stichting
mathematisch
centrum

2e boerhaavestraat 49 amsterdam

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 128/80 JANUARI

J.A. BERGSTRA & J.V. TUCKER

EQUATIONAL SPECIFICATIONS FOR COMPUTABLE DATA TYPES: SIX HIDDEN FUNCTIONS SUFFICE AND OTHER SUFFICIENCY BOUNDS

Preprint

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

Equational specifications for computable data types: six hidden functions suffice and other sufficiency bounds*

by

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

ABSTRACT

The ADJ Group's algebraic theory of data types identifies, semantically, each data type with a many-sorted algebra. In this technical paper, we prove that if A is a computable, infinite but finitely generated, many-sorted algebra with n sorts then A possesses a finite equational specification which involves at most n hidden constants and at most 3n+3 hidden functions. Thus in case A is single-sorted we have the bound of 6 mentioned in our title. Simple bounds on the number of equations used in the specifications are also included.

KEY WORDS & PHRASES: *algebraic data types, equational specifications with hidden functions, computable many-sorted algebras*

---

* This report is not for review: it will be submitted for publication elsewhere.

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

INTRODUCTION

In the ADJ Group's algebraic theory of data types the intended semantics of a data type is faithfully represented by a many-sorted algebra and to syntactically specify a data type by a particular method M is to use that technique to define some desired many-sorted algebra uniquely up to isomorphism. The question of adequacy for a specification method M is the informal question *Does the method* M *define all the data types one wants*? In principle, any algebra serves to model some aspect of data type semantics, but in theoretically testing the adequacy of method M it is more reasonable to ask M to specify only those algebras which are effectively computable in some precise sense. On establishing a rigorous definition of such computable semantics we are able to prove this rather striking adequacy theorem about equational techniques for data type specification:

THEOREM. *Let* A *be an infinite many-sorted algebra finitely generated by elements named in its signature* Σ. *If* A *is computable then* A *possesses a finite equational hidden functions specification* (Σ_0, E_0), *which is a hidden enrichment, and such that the number of hidden functions and the number of equations depend only on the signature* Σ *of* A *and not on any other properties of* A.

In precise terms, if Σ *names* n *sorts,* n_F *of which name finite domains* p *constants and* q *operations then* $\Sigma_0 - \Sigma$ *contains* n *constants and* 3n+3 *function symbols, and* E_0 *contains* $17+p+q+4(n-1)+n_F$ *equations.*

COROLLARY. *Let* A *be a single-sorted infinite computable algebra finitely generated by* p *constants and having* q *operations. Then* A *possesses a finite equational hidden enrichment specification with* 1 *hidden constant,* 6 *hidden operations and* 17+p+q *equations.*

To make clear what formal assumptions about data types are required to interpret the theorem stated, in section one we describe in more detail how the semantics and syntactic definitions of data types are depicted in the ADJ Group's theory. Section two defines the notion of a computable many-sorted algebra and states a deep result we use, but do not prove here: Matijacevič's *Diophantine Theorem*.

In section three we prove our theorem in the single-sorted case as this makes it easy for us to explain, and the reader to understand, the proof for the many-sorted case in section four.

This paper is the third in our series of mathematical studies of the comparative power and adequacy of algebraic specification methods for data types [1,2]; see also [3]. It is assumed the reader is cognisant with the work of the ADJ Group, at least with their paper [4] and has some experience of algebraic arguments. Knowledge of our previous work, although desirable, is not assumed.

We wish to thank H.C.J. Kleijn for her suggestions for improving the manuscript.

1. DATA TYPE SEMANTICS AND SPECIFICATION

The algebraic theory of data types acts on the assumption that a data type τ should be characterised in any programming system L or particular program P in which it occurs by defining it as a collection of operators Σ, with explicitly defined properties E, on different kinds of data obtained from a finite number of initial values. One intention, from the point of view of Programming Methodology, is that assignment and control structures intrinsic to the type become immediately visible as input/output format and extrinsic features of implementation fall away.

