stichting
mathematisch
centrum

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 124/79 NOVEMBER

J.A. BERGSTRA & J.V. TUCKER

A CHARACTERISATION OF COMPUTABLE DATA TYPES BY MEANS OF A FINITE, EQUATIONAL SPECIFICATION METHOD

Preprint

2e boerhaavestraat 49 amsterdam

BIBLIOTHEEK MATHEMATISCH CENTRUM
AMSTERDAM

CWI BIBLIOTHEEK
3 0054 00060 5494

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

A characterisation of computable data types by means of a finite, equational specification method [*)]

by

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

ABSTRACT

Within the framework of the ADJ Group's algebraic theory of data types, we are able to give a characterisation of those data types and data structures whose semantics are constructive in terms of the structural properties of algebraically styled "proof systems" available for their definition. It is proved that the computable data types are *precisely* those which may be defined by deductive systems, equationally specified in a finite way, and which satisfy two simple conditions on the deductions which may take place within them.

KEY WORDS & PHRASES: *algebraic data types, equational specifications with hidden functions, algebraic replacement systems, weak and strongly normalised Church-Rosser sytems, computable algebras*

[*)] 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

By refining the construction of finite, equational hidden function specifications of data types, as these are made in the initial algebra methodology of the ADJ Group, we are able to give an algebraic characterisation of the computable data types and data structures. Our technical motivation are the simple notions of strong and weakly normalised Church-Rosser replacement systems studied in the λ-Calculus, and in plain mathematical terms the theorem we prove is this.

THEOREM. *Let* A *be a many-sorted algebra finitely generated by elements named in its signature. Then the following statements are equivalent:*

1. A *is computable.*
2. A *possesses a finite, equational hidden enrichment replacement system specification which is Church-Rosser and strongly normalising.*
3. A *possesses a finite, equational hidden enrichment replacement system specification which is Church-Rosser and weakly normalising.*

The unexplained concepts are carefully defined in section two, on replacement system specifications, and in section three, on computable algebras. In sections four and five we prove the theorem. Section one explains in detail the theoretical issues to do with data types which the theorem attempts to resolve.

This paper continues our studies on the adequacy and power of definition of algebraic specification methods for data types which we began in [1], see also [2]. Here the reader is assumed well versed in the initial algebra specification methods of the ADJ GROUP [3], see also KAMIN [4]; knowledge of our [1] is desirable but not, strictly speaking, essential. Prior exposure to the λ-Calculus is not required, of course.

1. INITIAL ALGEBRA SEMANTICS AND DATA TYPES

A *data structure* is defined to be a many-sorted algebra A finitely generated from initial values $a_1,\ldots,a_n \in A$ named in its signature Σ. A *data type* is defined to be any class K of such data structures of common signature. At the heart of the ADJ Group's theory of data types is the idea that the semantics which characterise a data type K should be invested in the construction of an initial algebra I_K for K with the result that every data structure $A \in K$ is uniquely definable, up to isomorphism, as an epimorphic image of I_K. In its turn, this initial algebra I_K is uniquely definable, again up to isomorphism, as a factor algebra of the syntactic algebra $T(\Sigma)$ of all terms over Σ because $T(\Sigma)$ is initial for the class $ALG(\Sigma)$ of all Σ-algebras. Let $I_K \cong T(\Sigma)/\equiv_K$ where $\equiv_K$ is a congruence for which $t \equiv_K t'$ means that the terms t and t' are identical syntactic expressions as far as the semantics of K is concerned. Observe that in these circumstances we may plausibly call a data type (semantics for) K *computable* when $\equiv_K$ is decidable on $T(\Sigma)$. And that the problem of syntactically specifying the data type K can be investigated through the problem of specifying the congruence $\equiv_K$.

The preferred method of prescribing $\equiv_K$ is to use a finite set of equations or conditional equations E over $T(\Sigma)$ to establish a basic set of identifications $D_E \subset T(\Sigma) \times T(\Sigma)$ and to take $\equiv_K$ as the smallest congruence $\equiv_E$ on $T(\Sigma)$ containing D_E. With reference to our [1], it is known that this method will not define all computable data types, but that it is able to define many non-computable ones. Enriching the method to allow the use of a finite number of hidden functions does enable it to specify *any* computable data type, but, of course, greatly expands the number of non-computable data types it defines.

Our proposal here is to determine the congruences for initial algebras by means of *replacement systems*. A replacement system is intended to formalize a system of deductions, governed by simple algebraic substitution rules, within which a deduction $t \rightarrow t'$ says that the rôle of t can be played by t' as far as the semantics of K is concerned, meaning $t \rightarrow t'$ implies $t \equiv_K t'$, but not conversely. What we do is to make an analysis of the congruence $\equiv_K$ through the structural behaviour of algebraically styled "proof systems" for it and from this and an appropriate specification machinery we are able to guess the classification theorem for computable data types. That this is *precisely* the theorem stated in the introduction comes from the reflection that the semantics of a type K is *supposed* to be uniquely determined up to isomorphism with an initial algebra I_K, and not by a particular syntactical construction. Since the computability of $\equiv_K$ means the computability of the algebra $T(\Sigma)/\equiv_K$ under our definition and, in particular, since this notion of an algebra's computability is an isomorphism invariant, we can erase all mention of syntax in the semantical concept of a computable data type and identify these with the computable algebras.

