Automation
Computers
Applied Mathematics
ISSN 1221–437X
Vol. 19 (2010) no. 2
pp. 11–27

Comparative Metric Semantics for Modern Second Order Communication Abstractions

Mircea Ivan and Eneia Nicolae Todoran

Mircea Ivan: Department of Mathematics
Technical University of Cluj-Napoca
Str. G. Baritiu 25, 400027, Cluj-Napoca, Romania

Mircea.Ivan@math.utcluj.ro

Eneia Nicolae Todoran:
Department of Computer Science
Technical University of Cluj-Napoca
Faculty of Automation and Computer Science
Baritiu Street 28, 400027, Cluj-Napoca, Romania

Eneia.Todoran@cs.utcluj.ro

Abstract: We study the semantics of a language L_J^2 that provides second order communication and synchronization on multiple channels in the style introduced in Join calculus. We employ the mathematical methodology of metric semantics in designing and relating a denotational and an operational semantics for L_J^2. The semantic models are designed with continuations.

Key Words: Metric semantics, continuations for concurrency, second order communication, Join synchronization

Received: October 20, 2010

1 Introduction

In a recent paper [18] we presented a semantic study of an imperative language L_J that extends Hoare's CSP model [9] with communication on multiple channels and synchronization based on join patterns in the style introduced in Join calculus [8]. In this paper we consider a language L_J^2 that extends L_J with second order communication: sending and receiving of statements rather than values.

L_J^2 generalizes the traditional model of second order communication - studied, e.g., in [2, 3] and also in our previous work [19] - with a mechanism of synchronization inspired by the Join calculus. L_J^2 provides two primitives for concurrent interaction on multiple channels: $c!s$ and $c_1?x_1\& \cdots \& c_n?x_n$; we call the latter a *join pattern*. A communication in L_J^2 is an interaction between $n+1$ processes, one executing a join pattern $c_1?x_1 \& \cdots \& c_n?x_n$, and n other processes executing corresponding (send) actions $c_1!s_1, \cdots, c_n!s_n$. The statements $s_1, \cdots, s_n$ are transmitted concurrently along the channels $c_1, \cdots, c_n$ (each s_i is transmitted along the corresponding channel c_i) from the processes that execute the actions $c_1!s_1, \cdots, c_n!s_n$ to the process that holds the join pattern $c_1?x_1 \& \cdots \& c_n?x_n$. The latter stores the n received statements in the variables $x_1, \cdots, x_n$ (each s_i is assigned to the corresponding statement variable x_i). The communication only occurs when the $n+1$ processes are all ready for the interaction. The stored statements can be called by $\mathsf{call}(x_i)$ statements. When $n = 1$ the whole interaction behaves like a (classic second order) point-to-point communication.

In this paper we employ the mathematical methodology of metric semantics [4, 1] in designing and relating an operational and a denotational semantics for L_J^2. The semantic models are designed with the "continuation semantics for concurrency" (CSC) technique introduced by us in [22]. CSC is a general tool for designing denotational and operational models of concurrency in interleaving semantics. The central characteristic of the CSC technique is the modeling of continuations as structured configurations of computations (denotations) rather than the straightforward functions to some answer type that are used in the classic technique of continuations [16]. The structure of an CSC continuation is representative for the control concepts of the (concurrent) language under study.

The focus of this study is on the communication primitives of L_J^2. We show that CSC continuations can easily synchronize multiple communication attempts and allow us to combine in a simple and flexible manner traditional concurrent control concepts with second order communication primitives and advanced synchronization mechanisms. In [17] we presented a denotational semantics for a language similar to L_J^2. In this paper we offer both a denotational semantics and an operational semantics for L_J^2. Next, we establish the precise mathematical relation between the two semantic models, following the general methodology advocated in [4]. The first paper that uses metric spaces and continuations for concurrency in designing and relating semantic models of second order communication is [19]. [19] studies traditional communication between two processes that synchronize on a single (second-order) communication channel. The present paper offers a generalization of [19] to second-order communication on multiple channels, in the style introduced in Join calculus.

The Join calculus has recently inspired the design of Join Java [10], Polyphonic C# [5] (concurrent extensions of the mainstream languages Java and C#, respectively) and JoCaml [11] (a concurrent extension of the functional programming language ML). The relevance of our work is given by the presence of higher-order features in JoCaml. Also, second order communication provides a simple form of code mobility. Our work may be seen as a first step toward a formal treatment of object mobility in languages with Join methods like Join Java and Polyphonic C#.

Contribution

As far as we know this is the first paper that presents a comparative semantics study for (a form of) higher-order communication with synchronization on multiple channels in the style introduced in the Join calculus [8].

The CSC technique seems to simplify the semantic treatment of second order communication. We obtain a relatively simple relation between the operational and the denotational semantics of L_J^2 by using only basic techniques of metric semantics. In [2, 3] a comparative metric semantics study of second order communication is presented. The relation obtained there between the denotational and the operational models is more complex than the one reported by us in section 7 of this paper. Also, in [2, 3] more advanced techniques, like *processes as terms* [15] and *metric labelled transition systems* [6] are needed to establish that relation.

Overview

The rest of the paper is organized as follows. The formal syntax of L_J^2 is introduced in section 3. The operational and the denotational semantics are presented in sections 5 and 6 and are related in section 7.

2 Notation and theoretical preliminaries

The notation $(x \in)X$ introduces the set X with typical element x ranging over X. Let $f \in X \to Y$ be a function. The function $(f \mid x \mapsto y) : X \to Y$, is defined (for $x, x' \in X, y \in Y$) by:

$$(f \mid x \mapsto y)(x') = \begin{cases} y & \text{if } x' = x \\ f(x') & \text{if } x' \neq x \end{cases}$$

We write $(f \mid x_1 \mapsto y_1 \mid \cdots \mid x_n \mapsto y_n)$ as an abbreviation for $(\cdots(f \mid x_1 \mapsto y_1)\cdots \mid x_n \mapsto y_n)$. If $f : X \to X$ and $f(x) = x$ we call x a *fixed point* of f. When this fixed point is unique (see theorem 2.1) we write $x = fix(f)$.

The study presented in this paper takes place in the mathematical framework of *1-bounded complete metric spaces*. We assume known the following notions: *metric* (and *ultrametric*) space, *isometry* (distance preserving bijection between metric spaces; we denote it by '$\cong$'), *complete* metric space, and *compact* set. We recall that if (X, d_X), (Y, d_Y) are metric spaces, a function $f{:}X \to Y$ is a *contraction* if $\exists k \in \mathbb{R}, 0 \leq k < 1, \forall x_1, x_2 \in X : d_Y(f(x_1), f(x_2)) \leq k \cdot d_X(x_1, x_2)$. When $k = 1$ the function f is called *non-expansive*. In the sequel we denote the set of all nonexpansive functions from X to Y by $X \xrightarrow{1} Y$. The following theorem is at the core of metric semantics.

Theorem 2.1 *(Banach) Let (X, d_X) be a complete metric space. Each contracting function $f : X \to X$ has a* unique *fixed point.*

Definition 2.2 *Let $(X, d_X), (Y, d_Y)$ be (ultra) metric spaces. On $(x \in)X$, $(f \in)X \to Y$ (the function space), $((x, y) \in)X \times Y$ (the cartesian product), $(u, v \in)X + Y$ (the disjoint union of X and Y) and $(U, V \in)\mathcal{P}(X)$ (the power set of X), one can define the following metrics:*

(a) $d_{\frac{1}{2}\cdot X} : X \times X \rightarrow [0,1], \quad d_{\frac{1}{2}\cdot X}(x_1,x_2) = \frac{1}{2}\cdot d_X(x_1,x_2)$

(b) $d_{X\rightarrow Y} : (X\rightarrow Y)\times(X\rightarrow Y)\rightarrow[0,1], \quad d_{X\rightarrow Y}(f_1,f_2) = sup_{x\in X} d_Y(f_1(x),f_2(x))$

(c) $d_{X\times Y} : (X\times Y)\times(X\times Y)\rightarrow[0,1]$

$$d_{X\times Y}((x_1,y_1),(x_2,y_2)) = max\{d_X(x_1,x_2), d_Y(y_1,y_2)\}$$

(d) $d_{X+Y} : (X+Y)\times(X+Y)\rightarrow[0,1]$

$$d_{X+Y}(u,v) = \ \mathsf{if}\ (u,v\in X)\ \mathsf{then}\ d_X(u,v)\ \mathsf{else}\ \ \mathsf{if}\ (u,v\in Y)\ \mathsf{then}\ d_Y(u,v)\ \mathsf{else}\ 1$$

(e) $d_H : \mathcal{P}(X)\times\mathcal{P}(X)\rightarrow[0,1], \quad d_H(U,V) = max\{sup_{u\in U} d(u,V), sup_{v\in V} d(v,U)\}$ *where* $d(u,W){=}inf_{w\in W} d(u,w)$ *and by convention* $sup\emptyset{=}0$, $inf\emptyset{=}1$ *(*d_H *is the* Hausdorff *distance).*

We use the abbreviations $\mathcal{P}_{nco}(\cdot)$ and $\mathcal{P}_{finite}(\cdot)$ to denote the power sets of *non-empty and compact* and *finite* subsets of '$\cdot$', respectively. Also, we often suppress the metrics part in domain definitions, and write, e.g., $\frac{1}{2}\cdot X$ instead of $(X, d_{\frac{1}{2}\cdot X})$.

