotational

semantics in a metric style (subsection 3.3), and a brief description of the main steps in the equivalence proof (subsection 3.4).

The primary source for section 3 is the paper [BMOZ 2], in particular its second section. A preliminary version of this paper appeared in [BMOZ 1].

3.2. Operational semantics.

We first introduce as semantic universe for both operational and denotational semantics for L_0 the set of *streams* over A (cf [Bro1]), defined in

DEFINITION 3.2 (streams). Let A be an alphabet and $\perp \notin A$. The set A^{st} of streams over A is defined by

$$A^{st} =^{df} A^* \cup A^\omega \cup A^*.\{\perp\}$$

We assume the definitions of section 2.3, and use u,v,w to range over A^{st} and $X,Y,...$ for subsets of A^{st}. In addition, we postulate that $\perp.u = \perp$ for all u.

In (denotational) semantics, $\perp$ usually serves one (or both) of the following purposes:

1. It indicates incomplete information which may be filled in at a later stage of the computation
2. It indicates a nonterminating computation.

In the present context (of section 3) $\perp$ is used primarily for (an improper form of) nonterminating computations. Its role in the sense of (i) will follow in the order theoretic treatment (of L_1) in section 5.

The operational semantics for L_0 is based on the notion of a *transition system.* A transition describes what a statement s can do as its next step. This concept of a transition dates back to [Ke] and to automata theoretic notions [RS]. Following Hennessy and Plotkin [HP,Pl3,Pl4], a transition system is a syntax-directed deductive system for proving transitions (see also [Ap1,Ap2]).

A *configuration* is a pair $<s,w>$ or just a word w. A *transition relation* is a binary relation over configurations [Ke]. A *transition* is a formula $<s,w>\rightarrow<s',w'>$ or $<s,w>\rightarrow w'$ denoting an element of a transition relation. A *transition system* is a formal deductive system for proving transitions, based on *axioms* and *rules.* Using a self-explanatory notation, axioms have the format $1\rightarrow 2$, rules have the format $\frac{1\rightarrow 2}{3\rightarrow 4}$. Also, $1\rightarrow 2\,|\,3$ abbreviates $1\rightarrow 2$ and $1\rightarrow 3$, and $\frac{1\rightarrow 2\,|\,3}{4\rightarrow 5\,|\,6}$ abbreviates $\frac{1\rightarrow 2}{4\rightarrow 5}$ and $\frac{1\rightarrow 3}{4\rightarrow 6}$.

For a transition system T, $T \vdash 1\rightarrow 2$ expresses that transition $1\rightarrow 2$ is deducible in the system T. Then $1\rightarrow 2$ is also called a $T-$*transition.* For a finite sequence $1\rightarrow 2\rightarrow 3\rightarrow \cdots \rightarrow n$ of T-transitions, we also write $T \vdash 1\rightarrow^* n$. Finally, we shall allow several conclusions from one

premise in a transition rule. For example,

$$\frac{1\to 2}{\begin{array}{l}3\to 4\\5\to 6\end{array}}$$

abbreviates the two rules $\frac{1\to 2}{3\to 4}$ and $\frac{1\to 2}{5\to 6}$.

We now present the transition system T_0 for L_0. It will be convenient to extend the class of statements L_0 with an auxiliary statement E, satisfying the syntactic identities, for all s,w: $<E,w> = w$ and

$$E\ ;\ s = s\ ;\ E = s \parallel E = E \parallel s = s.$$

Moreover, we shall employ a notation for *substitution.* For s,t statements and x a statement variable, we write $s[t/x]$ for the result of substituting t for all free occurrences of x in s. The notion of free (and bound) variable is taken with respect to the variable binding operator $\mu x.\cdots$. The usual precautions avoiding clashes of free and bound variables apply. For example, we have $\mu x[(a_1;x) \cup (b;y)][(a_2;x)/y] = \mu x'[(a_1;x') \cup (b;a_2;x)]$, where x' is some fresh variable.

We shall define T_0 only for *syntactically closed* statements, i.e., for statements without free statement variables. A similar condition applies to all subsequent transition systems of our paper.

We now define T_0. For $w \in A^\omega \cup A^*.\{\perp\}$ we put

$$<s,w>\to w$$

and for $w \in A^*$ we distinguish the following cases:

(elementary action)

$$<a,w>\to w.a$$

(local nondeterminacy)

$$<s_1 \cup s_2>\to <s_1,w> \mid <s_2,w>$$

(recursion)

$$<\mu x[s],w>\to <s[\mu x[s]/x],w>$$

(sequential composition)

$$\frac{<s_1,w>\to <s',w'>}{<s_1\ ;\ s_2,w>\to <s'\ ;\ s_2\ ,w'>}$$

where s' may be E.

(shuffle)

$$\frac{<s_1,w>\rightarrow<s',w'>}{\begin{array}{l}<s_1 \parallel s_2,w>\rightarrow<s' \parallel s_2,w'> \\ <s_2 \parallel s_1,w>\rightarrow<s_2 \parallel s',w'>\end{array}}$$

where s' may be E.

Examples

1. We prove $<(a_1;a_2)\|b,\epsilon> \rightarrow <a_2\|b,a_1>$. We have, successively
   (1) $<a_1,\epsilon> \rightarrow a_1$ (by elementary action)
   (2) $<a_1;a_2,\epsilon> \rightarrow <a_2,a_1>$ (by (1) and seq. comp.)
   (3) $<(a_1;a_2)\|b,\epsilon> \rightarrow <a_2\|b,a_1>$ (by (2) and shuffle)
2. $<\mu x[(a_1;x)\cup b],\epsilon> \rightarrow <a_1;\mu x[(a_1;x)\cup b]\cup b,\epsilon> \rightarrow$
   $<a_1;\mu x[(a_1;x)\cup b],\epsilon> \rightarrow <\mu x[(a_1;x)\cup b],a_1> \rightarrow \cdots$
   by application of recursion, local nondeterminacy, elementary action + sequential composition,..., respectively.

Note that every rule in T_0 has only one premise. Thus every deduction of a transition

$$<s,w>\rightarrow w' \mid <s',w'> \qquad (*)$$

starts from a single axiom. Moreover, for different deductions of (*), these axioms may differ, but they will all be instances of the same scheme. Thus, if (Ax) is the name of this scheme, we call (*) an *Ax-transition* in T_0.

Examples.

(i) $<a;s,w>\rightarrow<s,wa>$ is an elementary action transition.

(ii) $<\mu x[x]\|\mu y[y],w>\rightarrow<\mu x[x]\|\mu y[y],w>$ is a recursion-transition (though there are two different deductions of this transition, one starting from $\mu x[x]$ and the other from $\mu y[y]$).

Note. In example (i) and often below, we omit the concatenation operator for easier readability.

We now proceed with the definition of the operational semantics $\mathcal{O}_0$ for L_0 based on T_0. We define $\mathcal{O}_0$ as a mapping

$$\mathcal{O}_0 : L_0 \rightarrow \mathbb{S}$$

with $\mathbb{S} = \mathcal{P}(A^{st})$. Following a semantic tradition, for $s \in L_0$ we write $\mathcal{O}_0[\![s]\!]$ rather than simply $\mathcal{O}(s)$ for the result of applying the function $\mathcal{O}_0$ to s. The definition of $\mathcal{O}_0$ is given in

DEFINITION 3.3. Let $s \in L_0$ be without free statement variables. We put $w \in \mathcal{O}_0[\![s]\!]$ iff one of the following two conditions is satisfied (always taking $<s_0,w_0>=<s,\epsilon>$):

1 There is a finite sequence of T_0-transitions

$$<s_0,w_0>\rightarrow \cdots \rightarrow <s_n,w_n>\rightarrow w.$$

2 There is an infinite sequence of T_0-transitions

$$<s_0,w_0>\rightarrow \cdots \rightarrow <s_n,w_n>\rightarrow \cdots$$

and $w=(sup_n w_n).\perp$, where *sup* is taken with respect to the prefix order $\leqslant$, see section 2.4.

Remark. It is not difficult to show that, if $<s,w>\rightarrow <s',w'>$ then $w\leqslant w'$. This justifies the clause $w=(sup_n w_n).\perp$ in clause 2. If $<w_n>_n$ is infinitely often increasing then $sup_n w_n \in A^\omega$ and, hence, $w=sup_n w_n$. Otherwise, there exists some n such that $w_{n+k}=w_n$, $k=0,1,2, \cdots$, and $w=w_n.\perp$. Here we see the role of $\perp$ in modelling improper nontermination.

Examples.
$\mathcal{O}_0[\![(a_1;a_2)\|a_3]\!]=\{a_1a_2a_3,\ a_1a_3a_2,\ a_3a_1a_2\}$,
$\mathcal{O}_0[\![\mu x[(a;x)\cup b]]\!]=a^*.b\cup\{a^\omega\}$,
$\mathcal{O}_0[\![\mu x[(x;a)\cup b]]\!]=b.a^*\cup\{\perp\}$.

Note that systems such as T_0 are used to deduce one step transitions $1\rightarrow 2$. *Sequences* of such transitions are used only to define $\mathcal{O}_0[\![\ \cdots\ ]\!]$.

We postpone till section 3.4 the statement of various fundamental facts about $\mathcal{O}_0$ which, besides being of importance in their own right, will in particular be applied in the comparison between operational and denotational semantics.

3.3. Denotational Semantics.
The operational semantics $\mathcal{O}_0$ for L_0 is *global* in the following sense: to determine $\mathcal{O}_0[\![s]\!]$ we first have to explore the T_0-transition sequences for all of s and only then we can retrieve the result $\mathcal{O}_0[\![s]\!]$. Further, in T_0 and thus in $\mathcal{O}_0$ recursion is dealt with by syntactic copying. We now look or a *denotational semantics* $\mathcal{D}_0$ for L_0. A denotational semantics should be *compositional* or *homomorphic,* i.e. for every syntactic operator *op* in L_0 there should be a corresponding operator $op^{\mathcal{D}_0}$ satisfying

$$\mathcal{D}_0[\![s_1\ op\ s_2]\!]=\mathcal{D}_0[\![s_1]\!]\ op^{\mathcal{D}_0}\ \mathcal{D}_0[\![s_2]\!]$$

and it should tackle recursion semantically with the help of *fixed points.* This of course requires a suitable structure of the underlying semantic domain.

For $\mathcal{D}_0$ we shall use metric spaces as semantic domain. (For the subsequent language L_1 we shall develop denotational models based both on metric and on order-theoretic structures.) Our approach is based on the papers [BBKM] and [BZ2]; related technical results appear in [BZ3]. For metric preliminaries see section 2.2. An important technical restriction is that we define $\mathcal{D}_0$ only for *guarded* statements (definition 3.4). The reason for this is that the metric treatment of recursive constructs is only valid under this requirement. As we shall see in section 5, order-

theoretic methods are (slightly) more adequate for the general case.

We now present the definition of guarded (cf. also [Mi2], or [Ni1] where *Greibach* replaces guarded). Intuitively, a statement s is guarded when all its recursive substatements $\mu x[t]$ satisfy the condition that (recursive) occurrences of x in t are semantically preceded by some statement. More precisely, we have

DEFINITION 3.4. (guarded statement)

a. We first define the notion of an occurrence of a variable x being *exposed* in s. The definition is by structural induction on s:

1 x is exposed in x.

2 If an occurrence of x is exposed in s_1, then it is exposed in $s_1;s_2$, $s_1\|s_2$, $s_2\|s_1$, $s_1 \cup s_2$, $s_2 \cup s_1$, and in $\mu y[s_1]$ for $x \neq y$.

b. A statement is defined to be guarded if for all its recursive substatements $\mu x[t]$, t contains no exposed occurrences of x.

Examples.

1 In the statement $x;a \cup b;x$, the first occurrence of x is exposed, and the second is not.

2 $\mu x[a;(x\|b)]$ is guarded, but $\mu x[x]$, $\mu y[y\|b]$, and $\mu y[\mu x[y]]$ (as well as any statement containing these) are not.

We now proceed with the definition of $\mathcal{D}_0$. The first step is to turn the set A^{st} into a metric space by defining a distance $d : A^{st} \times A^{st} \to [0,1]$ as follows:

DEFINITION 3.5. For $u,v \in A^{st}$ we put

$$d(u,v) = 2^{-sup\{n \mid u(n)=v(n)\}+1}$$

with the understanding that $2^{-\infty} = 0$.

Remark. Note that $d(u,v) = 2^{-n+1}$ iff u,v exhibit their first difference in the n-th position.

Example. $d(aba_1, aba_2) = 2^{-3+1} = \frac{1}{4}$, $d(a^n, a^\omega) = 2^{-n}$. We have

PROPOSITION 3.6.

(A^{st}, d) is a complete and compact metric space.

The proof can be found, for example, in [Ni1]. We next define a distance $\hat{d}$ on subsets X, Y of A^{st} in

DEFINITION 3.7. For $X, Y \in A^{st}$ we put

$$\hat{d}(X,Y) = 2^{-sup\{n : X(n)=Y(n)\}+1}$$

with $2^{-\infty}$ as before, and $X(n), Y(n)$ as in definition 2.17.

Let $\mathbb{S}_{nc}$ denote the set of all nonempty closed subsets of A^{st}. An example of a closed set is $a^*.b \cup \{a^\omega\}$. However, $a^*.b$ is not closed since the Cauchy sequence $<a^i.b>_i$ does not have its limit a^ω in $a^*.b$. We have (cf. definition 2.5, proposition 2.6 and [BBKM]):

PROPOSITION 3.8.

a. $(\mathbb{S}_{nc}, \hat{d})$ is a complete metric space.

b. $\hat{d}$ coincides with the Hausdorff distance on $\mathbb{S}_{nc}$ induced by the distance d on A^{st} as defined in definition 3.5.

We now define the semantic operators $;^{\mathfrak{D}_0}$, $\cup^{\mathfrak{D}_0}$ and $\|^{\mathfrak{D}_0}$ on $\mathbb{S}_{nc}$. (For ease of notation, we skip superscripts $\mathfrak{D}_0$ if no confusion arises.)

DEFINITION 3.9.

a. $X, Y \subseteq A^* \cup A^*.\{\perp\}$. For $X;Y =^{df} X.Y$ (concatenation) and $X \cup Y$ (set-theoretic union) we adopt the usual definitions (including the clause $\perp.u = \perp$ for all u). For $X \| Y$ (shuffle or merge) we introduce as auxiliary operator the so called left-merge $\lfloor\!\lfloor$ (from [BK1]). It enables a particularly simple definition of $\|$ by putting

$$X \| Y = (X \lfloor\!\lfloor Y) \cup (Y \lfloor\!\lfloor X)$$

where $\lfloor\!\lfloor$ is given recursively by $X \lfloor\!\lfloor Y = \bigcup \{u \lfloor\!\lfloor Y : u \in X\}$ with $\epsilon \lfloor\!\lfloor Y = Y$, $(a.u) \lfloor\!\lfloor Y = a.(\{u\} \| Y)$ and $\perp \lfloor\!\lfloor Y = \{\perp\}$.

b. $X, Y \in \mathbb{S}_{nc}$ where X, Y do not consist of finite words only. Then

$$X \; op \; Y = lim_i (X(i) \; op \; Y(i))$$

for $op \in \{;, \cup, \|\}$.

Examples

1. $\{a_1a_2, a_3\} \lfloor\!\lfloor b = (a_1a_2 \lfloor\!\lfloor \{b\}) \cup (a_3 \lfloor\!\lfloor \{b\}) =$
   $a_1.(\{a_2\} \| \{b\}) \cup \{a_3 b\} = \{a_1a_2b, a_1ba_2, a_3b\}$.
2. $\{ab\perp\} \| \{c\} = \{cab\perp, acb\perp, abc\perp, ab\perp\}$.
3. $a^\omega \| b^\omega = lim_n (a^n \| b^n) = \{a,b\}^\omega$.

The operators are well-defined and continuous, as stated in

PROPOSITION 3.9.

a. The operators $;, \cup, \|$ preserve closedness.

b. The operators $;, \cup, \|$ coincide on $\mathbb{S}_{nc} \times \mathbb{S}_{nc}$ with the operators as defined in definition 2.18.

c. The operators $;, \cup, \|$ are *continuous* mappings: $\mathbb{S}_{nc} \times \mathbb{S}_{nc} \to \mathbb{S}_{nc}$.

Most of this can be found in [BBKM]. Further and related information is contained in [BZ2,BZ3,Ro].

We proceed with the definition of $\mathcal{D}_0$. We introduce the usual notion of *environment* which is used to store and retrieve meanings of statement variables. Let $\Gamma_0 = Stmv \to \mathbb{S}_{nc}$ be the set of environments, and let $\gamma \in \Gamma_0$. We write $\gamma' =^{df} \gamma < X / x >$ for a *variant* of γ which is like γ but with $\gamma'(x) = X$. We define

$$\mathcal{D}_0 : guarded\ L_0 \to (\Gamma_0 \to \mathbb{S}_{nc})$$

in

DEFINITION 3.10.

1 $\mathcal{D}_0[\![a]\!](\gamma) = \{a\}$.

2 $\mathcal{D}_0[\![s_1\ op^{\mathcal{D}_0}\ s_2]\!](\gamma) = \mathcal{D}_0[\![s_1]\!](\gamma) op^{\mathcal{D}_0} \mathcal{D}_0[\![s_2]\!](\gamma)$.

3 $\mathcal{D}_0[\![x]\!](\gamma) = \gamma(x)$.

4 $\mathcal{D}_0[\![\mu x[s]]\!](\gamma) = lim_i X_i$, where $X_0 = \{\bot\}$ and $X_{i+1} = \mathcal{D}_0[\![s]\!](\gamma < X_i / x >)$.

Example. $\mathcal{D}_0[\![\mu x[(a;x) \cup b]]\!](\gamma) = lim_i X_i$, where $X_0 = \{\bot\}$, $X_{i+1} = \mathcal{D}_0[\![(a;x) \cup b]\!](\gamma < X_i / x >) = a.X_i \cup \{b\}$. Hence, $X_i = \{a^i \bot\} \cup \{a^j b : j \leqslant i-1\}$, and $lim_i X_i = \{a^\omega\} \cup a^* b$.

Definition 3.10, in particular its clause 4 is justified by

PROPOSITION 3.11.
For guarded s, the function $\Phi = \lambda X.\mathcal{D}_0[\![s]\!](\gamma < X / x >) : \mathbb{S}_{nc} \to \mathbb{S}_{nc}$ is contracting.

The proof is an inessential variation on the results in Appendix B of [BZ2]. Thus, by proposition 2.4, Φ has a unique fixed point which is obtained as limit of the Cauchy sequence $<X_i>_i$ as in the definition. The choice of $X_0 = \{\bot\}$ is for definiteness; it is also convenient in the proof of the result $\mathcal{O}_0 = \mathcal{D}_0$ (section 3.4).

For syntactically closed statements we write $\mathcal{D}_0[\![s]\!]$ instead of $\mathcal{D}_0[\![s]\!](\gamma)$. Since $\mathcal{D}_0[\![s]\!]$ is a set of (linear) streams, $\mathcal{D}_0$ is called a *linear time* semantics (cf. [BBKM]). Such a semantics may constitute the basis for a linear time temporal logic for L_0, cf. [Pn].

3.4. Equivalence of $\mathcal{O}_0$ and $\mathcal{D}_0$.
The following theorem can be established:

THEOREM 3.12. $\mathcal{O}_0[\![s]\!] = \mathcal{D}_0[\![s]\!]$ for each syntactically closed and guarded $s \in L_0$.

Theorem 3.12. is the first main result of [BMOZ 1,2]. We shall sketch here the outline of its proof by presenting its basic constituent lemmas. The full story is described in the original paper.

LEMMA 3.13. $\mathcal{O}_0$ behaves compositionally and satisfies the fixed point property:

a. $\mathcal{O}_0[\![s_1\ op\ s_2]\!] = \mathcal{O}_0[\![s_1]\!] op^{\mathcal{D}_0} \mathcal{O}_0[\![s_2]\!]$.
b. $\mathcal{O}_0[\![\mu x[s]]\!] = \mathcal{O}_0[\![s[\mu x[s]/x]]\!]$.

Part a. involves a lot of technical work for op = ||; part b. is direct from the definitions.

LEMMA 3.14. Consider the recursive statement $\mu x[s]$. Let Ω be an auxiliary statement with associated auxiliary transitions (added to T_0) $<\Omega,w> \rightarrow w\perp$, $<\Omega;s,w> \rightarrow <\Omega,w>$, $<\Omega \| s,w> \rightarrow <\Omega,w>$, $<s\|\Omega,w> \rightarrow <\Omega,w>$. Let $s^{(0)} =^{df} \Omega$, $s^{(n+1)} = s[s^{(n)}/x]$. We have

$$\mathcal{O}_0[\![\mu x[s]]\!] = lim_n \mathcal{O}_0[\![s^{(n)}]\!].$$

The proof is complicated and requires an elaborate analysis of transition sequences and their *truncations* for recursive constructs.

LEMMA 3.15. For each guarded s, $\mathcal{O}_0[\![s]\!]$ is a closed set.

Examples. For the guarded $s_1 \equiv \mu x[(a;x) \cup b]$, $\mathcal{O}_0[\![s_1]\!]$ equals the closed set $a^*.b \cup \{a^\omega\}$. For the unguarded $s_2 \equiv \mu x[(x;a) \cup b]$, $\mathcal{O}_0[\![s_2]\!]$ equals the nonclosed set $b.a^* \cup \{\perp\}$ (this set does not contain its limit point $b.a^\omega$).

