

**stichting  
mathematisch  
centrum**



---

AFDELING INFORMATICA  
(DEPARTMENT OF COMPUTER SCIENCE)

IW 82/77

MEI

J.W. DE BAKKER

SEMANTICS OF INFINITE PROCESSES USING  
GENERALIZED TREES

Preprint

---

**2e boerhaavestraat 49 amsterdam**

BIBLIOTHEEK MATHEMATISCH CENTRUM  
AMSTERDAM

*Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.*

*The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).*

---

AMS(MOS) subject classification scheme (1970): 68A05

---

ACM-Computing Reviews-categories: 5.24

Semantics of infinite processes using generalized trees \*)

by

J.W. de Bakker

ABSTRACT

A proposal is outlined for the definition of the meaning of infinite processes within the framework of denotational semantics. An infinite process obtains as its meaning a function from (generalized) trees to trees. A combination of least-fixed-point and greatest-fixed-point techniques is used to characterize both trees and processes.

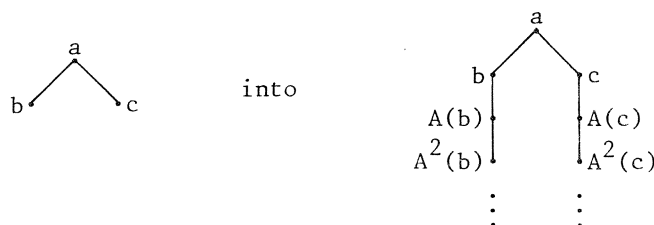
KEY WORDS & PHRASES: *denotational semantics, infinite processes, trees, least fixed points, greatest fixed points*

---

\*) This report will be submitted for publication elsewhere.

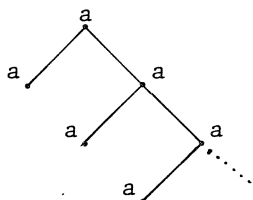
## 1. INTRODUCTION

In this paper, we give a preliminary account of an attempt we have made at using generalized trees for the definition of the semantics of infinite processes. Often, e.g. in the study of operating systems or of various forms of concurrent processes in general, one considers processes which continue indefinitely as normal rather than undesirable objects. A language construct which gives a first impression of the problems involved is, e.g.,  $\underline{\text{do}} S_1 \cup S_2 \underline{\text{od}}$ . The operational semantics of this construct is fairly clear: (\*) Choose, nondeterministically, between  $S_1$  and  $S_2$ , execute the selected statement, and repeat (\*). I.e., an infinite sequence of statements  $S^{(1)}, S^{(2)}, \dots$ , is performed, where each  $S^{(i)}$  is either  $S_1$  or  $S_2$ . However, it is less clear what meaning to attribute to such a statement in the framework of denotational semantics. There, it is customary to view the meaning of statements as functions from states to states, with the convention that for an input state for which the computation specified by the statement does not terminate, as output state some special undefined state, often denoted by " $\perp$ ", is delivered. Thus, all infinite computations become indistinguishable, and any analysis of their structure is virtually impossible. As a remedy we propose to attribute meaning to statements not as functions from states to states, but from (generalized) trees to trees. More specifically, each tree  $t$  (with states labelling its nodes) is transformed by (the function describing the meaning of) statement  $S$  into a tree  $t'$  which is an extension of  $t$  in that  $t'$  results from  $t$  by replacing its leaves by subtrees. E.g., the process  $\underline{\text{do}} A \underline{\text{od}}$  (with  $A$  some elementary action) transforms



To some extent, it may seem that this idea amounts to nothing but the incorporation of infinite computation sequences determined by statement  $S$  (or, rather, trees instead of sequences because of the presence of nondeterminacy) into the framework of denotational semantics. And, indeed, this is one way of viewing our proposal. There is more

to it, however. In order to deal with recursion, it is necessary to introduce some ordering on the various tree-transforming functions involved, which ordering is in its turn determined by an ordering on the trees. So we have to define this ordering, which has in addition to serve the following purpose: We want to be able to distinguish between two interpretations of a tree as indicated here, viz. as specifying on the one