Remark 2.3 *Let* $(X,d_X),(Y,d_Y),d_{\frac{1}{2}\cdot X}$, $d_{X\rightarrow Y}$, $d_{X\times Y}$, d_{X+Y} *and* d_H *be as in definition 2.2. In case* d_X, d_Y *are ultrametrics, so are* $d_{\frac{1}{2}\cdot X}, d_{X\rightarrow Y}, d_{X\times Y}, d_{X+Y}$ *and* d_H*. If in addition* $(X,d_X),(Y,d_Y)$ *are complete then* $\frac{1}{2}\cdot X$, $X\rightarrow Y$, $X\overset{1}{\rightarrow}Y$, $X\times Y, X+Y$, *and* $\mathcal{P}_{\mathrm{nco}}(X)$ *(with the metrics defined above) are also complete metric spaces.*

3 Syntax of L_J^2

The syntax of L_J^2 is given in BNF in 5.1. The basic components are a set $(v\in)Var$ of *variables*, a set $(e\in)Exp$ of *expressions*, a set $(c\in)Ch$ of (second order) *communication channels* and a set $(x\in)Svar$ of *statement variables.*

Definition 3.1 *(Syntax of* L_J^2*)*

(a) *(Join patterns)* $\quad j(\in J){::=}c?x \mid j\,\&\,j$

For an L_J^2 *program to be valid the channels* $c_1,\cdots,c_n$ *and the statement variables* $x_1,\cdots,x_n$ *in a join pattern* $j = (c_1?x_1\,\&\,\cdots\,\&\,c_n?x_n)$ *must be pairwise distinct.*

(b) *(Statements)* $\quad s(\in Stat) ::= \mathsf{skip} \mid v := e \mid c!s \mid j \mid \mathsf{call}(x) \mid s;s \mid s+s \mid s \parallel s$

The language L_J^2 provides assignment ($v := e$), recursion, sequential composition ($s;s$), nondeterministic choice ($s+s$), parallel composition ($s \parallel s$) and the communication mechanism (based on the interaction between a join pattern $c_1?x_1\,\&\,\cdots\,\&\,c_n?x_n$ and n corresponding send statements $c_1!s_1,\cdots,c_n!s_n$) that was explained informally in the introduction. We assume that the meaning of expressions is given by a valuation $\mathcal{E}[\![\cdot]\!] : Exp\rightarrow\Sigma\rightarrow Val$, where Val is a set of *values* and $(\sigma\in)\Sigma = Var\rightarrow Val$ is a set of *states*.

4 Continuation structure for L_J^2

In the definition of the denotational semantics we use a complete ultrametric space. The operational semantics is defined by means of a transition relation embedded in a deductive system define in the style of structured operational semantics [14]. Following [4], we use the term *resumption* as an operational counterpart of the term *continuation.* The distinctive characteristic of the CSC technique is the representation of continuations as structured configurations of computations (denotations of statements). Similarly, resumptions are structured configurations of statements. The structure of a CSC continuation (resumption) is representative of the language under study.

As shown in [22] in order to handle the general combination of sequential and parallel composition in L_J^2 we need a tree-like structure with active elements at the leaves. The major issue that gives rise to a tree-like structure is the presence of statements such as $(s_1 \parallel s_2); s_3$. In such a statement the execution of s_3 can only begin after the completion of the parallel execution of both s_1 and s_2. The basic idea is that we place s_3 as an inner node and s_1 and s_2 as leaves of such a tree and we give priority to the leaves. Following [22] we define the domain of continuations and the set of resumptions with the aid of an auxiliary set $(\alpha \in) Id$ of *identifiers* endowed with a partial ordering relation.

Definition 4.1

(a) Let $(\alpha \in) Id = \{1,2\}^*$ *be a set of* identifiers, *equipped with the following partial ordering:* $\alpha \leq \alpha$' *iff* $\alpha' = \alpha \cdot i_1 \cdots i_n$ *for* $i_1, \cdots, i_n \in \{1,2\}, n \geq 0$.

(b) We define a function $max : \mathcal{P}(Id) \to \mathcal{P}(Id)$ *by:*

$$max(A) = \{\alpha \mid \alpha \,\text{is a maximal element of}\,(A, \leq_A)\}$$

where $A \in \mathcal{P}(Id)$ *and* $\leq_A$ *is the restriction of* $\leq$ *to the subset* A *of* Id.

Id is the set of all finite (possibly empty) sequences over $\{1,2\}$. $\alpha \leq \alpha'$ iff α is a prefix of α'. In this paper we use the symbol '$\cdot$' as a concatenation operator over sequences and we use the symbol 'ϵ' to represent the empty sequence. We can represent tree-like structures as suggested below:

$$\begin{tikzcd}
\alpha \cdot 1 \cdot 1 \arrow[dr, dash] & & \alpha \cdot 1 \cdot 2 \arrow[dl, dash] & & \alpha \cdot 2 \cdot 1 \arrow[dr, dash] & & \alpha \cdot 2 \cdot 2 \arrow[dl, dash] \\
& \alpha \cdot 1 \arrow[dr, dash] & & & & \alpha \cdot 2 \arrow[dll, dash] & \\
& & & \alpha & & &
\end{tikzcd}$$

Let $A = \{\alpha, \alpha \cdot 1, \alpha \cdot 2, \alpha \cdot 1 \cdot 1, \alpha \cdot 1 \cdot 2, \alpha \cdot 2 \cdot 1, \alpha \cdot 2 \cdot 2\}$. The maximal elements of $(A, \leq_A)$ are exactly the *leaves* of the tree: $max(A) = \{\alpha \cdot 1 \cdot 1, \alpha \cdot 1 \cdot 2, \alpha \cdot 2 \cdot 1, \alpha \cdot 2 \cdot 2\}$.

Let $(\pi \in) \Pi = \mathcal{P}_{finite}(Id)$ and let $(x \in) \mathbf{X}$ be a metric domain. We use the following notation:

$$\{\!|\mathbf{X}|\!\} \overset{not.}{=} \Pi \times (Id \to \mathbf{X})$$

Let $\alpha \in Id, (\pi, \varpi) \in \{\!|\mathbf{X}|\!\}$ (with $\pi \in \Pi$, $\varpi \in Id \to \mathbf{X}$). We define $id : \{\!|\mathbf{X}|\!\} \to \Pi$, $id(\pi, \varpi) = \pi$. We also use the following abbreviations:

$$\begin{array}{rcll} (\pi,\varpi)(\alpha) & \overset{not.}{=} & \varpi(\alpha) & (\in \mathbf{X}) \\ (\pi,\varpi)\setminus\pi' & \overset{not.}{=} & (\pi\setminus\pi',\varpi) & (\in \{\!|\mathbf{X}|\!\}) \\ ((\pi,\varpi)\mid\alpha\mapsto x) & \overset{not.}{=} & (\pi\cup\{\alpha\},(\varpi\mid\alpha\mapsto x)) & (\in \{\!|\mathbf{X}|\!\}) \end{array}$$

Let $(\pi,\varpi)\in\{\!|\mathbf{X}|\!\}$. We treat (π,ϖ) as a 'function' with finite graph $\{(\alpha,\varpi(\alpha))\mid\alpha\in\pi\}$, thus ignoring the behavior of ϖ for any $\alpha\notin\pi$ (π is the 'domain' of (π,ϖ)). We use this mathematical structure to represent finite partially ordered *bags* (or *multisets*)[1] of computations. The set Id is used to distinguish between multiple occurrences of a computation in such a bag. We can obtain a domain by endowing the sets Id and Π with the discrete metric (which is an ultrametric).[2] $id(\pi,\varpi)$ returns the collection of identifiers for the computations contained in the bag (π,ϖ), $(\pi,\varpi)(\alpha)$ returns the computation with identifier α, $(\pi,\varpi)\setminus\pi$ removes the computations with identifiers in π, and $((\pi,\varpi)\mid\alpha\mapsto x)$ replaces the computation with identifier α.

