stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 115/79 AUGUSTUS
(DEPARTMENT OF COMPUTER SCIENCE)

J.A. BERGSTRA & J.V. TUCKER

ALGEBRAIC SPECIFICATIONS OF COMPUTABLE AND SEMICOMPUTABLE DATA STRUCTURES

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 Reviews-category: 4.34.

Algebraic specifications of computable and semicomputable data structures[*]

by

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

ABSTRACT

We address theoretical issues to do with algebraic specification methods for data structures, particularly the so called *hidden function* and *hidden sort* specifications. On giving exact definitions of the computable and semicomputable data structures we show that every computable data structure possesses a (special kind of) finite, equational hidden functions specification and that every semicomputable data structure possesses a (special kind of) finite, equational hidden sorts specification. In passing we answer 5 questions recently posed by S. Kamin and are able to virtually complete the classification of the comparative power of his 27 categories of algebraic specification methods.

KEY WORDS & PHRASES: *abstract data structures, algebraic specification, hidden functions, hidden sorts, computable and semicomputable algebras*

[*] This paper is not for review; it is meant for publication elsewhere.

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

INTRODUCTION

The purpose of this paper is to make some theoretical contributions to the algebraic theory of data type specification associated with the names J.V. Guttag, J.J. Horning, S. Zilles, B. Liskov, M. Majster, the ADJ Group. Underlying this theory is the idea that each data type τ in a programming system L or particular program P should be characterised explicitly in L or P in an algebraic fashion by defining it as a collection of operators Σ with explicitly defined properties E so that the control and assignment structure of the type are precisely specified and salient features of its implementation made visible. The algebraic theory of data type specification studies algebraic prescriptions (Σ,E) for data types. The rapid growth of such studies has not been unproblematical largely because of technical problems of an algebraic nature (one thinks of the literature generated by Majster's Transversal Stack [8] which fails to have the much favoured finite, equational specification, see KAPUR [7]). And this has led to a profusion of algebraic specification methods some ad hoc, designed for particular examples.

Recently, S. KAMIN [6] introduced a classification scheme, embracing many of the methods so far applied, and summarized what was known concerning their comparative power and adequacy, asking questions of the form: *Given two methods* M,M' *is* M *more generally applicable than* M'? and, *Does a given method* M *define all the data types one wants*? Here we continue the classification of these methods by proving two very general adequacy theorems and answering several questions recorded as open in Kamin's survey.

We do this by taking a point of view closer to that of Majster in her work on (in-)adequacy [9] than that of Kamin. Each type τ when specified algebraically by a pair (Σ,E) gives rise to a class K_τ of all algebras satisfying the properties of E and this class K_τ of all *data structures* of type τ completely determines the semantical structure of τ. Therefore to discuss the adequacy of the known specification methods we give, in section two, an exact definition of the computable and semicomputable data structures belonging to any type. We prove, in section three, that *every finitely generated computable data structure* A *has a particular kind of finite, equational hidden function specification, called a finite, equational hidden enrichment*; and, in section four, we prove that *every finitely generated semicomputable* *algebra has a particular kind of finite, equational hidden sorts specification called a finite, equational hidden enrichment by sorts.* Section one contains the algebra we need, and the other comparative/adequacy results appear mainly in sections two and five.

2. ALGEBRAIC PRELIMINARIES

The reader is assumed familiar with Kamin's admirable survey [6] and with the initial algebra methodology of the ADJ GROUP [1] on which it rests; an acquaintance with Majster's paper [9] is also useful. Here we shall collect a number of algebraic facts, but we begin by fixing terminology and notation: in so doing we shall note any correspondences between our technical vocabulary and the seven commonly occurring useful, distinct meanings of the word *data type* in the literature of Programming Methodology listed by D. Gries in [4, pp.263-268].

Each *data structure* A can be thought to consist of a finite family $A_1,\dots,A_n$ of *data domains*, or *component data structures,* together with a finite family of operations and relations of the forms

$$\sigma^{\lambda,\mu} = \sigma^{\lambda_1,\dots,\lambda_k,\mu} : A_{\lambda_1} \times\dots\times A_{\lambda_k} \to A_\mu$$

$$R^{\lambda} = R^{\lambda_1,\dots,\lambda_k} \subset A_{\lambda_1} \times\dots\times A_{\lambda_k}$$

for some $k \in \omega$, the natural numbers, and $\mu,\lambda_i \in \{1,\dots,n\}$, $1 \le i \le k$. Of course, the relations of A become redundant under the assumption that among the data domains is the Boolean $B = \{0,1\}$. Defining a data structure to be a heterogenous algebra in this way represents the third use of the word *type* in Gries's list and subsumes the second; we use the terms data structure and algebra synonymously.

The signature Σ_A of a heterogenous algebra A consists of a name, called a *sort*, for each of its domains and a standardised notation for each of its operations which names the sorts on which they are defined, such *data signatures* formalise the first use of *type* in Gries's list. By a *data type class* or a *data type semantics* we shall here mean any class K of data structures of common data signature, the fourth useage of type in Gries's list. Thus a basic task of algebraic data type specification is to give formal algebraic, syntactical definitions (Σ,E) of data type classes K and of *particular* data structures A either absolutely or relative to some class K; the sixth and seventh useages of *type* in Gries's list arise in such specification mechanisms. (The fifth, and last, useage in the list arises in specifying constructions of new data type classes and structures from old ones.) Henceforth, we concern ourselves with the problems of data structure specification.

We assume concepts such as *subalgebras; congruences; factor algebras; homo-, mono-, epi-,* and *iso-, morphisms* are known along with that of *initial algebras* for classes of algebras and this machinery which is the basis of Kamin's classification scheme (see [6,1,2]):

A data structure A is to be specified by a data signature Σ and a set of equations E over Σ where these equations are of three kinds. Let T_Σ be the Σ *term algebra* and $T_\Sigma[X_1,\dots,X_n]$ be the Σ *polynomial algebra* in the variables $X_1,\dots,X_n$. An identity $t = t'$ is called a *simple equation* over Σ if $t,t' \in T_\Sigma$ and is called a *(polynomial) equation* over Σ if $t,t' \in T_\Sigma[X_1,\dots,X_n]$. A *conditional equation* over Σ is a formula of the form $t_1 = t_1' \wedge\dots\wedge t_k = t_k' \rightarrow t = t'$ where $t,t',t_i,t_i' \in T_\Sigma[X_1,\dots,X_n]$ for $1 \le i \le k$. The sets of all such equations are nested

$$SEQ(\Sigma) \hookrightarrow EQ(\Sigma) \hookrightarrow CEQ(\Sigma).$$

The initiality of T_Σ for the class of $ALG(\Sigma)$ of all Σ algebras is used to construct from a set of such equations E in a specification (Σ,E) an initial object $T_{\Sigma,E}$ as a factor algebra of T_Σ for the class of all Σ algebras satisfying the properties of E: the class of all so called E-*algebras* with signature Σ. This is done as follows:

Any set of equations E defines a set of pairs $E_s \subset T_\Sigma \times T_\Sigma$ which satisfy the formulae of E in the usual way. Thus if $E \subset SEQ(\Sigma)$ then $E = E_s$; if $E \subset EQ(\Sigma)$ then $E_s = \{(t(s_1,\dots,s_n), t'(s_1,\dots,s_n)) : t = t' \in E, s_1,\dots,s_n \in T_\Sigma\}$ where if $t \in T_\Sigma[X_1,\dots,X_n]$ then $t(s_1,\dots,s_n)$ is the result of substituting s_i for X_i in t, $1 \le i \le n$; and so on. Then $T_{\Sigma,E}$ is by definition $T_\Sigma/\equiv_{E_S}$.

A pair (Σ,E) is a *specification* for the data structure A if $T_{\Sigma,E} \cong A$.

