Nonwellfounded Sets
and
Programming Language Semantics

J.J.M.M. Rutten

Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

ABSTRACT: For a large class of transition systems that are defined by specifications in the SOS style, it is shown how these induce a compositional semantics. The main difference with earlier work on this subject is the use of a nonstandard set theory that is based on Aczel's anti-foundation-axiom. Solving recursive domain equations in this theory leads to solutions that contain nonwellfounded elements. These are particularly useful for justifying recursive definitions, both of semantic operators and semantic models. The use of nonwellfounded sets further allows for the construction of compositional models for a larger class of transition systems than in the setting of complete metric spaces, which was used before.

1980 Mathematics Subject Classification: 68B10, 68C01.
1986 Computing Reviews Categories: D.3.1, F.3.2, F.3.3.
Key words and phrases: Structured operational semantics, labelled transition system, transition system specification, bisimulation, interpretation, compositionality, nonwellfounded sets, anti-foundation-axiom.
Note: This work was partially carried out in the context of ESPRIT Basic Research Action 3020 (Integration).

CONTENTS

1. INTRODUCTION

As a starting point for the semantics of programming languages we take the notion of *labelled transition system* (LTS) in the SOS style of Plotkin ([Pl81]). A LTS is a triple $<S, A, \rightarrow>$ of a set S of states, a set A of transition labels, and a transition relation $\rightarrow \subseteq S \times A \times S$. Every LTS induces a (strong) *bisimulation* equivalence on the set of states. (See [Pa81].) In this paper, it is shown how to derive from certain *transition system specifications*, used for defining LTS's, a denotational semantics that characterizes bisimulation in the sense that it assigns the same meaning to bisimilar states. The main difference with our previous work on this subject ([Ru90]) is the use of so-called *nonwellfounded* sets as a semantic universe. This leads to two considerable improvements: first, the semantic descriptions are more transparent (e.g., in that their well-definedness is simpler to verify); secondly, the class of LTS's that can be supplied with a denotational semantics is in an essential way more general.

The basic idea is the definition of a semantics $\mathfrak{M}$ that assigns to each state its unfolding under the transition relation. These unfoldings are represented as elements of a class P of commutative, tree-like structures called processes, satisfying

$$P = \mathfrak{P}(A \times P)$$

An immediate consequence of the definition of $\mathfrak{M}$ and the representation of unfoldings of states as elements of P, is the fact that whenever two states are bisimilar, they are assigned by $\mathfrak{M}$ to the same element in P. In other words, for every state $s \in S$ the process $\mathfrak{M}[\![s]\!]$ can be seen as a canonical representation of the bisimulation equivalence class of s.

As opposed to [Ru90], where the above equation was solved in a category of complete metric spaces (following [BZ82] and [AR89]), P is here formally defined in a non-standard set theory. It is based on the usual set-theoretic axioms but for the axiom of foundation, which is replaced by a strong version of its negation, the *anti-foundation-axiom* (AFA). Thus we work in the fascinating theory of nonwellfounded sets as presented by Aczel ([Ac88]). In section 2, a brief summary of his theory is given. (For a more extensive overview see [BE87].) Aczel formulates AFA in a very intuitive fashion, by viewing sets as graphs and the equality of sets as their being bisimilar (in a sense closely related to the original notion of Park). The existence of nonwellfounded sets, like the set a satisfying $a = \{a\}$, is an immediate consequence of AFA. The semantic universe P mentioned above will contain such nonwellfounded sets. A simple example is the process $p = \{<a, \varnothing>, <b, p>\}$, which represents an infinite binary tree at every node of which there is a choice between doing a and terminating, or doing b and continuing with again p.

An advantage of solving the above equation for P in the presence of AFA is the possibility of taking *arbitrary* subsets of $A \times P$, rather than metrically *closed* (or *compact*) ones only, which is necessary if one wants to define a metric on P. This allows for a description of LTS's that are not necessarily finitely branching or image finite. Moreover P is really equal to $\mathfrak{P}(A \times P)$, whereas in the metric and most other approaches, they are only isomorphic.

Another advantage of working in a set theory where AFA holds is constituted by the *solution lemma*, a direct consequence of AFA. It states the existence of a unique solution for a large class of recursive equations. Both for defining the semantic models and the semantics operators, the solution lemma is a very useful tool.

After the introduction of $\mathfrak{M}$ (in section 3), we consider in section 4 LTS's that are defined by means of transition system specifications (TSS). A TSS is a set of (axioms and) rules for defining transitions. These rules follow the syntactic structure of the states $s \in S$, which now are assumed to be terms over some (single-sorted) signature Σ: $S = T(\Sigma)$. Then the attention is focussed on TSS's of which the rules satisfy certain syntactic restrictions. The notion of syntactic *formats* of TSS's was recently studied in [GV88] (see also [BIM88]). There a special format for TSS's is introduced and it is shown that the bisimulation relation induced by such a TSS is a congruence with respect to the operators in Σ. In this paper, a restricted version of this format, called SOS, is treated, which is still sufficiently general to be of relevance for a large number of languages (see the examples in sections 5 and 6). It is shown that every TSS in SOS format induces a semantic interpretation for all operators in the signature Σ. These are next used to establish the fact that $\mathfrak{M}$ is compositional.

This constitutes another improvement on our previous work. There the compositionality of $\mathfrak{M}$ is proved by introducing a second model, which is defined compositionally using the semantic operators derived from the TSS, and which next is shown to be equal to $\mathfrak{M}$. Here the same result is obtained more directly.

