stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 133/80 APRIL

J.A. BERGSTRA & J.V. TUCKER

A NATURAL DATA TYPE WITH A FINITE EQUATIONAL FINAL SEMANTICS SPECIFICATION BUT NO EFFECTIVE EQUATIONAL INITIAL SEMANTICS SPECIFICATION

Preprint

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).

1980 Mathematics subject classification: 03D45 03D80 68B15

ACM - Computing Review-categories: 4.34

A natural data type with a finite equational final semantics specification but no effective equational initial semantics specification*

by

J.A. Bergstra** & J.V. Tucker

ABSTRACT

Initial and final algebra semantics are two ways of assigning a unique meaning to an axiomatic specification (Σ,E) of a data type. First, we point out how easy it is to find *natural* data types with effective specifications with respect to initial algebra semantics, but without effective specifications with respect to final algebra semantics. Secondly, we suggest that a *natural* source of data types for which the opposite is true are those programming systems with undecidable program equivalence problem. We work out in detail the situation when the denotational semantics of a system are the primitive recursive functions.

KEY WORDS & PHRASES: *Algebraic data types, algebraic specifications, initial algebra semantics, final algebra semantics, primitive recursive functions*

---

* This report will be submitted for publication elsewhere.

** Department of Computer Science, University of Leiden, Wassenaarseweg 80, Postbus 9512, 2300 RA LEIDEN, The Netherlands.

INTRODUCTION

Suppose you want to define a data type by a set of operators Σ satisfying some axioms E. Initial and final algebra semantics are two natural ways of assigning to the specification (Σ,E) a unique meaning in the class $ALG(\Sigma,E)$ of all Σ-algebras satisfying the properties of E. Initial semantics insists that two terms t,t' over Σ are identical iff t,t' can be *proved* equal from axioms E while final semantics agrees to identify t,t' as long as $t = t'$ is *consistent* with the axioms E. Both techniques have been discussed in the programming methodology and theoretical literatures with varying degrees of partiality: we assume the reader is aware of at least ADJ[7], BROY *et al* [6], GUTTAG & HORNING[8], KAMIN[9], WAND[13]. Here we wish to point out a pleasing mathematical symmetry: *if* (Σ,E) *is a specification in which* E *is an r.e. set of equations then the initial semantics of* (Σ,E) *is an r.e. semantics while the final semantics of* (Σ,E) *is a co-r.e. semantics. So a data type possessing effective specifications with respect to both initial and final algebra semantics must be computable.* (A more formal statement of this is Basic Lemma 2.1.)

Clearly, it is easy to find natural data types which fail to possess effective equational final algebra specifications for algebras with r.e., but not recursive, word problems abound. For natural systems with co-r.e., but not recursive, equality problems we look to the denotational semantics of those program languages where the program equivalence problem is undecidable but co-r.e. The easiest example is $\mathcal{PR}$ the unary primitive recursive functions PR on the natural numbers ω (because as a function algebra, made from the usual operators on PR and ω, it is a total algebra). In §3 we organise $\mathcal{PR}$ into a 2-sorted algebra A and prove it can be specified by finitely many equations and hidden operators with respect to final semantics. In §4 we present, as a curio, an initial specification of an impoverished fragment of A.

This little paper introduces final algebra semantics into our series of mathematical studies of the power of definition and adequacy of algebraic methods for data type definition [1,2,3,4], see also [5]. We would like to thank G. Rozenberg for encouraging us to write down these notes.

1. DATA TYPE SPECIFICATIONS

We assume the reader accustomed to working with many-sorted algebras and record here only terminology not to be found in the standard reference ADJ[7].

Let A be a many-sorted algebra. Then A is *minimal* if it is finitely generated by elements named in its signature Σ. All signatures are assumed finite, but not all algebras are minimal. By a *unit congruence* on A we mean a congruence which identifies all elements in one domain of A. Let $S \subset A \times A$. By $\equiv_{\min(S)}$ we denote the smallest congruence on A containing the identifications of S. By $\equiv_{\max(S)}$ we denote the largest congruence on A containing S which is *not* a unit congruence, *if such exists*, and otherwise we take $\equiv_{\max(S)}$ to be a unit congruence. The word "largest" in this context means that if $\equiv$ is any congruence, except a unit congruence, containing S then $\equiv$ is contained in $\equiv_{\max(S)}$.