The complexity of a specification (Σ,E) will be measured in terms of the kind of equations included in E and whether E is finite, recursive or recursively enumerable. In preparation for describing Kamin's classification more precisely, let us remark on the constructivity of our various sets of algebraic syntax. We assume T_Σ, $T_\Sigma \times T_\Sigma$, $T_\Sigma[X_1,\ldots,X_n]$, $SEQ(\Sigma)$, $EQ(\Sigma)$, $CEQ(\Sigma)$, have a standard gödel numbering δ so that given any gödel number of a term, polynomial, equation and so on, we can recursively calculate gödel numbers for all its component subterms. In saying, for example $E \subset CEQ(\Sigma)$ is a recursive or recursively enumerable set we will formally mean the set $\delta^{-1}(E)$ is recursive or r.e. and in saying $E = \{e_i : i \in \omega\}$ is recursively enumerated by $f(i) = e_i$ we will formally mean f is a recursive function $\omega \to \delta^{-1}(E)$ such that $\delta f: \omega \to E$ is surjective.

A set of equations E over a signature may be a finite (F), recursive (REC) or recursively enumerable (RE) set of simple (S), polynomial in several variables (V), or conditional (C) equations. Let $\Gamma \in \{F, REC, RE\} \times \{S,V,C\}$. Then a specification (Σ,E) is said to be of type Γ if E is a set of equations of type (abbreviated by) Γ.

A data structure A possesses a Γ *specification (without hidden functions or hidden sorts)* if there exists a specification (Σ,E) of type Γ such that $T_{\Sigma,E} \cong A$. These specifications are abbreviated (Γ,N) specifications: N for *no*, in anticipation of the more elaborate specifications which are the main subject of this paper. It is, perhaps, useful to place this simple fact here as an illustration of (Γ,N) specification.

If A is an algebra and $a_1,\ldots,a_n \in A$ then $(A_1,a_1,\ldots,a_n)$ is the algebra with domains and operations those of A but with $a_1,\ldots,a_n$ adjoined to the constants of A.

1.1. PROPOSITION. *Let* A *be finite and generated by* $a_1,\ldots,a_n$. *Then* $(A,a_1,\ldots,a_n)$ *has a* (F,S,N) *specification.*

PROOF. Let A_i, the component data domain of A named by sort i, consist of elements $b_1^i,\ldots,b_{m_i}^i$. For each i, choose m_i polynomials such that $t_j^i(a_1,\ldots,a_n) = b_j^i$, $1 \le j \le m_i$. Now for each operation $\sigma^{\lambda,\mu}$ of A write out the graph of $\sigma^{\lambda,\mu}$,

$$\mathrm{graph}(\sigma^{\lambda,\mu}) = \{(b_{j_1}^{\lambda_1},\ldots,b_{j_k}^{\lambda_k},b_j^\mu) : \sigma^{\lambda,\mu}(b_{j_1}^{\lambda_1},\ldots,b_{j_k}^{\lambda_k}) = b_j^\mu\},$$

in terms of these polynomials; thus

$$\sigma^{\lambda,\mu}(t_{j_1}^{\lambda_1}(a),\ldots,t_{j_k}^{\lambda_k}(a)) = t_j^{\mu}(a),$$

where $a = (a_1,\ldots,a_n)$, and collect these identities as simple equations over Σ, the signature of $(A,a_1,\ldots,a_n)$, into the finite set E. Then $(A,a_1,\ldots,a_n) \cong T_{\Sigma,E}$. Q.E.D.

If σ or a is an operation or constant of algebra A then we invariably denote the corresponding notations in Σ_A by $\underline{\sigma}$ or $\underline{a}$ respectively. A useful algebra is $(\omega;0,+1)$ where $+1(n) = n+1$, its signature we write $\Sigma_{0,S} = \{0,S\}$.

The *prime subalgebra* P_A of an algebra A is the intersection of all subalgebras of A: the *smallest* subalgebra of A. Equivalently, P_A is the subalgebra of A generated by the constants named in Σ_A. A is said to be *prime* if $A = P_A$ or, equivalently, if A has no proper subalgebras.

Hidden function and hidden sort specifications of algebras A are based upon signature contractions of algebras B where $\Sigma_B \supset \Sigma_A$. Two such contractions are important: let A be an algebra and $\Sigma_A \supset \Sigma$ then,

$A|_\Sigma$ denotes the algebra with domains those of A named by the sorts of Σ and operations only those of A named in Σ;

$\langle A\rangle_\Sigma$ denotes the prime subalgebra of $A|_\Sigma$, often termed the Σ-prime subalgebra of A.

The following facts are easy to see.

1.2. LEMMA. $A|_\Sigma \cong \langle A\rangle_\Sigma$ *implies* $A|_\Sigma = \langle A\rangle_\Sigma$

1.3. LEMMA. *Let* A *have signature* $\Sigma_A \supset \Sigma_1 \supset \Sigma_0$. *Then*

$$(A|_{\Sigma_1})|_{\Sigma_0} = A|_{\Sigma_0} \quad \textit{and} \quad \langle\langle A\rangle_{\Sigma_1}\rangle_{\Sigma_0} = \langle A\rangle_{\Sigma_0}.$$

1.4. LEMMA. *Let* A,B *be algebras of common signatures* $\Sigma \supset \Sigma_0$. *Any morphism* ϕ: A → B *is a morphism* $\phi: A|_{\Sigma_0} \to B|_{\Sigma_0}$ *and* $\langle A\rangle_{\Sigma_0} \to \langle B\rangle_{\Sigma_0}$.

KAMIN [6, p.37] uses these contractions to distinguish two kinds of hidden function specifications and two kinds of hidden sorts specification for data structures: let $\Gamma \in \{F,REC,RE\} \times \{S,V,C\}$.

A data structure A has a Γ *hidden function specification* (1) under the *usual interpretation* or (2) under the *subalgebra interpretation* if there is a $\Sigma \supset \Sigma_A$, containing exactly the sorts of Σ_A, and a set of equations of type Γ over Γ_Σ such that (1) $T_{\Sigma,E}|_{\Sigma_A} \cong A$ or (2) $<T_{\Sigma,E}>\Sigma_A \cong A$ respectively.

Similarly for Γ hidden sort specifications. Kamin's notation (Γ,HF) and (Γ,HS) specifications refer to hidden function and hidden sort specifications *using the usual interpretation*.

Notice that if the algebra A is prime then a hidden function specification (Σ,E) under the usual interpretation is also one under the subalgebra interpretation,

$$T_{\Sigma,E}|_{\Sigma_A} = <T_{\Sigma,E}>_{\Sigma A} \cong A.$$

Pairs (Σ,E) for which this occurs we define, in section three, to be *hidden enrichment* specifications, and they are the only hidden function specifications we use. Thus all data structures *when specified* will *appear* prime; this is not accidental.

Let A be a data structure. This is a semantical concept, and it seems reasonable to suppose it is finitely generated in a computation in which it appears by initial values $a_1,\ldots,a_n$ which are either fixed by its data type or are presented to the type as input. In either case the signature of any specification (Σ,E) for A should carry names $x_1,\ldots,x_n$ for otherwise programming over the specification would not allow one to access all of A (note that 'running' a toy program scheme over an algebra A computes strictly within the subalgebra of A generated by its input). This means that if A is any structure finitely generated by $a_1,\ldots,a_n$ which one feels acceptable as a data structure in a computation then one should ask if it is specifiable in the form $(A,a_1,\ldots,a_n)$. All this is implicit in the (Γ,N) specifications as defined: it is easy to see that the prime algebras of $ALG(\Sigma)$ are precisely the factor algebras of its initial algebra T_Σ.

We close this section with two lemmas we use later on.

1.5. <u>LEMMA</u>. *Let* (Σ_0,E_0) *and* (Σ,E) *be specifications with* $\Sigma_0 \subset \Sigma$ *and* $E_0 \subset E$. *If there exists a transversal* $J \subset T_{\Sigma_0}$ *such that*

(i) *for distinct* t_1, $t_2 \in J$, $t_1 \not\equiv_E t_2$;

(ii) *for each constant* $c \in \Sigma - \Sigma_0$, *there is a* $t \in J$ *such that* $c \equiv_E t$;

(iii) *for each* k-*ary* $\sigma \in \Sigma - \Sigma_0$, *and any* $t_1,\dots,t_k \in J$, *there is a* $t \in J$ *such that* $\sigma(t_1,\dots,t_k) \equiv_E t$; *then* $T_{\Sigma,E}|_{\Sigma_0} \cong T_{\Sigma_0,E_0}$.