The constructions above are illustrated by two small toy languages, which both are characterized by the fact that they contain a language construct that we were not able to model satisfactorily before. The first language is CCS-like ([Mi80]) but without synchronization; this has been left out for convenience sake, though it causes no additional problems to have it included. Its interest lies in the fact that it allows *unguarded* recursion. Secondly, this language is extended in two ways: the atomic actions are interpreted as transformations on some abstract set of states; further, a unary operator *atom* is added. For any statement s, the behaviour of $atom(s)$ is like an atomic action: it yields in one step a state transformation that is obtained by composing the successive steps of s. This construct was first introduced in [BaKo90], where it plays a crucial role in the semantic description of Concurrent Prolog. Here it is given a semantics that is both simpler and more abstract than in [BaKo90].

Acknowledgements: Discussions with Rob van Glabbeek and the Amsterdam Concurrency Group have been of help in understanding AFA and its applications.

2. NONWELLFOUNDED SETS

We shall work in the universe of *nonwellfounded* sets as presented by Peter Aczel in [Ac88]. (Note, however, that those sets were already conceived long before; see [Ac88] for an historic account.) For an overview of his theory we refer to the excellent summary in [BE87].

At the basis of Aczel's work lies the conception of sets as (pointed) graphs. Every set A gives rise to a graph by taking as nodes the transitive closure of A and as (directed) edges all pairs x and y with $y \in x$. Conversely, every graph is associated with a unique set.

It is this latter observation that Aczel turns into an axiom, the so-called *anti-foundation-axiom* (AFA). More formally it says: every graph has a unique decoration. Here a *decoration* for a graph is a function D that assigns to every node of the graph a set such that for each node x

$$D(x) = \{D(y): y \text{ is a child of } x\}$$

An immediate consequence of AFA is the existence of nonwellfounded sets: consider the one node graph with one edge leading from this node to itself. Since this graph has, by AFA, a decoration, there exists a set a with $a = \{a\}$ (which is moreover unique). The set-theoretic framework Aczel works in, is determined by the usual axioms of Zermelo-Fraenkel (ZFC), of which the axiom of foundation is omitted (yielding ZFC$^-$), and to which AFA is added. The resulting collection of axioms is denoted by ZFC$^-$ / AFA. (In [Ac88], the (relative) consistency of ZFC$^-$ / AFA is shown.)

We shall make use of two principles that are a direct consequence of AFA: the *solution lemma* and the principle of *strong extensionality*.

The *solution lemma* asserts the existence of a unique solution for a class of *systems* of (recursive) equations. It is formulated as follows. Consider a set X of variables x. (Formally these variables are called *atoms* or *Urelemente*.) A *system* of equations is a collection

$$\{x = a_x\}_{x \in X}$$

where, for every x, the set a_x may contain any of the variables occurring at the lefthand side of any of the equations. (A simple example of a system of equations is $\{x=\{x\}\}$.) A solution for such a system is a collection π of sets $\{\pi(x)\}_{x \in X}$ such that, for every x,

$$\pi(x) = a_x[\pi(x_1), \ldots, \pi(x_n)]$$

(Here we use the rather informal notation $a_x[\pi(x_1), \ldots, \pi(x_n)]$ to denote the set that is obtained from a_x by substituting in a_x every variable x_i by $\pi(x_i)$.) Now we can formulate the following theorem.

THEOREM 2.1 (Solution Lemma): *Every system of equations has a unique solution.*

In order to formulate the principle of *strong extensionality*, we first have to introduce the notion of *ϵ-bisimulation*. (Actually it is plainly called *bisimulation* in Aczel's book. The ϵ prefix is used to distinguish it from the usual notion of bisimulation, to be defined in the next section.)

DEFINITION 2.2 (ϵ-bisimulation): A binary relation R on sets is called an ϵ-bisimulation if it is symmetric and, for all sets a and b with aRb,

$$\forall x \in a\ \exists y \in b\ [xRy]$$

Two sets a and b are called *ϵ-bisimilar* (notation $a \equiv b$) if there exists an ϵ-bisimulation relation R with aRb.

Now the principle of strong extensionality says that whenever two sets are ϵ-bisimilar, they are equal.

THEOREM 2.3 (Strong extensionality): *For all sets a and b,*

$$a \equiv b \Rightarrow a = b$$

The principle of strong extensionality gives us a way of dealing with equality of nonwellfounded sets; e.g., it can be used to prove $a=b$ for $a=\{a\}$ and $b=\{b\}$. (Note that the usual axiom of extensionality does not help here.)

Finally, we mention a theorem stating the existence of fixed-points for a class of recursive domain equations. Again first a definition.

DEFINITION 2.4: A *class operator* Φ assigns to each class X a class ΦX. A class operator is *set-continuous* if, for each class X,

$$\Phi X = \bigcup \{\Phi x\colon x \text{ is a subset of } X\}$$

Aczel shows that every set-continuous class operator has a smallest and a largest fixed-point. In many cases, the smallest contains all wellfounded elements that are present in the latter, which moreover may contain nonwellfounded sets. We shall use only largest fixed-points, which are characterized in the following theorem.

THEOREM 2.5 (Largest fixed-point): *Let Φ be a set-continuous class operator. Let*

$$J_\Phi = \bigcup \{x : x \text{ is a subset of } \Phi x\}$$

Then J_Φ is the largest fixed-point of Φ.

Now we can solve recursive domain equations in the usual way by associating with such an equation a class operator. The fixed-points of this operator will satisfy the domain equation.

3. MODELS FOR BISIMULATION

As a starting point for our semantic considerations, we take the notion of *labelled transition system* (LTS) in the style of Plotkin's structured operational semantics (SOS). For every LTS $\mathcal{T}$ a semantics $\mathcal{M}_\mathcal{T}$ will be defined that assigns to every state of $\mathcal{T}$ its tree-like unfolding under the transition relation of $\mathcal{T}$. This semantics is characterized by the fact that for every state s its value under $\mathcal{M}_\mathcal{T}$ is a minimal canonical representative for the (strong) bisimulation equivalence class of s.