Let Σ be a signature. A set of equations E over Σ determines a set of basic identifications D(E) between elements of the term algebra $T(\Sigma)$. Let $T_I(\Sigma,E)$ be $T(\Sigma)/\equiv_{\min(E)}$ where $\equiv_{\min(E)}$ abbreviates $\equiv_{\min(D(E))}$ and let $T_F(\Sigma,E)$ be $T(\Sigma)/\equiv_{\max(E)}$ where $\equiv_{\max(E)}$ abbreviates $\equiv_{\max(D(E))}$.

The pair (Σ,E) is said to be a *finite equational specification* of algebra A with respect to (1) *initial algebra semantics* or (2) *final algebra semantics* if E is a finite set of equations over Σ and (1) $T_I(\Sigma,E) \cong A$ or (2) $T_F(\Sigma,E) \cong A$.

We now define our favoured method of making hidden function specifications.

Let A be a many-sorted algebra of signature Σ_A and let Σ be a signature $\Sigma \subset \Sigma_A$ having the same sorts as Σ_A. Then we mean by

$A|_\Sigma$ the Σ-algebra whose domains are those of A and whose operations and constants are those of A named in Σ: the Σ-*reduct* of A; and by

$\langle A\rangle_\Sigma$ the Σ-subalgebra of A generated by the operations and constants of A named in Σ *viz* the smallest Σ-subalgebra of $A|_\Sigma$.

The pair (Σ,E) is said to be a *finite equational hidden enrichment specification* of algebra A with respect to (1) *initial algebra semantics* or (2) *final algebra semantics* if $\Sigma_A \subset \Sigma$ and Σ contains exactly the sorts of Σ_A, and E is a finite set of equations over Σ such that

$$\text{(1)}\quad T_I(\Sigma,E)|_{\Sigma_A} = \langle T_I(\Sigma,E)\rangle_{\Sigma_A} \cong A$$

$$\text{or (2)}\quad T_F(\Sigma,E)|_{\Sigma_A} = \langle T_F(\Sigma,E)\rangle_{\Sigma_A} \cong A.$$

2. COMPUTABLE DATA TYPE SEMANTICS

Any countable many-sorted algebra A with component data domains $A_1,\ldots,A_n$ can be *effectively presented* in the following sense: to each A_i there is associated a recursive set $\Omega_i \subset \omega$ and a surjection $\alpha_i: \Omega_i \to A_i$ such that for each operation $\sigma: A_{\lambda_1}\times\ldots\times A_{\lambda_k} \to A_\mu$ of there is a recursive tracking function σ_α which commutes the diagram

$$\begin{tikzcd}
A_{\lambda_1}\times\ldots\times A_{\lambda_k} \arrow[r, "\sigma"] & A_\mu \\
\Omega_{\lambda_1}\times\ldots\times\Omega_{\lambda_k} \arrow[r, "\sigma_\alpha"] \arrow[u, "\alpha_{\lambda_1}\times\ldots\times\alpha_{\lambda_k}"] & \Omega_\mu \arrow[u, "\alpha_\mu"']
\end{tikzcd}$$

A many-sorted algebra A is said to be *computable* (*semicomputable* or *cosemicomputable*) if it can be effectively presented, just as above, and, in addition, each relation $\equiv_{\alpha_i}$ defined on Ω_i by

$$x \equiv_{\alpha_i} y \quad \text{if, and only if,} \quad \alpha_i(x) = \alpha_i(y) \text{ in } A_i$$

is recursive (r.e. or co-r.e.).

Together with finiteness, these notions of effectivity are isomorphism invariants and make up four basic properties of algebra semantics. See our [1] and, in particular, RABIN[11] and MAL'CEV[10] for further information.

2.1. BASIC LEMMA. *Let* (Σ,E) *be a specification with* E *a recursively enumerable set of equations. Then* $T_I(\Sigma,E)$ *is semicomputable and* $T_F(\Sigma,E)$ *is cosemicomputable. In particular, if algebra* A *possesses an r.e. equational hidden enrichment specification with respect to* (1) *initial algebra semantics or* (2) *final algebra semantics then* (1) A *is semicomputable or* (2) A *is cosemicomputable. If* A *possesses such specifications with respect to both initial and final algebra semantics then* A *is computable.*

In a forthcoming paper we shall prove theorems which may be taken as strong converses to implications indexed (1) and (2) in Lemma 2.1. These will yield a neat characterisation of computable data type semantics.

This last fact is taken from the proof of Theorem 3.1 of our [1].