A *semantic realisation* of the type τ specified by (Σ,E) one imagines to be any many-sorted algebra A of signature Σ, satisfying the properties of E, and being finitely generated by elements named as constants in Σ: for the want of a better term, a *data structure* of type τ as specified by (Σ,E). Automatically, the *complete semantics* of the type τ defined by (Σ,E) is the class $ALG^*(\Sigma,E)$ of all such data structures. (We carry the * in our notation to emphasise that we are exclusively concerned with algebras not only satisfying the conditions in E but which are *finitely generated by constants named in* Σ.) In the context of the ADJ Group's initial algebra semantics, the situation is further structured by first taking the sharpest notion of semantic identity to be the algebraic isomorphism of two data structures and, secondly, by identifying the *intended* semantics of a type

(Σ,E) with an initial algebra I_K for $K = ALG^*(\Sigma,E)$, necessarily unique up to isomorphism *whenever it exists*. So to syntactically specify a type by (Σ,E) is to specify only an initial algebra I_K, those algebras $A \in K$ not isomorphic to I_K being considered as *non-standard* or, possibly, *deviant* semantical realisations of the type.

If $A \in ALG^*(\Sigma,E)$ then A is uniquely definable as an epimorphic image of I_K. And this I_K is in turn uniquely definable as an epimorphic image of $T(\Sigma)$, the algebra of all terms over Σ, since $T(\Sigma)$ is initial for the class of all Σ-algebras. Thus $I_K \cong T(\Sigma)/\equiv_K$ where $\equiv_K$ is a congruence on $T(\Sigma)$ uniquely determined by the isomorphism type of I_K; in concrete terms, (Σ,E) specifies I_K in the sense that E defines $\equiv_E$ on $T(\Sigma)$ and this $\equiv_E$ is $\equiv_K$.

In theory and practice, the problem of specifying a data type is this discussion in reverse. Some finitely generated algebra A is given as modelling some data type whose complete semantics lies *within* the class HOM(A) of all homomorphic images of A. And one has to find an appropriate specification (Σ,E) which defines $T(\Sigma,E) = T(\Sigma)/\equiv_E$ so that the demonstration of correctness for the specification is the proof that $T(\Sigma,E) \cong A$. Or, equivalently, if $A \cong T(\Sigma)/\equiv_A$ then $\equiv_E$ is $\equiv_A$. One favoured method, pioneered in the literature of Programming Methodology, is to take E as a finite set of equations over Σ and to define $\equiv_E$ as the smallest congruence on $T(\Sigma)$ containing the identifications made by E. This natural technique is by no means the only algebraically styled method to win practical approval, but it is the most widely understood and, theoretically, it circumvents inumerable algebraic problems introduced on tampering with the type of axioms allowed in E or with how $\equiv_E$ is constructed. For example, when E contains only equations $ALG^*(\Sigma,E)$ has an initial object and this is $T(\Sigma,E)$; moreover, $HOM(T(\Sigma,E)) = ALG^*(\Sigma,E)$.

However, in MAJSTER [7] appeared a stack-like memory structure along with a plausible argument that it failed to have a finite equational specification. This initiated interest in related methods of specification: allowing conditional equations into E and using auxiliary or hidden operations. A clear account can be found in ADJ [11] where the authors formally prove that a simple data type *odd* has *no* finite equational specification but admits a neat finite conditional specification, as well as a finite equational specification involving hidden operators. Independently, in [1], we reported that the algebra

$$(\{0,1,\ldots\};\ 0,\ x+1,\ x^2)$$

has neither a finite equational specification nor a finite conditional equation specification. (For information on other methods see ADJ [11], BERGSTRA & TUCKER [1,2], KAMIN [5], MAJSTER [8] and the references there cited.)

While for many theoretical purposes it is wise to allow *any* finitely generated algebra A to represent some data type semantics, in seeking general theorems which assert a specification technique is powerful enough to characterise broad classes of algebras it is sensible to hypothesise these algebras are at least constructive. We choose to look only at those A which are *finite, computable, semicomputable* or *cosemicomputable*; these latter three categories putting on a proper semantical/algebraic foundation the (quasi-syntactic) ideas that $\equiv_A$ is a decidable, recursively enumerable, or co-r.e. relation on $T(\Sigma)$ respectively.

Here we are concerned exclusively with what we think the most important condition: computability. To complete the backcloth for the theorem we prove, we have only to mention that in our [1] it was shown that every computable data type possessed a finite, equational hidden enrichment specification but the methods we used give no hint that such specifications could be chosen either simply or uniformly.

(The finite algebras, incidentally, we postpone since we consider them sufficiently distinct and interesting, mathematically, to warrant a paper of their own, c.f. the results mentioned in [1,3]. The semicomputable and cosemicomputable algebras we will deal with in a comparative study of initial and terminal algebra specification techniques.)

Obviously, we have already assumed the reader quite familiar with the informal and technical issues to do with algebraic specification techniques; for this the basic reference is ADJ [4]; to conclude this section we settle the algebraic notation, and give the less widely known algebraic definitions, used in what follows.