First the notion of labelled transition system is introduced.

DEFINITION 3.1 (LTS): A *labelled transition system* is a triple $\mathcal{T}=(S,A,\rightarrow)$ consisting of a set of *states* S, a set of *labels* A, and a *transition relation* $\rightarrow \subseteq S \times A \times S$. We shall write $s \xrightarrow{a} s'$ for $(s,a,s') \in \rightarrow$.

DEFINITION 3.2 (Bisimulation): Let $\mathcal{T}=(S,A,\rightarrow)$ be a LTS. A relation $R \subseteq S \times S$ is called a *(strong) bisimulation* if it is symmetric and, for all $s,t \in S$ and $a \in A$,

$$(sRt \wedge s \xrightarrow{a} s') \Rightarrow \exists t' \in S \; [\, t \xrightarrow{a} t' \wedge s'Rt' \,]$$

Two states are *bisimilar* in $\mathcal{T}$, notation $s \leftrightarrow t$, if there exists a bisimulation relation R with sRt. (Note that bisimilarity is an equivalence relation on states.)

Next we introduce for every LTS $\mathcal{T}=(S,A,\rightarrow)$ a semantics $\mathcal{M}_\mathcal{T}$, which maps every state $s \in S$ onto its tree-like unfolding under the transition relation $\rightarrow$. It has as a co-domain the set P of *processes*, which is defined as follows.

DEFINITION 3.3 (P): Let P be the largest class satisfying

$$P = \mathcal{P}(A \times P)$$

(Here the set A is the set of labels of $\mathcal{T}$.) Formally, P is obtained as the largest fixed-point of the class operator Φ that assigns to every class X the class $\mathcal{P}(A \times X)$. It is straightforward to show that Φ is set-continuous. (The interpretation of $\mathcal{P}(A \times X)$ is of importance, however; it should be the class of all *subsets* of $A \times X$. This distinction between sets and classes also explains why there is no problem of cardinality.)

The following notion will be useful in many cases where equality of processes has to be established.

DEFINITION 3.4 (Process-bisimulation): A binary relation $R \subseteq P \times P$ is called a *process-bisimulation* if it is symmetric and, for all processes p and q with pRq,

$$\forall \langle a,p' \rangle \in p \; \exists \langle a,q' \rangle \in q \; [p'Rq']$$

Two processes p and q are called *process-bisimilar* (notation $p \equiv_P q$) is there exists a process-bisimulation relation R with pRq.

The following theorem is a direct consequence of the principle of strong extensionality.

THEOREM 3.5: *For all* $p,\ q \in P$

$$p \equiv_P q \Rightarrow p = q$$

PROOF: We show, for all $p,\ q \in P$,

$$p \equiv_P q \Rightarrow p \equiv q$$

From this and the principle of strong extensionality the theorem follows. Let $p \equiv_P q$. Then there exists a process-bisimulation R with pRq. We define

$$S = S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5$$

with

$$S_1 = R$$

$$S_2 = \{(\{a\}, \{a\}) \colon a \in A\}$$

$$S_3 = \{(a, a) \colon a \in A\}$$

$$S_4 = \{(\{a,p\}, \{a,q\}) \colon pRq\}$$

$$S_5 = \{(<a,p>, <a,q>) \colon pRq\}$$

It is not very difficult to show that S is an ϵ-bisimulation. (Note that $<x,y>$ is shorthand for $\{\{x\}, \{x,y\}\}$.) Thus $p \equiv q$. (End of proof.)

For every LTS $\mathcal{T}$ a model $\mathcal{M}_{\mathcal{T}}: S \to P$ is defined as follows.

DEFINITION 3.6 ($\mathcal{M}_{\mathcal{T}}$): Let $\mathcal{T} = (S, A, \to)$ be a LTS. We define a model $\mathcal{M}_{\mathcal{T}}: S \to P$ by

$$\mathcal{M}_{\mathcal{T}}[\![s]\!] = \{<a, \mathcal{M}_{\mathcal{T}}[\![s']\!]> \colon s \xrightarrow{a} s'\}$$

We can justify this recursive definition by an application of the Solution Lemma: consider the system of equations

$$\{x_s = \{<a, x_{s'}> \colon s \xrightarrow{a} s'\}\}_{s \in S}$$

assuming the presence of a set of variables $\{x_s\}_{s \in S}$. Let π be a unique solution for this system. Then we can define

$$\mathcal{M}_{\mathcal{T}}[\![s]\!] = \pi(x_s)$$

The fact that $\pi(x_s)$ is in P is a direct consequence of the fact that P is the *largest* class satisfying the equation used for its definition.

This model is of interest because it assigns the same meaning to states that are bisimilar. This we prove next. (See also [GR89]; in [Ab87] a similar result is given that additionally takes into account divergence, which we do not consider here.)

THEOREM 3.7: *Let* $\leftrightarrow \subseteq S \times S$ *denote the bisimilarity relation induced by the labelled transition system* $\mathcal{T} = (S, A, \to)$. *Then*

$$\forall s,t \in S\ [s \leftrightarrow t \Leftrightarrow \mathfrak{M}_{\mathcal{T}}[\![s]\!] = \mathfrak{M}_{\mathcal{T}}[\![t]\!]]$$

PROOF: Let $s,t \in S$.

$\Leftarrow$:

Suppose $\mathfrak{M}_{\mathcal{T}}[\![s]\!] = \mathfrak{M}_{\mathcal{T}}[\![t]\!]$. We define a relation $\equiv \subseteq S \times S$ by

$$s' \equiv t' \Leftrightarrow \mathfrak{M}_{\mathcal{T}}[\![s']\!] = \mathfrak{M}_{\mathcal{T}}[\![t']\!]$$

From the definition of $\mathfrak{M}_{\mathcal{T}}$ it is straightforward that $\equiv$ is a bisimulation relation on S.