2.2. LEMMA. *Let* A *be a computable minimal algebra of signature* Σ_A. *Then there exists a computable minimal* B *of signature* $\Sigma_B \supset \Sigma_A$ *having a finite equational specification* (Σ_B,E_B) *with respect to initial semantics such that*

$$B|_{\Sigma_A} = \langle B\rangle_{\Sigma_A} \cong A.$$

Moreover, B *and* (Σ_B,E_B) *can be chosen with* (1) *each domain* B_i *of* B *equal to* ω *or to a finite initial segment of* ω, (2) $0 \in B_i$ *as a constant of sort* i *in* Σ_B *and with* (3) *a unary function symbol* ${}^iS$ *of sort* i *such that the family of terms* $\{{}^iS^n(0) : n \in B_i\}$, *indexed by the sorts* i *of* Σ_B, *is a traversal or set of normal forms for* $\equiv_{E_B}$.

3. FINAL ALGEBRA SEMANTICS FOR $\mathcal{PR}$

We algebraically structure the primitive recursive functions on the natural numbers into a 2-sorted algebra A with domains ω and PR named in the signature Σ of A by sorts N and M (for "number" and "map"). A is defined by using a 2-sorted operation to glue a single-sorted arithmetic to a single-sorted function algebra.

Let A_N be the single-sorted algebra on ω with constant $0 \in \omega$ and operations $x + 1$, $x \dot{-} 1$, $x + y$, $\lambda(x) = x - \lfloor\sqrt{x}\rfloor^2$. Let $\Sigma_N = \{0,S,P,+,\lambda\}$ be the signature of A_N.

Let A_M be a single-sorted algebra on PR with constants the operations of A_N plus the everywhere zero function and whose operators are

$$sum(f,g)(x) = f(x)+g(x) \qquad it(f)(x) = \begin{cases} 0 & \text{if } x = 0 \\ f^x(0) & \text{if } x \neq 0 \end{cases}$$
$$comp(f,g)(x) = f(g(x))$$

Let $\Sigma_M = \{\text{ZERO},\text{SUCC},\text{PRED},\text{ADD},\Lambda,\text{SUM},\text{COMP},\text{IT}\}$ be the signature of A_M.

Now define A by joining A_M and A_N with *eval*: $\text{PR} \times \omega \to \omega$ defined by

$$eval(f,x) = f(x).$$

Let $\Sigma = \Sigma_N \cup \Sigma_M \cup \{\text{EVAL}\}$.

3.1. LEMMA. A *is a finitely generated minimal algebra which is cosemicomputable but not computable.*

PROOF. That A is finitely generated and minimal follows from ROBINSON[12] where it is shown that every unary primitive recursive function is the result of a finite number of applications of *sum, comp, it* to $0, x + 1, \lambda(x)$. The rest of the result we leave as an exercise in recursive function theory. Q.E.D.

3.1. THEOREM. *The algebra of primitive recursive functions* A *has a finite equational hidden enrichment specification with respect to final algebra semantics but fails to possess an r.e. conditional hidden enrichment* *specification with respect to initial algebra semantics.*

PROOF. The second statement follows from Lemma 2.1 and Lemma 3.1. We prove the existence of a final algebra specification for A.

By Lemma 2.2 there is a computable algebra A_N^0, with a finite equational *initial* semantics specification (Σ_N^0, E_N^0), with domain ω such that $A_N^0|_{\Sigma_N} = \langle A_N^0 \rangle_{\Sigma_N} = A_N$ and so $T_I(\Sigma_N^0, E_N^0)|_{\Sigma_N} = \langle T_I(\Sigma_N^0, E_N^0) \rangle_{\Sigma_N} \cong A_N$. Define a new algebra A_0 by replacing A_N in A by A_N^0. Clearly, $A_0|_\Sigma = \langle A_0 \rangle_\Sigma = A$. We will give A_0 the required finite equational specification (Σ_0, E_0) with respect to final semantics.

E_0 is defined to be E_N^0, interpreted as equations over Σ_0, plus the following equations.

$$\mathrm{EVAL}(\mathrm{ZERO},Y) = 0 \qquad \mathrm{EVAL}(\mathrm{PRED},Y) = P(Y)$$
$$\mathrm{EVAL}(\mathrm{SUCC},Y) = S(Y)$$
$$\mathrm{EVAL}(\Lambda,Y) = \lambda(Y)$$

$$\mathrm{EVAL}(\mathrm{SUM}(X_1,X_2),Y) = \mathrm{ADD}(\mathrm{EVAL}(X_1,Y),\mathrm{EVAL}(X_2,Y))$$
$$\mathrm{EVAL}(\mathrm{COMP}(X_1,X_2),Y) = \mathrm{EVAL}(X_1,\mathrm{EVAL}(X_2,Y))$$
$$\mathrm{EVAL}(\mathrm{IT}(X),0) = 0$$
$$\mathrm{EVAL}(\mathrm{IT}(X),S(Y)) = \mathrm{EVAL}(\mathrm{COMP}(X,\mathrm{IT}(X)),Y)$$

wherein X, X_1, X_2 are function indeterminates and Y is a numerical indeterminate.

