AUTOMATION
COMPUTERS
APPLIED MATHEMATICS
ISSN 1221–437X
VOL. 14 (2005) NO. 1
PP. 27–41

Comparative Semantics for the Basic Andorra Model

ENEIA TODORAN, PAULINA MITREA AND NIKOLAOS PAPASPYROU

ABSTRACT: This paper employs techniques from metric semantics in defining and relating an operational and a denotational semantics for a simple abstract language which embodies the main control flow notions of Warren's Basic Andorra Model. The both semantic models are designed with the "continuation semantics for concurrency" (CSC) technique.

1 Introduction

The Basic Andorra Model (BAM) was proposed by Warren [13] as a general framework for combining AND parallelism with OR parallelism in logic programming. It reduces the number of inferences (and thus it improves the execution speed) of logic programs by giving priority to deterministic computations over nondeterministic computations as nondeterministic steps could possibly (unnecessarily) multiply work. The BAM was implemented in the Andorra-I system [5] and in Pandorra [2].

The first denotational model for BAM was developed by us in [12]. The semantic model given in [12] was designed by using the "continuation semantics for concurrency" (CSC) technique [11]. Instead of using mathematical notation for the definition of the denotational semantics, in [12] we used the functional programming language Haskell [7].

In this paper, we apply the methodology of metric semantics [Balaganskij-Vlasov(1996), 1] in defining and relating an operational and a denotational semantics for a simple abstract language which embodies the main control flow notions of BAM. The both semantic models are designed with CSC. To the best of our knowledge, this is the first comparative semantic study of BAM.

2 Theoretical preliminaries

The notation $(x \in)X$ introduces the set X with typical element x ranging over X. For any set X, we denote by $|X|$ the *cardinal number* of X. $|X| = 0$ means that X is empty, $|X| < \infty$ means that X is finite and $|X| = \infty$ means that X is an infinite set. For X a set we denote by $\mathcal{P}_\pi(X)$ the collection of all subsets of X which have property π. Let $f \in X \to Y$. The function $f\{y/x\} : X \to Y$ is defined by: $f\{y/x\}(x) = y$ and for any $x' \in X$, $x' \neq x$, $f\{y/x\}(x') = f(x')$. If $f : X \to X$ and $f(x) = x$ we call x a *fixed point* of f. When this fixed point is unique (see 2.1) we write $x = fix(f)$.

Following [Balaganskij-Vlasov(1996)], the study presented in this paper takes place in the mathematical framework of 1-bounded complete metric spaces. We assume known the notions of *metric* and *ultrametric* space, *isometry* (distance preserving bijection between metric spaces; we denote it by '$\cong$') and *completeness* of metric spaces. If (X, d_X), (Y, d_Y) are metric spaces we recall that a function $f : X \to Y$ is a *contraction* if $\exists c \in \mathbf{R}$, $0 \le c < 1$: $\forall x_1, x_2 \in X$: $d_Y(f(x_1), f(x_2)) \le c \cdot d_X(x_1, x_2)$. Also, f is called *non-expansive* if $d_Y(f(x_1), f(x_2)) \le d_X(x_1, x_2)$. We denote the set of all c-contracting (non-expansive) functions from X to Y by $X \xrightarrow{c} Y$ ($X \xrightarrow{1} Y$).

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 [3].*

Let $(a, b \in) A$ be a set. The so-called *discrete metric* d_A on A is defined as follows: $d_A(a, b) = \textsf{if } a = b \textsf{ then } 0 \textsf{ else } 1 \textsf{ fi}$. The so-called *Baire metric* is defined on the set $(x, y \in) A^\infty = A^* \cup A^\omega$ (i.e. A^∞ is the collection of all finite and infinite words over A) by: $d_B(x, y) = 2^{-sup\{n \,|\, x[n] = y[n]\}}$, where for $x \in A^\infty$, $x[n]$ denotes the prefix of x in case $length(x) \ge n$ and x otherwise (where by convention $2^{-\infty} = 0$). For any set A, (A, d_A) and (A^∞, d_B) are complete ultrametric spaces.

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), $(u, v \in) X \sqcup Y$ (the disjoint union) and on $(U, V \in) \mathcal{P}(X)$ (the power set of X) one can define the following metrics:*

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

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

(c) $d_{X \times Y} : (X \times Y) \times (X \times Y) \to [0, 1]$: $d_{X \times Y}([x_1, y_1], [x_2, y_2]) = max\{d_X(x_1, x_2), d_Y(y_1, y_2)\}$

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

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

(e) $d_H : \mathcal{P}(X) \times \mathcal{P}(X) \to [0, 1]$ *is the so-called* Hausdorff distance *defined as follows:*

$$d_H(U, V) = max\{sup_{u \in U} d(u, V), sup_{v \in V} d(v, U)\}, \textit{ where } d(u, W) = inf_{w \in W} d(u, w)$$

with the convention that $sup\, \emptyset = 0$ and $inf\, \emptyset = 1$.

We recall that given a metric space (X, d_X) a subset A of X is called *compact* whenever each sequence in A has a convergent subsequence with limit in A. We will use the abbreviations $\mathcal{P}_{co}(\cdot)$ ($\mathcal{P}_{nco}(\cdot)$) to denote the power set of compact (non-empty and compact) subsets of '$\cdot$'.

Theorem 2.3 *Let $(X, d_X), (Y, d_Y), d_{\frac{1}{2} \cdot X}, d_{X \to Y}, d_{X \times Y}, d_{X \sqcup Y}$ and d_H be as in definition 2.2. In case d_X, d_Y are ultrametrics, so are $d_{\frac{1}{2} \cdot X}, d_{X \to Y}, d_{X \times Y}, d_{X \sqcup Y}$ and d_H. If in addition $(X, d_X), (Y, d_Y)$ are complete then $(X, d_{\frac{1}{2} \cdot X})$, $(X \to Y, d_{X \to Y})$, $(X \xrightarrow{1} Y, d_{X \to Y})$, $(X \times Y, d_{X \times Y})$, $(X \sqcup Y, d_{X \sqcup Y})$, $(\mathcal{P}_{co}(X), d_H)$ and $(\mathcal{P}_{nco}(X), d_H)$ are also complete metric spaces.*

In the sequel we will often suppress the metrics part in domain definitions. In particular we will write $\frac{1}{2} \cdot X$ instead of $(X, d_{\frac{1}{2} \cdot X})$.

3 Syntax and Operational Semantics ($\mathcal{O}$)

We consider a simple abstract language, called L_{BAM}, which embodies the main control flow notions of Warren's Basic Andorra Model [13]. The syntax of L_{BAM} is summarized in 3.1. The basic components are a set $(a \in)Act$ of *atomic actions* (denoting primitive relations that evaluate to $true$; we assume that $true \in Act$), a special symbol $fail$ (denoting any primitive relation that evaluates to $false$) and a set $(x, y, z... \in)PVar$ of procedure variables. We also let b range over $(b \in)Act \cup \{fail\}$.

Definition 3.1 *We define the syntax of* L_{BAM} *as follows:*

(a) (Statements) $s(\in Stat) ::= a \mid fail \mid \ll c \gg \mid \langle k \rangle \mid \#\langle k \rangle \mid x \mid s \| s$, *where*

$$c(\in CStat) ::= (b \to s) \mid c + c$$
$$k(\in KStat) ::= (b?s) \mid k + k$$

Let also: $o(\in OStat) ::= \ll c \gg \mid \langle k \rangle \mid \#\langle k \rangle$.