The proof of lemma 3.15 relies on the following fact:

LEMMA 3.16. For each guarded s,w,a there are only finitely many production sequences

$$<s,w> \rightarrow^* <s',wa>$$

(for some s').

Remark. Statement s_2 from the above example may be used to show that guardedness is necessary here.

Finally, we have the basic lemma relating $\mathcal{O}_0$ and $\mathcal{D}_0$:

LEMMA 3.17. Let $\{x_1, \cdots, x_n\}$ be the set of free statement variables in s. Let $t_1, \cdots, t_n$ be syntactically closed statements. Then if

$$\mathcal{O}_0[\![t_i]\!] = X_i\ ,\ i = 1, \cdots, n$$

then

$$\mathcal{O}_0[\![s[t_i/x_i]_{i=1}^n]\!] = \mathcal{D}_0[\![s]\!](\gamma <X_i/x_i>_{i=1}^n).$$

From this lemma -which is proved by structural induction on s- theorem 3.12 follows by taking for s a syntactically closed statement.

4. Synchronization merge and local nondeterminacy: operational and metric denotational semantics

4.1. Syntax and operational semantics.

For L_1 we introduce some structure to the finite alphabet A. Let $C \subseteq A$ be a subset of so-called *communications*. From now on let c range over C and a,b over $A \setminus C$. Similarly to CCS [Mi2] or CSP [Ho] we stipulate a bijection $\bar{}: C \to C$ with $\bar{\bar{c}} = c$ which for every $c \in C$ yields a *matching communication* $\bar{c}$. There is a special action $\tau \in A \setminus C$ denoting the result of a synchronization of c with $\bar{c}$ [Mi2]. As syntax for $s \in L_1$ we give now:

Definition 4.1.

$$s ::= a \mid c \mid s_1;s_2 \mid s_1 \cup s_2 \mid s_1 \| s_2 \mid x \mid \mu x[s].$$

Apart from a distinction between communications and ordinary elementary actions, the syntax of L_1 agrees with that of L_0. The difference between L_1 and L_0 lies in a more sophisticated interpretation of $s_1 \| s_2$ to be presented soon.

The introduction of communications is responsible for the fact that a statement s may now fail. In particular this happens when a communication is executed (See, however, the remark at the end of subsection 4.1.) We first extend our stream domain A^{st} with an element for failure: Let $\delta \notin A \cup \{\perp\}$ be an element indicating failure with $\delta.w = \delta$ for all w. The set of streams or words is extended to

$$A^{st}(\delta) = A^{st} \cup A^*.\delta$$

with u,v,w now ranging over $A^{st}(\delta)$. The transition system T_1 consists of all axioms and rules of T_0 extended with

$$<s,w> \to w \text{ for } w \in A^{\omega} \cup A^*.\{\delta, \perp\},$$

and for $w \in A^*$ with:

(communication)

$$<c,w> \to w\delta$$

(an individual communication fails)

(synchronization)

$$<c \| \bar{c}, w> \to w\tau$$

(synchronization in a context) (*)

(*) This rule (from [BMOZ2]) corrects an inadequate treatment of synchronization in T_1 of [BMOZ1].

$$\frac{<s_1\|s_2,w>\rightarrow<s'\|s'',w\tau>}{\begin{array}{l}<(s_1;s)\|s_2,w>\rightarrow<(s';s)\|s'',w\tau>\\ <(s_1\|s)\|s_2,w>\rightarrow<(s'\|s)\|s'',w\tau>\\ <(s\|s_1)\|s_2,w>\rightarrow<(s\|s')\|s'',w\tau>\\ <s_1\|(s_2;s),w>\rightarrow<s'\|(s'';s),w\tau>\\ <s_1\|(s_2\|s),w>\rightarrow<s'\|(s''\|s),w\tau>\\ <s_1\|(s\|s_2),w>\rightarrow<s'\|(s\|s''),w\tau>\end{array}}$$

where s' or s'' or both may be E and where the premise of the rule describes a synchronization-transition between s_1 and s_2 such that s' stems from s_1 and s'' stems from s_2.

The last rule requires some explanation. First consider a transition of the form

$$<s_1\|s_2,w>\rightarrow<s',w'>.$$

An occurrence of a substatement s of s' is said to *stem from* s_1 (or s_2) if whenever s_1 and s_2 were coloured 'blue' and 'green' respectively, then s would be exclusively coloured 'blue' (or 'green'). Note that the concept of colouring is just a convenient way of tracing occurrences in configurations changed by transitions. For example, in the transition

$$<(c;s_1)\|(\overline{c};s_2),w>\rightarrow<s_1\|s_2,w\tau>$$

s_1 stems from $c;s_1$ and s_2 stems from $\overline{c};s_2$.

Next note that as in T_0 we can talk of an *Ax-transition* for some axiom Ax of T_1. In particular, a transition of the form

$$<s_1\|s_2,w>\rightarrow<s',w\tau> \qquad (*)$$

is called a *synchronization-transition between* s_1 *and* s_2 if any deduction of (*) starts with a synchronization axiom

$$<c\|\overline{c},w>\rightarrow w\tau$$

such that s_1 has the same colour as c and s_2 has the same colour as $\overline{c}$. In contrast, a transition

$$<s_1\|s_2,w>\rightarrow<s'\|s'',w'> \qquad (**)$$

is called a *local transition* if any deduction of (**) starts with an axiom of the form $<s,w>\rightarrow w'$ such that s is a substatement of either s_1 or s_2. (Note: the "||" shown in (**) is introduced by the shuffle rule, not the synchronization rule, and so $s_2=s''$ or $s_1=s'$.)

Examples.

1. $<(c;s')\|((c\|\overline{c});s''),w>\rightarrow<s'\|(c;s''),w\tau>$ is a synchronization-transition between $s_1=c;s'$ and $s_2=(c\|\overline{c});s''$.
2. $<(c;s')\|((c\|\overline{c});s''),w>\rightarrow<(c;s')\|s'',w\tau>$ is a local transition involving only the second argument $s_2=(c\|\overline{c});s''$ of the top-level $\|$ operator.

Analogously to $\mathcal{O}_0$ we base an operational semantics $\mathcal{O}_1$ on T_1. $\mathcal{O}_1$ is a mapping $\mathcal{O}_1:L_1\rightarrow\mathbb{S}(\delta)$ with $\mathbb{S}(\delta)=\mathcal{P}(A^{st}(\delta))$, and $\mathcal{O}_1[\![s]\!]$ is defined exactly the same way as $\mathcal{O}_0[\![s]\!]$ in section 2.2.

Examples.

1. We show $<((c\|a);b)\|\overline{c},\epsilon> \rightarrow <a;b,\tau>$:
   (1) $<c\|\overline{c},\epsilon> \rightarrow <E\|E,\tau>$ (synchronization)
   (2) $<(c\|a)\|\overline{c},\epsilon> \rightarrow <a\|E,\tau>$ (synchr. in a context, (1))
   (3) $<((c\|a);b)\|\overline{c},\epsilon> \rightarrow <(a;b)\|E,\tau>$ (synchr. in a context, (2))
2. $\mathcal{O}_1[\![c]\!]=\{\delta\}$, $\mathcal{O}_1[\![c\|\overline{c}]\!]=\{\delta,\tau\}$, $\mathcal{O}_1[\![(a;b)\cup(a;c)]\!]=\mathcal{O}_1[\![a;(b\cup c)]\!]=\{ab,a\delta\}$.

From the second example we see that under $\mathcal{O}_1$ communications c always create failures whether or not they can synchronize with a matching communication $\overline{c}$. Also the two statements $(a;b)\cup(a;c)$ and $a;(b\cup c)$ obtain the same meaning by $\mathcal{O}_1$. This is characteristic for local nondeterminacy $s_1\cup s_2$ where the choice of s_1 or s_2 is independent of the form of the other component s_2 or s_1 respectively. A more refined treatment will be provided in section 6.

Remark 1. It is possible to do away with occurrences of δ in sets $\mathcal{O}_1[\![s]\!]$ in the case an alternative for the failure is available. Technically, this is achieved by imposing the axiom

$$\{\delta\}\cup X=X, \quad X\neq\varnothing \qquad (*)$$

In the above example applying the axiom would turn the sets $\{\delta\}$, $\{\delta,\tau\}$, and $\{ab,a\delta\}$ into $\{\delta\}$, $\{\tau\}$ and $\{ab\}$, respectively. (For the latter case we take $\{ab,a\delta\} = a.(\{b\}\cup\{\delta\}) = a.\{b\} = \{ab\}$.) The reader is, of course, free to impose (*) throughout section 4. Our reason for not doing this is that our main result relating $\mathcal{O}_1$ and $\mathcal{D}_1$ does not depend on it. For both $\mathcal{O}_1$ and $\mathcal{D}_1$, (*) may or may not be imposed simultaneously without affecting the result of section 4.3.

Remark 2. Clearly, by taking $C=\varnothing$ the semantics $\mathcal{O}_1$ coincides with the previous $\mathcal{O}_0$.

4.2. Denotational semantics.

This is as in section 3.3, but with the definition of $\|$ extended to include both shuffle and communication, in the following way: the operator $\| : \mathbb{S}_{nc}(\delta)\times\mathbb{S}_{nc}(\delta)\rightarrow\mathbb{S}_{nc}(\delta)$ is defined by

1. For $X,Y\subseteq A^*\cup A^\infty\cup\{\perp,\delta\}$ we define
   $X\|Y=(X\mathbb{L}Y)\cup(Y\mathbb{L}X)\cup(X\mid Y)$ where
   (i) $X\mathbb{L}Y=\bigcup\{u\mathbb{L}Y : u\in X\}$, where $\perp\mathbb{L}Y=\{\perp\}$, $\delta\mathbb{L}Y=\{\delta\}$, $\epsilon\mathbb{L}Y=Y$, $(a.w)\mathbb{L}Y=a.(\{w\}\|Y)$
   (ii) $X\mid Y=\bigcup\{u\mid v : u\in X, v\in Y\}$, where $(c.u_1)\mid(\overline{c}.v_1)=\tau.(\{u_1\}\|\{v_1\})$ and $u\mid v=\varnothing$, for u,v not of this form
2. For X or Y with infinite words, we define

$$X\|Y=\lim_n(X(n)\|Y(n))$$

where $X(n), Y(n)$ are, as before, the set of all n-prefixes of elements in X and Y.

Example. $\{ac\}\|\{b\overline{c}\}=\{ac\}\mathbb{L}\{b\overline{c}\}\cup\{b\overline{c}\}\mathbb{L}\{ac\}\cup\{ac\}\mid\{b\overline{c}\}=$
$a.(\{c\}\|\{b\overline{c}\})\cup b.(\{\overline{c}\}\|\{ac\})\cup\varnothing=a.((\{c\}\mathbb{L}\{b\overline{c}\})\cup(\{b\overline{c}\}\mathbb{L}\{c\})\cup(c\mid b\overline{c}))\cup b.(\cdots)=$
$a.(\{cb\overline{c}\}\cup b.(\{\overline{c}\}\|\{c\})\cup\varnothing)\cup b.(\cdots)=\{acb\overline{c},ab\overline{c}c,abc\overline{c},ab\tau\}\cup\{b\overline{c}ac,bac\overline{c},ba\overline{c}c,ba\tau\}$.

Remark. The definition of $X \| Y$ just given is a smooth extension of the one given in section 3.3. A definition based on the technique with (inverse) homomorphisms as in section 2.4 would be much less perspicuous.

The definition of $\mathcal{D}_1$ is now as follows: Let $\Gamma_1 = Stmv \to \mathbb{S}_{nc}(\delta)$, and let $\gamma \in \Gamma_1$. We define

$$\mathcal{D}_1 : \textit{guarded } L_1 \to (\Gamma_1 \to \mathbb{S}_{nc}(\delta))$$

in

DEFINITION 4.2.

1 $\mathcal{D}_1[\![a]\!](\gamma) = \{a\}$, (for $a \in A \setminus C$)

2 $\mathcal{D}_1[\![c]\!](\gamma) = \{c\}$, (for $c \in C$)

3-5 as clauses 2-4 in definition 3.10. The convention about dropping γ for syntactically closed s in $\mathcal{D}_1[\![s]\!](\gamma)$ is again adopted.

It is important to observe that $\mathcal{D}_1[\![s]\!]$ delivers no δ (for any s). The compositional definition of $\mathcal{D}_1[\![s_1 \| s_2]\!]$ does not allow to define $\mathcal{D}_1[\![c]\!]$ as $\{\delta\}$! In fact, $s_1 \| s_2$ would then miss the opportunity to synchronize between its two operands. More will be said about this in the next section.

4.3. Relationship between $\mathcal{O}_1$ and $\mathcal{D}_1$.

Here we do not simply have that

$$\mathcal{O}_1[\![s]\!] = \mathcal{D}_1[\![s]\!] \qquad (*)$$

holds for all guarded statements $s \in L_1$. As a simple counterexample take $s = c$. Then $\mathcal{O}_1[\![c]\!] = \{\delta\}$ but $\mathcal{D}_1[\![s]\!] = \{c\}$. Even worse, we can state:

LEMMA 4.3. There does not exist any denotational (implying compositional) semantics $\mathcal{D}$ satisfying (*).

The proof is based on

LEMMA 4.4. $\mathcal{O}_1$ does not behave compositionally over $\|$, i.e. there exists no "semantic" operator

$$\|^{\mathcal{D}} : \mathbb{S}(\delta) \times \mathbb{S}(\delta) \to \mathbb{S}(\delta)$$

such that

$$\mathcal{O}_1[\![s_1 \| s_2]\!] = \mathcal{O}[\![s_1]\!] \|^{\mathcal{D}} \mathcal{O}[\![s_2]\!]$$

holds for all (guarded) $s_1, s_2 \in L_1$

PROOF Consider $s_1 = c$ and $s_2 = \overline{c}$ in L_1. Than $\mathcal{O}_1[\![s_1]\!] = \mathcal{O}_1[\![s_2]\!] = \{\delta\}$. Suppose now that $\|^{\mathcal{D}}$ exists. Then $\{\delta\} = \mathcal{O}_1[\![s_1 \| s_1]\!] = \mathcal{O}_1[\![s_1]\!] \|^{\mathcal{D}} \mathcal{O}_1[\![s_1]\!] = \mathcal{O}_1[\![s_1]\!] \|^{\mathcal{D}} \mathcal{O}_1[\![s_2]\!] = \mathcal{O}_1[\![s_1 \| s_2]\!] = \{\delta, \tau\}$.

Contradiction. □

We remedy this not by redefining T_1 (which adequately captures the operational intuition for L_1), but rather by introducing an *abstraction operator* $\alpha_1 : \mathbb{S}(\delta) \to \mathbb{S}(\delta)$ such that

$$\mathcal{O}_1[\![s]\!] = \alpha_1(\mathcal{D}_1[\![s]\!]) \qquad (**)$$

holds for guarded $s \in L_1$. We take $\alpha_1 = restr_{\mathbb{S}}$ which for $W \in \mathbb{S}(\delta)$ is defined by

$$restr_{\mathbb{S}}(W) = \{w \mid w \in W \text{ does not contain any } c \in C\}$$

$$\cup \{w.\delta \mid \exists c', w' \in A^{st}(\delta): w.c'.w' \in W \text{ and } w \text{ does not contain any } c \in C\}$$

Informally, $restr_{\mathbb{S}}$ replaces all unsuccessful synchronizations by failure or deadlock. It thus resembles the restriction operator in CCS [Mi2].

But how to prove (**)? Note that we cannot prove (**) directly by structural induction on s because $\alpha_1 = restr_{\mathbb{S}}$ does not behave compositionally (over $\|$) due to lemma 4.4. Our solution to this problem is to introduce a new *intermediate operational semantics* $\mathcal{O}_1{}^*$ such that we on the one hand show

$$\mathcal{O}_1[\![s]\!] = restr_{\mathbb{S}}(\mathcal{O}_1{}^*[\![s]\!])$$

by purely operational, i.e. transition based arguments, and on the other hand show

$$\mathcal{O}_1{}^*[\![s]\!] = \mathcal{D}_1[\![s]\!]$$

for guarded s analogously to $\mathcal{O}_0[\![s]\!] = \mathcal{D}_0[\![s]\!]$ in section 3.4. Combining these two results we will obtain the desired relationship (**).

For $\mathcal{O}_1{}^*$ we modify the transition system T_1 into a system $T_1{}^*$ which is the same as T_1 except for the communication axiom which now takes the form:

(communication*)

$$< c, w > \to w.c$$

We base $\mathcal{O}_1{}^*$ on $T_1{}^*$ just as we based $\mathcal{O}_1$ on T_1.

Examples. $\mathcal{O}_1{}^*[\![c]\!] = \{c\}$, $\mathcal{O}_1{}^*[\![c \| \bar{c}]\!] = \{c\bar{c}, \bar{c}c, \tau\}$, $\mathcal{O}_1{}^*[\![(a;b) \cup (a;c)]\!] = \mathcal{O}_1{}^*[\![a;(b \cup c)]\!] = \{ab, ac\}$.

We first establish

THEOREM 4.5. $\mathcal{O}_1[\![s]\!] = restr_{\mathbb{S}}(\mathcal{O}_1{}^*[\![s]\!])$ for every $s \in L_1$.

The proof uses the following lemma which states the link between the underlying transition

systems T_1 and $T_1{}^*$.

LEMMA 4.6. For all $s \in L_1$, $s' \in L_1 \cup \{E\}$ and $w,w' \in (A \setminus C)^*$:
(i)$T_1 \vdash \langle s,w \rangle \rightarrow \langle s',w' \rangle$ iff $T_1{}^* \vdash \langle s,w \rangle \rightarrow \langle s',w' \rangle$
(ii)$T_1 \vdash \langle s,w \rangle \rightarrow \langle s',w\delta \rangle$ iff $\exists c \in C : T_1{}^* \vdash \langle s,w \rangle \rightarrow \langle s',wc \rangle$

For the proofs of lemma 4.6 and theorem 4.5 we refer to [BMOZ2]. Next we discuss:

THEOREM 4.7. $\mathcal{O}_1{}^*[\![s]\!] = \mathcal{D}_1[\![s]\!]$ for every syntactically closed and guarded $s \in L_1$.

Its proof has the same structure as that of '$\mathcal{O}_0[\![s]\!] = \mathcal{D}_0[\![s]\!]$' (theorem 3.12). In fact, the lemmas 3.14 to 3.17 also hold for $\mathcal{O}_1{}^*$, $\mathcal{D}_1$ and L_1 instead of $\mathcal{O}_0$, $\mathcal{D}_0$ and L_0, with identical proofs. We therefore mention here only the proof that $\mathcal{O}_1{}^*$ behaves compositionally over $\|$ (thereby extending lemma 3.13). More precisely, it can be shown

LEMMA 4.8. $\mathcal{O}_1{}^*[\![s_1 \| s_2]\!] = \mathcal{O}_1{}^*[\![s_1]\!] \|^{\mathcal{D}_1} \mathcal{O}_1{}^*[\![s_2]\!]$ for all $s_1,s_2 \in L_1$.

As an auxiliary tool we need a result recalling Apt's 'merging lemma' in [Ap2]:

LEMMA 4.9 (Synchronization). $\forall s_1,s_2 \in L_1 \ \forall s',s'' \in L_1 \cup \{E\} \ \forall w,w_1,w_2 \in A^*$

$T_1{}^* \vdash \langle s_1 \| s_2,w \rangle \rightarrow \langle s' \| s'',w\tau \rangle$ where the considered transition is a synchronization between s_1 and s_2 such that s' stems from s_1 and s'' stems from s_2
iff
$\exists c \in C$: $T_1{}^* \vdash \langle s_1,w_1 \rangle \rightarrow \langle s',w_1 c \rangle$ and $T_1{}^* \vdash \langle s_2,w_2 \rangle \rightarrow \langle s'',w_2 \bar{c} \rangle$

The proofs of lemma 4.9 and, especially, of lemma 4.8 take a lot of work for which we refer, once more, to [BMOZ2]. By combining theorems 4.5 and 4.6 we finally obtain our desired result:

THEOREM 4.10 $\mathcal{O}_1[\![s]\!] = restr_{\mathbb{S}}(\mathcal{D}_1[\![s]\!])$ for every syntactically closed and guarded $s \in L_1$.

5. SYNCHRONIZATION MERGE AND LOCAL NONDETERMINACY: TWO ORDER-THEORETIC MODELS

5.1. Introduction.

In this section we describe two further denotational semantics for L_1. The first is based on order-theoretic notions for the stream domain A^{st}, introducing an order both on streams and on (certain) sets of streams. This model is included in our presentation firstly since it fits into the tradition of denotational semantics of using order-theoretic models (see, e.g. [dB] or [St]), secondly since it provides a slight improvement over the metric case in that at least some -though not very informative- meaning is assigned to unguarded statements. Thirdly, the order-theoretic stream model is motivated since it provides a nice isomorphism with our second order-theoretic model. This one is based solely on (sets of) *finite observations*. It fits into the specification oriented approach to the semantics of concurrent statements [OH 1,2], a generalization of the specific failure semantics of [BHR]. An *observation* is a finitely representable piece of information about the computational behaviour of a process. Examples of observations are (finite) traces, traces with divergence information, ready pairs and failure pairs leading to the (increasingly sophisticated) trace, divergence, readiness and failure semantics for concurrent statements [OH2]. The specific observation semantics to be presented in section 5.3 can be seen as 'in between' the divergence and readiness semantics of [OH2].