PROOF. Since $E_0 \subset E$ and J is a transversal it is easy to see that $\phi([t]_{E_0}) = [t]_E$, for $t \in J$, well defines a Σ_0 homomorphism $T_{\Sigma_0,E_0} \to T_{\Sigma,E}|_{\Sigma_0}$. Condition (i) implies ϕ is injective because if $t_1,t_2 \in J$ and $[t_1]_{E_0} \neq [t_2]_{E_0}$ then $[t_1]_E \neq [t_2]_E$. Conditions (ii) and (iii) imply ϕ is surjective as follows: we show that for each $t \in T_\Sigma$ there is a $t_0 \in J$ such that $t \equiv_E t_0$. Now if $t \in T_{\Sigma_0}$ then $t \equiv_{E_0} t_0$, for some $t_0 \in J$, and so $t \equiv_E t_0$ as $\equiv_{E_0} \subset \equiv_E$. Assume $t \in T_\Sigma - T_{\Sigma_0}$. We argue by induction on the complexity of t.

The basis sees t as a constant in $\Sigma - \Sigma_0$ and is immediate from condition (ii).

Let $t = \sigma(s_1,\dots,s_k)$ for some $\sigma \in \Sigma$ and assume there exist $t_1,\dots,t_k \in J$ such that $s_i \equiv_E t_i$, $1 \le i \le k$. Then $t \equiv_E \sigma(t_1,\dots,t_k)$. If $\sigma \in \Sigma_0$ then $\sigma(t_1,\dots,t_k) \in T_{\Sigma_0}$ and obviously $t \equiv_E t_0$ for some $t_0 \in J$. If $\sigma \in \Sigma - \Sigma_0$ then $\sigma(t_1,\dots,t_k) \equiv_E t_0$ for some $t_0 \in J$ by condition (iii) and so $t \equiv_E t_0$. Q.E.D.

1.6 LEMMA. *Let* (Σ_0,E_0) *and* (Σ,E) *be specifications with* $\Sigma_0 \subset \Sigma$, $E_0 \subset E$ *and*

$$T_{\Sigma,E}|_{\Sigma_0} \cong T_{\Sigma_0,E_0}.$$

Let A *and* B *be* Σ_0 *and* Σ *algebras such that*

$$B|_{\Sigma_0} \cong A.$$

If $A \cong T_{\Sigma_0,E_0}$ *and* B *is an* E-*algebra then* $B \cong T_{\Sigma,E}$.

PROOF. The hypotheses imply $T_{\Sigma,E}|_{\Sigma_0} \cong B|_{\Sigma_0}$ by, say, Σ_0-isomorphism ϕ. Moreover, the initiality of $T_{\Sigma,0}|_{\Sigma_0}$ for E_0-algebras, inherited from T_{Σ_0,E_0}, implies that ϕ is the only Σ_0 homomorphism $T_{\Sigma,E}|_{\Sigma_0} \to B|_{\Sigma_0}$. Since B is an E-algebra there exists a Σ homomorphism ψ: $T_{\Sigma,E} \to B$ which restricts to a Σ_0 homomorphism $T_{\Sigma,E}|_{\Sigma_0} \to B|_{\Sigma_0}$. Thus $\psi = \phi$ and must be bijective; in particular, ψ must be a Σ isomorphism. Q.E.D.

2. COMPUTABLE AND SEMICOMPUTABLE ALGEBRAS

Our semantic measure of adequacy is invested in the concepts of *computable* and *semicomputable data structures* which are defined in a moment. These definitions are based upon work of M.O. RABIN [12] and, in particular, A.I. MAL'CEV [10] devoted to inventing a theory of computable algebraic systems and they represent a distinct improvement on other formulations, such as Majster's definition of a computable data type in [9], because they are completely formal and give concepts which are isomorphism invariants: the hall-mark of genuine algebraic properties. For background material we recommend MAL'CEV's [10].

A data structure A is said to be *effectively presented* if corresponding to its family of component data domains $A_1,\dots,A_n$ there are mutually disjoint recursive sets $\Omega_1,\dots,\Omega_n$, $\Omega_i \subset \omega$, $1 \le i \le n$, and surjections $\alpha_i\colon \Omega_i \to A_i$, $1 \le i \le n$ such that for each operation $\sigma = \sigma^{(\lambda_1,\dots,\lambda_k,\mu)}\colon A_{\lambda_1} \times\dots\times A_{\lambda_k} \to A_\mu$ of A there is a recursive tracking function $\sigma_\alpha = \sigma_\alpha^{(\lambda_1,\dots,\lambda_k,\mu)}\colon \Omega_{\lambda_1} \times\dots\times \Omega_{\lambda_k} \to \Omega_\mu$ which commutes the diagram:

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

wherein $\alpha_{\lambda_1} \times\dots\times \alpha_{\lambda_k}(x_{\lambda_1},\dots,x_{\lambda_k}) = (\alpha_{\lambda_1}(x_{\lambda_1}),\dots,\alpha_{\lambda_k}(x_{\lambda_k}))$.

A is a *computable data structure* if for each $1 \le i \le n$ the relation $\equiv_{\alpha_i}$, defined on Ω_i by $x \equiv_{\alpha_i} y$ iff $\alpha_i(x) = \alpha_i(y)$ in A_i, is recursive. And A is a *semicomputable data structure* if each of these $\equiv_{\alpha_i}$ is recursively enumerable.

Combining the $\Omega_1,\dots,\Omega_n$ and the $\alpha_1,\dots,\alpha_n$ we can obtain a recursive number of algebra Ω of signature Σ and a Σ epimorphism $\alpha\colon \Omega \to A$. Thus A is effectively presented when it is the homomorphic image of a recursive number algebra. Combining the $\equiv_{\alpha_i}$, $1 \le i \le n$, into $\equiv_\alpha$ identifies the computability or semicomputability of A with the recursiveness or recursive enumerability of $\equiv_\alpha$. Pairs (Ω,α) we refer to as effective, recursive (or computable), and semirecursive (or semicomputable) *coordinatisations*, accordingly.

Here are some facts which are easily proved.

1. Every countable data structure A possesses an effective coordinatisation.

2. If A is computable, or semicomputable, and B is isomorphic to A, then B is computable or semicomputable.

3. If A is a finitely generated data structure computable or semicomputable under both α: $\Omega_\alpha \to A$ and β: $\Omega_\beta \to A$ then α and β are *recursively equivalent* in the sense that there exist recursive functions f,g which commute the diagram:

$$\begin{tikzcd}
& A & \\
\Omega_\alpha \arrow[ur, "\alpha"] \arrow[rr, shift left] & & \Omega_\beta \arrow[ul, "\beta"'] \arrow[ll, shift left, "g"]
\end{tikzcd}$$

See MAL'CEV [10].

2.1. LEMMA. *Every computable data structure* A *is isomorphic to a recursive, number algebra* Ω *each of whose numerical data domains* Ω_i *is the set of natural numbers,* ω*, or the set of the first* m *natural numbers,* ω_m*, according to whether or not the corresponding data domain* A_i *is infinite or finite of cardinality* m.

PROOF. Since A is computable it possesses a computable coordinatisation (Ω_α,α) consisting of recursive sets Ω_i^α, surjections α_i: $\Omega_i^\alpha \to A_i$ with recursive congruences $\equiv_{\alpha_i}$, and recursive tracking operations, $1 \le i \le n$.