(b) (Guarded statements) $g(\in GStat) ::= a \mid fail \mid \ll c \gg \mid \langle k \rangle \mid \#\langle k \rangle \mid g \| g$

(c) (Declarations) $(D \in)Decl = Pvar \to GStat$. *Following [Balaganskij-Vlasov(1996)], we work with a fixed declaration D!*

(e) (Programs) $(\rho \in)L_{BAM} = Decl \times Stat$.

L_{BAM} provides an operator for parallel composition $'\|'$ (interpreted as parallel AND), an n-ary operator for don't care nondeterministic choice $'\ll\gg'$ and two n-ary operators for don't know nondeterministic choice $'\langle\rangle'$ (implemented as sequential OR, i.e. as a backtracking mechanism) and respectively $'\#\langle\rangle'$ (implemented as parallel OR). In L_{BAM} we encounter *don't care goals*, which are constructs of the form $\ll b_1 \to s_1 + ... + b_n \to s_n \gg$, and *don't know goals*, which are constructs of the form $\langle b_1?s_1 + ... + b_n?s_n \rangle$ or $\#\langle b_1?s_1 + ... + b_n?s_n \rangle$. Following the terminology in AKL [6], the guard operator $'\to'$ is called *commit*, and $'?'$ is called *wait*. The "prefix" b of a construct $b \to s$ or $b?s$ is either an element $a \in Act$ or the symbol $fail$. These prefixes will be the only means for detecting the determinacy of a nondeterministic choice statement. This design decision is in the spirit of Warren's Basic Andorra Model [13] which is based on flat guards (the so-called Extended Andorra Model [6], [14], which is based on deep guards is not discussed in this paper).

Our restriction to guarded recursion (3.1(b), 3.1(c)) is natural in the context of logic programming where the execution of each goal starts with head unification.

Execution in L_{BAM} alternates between (the *determinate* or) the AND-parallel phase and the *nondeterminate* phase which is activated by the *deadlock* phase. In the AND-parallel phase all goals in a conjunction are reduced concurrently. A don't care goal can always be reduced. A don't know goal can be reduced only if it is determinate, i.e. if at most one of its prefixes is $\neq fail$. When only (don't know) nondeterminate goals remain the deadlock phase is activated that chooses one of the alternatives for a don't know goal and proceeds. The multiple alternatives may be tried either in sequence (giving rise to a backtracking mechanism) or in parallel (giving rise to OR-parallelism).

In 3.2 we introduce a predicate *det* which returns $true$ if a goal is determinate and $false$ if the goal is nondeterminate. The auxiliary mappings NF_c, NF_k and NF_o are used for computing the non-failing alternatives of a nondeterministic choice statement. The well-definedness of *det* follows by induction on the complexity measure c_s defined in 3.7.

Definition 3.2

(a) *Let* $\tilde{c}(\in \tilde{C}Stat) ::= (a{\rightarrow}s) \mid \tilde{c}{+}\tilde{c}$. *We define* $NF_c{:}CStat{\rightarrow}(\tilde{C}Stat{\cup}\{fail\})$ *by putting:*

$$NF_c(fail{\rightarrow}s){=}fail, \quad NF_c(a{\rightarrow}s){=}a{\rightarrow}s, \quad \text{and} \quad NF_c(c_1 + c_2){=}NF_c(c_1)\tilde{+}NF_c(c_2),$$

where $\tilde{+}{:}(\tilde{C}Stat{\cup}\{fail\})^2{\rightarrow}(\tilde{C}Stat{\cup}\{fail\})$ *is given by:*

$$fail\tilde{+}fail{=}fail, \quad fail\tilde{+}\tilde{c}{=}\tilde{c}\tilde{+}fail{=}\tilde{c}, \quad \text{and} \quad \tilde{c}_1\tilde{+}\tilde{c}_2{=}\tilde{c}_1{+}\tilde{c}_2.$$

(b) *Let* $\hat{k}(\in \hat{K}Stat){::=}(a?s) \mid \hat{k}{+}\hat{k}$. *We define* $NF_k{:}KStat{\rightarrow}(\hat{K}Stat{\cup}\{fail\})$ *by putting:*

$$NF_k(fail?s){=}fail, \quad NF_k(a?s){=}a?s, \quad \text{and } NF_k(k_1 + k_2){=}NF_k(k_1)\hat{+}NF_k(k_2),$$

where $\hat{+}{:}(\hat{K}Stat{\cup}\{fail\})^2{\rightarrow}(\hat{K}Stat{\cup}\{fail\})$ *is defined as follows:*

$$fail\hat{+}fail{=}fail, \quad fail\hat{+}\hat{k}{=}\hat{k}\hat{+}fail{=}\hat{k}, \quad \text{and} \quad \hat{k}_1\hat{+}\hat{k}_2{=}\hat{k}_1{+}\hat{k}_2.$$

(c) *We also define* $NF_o : OStat{\rightarrow}(\tilde{C}Stat{\cup}\hat{K}Stat{\cup}\{fail\})$ *as follows:*

$$NF_o(\ll c \gg) = NF_c(c)$$

$$NF_o(\langle k \rangle) = NF_o(\#\langle k \rangle) = NF_k(k)$$

(d) *(Determinate statements)* $det : Stat{\rightarrow}Bool(= \{true, false\})$*:*

$$det(a) = det(fail) = det(\ll c \gg) = true$$

$$det(\langle k \rangle) = det(\#\langle k \rangle) = \text{if } length(NF_k(k)) \leq 1 \text{ then } true \text{ else } false \text{ fi}$$

$$det(x) = det(D(x))$$

$$det(s_1 \parallel s_2) = det(s_1) \vee det(s_2)$$

We proceed with the definition of a transition system for L_{BAM}. Following [11], we employ the CSC technique, and use *process identifiers* for the representation of resumptions and continuations.

Definition 3.3 *(Process identifiers) We use a set* $(\alpha \in)Id$ *of process identifiers - which we assume to be infinite - together with a function* $\nu_\alpha : \mathcal{P}_{finite}(Id){\rightarrow}Id$*, such that* $\nu_\alpha(A){\notin}A$*, for any* $A{\in}\mathcal{P}_{finite}(Id)$*. A possible example of such a set* Id *and function* ν_α *is* $Id{=}\mathbb{N}$ *and* $\nu_\alpha(A){=}max\{n \mid n{\in}A\} + 1$*. Let moreover* $\nu : \mathcal{P}(Id){\rightarrow}Id$ *be defined (for* $\overline{A}{\in}\mathcal{P}(Id)$*) by* $\nu(\overline{A}) =$ if $|\overline{A}|{<}\infty$ then $\nu_\alpha(\overline{A})$ else $\overline{\alpha}$ fi *, where* $\overline{\alpha}$ *is some arbitrary element of* Id*.*

Definition 3.4

(a) *(Resumptions) Let* $(r{\in})R = Id{\rightarrow}(\{\uparrow\}{\cup}Stat)$*, and* $id : R{\rightarrow}\mathcal{P}(Id)$*,* $id(r) = \{\alpha \mid r(\alpha) \neq \uparrow\}$*. The class* $(r \in)Res(\subseteq R)$ *of* resumptions *is given by:*

$$Res = \{r \mid r \in R, \mid id(r) \mid < \infty\}$$

Let $r_0 = \lambda\alpha.\uparrow$*. We define the predicate* $deadlock : Res{\rightarrow}Bool$ *as follows:*

$$deadlock(r) =$$

$$\text{if } r{=}\lambda\alpha.\uparrow \text{ then } false \text{ else if } (\exists \alpha \in id(r) : det(r(\alpha))) \text{ then } false \text{ else } true \text{ fi fi}$$

(b) (Nondeterministic alternatives) $(\eta\in)N = Res \cup (Stat\times Id\times Res)$. *We say that a nondeterministic alternative* η *is* derivable *if either* $\eta\in Res$ *or* $\eta=[s,\alpha,r](\in Stat\times Id\times Res)$ *and* $\alpha\notin id(r)$. *We note by* $(\eta\in)Ned$ *the class of* derivable nondeterministic alternatives.