In section 5.4 we establish the relationships between the metric and the two order-theoretic models. Let us introduce, for the remainder of section 5, the following notation: for the three respective denotational meanings of statement $s \in L_1$, we write $\mathcal{M}[\![s]\!]$ (rather than the previous $\mathcal{D}_1[\![s]\!]$) for the metric denotational meaning of s, $\mathbb{S}[\![s]\!]$ for the order-theoretic meaning referring to the stream model ($\mathbb{S}$ for Smyth which is the order to be employed (cf. definition 2.15)) and $\mathcal{F}[\![s]\!]$ for the order-theoretic meaning referring to the finite observations model. The following results relating the various semantics are available:

1 For $s \in L_1$, s syntactically closed and guarded, $\mathcal{M}[\![s]\!] = \mathbb{S}[\![s]\!]$.
2 The stream model equipped with the Smyth order is isomorphic with the finite observations model equipped with the order of reverse set-inclusion.
3 For each syntactically closed $s \in L_1$, $\mathbb{S}[\![s]\!] = \mathcal{F}[\![s]\!]$.

We shall provide pertinent definitions and an outline of the proofs of these results, together with a few general facts relating order and metric which also have somewhat wider scope.

The main sources for section 5 are the papers [BMO1] (preliminary version) and [BMO2] (full version). These in turn rely heavily on the papers [Me1,Me2] and [OH2]. A few new results are furthermore taken from [BMO3].

5.2 The order-theoretic stream model.

We first define an order on $A^{st}=A^* \cup A^\omega \cup A^*.\{\bot\}$. (We use A^{st} rather than $A^{st}(\delta)$ since δ appears only in operational meanings or after abstraction, a notion not studied in section 5.) We use the following terminology for elements u in A^{st}:

- u is finite iff $u \in A^*$
- u is infinite iff $u \in A^\omega$
- u is unfinished iff $u \in A^*.\{\bot\}$

We now give

DEFINITION 5.1. $u \sqsubseteq v$ iff

- u is finished or infinite then $u=v$
- if u is unfinished, i.e. of the form $u=u'\bot$, then $u' \leqslant v$.

Examples. $a\bot \sqsubseteq a$, $a\bot \sqsubseteq ab$, $a^n\bot \sqsubseteq a^\omega$, but $a \not\sqsubseteq a\bot$.

PROPOSITION 5.2.
$(A^{st}, \sqsubseteq, \bot)$ is a cpo.

We next introduce, for $u \in A^{st}$, the $\sqsubseteq$-truncation $u[n]$ (which is a variation on the $\leqslant$-prefix $u(n)$) as follows: if $|u| \leqslant n$ we put $u[n]=u$, if $|u| \geqslant n$ and $|u'| \leqslant u$ is such that $|u'|=n$ we put $u[n]=u'\bot$. Moreover, we define $X[n]=\{u[n]: u \in X\}$.

We recall from section 2.3 the definition of Smyth (pre) order: $X \sqsubseteq_S Y$ iff $\forall v \in Y \exists u \in X[u \sqsubseteq v]$. For example, $X \sqsubseteq_S Y$ when $X \supseteq Y$ ($\supseteq$ denoting set-containment). $\sqsubseteq_S$ is not antisymmetric and therefore, in general, not a partial order. However, the Smyth preorder generates an equivalence relation $\equiv_S$: $X \equiv_S Y$ iff $X \sqsubseteq_S Y$ and $Y \sqsubseteq_S X$. What are the sets identified by $\equiv_S$?

DEFINITION 5.3. $MIN_S(X)=\{v \in X: \neg\exists u \in X[u \sqsubseteq v \wedge u \neq v]\}$ is the set of *minimal* streams in X.

Then $X \equiv_S Y$ iff $MIN_S(X)=MIN_S(Y)$. Thus, the sets $MIN_S(X)$ form a system of representatives of the equivalence classes under $\equiv_S$. Note that MIN_S is flat: recall definition 2.13a and the notation $\mathcal{P}_f(\cdots)$ for the flat subsets of $\cdots$.

PROPOSITION 5.4.
a. $\mathcal{P}(A^{st}) / \equiv_S$ is isomorphic to $\mathcal{P}_f(A^{st})$.
b. $(\mathcal{P}_f(A^{st}), \sqsubseteq_S, \{\bot\})$ is a cpo.
The proof of part b can be found in [Me1,Me2].
Notation. For the least upper bound of a Smyth-ordered chain of sets $<X_i>_i$ we shall write $\sqcup_{S,i} X_i$.
The cpo $\mathcal{P}_f(A^{st})$ will be the starting point for our order-theoretic stream semantics. In order to

be sure that we stay within our domain, we provide an adapted version of the operators of concatenation, union and merge which ensures that the outcome is always a flat set:

DEFINITION 5.5. Let $op \in \{ \; ; \; , \; \cup \; , \; \| \; \}$, and let $op^{\S}$ stand for the operators adapted to the Smyth framework.

a. For $X, Y \subseteq A^* \cup A^{*\infty}.\{\perp\}$ we define $X ;^{\S} Y = MIN_S(X.Y)$, where . is the operation as defined in section 3.

$X \cup^{\S} Y = MIN_S(X \cup Y)$ where $\cup$ is the set-theoretic union.

$X \|^{\S} Y = \min_S(X \| Y)$ where $\|$ is as defined in section 4.

b. For X and Y with infinite words we define

$X op^{\S} Y = \bigsqcup_{S,n}(X[n] op^{\S} Y[n])$

Example. Take $X = \{\perp\}$, $Y = \{ab\}$. Then $X \cup Y = \{\perp, ab\}$, $X \| Y = \{\perp, a\perp, ab\perp\}$. However $X \cup^{\S} Y = X \|^{\S} Y = \{\perp\}$.

PROPOSITION 5.6.

The semantic operators

$$op^{\S} : \mathcal{P}_f(A^{st}) \times \mathcal{P}_f(A^{st}) \to \mathcal{P}_f(A^{st})$$

with $op \in \{ \; ; \; , \; \cup \; , \; \| \; \}$, are well defined and monotonic.

For the proof see [Me1,Me2]. Showing monotonicity is not trivial for ; and $\|$. To provide meaning to recursive constructs as well, we will have to show that the semantic operators $op^{\S}$ are continuous. For $\cup^{\S}$ this is easily seen. Unfortunately, the operators $;^{\S}$ and $\|^{\S}$ are not continuous on arbitrary flat sets of streams.

Counterexamples. Take $X = \{a^{\omega}\}$ and

$$X_n = \{u \in a^* : |u| \geq n\}, n \geq 0$$

Clearly, $< X_n >_n$ is a $\supseteq$-chain and, hence a $\sqsubseteq_S$ chain. Note that $\bigsqcup_{S,n} X_n = \varnothing$, whereas both $\bigsqcup_{S,n}(X_n ;^{\S} X) \neq \varnothing$ and $\bigsqcup_{S,n}(X_n \|^{\S} X) \neq \varnothing$. Thus

$$\bigsqcup_{S,n}(X_n op^{\S} X) \neq (\bigsqcup_{S,n} X_n) op^{\S} X$$

for both $op \in \{ \; ; \; , \; \| \; \}$.

To rescue the continuity of $;^{\S}$ and $\|^{\S}$ we shall restrict our domain to (nonempty) *closed* and flat sets of streams. We recall the definitions of closed in a metric setting (definition 2.2d) and in a cpo setting (2.14b). Fortunately no confusion arises since we have

LEMMA 5.7. Consider the set of streams $A^{st} = (A^{st}, \sqsubseteq, \perp, d)$ simultaneously as a metric space and as a cpo. A subset $X \subseteq A^{st}$ is d-closed whenever it is $\sqsubseteq$-closed.

Let $\mathcal{P}_{ncf}(A^{st})$ stand for the set of all nonempty closed and flat subsets of A^{st}. The following lemma is crucial for the further development:

LEMMA 5.8. If $<X_n>_n$ is a $\sqsubseteq_S$-chain of sets in $\mathcal{P}_{ncf}(A^{st})$ then $\sqcup_{S,n} X_n \neq \emptyset$.

The proof is rather involved [Me1,Me2]. We can now establish the following results:

PROPOSITION 5.9.

1 $(\mathcal{P}_{ncf}(A^{st}), \sqsubseteq_S, \{\perp\})$, where "closed" is with respect to the metric d as before, is a cpo.

2 The operators $;^{\mathcal{S}}$ and $\|^{\mathcal{S}}$ when restricted to $\mathcal{P}_{ncf}(A^{st})$ are continuous under $\sqsubseteq_S$.

The proof uses lemma 5.8 and otherwise follows [BBKM]; the case of $\|^{\mathcal{S}}$ is difficult.

Remarks

1. Lemma 5.8 and proposition 5.9.1 do not hold, in general, for *infinite* sets A of actions.
2. Instead of using the Smyth order $\sqsubseteq_S$ it is also possible to use the Egli-Milner order $\sqsubseteq_{E.M}$. In fact we have that $(\mathcal{P}_{ncf}(A^{st}), \sqsubseteq_{E.M}, \{\perp\})$ is a cpo. This is a result from [Ba]; here for the completeness already the restriction to closed sets is necessary. An advantage of the Egli-Milner model is that continuity of the operators holds independently of the finiteness of A. We here prefer the Smyth order because it is isomorphic to the finite observations model to be described in a moment. For both cpo's $(\mathcal{P}_{ncf}(A^{st}), \sqsubseteq_S, \{\perp\})$ and $(\mathcal{P}_{ncf}(A^{st}), \sqsubseteq_{E.M}, \{\perp\})$ it would be interesting to sort out the precise relationship with the general powerdomain constructions of Plotkin [Pl1], Smyth [Sm] or Scott [Sc], et al. We expect that our Smyth order based cpo is isomorphic to the general Smyth powerdomain over A^{st}, assuming the (usual) finiteness of A. Closed sets are then compact and the elements of $\mathcal{P}_{ncf}(A^{st})$ are candidates for a set of representatives of the elements in the general Smyth powerdomain. For the Egli-Milner order and its related general Plotkin powerdomain further analysis is needed.

We are now in a position to define the first order-theoretic denotational semantics for $s \in L_1$. Let, in the standard way, the set of environments Γ now be given by $\Gamma = Stmv \to \mathcal{P}_{ncf}(A^{st})$, and let $\gamma \in \Gamma$.

DEFINITION 5.10. The semantic mapping $\mathcal{S} : L_1 \to (\Gamma \to \mathcal{P}_{ncf}(A^{st}))$ is given by

1. $\mathcal{S}[\![a]\!](\gamma) = \{a\}$, $\mathcal{S}[\![c]\!](\gamma) = \{c\}$
2. $\mathcal{S}[\![s_1 \ op \ s_2]\!](\gamma) = \mathcal{S}[\![s_1]\!](\gamma) op^{\mathcal{S}} \mathcal{S}[\![s_2]\!](\gamma)$
3. $\mathcal{S}[\![x]\!](\gamma) = \gamma(x)$
4. $\mathcal{S}[\![\mu x[s]]\!](\gamma) = \sqcup_{S,n} X_n$, where $X_0 = \{\perp\}$, $X_{n+1} = \mathcal{S}[\![s]\!](\gamma < X_n / x >)$.

The reader should observe that, contrary to the domain of the function $\mathfrak{M} = \mathfrak{D}_1$ of section 4.2, we have here defined $\mathcal{S}[\![s]\!]$ for *all* $s \in L_1$, thus including the unguarded statements as well. However, this is not too informative. It is, in fact, not difficult to show:

PROPOSITION 5.11.
For $s \in L_1$ closed, we have $\mathcal{S}[\![s]\!](\gamma)=\{\perp\}$ iff s is unguarded.

Example. $\mathcal{S}[\![\mu x[x]]\!]=\mathcal{S}[\![\mu x[a\|x]]\!]=\mathcal{S}[\![\mu x[(x;a)\cup b]]\!]=\{\perp\}$.

Remarks

1 We needed flat sets to turn $\sqsubseteq_S$ into an order. Therefore, we flattened all outcomes of the operators $op^{\mathcal{S}}$, with the above effect for unguarded s.

2 The result $\mathcal{S}[\![s]\!]=\{\perp\}$ for unguarded s should be contrasted with the outcome of $\mathcal{O}_i[\![s]\!]$, $i=0,1$. For example, as we saw in section 3.2, $\mathcal{O}_0[\![\mu x[(x;a)\cup b]]\!]=\{\perp\}\cup ba^*$, a result which is clearly far more satisfactory. Broy [Bro1] has developed a denotational semantics for guarded and unguarded statements (in L_0) which yields the same result as our $\mathcal{O}_0$. However, his semantics uses three consecutive steps for obtaining the result and is based on a domain with several orderings. It is, as far as we know, still an open problem whether there exists some *direct* way, based upon a domain with one ordering, to assign a denotational meaning, say $\mathcal{X}$, to s such that $\mathcal{O}[\![s]\!]=\mathcal{X}[\![s]\!]$ for all syntactically closed s (in L_0 or L_1).

3 Though somewhat lacking in content, the outcome of $\mathcal{S}[\![s]\!]$ for unguarded s is more attractive than the result in the linear time semantics of [BBKM]. This uses a simplified order-theoretic model, viz the cpo $(\mathcal{P}_{nc}(A^\dagger),\supseteq,A^\dagger)$ where $\supseteq$ is set-containment. The underlying mathematics is somewhat simpler, but this model has the disadvantage that counterintuitive results are deliverd for unguarded statements. E.g., $LT[\![\mu x[x]]\!](\gamma)=A^\dagger$ but (surprisingly) $LT[\![\mu x[x;b]]\!](\gamma)=A^\omega$. Moreover, it lacks the advantage of the model based on $\sqsubseteq_S$, viz. that it is isomorphic to the observation model we now present:

5.3. The finite observations model.

Motivated by the failure semantics of [BHR], a new approach to the semantics of concurrent statements has been developed in [OH 1,2]. It is called "specification oriented" because it starts from the following simple concept of correctness for (concurrent) statements: a statement satisfies a specification S, abbreviated s **sat** S, if every observation we can make about s is allowed by S. The idea is that by varying the structure of observations we can express various types of semantics and correctness in a uniform way.

The principles of specification-oriented semantics are:

- an observation is a finitely representable unit of information about the operational behaviour of statements
- therefore the set of possible observations about a statement enjoys some natural closure properties with respect to certain predecessor and successor observations
- sets of observations are ordered by nondeterminism ordering (reverse set-inclusion) [BHR]
- this ordering leads to a simple mathematics, in particular a very simple continuity argument for the language operators.

Let us now start with an example of a semantics -not treated in [OH 2]- which fits into this

framework. We use distinct symbols $\surd, \uparrow \notin A$ to define the following set A^{fo} of observations, with $h \in A^{fo}$:

DEFINITION 5.12. $A^{fo} = A^* \cup A^*.\{\surd, \uparrow\}$

Observations are finite *traces* or *histories* over A and the extra symbols $\surd$ and $\uparrow$, representing *successful termination* [BHR] and *divergence* [OH 2], respectively. Divergence $\uparrow$ stands for the infinite internal loop of a process generated by unguarded recursion like $\mu x[x]$. Thus, in spite of their finite representation, not all observations can be made effectively; a similar concession is also present in the concept of testing due to [dNH].
Just as with streams we let ϵ denote the empty history and $\leqslant$ the prefix relation between histories. Apart from $\leqslant$ we do not introduce any further relation on A^{fo} which corresponds to $\sqsubseteq$ on A^{st}. Let $\mathcal{P}(A^{fo})$ denote the powerset of A^{fo}, with $H \in \mathcal{P}(A^{fo})$.

DEFINITION 5.13. $H \subseteq A^{fo}$ is called *saturated* if the following holds:

1. H includes the *least* observation, i.e. $\epsilon \in H$
2. H is *prefix closed,* i.e. $h \in H$ and $h' \leqslant h$ imply $h' \in H$.
3. H is *extensible,* i.e. $h \in H \setminus A^*.\{\surd\}$ implies $\exists \alpha \in A \cup \{\surd, \uparrow\} : h\alpha \in H$
4. H treats divergence as *chaos,* i.e. $h\uparrow \in H$ and $h' \in A^{fo}$ imply $hh' \in H$.

These closure properties are (partly) motivated by looking at saturated H's as the sets of possible observations about a concurrent statement:

1. As long as the statement has not yet started, we only observe the empty history ϵ.
2. Whenever we have observed a history h, also all its prefixes are observable.
3. Only histories $h\surd$ indicate the successful termination of the observed statement; for all other histories h some *extension* $\alpha \in A \cup \{\surd, \uparrow\}$ is certain to happen, but we do not know which one, by looking at h.
4. Identifying divergence $h\uparrow$ after a history h with *chaotic closure* $h.A^{fo}$ cannot be explained operationally. Rather, it originates from the desire to ban diverging processes from satisfying any reasonable specification. This idea is familiar from Dijkstra's weakest precondition semantics where a diverging program will not achieve any postcondition [Pl2].

Properties (i), (ii) are typical conditions on traces to be found in [BHR,FLP,OH 1,2]. Property (iii) is a new 'linear' version of the extensibility condition in the readiness [OH2] or failure semantics [BHR]. Property (iv) is typical for a simple but proper treatment of divergence [OH2]; without $\uparrow$ unsatisfactory results occur [BHR] akin to those in the LT semantics of [BBKM] (cf. remark 3 at the end of section 5.2 above).

DEFINITION 5.14.

a. $\mathcal{P}_{sat}(A^{fo})$ is the set of all saturated subsets of A.

b. The nondeterminism order $\sqsubseteq$- on $\mathcal{P}_{sat}(A^{fo})$ is defined by $H_1 \sqsubseteq\text{-} H_2$ iff $H_1 \supseteq H_2$.

PROPOSITION 5.15.
$(\mathcal{P}_{sat}(A^{fo}), \supseteq, A^{fo})$ is a cpo.

Remark. Finiteness of A is necessary for proposition 5.15. For example, let $A_i = \{a_i, a_{i+1}, \cdots\}, i = 0,1, \cdots$ be infinite alphabets, and consider the chain of saturated sets $A_0{}^* \supseteq A_1{}^* \supseteq \cdots$. Then $\cap_i A_i{}^* = \{\epsilon\}$, which is not a saturated set since $\{\epsilon\}$ does not satisfy the extensibility property.

The relationship between $(\mathcal{P}_{ncf}(A^{st}), \sqsubseteq_S, \{\perp\})$ and $(\mathcal{P}_{sat}(A^{fo}), \supseteq, A^{fo})$ will be completely settled in section 5.4. In fact, we shall exhibit a continuous isomorphism between the two cpo's. Before coming to this we proceed with the introduction of the finite observations semantics $\mathcal{F}$. We first introduce the familiar operators $op^{\mathcal{F}}$, $op \in \{;, \cup, \|\}$, where the superscript $\mathcal{F}$ indicates that we now work in the domain $\mathcal{P}_{sat}(A^{fo})$.

DEFINITION 5.16.

a. $H_1 ;^{\mathcal{F}} H_2 =$
$\{h_1 : h_1 \in H_1 \text{ and } h_1 \text{ does not contain } \surd\} \cup$
$\{h_1 h_2 : h_1 \surd \in H_1 \text{ and } h_2 \in H_2\} \cup$
$\{h_1 h : h_1 \uparrow \in H_1 \text{ and } h \in A^{fo}\}$

b. $H_1 \cup^{\mathcal{F}} H_2 = H_1 \cup H_2$

c. $H_1 \|^{\mathcal{F}} H_2 = \{h : \exists h_1 \in H_1, h_2 \in H_2 \quad [h \in h_1 \| h_2]\}$ where $h_1 \| h_2 = (h_1 \mathbin{\mathbb{L}} h_2) \cup (h_2 \mathbin{\mathbb{L}} h_1) \cup (h_1 \mid h_2)$, and where $\epsilon \mathbin{\mathbb{L}} \epsilon = \{\epsilon\}$, $(ah_1) \mathbin{\mathbb{L}} h_2 = a.(h_1 \| h_2)$, $\surd \mathbin{\mathbb{L}} h_2 = \{h_2\}$, $\uparrow \mathbin{\mathbb{L}} \epsilon = A^{fo}$, $(ch_1) \mid (\overline{c}h_2) = \tau.(h_1 \| h_2)$ and, finally, $h_1 \mathbin{\mathbb{L}} h_2 = \varnothing$ and $h_1 \mid h_2 = \varnothing$ for all h_1, h_2 not of the indicated form.

In the finite observations model a fundamental result is available which substantially simplifies the continuity proofs of the above operators. This result is in fact quite general and independent of the specific structure of the observations. Consider two sets $\mathbb{X}, \mathbb{Y}$ and a relation $R \subseteq \mathbb{X} \times \mathbb{Y}$. Then R induces an operator

$$op_R : \mathcal{P}(\mathbb{X}) \to \mathcal{P}(\mathbb{Y})$$

on the subsets of $\mathbb{X}$ by taking for every $X \subseteq \mathbb{X}$ the pointwise image under R, i.e.

$$op_R(X) = \{y \in \mathbb{Y} : \exists x \in X [xRy]\}$$

PROPOSITION 5.17 [OH2].
The operator op_R is $\supseteq$-monotonic. Moreover, if R is domain finite, i.e. for every $y \in \mathbb{Y}$ there exist only finitely many $x \in \mathbb{X}$ with xRy, then op_R is also $\supseteq$-continuous.

