Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

Topological Models for Higher Order Control Flow

J.W. de Bakker, F. van Breugel

Computer Science/Department of Software Technology

CS-R9340 1993

Topological Models for Higher Order Control Flow

J.W. de Bakker[1,2] and F. van Breugel[1,2]

[1] *CWI*
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
[2] *Vrije Universiteit*
P.O. Box 7161, 1007 MC Amsterdam, The Netherlands

Abstract

Semantic models are presented for two simple imperative languages with higher order constructs. In the first language the interesting notion is that of second order assignment $x := s$, for x a procedure variable and s a statement. The second language extends this idea by a form of higher order communication, with statements $c\,!\,s$ and $c\,?\,x$, for c a channel. We develop operational and denotational models for both languages, and study their relationships. Both in the definitions and the comparisons of the semantic models, convenient use is made of some tools from (metric) topology. The operational models are based on (SOS-style) transition systems; the denotational definitions use domains specified as solutions of domain equations in a category of 1-bounded complete ultrametric spaces. In establishing the connection between the two kinds of models, fruitful use is made of Rutten's *processes as terms* technique. Another new tool consists in the use of *metric* transition systems, with a metric defined on the configurations of the system. In addition to higher order programming notions, we use higher order definitional techniques, e.g., in defining the semantic mappings as fixed points of (contractive) higher order operators. By Banach's theorem, such fixed points are unique, yielding another important proof principle for our paper.

AMS Subject Classification (1991): 68Q55
CR Subject Classification (1991): D.3.1, F.3.2
Keywords & Phrases: operational semantics, denotational semantics, complete metric space, second order assignment, second order communication, processes as terms, metric transition system
Note: The work of F. van Breugel was partially supported by the Netherlands *Nationale Faciliteit Informatica* programme, project Research and Education in Concurrent Systems (REX). This paper will appear in Proceedings of the Ninth Conference on the Mathematical Foundations of Programming Semantics, New Orleans, LA, USA, April 7-10, 1993.

Introduction

In recent years, the study of higher order programming notions has become a central topic in the field of semantics. Seminal in this development have been two schools of research, viz. that of (typed) λ-calculus in the area of functional programming (see, e.g., [Bar92] for a survey of the current situation), and that of higher order processes in the theory of concurrency (see, e.g., [AR87, Tho90, MPW92]). ([LTLG92] can be used for a quick overview of much of the relevant literature.) The aim of the present paper is to provide another perspective on this problem area by studying higher order notions embedded in the traditional setting of imperative languages. First, we study *second order assignment*: the statement $x := s$, for x a procedure variable and s a statement, assigns s to x. In the operational semantics, this is modelled by storing (the syntactic entity) s in the current 'syntactic' state. Denotationally, the (function which is the) meaning of s is stored in the 'semantic' state. The second notion we study is *second order communication*. Recall that in a CSP- or occam-like language value-passing communication is expressed by the two actions $c\,!\,e$ and $c\,?\,v$ occurring in two parallel components (c a channel, e some expression, and v an individual variable), and synchronised execution

of these actions results in the transmission of the current value of e to v. A second order variant of this is the pair of communication constructs $c\,!\,s$ and $c\,?\,x$ (c, s, and x as above). Now a higher order value is passed at the moment of synchronised execution: in the operational semantics, we pass s (again a syntactic object); denotationally, the meaning of s is transmitted.

Though these notions are, we hope, conceptually quite simple, a not so simple arsenal of semantic tools is necessary to make the ideas just sketched precise, and to obtain a full picture of the relationships between the operational ($\mathcal{O}$) and denotational ($\mathcal{D}$) models. In both kinds of models, topological techniques play an essential rôle. More specifically, we work in a category of 1-bounded complete ultrametric spaces, and a variety of functors on this category is used to specify the domains we work with. (This type of domain equations originated with [BZ82]; the general theory is due to [AR89]. See also [BR92] for many further applications.)

For reasons of presentation, in addition to the languages with higher order assignment ($\mathcal{L}_{as_2}$) and communication ($\mathcal{L}_{co_2}$) we also discuss two simpler languages with only first order assignment ($\mathcal{L}_{as}$) and communication ($\mathcal{L}_{co}$), respectively. This allows a more leisurely development of the machinery: in particular, we are able to demonstrate in a simple setting another higher order phenomenon which is pervasive in this paper, viz. the use of (contractive) higher order mappings in both the definition and the comparison of semantic meaning functions. Each of the $\mathcal{O}$'s or $\mathcal{D}$'s to be defined is obtained as (unique) fixed point of some higher order mapping $\Phi_{\mathcal{O}}$ or $\Phi_{\mathcal{D}}$. By the uniqueness property, in order to establish $\mathcal{O} = \mathcal{D}$, it suffices to show, e.g., that $\Phi_{\mathcal{O}}(\mathcal{D}) = \mathcal{D}$.

The definition of each of the $\mathcal{O}$'s follows the customary pattern in that it is derived from some (SOS-style) transition system ([Plo81]). Mostly, these systems are *finitely branching*, a property on which the *compactness* of the resulting sets of meanings is based. However, in the comparative study of $\mathcal{L}_{co_2}$ we need a generalisation to *compactly branching* transition systems. This is, in turn, based on an extension of the metric framework consisting in the introduction of a metric on the configurations of the transition system (rather than only having a metric based on the standard distance between sequences of actions generated by successive transitions).

The key idea in the semantic analysis of $\mathcal{L}_{as_2}$ is the introduction of both syntactic and semantic states, and of a suitable mapping linking the two. Whereas the syntactic states are an immediate extension of those used for $\mathcal{L}_{as}$, the set of semantic states requires a system of (reflexive) domain equations for its specification. Once the appropriate definitions have become available, a concise (statement and) proof of the relationship between $\mathcal{O}$ and $\mathcal{D}$ is possible, thanks to the rather powerful general methodology.

The first order language $\mathcal{L}_{co}$ is a fairly typical language with imperative concurrency. Our design of $\mathcal{O}$ for $\mathcal{L}_{co}$ exhibits only some mild variations compared with the traditional approach. The denotational $\mathcal{D}$ is based on a 'branching time' process domain P of the 'nonuniform' variety (processes have a functional dependence on the state). It is not difficult to show (and implicit in [BZ82]) that P is *strongly extensional*: with a slight adaptation of the usual definition of bisimilarity, we have that bisimilarity on P coincides with identity. The various semantic operators on P may as well be defined by higher order techniques. The relationship between $\mathcal{O}$ and $\mathcal{D}$ for $\mathcal{L}_{co}$ involves a *trace* mapping from the denotational 'branching time' to the operational 'linear time' domain: among others, the branching structure is collapsed, and failing attempts at communication are deleted (and deadlock is delivered if no 'proper' action remains).

The paper culminates in the semantic study of $\mathcal{L}_{co_2}$, bringing a synthesis of many of the earlier techniques. The denotational domain, albeit rather complex due to the use of three domain equations, allows an appealingly simple denotational definition. This domain can also be shown to be strongly extensional (with some higher order generalisation of the bisimilarity definition, cf., e.g., [AGR92, MS92]). More work is needed to link $\mathcal{O}$ and $\mathcal{D}$. First, an idea already used for $\mathcal{L}_{co}$, viz. to design a variant of $\mathcal{O}$ delivering results in the denotational domain, is applied again. However, for $\mathcal{L}_{co_2}$

a complication arises, inducing the appearance of 'processes as terms' ([Rut92]). Also, this is the point where, as signalled earlier, a compactly branching transition system appears, a notion which presupposes a metric on the configurations ([Bre94]). In the final stage of the proof relating $\mathcal{O}$ and $\mathcal{D}$, a lemma relating the transitions of both the original system (on which $\mathcal{O}$ for $\mathcal{L}_{co_2}$ is based) and of the extended system (in which the configurations may involve semantic processes) provides the key technical step.

In the final section, the paper summarises the relationships between $\mathcal{O}$ and $\mathcal{D}$ for the four languages considered. We see as one of the achievements of our paper the transparency of the successive refinements, going from the simple $\mathcal{O} = \mathcal{D}$ result for $\mathcal{L}_{as}$ to the more elaborate theorem for $\mathcal{L}_{co_2}$.

We conclude this introduction with some remarks on related work. The idea to handle second order assignment $x := s$ through the storing of a pair (x, s) in the (syntactic) state is close to the explicit substitution (in the framework of the λ-calculus) of [Cur88, ACCL90], albeit that some stack like nesting of states - omitted in this paper not to overload the presentation - would be needed to allow a full correspondence. The language $\mathcal{L}_{co_2}$ should, after some massaging of the specific operator for parallelism, be able to at least model a key part of Thomsen's CHOCS ([Tho89, Tho90]), viz. that sublanguage which he uses to encode the lazy λ-calculus. However, a precise statement and, especially, a full proof of this claim demands a lot of further work. Other connections to explore include the relationships with the π-calculus ([MPW92, Mil92]), the higher order π-calculus ([San92, San93]), and the γ-calculus ([Bou89, BB92], cf. also [JP90]). In the π-calculus, channel names are transmitted rather than processes, so an immediate correspondence is not to be expected. For another reason, the same holds for the γ-calculus: the notion of sequential composition used there is essentially different from ours.