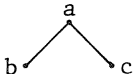
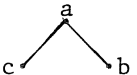
hand the set of all paths  $aa, aaa, \dots$  (i.e.  $aa^+$ ), and on the other hand this same set *together* with the infinite sequence  $a^\omega$  (i.e.,  $aa^+ \cup \{a^\omega\}$ ). Fixed points methods are used for this set, and the difference just mentioned is obtained through the distinction between least and greatest solutions of equations between (sets of paths associated with) trees. The following is an

example of this: Let, in general,  $t = \langle a, \{t_1, \dots, t_n\} \rangle$  be a tree with root  $a$  and subtrees  $t_1, \dots, t_n$ ,  $n \geq 0$ . Consider the equation  $t = \langle a, \{t, \langle a, \emptyset \rangle\} \rangle$ . Its least solution has as associated set of paths  $aa^+$ , whereas its greatest solution has as associated set of paths  $aa^+ \cup \{a^\omega\}$ . Corresponding to this distinction at the level of trees, we have a distinction in the semantics of the language construct of recursion. Consider - in a notation to be used only in this introduction - a recursive procedure  $P$ , the body of which consists of a choice between either some elementary action  $A$ , or  $A$  followed by a recursive call of  $P$ . Formally,  $P \Leftarrow A; P \cup A$ , say. For input state  $a$  we can choose between two possibilities for the set of output states:

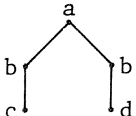
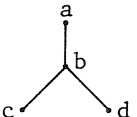
- $\{A(a), A^2(a), \dots, \perp\}$
- $\{A(a), A^2(a), \dots, A^\omega(a)\}$ .

Clearly, the second choice, which preserves more information, may be preferable in certain situations, and requires a formalism which refines the method of simply delivering  $\perp$  whenever an infinite computation is encountered.

Our system of generalized trees is a candidate for such a formalism. Why do we call them generalized? As suggested already, we view trees as specifying sets of paths, and identify trees which have identical such sets. Therefore we identify, e.g.,

1. Trees such as  and 

(both specify  $\{ab, ac\}$ ).

2. Trees such as  and 

(both specify  $\{abc, abd\}$ ).

Section 2 of the paper describes the syntax and semantics of our tree constructs, section 3 the programming language (with elementary actions, sequential composition, nondeterministic choice, (normal) recursion and infinite recursion (cf. the second

interpretation above)) to which meaning is attributed using fixed point techniques in the domain of operators on functions from trees to trees.

Greatest fixed points (but without our generalized trees) have been used by Hitchcock & Park [1], Mazurkiewicz [2], and De Roever [3]. For the framework of denotational semantics in general see, e.g., Scott & Strachey [4] or Milne & Strachey [5].

## 2. SYNTAX AND SEMANTICS OF GENERALIZED TREES

We first present the syntax. Let  $V$  be a set of *tree variables*, with typical elements  $x, y, \dots$  and let  $T$  be the set of trees, with typical elements  $t, \dots$ . Before giving the definition proper, we discuss a preliminary version which, for reasons to be mentioned in a moment, turns out to be insufficient. We use as syntactic formalism a variant of BNF which should be self-explanatory.

DEFINITION 2.0 (to be rejected).

$$t ::= 0 \mid x \mid \langle a, \tau \rangle \mid \mu x[t] \mid \nu x[t]$$

where  $\tau = \{t_1, \dots, t_n\}$  is a finite (possibly empty) set of trees

END 2.0

This definition tells us that a tree is one of

- a) the tree constant 0 (which stands for the undefined tree, and has the empty set of paths as its meaning);
- b) a tree variable  $x$ ;
- c) a tree with root  $a$  and subtrees  $t_1, \dots, t_n$ ;
- d) the *least* solution of the equation (in  $t_0$ )  $t_0 = t[t_0/x]$ ;
- e) the *greatest* solution of the equation  $t_0 = t[t_0/x]$ .