(c) (Configurations) Consider the signature $Sig = \{+,\oplus\}$, *and let* $Sig_N = Sig \cup Ned$. *All the nondeterministic alternatives* $\eta(\in Ned)$ *have arity* 0 *in* Sig_N *(and hence are to be considered as constants in the extended signature). The key idea is that we consider an expression like* $\eta_1+\eta_2$ *(or* $\eta_1\oplus\eta_2$*) as a term - albeit a mixed one - in the sense that it consists of both a syntactic entity* $+$ *(or* $\oplus$*) and semantic entities* η_1,η_2. *Such* mixed terms *were first used in [8]. The set of non-terminal configurations is* $Conf = T(Sig_N)$, *where* $T(Sig_N)$ *denotes the set of (closed) terms generated by* Sig_N. *Let also* $\surd$ *be a special symbol* $(\surd\notin Conf)$ *that we use to indicate* termination *in computations. Let* t *range over* $(t\in)Conf_{\surd} = Conf\cup\{\surd\}$.

Intuitively, configurations are OR-trees. '+' corresponds to sequential OR, and '$\oplus$' corresponds to parallel OR. Each nondeterministic alternative $\eta\in Ned$ contains a finite collection of processes, corresponding to the so-called *process teams* in Andorra [5].

Definition 3.5 *(Transition labels) Let* $(\pi\in)ACT = Act^+$ *and* $(\varpi\in)ACT_\epsilon = Act^* = ACT\cup\{\epsilon\}$ *(*ϵ *is the empty sequence). We find it convenient to use the notation* $[a_1,...,a_n]$ *for sequences over* ACT. $[a]$*-steps are called deterministic steps.* $[a_1,...,a_n]$*-steps are called nondeterministic steps when* $n>1$.

The operational semantics for L_{BAM} is based on a transition relation $\subseteq (Conf\times ACT\times Conf)\cup(Conf\times\{\epsilon\}\times\{\surd\})$, with elements $[t,\varpi,t']$ written in the notation $t\xrightarrow{\varpi}t'$. We use the predicate $t\downarrow$ which is $true$ if $t\xrightarrow{\epsilon}\surd$ is an element of the transition relation and $false$ otherwise. Our restriction to the class of derivable nondeterministic alternatives in the definition of $Conf$ will be justified in lemma 3.10. In the definition of the transition relation for L_{BAM} we use the following conventions:

$$\frac{premise}{\begin{array}{c}conclusion_1\\ \ldots\\ conclusion_n\end{array}} \quad \text{is an abbreviation for} \quad \frac{premise}{conclusion_1} \quad \ldots \quad \frac{premise}{conclusion_n}$$

and

$$t_1\to t_2 \qquad \text{is an abbreviation for} \qquad \frac{t_2\xrightarrow{\varpi}t'}{t_1\xrightarrow{\varpi}t'}$$

Definition 3.6 *(Transition system for* L_{BAM}: T_{BAM}*) The transition relation for* L_{BAM} *is the smallest subset of* $(Conf\times ACT\times Conf)\cup(Conf\times\{\epsilon\}\times\{\surd\})$, *satisfying the axioms and rules below. In rules (R9) and (R10),* $\alpha'=\nu(id(r))$ *and* $\alpha''=\nu(id(r)\cup\{\alpha'\})$. *Also, in axiom (A6) we assume left associativity in the expression* $r\{s_1/\alpha\}+...+r\{s_n/\alpha\}$, *which thus denotes* $(...(r\{s_1/\alpha\}+r\{s_2/\alpha\})+...+r\{s_n/\alpha\})$. *Similarly, in (A7)* $r\{s_1/\alpha\}\oplus...\oplus r\{s_n/\alpha\}$ *denotes* $(...(r\{s_1/\alpha\}\oplus r\{s_2/\alpha\})\oplus...\oplus r\{s_n/\alpha\})$.

$$(A1) \quad [a,\alpha,r] \xrightarrow{[a]} r$$

$$(A2) \quad [fail,\alpha,r]\downarrow$$

$$(A3) \quad [o,\alpha,r]\downarrow \qquad \text{if } NF_o(o)=fail$$

$$(A4) \quad [o,\alpha,r] \xrightarrow{[a]} r\{s/\alpha\} \qquad \text{if } (NF_o(o){=}a{\rightarrow}s \ \vee \ NF_o(o){=}a?s)$$

$$(A5) \quad [\ll c \gg,\alpha,r] \xrightarrow{[a_i]} r\{s_i/\alpha\} \qquad \forall 1{\le}i{\le}n \qquad \text{if } (NF_c(c)=a_1{\rightarrow}s_1{+}...{+}a_n{\rightarrow}s_n, \ \ n>1)$$

$$(A6) \quad [\langle k\rangle,\alpha,r] \xrightarrow{\pi} r\{s_1/\alpha\}+...+r\{s_n/\alpha\} \qquad \text{if } (NF_k(k)=a_1?s_1{+}...{+}a_n?s_n, \ \ n>1)$$
$$\textit{where} \ \ \pi=[a_1,...,a_n]$$

$$(A7) \quad [\#\langle k\rangle,\alpha,r] \xrightarrow{\pi} r\{s_1/\alpha\}\oplus...\oplus r\{s_n/\alpha\} \qquad \text{if } (NF_k(k)=a_1?s_1{+}...{+}a_n?s_n, \ \ n>1)$$
$$\textit{where} \ \pi=[a_1,...,a_n]$$

$$(R8) \quad [x,\alpha,r]\rightarrow[D(x),\alpha,r]$$

$$(R9) \quad [s_1\|s_2,\alpha,r]\rightarrow[s_1,\alpha',r\{s_2/\alpha''\}] \qquad \text{if } \mathsf{not}(det(s_1\parallel s_2))\vee det(s_1)$$

$$(R10) \quad [s_1\|s_2,\alpha,r]\rightarrow[s_2,\alpha'',r\{s_1/\alpha'\}] \qquad \text{if } \mathsf{not}(det(s_1\parallel s_2))\vee det(s_2)$$

$$(A11) \quad r_0\downarrow$$

$$(R12) \quad r\rightarrow[r(\alpha),\alpha,r\{\uparrow/\alpha\}] \qquad \forall\alpha\in id(r) \qquad \text{if } deadlock(r)$$

$$(R13) \quad r\rightarrow[r(\alpha),\alpha,r\{\uparrow/\alpha\}] \qquad \forall\alpha\in id(r): det(r(\alpha)) \qquad \text{if } \mathsf{not}(deadlock(r))$$

$$(R14\text{-}16) \quad \frac{t_1\downarrow \qquad t_2\xrightarrow{\varpi}t_2'}{\begin{array}{c} t_1+t_2\xrightarrow{\varpi}t_2' \\ t_1\oplus t_2\xrightarrow{\varpi}t_2' \\ t_2\oplus t_1\xrightarrow{\varpi}t_2' \end{array}} \qquad (R17\text{-}19) \quad \frac{t_1\xrightarrow{\pi}t_1'}{\begin{array}{c} t_1+t_2\xrightarrow{\pi}t_1'+t_2 \\ t_1\oplus t_2\xrightarrow{\pi}t_1'\oplus t_2 \\ t_2\oplus t_1\xrightarrow{\pi}t_2\oplus t_1' \end{array}}$$

We offer some explanations.