1. A sequential language with assignment

The first language we discuss, viz. $\mathcal{L}_{as}$, is quite simple, and chosen especially to illustrate the use of higher order techniques in defining and relating semantic models. Also, it prepares the way for the more interesting language with second order assignment considered in the next section. For $\mathcal{L}_{as}$, we shall define both $\mathcal{O}$ (operational) and $\mathcal{D}$ (denotational) semantics as (unique) fixed point of a suitable contractive mapping [1]. Banach's theorem[2] applies, since all spaces involved are complete. The semantics $\mathcal{O}$ and $\mathcal{D}$ shall be related by showing that both are fixed points of the same contractive mapping.

Let $(v \in) \mathit{IVar}$, $(x \in) \mathit{PVar}$ be alphabets of individual and procedure variables. Let $(e \in) \mathit{Exp}$ be a class of simple expressions (syntax left unspecified).

Definition 1.1 The language $\mathcal{L}_{as}$ is defined by

$$s ::= v := e \mid s\,;\, s \mid s + s \mid x \mid \mu x\,[s].$$

The prefix μx binds occurrences of procedure variable x. Our semantic definitions will throughout be given for closed constructs (no free procedure variables) only. To define the operational semantics we shall use transition systems. The configurations of the transition system are pairs of resumptions and states.

[1] Let (X, d_X) and $(X', d_{X'})$ be metric spaces. A function $f : X \to X'$ is called contractive if there exists an ϵ, with $0 \leq \epsilon < 1$, such that, for all x and x',

$$d_{X'}(f(x), f(x')) \leq \epsilon \cdot d_X(x, x').$$

[2] Let (X, d_X) be a complete metric space. If $f : X \to X$ is contractive then f has a unique fixed point $\mathit{fix}(f)$ (cf. [Ban22]).

DEFINITION 1.2 The class Res_1 of resumptions is defined by

$$r ::= \mathrm{E} \mid s : r.$$

The set $State_1$ of states is defined by

$$(\sigma \in)\, State_1 = IVar \to Val,$$

for $(\alpha \in)\, Val$ some set of values.

The (empty) resumption E will be used to denote termination. The state $\sigma\{\alpha/v\}$ has value α in v and equals σ elsewhere. Let $\mathcal{V}(e)(\sigma)$ denote the value of expression e in state σ. Let $s\{s'/x\}$ denote syntactic substitution of statement s' for the free occurrences of procedure variable x in statement s. The transition system $\mathcal{T}_1$ is introduced in

DEFINITION 1.3 The transition relation $\to$ of $\mathcal{T}_1$ is the smallest subset of

$$(Res_1 \times State_1) \times (Res_1 \times State_1)$$

satisfying the rules given below. A rule of the form

$$\text{if } [r_1, \sigma_1] \to [r, \sigma] \text{ then } [r_2, \sigma_2] \to [r, \sigma]$$

will be abbreviated to

$$[r_2, \sigma_2] \to_0 [r_1, \sigma_1];$$

the 0-subscript indicates that we have here a *zero-step* transition.

$$\begin{array}{lll}
(1) & [v := e : r, \sigma] & \to [r, \sigma\{\alpha/v\}], \text{ where } \alpha = \mathcal{V}(e)(\sigma) \\
(2) & [(s_1 ; s_2) : r, \sigma] & \to_0 [s_1 : (s_2 : r), \sigma] \\
(3) & [(s_1 + s_2) : r, \sigma] & \to_0 [s_1 : r, \sigma] \\
(4) & [(s_1 + s_2) : r, \sigma] & \to_0 [s_2 : r, \sigma] \\
(5) & [\mu x\, [s] : r, \sigma] & \to [s\{\mu x\, [s]/x\} : r, \sigma]
\end{array}$$

In the operational semantics we collect successive transitions. Each resumption is mapped to an element of the semantic domain P_1 presented in

DEFINITION 1.4 The domain P_1 is defined by

$$(p \in)\, P_1 = State_1 \to \mathcal{P}_{nc}(State_1^\infty).$$

The set $(\varsigma \in)\, State_1^\infty = State_1^* \cup State_1^\omega$ of finite and infinite sequences of states is endowed with the 1-bounded complete ultrametric d specified by

$$d(\varsigma, \varsigma') = \begin{cases} 0 & \text{if } \varsigma = \varsigma' \\ 2^{-n} & \text{otherwise} \end{cases}$$

where n is the length of the longest common prefix of ς and ς'. According to Kuratowski's theorem[3], the set $\mathcal{P}_{nc}(State_1^\infty)$ of nonempty compact subsets of $State_1^\infty$ endowed with the Hausdorff metric is a 1-bounded complete ultrametric space.

DEFINITION 1.5 The higher order mapping $\Phi_{\mathcal{O}^*} : (Res_1 \to P_1) \to (Res_1 \to P_1)$ is defined by

$$\begin{aligned}\Phi_{\mathcal{O}^*}(\phi)(\mathrm{E}) &= \lambda\sigma\,.\,\{\varepsilon\}\\ \Phi_{\mathcal{O}^*}(\phi)(s:r) &= \lambda\sigma\,.\,\bigcup\{\,\sigma'\cdot\phi(r')(\sigma') \mid [s:r,\,\sigma] \to [r',\,\sigma']\,\}\end{aligned}$$

The operational semantics $\mathcal{O}^* : Res_1 \to P_1$ is defined by

$$\mathcal{O}^* = fix(\Phi_{\mathcal{O}^*}).$$

In the above definition of $\Phi_{\mathcal{O}^*}$, $\sigma'\cdot\phi(r')(\sigma')$ is the result of prefixing the set of state sequences $\phi(r')(\sigma')$ by the state σ'. The well-definedness proof of $\Phi_{\mathcal{O}^*}$ exploits the fact that $\mathcal{T}_1$ is finitely branching. Obviously, $\Phi_{\mathcal{O}^*}$ is contractive. According to Banach's theorem, $\Phi_{\mathcal{O}^*}$ has a unique fixed point.

DEFINITION 1.6 The operational semantics $\mathcal{O} : \mathcal{L}_{as} \to P_1$ is defined by

$$\mathcal{O}(s) = \mathcal{O}^*(s:\mathrm{E}).$$

In the denotational semantics, we restrict ourselves to nonexpansive mappings[4] (notation $\to^1$).

DEFINITION 1.7 The higher order mapping $\Phi_{\mathcal{D}} : (\mathcal{L}_{as} \to P_1 \to^1 P_1) \to (\mathcal{L}_{as} \to P_1 \to^1 P_1)$ is defined by

$$\begin{aligned}\Phi_{\mathcal{D}}(\phi)(v := e)(p) &= \lambda\sigma\,.\,(\sigma\{\alpha/v\}\cdot p(\sigma\{\alpha/v\})),\text{ where }\alpha = \mathcal{V}(e)(\sigma)\\ \Phi_{\mathcal{D}}(\phi)(s_1\,;\,s_2)(p) &= \Phi_{\mathcal{D}}(\phi)(s_1)(\Phi_{\mathcal{D}}(\phi)(s_2)(p))\\ \Phi_{\mathcal{D}}(\phi)(s_1+s_2)(p) &= \lambda\sigma\,.\,(\Phi_{\mathcal{D}}(\phi)(s_1)(p)(\sigma)\cup\Phi_{\mathcal{D}}(\phi)(s_2)(p)(\sigma))\\ \Phi_{\mathcal{D}}(\phi)(\mu x\,[s])(p) &= \lambda\sigma\,.\,(\sigma\cdot\phi(s\{\mu x\,[s]/x\})(p)(\sigma))\end{aligned}$$

The denotational semantics $\mathcal{D} : \mathcal{L}_{as} \to P_1 \to^1 P_1$ is defined by

$$\mathcal{D} = fix(\Phi_{\mathcal{D}}).$$

The nonexpansiveness of $\Phi_{\mathcal{D}}(\phi)(s)$ and the contractiveness of $\Phi_{\mathcal{D}}$ can be proved by structural induction. Note that this definition of $\mathcal{D}$ implies, e.g., that $\mathcal{D}(\mu x\,[s])(p) = \lambda\sigma\,.\,(\sigma\cdot\mathcal{D}(s\{\mu x\,[s]/x\})(p)(\sigma))$. Well-definedness of $\mathcal{D}$ is a consequence of the contractiveness of $\Phi_{\mathcal{D}}$ (here ensured by the σ-step) rather than of a direct argument by structural induction on s.

DEFINITION 1.8 The denotational semantics $\mathcal{D}^* : Res_1 \to P_1$ is defined by

[3] If (X, d_X) is a 1-bounded complete ultrametric space then the set of nonempty and compact subsets of X, $\mathcal{P}_{nc}(X)$, endowed with the Hausdorff metric based on d_X is a 1-bounded complete ultrametric space (cf. [Kur56]).