(In clause d and e,  $t[t_0/x]$  denotes the result of substituting  $t_0$  for  $x$  in  $t$ .)

The problem with definition 2.0 is that, e.g., the equation  $t_0 = t_0$  does have a least solution but not a greatest solution. In fact, as least solution (denoted by  $\mu x[x]$ ) we have the tree (also denoted by 0) with associated set of paths  $\emptyset$ , whereas for the greatest solution ( $\nu x[x]$ ) we need a tree such that the associated set of paths is both a tree-set (in the sense of definition 2.3 below) and greatest within the collection of all tree-sets. No such set exists, which explains why we impose a syntactic restriction on the use of the  $\nu x[t]$  formation in that  $t$  has no free occurrences of  $x$ , unless these occurrences are shielded by some intermediate use of the  $\langle a, \tau \rangle$  formation rule. E.g., we do not allow  $\nu x[x]$ , nor  $\nu x[\mu y[x]]$ , but we do allow  $\nu x[\langle a, \{x\} \rangle]$ , or  $\nu x[\langle a, \{\mu y[\langle b, \{x, y\} \rangle] \rangle]$ .

DEFINITION 2.1. Let  $\Sigma$  be the set of *states*, with typical elements  $a, b, \dots, a_1, \dots, a', \dots$ .

For each  $W \subseteq V$  we define a tree  $t_W$  (where  $W$  records the tree variables which do not allow  $\nu$ -formation).

$$t_W ::= 0 \mid x \mid \langle a, \tau \rangle \mid \mu x [t_W] \mid \nu x [t_W \cup \{x\}]$$

- provided that  $x \notin W$
- where  $\tau = \{t_{\emptyset}^1, \dots, t_{\emptyset}^n\}$  is a finite (possibly empty) set of trees with respect to  $\emptyset$

END 2.1.

Below, we simply write  $t$  for  $t_{\emptyset}$ .

EXAMPLES. Possible trees are:  $0$ ,  $x$ ,  $\langle a, \{\langle b, \emptyset \rangle, \langle c, \{\langle d, \emptyset \rangle\} \rangle\} \rangle$ ,  $\mu x [\langle a, \{x\} \rangle]$ ,  $\nu x [\langle a, \{x\} \rangle]$ ,  $\mu x [\langle a, \{x, \langle a, \emptyset \rangle\} \rangle]$ ,  $\nu x [\langle a, \{x, \langle a, \emptyset \rangle\} \rangle]$ ,  $\nu x [\langle a, \{\mu y [\langle b, \{x, y\} \rangle]\} \rangle]$ .

We now define a way of assigning meaning to these constructs. As indicated already, we use sets of (finite or infinite) paths over  $\Sigma$  as meaning of  $t \in T$ . More specifically, we introduce

- $\Sigma$ , the set of states, as before
- $\Sigma^*$ , the set of all finite sequences of elements in  $\Sigma$   
(with  $\varepsilon$  denoting the empty sequence)
- $\Sigma^\omega$ , the set of all infinite sequences of elements in  $\Sigma$ ,  
 $s, \dots$  denote typical elements of  $\Sigma^* \cup \Sigma^\omega$
- $S$ , with typical elements  $\sigma, \sigma'$ , the set of  
all subsets of  $\Sigma^* \cup \Sigma^\omega$ .

The meaning  $T(t)$  of  $t \in T$  is given as an element  $\sigma \in S$ , but for the complication that we need an environment function mapping tree variables in  $V$  to elements in  $S$ , in order to cope with the presence of free tree variables in  $t$ .