By a slight abuse we will use the same notations when $(x\in)X$ is an ordinary set (rather than a metric domain): $\{\!|X|\!\}=\Pi\times(Id\to X)$; in this case we do not endow $\{\!|X|\!\}$ with a metric structure. Again, if $(\pi,\varpi)\in\{\!|X|\!\}$ we write: $id(\pi,\varpi)=\pi,(\pi,\varpi)(\alpha)=\varpi(\alpha),(\pi,\varpi)\setminus\pi'=(\pi\setminus\pi',\varpi)$ and $((\pi,\varpi)\mid\alpha\mapsto x)=(\pi\cup\{\alpha\},(\varpi\mid\alpha\mapsto x))$.

5 Operational semantics ($\mathcal{O}$)

In 5.1 we introduce the class of *resumptions* and the *configurations* of the transition system for L_J^2.

Definition 5.1 *We define $(\rho\in)Comp=J^{\diamondsuit}\cup(Ch\times Stat)\cup Stat$, where $J^{\diamondsuit}=\{\diamondsuit\}\times J$ ($J\subseteq Stat$, but $J^{\diamondsuit}\cap Stat=\emptyset$). For any $(\diamondsuit,j)\in J^{\diamondsuit}$ we use the notation $\langle j\rangle=(\diamondsuit,j)$. Also, for easier readability, we denote typical elements (c,s) of $(Ch\times Stat)$ by $c!s$.*

(a) The class $(r\in)Res$ of resumptions *is given by: $Res=\{\!|Comp|\!\}$.*

(b) Let $Res^{\alpha}=Res\times Id$. We denote typical elements (r,α) of Res^{α} by r^{α}. We define the class $(t\in)Conf$ of configurations *by:*

$$Conf=(Stat\times Res^{\alpha}\times\Theta\times\Sigma)\cup(Res\times\Theta\times\Sigma)$$

where $(\theta\in)\Theta$ is the class of syntactic stores

$$\Theta=Svar\to Stat$$

We say that a configuration $t\in Conf$ is derivable *if either $t\in(Res\times\Theta\times\Sigma)$ or $t=(s,r^{\alpha},\theta,\sigma)\in(Stat\times Res^{\alpha}\times\Theta\times\Sigma)$ with $\alpha\notin id(r)$ and $\alpha\in max(\{\alpha\}\cup id(r))$. Let $Conf'$ be the class of* derivable configurations*.*

We introduce three auxiliary mappings that produce values of the type $Sched=\mathcal{P}_{finite}(Id^{+})$, which are finite collections of schedules. A *schedule* ($\in Id^{+}$) is a finite and nonempty sequence of identifiers. We define the mappings $sched, scheda, scheds: Res\to Sched$ as follows:

[1] We avoid using the notion of a *partially ordered multiset* which usually denotes a more elaborated mathematical structure; see, e.g., ch. 16 of [4].

[2] $d(x,y)=0$ if $x=y$. If $x\neq y$ then $d(x,y)=1$.

$$sched(r) = scheda(r) \cup scheds(r)$$

$$scheda(r) = \{(\alpha) \mid \alpha \in max(id(r)), r(\alpha) \in Stat\}$$

$$\begin{aligned} scheds(r) = \{(\alpha, \alpha_1, \cdots, \alpha_n) \mid \\ & \alpha, \alpha_1, \cdots, \alpha_n \in max(id(r)), \\ & r(\alpha) \in J^{\Diamond}, \;\; r(\alpha_1), \cdots, r(\alpha_n) \in Ch \times Stat, \\ & r(\alpha) = \langle c_1?x_1\& \cdots \& c_n?x_n\rangle, \;\; r(\alpha_1) = c_1?s_1, \cdots, r(\alpha_n) = c_n?s_n\} \end{aligned}$$

scheda produces schedules of length 1, that are used to activate individual computations (statements). *scheds* produces schedules of length > 1 that are used to model the pattern matching synchronization that is characteristic for languages based on the Join calculus [8]. Obviously, $scheda(r) \cap scheds(r) = \emptyset$, for all $r \in Res$.

The operational semantics of L^2_J is based on a transition relation $\rightarrow \; \subseteq Conf' \times (Res \times \Theta \times \Sigma)$, with elements (t, t') written in the notation $t \rightarrow t'$. It is easy to check, by using the rules of T^2_J (definition 5.2), that if $t \in Conf'$ and $t \rightarrow t'$ then $t' \in (Res \times \Theta \times \Sigma)$, first for all $t = (s, r^\alpha, \theta, \sigma) \in (Stat \times Res^\alpha \times \Theta \times \Sigma)$ by structural induction on s, and next for all configuration of the form $t = (r, \theta, \sigma) \in (Res \times \Theta \times \Sigma)$, by using the fact that $id(r)$ is finite. We use the convention:

$$t_1 \nearrow t_2 \qquad \text{is an abbreviation for:} \qquad \frac{t_2 \rightarrow t'}{t_1 \rightarrow t'}$$

Definition 5.2 *(Transition system for L^2_J: T^2_J) The transition relation for L^2_J is the smallest subset of $Conf' \times (Res \times \Theta \times \Sigma)$ satisfying the rules below. In (A2) $\overline{\sigma} = (\sigma \mid v \mapsto \mathcal{E}[\![e]\!](\sigma))$.*

(A1) $(\mathsf{skip}, r^\alpha, \theta, \sigma) \rightarrow (r, \theta, \sigma)$

(A2) $(v := e, r^\alpha, \theta, \sigma) \rightarrow (r, \theta, \overline{\sigma})$

(A3) $(c!s, r^\alpha, \theta, \sigma) \rightarrow ((r \mid \alpha \mapsto c!s), \theta, \sigma)$

(A4) $(j, r^\alpha, \theta, \sigma) \rightarrow ((r \mid \alpha \mapsto \langle j\rangle), \theta, \sigma)$

(A5) $(\mathsf{call}(x), r^\alpha, \theta, \sigma) \rightarrow (\theta(x), r^\alpha, \theta, \sigma)$

(R6) $(s_1 + s_2, r^\alpha, \theta, \sigma) \nearrow (s_1, r^\alpha, \theta, \sigma)$

(R7) $(s_1 + s_2, r^\alpha, \theta, \sigma) \nearrow (s_2, r^\alpha, \theta, \sigma)$

(R8) $(s_1 ; s_2, r^\alpha, \theta, \sigma) \nearrow (s_1, (r \mid \alpha \mapsto s_2)^{\alpha \cdot 1}, \theta, \sigma)$

(R9) $(s_1 \| s_2, r^\alpha, \theta, \sigma) \nearrow (s_1, (r \mid \alpha \cdot 2 \mapsto s_2)^{\alpha \cdot 1}, \theta, \sigma)$

(R10) $(s_1 \| s_2, r^\alpha, \theta, \sigma) \nearrow (s_2, (r \mid \alpha \cdot 1 \mapsto s_1)^{\alpha \cdot 2}, \theta, \sigma)$

(A11) $(r, \theta, \sigma) \rightarrow (r \setminus \{\alpha, \alpha_1, \cdots, \alpha_n\}, \overline{\theta}, \sigma) \qquad \forall (\alpha, \alpha_1, \cdots, \alpha_n) \in scheds(r)$

where $r(\alpha) = \langle c_1?x_1\& \cdots \& c_n?x_n\rangle, \;\; r(\alpha_1) = c_1!s_1, \cdots, r(\alpha_n) = c_n!s_n,$

$\overline{\theta} = (\theta \mid x_1 \mapsto s_1 \mid \cdots \mid x_n \mapsto s_n)$

(R12) $(r, \theta, \sigma) \nearrow (r(\alpha), (r \setminus \{\alpha\})^\alpha, \theta, \sigma) \qquad \forall(\alpha) \in \mathit{scheda}(r)$

The CSC technique is a semantic formalization of a process scheduler [22]. In a configuration of the form $(s, r^\alpha, \theta, \sigma)$ the computation (statement) s is *active*. The other computations are contained in the resumption r and wait for their turn to be activated (scheduled for evaluation). The identifier α points to the conceptual position of the *active computation* s with respect to the computations contained in r. Each computation remains active only until it performs an elementary step (axioms A1-A5). Subsequently, another computation taken from the resumption is activated (rule (R12)). In this way it can be obtained the desired interleaving behavior for parallel composition.