Typically, a many-sorted algebras A consists of a finite family $A_1,\ldots,A_n$ of *domains* together with a finite family of operations of the form

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

for $k \in \omega$, the natural numbers, and $\lambda_i, \mu \in \{1,\ldots,n\}$, $1 \leq i \leq k$. Relations associated to A are subsumed in this description under the assumption that one of the domains is the Boolean $B = \{0,1\}$. The signature Σ_A of A carries names for each domain, called *sorts*, symbolic names for the operations and for a finite number of distinguished elements of A. Although a sort is formally a numeral i we occasionally refer to a domain A_i as a sort.

Let $\equiv$ be an equivalence relation on the many-sorted algebra A. A *traversal* for $\equiv$ is a family of sets $J_\lambda = \{a_i^\lambda : i \in I_\lambda, a_{i_\lambda}^\lambda \in A_\lambda\}$, $1 \leq \lambda \leq n$, such that for each $b \in A_\lambda$ there is one, and only one, $a_i^\lambda \in J_\lambda$ for which $b \equiv a_i^\lambda$.

Let Σ be a signature. By $T_\Sigma[X]$ we denote the Σ-algebra of polynomials over Σ in the many-sorted list of indeterminates $X = ({}^{\lambda_1}X_1,\ldots,{}^{\lambda_k}X_k)$ where ${}^{\lambda_i}X_i$ is some indeterminate of sort $\lambda_i \in \omega$. An equation over Σ is a pair $(t(X),t'(X))$ of polynomials over Σ of the same sort and which we hereafter write $t(X) = t'(X)$.

Let E be a set of equations over Σ. Then by $T(\Sigma,E)$ we mean the Σ-algebra $T(\Sigma)/{\equiv_E}$ where $\equiv_E$ is the smallest congruence on $T(\Sigma)$ containing the set

$$D_E = \{(t(s_1^{\lambda_1},\ldots,s_k^{\lambda_k}),t'(s_1^{\lambda_1},\ldots,s_k^{\lambda_k})) : t(X) = t'(X) \in E \ \& \ s_i^{\lambda_i} \in T(\Sigma) \text{ of sort } \lambda_i,\ 1 \leq i \leq k\}.$$

A many-sorted algebra A has a *finite equational specification* (Σ,E) if $\Sigma_A = \Sigma$, E is a finite set of equations over Σ, and $A \cong T(\Sigma,E)$.

Let A be a many-sorted algebra A of signature Σ_A. Let Σ be a signature $\Sigma \subset \Sigma_A$ and having the sorts of 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.

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

The algebra A is said to be Σ-*minimal* if $A|_{\Sigma} = \langle A \rangle_{\Sigma}$.

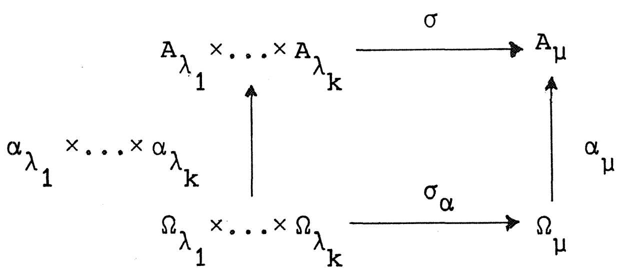
A many-sorted algebra A has a *finite, equational hidden enrichment specification* (Σ,E) if $\Sigma_A \subset \Sigma$, and Σ contains exactly the sorts of Σ_A, E is a finite set of equations over Σ such that

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

Henceforth whenever one signature is contained within another it is to be assumed they contain precisely the same sorts.

2. COMPUTABLE ALGEBRAS

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



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

A is a *computable many-sorted algebra* if, in addition, for each $1 \le i \le n$ the relation $\equiv_{\alpha_i}$ defined on Ω_i by

$$x \equiv_{\alpha_i} y \quad \text{iff} \quad \alpha_i(x) = \alpha_i(y) \text{ in } A_i$$

is recursive.

These definitions are based upon work of M.O. RABIN [10] and, in particular, A.I. MAL'CEV [8] aimed at creating a theory of computable algebraic systems. Noteworthy for us is the fact that they are completely formal notions and that computability is now a *finiteness condition* of Algebra: *an isomorphism invariant possessed of all finite structures.*

In case A is effectively presented, combining the $\Omega_1,\ldots,\Omega_n$ and the $\alpha_1,\ldots,\alpha_n$ we can obtain a recursive many-sorted algebra of numbers Ω of the same signature Σ as A 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 of A with the recursiveness of the congruence $\equiv_\alpha$. The pair (Ω,α) consisting of the algebra Ω and epimorphism α we refer to as *effective*, or *recurive*, *coordinatisations* of A accordingly.