So with regard to the content of our theorem, the reader may care to consider the ease with which statement (3) implies (1) is proved as evidence for the natural significance of strong and weakly normalised Church-Rosser replacement system specifications while the implication (1) implies (2) may be considered as the hard won answer to the question about adequacy: *Do these specifications define all the data types one wants*?

Because of the novelty of the specification technique and the involved proof of the theorem we shall work out most of the material in the case of a *single sorted algebra* after which it becomes much easier for us to explain, and the reader to understand, the proof of the theorem in the many-sorted case.

In what follows ω denotes the set of natural numbers.

2. REPLACEMENT SYSTEMS AND THEIR SPECIFICATION

The technical point of departure is the idea of a traversal for an equivalence relation. Let A be a set and $\equiv$ an equivalence relation on A. A *traversal* for $\equiv$ is a set $J = \{t_i : i \in I\} \subset A$ wherein

(i) for each $a \in A$ there is some $t \in J$ such that $t \equiv a$; and

(ii) if $i,j \in I$ and $i \neq j$ then $t_i \not\equiv t_j$

Consider an initial algebra specification (Σ,E) for a data type K where Σ gives the signature of K and E is some formula or other for axiomatizing its properties so the defining congruence $\equiv_K$ is $\equiv_E$. The choice of a traversal J for $\equiv_E$ fixes an operational view of the type as it is specified: given $t,t' \in T(\Sigma)$, to decide $t \equiv_E t'$ one imagines having to use E to calculate their prescribed "normal forms" $n,n' \in J$ and on completing these deductions $t \rightarrow_E n$, $t' \rightarrow_E n'$ one checks $n = n'$. The following bit of theory about algebraic replacement systems is made up with this in mind and is meant to abstract the bare essentials of such an operational view of data type specification.

Let A be a set and let R be a reflexive, transitive binary relation on A. We write R as $\to_R$ so to display membership $(a,b) \in R$ by $a \to_R b$ and say a *reduces* (under R or $\to_R$) to b or that b is a *reduct* (under R or $\to_R$) of a; and we shall call R and $\to_R$ a *reduction system* or a *replacement system* on the set A. Following the terminology of the λ-Calculus we make these distinctions:

An element $a \in A$ is a *normal form* for $\to_R$ if there is no $b \in A$ so that $a \neq b$ and $a \to_R b$; the set of all normal forms for $\to_R$ is denoted NF(R).

The reduction system $\to_R$ is *Church-Rosser* if for any $a \in A$ if there are $b_1, b_2 \in A$ so that $a \to_R b_1$ and $a \to_R b_2$ then there is $c \in A$ so that $b_1 \to_R c$ and $b_2 \to_R c$.

The reduction system $\to_R$ is *weakly normalising* if for each $a \in A$ there is some normal form $b \in A$ so that $a \to_R b$.

The reduction system $\to_R$ is *strongly normalising* if there does not exist an infinite chain

$$a_0 \to_R a_1 \to_R \cdots \to_R a_n \to_R \cdots$$

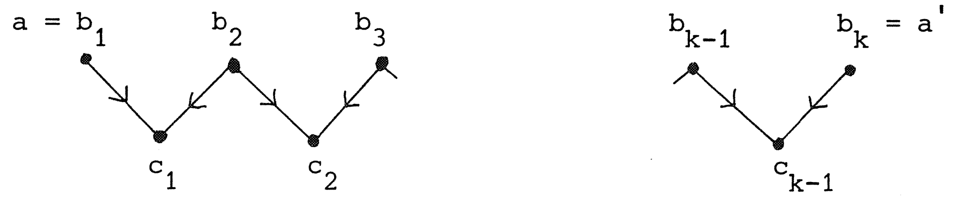
wherein for $i \in \omega$, $a_i \neq a_{i+1}$.

A reduction system is Church-Rosser and weakly normalising if, and only if, every element reduces to a unique normal form. Clearly strong normalisation entails weak normalisation.

Let $\equiv_R$ denote the smallest equivalence relation on A containing $\to_R$. It is an easy exercise to show that for $a, a' \in A$

$a \equiv_R a' \iff$ there is a sequence $a = b_1, \ldots, b_R = a'$ such that for each pair b_i, b_{i+1} there exists a common reduct c_i, $1 \leq i \leq k-1$.

Schematically:



Using this characterisation of $\equiv_R$ it is straight-forward to prove this fact:

2.1. LEMMA. *The replacement system* $\to_R$ *on* A *is Church-Rosser if, and only if, for any* a,a' $\in$ A *if* a $\equiv_R$ a' *then there is* c $\in$ A *so that* a $\to_R$ c *and* a' $\to_R$ c.

