stichting
mathematisch
centrum

AFDELING INFORMATICA IW 6/73 SEPTEMBER

W.P. DE ROEVER

OPERATIONAL AND MATHEMATICAL SEMANTICS
FOR RECURSIVE POLYADIC PROGRAM SCHEMATA
(Extended abstract)

2e boerhaavestraat 49 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), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

ACM - Computing Reviews - category: 5.24

Operational and mathematical semantics for recursive polyadic program schemata *)

W.P. de Roever

Abstract

The language *PL* for first-order recursive program schemes with call-by-value as parameter mechanism is described using models for sequential and independent parallel computation. The language *MU* for binary relations over cartesian products which has minimal fixed point operators is defined. An injection between *PL* and *MU* is specified together with the conditions subject to which this injection induces a translation.

MU is axiomatized using a many-sorted generalization of TARSKI's axioms for binary relations, SCOTT's induction rule and fixed point axiom and new axioms to characterize projection functions, whence by the translation result a *calculus for first-order program schemes* is obtained.

*) Reprinted from Proceedings of Symposium and Summer School "Mathematical Foundations of Computer Science", 3-8 September 1973, High Tatras, Czechoslovakia, pp. 293-298.

OPERATIONAL AND MATHEMATICAL SEMANTICS FOR
RECURSIVE POLYADIC PROGRAM SCHEMATA

W.P. de Roever
Mathematisch Centrum

1. First we define *PL*, a language for recursive polyadic program schemata.
These schemata are abstractions of certain classes of programs. The statements contained in these programs operate upon a state, whose components can be isolated by means of *projection* functions; a new state is obtained by (1) execution of *elementary* statements, the *dummy* statement or projection functions (2) calls of previously declared and possibly *recursive* procedures P_j (3) execution of conditional statements $(p \to S_1, S_2)$ (4) the parallel and independent execution of statements $S_1 \ldots S_n$ in the *call-by-value product* $[S_1,\ldots,S_n]$, a new construct which unifies properties of the *assignment* statement and the call-by-value *parameter mechanism* and allows for the expression of both of these concepts (5) composition of statements. A *declaration* is a possibly empty collection of pairs $P_j \Leftarrow S_j$ which are indexed by some index set J; for each $j \in J$ such a pair contains a *procedure* symbol P_j and a statement S_j. A *program* is a pair consisting of a declaration and a statement. By abstracting from the particular meanings of elementary statements, predicates and constants one obtains *statement schemes, declaration schemes* and *program schemes*.
The definition of the *operational* semantics of these schemes involves an abstraction from the actual processes taking place within a computer by describing a *model* for the computations evoked by execution of a program. The main problem in defining this model is the fact that the computations involved cannot be represented serially in any natural fashion: factors $S_1 \ldots S_n$ of a product $[S_1,\ldots,S_n]$ first all have to be executed independent of each other, before computation can continue. Therefore the computations involved are described as a parallel and sequentially structured hierarchy of actions, a *computation model*, which is defined below.
Let $\mathcal{O}_0$ be an *initial* interpretation, i.e., an interpretation of the elementary statement symbols, predicate symbols and constant symbols of *PL*, and D be a declaration scheme. Then a computation model for x S y is a pair

$$< x_1 S_1 x_2 \ldots x_n S_n x_{n+1},\ CM >$$

where S_i is, for $i=1,\ldots,n$, a statement scheme, $S_1 = S$, $x = x_1$ and $x_{n+1} = y$, consisting of a *computation sequence* and a *set* of computation models (relative to $\mathcal{O}_0$ and D), which satisfy the following conditions:

a. If $S_i = R$ or $S_i = R;S'$ with R an elementary statement symbol or constant symbol, then $<x_i, x_{i+1}> \in \mathcal{O}_0(R)$ and $i = n$ or $S_{i+1} = S'$.

b. If $S_i = P_j$ or $S_i = P_j;S'$ and $(P_j \Leftarrow S_j) \in D$, then $x_{i+1} = x_i$ and $S_{i+1} = S_j$ or $S_{i+1} = S_j;S'$.