This lemma was proved in our [1]:

2.1 LEMMA. *Every computable many-sorted algebra* A *is isomorphic to a recursive number algebra* Ω *each of whose numerical 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 domain* A_i *is infinite or finite of cardinality* m.

A reference for the elementary theory of the recursive functions is MACHTEY & YOUNG [6], unfortunately our main tool is in no way elementary:

Let $\mathbb{Z}[X_1,\ldots,X_n]$ denote the ring of polynomials in indeterminates $X_1,\ldots,X_n$. A set $\Omega \subset \omega^k$ is said to be *diophantine* if there exists a polynomial $p \in \mathbb{Z}[X_1,\ldots,X_k,Y_1,\ldots,Y_\ell]$ such that

$$(x_1,\ldots,x_k) \in \Omega \iff \exists y_1,\ldots,y_\ell \in \omega.\ p(x_1,\ldots,x_k,y_1,\ldots,y_\ell) = 0.$$

Of course, as far as the class of diophantine sets is concerned one could equivalently place any $r \in \omega$ in the position of 0 in the definition.

Clearly, each diophantine set is recursively enumerable; the converse is due to Y. Matijacevič:

2.2 DIOPHANTINE THEOREM

All recursively enumerable sets are diophantine

A good exposition of this result appears in MANIN [9].

3. PROOF OF THE THEOREM IN THE SINGLE-SORTED CASE

Let A be an infinite computable algebra with signature Σ and which is arbitrarily chosen. By Lemma 2.1, we can identify A with a recursive number algebra R whose domain is ω and concentrate on providing R with a finite, equational hidden enrichment specification wherein the number of hidden operations is independent of our choice of R. We do this by adding 1 constant and 6 functions to R to construct a new recursive number algebra R_0 such that $R_0|_\Sigma = \langle R_0\rangle_\Sigma = R$; and by then showing R_0 has a finite equational specification (Σ_0, E_0) in which only E_0 is dependent on R. The choice of the new functions and the structure of E_0 is determined by the following technical construction.

Let $f: \omega^k \to \omega$ be recursive. Thus the graph of f,

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

is recursively enumerable. By the *Diophantine Theorem,* there exists a polynomial $p_f \in \mathbb{Z}[x_1,\ldots,x_k,x_{k+1},y_1,\ldots,y_\ell]$ such that

$$\text{graph}(f) = \{(x_1,\ldots,x_k,x_{k+1}): x_1,\ldots,x_{k+1} \in \omega \;\&$$

$$\exists y_1,\ldots,y_\ell \in \omega.\; p_f(x_1,\ldots,x_{k+1},y_1,\ldots,y_\ell) = 1\}.$$

We take p_f and separate it into polynomials $p_f^+, p_f^- \in \omega[x_1,\ldots,x_k,x_{k+1},$ $y_1,\ldots,y_\ell]$ by combining those monomials $a_\lambda m_\lambda(x,y)$ whose coefficients $a_\lambda \in \mathbb{Z}$ are positive and negative respectively so that

$$\text{graph}(f) = \{(x_1,\ldots,x_k,x_{k+1}) \in \omega:$$

$$\exists y_1,\ldots,y_\ell \in \omega.[p_f^+(x_1,\ldots,x_{k+1},y_1,\ldots,y_\ell) \dot{-}$$

$$p_f^-(x_1,\ldots,x_{k+1},y_1,\ldots,y_\ell) = 1]\}.$$

Thus dissolving reference to $\mathbb{Z}$ in our enumeration of graph(f).

Define the multiargument function $h_f(x,y) = \min(p_f^+(x,y) \dot{-} p_f^-(x,y), 2).\mathrm{mod}\ 2$.

Notice that h_f is a polynomial function of this list Λ of functions, and a constant, over ω:

$$0,\ x+1,\ x+y,\ x\dot{-}y,\ x\cdot y,\ \min(x,2),\ x.\mathrm{mod}\ 2 \qquad (\Lambda)$$

And that

$$h_f(x,y) = \begin{cases} 0 & \text{if } p_f^+(x,y) \dot{-} p_f^-(x,y) \neq 1; \\ 1 & \text{if } p_f^+(x,y) \dot{-} p_f^-(x,y) = 1. \end{cases}$$

In particular: if $h_f(x_1,\ldots,x_k,x_{k+1},y) = 1$ then $f(x_1,\ldots,x_k) = x_{k+1}$.