We now have

PROPOSITION 5.18.
The operators $op^{\mathcal{F}}$, $op \in \{;, \cup, \|\}$ are monotonic and continuous.

We discuss only the proof for $;^{\mathcal{F}}$ ($\cup^{\mathcal{F}}$ is easy, $\|^{\mathcal{F}}$ is a slight variation). Take $\mathbb{X} = A^{fo} \times A^{fo}$, $\mathbb{Y} = A^{fo}$. Now we look for a domain finite relation $R \subseteq \mathbb{X} \times \mathbb{Y}$ such that

$$;^{\mathcal{F}} = op_R \lceil (\mathcal{P}_{sat}(A^{fo}) \times \mathcal{P}_{sat}(A^{fo})), \qquad (*)$$

where $\lceil \cdots$ denotes restriction to $\cdots$.
R can be read off from $;^{\mathcal{F}}$ as follows: $(h_1, h_2) R h$ iff

1. h_1 does not contain $\surd$, $h_2 = \epsilon$ and $h = h_1$, or
2. h_1 ends in $\surd$ and $h = (h_1 \setminus \surd).h_2$, or
3. h_1 ends in $\uparrow$, $h_2 = \epsilon$ and $h \in (h_1 \setminus \uparrow).A^{fo}$.

Here $h_1 \setminus \surd$ and $h_1 \setminus \uparrow$ result from h_1 by removing from h_1 the symbols $\surd$ and $\uparrow$, respectively. Clearly, R is domain finite, and we can apply the above fundamental result to obtain $\supseteq$-continuity.

Remarks

1 In the observation semantics the continuity proof for the operators $op_{\mathcal{F}}$ could be reduced to a simple test on domain finiteness. In the stream semantics (order-theoretic version) the operators $op^{\mathcal{S}}$, $op \in \{;, \|\}$ would *fail* such a test. For example, the infinite stream a^ω can originate from infinitely many pairs of streams u, v in the sense of both $u.v = a^\omega$ and $u \| v = a^\omega$. Thus finite observations are crucial here.

2 Another advantage of finite observations is that we can define the operators, in particular $\|^{\mathcal{F}}$, without reference to any semantic approximation of its arguments -unlike the stream operator $\|^{\mathcal{S}}$ where we put

$$X \|^{\mathcal{S}} Y = \sqcup_{S,n} (X[n] \| Y[n])$$

We can now define the denotational finite observations semantics $\mathcal{F}$. Again we use environments $\gamma \in \Gamma$, but now taking Γ as $Stmv \to \mathcal{P}_{sat}(A^{fo})$.

DEFINITION 5.19. The semantic mapping

$$\mathcal{F} : L_1 \to (\Gamma \to \mathcal{P}_{sat}(A^{fo}))$$

is given by

1. $\mathcal{F}[\![a]\!](\gamma) = \{\epsilon, a, a\surd\}$, and similarly with c replacing a.
2. $\mathcal{F}[\![s_1 \ op \ s_2]\!](\gamma) = \mathcal{F}[\![s_1]\!](\gamma) \ op^{\mathcal{F}} \mathcal{F}[\![s_2]\!](\gamma)$
3. $\mathcal{F}[\![x]\!](\gamma) = \gamma(x)$
4. $\mathcal{F}[\![\mu x[s]]\!](\gamma) = \sqcup_{S,n} X_n$, where $X_0 = A^{fo}$ and $X_{n+1} = \mathcal{F}[\![s]\!](\gamma < X_n / x >)$.

5.4. Relationship between metric and order-theoretic semantics.

We begin with two important general results relating order-theoretic and metric concepts.

PROPOSITION 5.20.

a. Let $<X_n>_n$ be a $\sqsubseteq_S$-chain of sets in $\mathcal{P}_{ncf}(A^{st})$. Then $<X_n>_n$ is also a Cauchy-sequence in $(\mathcal{P}_{nc}(A^{st}),\hat{d})$

b. Assume $<X_n>_n$ is both a $\sqsubseteq_S$-chain in $\mathcal{P}_{ncf}(A^{st})$ and (hence) a Cauchy-sequence in $(\mathcal{P}_{nc}(A^{st}),\hat{d})$. Then

$$\sqcup_{S,n} X_n = lim_n X_n$$

The proofs of these two statements are given in [BMO3]. Compactness (thanks to the finiteness of A) of (A^{st},d) is an important tool.

We now address the relationship between $\mathcal{M}[\![s]\!]$ and $\mathcal{S}[\![s]\!]$. By the restriction in the definition of $\mathcal{M}[\![s]\!]$, this relationship is meaningful only for s guarded.

Two subquestions have clearly to be settled. Firstly, what is the relationship between the operators $op^{\mathcal{M}}(=op^{\mathcal{P}_1})$ and $op^{\mathcal{S}}$ and, secondly, how do the outcomes of $\mathcal{M}[\![s]\!]$ and $\mathcal{S}[\![s]\!]$ relate for s a recursive statement. As to the first subquestion, we recall that, in general, $Xop^{\mathcal{M}}Y$ does not deliver a flat set whereas $Xop^{\mathcal{S}}Y$ does. (Example: $\{\bot\}\|^{\mathcal{M}}\{ab\} = \{\bot, a\bot, ab\bot\} \neq \{\bot\} = \{\bot\}\|^{\mathcal{S}}\{ab\}$.)

The next proposition settles the questions:

PROPOSITION 5.21 [BMO3].

a. For $X,Y \in \mathcal{P}_{nc}(A^{st})$, $MIN_S(Xop^{\mathcal{M}}Y) = MIN_S(X) op^{\mathcal{S}} MIN_S(Y)$

b. Assume s guarded. Let $var(s) = \{x_1, \ldots, x_n\}$.

If

$$MIN(X_i) = Y_i, \quad i = 1,\ldots,n$$

then

$$MIN(\mathcal{M}[\![s]\!](\gamma < X_i / x_i >_{i=1}^n)) = \mathcal{S}[\![s]\!](\gamma < Y_i / x_i >_{i=1}^n)$$

Part a is by definition satisfied for X,Y without infinite elements, and uses proposition 5.20b for X or Y with infinite elements. The proof of part b is by induction on the complexity of s, using part a and proposition 5.20b once more.

COROLLARY 5.22. For closed and guarded s, $\mathcal{M}[\![s]\!] = \mathcal{S}[\![s]\!]$.

The proof follows from the proposition (take n = 0) and the easily established fact that, for closed and guarded s, $\mathcal{M}[\![s]\!] \subseteq A^* \cup A^\omega$.

We next turn to the second relationship and investigate the connection between $\mathcal{S}[\![s]\!]$ and $\mathcal{F}[\![s]\!]$. Since these deliver outcomes in different domains, we cannot simply expect equality. Rather, we have to provide a more detailed analysis. First we establish the connection between $(\mathcal{P}_{ncf}(A^{st}),\sqsubseteq_S,\{\perp\})$ and $(\mathcal{P}_{sat}(A^{fo}),\supseteq,A^{fo})$. In fact, we have an important theorem which states that the two cpo's are isomorphic. Let the mapping $\Psi : A^{st}\to\mathcal{P}(A^{fo})$ be defined as follows: For $u\in A^*$ and $v\in A^\omega$ let

$$\Psi(u)=\{h\in A^*:h\leqslant u\}\cup\{u\surd\}$$

$$\Psi(v)=\{h\in A^*:h\leqslant v\}$$

$$\Psi(u\perp)=\{h\in A^*:h\leqslant u\}\cup\{uh:h\in A^{fo}\}$$

Remark. A finished stream u is translated to the set of all its prefixes plus $u\surd$ with $\surd$ signalling successful termination of u, an infinite stream is translated into the set of all its *finite* prefixes, and an unfinished stream $u\perp$ is translated into the set of all prefixes of u plus the chaotic closure $u.A^{fo}$ of divergence $u\uparrow$.

We extend Ψ pointwise to the mapping

$$\Psi : \mathcal{P}(A^{st})\to\mathcal{P}(A^{fo})$$

by defining

$$\Psi(X)=\bigcup_{w\in X}\Psi(w)$$

Examples. $\Psi(\{ab\})=\{\epsilon,a,ab,ab\surd\}$, $\Psi(\{a^\omega\})=\{a^n\mid n\geqslant 0\}$, $\Psi(\{\perp\})=A^{fo}$.

We now state

THEOREM 5.23. Ψ is a continuous isomorphism from the cpo $(\mathcal{P}_{ncf}(A^{st}),\sqsubseteq_S,\{\perp\})$ onto the cpo $(\mathcal{P}_{sat}(A^{fo}),\supseteq,A^{fo})$, i.e. Ψ is a bijection, yields $\Psi(\{\perp\})=A^{fo}$, it strongly preserves the partial orders:

$$X\sqsubseteq_S Y \text{ iff } \Psi(X)\supseteq\Psi(Y)$$

for all $X,Y\in\mathcal{P}_{ncf}(A^{st})$ and, finally, for $<X_n>_n$ a $\sqsubseteq_S$-chain in $\mathcal{P}_{ncf}(A^{st})$ we have

$$\Psi(\sqcup_{S,n}X_n)=\cap_n\Psi(X_n)$$

The proof is described in [BMO2].

Remarks. $\mathcal{P}_{ncf}(A^{st})$ has been constructed through a chain of clear domain theoretical notions: streams, sets of streams, Smyth order, flatness, continuity, topological closure, non-emptiness. The introduction of $\mathcal{P}_{sat}(A^{fo})$ with its saturation property may seem more ad hoc. But the theorem tells us that $\mathcal{P}_{sat}(A^{fo})$ can in fact be viewed as a *special representative* of the general

construct $\mathcal{P}_{ncf}(A^{st})$. This provides us with a new mutual understanding of the closedness properties in both domains: *topological closedness* on streams corresponds to taking *all finite prefixes* as observations, *flatness* of sets of streams corresponds to the *chaotic closure* on observations, *nonemptiness* of sets of streams does not simply correspond to the fact that saturated sets of observations include ϵ, but that in addition they are *extensible.* Whereas the nonemptiness of (lubs of) sets of streams is a global property, the extensibility of observations is a local property where every observation $h \notin A^*.\{\surd\}$ can be locally extended by another $\alpha \in A \cup \{\surd, \uparrow\}$. This issue of 'global' vs. 'local' hints at why it is more difficult to prove the cpo property for $\mathcal{P}_{ncf}(A^{st})$ than for $\mathcal{P}_{sat}(A^{fo})$.

We conclude this section with the theorem expressing equality of $\mathbb{S}$ and $\mathcal{F}$ when applied to $s \in L_1$. An important step is that Ψ is compatible with the language operators:

PROPOSITION 5.24.
For $op \in \{;, \cup, \|\}$ and $X, Y \in \mathcal{P}_{ncf}(A^{st})$ we have

$$\Psi(X \; op^{\mathbb{S}} \; Y) = \Psi(X) \; op^{\mathcal{F}} \; \Psi(Y)$$

The proof is, again, given in full in [BMO2]. Finally we have the desired result as

COROLLARY 5.25. For every $s \in L_1$ and $\gamma \in Stmv \to \mathcal{P}_{ncf}(A^{st})$,

$$\Psi(\mathbb{S}[\![s]\!](\gamma)) = \mathcal{F}[\![s]\!](\Psi \circ \gamma)$$

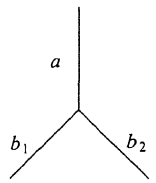
The proof follows by induction on the complexity of s, using proposition 5.24 and the continuity of Ψ.

6. SYNCHRONIZATION MERGE AND GLOBAL NONDETERMINACY: THE INTRODUCTION OF BRANCHING TIME

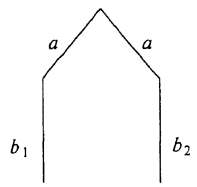
6.1. Introduction and operational semantics.

The central notion of section 6 is that of *global* nondeterminism. We introduce the language L_2 which is like L_1, but for the replacement of the latter's local nondeterminism ($s_1 \cup s_2$) by global nondeterminism, denoted by $s_1 + s_2$ (the notation "+" is from CCS [Mi2]).

Section 6 first brings the operational semantics for L_2 in terms of the transition system T_2. After the introduction of T_2 we also provide some explanatory comments on the difference between global and local nondeterminism. Next we present the (metric) denotational semantics for L_2. An essential difference with the denotational semantics for L_0 and L_1 is that we no longer employ the stream domain A^{st} but instead a domain which consists of tree-like objects, our so called *processes*. As we shall see, a process is a commutative (or unordered) tree with sets rather than multisets for successors of nodes (thus respecting the property that "+" is absorptive: $s + s = s$), and with a certain closure property. Trees exhibit branching behaviour: a typical example is the distinction between



and



Note that these two are identical as to their sets of paths ($\{ab_1, ab_2\}$ in both cases). This may explain the terminology 'Branching Time' (BT) for this denotational model. Branching time models preserve the moment of choice (first perform a, then choose between b_1 and b_2 versus choose between a and then b_1 or a and then b_2). Therefore, they are in particular pertinent in cases where deadlock is possible: consider the two statements $a;(b+c)$ and $(a;b)+(a;c)$ in a context where c has no matching communication $\overline{c}$ in a parallel statement. We shall design our model such that the first statement shows no deadlock (the failing c has an alternative b) and the second statement does have a deadlock possibility: After the choice for $a;c$, once a is executed no further action is possible. By way of contrast, we recall that for L_1 (with $\cup$ replacing $+$) we

obtain in both cases as outcome $\{ab, a\delta\}$ or, after possible simplification (cf. remark 1 in section 4.1), just $\{ab\}$.

The operational model still employs (sets of) sequences of actions. Thus, we have to be prepared for substantially more effort to relate $\mathcal{O}_2$ and $\mathcal{D}_2$ than with $\mathcal{O}_i$ and $\mathcal{D}_i$, $i=0,1$. Section 6.3 is devoted to this relationship. Just as for L_1 we define an abstraction operator α_2 such that (*) $\mathcal{O}_2 = \alpha_2 \circ \mathcal{D}_2$. However, both the definition of α_2 and the argument establishing (*) are much more involved than the corresponding ones for L_1. We shall in fact introduce *two* intermediate semantics, one operational ($\mathcal{O}_2$*) and one denotational ($\mathcal{D}_2$*) and, in addition, *four* auxiliary operators which play a role in the analysis of α_2. The definition of $\mathcal{O}_2$* is in particular interesting since it employs a version of the notion of ready set as encountered in papers such as [BHR, FLP, OH1, OH2, RB]; more detail will be supplied in section 6.3. The definition of $\mathcal{D}_2$* also refers to the model of ready sets, thus enabling a fairly direct proof that $\mathcal{O}_2^* = \mathcal{D}_2^*$. The remaining work has then to be spent on relating $\mathcal{O}_2$ and $\mathcal{O}_2$*, and $\mathcal{D}_2$ and $\mathcal{D}_2$*. Especially the latter connection requires a solid effort (to be found in [BMOZ2]).

The BT framework as presented in section 6.2 is a central notion in the research on modelling concurrency. It builds on the synchronization trees of CCS [Mi2], though the care we spend on the metric derivation of the defining domain equation is an issue not addressed in [Mi2]. (Early work of Milner such as [Mi1,MM] approaches domain equations for (communicating) processes in the style of Plotkin [Pl1] or Smyth [Sm].) In the work of Rounds et al ([GR,Ro]) the relationship between metric spaces and synchronization trees is further pursued. Moreover, the work of Rounds also introduces an *order* on processes. Its definition takes us into the realm of Scott's Information Systems ([Sc]), a territory with many ramifications left unexplored in our paper. Extensive investigations of processes with algebraic means have been reported by Bergstra, Klop and coworkers. We mention only [BK1, BK2, BK3]; in these and further papers a large variety of *process algebras* is defined and many applications are described.

After these introductory remarks we now present the syntax for L_2. Again, we use s to range over L_2.

DEFINITION 6.1

$$s ::= a \mid c \mid s_1;s_2 \mid s_1 + s_2 \mid s_1 \| s_2 \mid x \mid \mu x[s]$$

We now describe the transition system T_2. T_2 is like T_1, but without the axioms for local nondeterminacy and for communication ($<c,w> \rightarrow w\delta$). Instead we have as new rules

(global nondeterminacy)
[μ-unfolding]

$$\frac{<s_1,w>\rightarrow<s',w>}{\begin{array}{l}<s_1+s_2,w>\rightarrow<s'+s_2,w>\\ <s_2+s_1,w>\rightarrow<s_2+s',w>\end{array}}$$

Here the word on the right-hand side of the premise is equal to the word on the left-hand side ($=w$). This implies that the premise (and hence the conclusion) is a recursion transition.

[selection by action]

$$\frac{<s_1,w>\rightarrow<s',w'>}{\begin{array}{l}<s_1+s_2,w>\rightarrow<s',w'>\\ <s_2+s_1,w>\rightarrow<s',w'>\end{array}}$$

Here $w'=wa$ (and hence the premise is an elementary action transition) or $w'=w\tau$ (and hence the premise is a synchronization transition). Also s' may be E.

[selection by synchronization]

$$\frac{<s_1\|s_2,w>\rightarrow<s',w\tau>}{\begin{array}{l}<(s_1+s)\|s_2,w>\rightarrow<s',w\tau>\\ <(s+s_1)\|s_2,w>\rightarrow<s',w\tau>\\ <s_1\|(s_2+s),w>\rightarrow<s',w\tau>\\ <s_1\|(s+s_2),w>\rightarrow<s',w\tau>\end{array}}$$

where s' may be E and where the premise of the rule describes a synchronization transition between s_1 and s_2.

Note. We emphasize that the synchronization in a context rule(s) of T_1 remain valid.

Remarks. The essential difference between T_1 and T_2 is how communication is treated in the presence of nondeterminacy. For example, the L_1-statement

$$a \cup c$$

involving local nondeterminacy may choose "on its own" between a and c, i.e. in terms of T_1-transitions we have

$$<a \cup c,w>\rightarrow<a,w>,$$

$$<a \cup c,w>\rightarrow<c,w>.$$

The first alternative yields

$$<a,w>\rightarrow wa$$

whereas a communication can always deadlock in T_1:

$$<c,w>\rightarrow w\delta.$$

Contrast this behaviour with that of the L_2-statement

$$a + c$$

involving global nondeterminacy. The only transition possible is

$$<a + c, w> \rightarrow wa$$

(we say that the first alternative of $a + c$ is selected by the action a). In particular, a communication c in isolation does not produce anything in T_2. Only in cooperation with a matching communication $\overline{c}$ in another parallel component, c may produce a synchronization-transition:

$$<(a + c) \| \overline{c}, w> \rightarrow w\tau$$

(we say that the second alternative of $a + c$ is selected by the synchronization of c with $\overline{c}$).
This form of nondeterminacy is typical for languages like CSP [Ho], ADA [Ad] and Occam [In]. There the elementary action a corresponds to passing a true Boolean guard and the synchronization of c with $\overline{c}$ corresponds to matching communication guards in two parallel components. In the abstract setting of uniform concurrency global nondeterminacy has first been discussed by Milner [Mi2]. However, Milner takes from the very beginning a communication axiom corresponding (in our setting) to

$$(*) \quad <c, w> \rightarrow wc.$$

This enables him to state very simple transition rules for global nondeterminacy. We prefer not to adopt Milner's approach for T_2 because (*) does not correspond to the operational idea of CSP, ADA or Occam where a communication c proceeds only if a matching communication $\overline{c}$ is available.
Finally, note that in the case of a μ-term, global nondeterminacy $+$ allows us to unfold the recursion before selecting any alternative. For example,

$$<\mu x[a] + c, w> \rightarrow <a + c, w> \rightarrow wa$$

holds in T_2.

We continue with the description of the operational semantics $\mathcal{O}_2$ for L_2. $\mathcal{O}_2$ is a mapping $\mathcal{O}_2 : L_2 \rightarrow \mathbb{S}(\delta)$ with $\mathbb{S}(\delta) = \mathcal{P}(A^{st}(\delta))$ as for L_1. The definition of $w \in \mathcal{O}_2[\![s]\!]$ is as for $\mathcal{O}_0$ and $\mathcal{O}_1$, but with an additional third clause:

3. If there is a finite sequence of T_2-transitions

$$<s, \epsilon> = <s_0, w_0> \rightarrow \cdots \rightarrow <s_n, w_n>, \quad s_n \neq E$$

such that no further transition $<s_n, w_n> \rightarrow <s', w'>$ is deducible in T_2, we deliver $w = w_n\delta$ as an element of $\mathcal{O}_2[\![s]\!]$.

The pair $<s_n, w_n>$ in 3 is called a *deadlocking configuration*. Such configurations do not exist under T_0 or T_1. The following examples mark the differences from $\mathcal{O}_1$.
Examples. $\mathcal{O}_2[\![c]\!] = \{\delta\}$, $\mathcal{O}_2[\![c \| \overline{c}]\!] = \{\tau\}$, $\mathcal{O}_2[\![(a;b) + (a;c)]\!] = \{ab, a\delta\}$, $\mathcal{O}_2[\![a;(b + c)]\!] = \{ab\}$. Because it is important to see the difference between the last two examples, we shall show how they are

derived:

(i) $\mathcal{O}_2[\![(a;b)+(a;c)]\!]=\{ab,a\delta\}$.

Proof. Note that

$$<a;b,\epsilon>\rightarrow<b,a>\rightarrow ab$$

and

$$<a;c,\epsilon>\rightarrow<c,a>$$

are deducible. So by selection by elementary action we obtain also

$$<(a;b)+(a;c),\epsilon>\rightarrow ab$$

and

$$<(a;b)+(a;c),\epsilon>\rightarrow<c,a>.$$

So, since no further deductions can be made from $<c,a>$, we get, by the definition of $\mathcal{O}_2$: $\mathcal{O}_2[\![(a;b)+(a;c)]\!]=\{ab,a\delta\}$.