For each $1 \le i \le n$, define the recursive sets $\Gamma_i \subset \Omega_i^\alpha$ by

$$x \in \Gamma_i \iff x \in \Omega_i^\alpha \;\&\; (\forall z < x)[z \in \Omega_i^\alpha \to z \equiv_{\alpha_i} x]$$

so that α_i: $\Gamma_i \to A_i$ is bijective; let f_i be a recursive bijection $\omega \to \Gamma_i$, if Γ_i is infinite, or a finite bijection $\omega_{m_i} \to \Gamma_i$, if Γ_i is finite with m_i elements, and denote the domain of each f_i by Ω_i. Thus for $1 \le i \le n$, $\beta_i = \alpha_i f_i$ is a bijection $\Omega_i \to A_i$.

Now for each recursive tracking function $\sigma_\alpha^{\lambda,\mu}$: $\Omega_{\lambda_1}^\alpha \times \ldots \Omega_{\lambda_k}^\alpha \to \Omega_\mu^\alpha$

define the recursive function $\sigma_\beta^{\lambda,\mu}: \Omega_{\lambda_1} \times \ldots \times \Omega_{\lambda_k} \to \Omega_\mu$ by

$$\sigma_\beta^{\lambda,\mu}(x_1,\ldots,x_k) = f_\mu^{-1} g_\mu \sigma_\alpha^{\lambda,\mu}(f_{\lambda_1}(x_1),\ldots,f_{\lambda_k}(x_k)).$$

It is easy to check that $\sigma_\beta^{\lambda,\mu}$ tracks $\sigma^{\lambda,\mu}$ on $\Omega_{\lambda_1},\ldots,\Omega_{\lambda_k}$, Ω_μ whence it follows that combining the $\Omega_1,\ldots,\Omega_n$ and these $\sigma_\beta^{\lambda,\mu}$ makes a recursive numerical algebra Ω isomorphic to A under β as required in the lemma. Q.E.D.

Obviously, no isomorphic or *faithful* representation, such as that provided by Ω in Lemma 2.1, is available to the semicomputable data structures, else they would be computable; actually each semicomputable algebra A can be represented as the image of such a Ω under epimorphism β with $\equiv_\beta$ r.e. and we shall now show this in the finitely generated case.

The gödel numbering of a T_Σ, discussed in section one, is, of course, a canonical kind of recursive coordinatisation of T_Σ which makes it a computable Σ algebra. By Lemma 1.1 we can take the domain of this gödel numbering to be a Σ algebra with component domains ω or ω_m, for various m, and so speak of a canonical isomorphic representation (Ω_*,γ_*).

If A is finitely generated by $a_1,\ldots,a_n$ then by initially, $(A,a_1,\ldots,a_n)$ with signature Σ is the image of a unique epimorphism v: $T_\Sigma \to (A,a_1,\ldots,a_n)$ and so we have an effective coordinatisation $\gamma = v\gamma_*:\Omega_* \to T_\Sigma \to (A,a_1,\ldots,a_n)$. (This proves remark 1 in the finitely generated case). Notice that

$$\varepsilon = \{\gamma_*(i) = \gamma_*(j): (i,j) \in \equiv_\gamma\}$$

is the set of all simple equations over Σ true in A, $T_{\Sigma,\varepsilon} \cong (A,a_1,\ldots,a_n)$.

If A is semicomputable then it follows from remark 3 that $\equiv_\alpha$ is r.e. and we have in γ a representation for semicomputable algebras analogous to that in 1.1. But this observation also means that ε is r.e. and so we have shown one half of

2.2. PROPOSITION. *Let* A *be an algebra finitely generated by* $a_1,\ldots,a_n$. *Then* $(A,a_1,\ldots,a_n)$ *is semicomputable if, and only if, it possesses an* (RE,S,N) *specification.*

The converse is easy to formally verify. Now, in the argument above, if A is computable then we get $\equiv$ is recursive:

2.3. PROPOSITION. *Let* A *be a computable algebra finitely generated by* $a_1,\ldots,a_n$. *Then* $(A,a_1,\ldots,a_n)$ *has a* (REC,S,N) specification.

The converse of 2.3 is false as we shall see in section five.

Adequacy, for Kamin, is measured syntactically by means of (RE,S,N) specifications. The situation as far as specifications *not* involving hidden functions, or hidden sorts, he reports to be this.

The (RE,S,N), (RE,V,N) and (RE,C,N) all specify the same classes of data structures; later, in 5.1, we add the (REC,C,N) specifications to this list and then show, in 5.5, that (REC,V,N) specifications define strictly fewer data structures. (Note also 2.3 in this connection).

The (F,V,N) specifications define strictly fewer structures than (F,C,N) specifications, moreover, Kamin announces that the ADJ Group's discovery of a computable data structure with a (F,C,N) specification but no (F,V,N) specification; a computable algebra with no (F,C,N) specification can be obtained from the proof of 3.3. Kamin asked [6, p.34] if the much favoured (F,V,N) specifications admit algebras which are not computable, this we can answer straightaway:

2.4. THEOREM. *There exists a group* G *finitely generated by* $g_1,\ldots,g_n$ *which is semicomputable, but not computable, and contains (an isomorphic copy of) every semicomputable group, such that* $(G,g_1,\ldots,g_n)$ *has a* (F,V,N) *specification.*

PROOF. Let $\Sigma_0 = \{\cdot, {}^{-1}, 1\}$ and $\Sigma = \Sigma_0 \cup \{x_1,\ldots,x_n\}$. Let E_0 be a finite set of equations over Σ_0 which define group structures so that T_{Σ,E_0} is the free group on n generators, an initial algebra specification of the class GROUP(n) of all n generator groups. Let G be a group finitely generated by $g_1,\ldots,g_n$ and finitely presented by $(x_1,\ldots,x_n;r_1,\ldots,r_m)$; let $E = E_0 \cup \{r_1 = 1,\ldots,r_m=1\}$. Then $(G,g_1,\ldots,g_n)$ has the (F,V,N) specification (Σ,E). By Higman's Theorem [3] we can choose such a G not only to have insoluble word problem, but even to contain a copy of every finitely generated semicomputable group. Q.E.D.

This argument yields a useful reference point for the mathematical literature.

2.5. PROPOSITION. *Let* V *be a variety of algebras defined by a finite set of laws. If* $A \in V$ *is finitely generated by* $a_1, \ldots, a_n$ *and is finitely presented with respect to* V *then* $(A, a_1, \ldots, a_n)$ *has a* (F,V,N) *specification.*

So, for example, every finitely generated commutative ring is finitely presented with respect to the variety of commutative rings (by the Hilbert Basis Theorem) and so has a (F,V,N) specification: a fact not without interest if one wishes to apply these algebraic methods to data type specification in algebraic manipulation programs. Actually, such rings are all computable and, of course, finitely generated abelian groups are finitely presentable and computable. But it is worth noting that commutativity does not always guarantee computability: *there exists a finitely definable variety of commutative loops whose free loop on one generator is not computable,* see MAL'CEV [11].

The situation as far as specifications involving hidden functions and sorts Kamin reports is far less complete. Beyond the obvious equivalences of (RE,S,N) with (RE,S,HF), (RE,V,HF), (RE,C,HF), (RE,S,HS), (RE,V,HS), (RE,C,HS), and Majster's well known example showing that (F,V,N) and (F,V,HF) are distinct nothing much else is known. Here we contribute (REC,S,HE), and so (REC,V,HE), (REC,C,HE), as equivalents of (RE,S,N) in 5.2 as well as (F,V,HES), a special kind of (F,V,HS) specification in 4.1. The next section shows that (F,V,HE) are adequate for all computable algebras.

The main outstanding question is *Does every finitely generated semicomputable data structure possess a* (F,V,HE) *specification?*

3. HIDDEN ENRICHMENT SPECIFICATIONS OF COMPUTABLE ALGEBRAS

A data structure A has a Γ *hidden enrichment specification* if there is a $\Sigma \supset \Sigma_A$, containing exactly the sorts of Σ_A, and a set of equations E of type Γ over T_Σ such that

$$T_{\Sigma,E}|_{\Sigma_A} = \langle T_{\Sigma,E} \rangle_{\Sigma_A} \cong A.$$

This we abbreviate as a (Γ,HE) specification. Of special interest are the (F,V,HE) specifications so observe that an algebra A has a (F,V,HE) specification if there exists an algebra B, with $\Sigma_B \supset \Sigma_A$ and containing the same sorts, such that B has a (F,V,N) specification and $B|_{\Sigma_A} = \langle B\rangle_{\Sigma_A} \cong A$. In this paper all constructions involving hidden functions will be hidden enrichments; this section and the next use only (F,V,HE) specifications while section five considers (REC,S,HE) specifications.