$\Rightarrow$:

Consider s and t with $s \leftrightarrow t$. According to Theorem 3.5, it is sufficient to show that $\mathfrak{M}_{\mathcal{T}}[\![s]\!]$ and $\mathfrak{M}_{\mathcal{T}}[\![t]\!]$ are process-bisimilar. Let

$$R = \{(\mathfrak{M}_{\mathcal{T}}[\![u]\!], \mathfrak{M}_{\mathcal{T}}[\![v]\!]) : u \leftrightarrow v\}$$

It is not difficult to show that R is a process-bisimulation. (End of proof.)

4. TRANSITION SYSTEM SPECIFICATIONS AND COMPOSITIONALITY

In this section, we shall consider LTS's of a special format, namely, in which the set of states consists of the set of closed terms generated by a single sorted *signature*. The notion of *transition system specification* (TSS) will be introduced: a TSS is a set of axioms and rules for defining transitions; every TSS induces a LTS. Then it will be shown that every TSS $\mathfrak{R}$ that has a special format induces semantic interpretations for all syntactic operators in the signature Σ. Finally, these semantic operators will be used to prove that $\mathfrak{M}_{\mathcal{T}}$ is compositional, where $\mathcal{T}$ is the LTS induced by the TSS $\mathfrak{R}$.

A signature $\Sigma = (F,r)$ consists of a set $f \in F$ of *function names* and a *rank* function $r: F \to \mathbb{N}$ indicating for each function symbol its arity. Function symbols of arity 0 we call *constants*. Sometimes $f \in \Sigma$ is written for $f \in F$. Further we introduce a set of variables $x,y \in Var$. The set of *terms* $s,t,u \in T(\Sigma, Var)$ built from Σ and Var is defined as usual; using the so-called BNF syntax, it can be given by

$$t ::= x \mid f(t_1, \ldots, t_{r(f)})$$

Terms containing no variables are called *closed*. The set of closed terms is denoted by $T(\Sigma)$. Let $x_1, \ldots, x_k \in Var$ be distinct variables. For a term t we write $t_{(x_1, \ldots, x_k)}$ or $t_{\mathbf{x}}$ to indicate that the set of variables occurring in t is contained in the set $\{x_1, \ldots, x_k\}$. Whenever it is clear from the context what the free variables occurring in t are, these subscripts are omitted.

We have the usual syntactic substitution: We write $t_{(x_1, \ldots, x_k)}(u_1, \ldots, u_k)$, or $t_{\mathbf{x}}(\mathbf{u})$ for the term obtained by replacing every occurrence of x_i in t by u_i, for $1 \leq i \leq k$.

DEFINITION 4.1 (Interpretations): We define the set $(I \in)$ *IntPr* of *interpretations* for $\Sigma = (F,r)$ as the collection of all functions

$$I: F \to \bigcup_k (P^k \to P)$$

with $I(f) \in P^{r(f)} \to P$ for every $f \in F$. (Read P for $P^0 \to P$.) An interpretation I induces for every term $t_{(x_1, \ldots, x_k)}$ in $T(\Sigma, Var)$ a function $t_{\mathbf{x}}^I: P^k \to P$ that is inductively given by

(1) $(x_i)_{\mathbf{x}}^I(p_1, \ldots, p_k) = p_i$

(2) $f(t_1, \ldots, t_{r(f)})_{\mathbf{x}}^I(p_1, \ldots, p_k) =$

$$I(f)((t_1)_{\mathbf{x}}^I(p_1, \ldots, p_k), \ldots, (t_{r(f)})_{\mathbf{x}}^I(p_1, \ldots, p_k))$$

(We also write f^I for $I(f)$.)

Below we shall see how LTS's of a certain type induce an interpretation for Σ.
Any LTS that has the set $T(\Sigma)$ as states can be specified with the help of rules (and axioms).

DEFINITION 4.2 (TSS): A *transition system specification* (TSS) $\mathfrak{R}$ for Σ is a (possibly infinite) set of *rules* R of the form

$$\frac{\{t_i \xrightarrow{a_i} t_i' \colon 1 \leqslant i \leqslant n\}}{t \xrightarrow{a} t'}$$

where $n \geqslant 0$, t_i, t_i', t, $t' \in T(\Sigma, Var)$, and a_i, $a \in A$. The elements $\{t_i \xrightarrow{a_i} t_i' \colon 1 \leqslant i \leqslant n\}$ are called the premises and $t \xrightarrow{a} t'$ is called the conclusion of this rule. If $n = 0$ then a rule is called an *axiom*.

DEFINITION 4.3 (Transitions): An expression of the form $t \xrightarrow{a} t'$, with $t, t' \in T(\Sigma)$, is called a *transition*. Let $\mathfrak{R}$ be a TSS. A *proof tree* PT for a transition ψ from $\mathfrak{R}$ is defined in the usual way: it is a finite tree with root ψ such that the transition labelling a father node follows from the transitions labelling its sons by an application of (an instantiation of) a rule $R \in \mathfrak{R}$. Notation: $\mathfrak{R} \vdash_{PT} \psi$. We write $\mathfrak{R} \vdash \psi$ to express that there exists a proof tree PT with $\mathfrak{R} \vdash_{PT} \psi$. A transition may have many proof trees.

Every TSS leads naturally to the definition of a LTS.