DEFINITION 2.2. Let  $\Theta = V \rightarrow S$ , with typical elements  $\theta, \dots$ . Let, for  $\sigma_1, \sigma_2 \in S$ ,  $\sigma_1 \sqsubseteq \sigma_2$  whenever  $\sigma_1 \subseteq \sigma_2$  (where the second ordering is the customary set-theoretic inclusion). Let, for  $\phi_1, \phi_2 \in S \rightarrow S$ ,  $\phi_1 \sqsubseteq \phi_2$  whenever  $\phi_1(\sigma) \sqsubseteq \phi_2(\sigma)$  for all  $\sigma \in S$ . Let, for  $\phi$  a monotonic function, (i.e.,  $\sigma_1 \sqsubseteq \sigma_2 \Rightarrow \phi(\sigma_1) \sqsubseteq \phi(\sigma_2)$ ) from  $S$  to  $S$ ,  $\mu\phi$  denote its least fixed point and  $\nu\phi$  its greatest fixed point. (Both exist because of the Knaster-Tarski theorem.) Let, for each  $\theta \in \Theta$ ,  $x \in V$  and  $\sigma \in S$ ,  $\theta\{\sigma/x\}$  be an element of  $\Theta$  which is defined by:  $\theta\{\sigma/x\}(x) = \sigma$ ,  $\theta\{\sigma/x\}(y) = \theta(y)$  for  $x \neq y$ . Let, for each  $\theta, x$  and  $t$ ,  $\lambda\sigma. T(t)(\theta\{\sigma/x\})$  be that function from  $S$  to  $S$  which, when applied to  $\sigma_0 \in S$ , yields  $T(t)(\theta\{\sigma_0/x\}) \in S$  (i.e.,  $\lambda\sigma \dots$  embodies the customary  $\lambda$ -notation). We put, for each  $\theta \in \Theta$ :

- a)  $T(0)(\theta) = \emptyset$  (the empty subset of  $\Sigma^* \cup \Sigma^\omega$ )
- b)  $T(x)(\theta) = \theta(x)$
- c)  $T(\langle a, \tau \rangle)(\theta) = aT(\tau)(\theta)$
- d)  $T(\mu x [t])(\theta) = \mu [\lambda\sigma. T(t)(\theta\{\sigma/x\})]$

$$e) \quad T(\nu x[t])(\theta) = \nu[\lambda \sigma. T(t)(\theta\{\sigma/x\})]$$

where, in clause  $c$ ,  $T(\tau)(\theta)$  is defined as  $T(\tau)(\theta) = \bigcup_{t \in \tau} T(t)(\theta)$  ( $\tau \neq \emptyset$ ), and  $T(\emptyset)(\theta) = \{\epsilon\}$ . Moreover, we have, as usual, that  $a\sigma = \{as \mid s \in \sigma\}$ .

END 2.2.

EXAMPLES. Choose any  $\theta$ .  $T(\emptyset)(\theta) = \emptyset$ ,  $T(x)(\theta) = \theta(x)$ ,  $T(\langle a, \{\langle b, \emptyset \rangle, \langle c, \{\langle d, \emptyset \rangle\} \rangle \rangle})(\theta) = \{ab, acd\}$ ,  $T(\mu x[\langle a, \{x\} \rangle])(\theta) = \emptyset$ ,  $T(\nu x[\langle a, \{x\} \rangle])(\theta) = \{a^\omega\}$ ,  $T(\mu x[\langle a, \{x, \langle a, \emptyset \rangle\} \rangle])(\theta) = \{aa, aaa, \dots\}$ ,  $T(\nu x[\langle a, \{x, \langle a, \emptyset \rangle\} \rangle])(\theta) = \{aa, aaa, \dots, a^\omega\}$ ,  $T(\nu x[\langle a, \{\mu y[\langle b, \{x, y\} \rangle] \rangle])(\theta) = (ab^+)^{\omega}$ .