c. If $S_i = (p \to S', S'')$ or $S_i = (p \to S', S'');S'''$ and $\mathcal{O}_0(p)(x_i)$ is either <u>true</u> or <u>false</u>, then $x_{i+1} = x_i$ and, if $\mathcal{O}_0(p)(x_i)$ = <u>true</u> then $S_{i+1} = S'$ or $S_{i+1} = S';S'''$, and, if $\mathcal{O}_0(p)$ = <u>false</u> then $S_{i+1} = S''$ or $S_{i+1} = S'';S'''$.

d. If $S_i = [V_1,\ldots,V_k]$ or $S_i = [V_1,\ldots,V_k];S'$, then $x_{i+1} = <y_1,\ldots,y_k>$, where *CM* contains computation models for $x_i V_i y_l$, for $l = 1,\ldots,k$, and $i = n$ or $S_{i+1} = S'$.

This definition leads to the characterization of the *input-output behaviour* or *operational* semantics $\mathcal{O}(T)$ of the program scheme $T = \langle D,S\rangle$, in terms of which *correctness* criteria for T can be formulated.
The main technical result of this part is the *union* theorem (cf. DE BAKKER and MEERTENS [2a]): Consider the simultaneous declaration of recursive procedures $P_1\ldots P_n$ with bodies $S_1\ldots S_n$, respectively; for $j = 1,\ldots,n$, S_j contains occurrences of $P_1\ldots P_n$, whence we write $S_j(P_1,\ldots,P_n)$ for purposes of substitution. Then this theorem states that

$$P_j = \bigcup_{i=0}^{\infty} \mathcal{O}(S_j^i),\text{ with } S_j^i \text{ defined by } S_j^0 = \Omega,\text{ the \textit{undefined} statement scheme, and}$$

$$S_j^{i+1} = S_j(S_1^i,\ldots,S_n^i),\; j = 1\ldots n.$$

2. Next we define *MU*, a language for binary relations over cartesian products, which has *minimal fixed point operators* in order to characterize the input-output behaviour of recursive programs.
As the binary relations considered are subsets of the cartesian product of one domain or cartesian product of domains and another domain or cartesian product of domains, terms denoting these relations have to be *typed* in order to define operations on them. On account of limitations of space types will not be mentioned or discussed unless explicitly needed; we refer the interested reader to DE ROEVER [8] for a more extensive account.
Elementary terms are individual relation constants, boolean relation constants, logical relation constants (for the empty, identity, and universal relations Ω, E, U and projection functions π_i) and relation variables.
Compound terms are constructed by means of the operators ";" (relational or Peirce product), "$\cup$" (union), "$\cap$" (intersection), "$\smile$" (converse) and "$-$" (complementation) and the minimal fixed point operators "μ_i", which bind for $i = 1,\ldots,n$, n different relation variables in n-tuples of terms provided *none of these variables occurs in any complemented subterm*, i.e., these terms are *syntactically continuous* in these variables.
Terms of *MU* are elementary or compound terms.
The well-formed formulae of *MU* are called *assertions* and are of the form $\Phi \vdash \Psi$, where Φ and Ψ are sets of inclusions between terms.
The *mathematical* semantics *m* of *MU* is defined by:

(1) providing arbitrary (type-consistent) interpretations for the individual relation constants and relation variables, interpreting pairs $\langle p,p'\rangle$ of boolean relation constants as pairs $\langle m(p),m(p')\rangle$ of disjoint subsets of identity relations (cf. KARP [5]) and interpreting the logical relation constants as empty, identity and universal relations and projection functions,
(2) interpreting ";", "$\cup$", "$\cap$", "$\smile$", "$-$" as usual,
(3) interpreting μ-terms $\mu_i X_1\ldots X_n\,[\sigma_1\ldots\sigma_n]$ as the i-th component of the minimal fixed point of the functional $\langle\sigma_1\ldots\sigma_n\rangle$ acting on n-tuples of relations.