2.2. LEMMA. *Let* $\to_R$ *be a Church-Rosser weakly normalising replacement system on* A. *Then the set of normal forms* NF(R) *is a traversal for* $\equiv_R$.

PROOF. Since every element a $\in$ A reduces to some normal form n $\in$ NF(R), the set NF(R) contains representatives for each equivalence class of $\equiv_R$. To check uniqueness, let n,m $\in$ NF(R) and assume n $\equiv_R$ m. By Lemma 2.1, there is c $\in$ A so that n $\to_R$ c and m $\to_R$ c, but since n,m are normal forms n = c, m = c and so n = m. Q.E.D.

Suppose now that A is an algebra then by an *algebraic replacement system* $\to_R$ on the algebra A we mean a replacement system $\to_R$ on the domain of A which is closed under its operations in the sense that for each k-ary operation σ of A,

$$a_1 \to_R b_1, \ldots, a_k \to_R b_k \quad \text{implies } \sigma(a_1, \ldots, a_k) \to_R \sigma(b_1, \ldots, b_k).$$

This next fact is easily proven.

2.3. BASIC LEMMA. *If* $\to_R$ *is an algebraic replacement system on an algebra* A *then* $\equiv_R$ *is a congruence on* A. *If* $\to_R$ *is, in addition, Chruch-Rosser and weakly normalising then the set of normal forms of* $\to_R$ *is a traversal for* $\equiv_R$.

To achieve our goal of constructing algebraic replacement systems on the algebra T(Σ) we need to explain how a replacement system is generated by a set of one-step reductions and, furthermore, how these sets of one-step reductions can be determined from quite arbitrary sets. We must built up this equipment for both set-theoretic and algebraic replacement systems.

Let $\to_R$ be a replacement system on a set A. S $\subset$ A $\times$ A is said to *generate* $\to_R$ *as a set of one-step reductions* if S is reflexive and $\to_R$ is the smallest transitive set containing S, the so called *transitive closure* of S.

Let $\to_R$ be an algebraic replacement system on an algebra A. S $\subset$ A $\times$ A is said to *generate* $\to_R$ *as a set of algebraic one-step reductions* if S is reflexive, S is closed under *unit substitutions* in the following sense: writing $(a,b) \in S$ as $a \to_S b$, for any k-ary operation σ of A, for any $1 \leq i \leq k$ and $a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_k \in A$, and $a \to_S b$ it follows that $\sigma(a_1,\ldots,a_{i-1},a,a_{i+1},\ldots,a_k) \to_S \sigma(a_1,\ldots,a_{i-1},b,a_{i+1},\ldots,a_k)$. And $\to_R$ is the transitive closure of S.

In the set-theoretic case any reflexive set determines a replacement system in its transitive closure. In the algebraic case any reflexive set, closed under unit substitutions, can be shown to determine an algebraic replacement system in its transitive closure. Thus, in either case, starting with an arbitrary set $D \subset A \times A$ one can close it up to the smallest one-step reduction relation containing it, which we write $\to_{D(1)}$, and hence to the set-theoretic or algebraic replacement system $\to_D$ which is its transitive closure.

Let us now apply these ideas to specify algebraic replacement systems on $T(\Sigma)$. Let $T_\Sigma[X_1,\ldots,X_n]$ be the set of all polynomials over Σ in indeterminates $X_1,\ldots,X_n$. Let $T_\Sigma[X] = \bigcup_{n\in\omega} T_\Sigma[X_1,\ldots,X_n]$.

Given a set $E \subset T_\Sigma[X] \times T_\Sigma[X]$ first notice that if $(t,t') \in E$ then without loss of generality we can assume $t,t' \in T_\Sigma[X_1,\ldots,X_n]$ for sufficiently large n.

Then we can define a set $D_E \subset T(\Sigma) \times T(\Sigma)$ by

$$D_E = \{(t(s_1,\ldots,s_n),\ t'(s_1,\ldots,s_n)) \ :\ (t,t') \in E \;\&\; s_1,\ldots,s_n \in T(\Sigma)\}$$

and so obtain the smallest set of algebraic one-step reductions containing D_E, which we write $\to_{E(1)}$, and from it the algebraic replacement relation it generates, denoted $\to_E$. In these circumstances we denote by $NF(\Sigma,E)$ the set of all normal forms of $\to_E$ and by $\equiv_E$ the congruence associated to $\to_E$. Let $T(\Sigma,E) = T(\Sigma)/\equiv_E$. Finally, if $(t(X_1,\ldots,X_n),t'(X_1,\ldots,X_n)) \in E$ then we prefer to write $t(X_1,\ldots,X_n) \geq t'(X_1,\ldots,X_n) \in E$, which we refer to as a *reduction equation*.

From these definitions we see how to equationally specify algebraic replacement systems which in turn specify algebras. Combining these ideas in a particular case we can proceed to the basic concept of the paper.