DEFINITION 4.4 (Induced $\mathfrak{T}$): Every TSS $\mathfrak{R}$ for Σ induces a LTS $\mathfrak{T} = (T(\Sigma), A, \rightarrow)$ by taking $\rightarrow \subseteq T(\Sigma) \times A \times T(\Sigma)$ as

$$t \xrightarrow{a} t' \Leftrightarrow \mathfrak{R} \vdash t \xrightarrow{a} t'$$

We fix for the remainder of this section a signature Σ_{rec}, given by

$$\Sigma_{rec} = \Sigma \cup RecVar$$

Here Σ is arbitrary and $(X \in)$ $RecVar$ is a set of *recursion variables* (which are constants in the signature Σ_{rec}). The interpretation of recursion variables will be dependent on so-called *declarations*. The set of declarations is given by

$$(d \in)\ Decl = RecVar \rightarrow T(\Sigma_{rec})$$

In the remainder of this section, $d \in Decl$ is a fixed declaration for the recursion variables in $\Sigma_{rec} = \Sigma \cup RecVar$.

Next we consider TSS's for the signature Σ_{rec}, with a special format, the so-called SOS format. Then it is shown how a TSS in SOS format induces an interpretation for Σ_{rec}.

DEFINITION 4.5 (SOS format): A TSS $\mathfrak{R} = \mathfrak{R}_0 \cup \mathfrak{R}_{rec}$ for Σ_{rec} is in SOS format if $\mathfrak{R}_{rec}$ is a TSS for $RecVar$ given by, for every $X \in RecVar$,

$$\frac{d(X) \xrightarrow{a} y}{X \xrightarrow{a} y}$$

and if $\mathfrak{R}_0$ is a TSS for Σ, of which all rules are of the form

$$\frac{\{u_i \xrightarrow{a_i} v_i \colon 1 \leqslant i \leqslant n\}}{f(x_1, \ldots, x_{r(f)}) \xrightarrow{a} g(y_1, \ldots, y_{r(g)})}$$

with

$f, g \in \Sigma$,

$n \geqslant 0$,

$a_i,\ a \in A$, (A is the set of *labels*)

$x_i, v_i \in Var$, all distinct,

$u_1 \in \{x_1, \ldots, x_{r(f)}\}$,

$u_{i+1} \in \{x_1, \ldots, x_{r(f)}, v_1, \ldots, v_i\}$,

$\{y_1, \ldots, y_{r(g)}\} \subseteq \{x_1, \ldots, x_{r(f)}, v_1, \ldots, v_n\}$

As mentioned in the introduction, this format specializes the more general format introduced in [GV88]. For some more discussion and a comparison with other formats (like GSOS in [BIM88]), see section 7.

Every TSS in SOS format gives rise to an interpretation for all operators in Σ ($\subseteq \Sigma_{rec}$).

DEFINITION 4.6 ($I(\mathfrak{R})$): Let $\mathfrak{R}$ be a TSS for Σ_{rec} ($= \Sigma \cup RecVar$) in SOS format. An interpretation $I(\mathfrak{R})$ for Σ is defined as follows. Let $f \in \Sigma$ and $p_1, \ldots, p_{r(f)} \in P$. Let $\mathbf{p} = p_1, \ldots, p_{r(f)}$ and $\mathbf{q} = q_1, \ldots, q_n$. Then

$$f^{I(\mathfrak{R})}(\mathbf{p}) =$$

$$\{<a,\ g^{I(\mathfrak{R})}(y_1^{I(\mathfrak{R})}(\mathbf{p},\mathbf{q}), \ldots, y_{r(g)}^{I(\mathfrak{R})}(\mathbf{p},\mathbf{q}))>: \exists R \in \mathfrak{R}\, \forall i \in \{1, \ldots, n\}\ [<a_i,\ q_i> \in u_i^{I(\mathfrak{R})}(\mathbf{p},\mathbf{q})\,]\}$$

where

$$R = \frac{\{u_i \xrightarrow{a_i} v_i:\ 1 \leqslant i \leqslant n\}}{f(x_1, \ldots, x_{r(f)}) \xrightarrow{a} g(y_1, \ldots, y_{r(g)})}$$

satisfies the conditions of Definition 4.5.

(Note that $u_i^{I(\mathfrak{R})}(\mathbf{p},\mathbf{q})$ is p_k if $u_i = x_k$, and q_k if $u_i = v_k$. Similarly for $y_i^{I(\mathfrak{R})}(\mathbf{p},\mathbf{q})$.)

Again the existence of this recursively defined interpretation can be established with the help of the solution lemma. Intuitively, $f^{I(\mathfrak{R})}$ is obtained by semantically interpreting those rules in $\mathfrak{R}$ that have f for their conclusion; the satisfaction of the premises is mirrored by the presence of a pair starting with a_i in the with u_i corresponding process $u_i^{I(\mathfrak{R})}(\mathbf{p},\mathbf{q})$.

The above interpretation can be used to prove that for a LTS $\mathfrak{T}$ induced by a TSS $\mathfrak{R}$ in SOS format, the function $\mathfrak{M}_{\mathfrak{T}}$ is compositional.

THEOREM 4.7 (Compositionality of $\mathfrak{M}$): *Let $\mathfrak{R}$ be a TSS in SOS format and let $\mathfrak{T}$ be induced by $\mathfrak{R}$. For all operators $f \in \Sigma$ and terms $t_1, \ldots, t_{r(f)}$,*

$$\mathfrak{M}_{\mathfrak{T}}[\![f(t_1, \ldots, t_{r(f)})]\!] = f^{I(\mathfrak{R})}(\mathfrak{M}_{\mathfrak{T}}[\![t_1]\!], \ldots, \mathfrak{M}_{\mathfrak{T}}[\![t_{r(f)}]\!])$$

PROOF: Define

$$R = \{(\mathfrak{M}_{\mathfrak{T}}[\![f(t_1, \ldots, t_{r(f)})]\!],\ f^{I(\mathfrak{R})}(\mathfrak{M}_{\mathfrak{T}}[\![t_1]\!], \ldots, \mathfrak{M}_{\mathfrak{T}}[\![t_{r(f)}]\!])):\ f \in \Sigma,\ t_i \in T(\Sigma)\}$$

and show that R is a process-bisimulation. (End of proof.)