Therefore, for all $x = (x_1,\ldots,x_k,x_{k+1})$, $y = (y_1,\ldots,y_\ell)$

$$h_f(x,y).f(x_1,\ldots,x_k) = x_{k+1}.h_f(x,y) \qquad (*)$$

We set R_0 to be R with the list Λ adjoined. Let the signature Σ_0 of R_0 be Σ with signature Γ carrying these names for the functions of Λ:

$$0,\ S,\ SUM,\ DIFF,\ PROD,\ MIN_2,\ MOD_2.$$

Here is a prescription for a finite set of equations E_0 over Σ_0 to specify R_0.

First, for each constant $\underline{c} \in \Sigma$ naming numerical constant c in R, set

$$\underline{c} = S^c(0). \qquad (0)$$

Next came equations to define the functions in the list Λ; this is routine:

Addition SUM(X,0) = X (1)

SUM(X,S(Y)) = S(SUM(X,Y))

Subtraction DIFF(X,0) = X (2)

DIFF(0,Y) = 0

DIFF(X,S(Y)) = DIFF(DIFF(X,Y),S(0))

Multiplication	$PROD(X,0) = 0$	(3)
	$PROD(X,S(Y)) = SUM(PROD(X,Y),X)$	
Minimum	$MIN_2(0) = 0$	(4)
	$MIN_2(S(0)) = S(0)$	
	$MIN_2(S^2(Y)) = S^2(0)$	
Modulus	$MOD_2(0) = 0$	(5)
	$MOD_2(S(0)) = S(0)$	
	$MOD_2(S^2(Y)) = MOD_2(Y)$	
Zero	$PROD(X,0) = PROD(0,X) = 0$	(6)
Unity	$PROD(X,1) = PROD(1,X) = X$	(7)

Finally, we add an equational translation of (*) for each $\underline{f} \in \Sigma$ naming operation f of R. For each recursive operation f of R we make some h_f. To say h_f is a polynomial function of the list of functions Λ of course *means* h_f is the map $\omega^{k+1+\ell} \to \omega$ defined by some formal polynomial $H_f \in T_\Gamma[X_1,\ldots,X_k, X_{k+1},Y_1,\ldots,Y_\ell]$; we abbreviate this by $H_f(X,Y) \in T_\Gamma[X,Y]$.

For each h_f we choose an H_f and add the equation

$$H_f(X,Y)\cdot\underline{f}(X_1,\ldots,X_k) = X_{k+1}\cdot H_f(X,Y).$$

This is all of E_0. Notice it contains 17+p+q equations.

Let $\equiv$ abbreviate $\equiv_{E_0}$ and write elements of $T(\Sigma_0,E_0) = T(\Sigma_0)/\equiv$ in the form [t] for $t \in T(\Sigma_0)$.

We claim the map $\phi\colon R_0 \to T(\Sigma_0,E_0)$ defined by $\phi(n) = [S^n(0)]$ is an isomorphism. To prove ϕ bijective is to prove

<u>3.1 LEMMA</u>. *The set* $\{S^n(0)\colon n \in \omega\}$ *is a traversal for* $\equiv$.

<u>PROOF</u>. We leave to the reader the task of checking $S^n(0) = S^n(0) \Leftrightarrow n = m$. To show that for $t \in T(\Sigma_0)$ there is some $S^n(0)$ so that $t \equiv S^n(0)$ we use induction on the complexity of t.

The basis sees t as a constant of Σ_0 and is immediate from equations in E_0 of type (0). With a view to proving ϕ a homomorphism later on, we do the induction step in the form of this lemma.

<u>3.2 LEMMA</u>. *Let* $\underline{\lambda}$ *be a* k *-ary operation symbol of* Σ_0 *naming the function* λ *of* R_0. *Let* $s_1,\ldots,s_k \in T(\Sigma_0)$. *If* $s_i \equiv S^{z_i}(0)$ *for* $1 \le i \le k$ *then* $\underline{\lambda}(s_1,\ldots,s_k) \equiv S^{\lambda(z_1,\ldots,z_k)}(0)$.

PROOF. Let $t = \underline{\lambda}(S^{z_1}(0),\ldots,S^{z_k}(0))$. By considering cases for $\underline{\lambda} \in \Sigma_0$ we show $t \equiv S^{\lambda(z_1,\ldots,z_k)}(0)$. It is routine to do this, but important to take first the operation symbols of Γ in order and then the operation symbols of Σ. The first non-trivial case is addition.