An algebra A of signature Σ_A is said to have a *finite, equational replacement system specification* (Σ,E) if $\Sigma = \Sigma_A$ and E is a finite set of reduction equations over $T(\Sigma)$ such that the reduction system $\rightarrow_E$ on $T(\Sigma)$ defines a congruence $\equiv_E$ which specifies A by $T(\Sigma,E) \cong A$.

Recall that if A is an algebra of signature Σ and $\Sigma_0 \subset \Sigma$ then

$A|_{\Sigma_0}$ is the algebra obtained from A by deleting the operations of A not named in Σ_0.

$\langle A\rangle_{\Sigma_0}$ is smallest Σ_0-algebra contained in A.

An algebra A of signature Σ_A is said to have a *finite, equational hidden enrichment replacement specification* (Σ,E) if $\Sigma_A \subset \Sigma$ and E is a finite set of reduction equations over $T(\Sigma)$ such that the reduction system $\rightarrow_E$ on $T(\Sigma)$ determines the algebra $T(\Sigma,E)$ and

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

Notice that the algebras which model data structures and the initial algebras of data types are finitely generated by elements named in their signatures. Therefore any such algebra A is automatically *minimal* or *prime* in the sense that $A|_{\Sigma_A} = \langle A\rangle_{\Sigma_A}$.

The structural properties of a specification (Σ,E), such as the Church-Rosser and normalisation properties, are taken from those of its replacement relation $\rightarrow_E$. To gain acquaintance with the specification method, we leave to the reader the proof of this proposition.

2.4. LEMMA. *If* A *is a finite algebra then* A *possesses a finite, equational replacement system specification which is Church-Rosser and strongly normalising.*

And we conclude with a technical fact about set-theoretic replacement systems of use later on. Let A be a set. A set of one-step reductions $\rightarrow_{R(1)}$ which generates a reduction system $\rightarrow_R$ on A is said to be *finitely branching* if for each $a \in A$ the set $\{b \in A: a \rightarrow_{R(1)} b\}$ is finite.

The reduction system $\rightarrow_R$ on A together with its generating set of one-step reductions $\rightarrow_{R(1)}$ is said to be *weakly Church-Rosser* if for any $a \in A$, if $a \rightarrow_{R(1)} b_1$ and $a \rightarrow_{R(1)} b_2$ then there is $c \in A$ such that $b_1 \rightarrow_R c$ and

$b_2 \rightarrow_R c$.

2.5. LEMMA. *Let* $\rightarrow_R$ *be a strongly normalising reduction system on* A *generated by a finitely branching set of one-step reductions* $\rightarrow_{R(1)}$. *If* $\rightarrow_R$ *is weakly Church-Rosser with respect to* $\rightarrow_{R(1)}$ *then* $\rightarrow_R$ *is Church-Rosser.*

PROOF. By a chain of non-trivial one-step reductions from $a \in A$ of length k we mean a sequence $a = a_0 \rightarrow_{R(1)} a_1 \rightarrow_{R(1)} \dots \rightarrow_{R(1)} a_k$ wherein $a_i \neq a_j$ $0 \leq i,j \leq k$. Define $\|a\|$ = maximum length of any such chain from a. This $\|\cdot\| : A \rightarrow \omega$ is a total function thanks to *Konig's Infinity Lemma* and the hypothesis of strong normalisation. We prove the proposition by induction on the value of $\|a\|$. Figure one may be helpful in this.

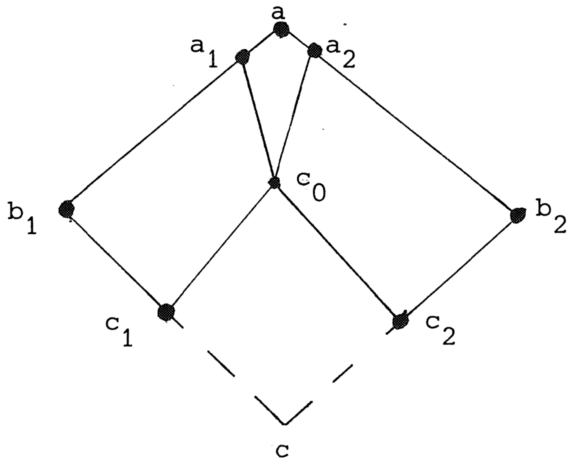
The basis case is automatic because $\|a\| = 0$ iff a is a normal form.

As induction hypothesis assume the Church-Rosser property true of all reducts of $b \in A$ such that $\|b\| < \|a\|$. Let $a \rightarrow_R b_1$ and $a \rightarrow_R b_2$. We take the non-trivial case where a, b_1, b_2 are mutually distinct.