- A configuration $[s,\alpha,r]$ contains an *active* process s with identifier α. The other processes are contained in the resumption r. Following the CSC technique [11], any process remains active only until it executes an atomic action. Subsequently, another process taken from the resumption is planned for execution. In this way it is obtained the interleaving behavior in the case of AND parallelism. Thus, in the modeling of AND parallelism we employ the CSC technique [11]. The sequential OR and the parallel OR operators are modeled using classic techniques (see the rules (R14-R19)).

- Axiom (A1) describes an elementary step. For simplicity, no formal distinction is made between *failure* and *successful termination.* Thus, if t fails or terminates successfully we put $t\downarrow$. In (A3), failure is produced by a construction $\ll c\gg$, $\langle k\rangle$ or $\#\langle k\rangle$.

- Axiom (A4) describes a *don't care* nondeterministic choice. An arbitrary non-failing alternative of a $\ll b_1 \to s_1 + ... + b_n \to s_n \gg$ construction is selected for execution; there is no *backtracking* in this case.
- Axioms (A6) and (A7) model nondeterministic promotion. In each case, the resumption r, is replicated for each nondeterministic alternative of a *don't know* nondeterministic choice statement, $\langle k \rangle$ or $\#\langle k \rangle$[1].
- The Andorra principle [13] gives priority to determinate goals over nondeterminate goals in parallel conjunctions. This priority mechanism is expressed in rules (R9), (R10), (R12) and (R13). In rule (R9), s_1 is selected for execution if it is determinate or if both s_1 and s_2 are nondeterminate. Rule (R10) is symmetric. Axiom (A11) models termination. Rule (R12) expresses the fact that in a *deadlock* state (when all the goals in a parallel conjunction are nondeterminate) any goal can be planned for execution. Rule (R13) gives priority to an (arbitrary) determinate goal.

In 3.7 we introduce some complexity measures which will be used in inductive reasonings. The mapping c_s is well defined due to our restriction to guarded recursion.

Definition 3.7 *(Complexity measures)* $c_s : Stat \to \mathbb{N}, c_\eta : Ned \to \mathbb{N}$ *and* $c_t : Conf \to \mathbb{N}$ *are defined as follows:*

$$\begin{array}{rcl}
c_s(a) & = & 1 \\
c_s(fail) & = & 1 \\
c_s(o) & = & 1 \\
c_s(x) & = & c_s(D(x)) \\
c_s(s_1 \parallel s_2) & = & 1 + max(c_s(s_1), c_s(s_2)) \\
c_\eta(r_0) & = & 1 \\
c_\eta(r(\neq r_0)) & = & 1 + max\{c_\eta([r(\alpha), \alpha, r\{\uparrow /\alpha\}]) \mid \alpha \in id(r)\} \\
c_\eta([s, \alpha, r]) & = & c_s(s) \\
c_t(\eta) & = & c_\eta(\eta) \\
c_t(t_1 + t_2) & = & 1 + c_t(t_1) + c_t(t_2) \\
c_t(t_1 \oplus t_2) & = & 1 + c_t(t_1) + c_t(t_2)
\end{array}$$

Definition 3.8 *(Operational semantics for L_{BAM})*

(1) Let $(p \in)\mathbb{P} = \mathcal{P}_{\text{nco}}(ACT^\infty)$, $(S \in)Sem_O = Conf \to \mathbb{P}$. *We define the mapping* $\Phi : Sem_O \to Sem_O$ *by:*

$$\Phi(S)(t) = \{\epsilon \mid t \downarrow\} \cup \bigcup \{\pi.S(t') \mid t \xrightarrow{\pi} t'\}$$

(2) We put $\mathcal{O} = fix(\Phi)$ *and define* $\mathcal{O}[\![\cdot]\!] : Stat \to \mathbb{P}$ *as follows:*

$$\mathcal{O}[\![s]\!] = \mathcal{O}([s, \nu(\emptyset), r_0]).$$

[1] Left associativity is assumed in the expressions occurring in (A6) and (A7). However, with the introduction of the denotational semantics for L_{BAM} - which equals the operational semantics - we will see that the order does not matter. The point is that the nondeterministic alternatives are executed either in sequence or in parallel and the both operations are associative.

Remarks 3.9

(1) One can prove that T_{BAM} *is* finitely branching *and thus it induces a compact operational semantics by induction on* $c_t(t)$*.*

(2) The mapping Φ *is contracting in particular due to the "*$\pi.\cdots$*"-step in its definition.*

(3) Lemma 3.10(2) can be proved by induction on $c_t(t_1)$*. 3.10(1) is immediate, and 3.10(3) follows easily from the rules of* T_{BAM}*.*

Lemma 3.10

(1) $[s, \nu(\emptyset), r_0] \in Conf$*.*

(2) If $t_1 \in Conf$ *and* $t_1 \stackrel{\varpi}{\to} t_2$ *then* $\varpi{=}\epsilon$ *and* $t_2{=}\surd$ *or* $\varpi{\in}ACT$ *and* $t_2{\in}Conf$*.*

(3) If $t_1 \in Conf$ *and* $t_1 \to t_2$ *then* $t_2 \in Conf$*.*

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

Following [12], in the definition of the denotational semantics for L_{BAM} we employ the CSC technique [11]. We use classic techniques in the modeling of OR parallelism; the CSC technique is employed in the semantic modeling of AND parallelism.

In the definition of the denotational semantics $\mathcal{D}$ for L_{BAM} we use the same semantic universe as in the case of operational semantics: $(p \in)\mathbb{P} = \mathcal{P}_{\rm nco}(ACT^{\infty})$; we also use the typical variable q to range over $(q \in)ACT^{\infty}$. The operator $\uplus$ (introduced in 4.1(3)) is used in the semantic modeling of the Andorra priority mechanism. $\uplus$ gives priority to determinate steps of the form $[a]$ over nondeterminate steps of the form $[a_1, ..., a_n]$ $(n > 1)$.

Definition 4.1

(1) $Det, NDet : \mathbb{P} \to \mathbb{P}$ *are given by (obviously,* $p = Det(p) \cup NDet(p)$*):*

$$Det(p) = \{\epsilon \mid \epsilon \in p\} \cup \{[a].q \mid [a].q \in p\}$$

$$NDet(p) = \{[a_1, ..., a_n].q \mid [a_1, ..., a_n].q \in p, \ \ n > 1\}$$

(2) $DET, NDET : \mathbb{P} \to Bool(= \{true, false\})$ *are given by:*

$$DET(p) = \textsf{if } (p = Det(p)) \textsf{ then } true \textsf{ else } false \textsf{ fi}$$

$$NDET(p) = \textsf{if } (p = NDet(p)) \textsf{ then } true \textsf{ else } false \textsf{ fi}$$

(3) $\uplus : \mathbb{P} \times \mathbb{P} \to \mathbb{P}$ *is defined by:*

$$p_1 \uplus p_2 = \textsf{if } (NDET(p_1) \wedge NDET(p_2)) \textsf{ then } (p_1 \cup p_2) \textsf{ else } (Det(p_1) \cup Det(p_2)) \textsf{ fi}$$

It is easy to check that $\uplus$ is well defined, non-expansive, associative and commutative. To model the OR connectives in L_{BAM} we use the operators $+$ and $\oplus$ defined below. $+$ is an operator for sequential composition. $\oplus$ is an operator for parallel composition in interleaving semantics (i.e. a *merge* operator); we also use the operator $\lfloor\!\lfloor$ (a *left merge*). Such operators can be formally defined as fixed points of appropriate higher order contractions using classic techniques. One can show that $+$ and $\oplus$ are well defined, non-expansive and associative; $\oplus$ is also commutative. $\lfloor\!\lfloor$ is well defined and non-expansive.