An assertion $\Phi \vdash \Psi$ is *valid* provided for all *m* the following holds:
If the inclusions contained in Φ are satisfied by *m*, then the inclusions contained in Ψ are satisfied by *m*.

The main technical result concerning *MU* is again a union theorem:

$$m(\mu_i X_1\ldots X_n[\sigma_1\ldots\sigma_n]) = \bigcup_{j=0}^{\infty} m(\sigma_i^j),\quad i = 1,\ldots,n,$$

with σ_i^j similarly defined as S_i^j (see section 1). In the proof of this theorem the *semantic* continuity of the terms $\sigma_1,\ldots,\sigma_n$, which follows from their *syntactic* continuity, plays an important role. One of the implications of this theorem is the *validity* of *Scott's induction rule*, to be defined in section 5.4.

3. The precise correspondence between the operational semantics $\mathcal{O}$ of *PL* and the mathematical semantics m of *MU* is specified by the translation theorem of chapter 3 of DE ROEVER [8]: After defining an injection tr between program schemes and terms (see the table below) we prove that tr induces a *meaning preserving* mapping, i.e., a *translation*, provided the interpretation of the elementary statement constants and predicate symbols specified by $\mathcal{O}$ "agrees" with the interpretation of the individual relation constants and boolean relation constants specified by m.
If these requirements are fulfilled, the resulting correspondence between *PL* and *MU* is illustrated by

$$\begin{tikzcd} T \arrow[r] \arrow[d, dashed] & tr(T) \arrow[d, dashed] \\ \mathcal{O}(T) \arrow[r, phantom, "="] & m(tr(T)) \end{tikzcd}$$

Thus we conclude that, in order to prove properties of T, it is sufficient to prove properties of $tr(T)$, whence the axiomatization of *MU* in section 5 below leads to a calculus for recursive polyadic program schemes.

The definition of tr is given below, with arguments to the left and images to the right:

PL		*MU*
Elementary statement A	...	Individual relation constant A
Dummy statement	...	Identity relation E
Projection function π_i	...	π_i
$S_1;S_2$	...	$tr(S_1);tr(S_2)$
$(p \rightarrow S_1,S_2)$	...	$p;tr(S_1) \cup p';tr(S_2)$
$[S_1,\ldots,S_n]$	...	$tr(S_1);\check{\pi}_1 \cap\ldots\cap tr(S_n);\check{\pi}_n$
P_i, relative to a declaration scheme $\{P_j \Leftarrow S_j\}_{j=1\ldots n}$, for $i = 1,\ldots,n$.	...	$\mu_i X_1\ldots X_n[tr(S_1(X_1,\ldots,X_n))\ldots tr(S_n(X_1,\ldots,X_n))]$, where $tr(S_j(X_1,\ldots,X_n))$ denotes the image of S_j under tr, with occurrences of $P_1\ldots P_n$ replaced by $X_1\ldots X_n$, respectively, for $i,j = 1,\ldots,n$.

4. In [6] MANNA and VUILLEMIN *discard* call-by-value as a computation rule, because, in their opinion, it does not lead to computation of the *minimal* fixed point. Clearly, our translation theorem *invalidates* their conclusion. As it happens, they work with a formal system in which minimal fixed points coincide with recursive solutions computed with *call-by-name* as rule of computation. Quite correctly they observe that *within such a system* call-by-value does not necessarily lead to computation of minimal fixed points. We may point out that observations like this one hardly justify discarding call-by-value as rule of computation *in general*.

5. The axiomatization of *MU* proceeds in four successive stages:

5.1. *Axiomatization of typed binary relation constants*

Consider the following sublanguage of *MU*, called MU_0:

The *elementary* terms of MU_0 are restricted to the individual relation constants, relation variables and logical constants Ω, E and U of *MU*, i.e., boolean constants and projection functions are excluded.

The *compound* terms of MU_0 are those terms of *MU* which are constructed using these elementary terms and the ";", "$\cup$", "$\cap$", "$\smile$" and "—" operators, i.e., the "μ_i" operators are excluded.