Since $\rightarrow_{R(1)}$ generates $\rightarrow_R$ choose a_1, a_2 such that, for i = 1,2

$$a \rightarrow_{R(1)} a_i \rightarrow_R b_i$$

and notice that $\|a_i\| \leq \|a\|$. Let c_0 be a common reduct of a_1, a_2 supplied by the weak Church-Rosser property. By the induction hypothesis applied to a_1, a_2 we can choose c_1, c_2 as common reducts of c_0, b_1 and c_0, b_2 respectively. Moreover since $\|c_0\| < \|a\|$ we can apply the induction hypothesis again to obtain c as a common reduct of c_1, c_2. Clearly c is also a common reduct of b_1, b_2. Q.E.D.



3. COMPUTABLE ALGEBRAS

Our definition of a *computable algebra* is taken from M.O. RABIN [7] and A.I. MAL'CEV [6], independent papers devoted to founding a general theory of computable algebraic systems and their computable morphisms. Here we will mention only those ideas and facts which contribute to the understanding or to the proof of the theorem; for further information about this powerful theoretical machinery we recommend the interested reader to consult the articles of Rabin and Mal'cev, and our earlier [1] where the subject is treated in the many-sorted case.

A (single sorted!) algebra A is said to be *computable* if there exists a recursive set of natural numbers Ω and a surjection $\alpha\colon \Omega \to A$ such that to each k-ary operation σ of A there corresponds a recursive *tracking function* $\bar{\sigma}\colon \omega^k \to \omega$ which commutes the following diagram,

$$\begin{tikzcd}
A^k \arrow[r, "\sigma"] & A \\
\Omega \arrow[u, "\alpha^k"] \arrow[r, "\bar{\sigma}"] & \Omega \arrow[u, "\alpha"']
\end{tikzcd}$$

wherein $\alpha^k(x_1,\ldots,x_k) = (\alpha x_1,\ldots,\alpha x_k)$. And, furthermore, the relation $\equiv_\alpha$, defined on Ω by $x \equiv_\alpha y$ iff $\alpha(x) = \alpha(y)$ in A, is recursive. In case this relation $\equiv_\alpha$ is recursively enumerable we say A is *semicomputable*.

Both notions, in these formal definitions, become so called *finiteness conditions* of Algebra: isomorphism invariants possessed of a-l finite structures. And also noteworthy is this other invariance property:

If A is a finitely generated algebra computable or semicomputable under both $\alpha\colon \Omega_\alpha \to A$ and $\beta\colon \Omega_\beta \to A$ then α and β are *recursively equivalent* in the sense that there exists recursive funtions f,g which commute the diagram:

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

See MAL'CEV [6].

Given A computable under α then combining the associating tracking functions on the domain Ω makes up a recursive algebra of numbers from which α is an epimorphism to A. Applying the recursiveness of $\equiv_\alpha$ to this obervation it is easy to prove this useful fact.

3.1. LEMMA. *Every computable algebra* A *is isomorphic to a recursive number algebra* Ω *whose domain is the set of natural numbers,* ω, *if* A *is infinite, or else is the set of the first* m *natural numbers,* ω_m, *if* A *is finite of cardinality* m.

We proved this in its many-sorted version in [1]. Obviously, no such isomorphic representation is possible for the semicomputable algebras for otherwise they would be computable.

If A is computable under α then a set $S \subset A^n$ is (α-)*computable* or (α-) *semicomputable* accordingly as $\alpha^{-1}(S) = \{(x_1,\ldots,x_n) \in \Omega^n : (\alpha x_1,\ldots,\alpha x_n) \in S\}$ is recursive or r.e.

3.2. LEMMA. *Let* A *be a computable algebra and* $\equiv$ *a congruence* A. *If* $\equiv$ *is computable or semicomputable then the factor algebra* $A/\equiv$ *is computable or semicomputable accordingly.*

The algebras $T(\Sigma)$ are always computable under any of their standard gödel numberings. Of course, it was this fact we had implicity in mind when we spoke of a data type K being computable when its defining congruence $\equiv_K$ is decidable. Wherever $\equiv_K$ is syntactically determined by some specification mechanism (Σ,E) it is customary to speak of the *word* or *term problem* for (Σ,E) and mean the decidability of $\equiv_K$. In any case, through Lemma 3.2 and isomorphism invariance, we can now redefine a data type to be computable when its initial algebra is computable.

Relying on the reader's experience in constructively manipulating syntax, we set him or her the proof of this lemma as an easy, though instructive, exercise.

3.3. LEMMA. *Let* (Σ,E) *be a finite, equational replacement system specification. Then the basis set* D_E, *the one-step reduction relation* $\rightarrow_{E(1)}$, *the* *replacement system* $\to_E$, *the set of normal forms* NF(Σ,E), *and the congruence* $\equiv_E$ *are all semicomputable. In particular,* T(Σ,E) *is a semicomputable algebra.*

We may now trivially prove

3.4. PROPOSITION. *Let* (Σ,E) *be a finite, equational replacement system specification which is Church-Rosser and weakly normalising. Then* T(Σ,E) *is a computable algebra.*

PROOF. Given $t \in T(\Sigma)$ we can interleave the algorithms enumerating NF(Σ,E) and $\equiv_E$ to seek its normal form which is guaranteed to exist from the weak normalisation hypothesis. Given $t,t' \in T(\Sigma)$, to decide $t \equiv_E t'$ we calculate their normal forms n,n' and using the uniqueness property of Church-Rosser systems we have only to check whether or not $n = n'$. Q.E.D.