Definition 4.2 *The operators* $+, \oplus, \lfloor\!\lfloor : \mathbb{P} \times \mathbb{P} \xrightarrow{1} \mathbb{P}$ *are defined as follows:*

$$p+p' = \{p' \mid \epsilon \in p\} \cup \bigcup\{\pi.(p_\pi+p') \mid p_\pi \neq \emptyset\}$$

$$p\oplus p' = p\lfloor\!\lfloor\, p' \cup p'\lfloor\!\lfloor\, p, \quad \textit{with} \qquad p\lfloor\!\lfloor\, p' = \{p' \mid \epsilon \in p\} \cup \bigcup\{\pi.(p_\pi\oplus p') \mid p_\pi \neq \emptyset\},$$

where we used the notation: $p_\pi \stackrel{def.}{=} \{q \mid \pi.q \in p\}$ *for* $\pi \in ACT$, $p \in \mathbb{P}$ *and* $q \in ACT^\infty$.

The denotational semantics $\mathcal{D}$ for L_{BAM} is of the type $Sem_D{=}Stat \to \mathbb{D}$, where:

$$\mathbb{D} \cong Id \to Cont \xrightarrow{1} \mathbb{P}$$

$$(\gamma \in)Cont \cong Id \to (\{\uparrow\} \sqcup \tfrac{1}{2}\cdot\mathbb{D}) \qquad (Cont \text{ is the domain of } \textit{continuations})$$

In the equations above the sets Id and $\{\uparrow\}$ are equipped with the discrete metric, which is an ultrametric. The other metric spaces are built up using the composite metrics given in 2.2. By using [1, Balaganskij-Vlasov(1996)], the solutions for $\mathbb{D}$ and $Cont$ can be obtained as complete ultrametric spaces.

In 4.3 we introduce the auxiliary mapping Id used in the definition of the denotational semantics, which is given in 4.5. Lemma 4.4 states two simple properties of Id.

Definition 4.3 $Id : Cont \to \mathcal{P}(Id)$ *is defined as follows* $Id(\gamma) = \{\alpha \mid \gamma(\alpha) \neq \uparrow\}$.

Lemma 4.4

(a) If $Id(\gamma_1) \neq Id(\gamma_2)$ *then* $d(\gamma_1, \gamma_2) = 1$.

(b) If $A = Id(\gamma_1) = Id(\gamma_2)$ *then* $d(\gamma_1, \gamma_2) = \frac{1}{2}\cdot sup_{\alpha\in A} d(\gamma_1(\alpha), \gamma_2(\alpha))$.

Definition 4.5 *(Denotational semantics ($\mathcal{D}$) for L_{BAM})*

(1) Let $C : Cont \to \mathbb{P}$ *be given by:*

$$C(\gamma) = \textsf{if } (|Id(\gamma)|{=}0 \vee |Id(\gamma)|{=}\infty) \textsf{ then } \{\epsilon\} \textsf{ else } \uplus_{\alpha\in Id(\gamma)}\, \gamma(\alpha)(\alpha)(\gamma\{\uparrow/\alpha\}) \textsf{ fi}$$

(2) We define $\Psi : Sem_D \to Sem_D$ *for* $S \in Sem_D$ *as follows:*

(1) $\Psi(S)(a)(\alpha)(\gamma) = [a].C(\gamma)$

(2) $\Psi(S)(fail)(\alpha)(\gamma) = \{\epsilon\}$

(3) $\Psi(S)(o)(\alpha)(\gamma) = \{\epsilon\} \qquad \textsf{if } NF_o(o) = fail$

(4) $\Psi(S)(o)(\alpha)(\gamma) = [a].C(\gamma\{S(s)/\alpha\}) \qquad \textsf{if } NF_o(o) = a{\to}s \vee NF_o(o) = a?s$

(5) $\Psi(S)(\ll c \gg)(\alpha)(\gamma) = [a_1].C(\gamma\{S(s_1)/\alpha\}) \cup ... \cup [a_n].C(\gamma\{S(s_n)/\alpha\})$
$\textsf{if } NF_c(c) = a_1{\to}s_1{+}...{+}a_n{\to}s_n, \quad n > 1$

(6) $\Psi(S)(\langle k\rangle)(\alpha)(\gamma) = [a_1, ..., a_n].(C(\gamma\{S(s_1)/\alpha\}){+}...{+}C(\gamma\{S(s_n)/\alpha\}))$
$\textsf{if } NF_k(k) = a_1?s_1{+}...{+}a_n?s_n, \quad n > 1$

(7) $\Psi(S)(\#\langle k\rangle)(\alpha)(\gamma) = [a_1, ..., a_n].(C(\gamma\{S(s_1)/\alpha\}){\oplus}...{\oplus}C(\gamma\{S(s_n)/\alpha\}))$
$\textsf{if } NF_k(k) = a_1?s_1{+}...{+}a_n?s_n, \quad n > 1$

(8) $\Psi(S)(x)(\alpha)(\gamma) = \Psi(S)(D(x))(\alpha)(\gamma)$

(9) $\Psi(S)(s_1\|s_2)(\alpha)(\gamma) = \Psi(S)(s_1)(\alpha')(\gamma\{S(s_2)/\alpha''\}) \uplus \Psi(S)(s_2)(\alpha'')(\gamma\{S(s_1)/\alpha'\})$

where in clause (9) $\alpha'=\nu(Id(\gamma))$ *and* $\alpha''=\nu(Id(\gamma)\cup\{\alpha'\})$.

(3) We put $\mathcal{D}=fix(\Psi)$. *Let* $\gamma_0=\lambda\alpha.\uparrow$. *We define* $\mathcal{D}[\![\cdot]\!]:Stat\to\mathbb{P}$ *by:*

$$\mathcal{D}[\![s]\!]=\mathcal{D}(s)(\nu(\emptyset))(\gamma_0).$$

Some explanations may help.

- The denotational semantics uses the CSC technique to model the AND parallelism. The OR parallelism is modeled by using the $\oplus$ operator introduced in 4.2. The operator $+$ models the backtracking mechanism in L_{BAM}. We model *failure* by an ϵ-step because we do not want to interrupt the collection of multiple solutions generated by the *don't know* nondeterminism in L_{BAM}.
- Clauses (1-4) handle determinate goals. Clause (5) models the *don't care* nondeterminism in L_{BAM}. Clauses (6) and (7), model the semantics of *don't know* goals. Such goals give rise to *nondeterministic promotion*, which is modeled denotationally by making copies of the continuation for each alternative of a nondeterminate goal. Next, these alternatives are executed either in sequence (clause (6)) or in parallel (clause (7)).
- The operator $\uplus$ gives priority to determinate steps of the form $[a]$ over nondeterminate steps of the form $[a_1,...,a_n]$ $(n>1)$ in a parallel conjunction.

Definition 4.5 is formally justified by using lemmas 4.6 and 4.7. We do not give here the proofs for lemmas 4.6 and 4.7, which are very similar to the proofs for corresponding lemmas given for the denotational models studied in [11]. By using 4.7(3), the higher order mapping Ψ introduced in 4.5 has a unique fixed point.