Here $t = \mathrm{SUM}(S^u(0),S^v(0))$, say, and we argue by induction on v. The basis $v = 0$ is immediate from equation (1). So assume the lemma true for $v = k$ and consider $v = k+1$.

$$
\begin{aligned}
\mathrm{SUM}(S^u(0),S^v(0)) &\equiv \mathrm{SUM}(S^u(0),S(S^k(0))) & \\
&\equiv S(\mathrm{SUM}(S^u(0),S^k(0))) & \text{by equation (1);} \\
&\equiv S(S^{u+k}(0)) & \text{by induction;} \\
&\equiv S^{u+k}(0) &
\end{aligned}
$$

Now for $\underline{\lambda} \in \Gamma$ the cases follow the same pattern though the case of SUM, just proven, is used as a lemma for multiplication; we omit these details and consider $\underline{\lambda} = \underline{f} \in \Sigma$.

Substituting $S^{z_1}(0)$ for $1 \le i \le k$ and an arbitrary list $\vec{r} = (r_1,\ldots,r_\ell) \in T(\Sigma_0)$ into equation (**) results in this identity, wherein $z = (z_1,\ldots,z_k)$:

$$
H_f(S^{z_1}(0),\ldots,S^{z_k}(0),S^{f(z)}(0),\vec{r})\cdot \underline{f}(S^{z_1}(0),\ldots,S^{z_k}(0)) \equiv
$$
$$
S^{f(z)}(0)\cdot H_f(S^{z_1}(0),\ldots,S^{z_k}(0),S^{f(z)}(0),\vec{r}).
$$

Thanks to the multiplication equation (3) and equations (6) and (7), it is sufficient to prove there exists $\vec{r}$ such that

$$
H_f(S^{z_1}(0),\ldots,S^{z_k}(0),S^{f(z)}(0),\vec{r}) \equiv S(0).
$$

Since there exist $y_1,\ldots,y_\ell \in \omega$ such that $h_f(z,f(z),y_1,\ldots,y_\ell) = 1$ we choose $\vec{r}$ to be $S^{y_1}(0),\ldots,S^{y_k}(0)$ whence the identity follows from a new lemma:

3.3 LEMMA. *Let $\tau \in T_\Gamma[X_1,\ldots,X_n]$ define function $\psi\colon\ \omega^n \to \omega$. Then for all $S^{z_1}(0),\ldots,S^{z_n}(0) \in T(\Gamma)$ substituting into $\tau(X_1,\ldots,X_n)$ we obtain*

$$\tau(S^{z_1}(0),\ldots,S^{z_n}(0)) \equiv S^{\psi(z_1,\ldots,z_n)}(0).$$

PROOF. We argue by induction on the complexity of τ. The basis is trivial as τ is either 0 or X_i. Assume as induction hypothesis that the lemma is true of all polynomials over Γ of lower complexity. Let $\tau(X) = \underline{\lambda}(\tau_1(X),\ldots,\tau_k(X))$ where $\underline{\lambda} \in \Gamma$ names function $\lambda \in \Lambda$ and $X = (X_1,\ldots,X_n)$. Let τ_i define $\psi_i\colon \omega^n \to \omega$, so the induction hypothesis says

$$\tau_i(S^{z_1}(0),\ldots,S^{z_n}(0)) \equiv S^{\lambda_i(z_1,\ldots,z_n)}(0)$$

for $1 \le i \le k$. To complete the proof we simply consider cases for $\underline{\lambda} \in \Gamma$.

For example, let $\underline{\lambda}$ = SUM. Then

$$\begin{aligned}\tau(S^{z_1}(0),\ldots,S^{z_k}(0)) &\equiv \mathrm{SUM}(S^{u(z_1,\ldots,z_n)}(0),S^{v(z_1,\ldots,z_n)}(0))\\ &\equiv S^{u(z_1,\ldots,z_n)+v(z_1,\ldots,z_n)}(0)\end{aligned}$$

by the already proven case of addition of Lemma 3.2. The other cases proceed exactly in the same way. Q.E.D.

This completes the proofs of Lemma 3.2 and 3.1.

To check ϕ is a homomorphism can be done using Lemma 3.2:

$$\begin{aligned}\phi(\lambda(x_1,\ldots,x_n)) &= [S^{\lambda(x_1,\ldots,x_k)}(0)]\\ &= [\underline{\lambda}(S^{x_1}(0),\ldots,S^{x_k}(0))] && \text{by Lemma 3.2;}\\ &= \underline{\lambda}([S^{x_1}(0)],\ldots,[S^{x_k}(0)]) && \text{by definition of } \underline{\lambda} \text{ on } T(\Sigma_0,E_0);\\ &= \underline{\lambda}(\phi(x_1),\ldots,\phi(x_k)).\end{aligned}$$

This completes the proof of the theorem in the single-sorted case.