This theorem shows that (F,V,HE) specifications are adequate for all data structures arising in Computer Science.

3.1. <u>THEOREM</u>. *Let* A *be a computable algebra finitely generated by* $a_1,\ldots,a_n$. *Then* $(A,a_1,\ldots,a_n)$ *has a* (F,V,HE) *specification.*

<u>PROOF</u>. We shall write down a detailed proof for the case that A is single sorted. From this the reader should find no difficulties, beyond those of notational complications, in preparing an equally precise proof for the case that A has more than one sort (we comment further on this at the end of the argument). The case when A is single sorted and finite is taken care of in Proposition 1.1 so assume A is infinite.

By Lemma 2.1, $(A,a_1,\ldots,a_n)$ is isomorphic to a recursive number algebra of the form $R = (\omega;f_1,\ldots,f_m,c_1,\ldots,c_n)$ where the f_i are recursive functions tracking the corresponding operations of A and the c_i are numbers corresponding to the a_i. Notice that R is prime since $(A,a_1,\ldots,a_n)$ is prime. We shall show R has a (F,V,HE) specification by constructing an appropriate algebra Γ, possessing a (F,V,N) specification, such that $\Gamma|_{\Sigma_R} = \langle\Gamma\rangle_{\Sigma_R} \cong R$.

We will need this technical lemma:

3.2. <u>LEMMA</u>. *Let* $f_1,\ldots,f_m$ *be primitive recursive functions and* $\lambda_1,\ldots,\lambda_\ell$ *the functions appearing in their explicit definitions. Then*

$$A = (\omega;0,+1,\lambda_1,\ldots,\lambda_\ell,f_1,\ldots,f_m)$$

has a (F,V,N) *specification.*

PROOF. Without loss of generality, we can assume the operations of A are ordered in list $0,+1,\theta_1,\ldots,\theta_{\ell+m}$ so that any function is to the right of all those functions appearing in its explicit definition. Define the sequence of algebras $A_0 = (\omega;0,+1)$ and $A_{n+1} = (A_n,\theta_{n+1})$ for $n = 0,\ldots,\ell+m-1$. We prove inductively that each A_n has a (F,V,N) specification so that, in particular, $A_{\ell+m} = A$ has.

At the base of the sequence this is obvious: let $\Sigma_0 = \{0,S\}$ then $A_0 \cong T_{\Sigma_0}$.

Assume A_n has a (F,V,N) specification (Σ_n,E_n) so that $A_n \cong T_{\Sigma_n,E_n}$ and consider A_{n+1}. Now the new function θ_{n+1} is either a projection function, or is defined by composition or primitive recursion over other θ_i where $i < n+1$. These three cases are treated in like manner so we shall write out only the case of primitive recursion.

Here

$$\theta_{n+1}(0,x_1,\ldots,x_k) = \theta_i(x_1,\ldots,x_k)$$

$$\theta_{n+1}(y+1,x_1,\ldots,x_k) = \theta_j(y,x_1,\ldots,x_k,\theta_{n+1}(y,x_1,\ldots,x_k)).$$

So set $\Sigma_{n+1} = \Sigma_n \cup \{\bar{\theta}_{n+1}\}$ and E_{n+1} to be E_n with these equations adjoined

$$\bar{\theta}_{n+1}(0,X_1,\ldots,X_k) = \bar{\theta}_i(X_1,\ldots,X_k)$$

$$\bar{\theta}_{n+1}(SY,X_1,\ldots,X_k) = \bar{\theta}_j(Y,X_1,\ldots,X_k,\bar{\theta}_{n+1}(Y,X_1,\ldots,X_k)).$$

Clearly (Σ_{n+1},E_{n+1}) is a (F,V,N) specification so we must show $T_{\Sigma_{n+1},E_{n+1}} \cong A_{n+1}$. We shall use Lemma 1.5. We know $A_{n+1}|_{\Sigma_n} = A_n$ and $A_n \cong T_{\Sigma_n,E_n}$ so we must verify that $T_{\Sigma_{n+1},E_{n+1}}|_{\Sigma_n} \cong T_{\Sigma_n,E_n}$ and for this we shall use Lemma 1.6.

Consider $J = \{S^r(0) : r \in \omega\}$. Now J is a transversal for T_{Σ_n,E_n} because $T_{\Sigma_n,E_n}|_{\Sigma_0} \cong A_n|_{\Sigma_0} \cong A_0 \cong T_{\Sigma_0}$. Condition (i) of Lemma 1.6 is fullfilled by E_{n+1} because A_{n+1} is an E_{n+1} algebra, condition (ii) is automatic and so we are left with condition (iii). This condition is checked by considering $\bar{\theta}_{n+1}(S^r(0),S^{r_1}(0),\ldots,S^{r_k}(0))$ and proving by induction on r that it is E_{n+1} equivalent to an element of J going by the equations for $\bar{\theta}_{n+1}$ to elements of T_{Σ_n} in which J is an $E_n \subset E_{n+1}$ transversal. Q.E.D.

We shall now construct Γ from $R = (\omega; f_1, \ldots, f_m, c_1, \ldots, c_n)$. Let $f\colon \omega^k \to \omega$ be a recursive function. Then the graph of f

$$\mathrm{graph}(f) = \{(x_1, \ldots, x_k, f(x_1, \ldots, x_k)) : x_1, \ldots, x_k \in \omega\}$$

is recursively enumerable, ROGERS [13]. Since every r.e. set has a primitive recursive enumeration, ROGERS [13], let $h_1, \ldots, h_k$, $g\colon \omega \to \omega$ be primitive recursive functions enumerating graph(f). Thus,

$$\mathrm{graph}(f) = \{(h_1(z), \ldots, h_k(z), g(z)) : z \in \omega\}$$

and, in particular, for all $z \in \omega$,

$$f(h_1(z), \ldots, h_k(z)) = g(z).$$

For each k_j-ary recursive operation f_j of R choose primitive recursive functions $h_1^j, \ldots, h_{k_j}^j, g^j$ which enumerate graph(f_j) and let $\overrightarrow{\lambda_{ij}}$ and $\overrightarrow{\mu_j}$ be the lists of functions making up the explicit definitions of the h_i^j and g^j respectively. Define

$$\Gamma = (\omega; 0, +1, \overrightarrow{\lambda_{ij}}, \overrightarrow{\mu_j}, h_1^j, \ldots, h_{k_j}^j, g^j, f_j, c_1, \ldots, c_n)_{1 \le j \le m, 1 \le i \le k_j}.$$

Clearly, $\Gamma|_{\Sigma_R} = \langle\Gamma\rangle_{\Sigma_R} \cong R$ because R is prime. We have to show Γ has a (F,V,N) specification.

First set $\Gamma_0 = (\omega; 0, +1, \overrightarrow{\lambda_{ij}}, \overrightarrow{\mu_j}, h_1^j, \ldots, h_{k_j}^j, g^j)_{1 \le j \le m, 1 \le i \le k_j}$ and let its signature be Σ_0. Then $\Gamma|_{\Sigma_0} = \langle\Gamma\rangle_{\Sigma_0} \cong \Gamma_0$ and, by Lemma 3.2, Γ_0 has a (F,V,N) specification (Σ_0, E_0). We now define a specification for Γ: let Γ have signature Σ, so $\Sigma = \Sigma_0 \cup \Sigma_R$, and let E be E_0 with these equations added:

for each constant $\bar{c}_j \in \Sigma_R$, $\bar{c}_j = S^{c_j}(0)$;

for each operation $\bar{f}_j \in \Sigma_R$, $\bar{f}_j(\bar{h}_1^j(X), \ldots, \bar{h}_k^j(X)) = \bar{g}^j(X)$.