The axioms (A3) and (A4) create communication attempts that are combined by using rule (A11). An L_J^2 communication is an interaction in which several statements are transmitted concurrently on different channels. In (A11) the process that executes the join pattern $c_1?x_1\& \cdots \& c_n?x_n$ receives the statements $s_1, \cdots, s_n$ and assigns them to corresponding statement variables $x_1, \cdots, x_n$. Rule (A5) specifies that the execution of procedure x in the current syntactic store θ amounts to the execution of $\theta(x)$. Rules (A11) and (R12) give priority to the computations at the leaves of the tree that represents the resumption. In the case of a sequential composition $s_1; s_2$ (rule (R8)) the computations s_1 and s_2 are given the identifiers $\alpha \cdot 1$ and α, respectively. As $\alpha \cdot 1 > \alpha$, the evaluation of s_2 will begin only after the completion of the evaluation of s_1. Rules (R9) and (R10) define the semantics of a parallel composition $s_1 \parallel s_2$. The computations s_1 and s_2 are given the identifiers $\alpha \cdot 1$ and $\alpha \cdot 2$, respectively; $\alpha \cdot 1$ and $\alpha \cdot 2$ are incomparable (with respect to $\leq$) therefore the statements s_1 and s_2 are evaluated in an interleaved manner.

Definition 5.3 *(Normal termination and deadlock) We define* $\mathit{terminates}, \mathit{blocks} : \mathit{Res} \to \mathit{Bool}$:

$$\mathit{terminates}(r) = (\mathit{id}(r) = \emptyset)$$

$$\mathit{blocks}(r) = (\mathit{id}(r) \neq \emptyset) \wedge (\mathit{sched}(r) = \emptyset)$$

Let $t(\in \mathit{Conf}')$. We say that t terminates *if $t = (r, \theta, \sigma) \in (\mathit{Res} \times \Theta \times \Sigma)$ and $\mathit{terminates}(r)$. Also, we say that t* blocks *if $t = (r, \theta, \sigma) \in (\mathit{Res} \times \Theta \times \Sigma)$ and $\mathit{blocks}(r)$.*

Let $t \in \mathit{Conf}'$. We write $t \to$ to express the fact that t has transitions, i.e. $\exists t' : t \to t'$. Also, we write $t \not\to$ to express the fact that t has no transitions, i.e. $\neg(\exists t' : t \to t')$.

Lemma 5.4

(a) $t \not\to \Leftrightarrow t$ terminates or t blocks

(b) If $t = (s, r^\alpha, \theta, \sigma) \in (\mathit{Stat} \times \mathit{Res}^\alpha \times \Theta \times \Sigma)$ then $t \to$.

Lemma 5.4 follows easily from the rules of T_J^2 (5.4(b) follows by structural induction on s).

Definition 5.5 *Let $(x \in)\mathbf{X}$ be a nonempty complete space. The space $\mathbf{X}_\delta^\infty$ is defined by the domain equation $\mathbf{X}_\delta^\infty \cong \{\epsilon\} + \{\delta\} + (\mathbf{X} \times \frac{1}{2} \cdot \mathbf{X}_\delta^\infty)$. ϵ models the* empty sequence. *δ is used to model* deadlock. *The elements of $\mathbf{X}_\delta^\infty$ are finite sequences (possibly followed by δ) or infinite sequences over $\mathbf{X}$. We usually write $x_1 x_2 \cdots x_n$ and $x_1 x_2 \cdots x_n \delta$ instead of $(x_1, (x_2, \cdots (x_n, \epsilon) \cdots))$ and $(x_1, (x_2, \cdots (x_n, \delta) \cdots))$. Also, instead of $(x_1, (x_2, \cdots))$ we write $x_1 x_2 \cdots$. If $\mathbf{X}$ is endowed with the discrete metric we obtain a Baire-like distance on $\mathbf{X}_\delta^\infty$ [4].*

Definition 5.6 *(Operational semantics ($\mathcal{O}$) for L_J^2)*

(a) *(Semantic universe) The final yield of the operational semantics is an element of a* linear-time domain*, i.e. a collection of execution traces (see, e.g. [4]); each execution trace is a sequence of pairs $(\theta,\sigma)\in\Theta\times\Sigma$. We endow the space $\Theta\times\Sigma$ with the discrete metric and define: $\mathbf{P_O}=\mathcal{P}_{nco}((\Theta\times\Sigma)^{\infty}_{\delta})$.*

(b) *Let $(S\in)Sem_O=Conf'\to\mathbf{P_O}$ and let $\Psi:Sem_O\to Sem_O$ be given by:*

$\Psi(S)(t)=$

$$\begin{cases}\{\epsilon\} & \text{if } t \textit{ terminates}\\ \{\delta\} & \text{if } t \textit{ blocks}\\ \cup\{(\theta,\sigma)\cdot S(r,\theta,\sigma)\mid t\to(r,\theta,\sigma)\} & \textit{otherwise}\end{cases}$$

(c) *Let $r_0(\in Res)=(\emptyset,\lambda\alpha.\mathsf{skip})$. We put $\mathcal{O}=fix(\Psi)$ and define $\mathcal{O}[\![\cdot]\!]:Stat\to(\Theta\times\Sigma)\to\mathbf{P_O}$*

$$\mathcal{O}[\![s]\!](\theta,\sigma)=\mathcal{O}(s,{r_0}^{\epsilon},\theta,\sigma)$$

It is easy to check that $(s,{r_0}^{\epsilon},\theta,\sigma)$ is a derivable configuration, and if t is derivable and $t\to t'$ or $t\nearrow t'$ then t' is also derivable. Also, one can check that T_J^2 is *finitely branching* (for all $t\in Conf'$ the set $\{t'\mid t\to t'\}$ is finite). The proof can proceed in two steps: first for all the derivable configurations of the form $(s,r^{\alpha},\theta,\sigma)$ by structural induction on s and next for all configurations of the form (r,θ,σ) by using the fact that the set $id(r)$ is finite. This implies that T_J^2 induces a compact operational semantics [4]. The mapping Ψ is a contraction (and thus it has a unique fixed point) due to the "(θ,σ)"-step in its definition.

Example 5.7 *Let* $s_!=(c_1!(v:=1))\parallel(c_2!\,\mathsf{skip})$ *and* $s_?=(c_1?x_1\&c_2?x_2);(\mathsf{call}(x_1)\parallel\mathsf{call}(x_2))$. *Let also $\sigma\in\Sigma,\theta\in\Theta$ and $\overline{\sigma}=(\sigma\mid v\mapsto 1)$, $\overline{\theta}=(\theta\mid x_1\mapsto(v:=1)\mid x_2\mapsto\mathsf{skip})$. One can check that:*

$$\begin{aligned}\mathcal{O}[\![s_!\parallel s_?]\!](\theta,\sigma)=\{&(\theta,\sigma)(\theta,\sigma)(\theta,\sigma)(\overline{\theta},\sigma)(\overline{\theta},\sigma)(\overline{\theta},\sigma)(\overline{\theta},\sigma)(\overline{\theta},\overline{\sigma}),\\ &(\theta,\sigma)(\theta,\sigma)(\theta,\sigma)(\overline{\theta},\sigma)(\overline{\theta},\sigma)(\overline{\theta},\sigma)(\overline{\theta},\overline{\sigma})(\overline{\theta},\overline{\sigma}),\\ &(\theta,\sigma)(\theta,\sigma)(\theta,\sigma)(\overline{\theta},\sigma)(\overline{\theta},\sigma)(\overline{\theta},\overline{\sigma})(\overline{\theta},\overline{\sigma})(\overline{\theta},\overline{\sigma})\}\end{aligned}$$

The first 3 steps correspond to applications of rules (A3) and (A4) of T_J^2. The 4th step results from an application of (A11). The 5th step is a silent step produced by using (A5). Depending on the moment when the assignment $v:=1$ is executed we obtain 3 different execution traces.