(ii) $\mathcal{O}_2[\![a;(b+c)]\!]=\{ab\}$.

Proof. First note that

$$<a;(b+c),\epsilon>\rightarrow<b+c,a>.$$

Since we have that

$$<b,a>\rightarrow ab,$$

we also have

$$<b+c,a>\rightarrow ab,$$

and therefore

$$<a;(b+c),\epsilon>\rightarrow ab.$$

Since we cannot deduce anything from $<c,a>$, ab is all we can deduce for $<a;(b+c),\epsilon>$. Consequently, $\mathcal{O}_2[\![a;(b+c)]\!]=\{ab\}$.

Thus with global nondeterminacy $+$, the statements $s_1=(a;b)+(a;c)$ and $s_2=a;(b+c)$ get different meanings under $\mathcal{O}_2$. The difference can be understood as follows: If s_1 performs the elementary action a, the remaining statement is either the elementary action b or the communication c. In case of c, a deadlock occurs since no matching communication is available. However, if s_2 performs a, the remaining statement is $b+c$ which cannot deadlock because the action b is always possible. Thus communications c create deadlocks only if neither a matching communication $\bar{c}$ nor an alternative elementary action b is available.

6.2. A branching time denotational semantics.

We follow [BZ1,BZ2,BBKM] in introducing a *branching time* semantics for L_2. Let, as usual, $\perp \notin A$ and let $A_\perp$ be short for $A \cup \{\perp\}$. Again, we assume a special element τ in A. Let the metric spaces $(\mathbb{P}_n, d_n)$, $n \geq 0$, be defined by

$$\mathbb{P}_0 = \mathcal{P}(A_\perp), \ \mathbb{P}_{n+1} = \mathcal{P}(A_\perp \cup (A \times \mathbb{P}_n))$$

where the metrics d_n will be defined in a moment. Let $\mathbb{P}_\omega = \bigcup_n \mathbb{P}_n$. Elements of $\mathbb{P}_\omega$ are called (finite) *processes,* and typical elements are denoted by $p,q, \cdots$. Processes $p,q \in \mathbb{P}_n$ are often denoted by $p_n, q_n, \cdots$. For $p \in \mathbb{P}_\omega$ we call the least n such that $p \in \mathbb{P}_n$ its *degree.* Note that each process is a *set;* hence, a process has elements for which we use $x,y, \cdots$ (not to be confused with $x,y \in Stmv$).

Examples of processes:

1. $\{\perp\}, \{[a,\{\perp\}]\}, \{[a,\{b_1\}],[a,\{b_2\}]\}, \{[a,\{b_1,b_2\}]\}$.
2. $p_0 = \{\perp\}, p_{i+1} = \{[a,p_i],[b,p_i]\}, i = 0,1, \cdots$.

For each p $(\in \mathbb{P}_\omega)$ we define its n-th projection $p(n) \in \mathbb{P}_n$ as follows:

$$p(n) = \{x(n) : x \in p\}, \ n = 0,1, \cdots$$

$$x(n) = x, \text{ if } x \in A_\perp, \ n = 0,1, \cdots$$

$$[a,p'](n) = \begin{cases} a, & n = 0 \\ [a,p'(n-1)], & n = 1,2, \cdots \end{cases}.$$

Examples of projections:

1. Let $p = \{[a,\{b_1\}],[a,\{b_2\}]\}, q = \{[a,\{b_1,b_2\}]\}$.
   Then $p(0) = \{[a,\{b_1\}](0),[a,\{b_2\}](0)\} = \{a\} = q(0) = \{[a,\{b_1,b_2\}](0)\}$, and $p(1) = p$, $q(1) = q$.
2. Let $p_0 = \{\perp\}$, $p_{i+1} = \{[a,p_i],[b,p_i]\}$. Then $p_0(k) = \{\perp\}$, $k = 0,1, \cdots$, and $p_{i+1}(0) = \{a,b\}, p_{i+1}(k+1) = \{[a,p_i(k)],[b,p_i(k)]\}$.

We can now define d_n by

$$d_0(p'_0,p''_0) = \begin{cases} 0, & \text{if } p'_0 = p''_0 \\ 1, & \text{if } p'_0 \neq p''_0 \end{cases}$$

$$d_{n+1}(p'_{n+1},p''_{n+1}) = 2^{-sup\{k : p'_{n+1}(k) = p''_{n+1}(k)\}+1}$$

with $2^{-\infty} = 0$ as before.

On $\mathbb{P}_\omega$ we define the metric d by putting $d(p,q) = d_n(p,q)$ if $n = max(degree(p), degree(q))$.

Example.

Let p,q,p_i be as in the example about projections. Then $d(p,q) = \frac{1}{2}$, $d(p_i,p_{i+1}) = 2^{-i}$.

We now define the set $\mathbb{P}$ of finite and infinite processes as the *completion* of $\mathbb{P}_\omega$ with respect to d. A fundamental result of [BZ2] is that we have the equality (more precisely, the isometry)

$$\mathbb{P} = \mathcal{P}_{closed}(A_\perp \cup (A \times \mathbb{P}))$$

Finite elements of $\mathbb{P}$ are the processes in the examples just given. An infinite element of $\mathbb{P}$ is, e.g., the process p which satisfies the equation $p = \{[a,p],[b,p]\}$. An explicit definition of this process would be: $p = lim_i p_i$, $p_0 = \{\perp\}$, $p_{i+1} = \{[a,p_i],[b,p_i]\}$. Processes are like commutative trees which have in addition sets rather than multisets for successors of nodes and which satisfy a closedness property. An example of a set which is not a process is $\{a, [a, \{a\}], [a, \{[a, \{a\}]\}], \cdots\}$ in the case that this set does not include the infinite branch of a's. Note that an attempt to obtain such an object as limit of finite approximations would 'automatically' include the infinite branch. (Cf. the LT case, where $lim_i\{a^j \mid j \leq i\}$ includes, besides a^* also the limit point a^ω.) Another example of an infinite process is the following: Take $p_0 = \{a\}$, $p_1 = \{[a,p_0]\}$, $p_{i+2} = \{[a,p_i],[a,p_{i-1}]\}$. Then $p = lim_i p_i = lim_i q_i$, where $q_0 = \{a\}$, $q_{i+1} = \{[a,q_i]\}$. Thus the branching structure of the p_i 'collapses' in its limit $lim_i p_i$.

Remark. We observe that the collection of all finite and infinite trees over $A_\perp$ (where $\perp$ occurs only at the leaves) modulo Park's equivalence of strong bisimulation, i.e. without special treatement of τ ([Pa]) is isomorphic to $\mathbb{P}$.

The empty set *is* a process and takes the role of δ. Note that in the previous linear time framework (LT) $\varnothing$ cannot replace δ since by the definition of concatenation (for LT) we have $a.\varnothing = \varnothing$ which is undesirable for an element modelling failure. (An action which fails should not cancel all previous actions.) In the present branching time framework, $\{[a, \varnothing]\}$ is a process which is indeed different from (and irreducible to) $\varnothing$.

The following operations on processes are defined: we first take the case that both processes are finite. Then we use induction on the degree(s) of the processes concerned:

concatenation $\circ$: $p \circ q = \{x \circ q : x \in p\}$, where $\perp \circ q = \perp$, $a \circ q = [a,q]$, $[a,p'] \circ q = [a, p' \circ q]$ and similar clauses with c replacing a.

union $\cup$: $p \cup q$ is the set-theoretic union of p and q.

merge $\|$: $p \| q = (p \lfloor\!\lfloor q) \cup (q \lfloor\!\lfloor p) \cup (p \mid q)$ where $p \lfloor\!\lfloor q = \{x \lfloor\!\lfloor q : x \in p\}$, $\perp \lfloor\!\lfloor q = \perp$, $a \lfloor\!\lfloor q = [a,q]$, $[a,p'] \lfloor\!\lfloor q = [a, p' \| q]$ and similar clauses with c replacing a.

Moreover, $p \mid q = \bigcup \{x \mid y : x \in p, y \in q\}$, where

$$[c,p'] \mid [\overline{c},q'] = \{[\tau, p' \| q']\}$$

$[c,p'] \mid \overline{c} = \{[\tau,p']\}$

$c \mid \overline{[c,p']} = \{[\tau,p']\}$

$c \mid \overline{c} = \{\tau\}$

and $x \mid y = \varnothing$ for x,y not of one of the above four forms.

For p or q infinite we have (since $\mathbb{P}$ is defined by completion of $\mathbb{P}_\omega$) that $p = lim_n p_n$, $q = lim_n q_n$, p_n and q_n finite, $n = 0,1, \cdots$, and we define $p\ op\ q = lim_n(p_n\ op\ q_n)$, $op \in \{\circ, \cup, \|\}$.

Examples

1. Concatenation: $\{\bot\}\circ p = \{\bot\circ p\} = \{\bot\}$, $\varnothing\circ p = \varnothing$, $\{[a,\{b\}],[a,\varnothing]\}\circ\{c\} = \{[a,\{b\}]\circ\{c\},[a,\varnothing]\circ\{c\}\} = \{[a,\{b\}\circ\{c\}],[a,\varnothing]\} = \{[a,\{b\circ\{c\}\}],[a,\varnothing]\} = \{[a,\{[b,\{c\}]\}],[a,\varnothing]\}$.
   Let $p_0 = \{a\}$, $p_{i+1} = \{[a,p_i]\}$. Then, for any q, $(lim_i p_i)\circ q = lim_i(p_i\circ q) = lim_i\{[a,p_i\circ q]\} = \{[a, lim_i(p_i\circ q)]\}$ (cf.the continuity result of the next proposition). Hence, $(lim_i p_i)\circ q = lim_i p_i$.
2. Union: $p \cup \varnothing = \varnothing \cup p = p$. Also, $\{[a,\{b_1\}]\} \cup \{[a,\{b_2\}]\} = \{[a,\{b_1\}],[a,\{b_2\}]\}$ (rather than $\{[a,\{b_1,b_2\}]\}$ which is not the union of any two nonempty processes).
3. Merge: For brevity, we here write for a process $\{[a,p],[b,q], \cdots\}$: $a.p' + b.q' + \cdots$, with p',q' abbreviations for $p,q, \cdots$. Then
   $(a.c)\|(b.\overline{c}) =$
   $a.(b.(c.\overline{c} + \overline{c}.c + \tau) + c.b.\overline{c}) +$
   $b.(a.(\overline{c}.c + c.\overline{c} + \tau) + \overline{c}.a.c)$.

Continuity of the process operators is expressed in

PROPOSITION 6.2.
Let $p = lim_i p_i$, $q = lim_i q_i$ (with p_i,q_i not necessarily finite). Then $p\circ q = lim_i(p_i\circ q_i)$, and similarly for $\cup, \|$.

For the proof of this statement see [BZ2].

It is now straightforward to define $\mathfrak{D}_2$: *guarded* $L_2 \to (\Gamma_2 \to \mathbb{P})$, where $\Gamma_2 = Stmv \to \mathbb{P}$, by following the clauses in the definition of $\mathfrak{D}_0, \mathfrak{D}_1$. Thus, we put

$\mathfrak{D}_2[\![a]\!](\gamma) = \{a\}$

$\mathfrak{D}_2[\![c]\!](\gamma) = \{c\}$

$\mathfrak{D}_2[\![s_1\ op\ s_2]\!](\gamma) = \mathfrak{D}_2[\![s_1]\!](\gamma)\, op^{\mathfrak{D}_2}\, \mathfrak{D}_2[\![s_2]\!](\gamma)$

for $op \in \{;, +, \|\}$, where $;^{\mathfrak{D}_2} = \circ$, $+^{\mathfrak{D}_2} = \cup$, $\|^{\mathfrak{D}_2} = \|$.

$\mathfrak{D}_2[\![x]\!](\gamma) = \gamma(x)$

$$\mathcal{D}_2[\![\mu x[s]]\!](\gamma) = lim_i p_i$$

where $p_0 = \{\bot\}$ and $p_{i+1} = \mathcal{D}_2[\![s]\!](\gamma < p_i / x >)$.

Examples

1. $\mathcal{D}_2[\![(a;b)+(a;c)\|\overline{c}]\!](\gamma) =$ (in the abbreviated notation just introduced) $a.(b.\overline{c}+\overline{c}.b)+a.(c.\overline{c}+\overline{c}.c+\tau)+\overline{c}.(a.b+a.c)$.
2. $\mathcal{D}_2[\![\mu x[a;x]]\!](\gamma) = lim_i p_i$, where $p_0 = \{\bot\}$, $p_{i+1} = \{[a,p_i]\}$.
3. $\mathcal{D}_2[\![\mu x[(a;x)+b]]\!](\gamma) = lim_i p_i$, where $p_0 = \{\bot\}$, $p_{i+1} = \{[a,p_i],b\}$.
4. $\mathcal{D}_2[\![\mu x[(a\|x)+b]]\!](\gamma)$ is undefined by unguardedness.

Mutatis mutandis, the contractivity results for $\mathcal{D}_0, \mathcal{D}_1$ hold again.

6.3. Relationship between $\mathcal{O}_2$ and $\mathcal{D}_2$.

For a suitable abstraction operator α_2 we shall show that

$$\mathcal{O}_2[\![s]\!] = \alpha_2(\mathcal{D}_2[\![s]\!]) \qquad (*)$$

holds for all guarded $s \in L_2$. We define $\alpha_2 : \mathbb{P} \to \mathbb{S}(\delta)$ in two steps (recall that $a \in A \setminus C$).

1. First we define a restriction mapping $restr_{\mathbb{P}} : \mathbb{P} \to \mathbb{P}$. For $p \in \mathbb{P}_\omega$ we put inductively:

$$restr_{\mathbb{P}}(p) = \{a \mid a \in p\} \cup \{[a, restr_{\mathbb{P}}(q)] \mid [a,q] \in p\}$$

For $p \in \mathbb{P} \setminus \mathbb{P}_\omega$ we have $p = lim_n p_n$, with $p_n \in \mathbb{P}_n$ and we put

$$restr_{\mathbb{P}}(p) = lim_n(restr_{\mathbb{P}}(p_n))$$

Example. Let $p = \mathcal{D}_2[\![(a+c)\|(b+\overline{c})]\!] = \mathcal{D}_2[\![(a;(b+\overline{c}))+(c;(b+\overline{c}))+(b;(a+c))+(\overline{c};(a+c))+\tau]\!]$. Then $restr_{\mathbb{P}}(p) = \{[a,\{b\}],[b,\{a\}],\tau\} = \mathcal{D}_2[\![(a;b)+(b;a)+\tau]\!]$.

2. Then we define a mapping $streams : \mathbb{P} \to \mathbb{S}_{nc}(\delta)$. For $p \in \mathbb{P}_\omega$ we put inductively:

$$streams(p) = \{a \mid a \in p\} \cup \{c \mid c \in p\} \cup \{a.streams(q) \mid [a,q] \in p\} \cup \{c.streams(q) \mid [c,q] \in p\}, \text{ if } p \neq \varnothing$$

$$= \{\delta\}, \text{ if } p = \varnothing$$

Note that $a.streams(q)$ and $c.streams(q)$ are themselves sets of streams. For $p \in \mathbb{P} \setminus \mathbb{P}_\omega$ we have $p = lim_n p_n$, with $p_n \in \mathbb{P}_n$ and we put

$$streams(p) = lim_n(streams(p_n))$$

Note that lim_n is taken with respect to the metric on $\mathbb{S}_{nc}(\delta)$ (see section 3.3).

Example. With p as above we have $streams(p) = \{ab, a\overline{c}, cb, c\overline{c}, ba, bc, \overline{c}a, \overline{c}c, \tau\}$ and $streams(restr_{\mathbb{P}}(p)) = \{ab, ba, \tau\}$.

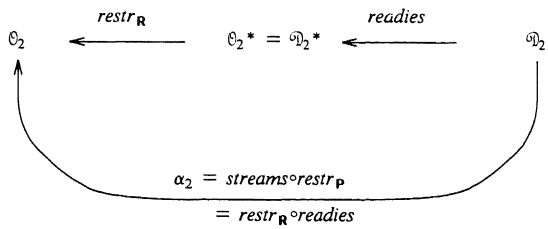
Finally we put

$$\alpha_2 = streams \circ restr_{\mathbb{P}}$$

in (*). Similarly to α_1, we cannot prove (*) directly by structural induction on s because α_2 does not behave compositionally. Thus again the question arises how to prove (*)? Note that here things are rather more difficult than with $\mathcal{O}_1[\![s]\!] = \alpha_1(\mathcal{D}_1[\![s]\!])$ because the semantic domains of $\mathcal{O}_1$ and $\mathcal{D}_1$ are quite different: linear streams vs. branching processes.

Our solution to this problem is to introduce
- a new intermediate semantic domain $\mathbb{R}$
- a new intermediate operational semantics $\mathcal{O}_2{}^*$ on $\mathbb{R}$
- a new intermediate denotational semantics $\mathcal{D}_2{}^*$ on $\mathbb{R}$

and then prove the following diagram:



where $restr_{\mathbb{R}}$ and *readies* are two further abstraction operators.

6.3.1. The intermediate semantic domain $\mathbb{R}$.

We start with the intermediate semantic domain. To motivate its construction, let us first demonstrate that a simple *stream-like* variant of $\mathcal{O}_2$ is not appropriate as intermediate operational semantics $\mathcal{O}_2{}^*$ here. Indeed, if we base $\mathcal{O}_2{}^*$ -similarly to $\mathcal{O}_1{}^*$- on a transition system obtained by just adding the axiom

$$<c,w> \rightarrow w.c$$

to T_2, we cannot retrieve $\mathcal{O}_2$ from $\mathcal{O}_2{}^*$. As a counterexample consider the programs $s_1 = (a;c_1)+(a;c_2)$, $s_2 = a;(c_1+c_2)$ and $s = \overline{c}_1$. Then $\mathcal{O}_2[\![s_1 \| s]\!] = \{a\tau, a\delta\} \neq \{a\tau\} = \mathcal{O}_2[\![s_2 \| s]\!]$, but, $\mathcal{O}_2{}^*[\![s_1 \| s]\!] = \mathcal{O}_2{}^*[\![s_2 \| s]\!]$. Thus whatever operator α we apply to $\mathcal{O}_2{}^*[\![.]\!]$, the results for $s_1 \| s$ and $s_2 \| s$ will turn out to be the same. Thus we cannot retrieve $\mathcal{O}_2$ from $\mathcal{O}_2{}^*$.

To solve this problem, we introduce for $\mathcal{O}_2{}^*$ a new semantic domain, which, besides streams $w \in A^{st}$, also includes a very weak information about the *local branching structure* of a process. This information is called *ready set* or *deadlock possibility;* it takes the form of a subset X of C, the set of communications, and may appear (locally) after every word $w \in A^*$ of successful

actions. Informally, such a set X after w indicates that after w the process is ready for all communications $c \in C$ and that deadlock can be avoided only if some communication $c \in C$ can synchronize with a matching communication $\overline{c}$ in some other parallel component. Thus X can be seen as a "more informative δ". This view is confirmed by the fact that there will be no ready set X after w if the process can do an elementary action $a \in A \setminus C$ and thus avoid deadlock on its own. With some variations this notion of a ready set appears in the work of [BHR,FLP,OH1,OH2,RB].

Formally, we take $\Delta = \mathcal{P}(C)$ and define the set of *streams with ready sets* as

$$A^{rd} = A^{st} \cup A^* : \Delta$$

where $A^* : \Delta$ denotes the set of all pairs of the form $w : X$ with $w \in A^*$ and $X \in \Delta$. For $X \in \Delta$, let $\overline{X} = \{\overline{c} \mid c \in X\}$. As intermediate domain we take the *ready domain*

$$\mathbb{R} = \mathcal{P}(A^{rd})$$

Just as we did for A^{st} and $A^{st}(\delta)$ we can define a metric d on A^{rd} and a corresponding metric $\hat{d}$ on $\mathbb{R}$. This $\hat{d}$ turns the collection $\mathbb{R}_{nc} \subseteq \mathbb{R}$ of closed non empty subsets of A^{rd} into a complete metric space $(\mathbb{R}_{nc}, \hat{d})$.

6.3.2. The intermediate operational semantics $\mathcal{O}_2$.*

We now turn to the intermediate operational semantics $\mathcal{O}_2$* on $\mathbb{R}$. It is based on the following transition system T_2* which consists of all axioms and rules of T_2 extended (for $w \in A^*$) by:

(communications*)

$$< c, w > \rightarrow w.c$$

(ready sets [or: deadlock possibilities])

$$< c, w > \rightarrow w : \{c\} \qquad \text{(i)}$$

$$\frac{< s_1, w > \rightarrow w : X}{< s_1 ; s_2, w > \rightarrow w : X} \qquad \text{(*)(ii)}$$

$$\frac{< s_1, w > \rightarrow w : X \ , \ < s_2, w > \rightarrow w : Y}{< s_1 + s_2, w > \rightarrow w : (X \cup Y)} \qquad \text{(iii)}$$

$$\frac{< s_1, w > \rightarrow w : X \ , \ < s_2, w > \rightarrow w : Y}{< s_1 \| s_2, w > \rightarrow w : (X \cup Y)} \qquad \text{(iv)}$$

(*) Rule (ii) was incorrectly omitted in [BMOZ1].

where $X \cap \overline{Y} = \varnothing$.