The pair (Σ, E) is a (F,V,N) specification so we verify $T_{\Sigma,E} \cong \Gamma$. This is done by Lemma 1.5.

Clearly Γ is an E-algebra so all that remains is the hypothesis $T_{\Sigma,E}|_{\Sigma_0} \cong$ $\cong T_{\Sigma_0,E_0}$; for this we look to Lemma 1.6.

Consider $J = \{S^r(0) : r \in \omega\}$. That J is a transversal for T_{Σ_0,E_0} follows from the fact that $T_{\Sigma_0,E_0}|_{0,S} \cong \Gamma_0|_{0,S} \cong T_{0,S}$. Conditions (i) and (ii) of Lemma 1.6 are true of J by inspection of E which leaves condition (iii). So consider the term $\bar{f}(S^{r_1}(0),\dots,S^{r_k}(0))$. The isomorphism between T_{Σ_0,E_0} and Γ_0 implies there is an $S^z(0)$ such that $S^{r_i}(0) \equiv_{E_0} \bar{h}_i S^z(0)$ for $1 \leq i \leq k$. Thus

$$\begin{aligned} \bar{f}(S^{r_1}(0),\dots,S^{r_k}(0)) &\equiv_E \bar{f}(\bar{h}_1 S^z(0),\dots,\bar{h}_k S^z(0)) \\ &\equiv_E \bar{g}(S^z(0)). \end{aligned}$$

Since $\bar{g}S^z(0) \in T_{\Sigma_0}$, and J is an E_0 transversal, $\bar{g}S^z(0) \equiv_{E_0} S^i(0)$ for some i whence the condition follows as $\equiv_{E_0} \subset \equiv_E$.

In the many sorted case, Lemma 2.1 provides an isomorphic many sorted recursive number algebra R whose infinite domains are all ω and whose finite domains are ω_m for various m. To reconstruct the proof one needs to introduce a sort index to the notation of the proof as one shows R has a (F,V,HE) specification and to use it to keep track of the distinction between those domains which are finite and those which are infinite, no new technical ideas are required beyond those of the above argument. Q.E.D.

The importance of hidden operations became apparent when MAJSTER [8] pointed out that a particular stack-like data structure failed to admit a (F,V,N) specification and JONES [5] and the ADJ Group [1] showed it could be given a (F,V,HF) specification. Here is a very simple example which separates the two methods.

Let $A = (\omega; 0, +1, f)$ where $f(n) = n^2$ for $n \in \omega$.

3.3. PROPOSITION. A *has a* (F,V,HE) *specification but possesses no* (F,V,N) *specification.*

PROOF. A (F,V,HE) specification of A follows from Theorem 3.1 and is obvious anyway. Suppose that (Σ,E) is a (F,V,N) specification of A. We assume that E contains no trivial equations of the form $t_1 = t_2$ where t_1 and t_2 are identical polynomials, and write $E = E_1 \cup E_2 \cup E_3$ where

E_1 contains the simple equations,

E_2 contains the equations of any one of the following three forms:

$t_1(x) = t_2$, $t_2 = t_3(x)$, $t_1(x) = t_3(x)$ with t_2 simple and x occurring free in $t_1(x)$ and $t_3(x)$, and

E_3 contains the equations of the form $t_1(x) = t_2(x)$.

First of all E_2 turns out to be empty; for instance $t_1(x) = t_2$ can never hold because all functions in A are injective, and consequently $t_1(x)$ is (interpreted by) an injective function that cannot have valve t_2 for all arguments. By substituting $\underline{0}$ for y in an equation $t_1(x) = t_3(y)$ we obtain an equation of the form $t_1(x) = t_2$.

Now we show that E_3 cannot be empty. To see this we assume the converse, i.e. $T_{\Sigma,E_1} \cong A$.

Let $E_k = \{\underline{f}(\underline{S}^n(\underline{0})) = \underline{S}^{n^2}(\underline{0}) : n \in \omega,\ n \leq k\}$. For a sufficiently large k, say k_0, E_k implies all equations in E_1. As T_{Σ,E_1} is an E_k-algebra, for each k, we have that T_{Σ,E_1} is an E_{k_0}-algebra and $T_{\Sigma,E_{k_0}}$ is an E_1-algebra, hence both are isomorphic, so $T_{\Sigma,E_{k_0}} \cong A$. This can be contradicted by giving an example of an E_{k_0}-algebra in which A cannot be homomorphically embedded. The example is this:

$$B = (\omega,+1,g) \text{ where } g(x) = \begin{cases} x^2 & \text{if } x \leq k_0 \\ k_0^2 & \text{otherwise.} \end{cases}$$

Thus we know that E_3 is not empty. We will derive a final contradiction from this. Let $t_1(x) = t_2(x)$ be an equation in E_3. Then $t_1(f(x)) = t_2(f(x))$ is a valid equation in A. Let Σ_A' be Σ_A minus $\underline{0}$; let B be the following structure:

$$B = (\omega_j f_0, f_2, f_2, \ldots) \text{ where } f_i(X) = X^2 + 1 \text{ for } i \in \omega.$$

Define the following map H: $T_{\Sigma_A'}[f(x)] \to T_{\Sigma_B}[x]$.

$$H(f(x)) = \underline{f}_0(x)$$

$$H(\underline{S}(t(x))) = \underline{f}_{a_0+1}(f_{a_1}(....f_{a_k}(x)..)) \text{ if } H(t(x)) = \underline{f}_{a_0}(...f_{\underline{a}_k}(x)..)$$

$$H(\underline{f}(t)) = \underline{f}_0(H(t)).$$

We observe that H is injective (use induction on complexity of terms in $T_{\Sigma'_A}[\underline{f}(x)]$) and that t and H(t) have the same interpretations as functions on ω (again with induction). Let

$$H(t_1(\underline{f}(x))) = f_{a_1}(....f_{a_p}(x)..);\ H(t_2(f(x))) = f_{b_1}(...(f_{b_a}(x))..).$$

We consider the semigroup G of functions on ω generated by $f_0, f_1, ...$ under composition. In G the following equation holds $f_{a_1}, ..., f_{a_p} = f_{b_1}, ..., f_{b_q}$ with $a_1 ... a_p$ and $b_1 ... b_q$ different sequences of indices. A contradiction finally follows from the observation that G is free.

To prove this assume $f_{a_1}, ..., f_{a_p} = f_{b_1}, ..., f_{b_q}$. If $p \neq q$ then, as polynomials on ω, both sides have different degrees (2^p and 2^q) and consequently cannot represent identical functions. So we may assume p = q.

We need some notation:

$\sigma^i = f_{a_i}, ..., f_{a_p}$; $\tau^i = f_{b_i}, ..., f_{b_p}$; $\delta^i = \sigma^i + \tau^i$; $\rho^i = \sigma^i - \tau^i$. Note that $\deg(\sigma^i) = \deg(\delta^i) = \deg(\tau^i) = 2^{p-i+1}$, for $i \leq p$.

Now suppose σ' and τ' are not equal terms, take j to be maximal such that $a_j \neq b_j$.

By induction on k one shows for $k \in \{0, ..., j-1\}$ that $\rho^{j-k} \neq 0$.

Basis, k = 0: there are two cases: j = n and j < n.

j = n implies $\rho^j = \sigma^j - \tau^j = (x^2 + a_n) - (x^2 + b_n) = a_n - b_n \neq 0$

j < n implies
$$\begin{aligned}\rho^j = \sigma^j - \tau^j &= f_{a_j}\, \sigma^{j+1} - f_{b_j}\, \tau^{j+1} \\ &= (\sigma^{j+1})^2 + a_j - ((\tau^{j+1})^2 + b_j) \\ &= (\sigma^{j+1})^2 - (\tau^{j+1})^2 + (a_j - b_j) \\ &\neq 0\end{aligned}$$

because $\sigma^{j+1} = \tau^{j+1}$ for i > j entails $a_i = b_i$.