[4] Let (X, d_X) and $(X', d_{X'})$ be metric spaces. A function $f : X \to X'$ is called nonexpansive if, for all x and x',

$$d_{X'}(f(x), f(x')) \le d_X(x, x').$$

$$\mathcal{D}^*(\mathrm{E}) = \lambda\sigma\,.\,\{\varepsilon\}$$
$$\mathcal{D}^*(s:r) = \mathcal{D}(s)(\mathcal{D}^*(r))$$

The operational and denotational semantics are related in

THEOREM 1.9 $\mathcal{O}^* = \mathcal{D}^*$.

PROOF For this theorem, we will sketch two alternative proofs.

1. We can prove that, for all r,

   $$\Phi_{\mathcal{O}^*}(\mathcal{D}^*)(r) = \mathcal{D}^*(r)$$

   by induction on the complexity of r. For example, for the resumption $(s_1\,;\,s_2):r$ we have that

   $$\begin{aligned}
   &\Phi_{\mathcal{O}^*}(\mathcal{D}^*)((s_1\,;\,s_2):r)\\
   &\quad= \Phi_{\mathcal{O}^*}(\mathcal{D}^*)(s_1:(s_2:r)) \quad \text{[the definition of the complexity is such}\\
   &\quad= \mathcal{D}^*(s_1:(s_2:r)) \qquad\qquad \text{that the induction hypothesis applies here]}\\
   &\quad= \mathcal{D}(s_1)(\mathcal{D}(s_2)(\mathcal{D}^*(r)))\\
   &\quad= \mathcal{D}^*((s_1\,;\,s_2):r).
   \end{aligned}$$

   Since $\mathcal{O}^*$ and $\mathcal{D}^*$ are both fixed point of $\Phi_{\mathcal{O}^*}$ and $\Phi_{\mathcal{O}^*}$ has a unique fixed point, $\mathcal{O}^*$ and $\mathcal{D}^*$ must be equal.

2. We can also prove that, for all r,

   $$d(\mathcal{O}^*(r),\mathcal{D}^*(r)) \le \tfrac{1}{2}\cdot\sup\{\,d(\mathcal{O}^*(r'),\mathcal{D}^*(r')) \mid r' \in \mathit{Res}_1\,\}$$

   by induction on the complexity of r. For example, for the resumption $v := e : r$ we have that

   $$\begin{aligned}
   &d(\mathcal{O}^*(v:=e:r),\mathcal{D}^*(v:=e:r))\\
   &\quad= d(\lambda\sigma\,.\,(\sigma\{\alpha/v\}\cdot\mathcal{O}^*(r)(\sigma\{\alpha/v\})),\lambda\sigma\,.\,(\sigma\{\alpha/v\}\cdot\mathcal{D}^*(r)(\sigma\{\alpha/v\})))\\
   &\quad= \tfrac{1}{2}\cdot d(\mathcal{O}^*(r),\mathcal{D}^*(r))\\
   &\quad\le \tfrac{1}{2}\cdot\sup\{\,d(\mathcal{O}^*(r'),\mathcal{D}^*(r')) \mid r' \in \mathit{Res}_1\,\}.
   \end{aligned}$$

   Consequently, for all r, $d(\mathcal{O}^*(r),\mathcal{D}^*(r)) = 0$. Hence $\mathcal{O}^* = \mathcal{D}^*$.

□

The first proof follows [KR90] (cf. [BM88]), but with a substantial simplification thanks to our avoiding procedure environments.

COROLLARY 1.10 *For all s, $\mathcal{O}(s) = \mathcal{D}(s)(\lambda\sigma\,.\,\{\varepsilon\})$.*

2. A SEQUENTIAL LANGUAGE WITH SECOND ORDER ASSIGNMENT

The central notion of this section is second order assignment, in the form of the statement $x := s$, for s itself a statement. In the operational semantics, the routine (program text) s is stored in the syntactic state σ as value for x; in the denotational semantics, the meaning $\mathcal{D}(s)$ is stored as value for x in the semantic state ρ. The definition of $\mathcal{O}$ and $\mathcal{D}$ for $\mathcal{L}_{as_2}$ allows a particularly succinct (statement and) proof of the relationship between $\mathcal{O}$ and $\mathcal{D}$.

DEFINITION 2.1 The language $\mathcal{L}_{as_2}$ is defined by

$$s ::= v := e \mid s\,;\,s \mid s + s \mid x \mid x := s.$$

The configurations of the transition system defining the operational semantics are pairs of resumptions (defined as in the previous section, but now named Res_2) and *syntactic states*, which are introduced in

DEFINITION 2.2 The set $SynState_2$ of syntactic states is defined by

$$(\sigma \in)\, SynState_2 = (IVar \to Val) \times (PVar \to \mathcal{L}_{as_2}).$$

Let, for the state $\sigma = (\sigma_1, \sigma_2)$, the states $\sigma\{\alpha/v\}$ and $\sigma\{s/x\}$ be short for $(\sigma_1\{\alpha/v\}, \sigma_2)$ and $(\sigma_1, \sigma_2\{s/x\})$, respectively. The transition system $\mathcal{T}_2$ is introduced in

DEFINITION 2.3 The transition relation $\to$ of $\mathcal{T}_2$ is the smallest subset of

$$(Res_2 \times SynState_2) \times (Res_2 \times SynState_2)$$

satisfying (1), (2), (3), (4) from Definition 1.3, and

$$(6) \quad [x : r,\, \sigma] \to [\sigma\,(x) : r,\, \sigma]$$
$$(7) \quad [x := s : r,\, \sigma] \to [r,\, \sigma\{s/x\}]$$

The definitions of $\mathcal{O}^*$ and $\mathcal{O}$ follow those of $\mathcal{O}^*$ and $\mathcal{O}$ of the previous section, but now using transition system $\mathcal{T}_2$ and semantic domain P_2, which is obtained from P_1 by replacing $State_1$ by $SynState_2$. We next present the (system of) domain equations[5] for the collection of *semantic states* $SemState_2$ and P_3, the denotational domain for $\mathcal{L}_{as_2}$.

DEFINITION 2.4 The domains $SemState_2$ and P_3 are defined by

$$(\rho \in)\, SemState_2 \cong (IVar \to Val) \times (PVar \to id_{\frac{1}{2}}(P_3 \to^1 P_3))$$
$$(p \in)\, P_3 \cong SemState_2 \to^1 \mathcal{P}_{nc}\,(SemState_2^{\infty})$$

DEFINITION 2.5 The denotational semantics $\mathcal{D} : \mathcal{L}_{as_2} \to P_3 \to^1 P_3$ is defined by

$$\begin{aligned}
\mathcal{D}\,(v := e)(p) &= \lambda\rho\,.\,(\rho\{\alpha/v\} \cdot p\,(\rho\{\alpha/v\})), \text{ where } \alpha = \mathcal{V}\,(e)(\rho)\\
\mathcal{D}\,(s_1\,;\, s_2)(p) &= \mathcal{D}\,(s_1)(\mathcal{D}\,(s_2)(p))\\
\mathcal{D}\,(s_1 + s_2)(p) &= \lambda\rho\,.\,(\mathcal{D}\,(s_1)(p)(\rho) \cup \mathcal{D}\,(s_2)(p)(\rho))\\
\mathcal{D}\,(x)(p) &= \lambda\rho\,.\,(\rho \cdot \rho\,(x)(p)(\rho))\\
\mathcal{D}\,(x := s)(p) &= \lambda\rho\,.\,(\rho\{\psi/x\} \cdot p\,(\rho\{\psi/x\})), \text{ where } \psi = \mathcal{D}\,(s)
\end{aligned}$$

The denotational semantics $\mathcal{D}$ closely follows the structure of the rules in transition system $\mathcal{T}_2$. Consider, for example, the case that a rule $[r, \sigma] \to [r', \sigma']$ (or $[r, \sigma] \to_0 [r', \sigma']$) is the sole rule for configuration $[r, \sigma]$ in $\mathcal{T}_2$. Let p and p' denote the denotational meanings of r and r', and let ρ and ρ' be the semantic states corresponding to σ and σ' (cf. Definition 2.6). Then the formula

[5] To solve these domain equations, we work in a category of 1-bounded complete ultrametric spaces and apply the methodology of solving domain equations in this category as developed in [AR89]. Functors F appearing in domain equations $X \cong F(X)$ - or rather $(X, d_X) \cong F(X, d_X)$ - with $\cong$ denoting isometry, may be built from the familiar operations on 1-bounded complete ultrametric spaces such as Cartesian product, disjoint union, (nonexpansive) function space, and (nonempty) compact power set, and the operation $id_{1/2}$ ($id_{1/2}(X, d_X) = (X, \frac{1}{2} \cdot d_X)$), starting from given 1-bounded complete ultrametric spaces (A, d_A) and the unknown space (X, d_X). The operation $id_{1/2}$ is used in particular to ensure contractiveness of the functor F, which induces uniqueness of the solution up to isometry.