The *assertions* of MU_0 are the assertions of MU containing inclusions between terms of MU_0. MU_0 is axiomatized by the following axioms (greek superscripts denoting types):

a. The typed versions of the axioms of boolean algebra.

b. The typed versions of Tarski's axioms for binary relations (cf. [10]):

$$T_1 : \vdash (X^{\eta,\theta};Y^{\theta,\zeta});Z^{\zeta,\xi} = X^{\eta,\theta};(Y^{\theta,\zeta};Z^{\zeta,\xi})$$

$$T_2 : \vdash \breve{\breve{X}}^{\eta,\xi} = X^{\eta,\xi}$$

$$T_3 : \vdash (X^{\eta,\theta};Y^{\theta,\xi})^{\smile} = \breve{Y}^{\theta,\xi};\breve{X}^{\eta,\theta}$$

$$T_4 : \vdash X^{\eta,\xi};E^{\xi,\xi} = X^{\eta,\xi}$$

$$T_5 : (X^{\eta,\theta};Y^{\theta,\xi}) \cap Z^{\eta,\xi} = \Omega^{\eta,\xi} \vdash (Y^{\theta,\xi};\breve{Z}^{\eta,\xi}) \cap \breve{X}^{\eta,\theta} = \Omega^{\theta,\eta}$$

c. $$U : \vdash U^{\eta,\xi} \subseteq U^{\eta,\theta};U^{\theta,\xi}$$

The introduction of axiom U is necessitated by the introduction of types (otherwise is $\vdash U^{\eta,\xi} = U^{\eta,\theta};U^{\theta,\xi}$ no longer provable).

In addition to (1) the results of TARSKI [10], properties such as

(2) $\vdash X;Y \cap Z = X;(\breve{X};Z \cap Y) \cap Z$, and

(3) $\vdash X = (X;U \cap E);X$, $\vdash X;U \cap E = X;\breve{X} \cap E$, $\vdash X;U = (X;U \cap E);U$ and

$X \subseteq Y,\ \breve{Y};Y \subseteq E \vdash (X;U \cap E);Y = X$

can be proved using these axioms (cf. DE BAKKER and DE ROEVER [2]).

5.2. *Axiomatization of boolean relation constants*

MU_0 is extended to MU_1 by adding the boolean relation constants of MU to the basic terms of MU_0.

MU_1 is axiomatized by adding the following axioms to those of MU_0:

$$P_1 : \vdash p^{\eta,\eta} \subseteq E^{\eta,\eta},\ p'^{\eta,\eta} \subseteq E^{\eta,\eta}$$

$$P_2 : \vdash p^{\eta,\eta} \cap p'^{\eta,\eta} = \Omega^{\eta,\eta}$$

The translation theorem implies $O(p \to S_1,S_2) = m(p;tr(S_1) \cup p';tr(S_2))$, provided $O(p)$ is represented by $\langle m(p),m(p')\rangle$. Thus leads axiomatization of MU_1 to a theory of conditionals, e.g., the usual axioms for conditionals, cf. McCARTHY [7], can be derived.

As first consequence of P_1 and P_2, one obtains

(4) $\vdash p = p;p,\ p;q = p \cap q$.

In expressing correctness properties of programs frequently the following operator is used: $X \circ p \underset{DEF}{=} X;p;U \cap E$. The properties of this operator are collected in

(5) $\vdash (X;Y)\circ p = X\circ(Y\circ p)$, $\vdash (X\cup Y)\circ p = X\circ p \cup Y\circ p$, $\vdash (X\cap Y)\circ p = X;p;\breve{Y} \cap E$, $\vdash X;p \subseteq X\circ p;X$,

$\breve{X};X \subseteq E \vdash X;p = X\circ p;X$ and $X;p \subseteq q;X \vdash X\circ p \subseteq q$

and proved in DE ROEVER [8].