Axiom (i) introduces ready sets or deadlock possibilities, and rules (ii)-(iv) propagate them. In particular, rule (iii) says that $s_1 + s_2$ has a deadlock possibility if s_1 and s_2 have, and rule (iv) says that $s_1 \| s_2$ has a deadlock possibility if both s_1 and s_2 have, and no synchronization is possible.

The intermediate operational semantics

$$\mathcal{O}_2^* : L_2 \to \mathbb{R}$$

is defined in terms of T_2^* just as $\mathcal{O}_2$ was defined in terms of T_2. In particular, $w:X \in \mathcal{O}_2^*[\![s]\!]$ iff there exists a sequence of T_2^*-transitions

$$<s,\epsilon> = <s_0,w_0> \to \cdots \to <s_n,w_n> \to w:X$$

Examples.

(i) $\mathcal{O}_2^*[\![a;(b+c)]\!] = \{ab,ac\}$. *Proof.* We explore all transition sequences in T_2^* starting in $<a;(b+c),\epsilon>$:

1 $<a,\epsilon> \to a$ (elementary action)

2 $<a;(b+c),\epsilon> \to <b+c,a>$ (sequential composition using 1)

3 $<b,a> \to ab$ (elementary action)

4 $<c,a> \to ac$ and $<c,a> \to a:\{c\}$ (communication*)

5 $<b+c,a> \to ab$, $<b+c,a> \to ac$ (global nondeterminacy using 3 and 4) No more transitions are deducible for $<b+c,a>$.

6 Thus $<a;(b+c),\epsilon> \to <b+c,a> \to ab$ or $\to ac$ are all the transition sequences starting in $<a;(b+c),\epsilon>$

This proves the claim. □

(ii) $\mathcal{O}_2^*[\![a;b+a;c]\!] = \{ab,ac,a:\{c\}\}$. *Proof.* Here we only exhibit all possible transition sequences in T_2^* starting in $<(a;b)+(a;c),\epsilon>$:

$$<a;b+a;c,\epsilon> \to <b,a> \to ab$$

$$\to <c,a> \to ac$$

$$\to a:\{c\}$$

□

Remark. Note that we can prove $<a;b+a;c,\epsilon> \to <c,a>$ and $<c,a> \to a:\{c\}$, and therefore $<a;b+a;c,\epsilon> \xrightarrow{*} a:\{c\}$. However, we have $<a;(b+c),\epsilon> \to <b+c,a>$, but we cannot prove $<b+c,a> \to a:\{c\}$. (By rule (iii) of ready sets this would only be the case if we could prove, besides $<c,a> \to a:\{c\}$, also $<b,a> \to a:X$ for some $X \subseteq \{c\}$. Since the only possibilities for X are $\varnothing$ and $\{c\}$, this cannot be proved.) Consequently, $<a;(b+c),\epsilon> \not\xrightarrow{*} a:\{c\}$.

6.3.3. The intermediate denotational semantics $\mathcal{D}_2{}^*$.

We start by defining semantic operators $;^{\mathcal{D}_2{}^*}$, $+^{\mathcal{D}_2{}^*}$ and $\|^{\mathcal{D}_2{}^*}$ on $\mathbb{R}_{nc}$. (Again we omit superscripts $\mathcal{D}_2{}^*$ whenever possible.) Let $W_1, W_2 \in \mathbb{R}_{nc}$, $w, w_1, w_2 \in A^{st}$.

a. $W_1, W_2 \subseteq A^* \cup A^*.\{\perp\} \cup A^* : \Delta$ Then

$$W_1 ; W_2 = \{w_1.w_2 \mid w_1 \in W_1 \text{ and } w_2 \in W_2\}$$
$$\cup \{w_1 : X \mid w_1 : X \in W_1\}$$
$$\cup \{w_1.w_2 : X \mid w_1 \in W_1 \text{ and } w_2 : X \in W_2\}$$

$$W_1 + W_2 = \{w \mid w \in W_1 \cup W_2\}$$
$$\cup \{\epsilon : (X \cup Y) \mid \epsilon : X \in W_1 \text{ and } \epsilon : Y \in W_2\}$$
$$\cup \{w : X \mid w \neq \epsilon \text{ and } w : X \in W_1 \cup W_2\}$$

$$W_1 \| W_2 = (W_1 \mathbb{L} W_2) \cup (W_2 \mathbb{L} W_1) \cup (W_1 \mid W_2) \cup (W_1 \# W_2)$$

where

- $W_1 \mathbb{L} W_2 = \bigcup \{w_1 \mathbb{L} W_2 : w_1 \in W_1\}$ with $\epsilon \mathbb{L} W_2 = W_2$, $(a.w_1) \mathbb{L} W_2 = a.(\{w_1\} \| W_2)$, $\perp \mathbb{L} W_2 = \{\perp\}$, $(\epsilon : X) \mathbb{L} W_2 = \varnothing$, $((a.w) : X) \mathbb{L} W_2 = a.(\{w : X\} \| W_2)$.
- $W_1 \mid W_2 = \bigcup \{(w_1 \mid w_2) : w_1 \in W_1 \text{ and } w_2 \in W_2\}$ with $c.w' \mid \overline{c}.w'' = \tau.(\{w'\} \| \{w''\})$, and $w_1 \mid w_2 = \varnothing$ for w_1, w_2 not of this form.
- $W_1 \# W_2 = \{\epsilon : (X \cup Y) \mid \epsilon : X \in W_1 \text{ and } \epsilon : Y \in W_2 \text{ and } X \cap \overline{Y} = \varnothing\}$.

b. $W_1, W_2 \in \mathbb{R}_{nc}$ and W_1, W_2 contain also infinite words. Then extend the previous definitions by taking limits in $\mathbb{R}_{nc}$.

Now we define

$$\mathcal{D}_2{}^* : \textit{guarded } L_2 \to (\Gamma_2{}^* \to \mathbb{R}_{nc}),$$

with $\Gamma_2{}^* = Stmv \to \mathbb{R}_{nc}$, in the usual way (but note the clause for $\mathcal{D}_2{}^*[\![c]\!](\gamma)$!)

1 $\mathcal{D}_2{}^*[\![a]\!](\gamma) = \{a\}$, $\mathcal{D}_2{}^*[\![c]\!](\gamma) = \{c, \epsilon : \{c\}\}$

2 $\mathcal{D}_2{}^*[\![s_1 \ op \ s_2]\!](\gamma) = \mathcal{D}_2{}^*[\![s_1]\!](\gamma) \ op^{\mathcal{D}_2{}^*} \ \mathcal{D}_2{}^*[\![s_2]\!](\gamma)$

3 $\mathcal{D}_2{}^*[\![x]\!](\gamma) = \gamma(x)$

4 $\mathcal{D}_2{}^*[\![\mu x[s]]\!](\gamma) = lim_i W_i$, where $W_0 = \{\perp\}$ and $W_{i+1} = \mathcal{D}_2{}^*[\![s]\!](\gamma < W_i / x >)$.

6.3.4. Relating $\mathcal{O}_2$ and $\mathcal{O}_2{}^*$, $\mathcal{D}_2$ and $\mathcal{D}_2{}^*$, and $\mathcal{O}_2$ and $\mathcal{D}_2$.

The relationship between $\mathcal{O}_2$ and $\mathcal{O}_2{}^*$ is similar to that between $\mathcal{O}_1$ and $\mathcal{O}_1{}^*$ in section 3.4. In fact, we shall prove:

THEOREM 6.3. $\mathcal{O}_2[\![s]\!] = restr_{\mathbb{R}}(\mathcal{O}_2{}^*[\![s]\!])$ for every $s \in L_2$.

Here $restr_{\mathbb{R}} : \mathbb{R} \to \mathbb{S}(\delta)$ is a restriction operator similar to $restr_{\mathbb{S}} : \mathbb{S}(\delta) \to \mathbb{S}(\delta)$ of section 4.3. For $W \in \mathbb{R}$ and $w \in A^{st}$ we define

$$restr_{\mathbb{R}}(W) = \{w \mid w \in W \text{ does not contain any } c \in C\}$$
$$\cup \{w.\delta \mid \exists X \in \Delta \text{ such that } w:X \in W \text{ and } w \text{ does not contain any } c \in C\}$$

For theorem 6.3 we need the following result concerning the transition systems T_2 and $T_2{}^*$. (Compare lemma 4.6.)

LEMMA 6.4. For all $s \in L_2$, $s' \in L_2 \cup \{E\}$ and $w,w' \in (A \setminus C)^*$:
(i) $T_2 \vdash <s,w> \to <s',w'>$ iff $T_2{}^* \vdash <s,w> \to <s',w'>$
(ii) $<s,w>$ is a deadlocking configuration for T_2 iff $\exists X \subseteq C : T_2{}^* \vdash <s,w> \to w:X$

Proof. See [BMOZ2].

Intuitively, lemma 6.4(ii) says that the ready set rules (i)-(iv) of $T_2{}^*$ are complete for detecting deadlocks. Using lemma 6.4 it is not difficult to prove theorem 6.3.

The relationship between $\mathcal{D}_2$ and $\mathcal{D}_2{}^*$ is given by an abstraction operator $readies : \mathbb{P} \to \mathbb{R}_{nc}$. Let -for the duration of this definition only- a,b range over all of A (and not just over $A \setminus C$). For $p = \{a_1, \ldots, a_m, [b_1,q_1], \ldots, [b_n,q_n]\} \in \mathbb{P}_\omega$ we put inductively

$$readies(p) = \{a_1, \ldots, a_m\}$$
$$\cup \{b_j.readies(q_j) \mid j = 1, \ldots, n\}$$
$$\cup \{\epsilon:X \mid \text{where } X = \{a_1, \ldots, a_m, b_1, \ldots, b_n\} \subseteq C\}$$

Example. Let $a \in A \setminus C$ (and $c_1,c_2 \in C$). Then $readies(\{[a, \{c_1,c_2\}]\}) = a.readies(\{c_1,c_2\}) = a.\{c_1,c_2,\epsilon:\{c_1,c_2\}\} = \{ac_1,ac_2,a:\{c_1,c_2\}\}$.

For $p \in \mathbb{P} \setminus \mathbb{P}_\omega$ we have $p = lim_n p_n$ with $p_n \in \mathbb{P}_n$, and we put

$$readies(p) = lim_n(readies(p_n))$$

where lim_n is taken (as before) with respect to the metric on $\mathbb{R}_{nc}$.

THEOREM 6.5. $\mathcal{D}_2{}^*[\![s]\!] = readies(\mathcal{D}_2[\![s]\!])$ for all syntactically closed and guarded $s \in L_2$.

The proof follows from:

THEOREM 6.6. The operator $readies : \mathbb{P} \to \mathbb{R}_{nc}$ is continuous and behaves homomorphically,i.e. for $op \in \{+,;,\|\}$ and $p,p' \in \mathbb{P}$

$$readies(p\ op^{\mathcal{D}_2}\ p') = readies(p)\ op^{\mathcal{D}_2{}^*}\ readies(p')$$

holds.

Proof. The proof is given in [BMOZ2].

Next we state

THEOREM 6.7. $\mathcal{O}_2{}^*[\![s]\!] = \mathcal{D}_2{}^*[\![s]\!]$ for every syntactically closed and guarded $s \in L_2$.

Again, its proof follows the structure of that for "$\mathcal{O}_0[\![s]\!] = \mathcal{D}_0[\![s]\!]$" (theorem 3.12). In particular, the lemmas 3.14, 3.15, 3.17 remain valid for $\mathcal{O}_2{}^*$, $\mathcal{D}_2{}^*$ and L_2 instead of $\mathcal{O}_0$, $\mathcal{D}_0$ and L_0. Thus it remains to show compositionality of $\mathcal{O}_2{}^*$, analogously to lemma 3.13 but now involving the ready domain $\mathbb{R}$ and global nondeterminacy $+$.

LEMMA 6.8. For $op \in \{+,;,\|\}$ and $s_1,s_2 \in L_2$ the equation $\mathcal{O}_2{}^*[\![s_1\ op\ s_2]\!] = \mathcal{O}_2{}^*[\![s_1]\!]\ op^{\mathcal{D}_2{}^*}\ \mathcal{O}_2{}^*[\![s_2]\!]$ holds.

For the proof -which involves a substantial amount of work- we again refer to [BMOZ2].

We finally prove the desired relationship between $\mathcal{O}_2$ and $\mathcal{D}_2$ (cf. (*) at the beginning of this section 6.3). First we need one more lemma.

LEMMA 6.9. For every $p \in \mathbb{P}$ the equation $streams(restr_{\mathbb{P}}(p)) = restr_{\mathbb{R}}(readies(p))$ holds.

Now we are prepared for the main result on L_2:

THEOREM 6.10. With $\alpha_2 = streams \circ restr_{\mathbb{P}}$ the equation $\mathcal{O}_2[\![s]\!] = \alpha_2(\mathcal{D}_2[\![s]\!])$ holds for all syntactically closed and guarded $s \in L_2$.

PROOF. Theorem 6.3 states $\mathcal{O}_2[\![s]\!] = restr_{\mathbb{R}}(\mathcal{O}_2{}^*[\![s]\!])$ for $s \in L_2$, theorem 6.4. states $\mathcal{D}_2{}^*[\![s]\!] = readies(\mathcal{D}_2[\![s]\!])$ for syntactically closed and guarded $s \in L_2$, and theorem 6.7 states $\mathcal{O}_2{}^*[\![s]\!] = \mathcal{D}_2{}^*[\![s]\!]$ for syntactically closed and guarded $s \in L_2$. Thus we obtain $\mathcal{O}_2[\![s]\!] = restr_{\mathbb{R}}(readies(\mathcal{D}_2[\![s]\!]))$. Using lemma 6.9 completes the proof of this theorem. □

7. A Nonuniform Language with value passing

7.1. Introduction.

We devote the final section of our paper to the discussion of a *nonuniform* language. Elementary actions are no longer uninterpreted but taken as either assignments or tests. Communication actions c, $\overline{c}$ are refined to actions $c?v$ and $c!e$ (with v a variable and e an expression), and successful communication now involves two effects: (i) synchronization (as in the languages L_1, L_2) and (ii) value passing: the (current) value of e is assigned to v. Thus, we have here the synchronous handshaking variety of message passing in the sense of CCS or CSP.
We shall introduce a language L_3 which embodies these features and present its operational and denotational semantics $\mathcal{O}_3$ and $\mathcal{D}_3$. Nonuniformity of L_3 calls for the notion of *state* in both semantic models: they now deliver sets of streams or processes over state transformations, not overr uninterpreted actions as in the previous sections. As a consequence the formulation of the relationship between $\mathcal{O}_3$ and $\mathcal{D}_3$ requires additional effort. In fact, we shall only state a conjecture as to the connection between $\mathcal{O}_3$ and $\mathcal{D}_3$ in the style of the previous sections.

The operational semantics $\mathcal{O}_3$ for L_3 is based on a fairly straightforward nonuniform version of the transition system for the uniform language L_2. It thus owes much to the original papers of Hennessy and Plotkin [HP,Pl3,Pl4] which all address nonuniform languages. Our emphasis on the uniform case in the preceding sections is inspired on the one hand by the pioneering work of Milner on CCS [Mi2], on the other hand by the theory of formal languages over infinite words as developed by Nivat et al (e.g. [AN, Ni1]). As the operational semantics for L_3 shows, the uniform case provides a very helpful step towards a full analysis of the nonuniform case. For the denotational semantics [BZ2] is the primary source. Our attempt to relate $\mathcal{O}_3$ and $\mathcal{D}_3$ follows the general plan of [BMOZ2] and has clearly influenced both definitions.

The language L_3 is a quite simple case of a nonuniform language with parallelism and communication. Some of the more advanced concepts in nonuniform languages are not covered here, for example 'mixed guards' involving both Boolean and communication parts as in CSP and OCCAM, or the ADA rendez-vous. However, we are confident that both operational and denotational techniques presented below are adaptable to these and similar concepts. Some evidence for this is provided by the investigations of the ADA rendez-vous in [BZ4] and the language POOL (a Parallel Object-Oriented Language, see [Am]) in [ABKR1,2]. There the techniques developed below are applied, albeit with various substantial refinements and extensions.

We now present the syntax of L_3. We use three new syntactic categories, viz.
- the set *Var,* with elements v, w, of individual *variables*
- the set *Exp,* with elements e, of *expressions*
- the set *Bexp,* with elements b, of *boolean expressions.*

We shall not specify a syntax for *Exp* or *Bexp*. We assume that (boolean) expressions are of an elementary kind; in particular, they have no side effects and their evaluation always terminates. Statement variables $x,y, \cdots$ are as before, as are the communications $c \in C$. The latter now appear syntactically as part of value passing communication actions $c?v$ or $c!e$.

DEFINITION 7.1 (syntax for L_3).

$$s ::= v:=e \mid b \mid c?v \mid c!e \mid s_1;s_2 \mid s_1+s_2 \mid s_1 \| s_2 \mid x \mid \mu x[s]$$

We observe that 'isolated' booleans appear as statements. For the reader who has not seen such a syntax before we list a few well-known constructs with their equivalent counterparts in L_3:

- **skip** $\rightsquigarrow$ **true** (the identically true boolean expression)
- **if** b **then** s_1 **else** s_2 **fi** $\rightsquigarrow (b;s_1)+(\neg b;s_2)$
- **while** b **do** s **od** $\rightsquigarrow \mu x[(b;s;x)+\neg b]$ (x not free in s)
- **do** $b_1 \rightarrow s_1 \ \square \cdots \square \ b_n \rightarrow s_n$ **od** $\rightsquigarrow \mu x[(b_1;s_1+\cdots+b_n;s_n);x+(\neg b_1 \wedge \cdots \wedge \neg b_n)]$, x not free in $s_1, \ldots, s_n$.

In the next two subsections we shall define operational and denotational semantics for L_3.

7.2. Operational semantics for a nonuniform language.

For both operational and denotational models the notion of *state* is fundamental. Elements v,w in *Var* will have values α,β in a set V, and a state is a function which maps variables to their (current) values. Accordingly, we take for the set of states Σ the function space

$$\Sigma = Var \rightarrow V$$

and we use $\sigma,\sigma', \cdots$ for the elements of Σ. It is thus meaningful to write $\sigma(v)=\alpha$. For states σ we use a variant notation just as for environments γ (cf. definition 3.10), and we write $\sigma' =^{df} \sigma<\alpha / v>$ for a state which is like σ but such that $\sigma'(v)=\alpha$. We shall also employ a special failure state δ with $\delta \notin \Sigma$, and we shall write Σ_δ for $\Sigma \cup \{\delta\}$. (For the moment we have no occasion to use a (nonterminating) state $\perp$.) For expressions e and booleans b we postulate a simple semantic evaluation mechanism, details of which we do not bother to provide. The values of e and b in state σ will be denoted simply by $[\![e]\!](\sigma)$ and $[\![b]\!](\sigma)$, respectively. Here $[\![e]\!](\sigma)$ is an element of V and $[\![b]\!](\sigma)$ of the set of truthvalues $\{tt,ff\}$. Thus, in particular, $[\![\textbf{true}]\!](\sigma)=tt$ for all $\sigma \in \Sigma$.

The configurations as used in the transition system are now of the form $<s,\sigma>$ or simply σ. As before we use the auxiliary statement E satisfying the identities $<E,\sigma>=\sigma$ and

$$E;s = s;E = E \| s = s \| E = s.$$

We now present the transition system T_3. It is quite similar to T_2, but for the axioms (assignment), (test) and (communication) dealing with the special nonuniform cases. However, we exhibit the system in full detail in order to be precise about the effect of having, throughout, σ rather

than w as a component of configurations.

Let $s \in L_3$. We put $<s,\delta> \rightarrow \delta$ and, for $\sigma \neq \delta$, the following axioms and rules hold:

(assignment)

$$<v:=t,\sigma> \rightarrow \sigma<[\![e]\!](\sigma)/v>$$

(test)

$$<b,\sigma> \rightarrow \sigma, \text{ if } [\![b]\!](\sigma)=tt$$

(communication)

$$<c?v \,\|\, c!e,\sigma> \rightarrow <\textbf{true},\sigma<[\![e]\!](\sigma)/v>>$$

(recursion)

$$<\mu x[s],\sigma> \rightarrow <s[\mu x[s]/x],\sigma>$$

(sequential composition, shuffle)

$$\frac{<s_1,\sigma_1> \rightarrow <s_2,\sigma_2>}{<s_1;s,\sigma_1> \rightarrow <s_2;s,\sigma_2>}$$

$$<s_1 \| s,\sigma_1> \rightarrow <s_2 \| s,\sigma_2>$$

$$<s \| s_1,\sigma_1> \rightarrow <s \| s_2,\sigma_2>$$

where s_2 may be E.

(communication in a context)

$$\frac{<s_1 \| s_2,\sigma> \rightarrow <s' \| s'',\sigma'>}{<(s_1;s) \| s_2,\sigma> \rightarrow <(s';s) \| s'',\sigma'>}$$

$$<(s_1 \| s) \| s_2,\sigma> \rightarrow <(s' \| s) \| s'',\sigma'>$$

$$<(s \| s_1) \| s_2,\sigma> \rightarrow <(s \| s') \| s'',\sigma'>$$

$$<s_1 \| (s_2;s),\sigma> \rightarrow <s' \| (s'';s),\sigma'>$$

$$<s_1 \| (s_2 \| s),\sigma> \rightarrow <s' \| (s'' \| s),\sigma'>$$

$$<s_1 \| (s \| s)_2,\sigma> \rightarrow <s' \| (s \| s)'',\sigma'>$$

where s',s'' or both may be E and where the premise of the rule describes a communication-transition between s_1 and s_2 such that s' stems from s_1 and s'' stems from s_2.