$p(\rho) = \rho' \cdot p'(\rho')$ (or $p(\rho) = p'(\rho')$) expresses the denotational counterpart of this rule. In this way the clause for $\mathcal{D}(x)(p)(\rho)$ may be understood from clause (6) of Definition 2.3.

The definition of $\mathcal{D}^*$ follows that of $\mathcal{D}^*$ of the previous section. To each syntactic state a corresponding semantic state is assigned by the mapping *sem* introduced in

DEFINITION 2.6 The mapping $sem : SynState_2 \to SemState_2$ is defined by

$$sem(\sigma) = (\sigma_1, \lambda x \,.\, \lambda p \,.\, \mathcal{D}(\sigma_2(x))(p)).$$

The mapping *sem* is extended in the natural way to a mapping from $\mathcal{P}_{nc}(SynState_2^\infty)$ to $\mathcal{P}_{nc}(SemState_2^\infty)$. By means of this mapping the operational and denotational semantics are related in

THEOREM 2.7 *For all r and σ, $sem(\mathcal{O}^*(r)(\sigma)) = \mathcal{D}^*(r)(sem(\sigma))$.*

PROOF This proof follows the second proof of Theorem 1.9. For example, for resumption $x := s : r$ we have that

$$\begin{aligned}
& d(sem(\mathcal{O}^*(x := s : r)(\sigma)), \mathcal{D}^*(x := s : r)(sem(\sigma))) \\
&= d(sem(\sigma\{s/x\}) \cdot \mathcal{O}^*(r)(\sigma\{s/x\})), \mathcal{D}(x := s)(\mathcal{D}^*(r))(sem(\sigma))) \\
&= d(sem(\sigma\{s/x\}) \cdot sem(\mathcal{O}^*(r)(\sigma\{s/x\})), sem(\sigma)\{\mathcal{D}(s)/x\} \cdot \mathcal{D}^*(r)(sem(\sigma)\{\mathcal{D}(s)/x\})) \\
&= \tfrac{1}{2} \cdot d(sem(\mathcal{O}^*(r)(\sigma\{s/x\})), \mathcal{D}^*(r)(sem(\sigma)\{\mathcal{D}(s)/x\})) \\
&\leq \tfrac{1}{2} \cdot \sup\{d(sem(\mathcal{O}^*(r')(\sigma')), \mathcal{D}^*(r')(sem(\sigma')) \mid r' \in Res_2,\, \sigma' \in State_2\},
\end{aligned}$$

since $sem(\sigma\{s/x\}) = sem(\sigma)\{\mathcal{D}(s)/x\}$.

□

COROLLARY 2.8 *For all s and σ, $sem(\mathcal{O}(s)(\sigma)) = \mathcal{D}(s)(\lambda\rho \,.\, \{\varepsilon\})(sem(\sigma))$.*

3. A PARALLEL LANGUAGE WITH COMMUNICATION

The language $\mathcal{L}_{co}$ studied here has first order communication (synchronised transmission of simple values) as its main concept. $\mathcal{L}_{co}$ is close to a language such as CSP ([Hoa85]); again, its main motivation in the present context is to pave the way for the second order variant. A further simplification with respect to the usual languages of this kind is that we assume one global state, rather than a distribution of local states over the various parallel components. The design of a mechanism for local states is well-understood (see, e.g., [ABKR89]), and we have kept it separate from the present development in order not to burden the presentation.

Let $(c \in)\ Chan$ be an alphabet of channel names.

DEFINITION 3.1 The language $\mathcal{L}_{co}$ is defined by

$$s ::= v := e \mid c\,!\,e \mid c\,?\,v \mid s\,;\,s \mid s + s \mid s \parallel s \mid x \mid \mu x\,[s].$$

The configurations of the transition system are pairs of resumptions and extended states.

DEFINITION 3.2 The class Res_3 of resumptions is defined by

$$r ::= \mathrm{E} \mid s.$$

The set $State_3$ of states is defined by

$$(\sigma \in)\, State_3 = State_1.$$

The set $State_3^{ext}$ of extended states is defined by

$$(\eta \in)\, State_3^{ext} = State_3 \cup (Chan \times Val) \cup (Chan \times IVar).$$

In the transition system, we will use the extended state (c, α) to denote that the value α is sent on channel c, and we will use (c, v) to denote that the value received on channel c should be assigned to the individual variable v. The transition system $\mathcal{T}_3$ is introduced in

DEFINITION 3.3 The transition relation $\rightarrow$ of $\mathcal{T}_3$ is the smallest subset of

$$(Res_3 \times State_3^{ext}) \times (Res_3 \times State_3^{ext})$$

satisfying

(1) $[v := e, \sigma] \rightarrow [\mathrm{E}, \sigma\{\alpha/v\}]$, where $\alpha = \mathcal{V}(e)(\sigma)$

(2) $[c\,!\,e, \sigma] \rightarrow [\mathrm{E}, (c, \alpha)]$, where $\alpha = \mathcal{V}(e)(\sigma)$

(3) $[c\,?\,v, \sigma] \rightarrow [\mathrm{E}, (c, v)]$

(4) $[s_1 + s_2, \sigma] \rightarrow_0 [s_1, \sigma]$

(5) $[s_1 + s_2, \sigma] \rightarrow_0 [s_2, \sigma]$

(6) $[\mu x\,[s], \sigma] \rightarrow [s\{\mu x\,[s]/x\}, \sigma]$

(7) if $[s_1, \sigma] \rightarrow [r_1, \eta]$ then $[s_1 ; s_2, \sigma] \rightarrow [r_1 ; s_2, \eta]$

(8) if $[s_1, \sigma] \rightarrow [r_1, \eta]$ then $[s_1 \parallel s_2, \sigma] \rightarrow [r_1 \parallel s_2, \eta]$

(9) if $[s_2, \sigma] \rightarrow [r_2, \eta]$ then $[s_1 \parallel s_2, \sigma] \rightarrow [s_1 \parallel r_2, \eta]$

(10) if $[s_1, \sigma] \rightarrow [r_1, (c, \alpha)]$ and $[s_2, \sigma] \rightarrow [r_2, (c, v)]$
then $[s_1 \parallel s_2, \sigma] \rightarrow [r_1 \parallel r_2, \sigma\{\alpha/v\}]$

(11) if $[s_1, \sigma] \rightarrow [r_1, (c, v)]$ and $[s_2, \sigma] \rightarrow [r_2, (c, \alpha)]$
then $[s_1 \parallel s_2, \sigma] \rightarrow [r_1 \parallel r_2, \sigma\{\alpha/v\}]$

In the above, we adopt the convention that $\mathrm{E} ; s = \mathrm{E} \parallel s = s \parallel \mathrm{E} = s$, and $\mathrm{E} \parallel \mathrm{E} = \mathrm{E}$. We say that $[s, \sigma]$ *blocks* if there do not exist a resumption r and a state (not an extended state) σ' such that $[s, \sigma] \rightarrow [r, \sigma']$. The semantic domain for the operational semantics is introduced in

DEFINITION 3.4 The domain P_4 is defined by

$$(p \in)\, P_4 = State_3 \rightarrow \mathcal{P}_{nc}((State_3)_\delta^\infty).$$

The set $(\varsigma \in)\,(State_3)_\delta^\infty = State_3^* \cup State_3^\omega \cup State_3^* \cdot \{\delta\}$ of finite and infinite sequences of states possibly ending with δ is endowed with the ultrametric described after Definition 1.4.

DEFINITION 3.5 The operational semantics $\mathcal{O}^* : Res_3 \rightarrow P_4$ is the unique mapping satisfying

$$\mathcal{O}^*(\mathrm{E}) = \lambda\sigma\,.\,\{\varepsilon\}$$
$$\mathcal{O}^*(s) = \lambda\sigma\,.\,\begin{cases}\{\delta\} & \text{if } [s,\sigma] \text{ blocks}\\ \bigcup\{\,\sigma'\cdot\mathcal{O}^*(r)(\sigma') \mid [s,\sigma]\to[r,\sigma']\,\} & \text{otherwise}\end{cases}$$

The operational semantics $\mathcal{O}$ is defined as the restriction of $\mathcal{O}^*$ to $\mathcal{L}_{co}$. It is important to observe that $\mathcal{O}^*$, and hence $\mathcal{O}$, is not compositional, i.e. there is no semantic operator $\|$ satisfying $\mathcal{O}^*(s_1 \,\|\, s_2) = \mathcal{O}^*(s_1) \,\|\, \mathcal{O}^*(s_2)$.