5.3. *Axiomatisation of binary relations over cartesian products*

The language MU_2 for binary relations over cartesian products is obtained from MU_1 by adding, for $i = 1,\ldots,n$, projection function symbols
$\pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i}$ to the basic terms of MU_1. (A term with superscript $\langle\eta_1\times\ldots\times\eta_n,\theta_1\times\ldots\times\theta_m\rangle$ is interpreted as a subset of $(D_{\eta_1}\times\ldots\times D_{\eta_n})\times(D_{\theta_1}\times\ldots\times D_{\theta_m})$, where D_{η_i} and D_{θ_j} are domains of type η_i and θ_j, respectively, for $i = 1,\ldots,n$, $j = 1,\ldots,m$).
MU_2 is axiomatized by adding the following two axiom schemes to the axioms of MU_1:

$$C_1 : \vdash \pi_1;\breve{\pi}_1 \cap\ldots\cap \pi_n;\breve{\pi}_n = E^{\eta_1\times\ldots\times\eta_n,\eta_1\times\ldots\times\eta_n}$$

$$C_2 : \vdash X_1;Y_1 \cap\ldots\cap X_n;Y_n = (X_1;\breve{\pi}_1 \cap\ldots\cap X_n;\breve{\pi}_n);(\pi_1;Y_1 \cap\ldots\cap \pi_n;Y_n).$$

An assignment $x_i := f(x_1\ldots x_n)$ is modelled by a program scheme $[\pi_1,\ldots,\pi_{i-1},S,\pi_{i+1},\ldots,\pi_n]$.
The translation theorem implies that
$\mathcal{O}([\pi_1,\ldots,\pi_{i-1},S,\pi_{i+1},\ldots,\pi_n]) = m(\pi_1;\breve{\pi}_1 \cap\ldots\cap \pi_{i-1};\breve{\pi}_{i-1} \cap \mathit{tr}(S);\breve{\pi}_i \cap \pi_{i+1};\breve{\pi}_{i+1} \cap\ldots\cap \pi_n;\breve{\pi}_n)$.
Thus leads the axiomatization of MU_2 to a theory of assignments. It can be verified that this class of assignments coincides with the class of assignments described by HOARE in [4].
Axioms C_1 and C_2 imply the following new results:

(6) $\vdash \pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i} \circ E^{\eta_i,\eta_i} = E^{\eta_1\times\ldots\times\eta_n,\eta_1\times\ldots\times\eta_n}$, $\vdash \pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i};U^{\eta_i,\xi} = U^{\eta_1\times\ldots\times\eta_n,\xi}$,

$\vdash \breve{\pi}_i^{\eta_i,\eta_1\times\ldots\times\eta_n};\pi_i^{\eta_1\times\ldots\times\eta_n,\eta_i} = E^{\eta_i,\eta_i}$,

$\vdash \breve{\pi}_i^{\eta_i,\eta_1\times\ldots\times\eta_n};\pi_j^{\eta_1\times\ldots\times\eta_n,\eta_j} = U^{\eta_i,\eta_j}$, for $i \neq j$, $i,j = 1,\ldots,n$.

(7) For $k,l \leq n$:

$$\vdash X_{i_1}\circ E;\ldots;X_{i_k}\circ E;\Big(\bigcap_{\substack{i_j=s_t,\,j=1\ldots k,\\ t=1\ldots l}} X_{i_j};Y_{s_t}\Big);\breve{Y}_{s_1}\circ E;\ldots;\breve{Y}_{s_l}\circ E = \Big(\bigcap_{j=1}^{k} X_{i_j};\breve{\pi}_{i_j}\Big);\Big(\bigcap_{t=1}^{l} \pi_{s_t};Y_{s_t}\Big).$$