Lemma 4.6

(1) $C:Cont\to\mathbb{P}$ *is well defined (i.e.* $\forall\gamma\in Cont$, $C(\gamma)$ *is compact).*

(2) $\forall\gamma_1\gamma_2\in Cont: d(C(\gamma_1),C(\gamma_2))\le 2\cdot d(\gamma_1,\gamma_2)$.

Lemma 4.7 *For any* $S\in Sem_D, s\in Stat,\alpha\in Id$ *and* $\gamma\in Cont$ *we have:*

(1) $\Psi(S)(s)(\alpha)(\gamma)\in\mathbb{P}$ *(it is well defined).*

(2) $\Psi(S)(s)(\alpha)$ *is non-expansive in* γ.

(3) Ψ *is* $\frac{1}{2}$*-contractive in* S.

Example 4.8 *Let* $\alpha_0=\nu(\emptyset)$ *and* $\alpha_i=\nu(\{\alpha_j\mid 0\le j<i\})$, *for* $i>0$. *Let also* $(D\in Decl)$ $D(x)=\langle a_3?a_3'+a_4?a_4'\rangle\|a_2$. *We compute the denotation of* $a_1\|x$.

$$\mathcal{D}[\![a_1\|x]\!]=\mathcal{D}(a_1\|x)(\alpha_0)(\gamma_0)=\mathcal{D}(a_1)(\alpha_1)(\gamma_0\{\mathcal{D}(x)/\alpha_2\})\uplus\mathcal{D}(x)(\alpha_2)(\gamma_0\{\mathcal{D}(a_1)/\alpha_1\})$$

We begin with the first sub-expression.

$$\mathcal{D}(a_1)(\alpha_1)(\gamma_0\{\mathcal{D}(x)/\alpha_2\})=[a_1].C(\gamma_0\{\mathcal{D}(x)/\alpha_2\})=[a_1].\mathcal{D}(x)(\alpha_2)(\gamma_0)$$

$$=[a_1].(\mathcal{D}(\langle a_3?a_3'+a_4?a_4'\rangle)(\alpha_3)(\gamma_0\{\mathcal{D}(a_2)/\alpha_4\})\uplus\mathcal{D}(a_2)(\alpha_4)(\gamma_0\{\mathcal{D}(\langle a_3?a_3'+a_4?a_4'\rangle)/\alpha_3\})$$

It is easy to check that $\mathcal{D}(\langle a_3?a_3'+a_4?a_4'\rangle)(\alpha_3)(\gamma_0\{\mathcal{D}(a_2)/\alpha_4\}) = [a_3,a_4].p$ *(for* $p \in \mathbb{P}$*). It is not necessary to compute* p *because the process* $[a_3,a_4].p$ *is removed by the operator* $\uplus$ *from the result of the denotational semantics. We compute the other operand of* $\uplus$*.*

$$\mathcal{D}(a_2)(\alpha_4)(\gamma_0\{\mathcal{D}(\langle a_3?a_3'+a_4?a_4'\rangle)/\alpha_3\}) = [a_2].C(\gamma_0\{\mathcal{D}(\langle a_3?a_3'+a_4?a_4'\rangle)/\alpha_3\})$$

$$= [a_2].\mathcal{D}(\langle a_3?a_3'+a_4?a_4'\rangle)(\alpha_3)(\gamma_0) = [a_2].\{[a_3,a_4][a_3'][a_4']\} = \{[a_2][a_3,a_4][a_3'][a_4']\}$$

Therefore we have:

$$\mathcal{D}(a_1)(\alpha_1)(\gamma_0\{\mathcal{D}(x)/\alpha_2\}) = [a_1].([a_3,a_4].p\uplus[a_2][a_3,a_4][a_3'][a_4']) = \{[a_1][a_2][a_3,a_4][a_3'][a_4']\}$$

For the other sub-expression we get: $\mathcal{D}(x)(\alpha_2)(\gamma_0\{\mathcal{D}(a_1)/\alpha_1\}) = \{[a_2][a_1][a_3,a_4][a_3'][a_4']\}$.

Finally, we get: $\mathcal{D}[\![a_1\|x]\!] = \{[a_1][a_2][a_3,a_4][a_3'][a_4'],[a_2][a_1][a_3,a_4][a_3'][a_4']\}$.

The denotation of the nondeterminate goal is:

$$\mathcal{D}[\![\langle a_3?a_3'+a_4?a_4'\rangle]\!]=\mathcal{D}(\langle a_3?a_3'+a_4?a_4'\rangle)(\alpha_0)(\gamma_0) = \{[a_3,a_4][a_3'][a_4']\}.$$

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

In this section we show that $\forall s \in \mathit{Stat} : \mathcal{O}[\![s]\!]=\mathcal{D}[\![s]\!]$. In 5.1 we introduce an auxiliary mapping $\mathcal{R} : \mathit{Conf} \to \mathbf{P}$ and we show that $\mathcal{O}=\mathcal{R}$ (lemma 5.5) by using Banach's fixed point theorem 2.1. Lemmas 5.3 and 5.4 are useful in the proof of lemma 5.5. The desired result ($\mathcal{O}[\![s]\!]=\mathcal{D}[\![s]\!]$) is obtained in 5.6.

Definition 5.1 *Let* $\Gamma : \mathit{Res} \to \mathit{Cont}$ *be given by:*

$$\Gamma(r) = \lambda\alpha.\,\mathsf{if}\ r(\alpha) = \uparrow\ \mathsf{then}\ \uparrow\ \mathsf{else}\ \mathcal{D}(r(\alpha))\ \mathsf{fi}$$

We define $\mathcal{R} : \mathit{Conf} \to \mathbf{P}$ *as follows:*

$$\mathcal{R}(r) = C(\Gamma(r)) \qquad \mathcal{R}([s,\alpha,r]) = \mathcal{D}(s)(\alpha)(\Gamma(r))$$

$$\mathcal{R}(t_1+t_2) = \mathcal{R}(t_1)+\mathcal{R}(t_2) \qquad \mathcal{R}(t_1\oplus t_2) = \mathcal{R}(t_1)\oplus\mathcal{R}(t_2)$$

Remark 5.2 *The operators* $+$ *and* $\oplus$ *(introduced in section 4) are associative. We have:*

$$\mathcal{R}(t_1+(t_2+t_3)) = \mathcal{R}(t_1)+\mathcal{R}(t_2+t_3) = \mathcal{R}(t_1)+(\mathcal{R}(t_2)+\mathcal{R}(t_3))$$

$$= (\mathcal{R}(t_1)+\mathcal{R}(t_2))+\mathcal{R}(t_3) = \mathcal{R}(t_1+t_2)+\mathcal{R}(t_3) = \mathcal{R}((t_1+t_2)+t_3)$$

Thus, in order to simplify the notation in 5.3(8) we will write $\mathcal{R}(t_1+\ldots+t_n)$ *instead of* $\mathcal{R}((\ldots(t_1+t_2)+\ldots t_n))$*. Similarly, in 5.3(9) we will write* $\mathcal{R}(t_1\oplus\ldots\oplus t_n)$ *instead of* $\mathcal{R}((\ldots(t_1\oplus t_2)\oplus\ldots\oplus t_n))$*. Similar notations will be used in 5.5. The notation conventions used here and in 3.6 are formally justified in 5.5, where we show that* $\mathcal{R}=\mathcal{O}$*.*

Lemma 5.3

(1) $\mathcal{R}(r_0) = \{\epsilon\}$.