Since it is immediate that for all $X \in RecVar$,

$$\mathcal{M}_{\mathcal{T}}[\![X]\!] = \mathcal{M}_{\mathcal{T}}[\![d(X)]\!]$$

the model $\mathcal{M}$ can be characterized as being what is usually called *denotational*: let the set of environments $(\gamma \in)$ Env be given by

$$Env = RecVar \to P$$

Define $\mathcal{D}$: $T(\Sigma) \to Env \to P$ by, for all operators $f \in \Sigma$, terms $t_1, \ldots, t_{r(f)}$ and $X \in RecVar$,

$$\mathcal{D}[\![f(t_1, \ldots, t_{r(f)})]\!](\gamma) = f^{I(\mathfrak{M})}(\mathcal{D}[\![t_1]\!](\gamma), \ldots, \mathcal{D}[\![t_{r(f)}]\!](\gamma))$$

$$\mathcal{D}[\![X]\!](\gamma) = \gamma(X)$$

Let γ_d be defined by, for all $X \in RecVar$,

$$\gamma_d(X) = \mathcal{M}_{\mathcal{T}}[\![d(X)]\!]$$

Then we have the following theorem.

THEOREM 4.8 ($\mathcal{M}_{\mathcal{T}}$ is denotational): For all $s \in T(\Sigma)$,

$$\mathcal{M}_{\mathcal{T}}[\![s]\!] = \mathcal{D}[\![s]\!](\gamma_d)$$

5. A SIMPLE LANGUAGE WITH UNGUARDED RECURSION

As an example we consider a signature $\Sigma_{rec} = \langle F, r \rangle$ that is defined as follows. Let the set F of function symbols be given by

$$F = Act \cup \{ ;, \|, + \} \cup RecVar$$

where $(X \in)$ $RecVar$ is the set of recursion variables and $(a \in)$ Act is an abstract set of basic actions. The rank function r of Σ_{rec} is defined by

$$r(a) = 0, \quad \text{for every } a \in Act$$

$$r(X) = 0, \quad \text{for every } X \in RecVar$$

$$r(;) = r(\|) = r(+) = 2$$

The set $T(\Sigma_{rec})$ of closed terms over Σ_{rec} is called a *language*. In BNF notation it can be defined as the language $(s, t \in)$ $\mathcal{L}$ given by

$$s ::= a \mid s;t \mid s\|t \mid s+t \mid X$$

The interpretation of the operators ;, $\|$ and $+$, for sequential, parallel and non-deterministic composition, respectively, is as usual.

Next we define a LTS $\mathcal{T} = \langle T(\Sigma_{rec}), A, \to \rangle$. The set $(\alpha \in)$ A of labels is given by

$$A = Act \cup \overline{Act}$$

The elements of $\overline{Act}$ $(= \{\bar{a}: a \in Act\})$ are used to indicate termination (see the rules for sequential and parallel composition below).

The transition relation $\to$ of $\mathcal{T}$ is induced by the following TSS $\mathcal{R}$. The axiom for the basic actions is

$$a \xrightarrow{\bar{a}} \delta$$

where δ is a special element of Act denoting termination. Further rules in $\mathfrak{R}$ are

$$\frac{x_1 \xrightarrow{\bar{a}} y_1}{\begin{array}{c} x_1;x_2 \xrightarrow{a} x_2 \\ x_1 \| x_2 \xrightarrow{a} x_2 \\ x_2 \| x_1 \xrightarrow{a} x_2 \end{array}} \qquad \frac{x_1 \xrightarrow{a} y_1}{\begin{array}{c} x_1;x_2 \xrightarrow{a} y_1;x_2 \\ x_1 \| x_2 \xrightarrow{a} y_1 \| x_2 \\ x_2 \| x_1 \xrightarrow{a} x_2 \| y_1 \end{array}} \qquad \frac{x_1 \xrightarrow{\alpha} y_1}{\begin{array}{c} x_1+x_2 \xrightarrow{\alpha} y_1 \\ x_2+x_1 \xrightarrow{\alpha} y_1 \end{array}}$$

Finally, the rule for the recursion variables is

$$\frac{d(X) \xrightarrow{\alpha} y}{X \xrightarrow{\alpha} y}$$

were d is some fixed declaration.

Next we will apply the definitions and theorem of the previous sections. Definition 3.6 yields a model $\mathfrak{M}_{\mathcal{T}}$: $T(\Sigma_{rec}) \rightarrow P$ given by

$$\mathfrak{M}_{\mathcal{T}}[\![s]\!] = \{<a, \mathfrak{M}[\![s']\!]>: s \xrightarrow{a} s'\}$$

Moreover, $\mathfrak{R}$ is in SOS format. Thus it induces an interpretation $I(\mathfrak{R})$ for Σ according to Definition 4.6. We have for the interpretations of the function symbols the following equalities.