5.4. *Axiomatisation of the "μ_i" operators*

MU is obtained from MU_2 by introducing the minimal fixed point operators "μ_i", and axiomatized by adding *Scott's induction rule* I, formulated for the first time in SCOTT and DE BAKKER [9], and axiom M to the axioms and rules of MU_2:

$$I : \quad \frac{\Phi \vdash \Psi(\Omega^{\eta_1,\xi_1},\ldots,\Omega^{\eta_n,\xi_n}) \qquad \Phi,\Psi \vdash \Psi(\sigma_1^{\eta_1,\xi_1},\ldots,\sigma_n^{\eta_n,\xi_n})}{\Phi \vdash \Psi(\mu_1X_1\ldots X_n[\sigma_1\ldots\sigma_n],\ldots,\mu_nX_1\ldots X_n[\sigma_1\ldots\sigma_n]).}$$

with Φ only containing occurrences of X_i which are *bound*, i.e., contained in (sub)terms $\mu_k\ldots X_i\ldots[\ldots\tau_i\ldots]$, and Ψ only containing occurrences of X_i which are *not* complemented, $i=1,\ldots,n$.

$$M : \vdash \{\sigma_j(\mu_1X_1\ldots X_n[\sigma_1\ldots\sigma_n],\ldots,\mu_nX_1\ldots X_n[\sigma_1\ldots\sigma_n]) \subseteq \mu_jX_1\ldots X_n[\sigma_1\ldots\sigma_n]\}_{j=1\ldots n}$$

Now properties such as ***monotonicity*** of terms and the *fixed point property* (cf. SCOTT and DE BAKKER [9]), the ***minimal fixed point property*** and *iteration* (cf. HITCHCOCK and PARK [3]), and ***modularity*** (cf. DE ROEVER [8]) can be proved.

6. The calculus for recursive polyadic program schemata can be applied to the axiomatic characterization of recursive data structures such as the natural numbers, lists, linear lists and ordered linear lists (cf. DE ROEVER [8]), strings of symbols (cf. DE BAKKER [1]) and trees (cf. DE BAKKER and DE ROEVER [2]). Also finite domains with a fixed number of elements can be characterized. Numerous properties of both recursive schemata, such as the regularization of linear recursive schemata (cf. WRIGHT [11]), and recursive data structures and schemata manipulating these structures can be deduced, culminating in a correctness proof for a schema of the TOWERS OF HANOI (cf. DE ROEVER [8]).

7. *References.*

[1] de Bakker, J.W., Recursion, induction and symbol manipulation, in Proc. MC-25 Informatica Symposium, Mathematical Centre Tracts 37, Mathematical Centre, Amsterdam.

[2] de Bakker, J.W. and de Roever, W.P., A calculus for recursive program schemes, in Automata, Languages and Programming, M. Nivat (editor), North-Holland, Amsterdam.

[2a] de Bakker, J.W. and Meertens, L.G.L.Th., Simple recursive program schemes and inductive assertions, MC Report 142, Mathematisch Centrum, Amsterdam.

[3] Hitchcock, P. and Park, D., Induction rules and termination proofs. (In same edition as [2]).

[4] Hoare, C.A.R., An axiomatic basis for computer programming, Comm. ACM 12, pp. 576-583 (1969).

[5] Karp, R.M., Some applications of logical syntax to digital computer programming, Thesis Harvard University.

[6] Manna, Z. and Vuillemin, J.M., Fix-point approach to the theory of computation, Comm. ACM 15, pp. 528-536 (1972).

[7] McCarthy, J., A basis for a mathematical theory of computation, in Computer Programming and Formal Systems, pp. 33-70 (eds. P. Braffort and P. Hirschberg), North-Holland, Amsterdam.

[8] de Roever, W.P., Operational and axiomatized semantics for first-order recursive program schemes. Mathematical Centre Report, Mathematical Centre, Amsterdam (forth coming).

[9] Scott, D. and de Bakker, J.W., A theory of programs, Unpublished notes.

[10] Tarski, A., On the calculus of relations, J. Symbolic Logic, 6, pp. 73-89.

[11] Wright, J.B., Characterization of recursively enumerable sets, Report RC 3419, IBM Research Centre Yorktown Heights.

Acknowledgement: I wish to thank J.W. de Bakker, for his continuous help, advice and criticism, P. van Emde Boas, whose mastery of abstract mathematics was a valuable help, and P. Vitanyi, for his contributions to the intelligibility of this paper, such as it is.