The semantic domain for the denotational semantics is presented in

DEFINITION 3.6 The domain P_5 is defined by

$$(p \in)\, P_5 \cong \{\mathrm{E}\} \cup (State_3 \to \mathcal{P}_{co}(State_3^{ext} \times id_{\frac{1}{2}}(P_5))).$$

In the above definition, $\mathcal{P}_{co}$ denotes the compact power set operator. The domain P_5 is a branching domain. Its core structure is as that of a P_5' solving $P_5' \cong \mathcal{P}_{co}(State_3^{ext} \times id_{\frac{1}{2}}(P_5'))$; additional structure is provided by the nil process E and by P_5's functional dependence on arguments in $State_3$. It is not difficult to define (a natural extension of) bisimilarity (notation $\sim$) on P_5, and to show that P_5 is strongly extensional, viz. $p_1 \sim p_2$ if and only if $p_1 = p_2$ (cf. [RT92, Bre93]).

DEFINITION 3.7 The operator $; : P_5 \times P_5 \to^1 P_5$ is the unique mapping satisfying

$$p_1 ; p_2 = \begin{cases} p_2 & \text{if } p_1 = \mathrm{E}\\ \lambda\sigma\,.\,\{\,(\eta, p_1' ; p_2) \mid (\eta, p_1') \in p_1(\sigma)\,\} & \text{otherwise}\end{cases}$$

The operator $+ : P_5 \times P_5 \to^1 P_5$ is defined by

$$p_1 + p_2 = \begin{cases} p_2 & \text{if } p_1 = \mathrm{E}\\ p_1 & \text{if } p_2 = \mathrm{E}\\ \lambda\sigma\,.\,(p_1(\sigma) \cup p_2(\sigma)) & \text{otherwise}\end{cases}$$

The operator $\| : P_5 \times P_5 \to^1 P_5$ is the unique mapping satisfying

$$p_1 \,\|\, p_2 = (p_1 \mathbin{\lfloor\!\lfloor} p_2) + (p_2 \mathbin{\lfloor\!\lfloor} p_1) + (p_1 \mathbin{\lfloor} p_2) + (p_2 \mathbin{\lfloor} p_1)$$

where

$$p_1 \mathbin{\lfloor\!\lfloor} p_2 = \begin{cases} p_2 & \text{if } p_1 = \mathrm{E}\\ \lambda\sigma\,.\,\{\,(\eta, p_1' \,\|\, p_2) \mid (\eta, p_1') \in p_1(\sigma)\,\} & \text{otherwise}\end{cases}$$

and, for $p_1 = \mathrm{E}$ or $p_2 = \mathrm{E}$,

$$p_1 \mathbin{\lfloor} p_2 = \mathrm{E},$$

otherwise

$$p_1 \mathbin{\lfloor} p_2 = \lambda\sigma\,.\,\{\,(\sigma\{\alpha/v\}, p_1' \,\|\, p_2') \mid ((c,\alpha), p_1') \in p_1(\sigma),\, ((c,v), p_2') \in p_2(\sigma)\,\}.$$

The above definition can be made rigorous by another appeal to higher order techniques. For example, for the operator ; we should introduce a higher order mapping $\Phi_; : (P_5 \times P_5 \to^1 P_5) \to (P_5 \times P_5 \to^1 P_5)$ defined by

$$\Phi_{;}(\phi)(p_1,p_2) = \begin{cases} p_2 & \text{if } p_1 = \mathrm{E} \\ \lambda\sigma\,.\,\{\,(\eta,\phi\,(p_1',p_2) \mid (\eta,p_1') \in p_1\,(\sigma)\,\} & \text{otherwise} \end{cases}$$

DEFINITION 3.8 The denotational semantics $\mathcal{D} : \mathcal{L}_{co} \to P_5$ is the unique mapping satisfying

$$\begin{aligned}
\mathcal{D}\,(v := e) &= \lambda\sigma\,.\,\{(\sigma\{\alpha/v\},\mathrm{E})\}, \text{ where } \alpha = \mathcal{V}\,(e)(\sigma) \\
\mathcal{D}\,(c\,!\,e) &= \lambda\sigma\,.\,\{((c,\alpha),\mathrm{E})\}, \text{ where } \alpha = \mathcal{V}\,(e)(\sigma) \\
\mathcal{D}\,(c\,?\,v) &= \lambda\sigma\,.\,\{((c,v),\mathrm{E})\} \\
\mathcal{D}\,(s_1\,;\,s_2) &= \mathcal{D}\,(s_1)\,;\,\mathcal{D}\,(s_2) \\
\mathcal{D}\,(s_1 + s_2) &= \mathcal{D}\,(s_1) + \mathcal{D}\,(s_2) \\
\mathcal{D}\,(s_1 \parallel s_2) &= \mathcal{D}\,(s_1) \parallel \mathcal{D}\,(s_2) \\
\mathcal{D}\,(\mu x\,[s]) &= \lambda\sigma\,.\,\{(\sigma,\mathcal{D}\,(s\{\mu x\,[s]/x\}))\}
\end{aligned}$$

We now prepare the way for the statement relating $\mathcal{O}$ and $\mathcal{D}$. We first define a 'hybrid' operational semantics, based on $\mathcal{T}_3$ but yielding elements in the denotational domain P_5.

DEFINITION 3.9 The operational semantics $\mathcal{O}^{\#} : Res_3 \to P_5$ is the unique mapping satisfying

$$\begin{aligned}
&\mathcal{O}^{\#}\,(\mathrm{E}) = \mathrm{E} \\
&\mathcal{O}^{\#}\,(s) = \lambda\sigma\,.\,\{\,(\eta,\mathcal{O}^{\#}\,(r)) \mid [s,\,\sigma] \to [r,\,\eta]\,\}
\end{aligned}$$

Second, we extend the denotational semantics $\mathcal{D}$ to a denotational semantics $\mathcal{D}^{\#}$ from Res_3 to P_5 by defining $\mathcal{D}^{\#}\,(\mathrm{E}) = \mathrm{E}$.

LEMMA 3.10 $\mathcal{O}^{\#} = \mathcal{D}^{\#}$.

PROOF Following the first proof of Theorem 1.9, it suffices to show that the higher order mapping $\Phi_{\mathcal{O}^{\#}}$ underlying Definition 3.9 has $\mathcal{D}^{\#}$ as fixed point.

□

Finally, we show how the operational semantics $\mathcal{O}^{\#}$ and $\mathcal{O}^{*}$ are connected. Semantic domain $(p_4 \in)\,P_4$ is simpler than $(p_5 \in)\,P_5$ in three ways;

- for all σ, the branching structure of $p_5\,(\sigma)$ is collapsed, leaving in $p_4\,(\sigma)$ only a set of paths of $p_5\,(\sigma)$,
- failing attempts at communication (c,α) or (c,v) appear in $p_5\,(\sigma)$ but not in $p_4\,(\sigma)$, and
- $p_5\,(\sigma)$ contains, in general, pairs (σ',p_5'). Here p_5' models the continuation of the execution after σ' has been delivered. This allows that an interleaving action of some $\bar{p}_5$ might change σ' before p_5' is applied. However, this does not hold for $p_4\,(\sigma)$ which contains sets of the form $\sigma' \cdot p_4'\,(\sigma')$.

The combined effect of these simplifications is yielded by *trace* defined in

DEFINITION 3.11 The mapping $trace : P_5 \to^1 P_4$ is the unique mapping satisfying

$$\begin{aligned}
&trace\,(\mathrm{E}) = \lambda\sigma\,.\,\{\varepsilon\} \\
&trace\,(p) = \lambda\sigma\,.\begin{cases} \{\delta\} & \text{if } p\,(\sigma) \text{ blocks} \\ \bigcup\{\,\sigma' \cdot trace\,(p')(\sigma') \mid (\sigma',p') \in p\,(\sigma)\,\} & \text{otherwise} \end{cases}
\end{aligned}$$

where $p(\sigma)$ blocks if there does not exist a pair (σ', p') in $p(\sigma)$.

The well-definedness proof of the higher order mapping Φ_{trace} underlying the above definition relies on Michael's theorem[6].

LEMMA 3.12 $\mathcal{O}^* = trace \circ \mathcal{O}^{\#}$.

PROOF Again we can follow the first proof of Theorem 1.9 by showing that the higher order mapping $\Phi_{\mathcal{O}^*}$ underlying Definition 3.5 has $trace \circ \mathcal{O}^{\#}$ as fixed point.

□

THEOREM 3.13 $\mathcal{O} = trace \circ \mathcal{D}$.

4. A PARALLEL LANGUAGE WITH SECOND ORDER COMMUNICATION

This is the culminating section of our paper, providing a synthesis of ideas from the Sections 2 and 3. In addition, we need some novel techniques to establish the relationship between $\mathcal{O}$ and $\mathcal{D}$ for $\mathcal{L}_{co_2}$. In particular, we use

- the 'processes as terms' approach of [Rut92], and
- a metric on configurations of a transition system ([Bre94]).

As in Section 2, a more realistic language could be based on local states. In such a setting it would be meaningful to transmit a *closure*, a pair consisting of a statement and a local state, rather than just a statement (as we do in the operational model for $\mathcal{L}_{co_2}$).