The argument of Proposition 3.4 is also that of this companion lemma to Lemma 3.2.

3.5. LEMMA. *Let* A *be a semicomputable algebra with semicomputable congruence* ≡. *If there exists a semicomputable traversal for* ≡ *then the factor algebra* A/≡ *is a computable algebra.*

4. PROOF OF THE THEOREM

A strongly normalising reduction system specification is at the same time a weakly normalising reduction system specification so statement (2) automatically implies statement (3). Since computability is an isomorphism invariant, Proposition 3.4 proves (3) implies (1). Thus this section is devoted to proving statement (3) implies statement (1). The case where A is finite the reader has proved as Lemma 2.4 and so we assume A to be infinite.

By Lemma 3.1, we can take A isomorphic to a recursive number algebra $R = (\omega; f_1,\ldots,f_p,c_i,\ldots,c_q)$ and concentrate on building a replacement system specification for R. First we shall build a complicated recursive number algebra R_0 by adding to R a variety of recursive functions.

Given a total recursive function f: $\omega^k \to \omega$ then, by the *Kleene Normal Form Theorem,* this may be written

$$f(x) = U(\mu z.T(e,x,z))$$

where U and T are the so called *Kleene computation function* and T-*predicate,* respectively, and e is some index for f. Since U and T are primitive recursive so are the functions

$$h(z,x) = U(\mu z' \leq z.[z' = z \vee T(e,x,z')])$$
$$g(z,x) = \begin{cases} 0 & \text{if } \exists z' \leq z.\ T(e,x,z) \\ 1 & \text{otherwise.} \end{cases}$$

From these functions we can define a recursive function

$$t(z,x,0) = h(z,x)$$
$$t(z,x,y+1) = t(z+1,x,g(z+1,x))$$

so that f is factorised into t,h,g in the sense that f(x) = t(0,x,1). (The uninitiated reader should consult M. MACHTEY & P. YOUNG [5].)

R_0 is constructed by adding 0 and the successor function x+1 on ω to R and, for each recursive operation f of R, adding the corresponding factorising functions h,g,t along with the list Δ of all primitive recursive functions used in the primitive recursive definitions of h and g.

Clearly, $R_0|_\Sigma = \langle R_0\rangle_\Sigma = R$ and so it is sufficient to show R_0 has a finite, equational replacement system specification which is Church-Rosser and strongly normalising. Let Σ_0 be the signature of R_0. The specifying reduction equations E_0 in mind are defined as follows. For each operation f,t,h,g of R_0, of the kind last mentioned, if $\underline{f},\underline{t},\underline{h},\underline{g}$ are their corresponding function symbols in Σ_0, then we take

(-1) $\underline{f}(X) \geq \underline{t}(\underline{0},X,S(\underline{0}))$

(0) $\underline{t}(Z,X,\underline{0}) \geq \underline{h}(Z,X)$

$\underline{t}(Z,X,S(y)) \geq \underline{t}(S(Z),X,\underline{g}(S(Z),X))$

where $X = (X_1,\dots,X_k)$.

For each function symbol $\underline{\lambda} \in \Sigma_0$ corresponding to a primitive recursive function λ in the list $\Delta \cup \{h,g\}$ we add equations determined by these case distinctions.

(1) If $\lambda(x_1,\dots,x_k) = x_i$ then add $\underline{\lambda}(X_1,\dots,X_k) \geq X_i$

(2) If $\lambda(y) = y+1$ then add $\underline{\lambda}(Y) \geq S(Y)$.

(3) If $\lambda(x) = \mu(\mu_1(x),\dots,\mu_n(x))$ then add $\underline{\lambda}(X) \geq \underline{\mu}(\underline{\mu}_1(X),\dots,\underline{\mu}_n(X))$ where here $x = (x_1,\dots,x_k)$ and $X = (X_1,\dots,X_k)$.

(4) If
$$\lambda(0,x) = \mu_1(x)$$
$$\lambda(y+1,x) = \mu_2(y,x,\lambda(y,x))$$
then add
$$\underline{\lambda}(\underline{0},X) \geq \underline{\mu}_1(X)$$
$$\underline{\lambda}(S(Y),X) \geq \underline{\mu}_2(Y,X,\underline{\lambda}(Y,X))$$
where, again, x and X are possibly vectors.

Finally, we must take care of the constants of Σ in Σ_0. If $\underline{c}$ names the numerical constant c then add $\underline{c} \geq S^c(\underline{0})$. We number this as equation (5).

Thus (Σ_0,E_0) is a finite, equational replacement system specification and it remains to verify the Church-Rosser property and strong normalisation, and to show $T(\Sigma_0,E_0) \cong R_0$.