DEFINITION 2.3. A set of sequences  $\sigma \in S$  is called a *tree-set* whenever the following two conditions are satisfied:

- Each pair  $s_1, s_2 \in \sigma$  has a finite non-empty sequence  $s$  as common prefix.
- For each finite non-empty sequence  $s$ , the set of all sequences in  $\sigma$  which have  $s$  as maximal common prefix is finite.

END 2.3.

EXAMPLES. The set  $\{a, b\}$  is not a tree-set, because it violates condition a, and the set  $\{aa_1, aa_2, \dots\}$  is not a tree-set, because it violates condition b.

The following proposition is stated here without proof:

PROPOSITION 2.4. For each  $t \in T$  and each  $\theta \in \Theta$  such that  $\theta(x)$  is a tree-set for all  $x \in V$ , we have that  $T(t)(\theta)$  is a tree-set.

END 2.4.

## 2. SYNTAX AND SEMANTICS OF INFINITE PROCESSES

Let  $A$  be a set of elementary actions, with typical elements  $A, A_1, \dots$ , and  $X$  a set of statement variables, with typical elements  $X, X_1, \dots$ . We define the class of statements *Stat*, with typical elements  $S, \dots$ , in

DEFINITION 3.1.

$$S ::= A \mid X \mid S_1 ; S_2 \mid S_1 \cup S_2 \mid \mu X[S] \mid \nu X[S]$$

END 3.1.

Here,  $S_1 \cup S_2$  indicates a nondeterministic choice between  $S_1$  and  $S_2$ . Also,  $\mu X[S]$  corresponds to a parameterless recursive procedure declared in an ALGOL 60-like language by `proc P; S[P/X]`. Moreover,  $\nu X[S]$  extends  $\mu X[S]$  in that possible infinite computations are also taken into account. E.g., the two procedures mentioned in the introduction are written in this syntax as  $\mu X[A; X \cup A]$  and  $\nu X[A; X \cup A]$ .



Now let  $M = T \rightarrow T$ , with, for  $\phi_1, \phi_2 \in M$ ,  $\phi_1 \subseteq \phi_2$  iff  $\phi_1(t) \subseteq \phi_2(t)$  for all  $t \in T$  (where  $t_1 \subseteq t_2$  iff  $T(t_1)(\theta) \subseteq T(t_2)(\theta)$  for all  $\theta \in \Theta$ ). Let, moreover,  $\Lambda$  be the set of all *finite* subsets of  $\Sigma$ , and let  $\Gamma = (X \rightarrow M) \cup (A \rightarrow (\Sigma \rightarrow \Lambda))$ . (The first part of each  $\gamma \in \Gamma$  gives some (arbitrary) meaning to statement variables, the second part tells us how elementary actions  $A \in A$  transform  $a \in \Sigma$  to  $\{a_1, \dots, a_n\} (n \geq 0) \in \Lambda$ . I.e., elementary actions are of bounded nondeterminacy.) We now give the definition of the meaning of a statement.

**DEFINITION 3.2.**  $M: Stat \rightarrow (\Gamma \rightarrow M)$  is given by

a) For each  $t$  which is not of the form  $\langle a, \emptyset \rangle$ :

$$M(S)(\gamma)(0) = 0, \quad M(S)(\gamma)(x) = x, \quad M(S)(\gamma)(\langle a, \tau \rangle) = \langle a, M(S)(\gamma)(\tau) \rangle \quad (\tau \neq \emptyset),$$

$$M(S)(\gamma)(\mu x[t]) = \mu x[M(S)(\gamma)(t)], \quad M(S)(\gamma)(\nu x[t]) = \nu x[M(S)(\gamma)(t)].$$

$$\text{(Here, } M(S)(\gamma)(\tau) = \bigcup_{t \in \tau} M(S)(\gamma)(t)\text{.)}$$

b) For  $t = \langle a, \emptyset \rangle$  we use induction on the structure of  $S$ :