DEFINITION 4.1 The language $\mathcal{L}_{co_2}$ is defined by

$$s ::= v := e \mid s\,;\, s \mid s + s \mid s \parallel s \mid x \mid c\,!\,s \mid c\,?\,x.$$

The configurations of the transition system are pairs of resumptions (defined as in the previous section, but now named Res_4) and extended syntactic states.

DEFINITION 4.2 The set $SynState_4$ of syntactic states is defined by

$$(\sigma \in)\, SynState_4 = (IVar \to Val) \times (PVar \to \mathcal{L}_{co_2}).$$

The class $SynState_4^{ext}$ of extended syntactic states is defined by

$$(\eta \in)\, SynState_4^{ext} = SynState_4 \cup (Chan \times \mathcal{L}_{co_2}) \cup (\overline{Chan} \times PVar),$$

where $\overline{Chan} = \{\, \bar{c} \mid c \in Chan \,\}$.

We introduce $\overline{Chan}$ to avoid a possible ambiguity: we distinguish between the extended state denoting that statement x is sent on channel c - denoted by (c, x) -, and the extended state denoting that the statement received on channel c should be assigned to procedure variable x - denoted by $(\bar{c}, x)$. The transition system $\mathcal{T}_4$ is presented in

[6] Let (X, d_X) be a metric space. If $\mathcal{X} \in \mathcal{P}_{co}(\mathcal{P}_{co}(X))$ then $\bigcup \mathcal{X} \in \mathcal{P}_{co}(X)$ (cf. [Mic51]).

DEFINITION 4.3 The transition relation $\rightarrow$ of $\mathcal{T}_4$ is the smallest subset of

$$(Res_4 \times SynState_4^{ext}) \times (Res_4 \times SynState_4^{ext})$$

satisfying (1), (4), (5), (7), (8), (9) from Definition 3.3, and

(12) $[x, \sigma] \rightarrow [\sigma(x), \sigma]$

(13) $[c\,!\,s, \sigma] \rightarrow [\mathrm{E}, (c, s)]$

(14) $[c\,?\,x, \sigma] \rightarrow [\mathrm{E}, (\bar{c}, x)]$

(15) if $[s_1, \sigma] \rightarrow [r_1, (c, s)]$ and $[s_2, \sigma] \rightarrow [r_2, (\bar{c}, x)]$
then $[s_1 \parallel s_2, \sigma] \rightarrow [r_1 \parallel r_2, \sigma\{s/x\}]$

(16) if $[s_1, \sigma] \rightarrow [r_1, (\bar{c}, x)]$ and $[s_2, \sigma] \rightarrow [r_2, (c, s)]$
then $[s_1 \parallel s_2, \sigma] \rightarrow [r_1 \parallel r_2, \sigma\{s/x\}]$

The definitions of $\mathcal{O}^*$ and $\mathcal{O}$ follow those of $\mathcal{O}^*$ and $\mathcal{O}$ of the previous section, but now using transition system $\mathcal{T}_4$ and semantic domain P_6 introduced in

DEFINITION 4.4 The domain P_6 is defined by

$$(p \in) P_6 = SynState_4 \rightarrow \mathcal{P}_{nc}((SynState_4)_\delta^\infty).$$

Next, we define the collection of (extended) semantic states $SemState_4$ ($SemState_4^{ext}$), and the domain P_7 of denotational meanings for $\mathcal{L}_{co_2}$.

DEFINITION 4.5 The domains $SemState_4$, $SemState_4^{ext}$, and P_7 are defined by

$$\begin{aligned}
(\rho \in) SemState_4 &\cong (IVar \rightarrow Val) \times (PVar \rightarrow id_{\frac{1}{2}}(P_7)) \\
(\xi \in) SemState_4^{ext} &\cong SemState_4 \,\bar{\cup}\, (Chan \times id_{\frac{1}{2}}(P_7)) \,\bar{\cup}\, (\overline{Chan} \times PVar) \\
(p \in) P_7 &\cong \{\mathrm{E}\} \,\bar{\cup}\, (SemState_4 \rightarrow^1 \mathcal{P}_{co}(SemState_4^{ext} \times id_{\frac{1}{2}}(P_7)))
\end{aligned}$$

Note the correspondence of the definitions of the domains $SemState_4$, $SemState_4^{ext}$, and P_7 with those of $SynState_4$, $SynState_4^{ext}$, and P_6, respectively. On domain P_7 we can define (higher order) bisimilarity in several ways. Based on these definitions, the domain can be shown to be strongly extensional. Whether one of the bisimilarity notions gives us the 'right' equivalence needs further study.

DEFINITION 4.6 The denotational semantics $\mathcal{D} : \mathcal{L}_{co_2} \rightarrow P_7$ is defined by

$$\begin{aligned}
\mathcal{D}(v := e) &= \lambda\rho\,.\,\{(\rho\{\alpha/v\}, \mathrm{E})\}, \text{where } \alpha = \mathcal{V}(e)(\rho) \\
\mathcal{D}(s_1\,;\,s_2) &= \mathcal{D}(s_1)\,;\,\mathcal{D}(s_2) \\
\mathcal{D}(s_1 + s_2) &= \mathcal{D}(s_1) + \mathcal{D}(s_2) \\
\mathcal{D}(s_1 \parallel s_2) &= \mathcal{D}(s_1) \parallel \mathcal{D}(s_2) \\
\mathcal{D}(x) &= \lambda\rho\,.\,\{(\rho, \rho(x))\} \\
\mathcal{D}(c\,!\,s) &= \lambda\rho\,.\,\{((c, p), \mathrm{E})\}, \text{ where } p = \mathcal{D}(s) \\
\mathcal{D}(c\,?\,x) &= \lambda\rho\,.\,\{((\bar{c}, x), \mathrm{E})\}
\end{aligned}$$

The semantic operators used here are defined quite similarly to those of Definition 3.7. For example, for the operator $\lfloor$ we have, for $p_1 \neq \mathrm{E}$ and $p_2 \neq \mathrm{E}$,

$$p_1 \lfloor p_2 = \lambda\rho \,.\, \{\, (\rho\{p/x\}, p_1' \parallel p_2') \mid ((c,p), p_1') \in p_1\,(\rho), ((\bar{c},x), p_2') \in p_2\,(\rho)\,\}.$$

In order to relate $\mathcal{O}$ and $\mathcal{D}$, we need various preparations. First, we want to mimic the introduction of $\mathcal{O}^{\#}$ (cf. Definition 3.9), delivering denotational meanings. This requires using ρ's rather than σ's. Clause (12) of Definition 4.3 then obtains the form $[x,\, \rho] \rightarrow [\rho(x),\, \rho]$. As a consequence, semantic entities $p \in P_7$ appear in the new $\mathcal{T}_4'$, with respect to the extended class of resumptions introduced in Definition 4.7. In Definition 4.8, we introduce the induced transition system. Note that $\mathcal{T}_4'$ is no more finitely branching, and the higher order definition of $\mathcal{O}^{\#}$ based on $\mathcal{T}_4'$ requires separate justification.

DEFINITION 4.7 The class Res_4' is defined by

$$u ::= \mathrm{E} \mid t$$

where

$$t ::= v := e \mid t\,;t \mid t + t \mid t \parallel t \mid x \mid c\,!\,t \mid c\,?\,x \mid p.$$

DEFINITION 4.8 The transition relation $\rightarrow$ of $\mathcal{T}_4'$ is the smallest subset of

$$(Res_4' \times SemState_4^{ext}) \times (Res_4' \times SemState_4^{ext})$$

satisfying

(1) $[v := e,\, \rho] \;\rightarrow\; [\mathrm{E},\, \rho\{\alpha/v\}]$, where $\alpha = \mathcal{V}(e)(\rho)$

(2) $[t_1 + t_2,\, \rho] \rightarrow_0 [t_1,\, \rho]$

(3) $[t_1 + t_2,\, \rho] \rightarrow_0 [t_2,\, \rho]$

(4) $[x,\, \rho] \;\rightarrow\; [\rho(x),\, \rho]$

(5) $[c\,!\,t,\, \rho] \;\rightarrow\; [\mathrm{E},\, (c,p)]$, where $p = \mathcal{D}^{\#}(t)$ (cf. Definition 4.12)

(6) $[c\,?\,x,\, \rho] \;\rightarrow\; [\mathrm{E},\, (\bar{c},x)]$

(7) if $[t_1,\, \rho] \rightarrow [u_1,\, \xi]$ then $[t_1\,;t_2,\, \rho] \rightarrow [u_1\,;t_2,\, \xi]$

(8) if $[t_1,\, \rho] \rightarrow [u_1,\, \xi]$ then $[t_1 \parallel t_2,\, \rho] \rightarrow [u_1 \parallel t_2,\, \xi]$