Induction step let $\rho^\ell \neq 0$. Then $\rho^{\ell-1} = f_{a_{\ell-1}}\, \sigma^\ell - f_{b_{\ell-1}}\, \tau^\ell$

$$\begin{aligned} &= (\sigma^{\ell})^2 + a_{\ell-1} - (\tau^2)^2 - b_{\ell-1} \\ &= \rho^{\ell}\,\delta^{\ell} + (a_{\ell-1} - b_{\ell-1}) \\ &\neq 0 \end{aligned}$$

because $\rho^{\ell} \neq 0$ implies $\deg(\rho^{\ell}\,\delta^{\ell}) \geq \deg(\delta^{\ell}) \geq 2^{\rho-\ell+1} \geq 2$. Hence $\deg(\rho^{\ell-1}) \geq 2$. This concludes the argument. Q.E.D.

It is worth noting this proof can be adapted to show that A has no (F,C,N) specification.

4. HIDDEN ENRICHMENT BY SORTS SPECIFICATIONS OF SEMICOMPUTABLE ALGEBRAS

A data structure A has a Γ *hidden enrichment by sorts specification* if there exists a $\Sigma \supset \Sigma_A$ and a set of equations E of type Γ over T_Σ such that

$$T_{\Sigma,E}|_{\Sigma_A} = \langle T_{\Sigma,E}\rangle_{\Sigma_A} \cong A.$$

This we abbreviate as a (Γ,HES) specification. We work only with (F,V,HES) specifications so observe that an algebra A has a (F,V,HES) specification if there exists an algebra B with a (F,V,HE) specification such that $B|_{\Sigma_A} = \langle B\rangle_{\Sigma_A} \cong A$. Hidden sorts seems first to have been used by P.A. SUBRAHMANYAM [14] in order to sepcify MAJSTER's Traversable Stack [8].

4.1. THEOREM. *Let* A *be a semicomputable algebra finitely generated by* $a_1,\ldots,a_n$. *Then* $(A,a_1,\ldots,a_n)$ *has a* (F,V,HES) *specification.*

PROOF. Our remarks commencing the proof of Theorem 3.1 are once more applicable here so we proceed to write down the argument in case A is single sorted and infinite.

Let Σ be the signature of $(A,a_1,\ldots,a_n)$. Since A is semicomputable we can choose a recursive number algebra $R = (\omega;\sigma_\gamma^1,\ldots,\sigma_\gamma^\ell,c_1,\ldots,c_n)$ and an epimorphism $\gamma\colon R \to (A,a_1,\ldots,a_n)$ such that $\gamma(c_i) = a_i$ and $\equiv_\gamma$ is r.e., and where γ is factored by a canonical Σ isomorphism $\gamma_*\colon R \to T_\Sigma$,

$$\begin{tikzcd} T_\Sigma \arrow[r, "v"] & (A,a_1,\ldots,a_n) \\ R \arrow[u, "\gamma_*"] \arrow[ur, "\gamma"'] & \end{tikzcd}$$

so that $\varepsilon = \{\gamma_*(i) = \gamma_*(j)\colon (i,j) \in \equiv_\gamma\}$ is the set of all identities in T_Σ true in A, see section two.

By ROGERS [13], we can choose primitive recursive functions f,g to enumerate $\equiv_\gamma$ so that $\equiv_\gamma = \{(f(z),g(z))\colon z \in \omega\}$ and $\varepsilon = \{\gamma_* f(z) = \gamma_* g(z)\colon z \in \omega\}$. Adjoin these functions to R to make (R,f,g) whose signature we refer to as $\Sigma(f,g)$.

Consider this structure made by adjoining (R,f,g) to $(A,a_1,\ldots,a_n)$ as a new sort using γ:

$$B = (A,\omega;\sigma_1,\ldots,\sigma_\ell,a_1,\ldots,a_n,\sigma_\gamma^1,\ldots,\sigma_\gamma^\ell,c_1,\ldots,c_n,f,g,\gamma).$$

Clearly $B|_\Sigma = (A,a_1,\ldots,a_n)$ because ω is not a sort of Σ. We shall prove the theorem by showing B has a (F,V,HE) specification.

Since (R,f,g) is computable we can apply the argument of Theorem 3.1 to obtain a (F,V,N) specification (Σ_0,E_0) of a new recursive number algebra R_0 such that $T_{\Sigma_0,E_0} \cong R_0$, $R_0|_{\Sigma(f,g)} \cong (R,f,g)$ and $R_0|_{0,S} \cong (\omega;0,+1)$. In particular,

$$T_{\Sigma_0,E_0}|_{\Sigma(f,g)} = \langle T_{\Sigma_0 E_0}\rangle_{\Sigma(f,g)} \cong (R,f,g) \text{ and } T_{\Sigma_0,E_0}|_{0,S} \cong T_{0,S}.$$

Define B_1 to be B with all the new operations of R_0 adjoined: $B_1|_{\Sigma B} = B$. If $\Sigma_1 = \Sigma_{B_1}$ then $\Sigma_1 \supset \Sigma$, $\Sigma_1 \supset \Sigma_0 \supset \Sigma(f,g)$ and $\Sigma_1 \supset \Sigma_{0,S}$. We show B_1 has a (F,V,N) specification (Σ_1,E_1).

Define E_1 to be E_0 together with the equations over Σ_1,

$$\bar{\gamma}(\bar{c}_i) = \bar{a}_i$$

$$\bar{\gamma}(\bar{\sigma}_\gamma(X_1,\ldots,X_k)) = \bar{\sigma}(\bar{\gamma}(X_1),\ldots,\bar{\gamma}(X_k))$$

$$\bar{\gamma}\bar{f}(X) = \bar{\gamma}\bar{g}(X);$$

(Σ_1,E_1) is a (F,V,N) specification. To show $T_{\Sigma_1,E_1} \cong B_1$ we proceed in two steps. First we claim T_{Σ_1,E_1} is an ε-algebra so $T_{\Sigma_1,E_1} \cong T_{\Sigma_1,E_1\cup\varepsilon}$. Secondly we claim $B_1 \cong T_{\Sigma_1,E_1\cup\varepsilon}$. Consider this second claim first. B_1 is an $E_1 \cup \varepsilon$ algebra so, by initiality and fact that B_1 is prime, there is a unique epimorphism $\phi\colon T_{\Sigma_1,E_1\cup\varepsilon} \to B_1$. So check injectivity for ϕ: split ϕ into $\phi_1 = \phi(T_{\Sigma_1,E_1\cup\varepsilon})|_\Sigma$ and $\phi_2 = \phi(T_{\Sigma_1,E_1\cup\varepsilon})|_{\Sigma_0}$.

Now $T_{\Sigma_1,E_1\cup\varepsilon}|_\Sigma$ is an ε-algebra and $\phi_1\colon T_{\Sigma_1,E_1\cup\varepsilon}|_\Sigma \to B_1|_\Sigma = A \cong T_{\Sigma,\varepsilon}$. Hence $T_{\Sigma_1,E_1\cup\varepsilon}|_\Sigma$ is initial for ε-algebras and $T_{\Sigma_1,E\cup\varepsilon}|_\Sigma \cong B_1|_\Sigma$ by ϕ. The case of ϕ_2 follows the same lines.

So consider the first claim. Observe that $\{S^n(0)\colon n \in \omega\}$ is a transversal for T_{Σ_0,E_0} so that $\bar{f}S^n(0) \equiv_{E_0} S^{f(n)}$ and $\bar{g}S^n(0) \equiv_{E_0} S^{g(n)}$ since $T_{\Sigma_0,E_0}|_{\Sigma(f,g)} \cong (R,f,g)$. Moreover one may now use the equations given for E_1 to show $\gamma(S^n(0)) \equiv_{E_1} \gamma_*(n)$ by induction on the complexity of terms. From these observations: $\gamma_*(f(z)) \equiv_{E_1} \bar{\gamma}(S^{f(z)}(0)) \equiv_{E_1} \bar{\gamma}(\bar{f}S^n(0)) \equiv_{E_1} \bar{\gamma}(\bar{g}S^n(0)) \equiv_{E_1} \bar{\gamma}(S^{g(n)}(0)) \equiv_{E_1} \gamma_*(g(z))$ whence T_{Σ_1,E_1} is an ε-algebra. Q.E.D.