$$1. \quad M(A)(\gamma)(\langle a, \emptyset \rangle) = \begin{cases} \langle a, \{\langle a_i, \emptyset \rangle \mid a_i \in \gamma(A)(a)\} \rangle, & \text{if } \gamma(A)(a) \neq \emptyset \\ \langle a, \{0\} \rangle, & \text{otherwise} \end{cases}$$

$$2. \quad M(X)(\gamma)(\langle a, \emptyset \rangle) = \gamma(X)(\langle a, \emptyset \rangle)$$

$$3. \quad M(S_1; S_2)(\gamma)(\langle a, \emptyset \rangle) = M(S_2)(\gamma)(M(S_1)(\gamma)(\langle a, \emptyset \rangle))$$

$$4. \quad M(S_1 \cup S_2)(\gamma)(\langle a, \emptyset \rangle) = \langle a, \{M(S_1)(\gamma)(\langle a, \emptyset \rangle), M(S_2)(\gamma)(\langle a, \emptyset \rangle)\} \rangle$$

$$5. \quad M(\mu X[S])(\gamma)(\langle a, \emptyset \rangle) = \mu[\lambda \phi. M(S)(\gamma\{\phi/X\})](\langle a, \emptyset \rangle)$$

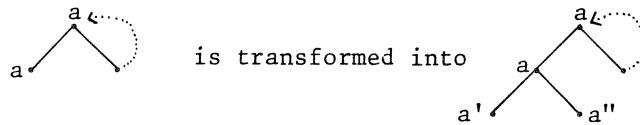
$$6. \quad M(\nu X[S])(\gamma)(\langle a, \emptyset \rangle) = \nu[\lambda \phi. M(S)(\gamma\{\phi/X\})](\langle a, \emptyset \rangle)$$

END 3.2.

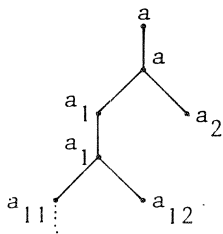
**EXAMPLES.** Let  $\gamma(A)(a) = \{a', a''\}$ ,  $\gamma(A_1)(a) = \{a_1\}$ ,  $\gamma(A_2)(a) = \{a_2\}$ .

$$1. \quad M(A)(\gamma)(\mu x[\langle a, \{x, \langle a, \emptyset \rangle\} \rangle]) = \mu x[M(A)(\gamma)(\langle a, \{x, \langle a, \emptyset \rangle\} \rangle)] = \mu x[\langle a, \{M(A)(\gamma)(x), M(A)(\gamma)(\langle a, \emptyset \rangle)\} \rangle] = \mu x[\langle a, \{x, \langle a, \{\langle a', \emptyset \rangle, \langle a'', \emptyset \rangle\} \rangle\} \rangle].$$

Thus,



$$2. \quad M(\mu X[A_1; X \cup A_2])(\gamma)(\langle a, \emptyset \rangle) \text{ and } M(\nu X[A_1; X \cup A_2])(\gamma)(\langle a, \emptyset \rangle)$$



yield this tree with the infinite path

$a \ a_1 \ a_1 \ a_{11} \ a_{11} \ \dots$  excluded in the  $\mu$ -case and included in the  $\nu$ -case.

## REFERENCES

1. Hitchcock, P. and D.M.R. Park, Induction rules and proofs of termination, Proc. 1<sup>st</sup> ICALP (M. Nivat, ed.), pp. 225-251, 1973, North-Holland, Amsterdam.
2. Mazurkiewicz, A., Proving properties of processes, CC PAS reports 134, Warsaw, 1973.
3. De Roever, W.P., Maximal fixed points solve some of the problems with Scott induction, Proc. 4<sup>th</sup> ICALP, to appear.
4. Scott, D. and C. Strachey, Towards a mathematical semantics for computer languages, Proc. Symp. Computers and Automata (J. Fox, ed.), pp. 19-46, 1971, Polytechnic Institute of Brooklyn.
5. Milne, R. and C. Strachey, A Theory of Programming Language Semantics, Chapman & Hall, 1976.