(selection)

[selection by μ-unfolding]

$$\frac{<s_1,\sigma>\rightarrow<s_2,\sigma>}{<s_1+s,\sigma>\rightarrow<s_2+s,\sigma>}$$

$$<s+s_1,\sigma>\rightarrow<s+s_2,\sigma>$$

where the premise is a recursion-transition

[selection by action]

$$\frac{<s_1,\sigma>\rightarrow<s_2,\sigma>}{<s_1+s,\sigma>\rightarrow<s_2,\sigma>}$$

$$<s+s_1,\sigma>\rightarrow<s_2,\sigma>$$

where the premise of the rule is an assignment-transition or a test-transition or a communication-transition, and where s_2 may be E

[selection by communication]

$$\frac{<s_1\|s_2,\sigma>\rightarrow<s',\sigma'>}{<(s_1+s)\|s_2,\sigma>\rightarrow<s',\sigma'>}$$

$$<(s+s_1)\|s_2,\sigma>\rightarrow<s',\sigma'>$$

$$<s_1\|(s_2+s),\sigma>\rightarrow<s',\sigma'>$$

$$<s_1\|(s+s_2),\sigma>\rightarrow<s',\sigma'>$$

where the premise of the rule describes a communication-transition between s_1 and s_2.

Remarks

1. Observe the role of **true** in the axiom for communication (which, by the axiom for tests, amounts to a skip action). It is included to obtain the effect of a silent step, mirroring the τ in the uniform setting. Moreover, the definition of 'stem from' should be updated in such a way that occurrences of **true** as introduced by the communication axiom stem from each of the statements $c?v$, $c!e$ on the left-hand side. (We can then choose whether to read **true** as **true** || E or E || **true,** in case we want to apply the rule fot communication in a context.)
2. Each test which is false in the current state and each individual communication ($c?v$ or $c!e$) fails in the sense that no transition is defined for such a pair $<s,\sigma>$. In a moment we shall see how $\mathcal{O}_3$ handles this case by delivering δ (as for L_2).

We proceed with the definition of $\mathcal{O}_3$. The relationship between $\mathcal{O}_3$ and T_3 is slightly different from that between $\mathcal{O}_i$ and T_i, i=0,1,2. Whereas T_i and $\mathcal{O}_i$, $i \leqslant 2$, all involve streams, we here have the situation that T_3 only refers to (single) states whereas $\mathcal{O}_3$ as defined below involves again streams (of states). Thus, the presentation of T_3 is close to the original work in [HP, Pl3, Pl4]. Of course, we might have adopted a simpler domain for (the outcome of) $\mathcal{O}_3$ as well. However, we prefer the present approach since we expect that $\mathcal{O}_3$ as in definition 7.2 is amenable to a comparison with $\mathcal{D}_3$ (as defined in section 7.3) which follows closely the $\mathcal{O}_2 \tilde{} \mathcal{D}_2$ correspondence from section 6. (Another alternative we have not adopted is to burden the definition of T_3 by using streams of states rather than single states in the components of configurations.)

For $\mathcal{O}_3$ we first need the usual stream definition:

$$\Sigma^{st}(\delta)=\Sigma^* \cup \Sigma^\omega \cup \Sigma^*.\{\delta\}$$

Let ρ range over $\Sigma^{st}(\delta)$. As always, we put $\delta.\rho=\delta$ for all ρ. Let, as before, $\mathbb{S}(\delta) =^{df} \mathcal{P}(\Sigma^{st}(\delta))$ be the collection of all subsets fo $\Sigma^{st}(\delta)$. We define the mapping

$$\mathcal{O}_3 : (\textit{syntactically closed})\ L_3 \rightarrow (\Sigma_\delta \rightarrow \mathbb{S}(\delta))$$

as follows

DEFINITION 7.2. Let $s \in L_3$. We put $\mathcal{O}_3[\![s]\!](\delta)=\{\delta\}$, and, for $\sigma \neq \delta$, let $<s_0,\sigma_0> =^{df} <s,\sigma>$.

1. We put $\sigma_0\sigma_1 \cdots \sigma_n\sigma' \in \mathcal{O}_3[\![s]\!](\sigma)$ if there is a finite sequence of T_3-transitions

$$<s_0,\sigma_0> \rightarrow <s_1,\sigma_1> \rightarrow \cdots \rightarrow <s_n,\sigma_n> \rightarrow \sigma'$$

2. We put $\sigma_0\sigma_1 \cdots \sigma_n\delta \in \mathcal{O}_3[\![s]\!](\sigma)$ if there is a finite sequence of T_3-transitions

$$<s_0,\sigma_0> \rightarrow <s_1,\sigma_1> \rightarrow \cdots \rightarrow <s_n,\sigma_n>$$

with $s_n \neq E$ such that no transition $<s_n,\sigma_n> \rightarrow <s',\sigma'>$ is deducible in T_3

3. We put $\sigma_0\sigma_1 \cdots \sigma_n \cdots \in \mathcal{O}_3[\![s]\!](\sigma)$ if there is an infinite sequence of T_3-transitions

$$<s_0,\sigma_0> \rightarrow <s_1,\sigma_1> \rightarrow \cdots \rightarrow <s_n,\sigma_n> \rightarrow \cdots$$

Examples

1. $\mathcal{O}_3[\![v:=0]\!](\sigma)=\{\sigma\sigma<0/v>\}$
   $\mathcal{O}_3[\![(v:=0)||(v:=1;v:=v+1)]\!](\sigma)= \{\sigma\sigma<0/v>\sigma<1/v>\sigma<2/v>,$
   $\sigma\sigma<1/v>\sigma<0/v>\sigma<2/v>,\ \sigma\sigma<1/v>\sigma<2/v>\sigma<0/v>\}.$
2. $\mathcal{O}_3[\![\mu x[x]]\!](\sigma)=\{\sigma\sigma \cdots\}$, $\mathcal{O}_3[\![\mu x[(v:=v+1;x)]]\!](\sigma<0/v>)=$
   $\{\sigma<0/v>\sigma<0/v>\sigma<1/v>\sigma<1/v>\sigma<2/v>\sigma<2/v> \cdots\},$
3. $\mathcal{O}_3[\![(v>0;v:=2)+(v\leqslant 0)]\!](\sigma<1/v>)= \{\sigma<1/v>\sigma<1/v>\sigma<2/v>\}$
   $\mathcal{O}_3[\![v\leqslant 0]\!](\sigma<1/v>)= \{\sigma<1/v>\delta\}$
4. $\mathcal{O}_3[\![c?v||c!3]\!](\sigma)= \{\sigma\sigma\sigma<3/v>\}$

Remark. Note that, contrary to the situation for L_0 to L_2, we have no means to distinguish proper nontermination (delivery of an infinite stream) from improper termination (delivery of an unfinished stream, for terminology cf. section 5). Another way of looking at this phenomenon is that we have, by our use of (the effect of) a skip transition at each procedure call, effectively turned each recursive construct into a guarded one.

7.3. Denotational semantics for a nonuniform language.

We provide a branching time denotational semantics for L_3. We again use a domain of processes, in the sense as encountered in section 6, as meanings for statements $s \in L_3$. However, processes are now more complicated entities. In particular, processes depend on states: rather than using the what may be called uniform domain equation of section 6:

$$P = \mathcal{P}_{closed}(A_\perp \cup (A \times P)), \qquad (*)$$

we shall adopt (a modification and extension of) the basic nonuniform equation

$$P = \Sigma \to \mathcal{P}_{closed}(\Sigma \cup (\Sigma \times P)). \qquad (**)$$

Using techniques which are a natural extension of those sketched in section 6.2, we can solve (**). The extension involves the definition of a distance between p,q as *functions.* This is defined simply by $d(p,q) = sup_\sigma d(p(\sigma),q(\sigma))$. Again, P is obtained by defining the finite processes P_n, $n = 0,1,...$ and by taking P as the completion of $\bigcup_n P_n$ with respect to the metric d.

Before presenting more of the details on the use of these nonuniform processes in the definition of $\mathcal{D}_3[\![s]\!]$, it may be useful to devote a few words to the necessity of a domain in this style of (**) (Plotkin [Pl1] calls the elements of this domain *resumptions*) for the denotational semantics of a nonuniform language with merge. Consider by way of contrast a nonuniform sequential language, and two simple statements $s_1 = (v_1 := 0; v_2 := v_1 + 1)$ and $s_2 = (v_2 := 2)$. In order to determine the meaning of $s_1; s_2$, we determine the state-transforming functions (in $\Sigma \to \Sigma$) $\phi = \mathcal{D}[\![s_1]\!]$ and $\psi = \mathcal{D}[\![s_2]\!]$, and then form the functional composition $\phi \circ \psi = \lambda\sigma.\psi(\phi(\sigma))$. Also, ϕ is made up from the functions ϕ_1, ϕ_2, i.e. $\phi = \phi_2 \circ \phi_1$, where $\phi_1 = \mathcal{D}[\![v_1 := 0]\!]$, and $\phi_2 = \mathcal{D}[\![v_2 := v_1 + 1]\!]$. It is important to realize that, in order to determine $\phi \circ \psi$, the fact that ϕ is composed from the two functions ϕ_1, ϕ_2 is no longer relevant. The situation is different, however, when we want to define the meaning of $s_1 \| s_2$. Here we assume -to be somewhat specific; variations are possible depending on what operations are taken as indivisible- that the intended meaning of $s_1 \| s_2$ equals the meaning of the sum $(v_1 := 0; v_2 := v_1 + 1; v_2 := 2) + (v_1 := 0; v_2 := 2; v_2 := v_1 + 1) + (v_2 := 2; v_1 := 0; v_2 := v_1 + 1)$. Now we observe that, once $\phi = \mathcal{D}[\![s_1]\!]$ and $\psi = \mathcal{D}[\![s_2]\!]$ have been determined, we cannot form the semantic merge $\phi \| \psi$, since the way ϕ was obtained from ϕ_1, ϕ_2 is necessary for the determination of the merge , but lost in ϕ. In summary, state transforming functions cannot be merged since they do not contain information on the way they are built up from elementary components. Two ways out appear. The first would be to apply a two stage process

in order to obtain $\mathcal{D}[\![s_1 \| s_2]\!]$. Firstly, decompose s_1, s_2 into elementary actions $a_1 = (v_1 := 0), \ldots, a_3 := (v_2 := 2)$. Then merge $a_1; a_2$ and a_3 as uniform constructs, and after completion of this determine the meaning of each uniform process obtained in this way as a state transformation. We do not adopt this approach since we do not like such a two stage procedure and, more importantly, since the reduction to the uniform case is problematic or intractable as soon as a more advanced flow of control is encountered. We have in mind notions such as test and set, critical sections or, more ambitiously, communication or dynamic process creation in languages such as POOL.

The second way out is the solution we adopt below. The meaning of s is an entity (nonuniform process) p depending on a state σ, and when applied in σ yields a new state σ' and a continuation p' (or, in general, a set of such pairs $[\sigma', p']$). A statement such as s_1 above obtains as its meaning $\mathcal{D}[\![s_1]\!] = p_1 = \lambda\sigma.\{[\sigma < 0 / v_1, \lambda\bar{\sigma}.\{\bar{\sigma} < \bar{\sigma}(v_1) + 1 / v_2 >\}]\}$. Clearly, p_1 is a semantic object which has preserved the information on how it was built up, and we may expect to be able to define a semantic merge on such objects (cf. definition 7.3).

We continue with the development of the denotational semantics for L_3 -for which we shall from now on use the familiar notation $\mathcal{D}_3$- based on a domain similar to (**) above. In fact, we shall use an equation which is somewhat more complex. As minor variation, we find it advantageous at this place to use a *nil* process p_0. This process is not the meaning of any statement, but serves to provide a more unified structure to our processes and, in this way, facilitates the definition below of the various process operators. (Technically, p_0 may be seen as labelling the leaves in the 'process tree'.) We thus start from the modification from (**):

$$P = \{p_0\} \cup (\Sigma \to \mathcal{P}_{closed}(\Sigma \times P))$$

Secondly, and more importantly, we have to cater for (synchronization and) communication. Two steps are taken here. First, we extend the possibilities for $p(\sigma)$ from a set of pairs $[\sigma', p']$ (elements of $\mathcal{P}_{closed}(\Sigma \times P)$) to an enlarged set $\mathcal{P}_{closed}((\Sigma \times P) \cup (C \times \cdots))$, where C is the set of communication actions with elements c, and the $\cdots$ is a structure, yet to be filled in, coping with the additional value transmitting function of $c?v$ and $c!e$.

Now supplying the details, we shall use the domain defined by the equation

$$P = \{p_0\} \cup (\Sigma \to \mathcal{P}_{closed}((\Sigma \times P) \cup (C \times ((V \to P) \cup (V \times P))))) \qquad (***)$$

where elements ϕ in $V \to P$ denote processes depending on an argument α in V, and elements in $V \times P$ are pairs $[\alpha, p]$. Objects $[c, \phi]$ appear in the meaning of $c?v$, and objects $[c, [\alpha, p]]$ in the meaning of $c!e$. More specifically, $c?v$ has as effect the assignment of an, as yet unknown, value α to v. Accordingly, we deliver a function ϕ which, when given some α, returns process $\phi(\alpha)$ which performs the assignment. Usually, it is slightly more complex since ϕ -and, thus, $\phi(\alpha)$- not only describes the effect of the assignment of α to v, but has, in addition, accumulated the meaning of the statements executed after $c?v$.

Also, $c!e$ has as effect that the value of e is determined (in the current state) and kept in store for the (handshake) transmission to the receiving variable v on 'channel' c. The process p in the pair $[\alpha,p]$ describes the continuation after $c!e$. The actual communication and associated value passing take place in the definition of $p \| q$ and $X \mid Y$ given below. *Failing* attempts at communication receive the usual treatment, cf. the models for L_1, L_2: they remain as traces of unsuccessful attempts and are, if desired, cleaned away by a suitable restriction operator (as we shall define below as well).

From now on, we consider processes $p \in P$, with P as defined in (***). We next define the operators $\circ$, $\cup$, $\|$ upon them. We recapitulate the various (semantic) sets and the variables ranging over them, and introduce, for convenience, a few auxiliary sets:

$$p \in P$$

$$c \in C$$

$$\alpha,\beta \in V$$

$$\phi \in V \to P$$

$$X, Y \in \mathcal{P}_{closed}((\Sigma \times P) \cup (C \times (V \to P) \cup (V \times P)))$$

$$x,y \in X$$

$$\lambda \in \Sigma \cup V \cup C$$

$$\rho \in P \cup (V \to P) \cup (V \times P)$$

We first define the operators for *finite* processes:

DEFINITION 7.3.

a. Concatenation. $p_0 \circ p = p \circ p_0 = p$. For $p \neq p_0$, $p \circ q = \lambda\sigma.(p(\sigma) \circ q)$, $X \circ q = \{x \circ q \mid x \in X\}$, $\phi \circ q = \lambda\alpha.(\phi(\alpha) \circ q)$, $[\lambda,\rho] \circ q = [\lambda, \rho \circ q]$

b. Union. $p_0 \cup p = p \cup p_0 = p$. For $p,q \neq p_0$ we put $p \cup q = \lambda\sigma.(p(\sigma) \cup q(\sigma))$.

c. Merge. $p_0 \| p = p \| p_0 = p$. For $p,q \neq p_0$ we put

- $p \| q = \lambda\sigma.((p(\sigma) \| q) \cup (q(\sigma) \| p) \cup (p(\sigma) \mid_\sigma q(\sigma)))$.
- $X \| q = \{x \| q : x \in X\}$,
  $[\lambda,\rho] \| q = [\lambda, \rho \| q]$,
  $\phi \| q = \lambda\alpha.(\phi(\alpha) \| q)$.
- $X \mid_\sigma Y = \bigcup \{x \mid_\sigma y : x \in X, y \in Y\}$,
  $[c,\phi] \mid_\sigma [c,[\alpha,p]] = \{[\sigma, \phi(\alpha) \| p]\}$,
  $x \mid_\sigma y = \varnothing$, for x,y not of the above form.

Definition 7.3 is extended in the usual way for p or q infinite.

We are now, at last, in the position to define the denotational meaning for $s \in L_3$. We do not require s to be guarded here, since guardedness is achieved automatically by the 'silent step' at procedure call in the same way as in the operational model. Let $\Gamma = Stm^v \to P$, and let $\gamma \in \Gamma$. We give

DEFINITION 7.4.

1. $\mathcal{D}_3[\![v:=e]\!](\gamma) = \lambda\sigma.\{[\sigma<[\![e]\!](\sigma)/v>, p_0]\}$
2. $\mathcal{D}_3[\![b]\!](\gamma) = \lambda\sigma.$**if** $[\![b]\!](\sigma)$ **then** $\{[\sigma, p_0]\}$ **else** $\varnothing$ **fi**
3. $\mathcal{D}_3[\![c?v]\!](\gamma) = \lambda\sigma.\{[c, \lambda\alpha.\lambda\bar{\sigma}.\{[\bar{\sigma}<\alpha/v>, p_0]\}]\}$
4. $\mathcal{D}_3[\![c!e]\!](\gamma) = \lambda\sigma.\{[c, [[\![e]\!](\sigma), p_0]]\}$
5. $\mathcal{D}_3[\![s_1;s_2]\!](\gamma) = \mathcal{D}_3[\![s_1]\!](\gamma) \circ \mathcal{D}_3[\![s_2]\!](\gamma)$
   $\mathcal{D}_3[\![s_1 + s_2]\!](\gamma) = \mathcal{D}_3[\![s_1]\!](\gamma) \cup \mathcal{D}_3[\![s_2]\!](\gamma)$
   $\mathcal{D}_3[\![s_1 \| s_2]\!](\gamma) = \mathcal{D}_3[\![s_1]\!](\gamma) \| \mathcal{D}_3[\![s_2]\!](\gamma)$
6. $\mathcal{D}_3[\![x]\!](\gamma) = \gamma(x)$
7. $\mathcal{D}_3[\![\mu x[s]]\!](\gamma) = lim_i p_i$, where $(p_0 = p_0$ and$)\ p_{i+1} = \lambda\sigma.\{[\sigma, \mathcal{D}_3[\![s]\!](\gamma<p_i/x>)]\}$

Examples (we omit the γ-arguments)

1. $\mathcal{D}_3[\![(v_1:=0; v_2:=v_1+1) \| (v_2:=2)]\!] =^{df} p =$

$$\lambda\sigma.\quad \{[\sigma<0/v_1>,\ \lambda\bar{\sigma}.\{[\ \bar{\sigma}<\bar{\sigma}(v_1)+1/v_2>,\ \lambda\bar{\bar{\sigma}}.\{[\bar{\bar{\sigma}}<2/v_2>, p_0]\}]\}],$$
$$\cdots,$$
$$[\sigma<2/v_2>,\ \lambda\bar{\sigma}.\{[\ \bar{\sigma}<0/v_1>,\ \lambda\bar{\bar{\sigma}}.\{[\bar{\bar{\sigma}}<\bar{\bar{\sigma}}(v_1)+1/v_2>, p_0]\}]\}]$$
$$\}.$$

Below, we shall discuss how to obtain from this p its state transforming effect p^+: the processes $\lambda\bar{\sigma}.\cdots$, $\lambda\bar{\bar{\sigma}}.\cdots$ are 'ready to execute', and an additional mechanism has to be invoked in order to start their execution.

2. Let $s_1 = c?v$, $s_2 = c!2$. We determine $\mathcal{D}_3[\![s_1 \| s_2]\!] =^{df} p$, where $p = p_1 \| p_2$, $p_i = \mathcal{D}_3[\![s_i]\!]$.
   $p_1 = \lambda\sigma.\{[c, \phi]\}$, where $\phi = \lambda\alpha.\lambda\bar{\sigma}.\{[\bar{\sigma}<\alpha/v>, p_0]\}$.
   $p_2 = \lambda\sigma.\{[c, [2, p_0]]\}$.
   $p_1 \| p_2 = \lambda\sigma.((p_1(\sigma) \| p_2) \cup (p_2(\sigma) \| p_1) \cup (p_1(\sigma) \mid_\sigma p_2(\sigma)))$,
   $p_1(\sigma) \| p_2 = \{[c, \phi \| p_2]\} = \{[c, \lambda\alpha.(\phi(\alpha) \| p_2)]\}$,
   $p_2(\sigma) \| p_1 = \{[c, [2, p_0] \| p_1]\} = \{[c, [2, p_1]]\}$,
   $p_1(\sigma) \mid_\sigma p_2(\sigma) = \{[\sigma, \phi(2) \| p_0]\} = \{[\sigma, \lambda\bar{\sigma}.\{[\bar{\sigma}<2/v>, p_0]\}]\}$

   Altogether, we obtain for p:

$$\lambda\sigma.\{[c, \lambda\alpha.(\phi(\alpha) \| p_2)], [c, [2, p_1]], [\sigma, \lambda\bar{\sigma}.\{[\bar{\sigma}<2/v>, p_0]\}]\}$$

   In a moment we shall see how we may get rid, by applying the function $restr_{\mathbb{P}}$, of the $[c, \cdots]$

parts in this outcome.