6 Denotational semantics ($\mathcal{D}$)

In this section we employ the CSC technique in designing a continuation-based denotational (compositional) semantics for L_J^2. The final yield of the denotational semantics is also an element of a linear-time domain $\mathbf{P_D}$ (a collection of execution traces). In the case of denotational semantics each trace is a sequence of pairs (ω,σ), where $\omega(\in\Omega)$ is a *semantic store.* The type of the denotational semantics $\mathcal{D}$ for L_J^2 is $Sem_D=Stat\to\mathbf{D}$:

$$(\phi\in)\mathbf{D}\cong(\mathbf{Cont}^{\alpha}\times\Omega)\xrightarrow{1}\Sigma\to\mathbf{P_D}$$

$$(\gamma \in)\mathbf{Cont} = \{\!|\mathbf{Comp}|\!\} \qquad \mathbf{Cont}^\alpha = \mathbf{Cont} \times Id$$

$$(\varrho \in)\mathbf{Comp} = J^{\diamondsuit} + (Ch \times \tfrac{1}{2} \cdot \mathbf{D}) + \tfrac{1}{2} \cdot \mathbf{D}$$

$$(\omega \in)\Omega = Svar \to \tfrac{1}{2} \cdot \mathbf{D} \qquad (p \in)\mathbf{P_D} = \mathcal{P}_{nco}((\Omega \times \Sigma)^{\infty}_{\delta})$$

A continuation is a configuration of computations ($\in$ **Comp**). In [21] the elements of **Cont** are called *closed continuations*, and the elements of $\mathbf{Cont}^\alpha$ are called *open continuations*. Both open and closed continuations are representations of the *rest of the computation* [16]. An open continuation is also an *evaluation context* [7, 21] for the active computation (denotation). In an expression $\mathcal{D}(s)(\gamma^\alpha, \omega)(\sigma)$ the identifier α points to the conceptual position of the active computation $\mathcal{D}(s)$ with respect to the other computations contained in the continuation γ. We denote typical elements (γ, α) of $\mathbf{Cont}^\alpha$ by γ^α. Also, for easier readability we denote typical elements (c, ϕ) of $Ch \times \mathbf{D}$ by $c!\phi$. In the equations above the sets Id (and Π), Σ, $J^{\diamondsuit}$ and Ch are endowed with the discrete metric, which is an ultrametric. The above system of domain equations has a unique solution (up to isometry) [1]. The solutions for **D** and the other domains are obtained as complete ultrametric spaces.

We define the mappings $Sched, Scheda, Scheds : \mathbf{Cont} \to Sched$ that compute schedules and the predicates $Terminates, Blocks : \mathbf{Cont} \to Bool$ that we need to detect normal termination and deadlock. Obviously, $Scheda(\gamma) \cap Scheds(\gamma) = \emptyset$, for all $\gamma \in \mathbf{Cont}$.

$$Sched(\gamma) = Scheda(\gamma) \cup Scheds(\gamma)$$

$$Scheda(\gamma) = \{(\alpha) \mid \alpha \in max(id(\gamma)), \gamma(\alpha) \in \mathbf{D}\}$$

$$\begin{aligned} Scheds(\gamma) = \{&(\alpha, \alpha_1, \cdots, \alpha_n) \mid \alpha, \alpha_1, \cdots, \alpha_n \in max(id(\gamma)),\\ &\gamma(\alpha) \in J^{\diamondsuit}, \gamma(\alpha_1), \cdots, \gamma(\alpha_n) \in Ch \times \mathbf{D},\\ &\gamma(\alpha) = \langle c_1?x_1 \& \cdots \& c_n?x_n\rangle, \gamma(\alpha_1) = c_1?\phi_1, \cdots, \gamma(\alpha_n) = c_n?\phi_n\} \end{aligned}$$

$$Terminates(\gamma) = (id(\gamma) = \emptyset)$$

$$Blocks(\gamma) = (id(\gamma) \neq \emptyset) \wedge (Sched(\gamma) = \emptyset)$$

Definition 6.1 *(Denotational semantics ($\mathcal{D}$) for L^2_J)*

(a) We define $C : (\mathbf{Cont} \times \Omega) \to \Sigma \to \mathbf{P_D}$ *by:*

```
$C(\gamma, \omega)(\sigma) =$
    if $Terminates(\gamma)$ then $\{\epsilon\}$
    else if $Blocks(\gamma)$ then $\{\delta\}$
    else $(\bigcup\{\gamma(\alpha)((\gamma \setminus \{\alpha\})^\alpha, \omega)(\sigma) \mid (\alpha) \in Scheda(\gamma)\})$   $\bigcup$
        $(\bigcup\{(\overline{\omega}, \sigma) \cdot C(\gamma \setminus \{\alpha, \alpha_1, \cdots, \alpha_n\}, \overline{\omega})(\sigma)$
            where  $\gamma(\alpha) = \langle c_1?x_1 \& \cdots \& c_n?x_n\rangle, \gamma(\alpha_1) = c_1!\phi_1, \cdots, \gamma(\alpha_n) = c_n!\phi_n,$
                $\overline{\omega} = (\omega \mid x_1 \mapsto \phi_1 \mid \cdots \mid x_n \mapsto \phi_n)$
```

$\mid (\alpha, \alpha_1, \cdots, \alpha_n) \in Scheds(\gamma)\})$

(b) We define $\Phi : Sem_D \to Sem_D$ (for $(S \in)Sem_D$) by:

$$\Phi(S)(\,\mathsf{skip}\,)(\gamma^\alpha, \omega)(\sigma) = (\omega, \sigma) \cdot C(\gamma, \omega)(\sigma)$$

$$\Phi(S)(v := e)(\gamma^\alpha, \omega)(\sigma) = (\omega, \overline{\sigma}) \cdot C(\gamma, \omega)(\overline{\sigma}) \qquad where \quad \overline{\sigma} = (\sigma \mid v \mapsto \mathcal{E}[\![e]\!](\sigma))$$

$$\Phi(S)(c!s)(\gamma^\alpha, \omega)(\sigma) = (\omega, \sigma) \cdot C((\gamma \mid \alpha \mapsto c!S(s)), \omega)(\sigma)$$

$$\Phi(S)(j)(\gamma^\alpha, \omega)(\sigma) = (\omega, \sigma) \cdot C((\gamma \mid \alpha \mapsto \langle j \rangle), \omega)(\sigma)$$

$$\Phi(S)(\,\mathsf{call}(x)\,)(\gamma^\alpha, \omega)(\sigma) = (\omega, \sigma) \cdot \omega(x)(\gamma^\alpha, \omega)(\sigma)$$

$$\Phi(S)(s_1 + s_2)(\gamma^\alpha, \omega)(\sigma) = \Phi(S)(s_1)(\gamma^\alpha, \omega)(\sigma) \;\cup\; \Phi(S)(s_2)(\gamma^\alpha, \omega)(\sigma)$$

$$\Phi(S)(s_1; s_2)(\gamma^\alpha, \omega)(\sigma) = \Phi(S)(s_1)((\gamma \mid \alpha \mapsto S(s_2))^{\alpha \cdot 1}, \omega)(\sigma)$$

$$\Phi(S)(s_1 \| s_2)(\gamma^\alpha, \omega)(\sigma) =$$

$$\Phi(S)(s_1)((\gamma \mid \alpha \cdot 2 \mapsto S(s_2))^{\alpha \cdot 1}, \omega)(\sigma) \;\cup\; \Phi(S)(s_2)((\gamma \mid \alpha \cdot 1 \mapsto S(s_1))^{\alpha \cdot 2}, \omega)(\sigma)$$

(c) We put $\mathcal{D} = fix(\Phi)$. Let $\gamma_0 = (\emptyset, \lambda\alpha.\mathcal{D}(\,\mathsf{skip}\,))$. We define $\mathcal{D}[\![\cdot]\!] : Stat \to (\Omega \times \Sigma) \to \mathbf{P_D}$ by:

$$\mathcal{D}[\![s]\!](\omega, \sigma) = \mathcal{D}(s)({\gamma_0}^\epsilon, \omega)(\sigma)$$

Definition 6.1 is justified by lemma 6.2.