Call a term $t \in T(\Sigma)$ *strongly normalising* (with respect to E_0) if there does not exist an infinite chain $t = t_0 \to t_1 \to \dots \to t_n \to \dots$ where for $i,j \in \omega$ $t_i \neq t_j$ and $\to$ is the reduction relation determined by E_0. Most of the theorem is proved on showing.

4.1. LEMMA. *If* t *is strongly normalising then it possesses a unique normal form of the kind* $S^n(0)$ *for some* $n \in \omega$.

4.2. LEMMA. *Every term in* $T(\Sigma)$ *is strongly normalising.*

The proof of Lemma 4.1 verifies the Church-Rosser property and combined with Lemma 4.2 shows our specification (Σ_0,E_0) to be of the required type. Given these lemmas, we know from Basic Lemma 2.3 that $\{S^n(\underline{0}) : n \in \omega\}$ is a traversal for $T(\Sigma_0,E_0)$, and to prove this algebra isomorphic to R_0 we can use the map $\phi(n) = [S^n(\underline{0})]$. Since ϕ is known to be a bijection $R_0 \to T(\Sigma_0,E_0)$, all that must be verified is that ϕ is a homomorphism. This requires an inductive argument on the complexity of terms along the lines of the proof of Lemma 4.2. Because the reasoning is much simpler than that for Lemma 4.2, and routine for any reader with a little algebraic experience, we take the liberty of omitting it. Thus, to complete the theorem it remains for us to prove Lemmas 4.1 and 4.2.

PROOF OF LEMMA 4.1. For $t \in T(\Sigma_0)$ the restriction of $\to$ defines a replacement system on the *set*

$$\mathrm{Red}(t) = \{s \in T(\Sigma_0): t \to s\}$$

which is generated by any set of one-step reductions $\to_1$ for $\to$ also restricted to Red(t). If t is strongly normalising with respect to $\to$ then (Red(t),$\to$) is a strongly normalising set-theoretic replacement system. It is a routine matter to check that $\to$ is weakly Church-Rosser with respect to $\to_1$ by considering term complexity and to see that $\to_1$ is finitely branching. So we may apply Lemma 2.4 to deduce that (Red(t),$\to$) is Church-Rosser as well as strongly normalising. (This together with Lemma 4.1 proves our specification Church-Rosser!) A corollary of this is the fact that t has a normal form with respect to $\to$ and it is unique.

Now we argue that $NF(\Sigma_0,E_0) = \{S^n(\underline{0}): n \in \omega\}$. It is easy to see that $\{S^n(\underline{0}): n \in \omega\} \subset NF(\Sigma_0,E_0)$ because a term $S^n(\underline{0})$ cannot be further reduced by equations from E_0. On the other hand we can rule out all other terms as normal forms by these case distinctions. Let $t \in T(\Sigma_0)$.

If $t = \underline{c} \in \Sigma_0$, a constant naming $c \neq 0$, then equation (5) permits a reduction to $S^c(\underline{0})$ and so since $\underline{c}$ can be reduced it is not a normal form.

If $t = \underline{\lambda}(s_1,\ldots,s_k)$ where $\underline{\lambda}$ is any function symbol of Σ_0 *except* S then, again, there is a reduction to a distinct term to be had from the equations written down for $\underline{\lambda}$ in the construction of E_0.

Finally, if $t = S^n(r)$, where r is a term of the first two kinds, then since r has been seen to possess some non-trivial reduction so does t (as $\to$ is an algebraic replacement system). Q.E.D.

PROOF OF LEMMA 4.2. We prove that each $t \in T(\Sigma_0)$ is strongly normalising by induction on the complexity of t.

As basis consider all constants. Let $\underline{c} \in \Sigma_0$ name the numerical constant c. By inspection of E_0, there is at most one reduction possible from t and this leads to a normal form, *viz*. $\underline{c} \geq S^c(\underline{0})$.

The induction step is precisely this lemma.

4.3. LEMMA. *Let* $s_1,\ldots,s_k \in T(\Sigma_0)$ *be strongly normalising and let* $\underline{\lambda}$ *be a k-ary function symbol of* Σ_0. *Then* $\underline{\lambda}(s_1,\ldots,s_k)$ *is strongly normalising.*

PROOF. First we order the signature Σ_0. For each operation f_i of R let h_i, g_i, t_i be the functions factoring f_i and let Δ_i be the list of primitive recursive functions used in the definitions of the h_i and g_i, those of h_i preceeding those of g_i and each of these two lists ordered by the complexity of the primitive recursive definitions of the h_i and g_i respectively. Thus we order the constants and operations of R_0 into the list

$$0, c_1,\ldots,c_q,\ x+1, \Delta_1,\ldots,\Delta_p,\ h_1,\ldots,h_p, g_1,\ldots,g_p, t_1,\ldots,t_p, f_1,\ldots,f_p$$

and let the signature Σ_0 of R_0 be ordered in this way. We shall now prove the lemma by induction on the position of $\underline{\lambda}$ in the ordering of Σ_0. One general remark, for any term $t = \underline{\lambda}(s_1,\ldots,s_k)$, is that an infinite reduction sequence from t which does not involve a reduction from E_0 determined by $\underline{\lambda}$ would require an infinite reduction sequence from one of its subterms in contradiction to the assumption that they are strongly normalising. Thus in the argument *we need only consider reduction sequences from* $t = \underline{\lambda}(s_1,\ldots,s_k)$ *which apply the reduction equations in* E_0 *written down for* $\underline{\lambda}$.