Let ϕ be the unique epimorphism $T(\Sigma_0) \to A_0$. Then $A_0 \cong T(\Sigma_0)/\equiv_\phi$ and so what we have to prove is that $\equiv_\phi$ is $\equiv_{\max(E_0)}$. Clearly, $\equiv_\phi$ is non-unit (because A_0 is non-trivial) and $\equiv_{E_0} = \equiv_{\min(E_0)}$ is contained in $\equiv_\phi$ (because A_0 is an E_0-algebra). What remains to be shown is that any non-unit congruence $\equiv$ extending $\equiv_{E_0}$ is contained within $\equiv_\phi$.

Let $\equiv$ be any non-unit congruence extending $\equiv_{E_0}$ composed of the two component relations $\equiv_N$ and $\equiv_M$. Let ϕ split into component functions ϕ_N and ϕ_M with $\equiv_\phi$ consisting of $\equiv_{\phi_N}$ and $\equiv_{\phi_M}$.

We consider maximality for the numerical terms first.

3.3. LEMMA. *Let* t *be a numerical term of* $T(\Sigma_0)$. *Then* $t \equiv_{E_0} S^{\phi_N(t)}(0)$.

Maximality follows easily: let t_1, t_2 be numerical terms in $T(\Sigma_0)$ and suppose $t_1 \equiv_N t_2$. Then by Lemma 3.3 we can write $t_i \equiv_{E_0} S^{\phi_N(t_i)}(0)$ for $i = 1,2$ and, since $\equiv$ extends $\equiv_{E_0}$, we have $S^{\phi_N(t_1)}(0) \equiv_N S^{\phi_N(t_2)}(0)$. Now if $\phi_N(t_1) \neq \phi_N(t_2)$ then, using the predecessor functions, all numerals can be identified under $\equiv_N$. This contradicts the non-triviality of $\equiv_N$ because the numerals $\{S^n(0) : n \in \omega\}$ were already a complete set of coset representatives for $\equiv_{E_0}$ (Lemma 3.3). Thus $\phi_N(t_1) = \phi_N(t_2)$ and $t_1 \equiv_{\phi_N} t_2$.

Before proving Lemma 3.3 we consider maximality for the function terms. Let t_1, t_2 be map terms of $T(\Sigma_0)$ and assume $t_1 \equiv_M t_2$. For any $n \in \omega$ we know that

$EVAL(t_1, S^n(0)) \equiv_N EVAL(t_2, S^n(0))$	because $\equiv$ is a congruence;
$\phi_N EVAL(t_1, S^n(0)) \equiv_N \phi_N EVAL(t_2, S^n(0))$	because we have shown $\equiv_N \subset \equiv_{\phi_N}$
$eval(\phi_M, t_1, n) \equiv eval(\phi_M t_2, n)$ in A_0	because ϕ is a homomorphism and $\phi_N(S^n(0)) = n$.

Therefore $t_1 \equiv_{\phi_M} t_2$.

PROOF OF LEMMA 3.3

This is done by induction on the complexity of numerical term t. Before describing the argument it is necessary to fix in the mind all the components of the algebras and specifications involved. These are best displayed in a diagram of Σ^0_N-algebras:

$$
\begin{tikzcd}
A|_{\Sigma^0_N} = A^0_N \arrow[rrr, leftrightarrow, "\cong"] & & & T_I(\Sigma^0_N, E^0_N) \arrow[dl, dashed] \\
& {[T(\Sigma_0)/\equiv_\phi]|_{\Sigma^0_N}} \arrow[ul, "\cong"'] & T_I(\Sigma_0, E_0)|_{\Sigma^0_N} \arrow[l, dashed] & \\
T(\Sigma_0)|_{\Sigma^0_N} \arrow[uu, dashed, "\phi_N"] \arrow[ur, "v"] & & & T(\Sigma^0_N) \arrow[lll, hook', "i"'] \arrow[ul, dashed] \arrow[uu, "v'"']
\end{tikzcd}
$$

In this diagram i is inclusion; v,v' denote projection maps; broken arrows are maps uniquely defined by initiality.