(2) If $r \neq r_0$ *then* $\mathcal{R}(r) = \uplus_{\alpha\in id(r)}\mathcal{R}([r(\alpha),\alpha,r\{\uparrow/\alpha\}])$.

(3) $\mathcal{R}([a,\alpha,r]) = [a].\mathcal{R}(r)$.

(4) $\mathcal{R}([fail, \alpha, r]) = \{\epsilon\}$.

(5) *If* $NF_o(o) = fail$ *then* $\mathcal{R}([o, \alpha, r]) = \{\epsilon\}$.

(6) *If* $NF_o(o) = a{\rightarrow}s$ *or* $NF_o(o) = a?s$ *then* $\mathcal{R}([o, \alpha, r]) = [a].\mathcal{R}(r\{s/\alpha\})$.

(7) *If* $NF_c(c) = a_1{\rightarrow}s_1 + ... + a_n{\rightarrow}s_n$ *and* $n > 1$ *then:*

$$\mathcal{R}([\ll c \gg, \alpha, r]) = [a_1].\mathcal{R}(r\{s_1/\alpha\}) \cup ... \cup [a_n].\mathcal{R}(r\{s_n/\alpha\}).$$

(8) *If* $NF_k(k) = a_1?s_1 + ... + a_n?s_n$ *and* $n > 1$ *then:*

$$\mathcal{R}([\langle k \rangle, \alpha, r]) = [a_1, ..., a_n].\mathcal{R}(r\{s_1/\alpha\} + ... + r\{s_n/\alpha\}).$$

(9) *If* $NF_k(k) = a_1?s_1 + ... + a_n?s_n$ *and* $n > 1$ *then:*

$$\mathcal{R}([\#\langle k \rangle, \alpha, r]) = [a_1, ..., a_n].\mathcal{R}(r\{s_1/\alpha\} \oplus ... \oplus r\{s_n/\alpha\}).$$

(10) $\mathcal{R}([x, \alpha, r]) = \mathcal{R}([D(x), \alpha, r])$.

(11) *Let* $\alpha' = \nu(id(r)), \alpha'' = \nu(id(r) \cup \{\alpha'\})$. *We have:*

$$\mathcal{R}([s_1 \| s_2, \alpha, r]) = \mathcal{R}([s_1, \alpha', r\{s_2/\alpha''\}]) \uplus \mathcal{R}([s_2, \alpha'', r\{s_1/\alpha'\}])$$

Lemma 5.3 can be proved without difficulty by applying the definition of $\mathcal{R}$.

Lemma 5.4

(1) *If* $t \downarrow$ *then* $\epsilon \in \mathcal{R}(t)$.

(2) $\forall \alpha \in Id, \forall r \in Res, \alpha \notin id(r) : det(s) = DET(\mathcal{R}([s, \alpha, r]))$.

(3) $\forall \alpha \in Id, \forall r \in Res, \alpha \notin id(r) : \mathsf{not}(det(s)) = NDET(\mathcal{R}([s, \alpha, r]))$.

(4) *If* $deadlock(r)$ *then* $\mathcal{R}(r) = \cup_{\alpha \in id(r)} \mathcal{R}([r(\alpha), \alpha, r\{\uparrow/\alpha\}])$.

(5) *If* $r \neq r_0$ *and* $\mathsf{not}(deadlock(r))$ *then* $\mathcal{R}(r) = \cup_{\alpha \in id(r): det(r(\alpha))} \mathcal{R}([r(\alpha), \alpha, r\{\uparrow/\alpha\}])$.

(6) $\mathsf{not}(deadlock(r)) = DET(\mathcal{R}(r))$.

(7) $deadlock(r) = NDET(\mathcal{R}(r))$.

(8) $\mathcal{R}([s_1 \parallel s_2, \alpha, r]) =$

$$(\,\mathsf{if}\ (\,\mathsf{not}(det(s_1 \parallel s_2)) \vee det(s_1))\ \mathsf{then}\ \mathcal{R}([s_1, \alpha', r\{s_2/\alpha''\}])\ \mathsf{else}\ \emptyset\ \mathsf{fi}\,) \cup$$
$$(\,\mathsf{if}\ (\,\mathsf{not}(det(s_1 \parallel s_2)) \vee det(s_2))\ \mathsf{then}\ \mathcal{R}([s_2, \alpha'', r\{s_1/\alpha'\}])\ \mathsf{else}\ \emptyset\ \mathsf{fi}\,)$$

where $\alpha' = \nu(id(r))$ *and* $\alpha'' = \nu(id(r) \cup \{\alpha'\})$.

Proof We only treat 5.4(2) and 5.4(3). We prove 5.4(2) and 5.4(3) together with:

$$\forall s \in Stat, \alpha \in Id, r \in Res\ [DET(\mathcal{R}([s, \alpha, r])) \vee NDET(\mathcal{R}([s, \alpha, r])) = true]^{(*)}$$

by simultaneous induction on $c_s(s)$. We consider two basic sub-cases, for which $c_s(s) = 1$.

Case $s \equiv a$. $det(a) = true$, $DET(\mathcal{R}([a, \alpha, r])) =$ [5.3(3)] $DET([a].\mathcal{R}(r)) = true$, and $NDET(\mathcal{R}([a, \alpha, r])) = false$.

Case $s \equiv \langle k \rangle$, when $NF_k(k) = a_1?s_1 + ... + a_n?s_n$ $(n > 1)$. In this case we have $det(\langle k \rangle) = false$, $DET(\mathcal{R}(\langle k \rangle))$ [5.3(8)] $= DET([a_1, ..., a_n].\mathcal{R}(r\{s_1/\alpha\} + ... + r\{s_n/\alpha\})) = false$, and $NDET(\mathcal{R}([\langle k \rangle, \alpha, r])) = true$.

We also consider one sub-case when $c_s(s) > 1$.

Case $s \equiv x$. We have $det(x){=}det(D(x)) \overset{ind.}{=} DET(\mathcal{R}([D(x), \alpha, r])){=}DET(\mathcal{R}([x, \alpha, r]))$, and $\mathsf{not}(det(x)) = \mathsf{not}(det(D(x))) \overset{ind.}{=} NDET(\mathcal{R}([D(x), \alpha, r])) = NDET(\mathcal{R}([x, \alpha, r]))$. Also, by using the property 5.3(10) we get $DET(\mathcal{R}([x, \alpha, r])) \vee NDET(\mathcal{R}([x, \alpha, r]))$ $= DET(\mathcal{R}([D(x), \alpha, r])) \vee NDET(\mathcal{R}([D(x), \alpha, r])) \overset{ind.}{=} true$.

□

From 5.4(2, 3, 6, 7) we infer that $\forall t \in Conf$ either $DET(\mathcal{R}(t))$ or $NDET(\mathcal{R}(t))$, i.e. we either have $\mathcal{R}(t) = Det(\mathcal{R}(t))$ or we have $\mathcal{R}(t) = NDet(\mathcal{R}(t))$. For L_{BAM}, this expresses the natural property that a process can not execute alternatively deterministic and nondeterministic actions. Any process, can perform nondeterministic steps only in the deadlock state, in which it can not execute any deterministic action. In an ordinary state (i.e. in a non-deadlock state) any process can only execute deterministic steps. All these are a natural consequence of the fact that in L_{BAM} deterministic steps are given priority over nondeterministic steps.