Lemma 6.2

(a) The mapping C (introduced in 6.1) is well-defined.

(b) For all $\gamma_1, \gamma_2 \in \mathbf{Cont}, \omega_1, \omega_2 \in \Omega$: $d(C(\gamma_1, \omega_1), C(\gamma_2, \omega_2)) \leq 2 \cdot d((\gamma_1, \omega_1), (\gamma_2, \omega_2))$.

Also, for all $S \in Sem_D, s \in Stat, \gamma^\alpha \in \mathbf{Cont}^\alpha, \omega \in \Omega, \sigma \in \Sigma$:

(c) $\Phi(S)(s)(\gamma^\alpha, \omega)(\sigma) \in \mathbf{P_D}$ (it is well defined),

(d) $\Phi(S)(s)$ is nonexpansive (in (γ^α, ω)), and

(e) Φ is $\frac{1}{2}$ - contractive (in S).

Proof. Similar lemmas are given in [18, 22]. Here we only prove 6.2(e). We proceed by structural induction on s. We only consider the subcase $s \equiv s_1 \parallel s_2$.

$$d(\Phi(S_1)(s_1 \parallel s_2)(\gamma^\alpha, \omega)(\sigma), \Phi(S_2)(s_1 \parallel s_2)(\gamma^\alpha, \omega)(\sigma))$$

$$= d(\Phi(S_1)(s_1)((\gamma \mid \alpha \cdot 2 \mapsto S_1(s_2))^{\alpha \cdot 1}, \omega)(\sigma) \cup \Phi(S_1)(s_2)((\gamma \mid \alpha \cdot 1 \mapsto S_1(s_1))^{\alpha \cdot 2}, \omega)(\sigma),$$

$$\Phi(S_2)(s_1)((\gamma \mid \alpha \cdot 2 \mapsto S_2(s_2))^{\alpha \cdot 1}, \omega)(\sigma) \cup \Phi(S_2)(s_2)((\gamma \mid \alpha \cdot 1 \mapsto S_2(s_1))^{\alpha \cdot 2}, \omega)(\sigma))$$

$\leq$ ['$\cup$' is nonexpansive]

$$max\{d(\Phi(S_1)(s_1)((\gamma \mid \alpha \cdot 2 \mapsto S_1(s_2))^{\alpha \cdot 1}, \omega)(\sigma)$$

$$\Phi(S_2)(s_1)((\gamma \mid \alpha \cdot 2 \mapsto S_2(s_2))^{\alpha \cdot 1}, \omega)(\sigma))^{(*)}$$

$d(\Phi(S_1)(s_2)((\gamma \mid \alpha\cdot 1 \mapsto S_1(s_1))^{\alpha\cdot 2}, \omega)(\sigma),$

$\Phi(S_2)(s_2)((\gamma \mid \alpha\cdot 1 \mapsto S_2(s_1))^{\alpha\cdot 2}, \omega)(\sigma))^{(**)}\}$

We only treat $^{(*)}$; $^{(**)}$ can be handled similarly.

$^{(*)} \leq$ [d is an ultrametric]

$max\{d(\Phi(S_1)(s_1)((\gamma \mid \alpha\cdot 2 \mapsto S_1(s_2))^{\alpha\cdot 1}, \omega)(\sigma), \Phi(S_1)(s_1)((\gamma \mid \alpha\cdot 2 \mapsto S_2(s_2))^{\alpha\cdot 1}, \omega)(\sigma)),$

$d(\Phi(S_1)(s_1)((\gamma \mid \alpha\cdot 2 \mapsto S_2(s_2))^{\alpha\cdot 1}, \omega)(\sigma), \Phi(S_2)(s_1)((\gamma \mid \alpha\cdot 2 \mapsto S_2(s_2))^{\alpha\cdot 1}, \omega)(\sigma))\}$

$\leq$ [6.2(d), ind. hyp.]

$max\{d((\gamma \mid \alpha\cdot 2 \mapsto S_1(s_2)), (\gamma \mid \alpha\cdot 2 \mapsto S_2(s_2))), \frac{1}{2}\cdot d(S_1,S_2)\}$

$= max\{\frac{1}{2}\cdot d(S_1(s_2), S_2(s_2)), \frac{1}{2}\cdot d(S_1,S_2)\} \leq \frac{1}{2}\cdot d(S_1,S_2)$

■

7 Relating $\mathcal{O}$ and $\mathcal{D}$

We prove that $\forall s \in Stat, \theta \in \Theta, \sigma \in \Sigma : SEM(\mathcal{O}[\![s]\!](\theta,\sigma)) = \mathcal{D}[\![s]\!](sem(\theta),\sigma)$, where:

Definition 7.1

(a) $sem : \Theta \to \Omega$ is defined by: $sem(\theta) = \lambda x.\mathcal{D}(\theta(x))$.

(b) We let w range over $(w \in)(\Theta \times \Sigma)^{\infty}_{\delta}$ and define $Sem : (\Theta\times\Sigma)^{\infty}_{\delta} \to (\Omega\times\Sigma)^{\infty}_{\delta}$ as the unique mapping satisfying: $Sem(\epsilon) = \epsilon$, $Sem(\delta) = \delta$ and $Sem((\theta,\sigma)\cdot w) = (sem(\theta),\sigma)\cdot Sem(w)$.

(c) Recall that $\mathbf{P_O} = \mathcal{P}_{nco}((\Theta\times\Sigma)^{\infty}_{\delta})$ and $\mathbf{P_D} = \mathcal{P}_{nco}((\Omega\times\Sigma)^{\infty}_{\delta})$. We let W range over $(W \in)\mathbf{P_O}$ and define $SEM : \mathbf{P_O} \to \mathbf{P_D}$ by: $SEM(W) = \{Sem(w) \mid w \in W\}$.

Definition 7.2 *Let $\varsigma : Comp \to \mathbf{Comp}$, $\varsigma(\langle j\rangle) = \langle j\rangle$, $\varsigma(c!s) = c!\mathcal{D}(s)$ and $\varsigma(s) = \mathcal{D}(s)$. We define $\Gamma : Res \to \mathbf{Cont}$ by $\Gamma(r) = (id(r), \lambda\alpha.\varsigma(r(\alpha)))$. By using the abbreviation introduced in section 4 we can write $\Gamma(r)(\alpha) = \varsigma(r(\alpha))$. Also, we define $\mathcal{R} : Conf' \to \mathbf{P_D}$:*

$\mathcal{R}(r,\theta,\sigma) = C(\Gamma(r), sem(\theta))(\sigma)$

$\mathcal{R}(s, r^{\alpha}, \theta, \sigma) = \mathcal{D}(s)(\Gamma(r)^{\alpha}, sem(\theta))(\sigma)$

As $\mathcal{D}[\![s]\!](sem(\theta),\sigma) = \mathcal{D}(s)(\gamma_0{}^{\epsilon}, sem(\theta))(\sigma) = \mathcal{D}(s)(\Gamma(r_0)^{\epsilon}, sem(\theta))(\sigma) = \mathcal{R}(s, r_0{}^{\epsilon}, \theta, \sigma)$, and $\mathcal{O}[\![s]\!](\theta,\sigma) = \mathcal{O}(s, r_0{}^{\epsilon}, \theta, \sigma)$, in order to prove that $SEM(\mathcal{O}[\![s]\!](\theta,\sigma)) = \mathcal{D}[\![s]\!](sem(\theta),\sigma)$ it suffices to show that $SEM \circ \mathcal{O} = \mathcal{R}$. In 7.4 and 7.6 we prove that both $SEM \circ \mathcal{O}$ and $\mathcal{R}$ are fixed points of the same (higher-order) contraction $\Phi_{\mathcal{R}}$, defined in 7.3.