We consider the induction step only. This is divided into cases determined by the leading function symbol $\sigma \in \Sigma_N^0 \cup \{EVAL\}$ of t. Those cases when $\sigma \in \Sigma_N^0$ are routine because $E_N^0 \subset E_0$, but we do one as an example; let $t = ADD(t_1,t_2)$ where t_1,t_2 are assumed to be numerical terms in $T(\Sigma_0)$ of which Lemma 3.3 is true. Since $t_i \equiv_{E_0} S^{\phi_N(t_i)}(0)$ for i = 1,2 we can calculate as follows

$$t \equiv_{E_0} ADD(S^{\phi_N(t_1)}(0),S^{\phi_N(t_2)}(0)) \qquad \text{by substitution;}$$

$$t \equiv_{E_0} S^{\phi_N(t_1)+\phi_N(t_2)}(0) \qquad \text{because } E_N^0 \subset E_0\text{, or, more formally, by diagram chasing;}$$

$$t \equiv_{E_0} S^{\phi_N(ADD(t_1,t_2))}(0) \qquad \text{because } \phi_N \text{ is a } \Sigma_N^0 \text{ homomorphism } T(\Sigma_0)|_{\Sigma_N^0} \to A_N^0.$$

Let us turn to the interesting case of σ = EVAL. This follows directly from this next fact.

3.4. LEMMA. *For any* function *term* $t \in T(\Sigma_0)$ *and for any* $n \in \omega$

$$EVAL(t,S^n(0)) \equiv_{E_0} S^{\phi_N(EVAL(t,S^n(0))}(0)$$

PROOF. This is done by induction on the complexity of t. The basis cases are direct calculations. Consider the induction step in case $t = SUM(t_1,t_2)$ where t_1,t_2 are function terms in $T(\Sigma_0)$ for which it is assumed that Lemma 3.4 is true. We calculate

$$\begin{aligned} EVAL(t,S^n(0)) &\equiv_{E_0} ADD(EVAL(t_1,S^n(0)),EVAL(t_2,S^n(0))) \\ &\equiv_{E_0} ADD(S^{\phi_N(EVAL(t_1,S^n(0))}(0),S^{\phi_N(EVAL(t_2,S^n(0)))}(0)) \\ &\qquad \text{by induction;} \\ &\equiv_{E_0} S^{\phi_N(ADD(EVAL(t_1,S^n(0)),EVAL(t_2,S^n(0))))}(0) \\ &\qquad \text{as above;} \\ &\equiv_{E_0} S^{\phi_N(EVAL(t,S^n(0)))} \qquad \text{because } \equiv_{E_0} \subset \equiv_{\phi}. \end{aligned}$$

The case $t = COMP(t_1,t_2)$ follows the same pattern. The case $t = IT(t_0)$ requires a secondary induction on n for $EVAL(IT(t_0),S^n(0))$, but it is nevertheless straightforward. Q.E.D.

4. INITIAL ALGEBRA SEMANTICS FOR *PR*

By stripping down the algebra A an initial algebra specification of the primitive recursive functions can found. Let B be the 2-sorted algebra, with domains PR and ω, obtained by deleting all the operations of A which are internal to PR. Thus B consists of A_N joined to the *set* PR by *eval*: $PR \times \omega \to \omega$; put simply: if $\Sigma = \Sigma_N \cup \{EVAL\}$ then $B = A|_\Sigma$. Of course, B is *not* a finitely generated algebra, but

4.1. THEOREM. *With respect to initial algebra semantics,* B *possesses a finite equational specification* (Σ_0,E_0) *involving hidden functions such that*

$$T_I(\Sigma_0,E_0)|_\Sigma \cong B.$$

PROOF. We shall first show that B is a computable algebra.

4.2. LEMMA. *There is a computable enumeration of the primitive recursive functions* $\beta\colon \omega \to PR$ *which is bijective and possesses a recursive universal function* $U_\beta(e,x) = \beta(e)(x)$.