(9) if $[t_2,\, \rho] \rightarrow [u_2,\, \xi]$ then $[t_1 \parallel t_2,\, \rho] \rightarrow [t_1 \parallel u_2,\, \xi]$

(10) if $[t_1,\, \rho] \rightarrow [u_1,\, (c,p)]$ and $[t_2,\, \rho] \rightarrow [u_2,\, (\bar{c},x)]$
then $[t_1 \parallel t_2,\, \rho] \rightarrow [u_1 \parallel u_2,\, \rho\{p/x\}]$

(11) if $[t_1,\, \rho] \rightarrow [u_1,\, (\bar{c},x)]$ and $[t_2,\, \rho] \rightarrow [u_2,\, (c,p)]$
then $[t_1 \parallel t_2,\, \rho] \rightarrow [u_1 \parallel u_2,\, \rho\{p/x\}]$

(12) if $(\xi, p') \in p\,(\rho)$ then $[p,\, \rho] \rightarrow [p',\, \xi]$

DEFINITION 4.9 The operational semantics $\mathcal{O}^{\#} : Res_4' \rightarrow^1 P_7$ is the unique mapping satisfying

$$\mathcal{O}^{\#}(\mathrm{E}) = \mathrm{E}$$
$$\mathcal{O}^{\#}(t) \;= \lambda\rho \,.\, \{\, (\xi, \mathcal{O}^{\#}(u)) \mid [t,\, \rho] \rightarrow [u,\, \xi]\,\}$$

Note that the $\rightarrow^1$ in the above definition assumes a metric on Res_4'. This is presented in

DEFINITION 4.10 The metric $d : Res_4' \times Res_4' \rightarrow [0,1]$ is defined by

$$\begin{array}{ll} d(u,u) & = 0 \\ d(p,p') & = d_{P_7}(p,p') \\ d(t_1 ; t_2, t_1' ; t_2') & = \max\{d(t_1,t_1'), d(t_2,t_2')\} \\ d(t_1 + t_2, t_1' + t_2') & = \max\{d(t_1,t_1'), d(t_2,t_2')\} \\ d(t_1 \parallel t_2, t_1' \parallel t_2') & = \max\{d(t_1,t_1'), d(t_2,t_2')\} \\ d(c\,!\,t, c\,!\,t') & = d(t,t') \\ d(t,t') & = 1, \text{ otherwise} \end{array}$$

We shall also need the mapping $\mathcal{S}$ defined in

DEFINITION 4.11 The mapping $\mathcal{S} : (Res_4' \times SemState_4) \rightarrow^1 \mathcal{P}_{co}(Res_4' \times SemState_4^{ext})$ is defined by

$$\mathcal{S}(u,\rho) = \{\, [u',\, \xi] \mid [u,\, \rho] \rightarrow [u',\, \xi] \,\}.$$

Let $\Phi_{\mathcal{O}^\#}$ be the higher order mapping associated in the natural way with the definition of $\mathcal{O}^\#$. Well-definedness of $\Phi_{\mathcal{O}^\#}$ follows by noting that

- $\mathcal{S}$ is well-defined, i.e., for all u and ρ, $\mathcal{S}(u,\rho)$ is compact and $\mathcal{S}$ is nonexpansive,
- for all t and ρ, the set $\{\, (\xi, \phi(u)) \mid [t,\, \rho] \rightarrow [u,\, \xi] \,\}$ is compact, since $\mathcal{S}$ delivers compact sets and ϕ is nonexpansive,
- for all t, the mapping $\lambda\rho \,.\, \{\, (\xi, \phi(u)) \mid [t,\, \rho] \rightarrow [u,\, \xi] \,\}$ is nonexpansive, since $\mathcal{S}$ and ϕ are nonexpansive.

Second, we extend the denotational semantics $\mathcal{D}$.

DEFINITION 4.12 The denotational semantics $\mathcal{D}^\# : Res_4' \rightarrow^1 P_7$ is defined by

$$\begin{array}{ll} \mathcal{D}^\#(\mathrm{E}) & = \mathrm{E} \\ \mathcal{D}^\#(v := e) & = \lambda\rho \,.\, \{(\rho\{\alpha/v\}, \mathrm{E})\}, \text{ where } \alpha = \mathcal{V}(e)(\rho) \\ \mathcal{D}^\#(t_1 ; t_2) & = \mathcal{D}^\#(t_1) ; \mathcal{D}^\#(t_2) \\ \mathcal{D}^\#(t_1 + t_2) & = \mathcal{D}^\#(t_1) + \mathcal{D}^\#(t_2) \\ \mathcal{D}^\#(t_1 \parallel t_2) & = \mathcal{D}^\#(t_1) \parallel \mathcal{D}^\#(t_2) \\ \mathcal{D}^\#(x) & = \lambda\rho \,.\, \{(\rho, \rho(x))\} \\ \mathcal{D}^\#(c\,!\,t) & = \lambda\rho \,.\, \{((c,p), \mathrm{E})\}, \text{ where } p = \mathcal{D}^\#(t) \\ \mathcal{D}^\#(c\,?\,x) & = \lambda\rho \,.\, \{((\bar{c},x), \mathrm{E})\} \\ \mathcal{D}^\#(p) & = p \end{array}$$

LEMMA 4.13 $\mathcal{O}^\# = \mathcal{D}^\#$.

PROOF This proof follows the first proof of Theorem 1.9. For example, for resumption x we have that

$$\begin{aligned}
&\Phi_{\mathcal{O}^\#}(\mathcal{D}^\#)(x)\\
&\quad = \lambda\rho\,.\,\{(\rho, \mathcal{D}^\#(\rho(x)))\}\\
&\quad = \lambda\rho\,.\,\{(\rho, \rho(x))\}\\
&\quad = \mathcal{D}^\#(x).
\end{aligned}$$

□

To each extended syntactic state an extended semantic state is assigned by the mapping *sem*.

DEFINITION 4.14 The mapping $sem : SynState_4^{ext} \to SemState_4^{ext}$ is defined by

$$\begin{aligned}
sem(\sigma) &= (\sigma_1,\ \lambda x\,.\,\mathcal{D}^\#(\sigma_2(x)))\\
sem((\bar{c}, x)) &= (\bar{c}, x)\\
sem((c, s)) &= (c, \mathcal{D}^\#(s))
\end{aligned}$$

The mapping *sem* is, again, extended in the natural way to a mapping from $\mathcal{P}_{nc}((SynState_4)_\delta^\infty)$ to $\mathcal{P}_{nc}((SemState_4)_\delta^\infty)$. The next lemma is the key technical result on which the relationship between $\mathcal{O}$ and $\mathcal{D}$ is based. The lemma expresses a canonical correspondence between transitions of $\mathcal{T}_4$ and $\mathcal{T}_4'$.

LEMMA 4.15 *For all* s, r, u, σ, σ', *and* ξ,

$$\begin{aligned}
&\textit{if } [s,\ \sigma] \to [r,\ \sigma'] \textit{ then } [s,\ sem(\sigma)] \to [u',\ sem(\sigma')]\\
&\textit{and } \mathcal{O}^\#(u') = \mathcal{O}^\#(r) \textit{ for some } u'
\end{aligned}$$

and

$$\begin{aligned}
&\textit{if } [s,\ sem(\sigma)] \to [u,\ \xi] \textit{ then } [s,\ \sigma] \to [r',\ \sigma'']\\
&\textit{and } \mathcal{O}^\#(r') = \mathcal{O}^\#(u) \textit{ and } sem(\sigma'') = \xi \textit{ for some } r' \textit{ and } \sigma''.
\end{aligned}$$

PROOF This lemma can be proved by structural induction on s. We will only consider the first part for statement $s_1\,;\,s_2$. We distinguish two cases.

1. Assume $[s_1\,;\,s_2,\ \sigma] \to [s_2,\ \sigma']$. Then $[s_1,\ \sigma] \to [\text{E},\ \sigma']$. By induction, $[s_1,\ sem(\sigma)] \to [u',\ sem(\sigma')]$ and $\mathcal{O}^\#(u') = \mathcal{O}^\#(\text{E})$. Consequently, $u' \equiv \text{E}$. So, $[s_1\ ;\ s_2,\ sem(\sigma)] \to [s_2,\ sem(\sigma')]$.
2. Assume $[s_1\ ;\ s_2, \sigma] \to [s_1'\ ;\ s_2, \sigma']$. Then $[s_1, \sigma] \to [s_1', \sigma']$. By induction, $[s_1,\ sem(\sigma)] \to [u',\ sem(\sigma')]$ and $\mathcal{O}^\#(u') = \mathcal{O}^\#(s_1')$. Consequently, $u' \not\equiv \text{E}$. So, $[s_1\ ;\ s_2,\ sem(\sigma)] \to [u'\ ;\ s_2,\ sem(\sigma')]$ and
$$\begin{aligned}
&\mathcal{O}^\#(u'\ ;\ s_2)\\
&\quad = \mathcal{D}^\#(u'\ ;\ s_2)\\
&\quad = \mathcal{O}^\#(u')\,;\,\mathcal{D}^\#(s_2)\\
&\quad = \mathcal{O}^\#(s_1')\,;\,\mathcal{D}^\#(s_2)\\
&\quad = \mathcal{O}^\#(s_1'\ ;\ s_2).
\end{aligned}$$

□

The mapping *trace* used for $\mathcal{L}_{co_2}$ is obtained from Definition 3.11 by replacing σ's by ρ's:

DEFINITION 4.16 The mapping $trace : P_7 \to^1 SemState_4 \to^1 \mathcal{P}_{nc}((SemState_4)^\infty_\delta)$ is defined by

$$trace(\mathrm{E}) = \lambda\rho\,.\,\{\varepsilon\}$$
$$trace(p) = \lambda\rho\,.\,\begin{cases}\{\delta\} & \text{if } p(\rho) \text{ blocks}\\ \bigcup\{\,\rho' \cdot trace(p')(\rho') \mid (\rho', p') \in p(\rho)\,\} & \text{otherwise}\end{cases}$$

The operational semantics $\mathcal{O}^*$ and $\mathcal{O}^\#$ are related by means of the mappings *sem* and *trace*.