$$a^{I(\mathfrak{R})} = \{<\bar{a}, \varnothing>\}$$

$$p;^{I(\mathfrak{R})} q = \{<a,q>: <\bar{a},p'> \in p\} \cup \{<a,p';^{I(\mathfrak{R})} q>: <a,p'> \in p\}$$

$$p \|^{I(\mathfrak{R})} q = \{<a,q>: <\bar{a},p'> \in p\} \cup \{<a,p' \|^{I(\mathfrak{R})} q>: <a,p'> \in p\} \cup$$
$$\{<a,p>: <\bar{a},q'> \in q\} \cup \{<a,p \|^{I(\mathfrak{R})} q'>: <a,q'> \in q\}$$

$$p +^{I(\mathfrak{R})} q = p \cup q$$

$$X^{I(\mathfrak{R})} = (d(X))^{I(\mathfrak{R})}$$

For $\mathfrak{M}_{\mathcal{T}}$ the following equalities hold, according to Theorem 4.7:

$$\mathfrak{M}_{\mathcal{T}}[\![a]\!] = \{<\bar{a}, \varnothing>\}$$

$$\mathfrak{M}_{\mathcal{T}}[\![s;t]\!] = \mathfrak{M}_{\mathcal{T}}[\![s]\!] ;^{I(\mathfrak{R})} \mathfrak{M}_{\mathcal{T}}[\![t]\!]$$

$$\mathfrak{M}_{\mathcal{T}}[\![s \| t]\!] = \mathfrak{M}_{\mathcal{T}}[\![s]\!] \|^{I(\mathfrak{R})} \mathfrak{M}_{\mathcal{T}}[\![t]\!]$$

$$\mathfrak{M}_{\mathcal{T}}[\![s+t]\!] = \mathfrak{M}_{\mathcal{T}}[\![s]\!] +^{I(\mathfrak{R})} \mathfrak{M}_{\mathcal{T}}[\![t]\!]$$

$$\mathfrak{M}_{\mathcal{T}}[\![X]\!] = \mathfrak{M}_{\mathcal{T}}[\![d(X)]\!]$$

Next we mention a characterization of $\mathfrak{M}_{\mathcal{T}}$ as a hereditary union. Let $\rightarrow_f$ be the smallest transition relation satisfying all the axioms and rules in the definition of $\rightarrow$ above except the rule for recursion. Thus $\rightarrow_f \subseteq \rightarrow$ but not the other way around. Next the notion of repeated "body replacement" is introduced.

DEFINITION 5.1: For all $n \geqslant 0$ and statements $s \in \mathfrak{L}$ the statement s^n is inductively defined by

$$s^0 = s$$

$$s^{n+1} = s^n[d(X_1)/X_1, \ldots, d(X_k)/X_k]$$

assuming that the set of recursion variables occurring in s^n is $\{X_1, \ldots, X_k\}$. (The term $s^n[d(X_1)/X_1, \ldots, d(X_k)/X_k]$ is obtained by replacing in s^n every occurrence of X_i by $d(X_i)$.)

Now $\mathcal{M}_{\mathcal{T}}$ can be characterized as follows.

THEOREM 5.2: $\mathcal{M}_{\mathcal{T}}[\![s]\!] = \bigcup_n \{<a, \mathcal{M}_{\mathcal{T}}[\![s']\!]>: s^n \xrightarrow{a}_f s'\}$

Interestingly, it is not necessary to restrict recursion to the case where all statements $d(X)$ are *guarded* in X, as is done usually. In [Ru90], only guarded recursion is treated because the unguarded case does not fit into the metric framework used there. In [BeKl87], unguarded equations are considered for which solutions are found via an interesting but complicated combinatorial technique.

6. ANOTHER EXAMPLE: 'ATOMIZED' STATEMENTS

As a second example, we extend the signature $\Sigma_{rec} = <F, r>$ of the previous section with a unary operator *atom*, thus yielding

$$F = Act \cup \{\,;, \|, +, atom\,\} \cup RecVar$$

Again, $(X \in)$ *RecVar* is the set of recursion variables and $(a \in)$ *Act* is an abstract set of atomic actions. In BNF notation, the set $T(\Sigma_{rec})$ of closed terms over Σ_{rec} be defined as the language $(s \in)$ $\mathcal{L}$ given by

$$s ::= a \mid s;t \mid s\|t \mid s+t \mid atom(s) \mid X$$

Atomic actions are now interpreted as state transformations: let *States* be some set of abstract states. We assume the presence of an interpretation function

$$[\![\;]\!]: Act \to (States \to_{part} States)$$

that assigns to every atomic action a a partial function $[\![a]\!]$ from states to states.

The interpretation of the operators ;, $\|$ and $+$, is as before. For a statement s, the statement $atom(s)$ behaves like an 'atomized' version of s: for every finite sequence of state transformations in the behaviour of s, it yields in one step (like an atomic action) a state transformation that is the composition of this sequence.

Again a LTS $\mathcal{T} = <T(\Sigma_{rec}), A, \to>$ is defined. The set $(\alpha \in)$ A of labels is now given by

$$A = SPair \cup \overline{SPair}$$

where $(\pi \in)$ $SPair = States \times States$ and $\overline{SPair} = \{\bar{\pi}: \pi \in SPair\}$. The latter are again used to indicate termination.

The transition relation $\to$ of $\mathcal{T}$ is induced by the following TSS $\mathcal{R}$. The axiom for atomic actions is

$$a \xrightarrow{\bar{\pi}} \delta$$

where $\pi = (\sigma, [\![a]\!](\sigma))$, with $\sigma \in States$ such that $[\![a]\!](\sigma)$ is defined. The rules for ;, $\|$, $+$ and X are as before. For *atom* we have

$$\frac{u_1 \xrightarrow{\pi_1} u_2, \ldots, u_{n-1} \xrightarrow{\pi_{n-1}} u_n,\; u_n \xrightarrow{\bar{\pi}_n} u_{n+1}}{atom(u_1) \xrightarrow{\bar{\pi}} u_{n+1}}$$

where

$$\pi_1 = (\sigma, \sigma_1),\; \pi_2 = (\sigma_1, \sigma_2), \ldots, \pi_n = (\sigma_{n-1}, \sigma'),\; \pi = (\sigma, \sigma')$$