PROOF. Let $c\colon S \to PR$ be any standard enumeration of PR having recursive universal function U_c. The problem is to remove the repititions in c hence we define a recursive function $h\colon \omega \to \omega$ which will list from S one, and only one, code for each function. This done we can set $\beta = ch\colon \omega \to PR$ and take $U_\beta(e,x) = U_c(h(e),x)$ as a recursive universal function.
Here is an h, defined inductively, which will find the smallest c-code for each primitive recursive function. *Base*: $h(0) = 0$. *Induction Step*: suppose $h(0),\dots,h(n)$ have been computed. To compute $h(n+1)$ search out the smallest bound $b \in \omega$ for which there is a c-code $e < b$ such that
(1) $(\forall e'<e)(\exists x<b)[U_c(e',x) \neq U_c(e,x)]$ and (2) $e \notin \{h(0),\dots,h(n)\}$. Now seek

the smallest c-code $e_0 < b$ satisfying (1) and (2) and take $h(n+1) = e_0$. We leave the reader to check h satisfies the required conditions. Q.E.D.

Thus we may now fix up a computable numbering for B by using the identity map $i: \omega \to \omega$ and $\beta: \omega \to PR$ of Lemma 4.2. The operations of B are all recursive with respect to this pair (i,β) as are the induced equality relations (because both maps are bijections). Theorem 4.1 now follows from this next lemma used in connection with Lemma 2.2.

4.3. LEMMA. *Let* A *be any many-sorted computable algebra of signature* Σ *one of whose domains is* ω. *Then there exists a finitely generated minimal computable algebra* A_{min} *of signature* $\Sigma_{min} \supset \Sigma$ *such that* $A_{min}|_\Sigma \cong A$.

PROOF. To make A_{min} from A first add $0 \in \omega$ as a constant and successor $x+1$ as a unary operation to A. Next choose any computable numbering β of A each of whose component mappings β_i have domain ω and add each $\beta_i: \omega \to A_i$ as a new operation. This is A_{min} and it is clearly minimal and computable (even without the informal hypothesis that β is effective for β as an operation of A_{min} is officially tracked by the identity in the original β coding of A!). Forgetting all these new operations, we see $A_{min}|_\Sigma = A$. Q.E.D.

REFERENCES

[1] BERGSTRA, J.A. & J.V. TUCKER, *Algebraic specifications of computable and semicomputable data structures*, Mathematical Centre, Department of Computer Science Research Report IW 115, Amsterdam, 1979.

[2] ________, *A characterisation of computable data types by means of a finite, equational specification method*, Mathematical Centre, Department of Computer Science Research Report IW 124, Amsterdam, 1979. (To appear in ICALP'80 Springer Verlag, Berlin 1980.)

[3] ________, *Equational specifications for computable data types: six hidden functions suffice and other sufficiency bounds*, Mathematical Centre, Department of Computer Science Research Report IW 128, Amsterdam, 1980.

[4] BERGSTRA, J.A. & J.V. TUCKER, *On bounds for the specification of finite data types by means of equations and conditional equations*, Mathematical Centre, Department of Computer Science Research Report IW 131, Amsterdam, 1980.

[5] _______, *On the adequacy of finite equational methods for data type specification*, ACM-SIGPLAN Notices 14 (11) (1979) 13-18.

[6] BROY, M., W. DOSCH, H. PARTSCH, P. PEPPER, M. WIRSING, *Existential quantifiers in abstract data types*, in H. MAURER (ed.) *Automata, languages and programming, 6th Colloquium, Graz, July* 1979 Springer-Verlag, Berlin, 1979, 72-87.

[7] GOGUEN, J.A., J.W. THATCHER & E.G. WAGNER, *An initial algebra approach to the specification, correctness and implementation of abstract data types*, in R.T. YEH (ed.) *Current trends in programming methodology* IV, *Data structuring*, Prentice-Hall, Engelwood Cliffs, New Jersey, 1978, 80-149.

[8] GUTTAG, J.V. & J.J. HORNING, *The algebraic specification of abstract data types*, Acta Informatica 10 (1978) 27-52.

[9] KAMIN, S., *Final data type specifications: a new data type specification method*, 7th POPL Conference, Las Vegas, ACM, 1980, 131-138.

[10] MAL'CEV, A.I., *Constructive algebras*, I., *Russian Mathematical Surveys*, 16, (1961) 77-129.

[11] RABIN, M.O., *Computable algebra, general theory and the theory of computable fields*, Transactions American Mathematical Society, 95 (1960) 341-360.

[12] ROBINSON, R.M., *Primitive recursive functions*, Bull. American Math. Soc., 53 (1947) 925-942.

[13] WAND, M., *Final algebra semantics and data type extensions*, J. Computer Systems Sciences 19 (1979), 27-44.