LEMMA 4.17 *For all r and σ, $sem(\mathcal{O}^*(r)(\sigma)) = trace(\mathcal{O}^\#(r))(sem(\sigma))$.*

PROOF We can prove this lemma by means of the proof principle exploited in the second proof of Theorem 1.9 using Lemma 4.15.

□

THEOREM 4.18 *For all s and σ, $sem(\mathcal{O}(s)(\sigma)) = trace(\mathcal{D}(s))(sem(\sigma))$.*

SUMMARY

The results from the Sections 1 to 4 relating $\mathcal{O}$ and $\mathcal{D}$ for the four languages considered are summarised in the following table (putting $\mathcal{O}[\![s]\!] = \mathcal{O}(s)$ for each of the four languages, $\mathcal{D}[\![s]\!] = \mathcal{D}(s)(\lambda\sigma\cdot\{\varepsilon\})$ for $\mathcal{L}_{as}$, $\mathcal{D}[\![s]\!] = \mathcal{D}(s)(\lambda\rho\cdot\{\varepsilon\})$ for $\mathcal{L}_{as_2}$, and $\mathcal{D}[\![s]\!] = \mathcal{D}(s)$ for $\mathcal{L}_{co}$ and $\mathcal{L}_{co_2}$):

$$\begin{array}{lll}
\mathcal{L}_{as} & : & \mathcal{O}[\![s]\!] = \mathcal{D}[\![s]\!]\\
\mathcal{L}_{as_2} & : & sem \circ \mathcal{O}[\![s]\!] = \mathcal{D}[\![s]\!] \circ sem\\
\mathcal{L}_{co} & : & \mathcal{O}[\![s]\!] = (trace \circ \mathcal{D})[\![s]\!]\\
\mathcal{L}_{co_2} & : & sem \circ \mathcal{O}[\![s]\!] = (trace \circ \mathcal{D})[\![s]\!] \circ sem
\end{array}$$

REFERENCES

[ABKR89] P. America, J.W. de Bakker, J.N. Kok, and J.J.M.M. Rutten. Denotational Semantics of a Parallel Object-Oriented Language. *Information and Computation*, 83(2):152–205, November 1989.

[ACCL90] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. In *Proceedings of the 17th Annual ACM Symposium on Principles of Programming Languages*, pages 31–46, San Francisco, January 1990.

[AGR92] E. Astesiano, A. Giovini, and G. Reggio. Observational Structures and their Logics. *Theoretical Computer Science*, 96(1):249–283, April 1992.

[AR87] E. Astesiano and G. Reggio. SMoLCS-driven Concurrent Calculi. In H. Ehrig, R. Kowalski, G. Levi, and U. Montanari, editors, *Proceedings of the International Joint Conference on Theory and Practice of Software Development*, volume 249 of *Lecture Notes in Computer Science*, pages 169–201, Pisa, March 1987. Springer-Verlag.

[AR89] P. America and J.J.M.M. Rutten. Solving Reflexive Domain Equations in a Category of Complete Metric Spaces. *Journal of Computer and System Sciences*, 39(3):343–375, December 1989.

[Ban22] S. Banach. Sur les Opérations dans les Ensembles Abstraits et leurs Applications aux Equations Intégrales. *Fundamenta Mathematicae*, 3:133–181, 1922.

[Bar92] H.P. Barendregt. Lambda Calculi with Types. In S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, Background: Computational Structures, chapter 2, pages 117–309. Clarendon Press, Oxford, 1992.

[BB92] G. Berry and G. Boudol. The Chemical Abstract Machine. *Theoretical Computer Science*, 96(1):217–248, April 1992.

[BM88] J.W. de Bakker and J.-J.Ch. Meyer. Metric Semantics for Concurrency. *BIT*, 28:504–529, 1988.

[Bou89] G. Boudol. Towards a Lambda-Calculus for Concurrent and Communicating Systems. In J. Diaz and F. Orejas, editors, *Proceedings of the International Joint Conference on Theory and Practice of Software Development*, volume 351 of *Lecture Notes in Computer Science*, pages 149–162, Barcelona, March 1989. Springer-Verlag.

[BR92] J.W. de Bakker and J.J.M.M. Rutten, editors. *Ten Years of Concurrency Semantics, selected papers of the Amsterdam Concurrency Group*. World Scientific, Singapore, 1992.

[Bre93] F. van Breugel. Three Metric Domains of Processes for Bisimulation. Report, CWI, Amsterdam, June 1993. To appear in *Proceedings of the 9th Conference on the Mathematical Foundations of Programming Semantics*, *Lecture Notes in Computer Science*, New Orleans, April 1993. Springer-Verlag.

[Bre94] F. van Breugel. *Topological Models in Comparative Semantics.* PhD thesis, Vrije Universiteit, Amsterdam, 1994. In preparation.

[BZ82] J.W. de Bakker and J.I. Zucker. Processes and the Denotational Semantics of Concurrency. *Information and Control*, 54(1/2):70–120, July/August 1982.

[Cur88] P.-L. Curien. The $\lambda\rho$-calculus: An Abstract Framework for Environment Machines. Report, LIENS, Paris, October 1988.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Series in Computer Science. Prentice/Hall International, London, 1985.

[JP90] R. Jagadeesan and P. Panangaden. A Domain-theoretic Model for a Higher-order Process Calculus. In M.S. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 181–194, Coventry, July 1990. Springer-Verlag.

[KR90] J.N. Kok and J.J.M.M. Rutten. Contractions in Comparing Concurrency Semantics. *Theoretical Computer Science*, 76(2/3):179–222, 1990.

[Kur56] K. Kuratowski. Sur une Méthode de Métrisation Complète des Certains Espaces d'Ensembles Compacts. *Fundamenta Mathematicae*, 43:114–138, 1956.

[LTLG92] J.-J. Lévy, B. Thomsen, L. Leth, and A. Giacalone. CONcurrency and Functions: Evaluation and Reduction. *Bulletin of the European Association for Theoretical Computer Science*, 48:88–106, October 1992.

[Mic51] E. Michael. Topologies on Spaces of Subsets. *Transactions of the American Mathematical Society*, 71:152–182, 1951.

[Mil92] R. Milner. Functions as Processes. *Mathematical Structures in Computer Science*, 2(2):119–141, June 1992.

[MPW92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, I and II. *Information and Computation*, 1(100):1–40 and 41–77, September 1992.

[MS92] R. Milner and D. Sangiorgi. Barbed Bisimulation. In W. Kuich, editor, *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695, Vienna, July 1992. Springer-Verlag.

[Plo81] G.D. Plotkin. A Structural Approach to Operational Semantics. Report DAIMI FN-19, Aarhus University, Aarhus, September 1981.

[RT92] J.J.M.M. Rutten and D. Turi. On the Foundations of Final Semantics: non-standard sets, metric spaces, partial orders. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop on Semantics: Foundations and Applications*, volume 666 of *Lecture Notes in Computer Science*, pages 477–530, Beekbergen, June 1992. Springer-Verlag.

[Rut92] J.J.M.M. Rutten. Processes as Terms: Non-Well-Founded Models for Bisimulation. *Mathematical Structures in Computer Science*, 2(3):257–275, September 1992.

[San92] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD thesis, University of Edinburg, Edinburg, 1992.

[San93] D. Sangiorgi. An Investigation into Functions as Processes. To appear in *Proceedings of the 9th Conference on the Mathematical Foundations of Programming Semantics*, *Lecture Notes in Computer Science*, New Orleans, April 1993. Springer-Verlag.

[Tho89] B. Thomsen. A Calculus of Higher Order Communicating Systems. In *Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages*, pages 143–154, Austin, January 1989.

[Tho90] B. Thomsen. *Calculi for Higher Order Communicating Systems.* PhD thesis, Imperial College, London, September 1990.