4. THE MANY-SORTED CASE

Dispensing with the case that A is finite, we assume A to be a computable, finitely generated many-sorted algebra with at least one domain infinite. Without loss of generality we can assume these domains to be $A_1,\ldots,A_{n_I}$, $B_1,\ldots,B_{n_F}$ where the A_i are infinite and the B_i are finite of cardinality b_i+1. Lemma 2.1 identifies A with a recursive many-sorted algebra of numbers R with domains $\Omega_1,\ldots,\Omega_{n_I}$ and $\Gamma_1,\ldots,\Gamma_{n_F}$ where $\Omega_i = \omega$ for $1 \le i \le n_I$ and $\Gamma_i = \{0,1,\ldots,b_i\}$ for $1 \le i \le n_F$. When not interested in the cardinality of a domain of R we refer to it as some R_i, $1 \le i \le n_I+n_F$. We wish to give R a finite equational hidden enrichment specification (Σ_0,E_0) which meets the conditions mentioned in the theorem.

The idea is to build a mechanism to simulate the many-sorted algebra R over its first infinite domain Ω_1 and to handle the encoding within Ω_1 as in the single-sorted case. It is this machinery we add to R to make a new recursive number algebra R_0 which we provide with a finite equational specification (Σ_0,E_0) having the appropriate independence properties.

To begin, we add the list Λ, of the last section, as a new constant and new functions to the infinite sort Ω_1.

For each sort $i \neq 1$, add as a new constant of sort i the number $0 \in R_i$.

For each infinite sort $i \neq 1$, add the successor function x+1 to Ω_i and for each finite sort i add its imitation,

$$ {}^{i}succ(x) = \begin{cases} x+1 & \text{if } x < b_i \\ b_i & \text{otherwise.} \end{cases} $$

Next add for every sort $i \neq 1$ a map ${}^{i}copy\colon \Omega_1 \to R_i$ defined by ${}^{i}copy(x) = x$ when i is an infinite sort and by

$$ {}^{i}copy(x) = \begin{cases} x & \text{if } x < b_i \\ b_i & \text{otherwise} \end{cases} $$

when i is a finite sort.

And, finally, we add for every sort $i \neq 1$ the function ${}^{i}g\colon R_i^2 \to R_i$ defined by

$$ {}^{i}g(x,y) = \begin{cases} 0 & \text{if } y = 0 \\ x & \text{otherwise.} \end{cases} $$

To understand the role of these ${}^{i}g$, observe that on interpreting each operation $f^{\lambda,\mu}$ of R as just a recursive function $f\colon \omega^k \to \omega$ in the obvious way and constructing an h_f as in section three, but thinking of it as a function on the first domain Ω_1, we may formally write

$$^{\mu}g(f^{\lambda,\mu}(^{\lambda_1}copy(x_1),\ldots,{}^{\lambda_k}copy(x_k)),{}^{\mu}copy(h_f(x,z,y))) =$$

$$= {}^{\mu}g({}^{\mu}copy(z),{}^{\mu}copy(h_f(x,z,y)))$$

where $x = (x_1,\ldots,x_k) \in \Omega_1^k$, $y = (y_1,\ldots,y_\ell) \in \Omega_1^\ell$ and $z \in \Omega_1$.

Notice, too, that we have added to R $n = n_I+n_F$ constants and $6+3(n-1) = 3n+3$ new functions and that obviously $R_0|_\Sigma = \langle R_0\rangle_\Sigma = R$.

Let Σ_0 be the signature of R_0 where ${}^{i}0$, ${}^{i}S$, ${}^{i}COPY$, ${}^{i}G$ name the zero successor and other maps associated to sort i; we let 0,S name name the zero and successor function assigned to Ω_1. The equations E_0 which specify R_0 are as follows.

First, if ${}^{i}\underline{c} \in \Sigma$ is a constant naming $c \in R_i$ then add

$${}^{i}\underline{c} = {}^{i}S^{c}({}^{i}0).$$

(Here $1 \leq i \leq n$.)

Next add all the equations (1)-(7) associated with the list of functions Λ.