5. MISCELLANY

Here we prove three propositions which will answer Kamin's second and third questions.

5.1. <u>PROPOSITION</u>. *Let* A *be a semicomputable algebra finitely generated by* $a_1,\ldots,a_n$. *Then* $(A,a_1,\ldots,a_n)$ *has a* (REC,C,N) *specification.*

<u>PROOF</u>. By Proposition 2.2, $(A,a_1,\ldots,a_n)$ has an (RE,S,N) specification (Σ,E) where $E = \{e_i\colon i \in \omega\} \subset \mathrm{SEQ}(\Sigma)$ is r.e. enumerated by $f(i) = e_i$. Let x be a constant symbol of Σ and define E_x to be the set of all conditionals of the form

$$\underbrace{x = x \wedge \ldots \wedge x = x}_{i \text{ times}} \to e_i$$

for $i \in \omega$. Clearly $T_{\Sigma,E_x} \cong T_{\Sigma,E} \cong A$. But E_x is a recursive subset of $\mathrm{CEQ}(\Sigma)$ for, given any conditional $C \equiv c \to e$, to decide $C \in E_x$ one first decides if c is an iteration of $x = x$: if it is not then $C \notin E_x$; if it is then knowing it is, say, i conjunctions of $x = x$ one computes $f(i) = e_i$ and tests whether or not $e_i \doteq e$. Q.E.D.

5.2. <u>PROPOSITION</u>. *Let* A *be a semicomputable algebra finitely generated by* $a_1,\ldots,a_n$. *Then* $(A,a_1,\ldots,a_n)$ *has a* (REC,S,HE) *specification.*

PROOF. First, using Proposition 2.2, take E to be an r.e. set of simple equations such that $T_{\Sigma,E} \cong (A,a_1,\ldots,a_n)$; write $E = \bigcup_s E_s$ where E_s is the subset of E pertaining to sort s of Σ and choose recursive functions f_s, g_s to enumerate E_s so that $E_s = \{(f_s(i),g_s(i)) : i \in \omega\}$.

For each sort s adjoin to Σ a new function symbol I_s, to form a new signature Σ', and define E'_s to be the set of simple equations

$$\begin{array}{ll} I_s(t) = t & \text{for } t \in T_{\Sigma}, \\ I_s^i(f_s(i)) = g_s(i) & \text{for } i \in \omega \end{array}$$

$E' = \bigcup_s E'_s$ is a recursive set of simple equations over Σ', by reasoning analogous to that in Proposition 5.1, and $T_{\Sigma',E'}|_{\Sigma} \cong T_{\Sigma,E}$. Q.E.D.

Thus we have from 5.2 a counter-example to the converse of Proposition 2.3: if $T_{\Sigma',E'}$ were always computable then $T_{\Sigma,E}$ would be always computable.

Kamin's second and third questions asked if (REC,V,N) specifications defined fewer data structures than the (REC,V,HF) and (REC,C,N) specifications: these are answered affirmatively by combining this last fact with 5.2 and 5.1 respectively.

5.3. PROPOSITION. *There are semicomputable algebras* A *with no* (REC,V,N) *specifications.*

PROOF. Let $\Sigma = \{0,S,f,g\}$ where f,g are unary function symbols. Let $W \subset \omega$ and define $E_W \subset T_\Sigma \times T_\Sigma$ by

$$E_W = \{gS^n(0) = fS^n0 : n \in W\}.$$

Observe, now, that for any $t,t' \in T_\Sigma$, $t \equiv_{E_W} t'$ iff either $t = t'$ in T_Σ or $t = \sigma(fS^n(0))$ and $t' = \sigma(gS^n(0))$, or vice versa, for some polynomial $\sigma(X)$ over Σ and $n \in W$; in particular, notice that each equivalence class of $\equiv_{E_W}$ has at most two elements.

5.4. LEMMA. *If* T_{Σ,E_W} *has a* (REC,V,N) *specification then* W *is recursive.*

Now 5.3 follows from 5.4 on choosing W to be an r.e., non-recursive set and setting $A = T_{\Sigma,E_W}$ (because $n \in W$ iff $fS^n(0) \equiv_{E_W} gS^n(0)$). Here is the proof of the lemma:

Assume $T_{\Sigma,E_W} \cong T_{\Sigma,E}$ where E is a recursive set of polynomial equations over Σ. Then, from initiality, $\equiv_{E_W}$ is $\equiv_E$ and $T_{\Sigma,E_W} = T_{\Sigma,E}$. Partition E into its simple equations E_1 and its non-simple equations E_2 and consider $t_1(X) = t_2(Y) \in E_2$ (notice that equations may contain at most two variables since Σ contains only unary function symbols). If X,Y are different variables then putting $Y = 0$ the set $\{t_1(s): s \in T_\Sigma\} \subset [t_2(0)]_E$ which contradicts the finiteness of E equivalence classes: so no such equations may belong to E. If $X = Y$ then $t_1(X) = t_2(x)$ again fails to be valid in T_{Σ,E_W} unless t_1 and t_2 are identical polynomials and so the equation is trivial; to see this choose $n \in W$ and, substituting,

$$t_1(fS^n(0)) \equiv_E t_2(fS^n(0)) \equiv_E t_1(gS^n(0)) \equiv_E t_2(gS^n(0))$$

obtain an equivalence class of four elements (if W is empty then it is recursive, of course).

Thus E can be taken to consist of simple axioms only and we can deduce W is recursive from

$$n \in W \iff fS^n(0) = gS^n(0) \in E.$$

If $fS^n(0) = gS^n(0) \in E$ then $fS^n(0) \equiv_{E_W} gS^n(0)$ and $n \in W$. So assume $fS^n(0) = gS^n(0) \notin E$. The set $\{\sigma(t) = \sigma(t'): t = t' \in E, \sigma$ any polynomial over $\Sigma\}$ defines an equivalence relation $\equiv$ on T_Σ extending $\equiv_E$ (in fact it is $\equiv_E$) but $fS^n(0) = gS^n(0) \notin \equiv$ so $fS^n(0) = gS^n(0) \notin \equiv_E = \equiv_{E_W}$ and so $n \notin W$. Q.E.D.

REFERENCES

[1] 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.

[2] GOGUEN, J.A., E.G. WAGNER & J.B. WRIGHT, *Specification of abstract data types using conditional axioms,* IBM Research Report, RC 6214, Yorktown Heights, 1979.

[3] HIGMAN, G., *Subgroups of finitely presented groups,* Proceedings Royal Society, London, (A) 262 455-475.

[4] GRIES, D., (ed.), *Programming methodology,* Springer Verlag, New York, 1978.

[5] JONES, D.W., *A note on some limits of the algebraic specification method,* SIGPLAN Notices 13 (4) (1978) 64-67.

[6] KAMIN, S., *Some definitions for algebraic data type specifications,* SIGPLAN Notices 14 (3) (1979) 28-37.

[7] KAPUR, D., *Specifications of Majster's Traversable Stack and Veloso's Traversable Stack,* SIGPLAN Notices 14 (5) (1979) 46-53.

[8] MAJSTER, M.E., *Limits of the "algebraic" specification of abstract data types,* SIGPLAN Notices 12 (10) (1977) 37-42.

[9] ____________ , *Data types, abstract data types and their specification problem,* Theoretical Computer Science 8 (1979) 89-127.

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

[11] ____________ , *Identical relations on varieties of quasigroups,* American Mathematical Society Translations 82 (1969) 225-235.

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

[13] ROGERS, H., *Theory of recursive functions and effective computability,* McGraw-Hill, New York, 1967.

[14] SUBRAHMANYAM, P.A., *On a finite axiomatisation of the data type* L, SIGPLAN Notices 13 (4) (1978) 80-84.