Definition 7.3 *Let* $(S \in) Sem_{\mathcal{R}} = Conf' \to \mathbf{P_D}$. *We define* $\Phi_{\mathcal{R}} : Sem_{\mathcal{R}} \to Sem_{\mathcal{R}}$ *by:*

$$\Phi_{\mathcal{R}}(S)(t) =$$

$$\begin{cases} \{\epsilon\} & \text{if } t \text{ terminates} \\ \{\delta\} & \text{if } t \text{ blocks} \\ \cup\{(sem(\theta),\sigma) \cdot S(r,\theta,\sigma) \mid t \to (r,\theta,\sigma)\} & \text{otherwise} \end{cases}$$

The transitions $t \to (r,\theta,\sigma)$ *are with respect to* T_J^2 *(5.2).*

Lemma 7.4 $SEM \circ \mathcal{O} = fix(\Phi_{\mathcal{R}})$

Proof. We show that for all $t \in Conf' : (SEM \circ \mathcal{O})(t) = \Phi_{\mathcal{R}}(SEM \circ \mathcal{O})(t)$. If t terminates $\Phi_{\mathcal{R}}(SEM \circ \mathcal{O})(t) = \{\epsilon\} = (SEM \circ \mathcal{O})(t)$. Also, if t blocks $\Phi_{\mathcal{R}}(SEM \circ \mathcal{O})(t) = \{\delta\} = (SEM \circ \mathcal{O})(t)$. It is easy to check that $SEM((\theta,\sigma) \cdot W) = (sem(\theta),\sigma) \cdot SEM(W)$ and $SEM(W_1 \cup W_2) = SEM(W_1) \cup SEM(W_2)$. So, if t has indeed transitions we have:

$$\Phi_{\mathcal{R}}(SEM \circ \mathcal{O})(t) = \cup\{(sem(\theta),\sigma) \cdot (SEM \circ \mathcal{O})(r,\theta,\sigma) \mid t \to (r,\theta,\sigma)\}$$

$$= \cup\{SEM((\theta,\sigma) \cdot \mathcal{O}(r,\theta,\sigma)) \mid t \to (r,\theta,\sigma)\} \qquad [T_J^2 \text{ is finitely branching}]$$

$$= SEM(\cup\{(\theta,\sigma) \cdot \mathcal{O}(r,\theta,\sigma) \mid t \to (r,\theta,\sigma)\})$$

$$= SEM(\mathcal{O}(t)) = (SEM \circ \mathcal{O})(t)$$

■

Lemma 7.5 collects some properties that are used in the proof of 7.6. The proof of 7.5 is simple enough and left to the reader.

Lemma 7.5

(a) If t *terminates then* $\mathcal{R}(t) = \{\epsilon\}$.

(b) Also, if t *blocks then* $\mathcal{R}(t) = \{\delta\}$.

(c) For all $r \in Res : scheda(r) = Scheda(\Gamma(r))$.

(d) For all $r \in Res : scheds(r) = Scheds(\Gamma(r))$.

(e) For all $r \in Res, \pi \in \Pi : \Gamma(r \setminus \pi) = \Gamma(r) \setminus \pi$

(f) For all $r \in Res, \alpha \in Id, \rho \in Comp$: $\Gamma(r \mid \alpha \mapsto \rho) = (\Gamma(r) \mid \alpha \mapsto \varsigma(\rho))$

(g) For all $\theta \in \Theta, x_1, \cdot, x_n \in Svar, s_1, \cdots, s_n \in Stat$:

$$sem(\theta \mid x_1 \mapsto s_1 \mid \cdots \mid x_n \mapsto s_n) = (sem(\theta) \mid x_1 \mapsto \mathcal{D}(s_1) \mid \cdots \mid x_n \mapsto \mathcal{D}(s_n))$$

Lemma 7.6 $\mathcal{R} = fix(\Phi_{\mathcal{R}})$

The proof of lemma 7.6 is relegated to the appendix. The main result of the paper is obtained in theorem 7.7. The proof of 7.7 combines 7.4, 7.6 and Banach's theorem 2.1.

Theorem 7.7 $SEM(\mathcal{O}[\![s]\!](\theta,\sigma)) = \mathcal{D}[\![s]\!](sem(\theta),\sigma)$, $\forall s \in Stat, \theta \in \Theta, \sigma \in \Sigma$.

Proof.

$$SEM(\mathcal{O}[\![s]\!](\theta,\sigma)) = (SEM \circ \mathcal{O})(s, {r_0}^{\epsilon}, \theta, \sigma) \quad [7.4,7.6,2.1]$$

$$= \mathcal{R}(s, {r_0}^{\epsilon}, \theta, \sigma) = \mathcal{D}(s)(\Gamma(r_0)^{\epsilon}, sem(\theta))(\sigma)$$

$$= \mathcal{D}[\![s]\!](sem(\theta),\sigma)$$

■

8 Concluding remarks and future research

In this paper we extended the semantic models given in [18] with second order communication: sending and receiving of statements rather than simple data values. As far as we know this is the first paper that reports a denotational semantics for (a form of) higher-order communication combined with synchronization on multiple channels in the style introduced in the Join calculus [8]. The semantic models given in this paper and in [18] are designed with CSC continuations, a technique introduced by us in previous work [22]. In a denotational model designed with the CSC technique all (concurrent) control concepts are represented as operations manipulating continuations. The use of CSC continuations proved to be fruitful. We obtained a relatively simple relation between the denotational semantics and the operational semantics by using only basic techniques of metric semantics [4].

In [20] we presented a semantic interpreter designed with the CSC technique for a concurrent object-oriented language with Join methods, inspired by Join Java [10] and Polyphonic C# [5], which in turn are based on the Join calculus [8]. The communication of statements provides a simple form of code mobility. The research reported in this paper shows that CSC continuations can provide a framework for combining code mobility with other fundamental concurrent control concepts in a uniform manner. In the near future we intend to extend the object-oriented system with Join methods introduced in [20] with primitives for object mobility.

Second order communication is a particular form of higher order interaction. Another objective of future research is the study of ω-order communication (parameterized processes of arbitrary high order) in the presence of Join synchronization. In recent unpublished work we developed in Haskell [13] a semantic interpreter that implements a dynamic denotational semantics designed with CSC continuations for a language with ω-order communication and classic point-to-point communication. It should be possible to extend that interpreter with Join synchronization by using the technique presented in this paper. The complete mathematical formalization is again subject of future research that would represent an important step toward a formal denotational semantics for JoCaml [11].

9 Acknowledgements

This paper is based on work partially funded by ANCS, CNMP-PC project no. 11-052/2007, project title: "GlobalComp: Models, semantics logics and technologies for global computing" (2007-2010).

References

[1] 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, 1989.

[2] J.W. de Bakker, and F. van Breugel. Topological models for higher order control flow. *Lecture Notes in Computer Science*, 802:122–142, 1994.

[3] J.W. de Bakker, and F. van Breugel. From Banach to Milner: Metric semantics for second order communication and concurrency. In *Proof, Language and Interaction: Essays in honor of Robin Milner*, Foundations of Computing Series, MIT Press, pages 99–132, 2000.

[4] J.W.de Bakker, and E.P. de Vink. *Control flow semantics.* MIT Press, 1996.

[5] N. Benton, L. Cardelli, and C. Fournet. Modern concurrency abstractions for C#. *ACM Transactions on Programming Languages and Systems*, 25(5):769–804, 2004.

[6] F. van Breugel. Generalizing finiteness conditions of labelled transition systems. *Lecture Notes in Computer Science*, 820:376-387, 1994.

[7] O. Danvy. On Evaluation Contexts, Continuations, and the Rest of the Computation. In H. Thielecke, editor, *Proc. of the 4th ACM SIGPLAN Continuations Workshop*, pages 13–23, 2004.

[8] C. Fournet, and G.Gonthier. The Join calculus: a language for distributed mobile programming. *Lecture Notes in Computer Science*, 25:268–332, 2002.

[9] C.A.R. Hoare. *Communicating sequential processes.* Prentice Hall, 1985.

[10] G.S. von Itzstein. *Introduction of high level concurrency semantics in object-oriented languages.* Ph.D. Thesis, University of South Australia, 2005.

[11] L. Mandel, and L. Maranget. *The JoCaml language, release 3.10.* Available from: `http://jocaml.inria.fr/`, 2007.

[12] INMOS Ltd. *Occam programming manual.* Prentice-Hall, 1984.

[13] S. Peyton Jones, and J. Hughes, editors. Report on the Programming Language Haskell 98: A Non-Strict Purely Functional Language, 1999. Available from `http://www.haskell.org/`.

[14] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Aarhus University, 1981.

[15] J. Rutten. Processes as terms: non-well founded models for bisimulation. *Mathematical Structures in Computer Science*, 2:257–245, 1992.

[16] C. Stratchey, and C. Wadsworth. Continuations: a mathematical semantics for handling full jumps. *Journal of Higher-Order and Symbolic Computation*, 13(1):135–152, 2000 (Reprint of the Technical Monograph PRG-11, Oxford University, Computing Laboratory, 1974).