We conclude this section, with, firstly, a discussion on the relationship between $\mathcal{O}_3$ and $\mathcal{P}_3$, and, secondly, on the definition of the *yield* p^+ of a process p, retrieving its state transforming function.

As stated already in the introduction of section 7, we have no firm results on the relationship between $\mathcal{O}_3$ and $\mathcal{P}_3$. However, it is not too farfetched to state a conjecture relating $\mathcal{O}_3$ and $\mathcal{P}_3$. We first define a restriction operator $restr_{\mathbb{P}}: P \to P$ in the usual way: For finite p,

$$restr_{\mathbb{P}}(p_0) = p_0,$$

$$restr_{\mathbb{P}}(\lambda\sigma.X) = \lambda\sigma.restr_{\mathbb{P}}(X)$$

$$restr_{\mathbb{P}}(X) = \{[\sigma, restr_{\mathbb{P}}(p')]:[\sigma,p'] \in X\}.$$

For infinite p, $restr_{\mathbb{P}}$ is defined in the usual way by taking limits.

Next, we define *streams*: $P \setminus \{p_0\} \to (\Sigma \to \mathbb{S}(\delta))$ by (finite case only presented; we have no occasion to use $streams(p_0)$):

$$streams(\lambda\sigma.X) = \lambda\sigma.(\sigma.streams(X))$$

$$streams(X) = \bigcup \{streams(x): x \in X\},\ X \neq \varnothing$$

$$= \{\delta\},\ X = \varnothing$$

$$streams([\sigma,p]) = \sigma.streams(p(\sigma)),\ p \neq p_0$$

$$streams([\sigma,p_0]) = \{\sigma\}$$

$$streams([c,\phi]) = streams([c,[\alpha,p]]) = \{\delta\}$$

Note in particular the clause for $streams([\sigma,p])$, $p \neq p_0$. We do not concatenate σ with $streams(p)$. Rather, we deliver σ concatenated with the application of *streams* to the result of *applying* p to σ. (This is another example where a process, ready to execute, is made to start its execution.)

We now conjecture, for each syntactically closed s:

$$\mathcal{O}_3[\![s]\!] = streams(restr_{\mathbb{P}}(\mathcal{P}_3[\![s]\!]))$$

In order to settle the conjecture, it will be necessary to investigate to what extent the argument of section 6.3, in particular the use of ready sets, can be carried over to a nonuniform setting.

As last topic we discuss the yield p^+ of any (non nil) process p. The yield of a process retrieves its state transforming function. For example, let $s \equiv v_1 := 0; v_2 := v_1 + 1$. For $p = \mathcal{P}_3[\![s]\!]$ we obtain

$$p = \lambda\sigma.\{[\sigma<0/\nu_1>, p_1]\}$$

$$p_1 = \lambda\bar{\sigma}.\{[\bar{\sigma}<\bar{\sigma}(\nu_1)+1/\nu_2>, p_0]\}$$

By the definition below, for given input state σ, $p^+(\sigma)$ yields output state $\sigma<0/\nu_1><1/\nu_2>$. This can be understood as follows: Assume that p is applied to some σ. Then the intermediate state $\sigma_1 = \sigma<0/\nu_1>$ is delivered, together with the continuation p_1. Note that p_1 is not (yet) applied to σ_1. Now this application of p_1 to σ_1 -and, in general, of subsequent continuations p_i to corresponding states σ_i- is the purpose of the yield definition. All processes which are ready to execute -but which have suspended their application in view of a possible interleaving action from a parallel component- are made to execute through a cascade of applications triggered by the yield definition. A complication is due to the possible presence of an infinite path in the 'proces tree'. In this case we want to deliver the bottom state as output corresponding to this path. This requires a suitable limit concept for which we take (a simple version of) the Egli-Milner cpo structure. First a few auxiliary definitions. We reinstall $\perp$ (with $\perp \notin \Sigma_\delta$), this time as a state indicating nontermination. Let us put $\Sigma_\perp = \Sigma_\delta \cup \{\perp\}$. We define an order on $\Sigma_\perp$ by putting: $\sigma_1 \sqsubseteq \sigma_2$ iff $\sigma_1 = \perp$ or $\sigma_1 = \sigma_2$. Let $\mathfrak{T} = \mathfrak{P}(\Sigma_\perp)$, and let $T_1, T_2 \in \mathfrak{T}$. We define $T_1 \sqsubseteq_T T_2$ iff $\perp \in T_1$ and $T_1 \subseteq T_2$ or $T_1 = T_2$. (This is in fact the Egli-Milner order from definition 2.15c.) Then $(\mathfrak{T}, \sqsubseteq_T, \{\perp\})$ is a cpo.

We now give

DEFINITION 7.5. The mapping $+ : P \setminus \{p_0\} \to (\Sigma \to \mathfrak{T})$ is given by

$$p^+ = \lambda\sigma.p(\sigma)^+$$

$$X^+ = \sqcup_{T,n} X^{<n>}$$

where $\sqcup_T$ is the lub in $(\mathfrak{T}, \sqsubseteq_T, \{\perp\})$ and $X^{<n>}$ is defined as follows:

$$X^{<0>} = \{\perp\}$$

$$X^{<n+1>} = \bigcup \{x^{<n+1>} : x \in X\}$$

$$[\sigma, p_0]^{<n+1>} = \{\sigma\}$$

$$[c, \cdots]^{<n+1>} = \{\delta\}$$

$$[\sigma, p]^{<n+1>} = p(\sigma)^{<n>} \quad , p \neq p_0$$

Examples

1. Let ψ be some function: $\Sigma \to \Sigma$, and let p be such that it satisfies the equation:

$$p = \lambda\sigma.\{[\sigma, p_0], [\psi(\sigma), p]\}$$

$p(\sigma)^+$ is obtained as lub of the sequence

$$X^{<0>} = \{\bot\}$$

$$X^{<1>} = \bigcup \{[\sigma,p_0]^{<1>},[\psi(\sigma),p]^{<1>}\}$$

$$= \bigcup \{\{\sigma\},p(\psi(\sigma))^{<0>}\}$$

$$= \{\sigma,\bot\}$$

$$X^{<2>} = \bigcup \{[\sigma,p_0]^{<2>},[\psi(\sigma),p]^{<2>}\}$$

$$= \bigcup \{\{\sigma\},p(\psi(\sigma))^{<1>}\}$$

$$= \bigcup \{\{\sigma\},\{[\psi(\sigma),p_0],[\psi(\psi(\sigma)),p]\}^{<1>}\}$$

$$= \{\sigma,\psi(\sigma),\bot\}$$

$$X^{<3>} = \cdots$$

Hence, we obtain $p(\sigma)^+ = \bigsqcup_n X^{<n>} = \{\bot,\sigma,\psi(\sigma),\psi(\psi(\sigma)),\cdots\}$. The $+$-operator unwinds the process p in σ; the crucial step in the definition is $[\sigma,p]^{<n+1>} = p(\sigma)^{<n>}$. We observe that computations for p in σ which determine an infinite path in the 'process tree' contribute $\bot$ to the outcome.

2. Let $p = \lambda\sigma.\{[\sigma<0/\nu_1>, \lambda\bar{\sigma}.\{[\bar{\sigma}<\bar{\sigma}(\nu_1)+1/\nu_2>, \lambda\bar{\bar{\sigma}}.\{[\bar{\bar{\sigma}}<2/\nu_2>,p_0]\}]\}$. (This is part of the process obtained in the first example after definition 7.4.) We show how to calculate $p(\sigma)^+$. Let

$$p_1 =^{df} \lambda\bar{\sigma}.\{[\bar{\sigma}<\bar{\sigma}(\nu_1)+1/\nu_2>,p_2]\}$$

$$p_2 =^{df} \lambda\bar{\bar{\sigma}}.\{[\bar{\bar{\sigma}}<2/\nu_2>,p_0]\}$$

We have $p(\sigma)^+ = \bigsqcup_n X^{<n>}$, where

$$X^{<0>} = \{\bot\}$$

$$X^{<1>} = \bigcup \{[\sigma<0/\nu_1>,p_1]^{<1>}\}$$

$$= \bigcup \{p_1(\sigma<0/\nu_1>)^{<0>}\}$$

$$= \{\bot\}$$

$$X^{<2>} = \cdots = \{\bot\}$$

$$X^{<3>} = \bigcup \{[\sigma<0/\nu_1>,p_1]^{<3>}\}$$

$$= \bigcup \{p_1(\sigma<0/\nu_1>)^{<2>}\}$$

$$= \bigcup \{p_2(\sigma<0/\nu_1><\sigma<0/\nu_1>(\nu_1)+1/\nu_2>)^{<1>}\}$$

$$= \bigcup \{p_2(\sigma<0/\nu_1><1/\nu_2>)^{<1>}\}$$

$$= \bigcup \{\{\sigma<0/v_1><1/v_2><2/v_2>\}\}$$

$$= \{\sigma<0/v_1><2/v_2>\}.$$

This brings our discussion of the semantics of L_3 and, at the same time, of various contrasting themes in the semantics of imperative concurrency, to an end.

References

[Ad] ADA, *The Programming Language ADA, Reference Manual,* American National Standards Institute, Inc. ANSI/MIL-STD-1815A-1983, LNCS 155 Springer, 1983.

[Am] P. AMERICA, *Definition of the programming language POOL-T,* ESPRIT project 415, Doc. Nr. 0091, Philips Research Laboratories, Eindhoven, June 1985.

[ABKR1] P. AMERICA, J.W. DE BAKKER, J.N. KOK, J.J.M.M. RUTTEN, *Operational semantics of a parallel object-oriented language,* CS-R8515, Centre for Mathematics and Computer Science, 1985.

[ABKR2] P. AMERICA, J.W. DE BAKKER, J.N. KOK, J.J.M.M. RUTTEN, *Denotational semantics of a parallel object-oriented language,* in preparation.

[Ap1] K.R. APT, *Recursive assertions and parallel programs,* Acta Inf. 15 (1981) 219-232.

[Ap2] K.R. APT, *Formal justification of a proof system for communicating sequential processes,* J. Assoc. Comput. Mach., 30 1 (1983) 197-216.

[Ap3] K.R. APT (ed.), *Logics and Models of Concurrent systems,* Springer, 1985.

[AN] A. ARNOLD, M. NIVAT, *Metric interpretations of infinite trees and semantics of non-deterministic recursive programs,* Theor. Comp. Science 11 (1980) 181-206.

[Ba] R.J. BACK, *A continuous semantics for unbounded nondeterminism,* Theoret. Comp. Sci. 23 (1983) 187-210.

[dB] J.W. DE BAKKER, *Mathematical theory of program correctness,* Prentice Hall International, London, 1980.

[BBKM] J.W. DE BAKKER, J.A. BERGSTRA, J.W. KLOP, J.-J. CH. MEYER, *Linear time and branching time semantics for recursion with merge,* TCS 34 (1984) 135-156.

[BKo] J.W. DE BAKKER, J.N. KOK, *Towards a uniform topological treatment of streams and functions on streams,* in: Proc. 12th ICALP (W. Brauer, ed.), LNCS 194, Springer (1985), 140-148.

[BMO1] J.W. DE BAKKER, J.-J. CH. MEYER, E.-R. OLDEROG, *Infinte streams and finite observations in the semantics of uniform concurrency,* in: Proceedings 12th ICALP (W. Brauer, ed.), LNCS 194, Springer (1985) 149-157.

[BMO2] J.W. DE BAKKER, J.-J. CH. MEYER, E.-R. OLDEROG, *Infinite streams and finite observations in the semantics of uniform concurrency,* Report CS-R8512, Centre for Mathematics and Computer Science, 1985. (full version of [BMO1]).

[BMO3] J.W. DE BAKKER, J.-J. CH. MEYER, E.-R. OLDEROG, *Hiding discriminates between order and metric in the stream semantics of concurrency,* Report CS-R85.., Centre for Mathematics and Computer Science, to appear.

[BMOZ1] J.W. DE BAKKER, J.-J. CH. MEYER, E.-R. OLDEROG, J.I. ZUCKER, *Transition systems, infinitary languages and the semantics of uniform concurrency,* in: Proceedings 17th ACM STOC, Providence, R.I. (1985) 252-262.

[BMOZ2] J.W. DE BAKKER, J.-J. CH. MEYER, E.-R. OLDEROG, J.I. ZUCKER, *Transition systems, metric spaces and ready sets in the semantics of uniform concurrency,* preprint SUNY at Buffalo, 1985 (full version of BMOZ1), to appear.

[BZ1] J.W. DE BAKKER, J.I. ZUCKER, *Denotational semantics of concurrency,* in: Proceedings 14th Assoc. Comput. Mach. Symp. on Theory of Computing (1982) 153-158.

[BZ2] J.W. DE BAKKER, J.I. ZUCKER, *Processes and the denotational semantics of concurrency,* Inform. and Control 54 (1982) 70-120.

[BZ3] J.W. DE BAKKER, J.I. ZUCKER, *Compactness in semantics for merge and fair merge,* in: Proceedings Workshop Logics of Programs, (E. Clarke & D. Kozen, eds.) Pittsburgh, LNCS 164 Springer (1983) 18-33.

[BZ4] J.W. DE BAKKER, J.I. ZUCKER, *Processes and a fair semantics for the ADA rendez-vous,* in: Proceedings 10th ICALP (J. Diaz, ed.) LNCS 154, Springer (1983) 52-66.

[BK1] J.A. BERGSTRA, J.W. KLOP, *Process algebra for synchronous communication,* Information and Control, 60 (1984) 109-137.

[BK2] J.A. BERGSTRA, J.W. KLOP, *Algebra of communicating processes with abstraction,* TCS 37 (1985) 77-121.

[BK3] J.A. BERGSTRA, J.W. KLOP, *Algebra of communicating processes,* in: Proceedings CWI Symposium (J.W. de Bakker, M. Hazewinkel, J.K. Lenstra, eds.), CWI Monographs, North-Holland, Amsterdam, to appear.

[BKO] J.A. BERGSTRA, J.W. KLOP, E.R. OLDEROG, *Readies and failures in the algebra of communicating processes,* CWI Report CS-R85.., Amsterdam, 1985.

[Be] E. BEST, *Relational semantics of concurrent programs (with some applications),* in: Proceedings IFIP TC2 Working Conference (D. Bjørner, ed.), North-Holland, Amsterdam (1982) 431-452.

[Br] S.D. BROOKES, *On the relationship of CCS and CSP,* in: Proceedings 10th ICALP (J. Diaz, ed.), LNCS 154, Springer (1983) 83-96.

[BHR] S.D. BROOKES, C.A.R. HOARE, A.W. ROSCOE, *A theory of communicating sequential processes,* J. ACM 31 (1984) 560-599.

[BRW] S.D. BROOKES, A.W. ROSCOE, G. WINSKEL (eds.), *Seminar on Concurrency,* LNCS 197, Springer, 1985.

[Bro1] M. BROY, *Fixed point theory for communication and concurrency,* in: Formal Description of Programming Concepts II (D. Bjørner ed.), North-Holland, Amsterdam, (1983) 125-146.

[Bro2] M. BROY, *Applicative real time programming,* in: IFIP Information Processing 83 (R.E.A. Mason, ed.) North-Holland, Amsterdam, (1983) 259-264.

[DM] P. DEGANO, U. MONTANARI, *Liveness properties as convergence in metric spaces,* Proc. 16th ACM STOC (1984) 31-38.

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

[Dij] E.W. DIJKSTRA, *Cooperating Sequential Processes,* in Programming Languages (F. Genuys, ed.), Academic Press, 1968.

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

[FHLR] N. FRANCEZ, C.A.R. HOARE, D.J. LEHMANN, W.P. DE ROEVER, *Semantics of nondeterminism, concurrency and communication,* JCSS 19 (1979) 290-308.

[FLP] N. FRANCEZ, D.J. LEHMANN, A. PNUELI, *A linear history semantics for languages for*

distributed programming, TCS 32 (1984) 25-46.

[GR] W.G. GOLSON, W.C. ROUNDS, *Connections between two theories of concurrency: metric spaces and synchronization trees,* Inform. and Control 57 (1983) 102-124.

[Ha] H. HAHN, *Reelle Funktionen,* Chelsea, New York, 1948.

[He1] M.C.B. HENNESSY, *Synchronous and asynchronous experiments on processes,* Information and Control 59 (1983) 36-83.

[He2] M.C.B. HENNESSY, *An algebraic theory of fair asynchronous communicating processes,* Manuscript, Dept. of Comp. Sci., Univ. of Edinburgh, 1984.

[Ho] C.A.R. HOARE, *Communicating sequential processes,* Comm. ACM 21 (1980) 666-677.

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

[HU] J.E. HOPFCROFT, J.D. ULLMAN, *Introduction to automata theory, languages and computation,* Addison-Wesley, Reading, Mass., 1979.

[In] INMOS LTD., *The Occam Programming Manual,* Prentice-Hall International, London, 1984.

[Ke] R. KELLER, *Formal verification of parallel programs,* Comm. Assoc. Comput. Mach. 19 (1976) 371-384.

[Ku] R. KUIPER, *An operational semantics for bounded nondeterminism equivalent to a denotational one,* IFIP TC2-MC Symp. on Algorithmic Languages (J.W. de Bakker & J.C. van Vliet, eds.) North-Holland, Amsterdam (1981) 373-398.

[Ma] A. MAZURKIEWICZ, *Concurrent program schemes and their interpretations,* DAIMI, PB 78, Aarhus University, 1977.

[Me1] J.-J. CH. MEYER, *Fixed points and the arbitrary and fair merge of a fairly simple class of processes,* Tech. Reports IR-89/IR-92, Free University, Amsterdam, 1984.

[Me2] J.-J. CH. MEYER, *Programming calculi based on fixed point transformations: semantics and applications,* dissertation, Free University of Amsterdam, 1985.

[Mic] E. MICHAEL, *Topologies on spaces of subsets,* Trans. AMS 71 (1951) 152-182.

[MM] G. MILNE, R. MILNER *Concurrent processes and their syntax,* J. ACM 26 (1979) 302-321.

[Mi1] R. MILNER, *Processes: a mathematical model of computing agents,* in: Proceedings Logic Coll. 73 (Rose & Shepherdson, eds.) North-Holland, Amsterdam, 1983.

[Mi2] R. MILNER, *A calculus of communicating systems,* LNCS 92, Springer, 1980.

[Mi3] R. MILNER, *Calculi for synchrony and asynchrony,* TCS 25 (1983) 267-310.

[dNH] R. DE NICOLA, M.C.B. HENNESSY, *Testing equivalences for processes,* TCS 34 (1984) 83-134.

[Ni1] M. NIVAT, *Infinite words, infinite trees, infinite computations,* Foundations of Computer Science III. 2, Mathematical Centre Tracts 109 (1979) 3-52.

[Ni2] M. NIVAT, *Synchronization of concurrent processes,* in Formal Language Theory (R.V. Book, ed.), Academic Press, New York (1980) 429-454.

[OH1] E.-R. OLDEROG, C.A.R. HOARE, *Specification-oriented semantics for communicating*

processes, in: Proceedings 10th ICALP (J. Diaz, ed.), LNCS 154 Springer (1983) 561-572.

[OH2] E.-R. OLDEROG, C.A.R. HOARE, *Specification-oriented semantics for communicating processes,* Tech. Monograph PRG-37, Prog. Research Group, Oxford Univ., 1984 (to appear in Acta Informatica).

[Pa] D. PARK, *Concurrency and automata on infinite sequences,* Proceedings, Theor. Comp. Sci. (P. Deussen, ed.), LNCS 104, Springer, 1981.

[Pl1] G.D. PLOTKIN, *A powerdomain construction,* SIAM J. Comp. 5 (1976) 452-487.

[Pl2] G.D. PLOTKIN, *Dijkstra's predicate transformers and Smyth's powerdomains,* in: Abstract Software Specification (D. Bjørner ed.), LNCS 86, Springer (1980) 527-553.

[Pl3] G.D. PLOTKIN, *A structural approach to operational semantics,* Report DAIMI FN-19, Comp. Sci. Dept., Aarhus Univ. 1981.

[Pl4] G.D. PLOTKIN, *An operational semantics for CSP,* in: Formal Description of Programming Concepts II (D. Bjørner ed.) North-Holland, Amsterdam (1983) 199-223.

[Pn] A. PNUELI, *Linear and branching structures in the semantics and logics of reactive systems,* in: Proceedings 12th ICALP (W. Brauer, ed.), LNCS 194, Springer (1985) 15-32.

[Ro] W.C. ROUNDS *On the relationships between Scott domains, synchronization trees and metric spaces,* Report Univ. of Michigan CRL-TR-25-83, 1983.

[RB] W.C. ROUNDS, S.D. BROOKES, *Possible futures, acceptances, refusals, and communicating processes,* in: Proceedings 22nd Symp. Found. of Comp. Sc. IEEE (1981), 140-149.

[RS] M.O. RABIN, D.S. SCOTT, *Finite automata and their decision problems,* IBM J. Res. 3:2, 1959.

[Sc] D.S. SCOTT, *Domains for denotational semantics,* Proceedings 9th ICALP (M.Nielsen & E.M. Schmidt, eds.) LNCS 140, Springer (1982) 577-613.

[Sm] M.B. SMYTH, *Power domains,* JCSS 16 (1978) 23-26.

[St] J. STOY, *Denotational semantics: The Scott-Strachey approach to programming language theory,* MIT Press, Cambridge, Mass, 1977.

[Wi] G. WINSKEL, *Synchronisation trees,* TCS 34 (1984) 33-82.