As before $\mathcal{R}$ is in SOS format. Thus it induces an interpretation $I(\mathcal{R})$ for Σ according to Definition 4.6. For the interpretation of $a \in Act$ and *atom*, we have the following equalities:

$$a^{I(\mathfrak{A})} = \{ <(\sigma, [\![a]\!](\sigma)), \ \varnothing >: \sigma \in States, \ [\![a]\!](\sigma) \text{ defined } \}$$

$$atom^{I(\mathfrak{A})}(p) = \{ <\bar{\pi}, q>: \pi_1 \cdots \bar{\pi}_n \cdot q \text{ is a path in } p \}$$

where

$$\pi = (\sigma, \sigma'), \ \pi_1 = (\sigma, \sigma_1), \ \pi_2 = (\sigma_1, \sigma_2), \ldots, \pi_n = (\sigma_{n-1}, \sigma')$$

The language construct *atom* was first introduced in [BaKo90], where it is used in the semantics of guarded clauses of Concurrent Prolog, a parallel logic programming language. The semantic description of *atom* given there is quite involved. This is mainly caused by the necessity, imposed by the metric framework which is used, to keep track of all internal intermediate steps that *atom*(s) can make. In the present model, this is not needed. As a consequence, it is more transparent and, more importantly, more abstract: in the above model, two different statements s and s' can have a different semantics whereas the semantics of *atom*(s) and *atom*(s') is the same.

7. Discussion

The central result of this paper is the construction of an interpretation for a signature Σ from a TSS for Σ that is in SOS format (Definition 4.6). Due to the use of nonwellfounded sets, this construction is more general than the one given in [Ru90]: first, it can handle TSS's that need not be finitely branching. Secondly, the SOS format is more general than the GSOS format that was used in [Ru90]: in the premises of a rule in SOS format, a so-called *look-ahead*, like $\{ u_1 \rightarrow v_1, \ v_1 \rightarrow v_2 \}$, is allowed. This type of premises is excluded by the *GSOS* format. It is exactly this difference that is illustrated by the example in section 6.

There is also a respect in which the SOS format is less general than the GSOS format: at the right-hand side of the conclusion of a rule in GSOS format arbitrary terms are allowed, whereas in the SOS format only terms with one function symbol may occur.

At present, we are working on a generalization of the present approach, which overcomes this problem ([Ru92]). Moreover, it is applicable to the so-called *tyft* format introduced in [GV88], which is more general than both the SOS and the GSOS format in that it allows arbitrary terms both at the left-hand sides of the premises and at the right-hand side of the conclusion of the rules.

8. References

[Ab87] S. Abramsky, *Domain theory and the logic of observable properties*, PhD thesis, University of London, October 1987.

[Ac88] P. Aczel, *Non-well-founded sets*, CSLI Lecture Notes No. 14, 1988.

[AR89] P. America, J.J.M.M. Rutten, *Solving reflexive domain equations in a category of complete metric spaces*, Journal of Computer and System Sciences, Vol. 39, No. 3, 1989, pp. 343-375.

[BE87] J. Barwise, J. Etchemendy, *The Liar, an essay on truth and circularity*, Oxford University Press, 1987.

[BaKo90] J.W. de Bakker, J.N. Kok, *Comparative metric semantics for Concurrent Prolog*, Theoretical Computer Science 75, 1990, pp. 15-43.

[BeKl87] J.A. Bergstra, J.W. Klop, *A convergence theorem in process algebra*, Technical Report CS-R8733, Centre for Mathematics and Computer Science, Amsterdam, 1987.

[BIM88] B. Bloom, S. Istrail, A.R. Meyer, *Bisimulation can't be traced: preliminary report*, in: Proceedings of the Fifteenth POPL, San Diego, California, 1988, pp. 229-239.

[BZ82] J.W. de Bakker, J.I. Zucker, *Processes and the denotational semantics of concurrency*, Information and Control 54, 1982, pp. 70-120.

[GR89] R.J. VAN GLABBEEK, J.J.M.M. RUTTEN, *The processes of De Bakker and Zucker represent bisimulation equivalence classes,* in: J.W. de Bakker, 25 jaar semantiek, Centre for Mathematics and Computer Science, Amsterdam, 1989.

[GV88] J.F. GROOTE, F. VAANDRAGER, *Structured operational semantics and bisimulation as a congruence,* Technical Report CS-R8845, Centre for Mathematics and Computer Science, Amsterdam, 1988. (To appear in Information and Computation. Extended abstract in: Proceedings 16th ICALP, Stresa, Lecture Notes in Computer Science 372, Springer-Verlag, 1989, pp. 423-438.)

[Mi80] R. MILNER, *A calculus of communicating systems,* Lecture Notes in Computer Science 92, Springer-Verlag, 1980.

[Pa81] D.M.R. PARK, *Concurrency and automata on infinite sequences,* in: Proceedings 5th GI conference, Lecture Notes in Computer Science 104, Springer-Verlag, 1981, pp. 15-32.

[Pl81] G.D. PLOTKIN, *A structural approach to operational semantics,* Report DAIMI FN-19, Comp. Sci. Dept., Aarhus Univ. 1981.

[Ru90] J.J.M.M. RUTTEN, *Deriving denotational models for bisimulation from structured operational semantics,* in: (M. Broy and C.B. Jones, eds.) Proceedings IFIP TC2 Working Conference on Programming Concepts and Methods, Israel, 1990, pp. 155-177.

[Ru92] J.J.M.M. RUTTEN, *Processes as terms: Nonwellfounded models for bisimulation,* Technical Report, CWI, Amsterdam, in preparation.