[17] E.N. Todoran. Metric semantics for modern second order communication abstractions. In *Proceedings of 2009 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP'09)*, pages 215-219, 2009.

[18] E.N. Todoran. Comparative semantics for modern communication abstractions. In *Proceedings of 2008 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP'08)*, pages 153–161, 2008.

[19] E.N. Todoran, and D.M. Ivan. Metric semantics for second order communication: a continuation-based approach. *Automation, Computers, Applied Mathematics, 16(2)*, pages 201–215, 2007.

[20] E.N. Todoran, and N. Gherman. Semantic interpreter for modern communication abstractions in concurrent object-oriented programming. In *Proc. of 2008 IEEE 10th Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'08)*, pages 289-295, Computer Press, 2008.

[21] E.N. Todoran, and N. Papaspyrou. Continuations for prototyping concurrent languages. Technical Report CSD-SW-TR-1-06, National Technical University of Athens, Software Engineering Laboratory, 2006.

[22] E.N. Todoran. Metric semantics for synchronous and asynchronous communication: a continuation-based approach. *Electronic Notes in Theoretical Computer Science*, 28:119–146, Elsevier, 2000.

A Appendix

Proof. Lemma 7.6 We show that $\forall t \in Conf' : \Phi_{\mathcal{R}}(\mathcal{R})(t) = \mathcal{R}(t)$. This follows immediately from lemma 7.5(a) and 7.5(b) if t terminates or t blocks (i.e. if $t \not\to$). We handle the case when t has indeed transitions in two steps. First, we prove the desired property for all t of the form $t = (s, r^\alpha, \theta, \sigma)(\in Stat \times Res^\alpha \times \Theta \times \Sigma)$ by structural induction on s.

- Case $s \equiv j$.

  $\Phi_{\mathcal{R}}(\mathcal{R})(j, r^\alpha, \theta, \sigma) = (sem(\theta), \sigma) \cdot \mathcal{R}((r \mid \alpha \mapsto \langle j \rangle), \theta, \sigma)$

  $= (sem(\theta), \sigma) \cdot C(\Gamma(r \mid \alpha \mapsto \langle j \rangle), sem(\theta))(\sigma)$ [7.5(f)]

  $= (sem(\theta), \sigma) \cdot C((\Gamma(r) \mid \alpha \mapsto \langle j \rangle), sem(\theta))(\sigma)$

  $= \mathcal{D}(j)(\Gamma(r)^\alpha, sem(\theta))(\sigma) = \mathcal{R}(j, r^\alpha, \theta, \sigma)$

- Case $s \equiv s_1 \parallel s_2$.

  $\Phi_{\mathcal{R}}(\mathcal{R})(s_1 \parallel s_2, r^\alpha, \theta, \sigma)$ [def. T_J^2]

  $= (\bigcup\{(sem(\theta'), \sigma') \cdot \mathcal{R}(r', \theta', \sigma') \mid (s_1, (r \mid \alpha \cdot 2 \mapsto s_2)^{\alpha \cdot 1}, \theta, \sigma) \to (r', \theta', \sigma')\}) \cup$

  $(\bigcup\{(sem(\theta'), \sigma') \cdot \mathcal{R}(r', \theta', \sigma') \mid (s_2, (r \mid \alpha \cdot 1 \mapsto s_1)^{\alpha \cdot 2}, \theta, \sigma) \to (r', \theta', \sigma')\})$

  $= \Phi_{\mathcal{R}}(\mathcal{R})(s_1, (r \mid \alpha \cdot 2 \mapsto s_2)^{\alpha \cdot 1}, \theta, \sigma) \cup \Phi_{\mathcal{R}}(\mathcal{R})(s_2, (r \mid \alpha \cdot 1 \mapsto s_1)^{\alpha \cdot 2}, \theta, \sigma)$ [ind. hyp.]

  $= \mathcal{R}(s_1, (r \mid \alpha \cdot 2 \mapsto s_2)^{\alpha \cdot 1}, \theta, \sigma) \cup \mathcal{R}(s_2, (r \mid \alpha \cdot 1 \mapsto s_1)^{\alpha \cdot 2}, \theta, \sigma)$ [def. $\mathcal{R}$, 7.5(f)]

$$= \mathcal{D}(s_1)((\Gamma(r) \mid \alpha\cdot 2 \mapsto \mathcal{D}(s_2))^{\alpha\cdot 1}, sem(\theta))(\sigma) \cup$$

$$\mathcal{D}(s_2)((\Gamma(r) \mid \alpha\cdot 1 \mapsto \mathcal{D}(s_1))^{\alpha\cdot 2}, sem(\theta))(\sigma)$$

$$= \mathcal{D}(s_1 \parallel s_2)(\Gamma(r)^{\alpha}, sem(\theta))(\sigma) = \mathcal{R}(s_1 \parallel s_2, r^{\alpha}, \theta, \sigma)$$

In the second step we prove that $\Phi(\mathcal{R})(r,\theta,\sigma) = \mathcal{R}(r,\theta,\sigma)$ when $(r,\theta,\sigma)\to$. By 5.6 and 5.2 (rules (A11) and (R12)):

$$\Phi_{\mathcal{R}}(\mathcal{R})(r,\theta,\sigma)$$

$$= (\bigcup\{(sem(\theta'),\sigma')\cdot\mathcal{R}(r',\theta',\sigma') \mid (\alpha)\in scheda(r), (r(\alpha),(r\setminus\{\alpha\})^{\alpha},\theta,\sigma)\to(r',\theta',\sigma')\})^{(*)} \cup$$

$$(\bigcup\{(sem(\overline{\theta}),\sigma)\cdot\mathcal{R}(r\setminus\{\alpha,\alpha_1,\cdots,\alpha_n\},\overline{\theta},\sigma)$$

$$\text{where } r(\alpha) = \langle c_1?x_1\&\cdots\&c_n?x_n\rangle, r(\alpha_1) = c_1!s_1,\cdots, r(\alpha_n) = c_n!s_n,$$

$$\overline{\theta} = (\theta \mid x_1 \mapsto s_1 \mid \cdots \mid x_n \mapsto s_n)$$

$$\mid (\alpha,\alpha_1,\cdots,\alpha_n) \in scheds(r)\})^{(**)}$$

In the sequel we obtain:

$$^{(*)} = \bigcup\{\Phi_{\mathcal{R}}(\mathcal{R})(r(\alpha),(r\setminus\{\alpha\})^{\alpha},\theta,\sigma) \mid (\alpha)\in scheda(r)\} \quad \text{[first step of the proof]}$$

$$= \bigcup\{\mathcal{R}(r(\alpha),(r\setminus\{\alpha\})^{\alpha},\theta,\sigma) \mid (\alpha)\in scheda(r)\} \quad \text{[7.5(c), 7.5(e)]}$$

$$= \bigcup\{\Gamma(r)(\alpha)((\Gamma(r)\setminus\{\alpha\})^{\alpha}, sem(\theta))(\sigma) \mid (\alpha)\in Scheda(\Gamma(r))\}$$

By using 7.5(d) and 7.5(g), for $^{(**)}$ we have:

$$^{(**)} = \bigcup\{(\overline{\omega},\sigma)\cdot C(\Gamma(r)\setminus\{\alpha,\alpha_1,\cdots,\alpha_n\},\overline{\omega})(\sigma)$$

$$\text{where } \Gamma(r)(\alpha) = \langle c_1?x_1\&\cdots\&c_n?x_n\rangle,$$

$$\Gamma(r)(\alpha_1) = c_1!\phi_1,\cdots,\Gamma(r)(\alpha_n) = c_n!\phi_n,$$

$$\overline{\omega} = (sem(\theta) \mid x_1 \mapsto \phi_1 \mid \cdots \mid x_n \mapsto \phi_n)$$

$$\mid (\alpha,\alpha_1,\cdots,\alpha_n) \in Scheds(r)\}$$

In $^{(**)}$ $\phi_i = \mathcal{D}(s_i), 1 \le i \le n$. We obtain:

$$\Phi_{\mathcal{R}}(\mathcal{R})(r,\theta,\sigma) =^{(*)} \cup^{(**)}$$

$$= C(\Gamma(r), sem(\theta))(\sigma)$$

$$= \mathcal{R}(r,\theta,\sigma)$$

■