Lemma 5.5 $\mathcal{R} = fix(\Phi)$ *(with Φ defined in 3.8). By using 2.1 we infer $\mathcal{R} = \mathcal{O}$.*

Proof We show that $\mathcal{R}(t) = \Phi(\mathcal{R})(t)$, $\forall t \in Conf$ by induction on $c_t(t)$. We treat three sub-cases.

Case $t = r_0$. $\Phi(\mathcal{R})(r_0) = \{\epsilon\} = [5.3(1)]\ \mathcal{R}(r_0)$

Case $t = [\#\langle k \rangle, \alpha, r]$, when $NF_k(k) = a_1?s_1 + ... + a_n?s_n$ and $n > 1$.

$$\Phi(\mathcal{R})([\#\langle k \rangle, \alpha, r]) = [\text{def. } \Phi] \quad [a_1, ..., a_n].\mathcal{R}(r\{s_1/\alpha\} \oplus ... \oplus r\{s_n/\alpha\})$$
$$= [5.3(9)]\ \mathcal{R}([\#\langle k \rangle, \alpha, r])$$

Case $t = [s_1 \parallel s_2, \alpha, r]$. Let $t_1 = [s_1, \alpha', r\{s_2/\alpha''\}]$ and $t_2 = [s_2, \alpha'', r\{s_1/\alpha'\}]$.

$$\begin{aligned}
&\Phi(\mathcal{R})([s_1 \parallel s_2, \alpha, r]) \qquad [\text{def. } \Phi]\\
&= (\,\mathsf{if}\ (\,\mathsf{not}(det(s_1 \parallel s_2)) \vee det(s_1))\ \mathsf{then}\ \{\epsilon \mid t_1\downarrow\} \cup \bigcup\{\pi.\mathcal{R}(t') \mid t_1 \xrightarrow{\pi} t'\}\ \mathsf{else}\ \emptyset\ \mathsf{fi}\,) \cup\\
&\quad (\,\mathsf{if}\ (\,\mathsf{not}(det(s_1 \parallel s_2)) \vee det(s_2))\ \mathsf{then}\ \{\epsilon \mid t_2\downarrow\} \cup \bigcup\{\pi.\mathcal{R}(t') \mid t_2 \xrightarrow{\pi} t'\}\ \mathsf{else}\ \emptyset\ \mathsf{fi}\,)\\
&= (\,\mathsf{if}\ (\,\mathsf{not}(det(s_1 \parallel s_2)) \vee det(s_1))\ \mathsf{then}\ \Phi(\mathcal{R})(t_1)\ \mathsf{else}\ \emptyset\ \mathsf{fi}\,) \cup\\
&\quad (\,\mathsf{if}\ (\,\mathsf{not}(det(s_1 \parallel s_2)) \vee det(s_2))\ \mathsf{then}\ \Phi(\mathcal{R})(t_2)\ \mathsf{else}\ \emptyset\ \mathsf{fi}\,) \qquad [\text{ind.}]\\
&= (\,\mathsf{if}\ (\,\mathsf{not}(det(s_1 \parallel s_2)) \vee det(s_1))\ \mathsf{then}\ \mathcal{R}(t_1)\ \mathsf{else}\ \emptyset\ \mathsf{fi}\,) \cup\\
&\quad (\,\mathsf{if}\ (\,\mathsf{not}(det(s_1 \parallel s_2)) \vee det(s_2))\ \mathsf{then}\ \mathcal{R}(t_2)\ \mathsf{else}\ \emptyset\ \mathsf{fi}\,) \qquad [5.4(8)]\\
&= \mathcal{R}([s_1 \parallel s_2, \alpha, r])
\end{aligned}$$

□

By using 5.5 we obtain the main result of the paper.

Theorem 5.6 $\mathcal{O}[\![s]\!] = \mathcal{D}[\![s]\!], \forall s \in Stat.$

Proof $\mathcal{O}[\![s]\!] = \mathcal{O}([s, \nu(\emptyset), r_0]) = [5.5]\ \mathcal{R}([s, \nu(\emptyset), r_0]) = \mathcal{D}(s)(\nu(\emptyset))(\gamma_0) = \mathcal{D}[\![s]\!]$ □

6 Concluding remarks and future work

In this paper we applied the CSC technique [11] in the semantic modeling of a simple abstract language L_{BAM} embodying the main control flow notions of Warren's Basic Andorra Model [13]. By using techniques from metric semantics [Balaganskij-Vlasov(1996)], we defined and related an operational and a denotational semantics for L_{BAM}.

The semantic framework presented in this paper is very flexible, allowing for further refinements. In the near future we are mainly interested in the application of the CSC technique in the specification and design of concurrent constraint (logic) programming languages [9]. We also plan to apply the CSC technique to parallel logic programming languages with deep guards. In doing so, we intend to move from the Basic Andorra Model (which is based on flat guards) to the Extended Andorra Model [14], which has been implemented in languages (that incorporate the constraint programming paradigm) like AKL [6] and Oz [10].

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*, 39:343-375, 1989.

[2] R. Bahgat. *Pandora: Non-Deterministic Parallel Logic Programming.* World Scientific Publishing Co., 1993.

[3] S. Banach. Sur les operations dans les ensambles abstraits et leurs applications aux equations integrales. *Fundamenta Mathematicae*, 3:133-181, 1922.

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

[5] V.S.Costa, D.H.D. Warren and R. Yang. Andorra-I: a parallel Prolog system that transparently exploits both AND- and OR-parallelism. In *Proc. 3rd ACM-SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ACM Press, 1991.

[6] S. Haridi, S. Janson. Kernel Andorra Prolog and its computation model. In *Proc. of International Conference on Logic Programming (ICLP'90)*, MIT Press, 1990.

[7] 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/`.

[8] J.J.M.M. Rutten. Processes as terms: non-well-founded models for bisimulation. *Mathematical Structures in Computer Science* 2:257-275, 1992.

[9] V.A. Saraswat. *Concurrent constraint programming.* MIT Press, 1993.

[10] G. Smolka. The Oz programming model. In *Computer Science Today*, volume 1000 of *LNCS* pages 324–343, Springer, 1995.

[11] E. Todoran. Metric semantics for synchronous and asynchronous communication: a continuation-based approach. In *Proc. of FCT'99 Workshop on Distributed Systems, Electronic Notes in Theoretical Computer Science (ENTCS)*, vol. 28, pages 119–146, Elsevier, 2000.

[12] E. Todoran and N. Papaspyrou. Continuations for parallel logic programming, In *Proc. of 2nd International ACM-SIGPLAN Conference on Principles and practice of Declarative Programming (PPDP'00)*, pages 257–267, ACM Press, 2000.

[13] D.H.D. Warren. The Andorra principle. Talk given at Gigalips Workshop, SICS, Sweden, 1988.

[14] D.H.D. Warren. Extended Andorra Model with implicit control. Talk given at Logic Programming Symposium, Eilat, Israel, 1990.

Eneia Todoran
Technical University of Cluj-Napoca
Faculty of Automation and Computer Science
Department of Computer Science
Baritiu Str. 28, Cluj-Napoca
ROMANIA
Eneia.Todoran@cs.utcluj.ro

Paulina Mitrea
Technical University of Cluj-Napoca
Faculty of Automation and Computer Science
Department of Computer Science
Baritiu Str. 28, Cluj-Napoca
ROMANIA
Paulina.Mitrea@cs.utcluj.ro

Nikolaos Papaspyrou
National Technical University of Athens
Department of Electrical and Computer Engineering
Software Engineering Laboratory
Polytechnioupoli, 15780 Zografou, Athens
GREECE
nickie@softlab.ntua.gr