For each finite sort i we add the equation

$${}^{i}S({}^{i}S^{b_i}({}^{i}0)) = {}^{i}S^{b_i}({}^{i}0).$$

For each sort $i \neq 1$, add the equations

$${}^{i}COPY(0) = {}^{i}0$$

$${}^{i}COPY(S(X)) = {}^{i}S({}^{i}COPY(X))$$

where X is a variable of sort 1.

For each sort $i \neq 1$, add the equations

$$^{i}G(^{i}X,{}^{i}0) = {}^{i}0$$

$$^{i}G(^{i}X,{}^{i}S(^{i}Y)) = {}^{i}X$$

where $^{i}X, {}^{i}Y$ are variables of sort i.

Lastly, for each $\underline{f}^{\lambda,\mu} \in \Sigma$ naming $f^{\lambda,\mu}$, an operation of R, we treat $f^{\lambda,\mu}$ as a recursive function $\omega^{k} \to \omega$, make h_f, and choose some H_f, as a polynomial of sort 1, so as to add the equation

$$^{\mu}G(\underline{f}^{\lambda,\mu}(^{\lambda_1}COPY(X_1),\ldots,{}^{\lambda_k}COPY(X_k)),{}^{\mu}COPY(H_f(X,Z,Y))) =$$

$$= {}^{\mu}G(^{\mu}COPY(Z),{}^{\mu}COPY(H_f(X,Z,Y)))$$

where $X = (X_1,\ldots,X_k)$, $Y = (Y_1,\ldots,Y_\ell)$ and Z are variables of sort 1.

This being all of E_0, notice that if A possesses p constants and q operations then E_0 contains

$$p + 17 + n_F + 2(n-1) + 2(n-1) + q = 17+p+q+4(n-1)+n_F$$

equations as required. It remains to prove $R_0 \cong T(\Sigma_0,E_0)$ and to do this we follow exactly the same strategy as in section three.

Corresponding to Lemma 3.1, the family of sets $J_i = \{^{i}S^{z}(^{i}0)\colon z \in R_i\}$ for $1 \le i \le n$, is proved to be a traversal for $\equiv_{E_0}$. In particular, Lemma 3.2 is lifted by a simple inductive argument on term complexity, involving case distinctions based on Σ_0. We look at the induction step in case the leading function symbol of term $t \in T(\Sigma_0)$ is $\underline{f}^{\lambda,\mu}$, naming $f^{\lambda,\mu}$.

It is assumed $\tau_1,\ldots,\tau_k$ are terms of sorts $\lambda_1,\ldots,\lambda_k$ respectively and that for $1 \le i \le k$ $\tau_i \equiv_{E_0} {}^{\lambda_i}S^{z_i}(^{\lambda_i}0)$ for $z_i \in \omega$. We are to show $t = \underline{f}^{\lambda,\mu}(\tau_1,\ldots,\tau_k) \equiv_{E_0} {}^{\mu}S^{f^{\lambda,\mu}(z_1,\ldots,z_n)}(^{\mu}0)$.

Taking this identity as a trivial lemma:

$$^{i}COPY(S^{z}(0)) = {}^{i}S^{z}(^{i}0) \quad \text{for } 1 \le i \le n \text{ and } z \in \omega$$

we have $t \equiv_{E_0} \underline{f}^{\lambda,\mu}(^{\lambda_1}COPY(S^{z_1}(0)),\ldots,{}^{\lambda_k}COPY(S^{z_k}(0)))$. Thanks to the equations defining $^{\mu}G$, $^{\mu}COPY$ and the identity involving $\underline{f}^{\lambda,\mu}$, we have only to prove there exist some numbers $y_1,\dots,y_\ell$ such that

$$H_f(S^{z_1}(0),\dots,S^{z_k}(0),S^{f(z_1,\dots,z_k)}(0),S^{y_1}(0),\dots,S^{y_k}(0)) \equiv_{E_0} S(0).$$

At this point we return the reader to the argument of Lemma 3.3.

That the family of mappings defined ${}^{i}\phi(z) = [{}^{i}S^{z}({}^{i}0)]$ for $1 \leq i \leq n$ is an isomorphism $R_0 \to T(\Sigma_0,E_0)$ is now obvious.

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.

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

[4] 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.

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

[6] MACHTEY, M. & P. YOUNG, *An introduction to the general theory of algorithms*, North-Holland, New York, 1978.

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

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

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

[10] MANIN, Y., *A course in mathematical logic*, Springer-Verlag, New York, 1977.

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

[12] THATCHER, J.W., E.G. WAGNER, & J.B. WRIGHT, *Data type specification: parameterization and the power of specification techniques*, IBM-T.J. Watson Research Center Report RC 7757, Yorktown Heights, 1979.