For this reason the basis $\underline{\lambda} = S$ is obvious. If $t = S(r)$ then inspection of E_0 confirms no reduction from t determined by S to be possible since r is irreducible.

So assume as induction hypothesis that $\underline{\mu}(s_1,\ldots,s_k)$ is strongly normalising for all function symbols $\underline{\mu}$ preceeding $\underline{\lambda}$ in Σ_0. The proof of the induction step divides into 6 cases conveniently distinguished by λ (rather than $\underline{\lambda}$)

CASE 1. $\lambda(x_1,\ldots,x_k) = x_i$

Let $t = \underline{\lambda}(s_1,\ldots,s_k)$. Let $\lambda(s_1',\ldots,s_k')$ be the stage in a reduction sequence from t at which equation (1) is applied, where $s_i \to s_j'$, $1 \leq j \leq k$. Then the next element in the sequence is s_i' and since this is strongly normalising the sequence must terminate.

CASE 2. $\lambda(y) = y + 1$.

This is easy, for the first application of equation (2) in a reduction sequence from $t = \lambda(s)$ introduces a term of the kind considered in the basis of the induction.

CASE 3. $\lambda(x) = \mu(\mu_1(x),\ldots,\mu_n(x))$.

Let $t = \underline{\lambda}(s)$ where $s = (s_1,\ldots,s_k)$ corresponding to $x = (x_1,\ldots,x_k)$. Let $\lambda(s')$ be the stage in a reduction sequence from t at which equation (3) is first applied, where $s' = (s_1',\ldots,s_k')$ and $s_i \to s_i'$, $1 \leq i \leq k$. Then the next element in the sequence is $\underline{\mu}(\underline{\mu}_1(s'),\ldots,\underline{\mu}_n(s'))$. By the induction hypothesis, for each $1 \leq i \leq n$, $\underline{\mu}_i(s')$ is strongly normalising since $\underline{\mu}_i$ preceeds $\underline{\lambda}$ in Σ_0. And since $\underline{\mu}$ also preceeds $\underline{\lambda}$ in Σ_0 another appeal to the induction hypothesis shows the term to be strongly normalising. Hence the sequence must terminate.

CASE 4. $\lambda(0,x) = \mu_1(x)$

$\lambda(y+1,x) = \mu_2(y,x,\lambda(y,x))$

Let $t = \underline{\lambda}(r,s)$ where $s = (s_1,\ldots,s_k)$ corresponding to $x = (x_1,\ldots,x_k)$. Now by Lemma 4.1 any strongly normalising term τ reduces to a unique normal form $S^n(\underline{0})$ from which we can define the value of τ to be $\mathrm{val}(\tau) = n$. We do this case by an induction argument on the value of r.

First of all observe that at the stage in a reduction from t at which (4) is applied r must have been reduced to $\underline{0}$ or so some $S(\tau)$. In the former case we are in the basis of the induction for $\mathrm{val}(r) = 0$. The next term in the sequence has leading function symbol $\underline{\mu}_1$ which preceeds $\underline{\lambda}$ and so we are done by the main induction hypothesis.

Consider $\mathrm{val}(r) = n > 0$ and assume as induction hypothesis that for all strongly normalising terms τ with $\mathrm{val}(\tau) < n$ then $\underline{\lambda}(\tau,s)$ is strongly normalising. Since $\mathrm{val}(r) \neq 0$ we know that on the first application of equation (4) in a reduction sequence from t that r has been reduced to some $S(\tau)$. And that the next element in the sequence is $\underline{\mu}_2(\tau,s',\lambda(\tau,s'))$ where $s' = (s'_1,\dots,s'_k)$ and $s_i \to s'_i$, $1 \le i \le k$. Now since s and r are strongly normalising so are s' and τ. Moreover, since $\mathrm{val}(\tau) < n$ by our latest induction hypothesis $\underline{\lambda}(\tau,s')$ is strongly normalising. Since $\underline{\mu}_2$ preceeds $\underline{\lambda}$ in Σ_0, the main induction hypothesis shows the reduct strongly normalising and the sequence to terminate.

Remember this case covers function symbols corresponding to h_i, g_i as well as those functions in Δ_i.

CASE 5. $\lambda(z,x,0) = h(z,x)$

$\lambda(z,x,y+1) = \lambda(z+1,x,g(z+1,x))$

Let $t = \underline{\lambda}(r,s,u)$ where $s = (s_1,\dots,s_k)$ corresponding to $x = (x_1,\dots,x_k)$. As before, observe that at the first stage in a reduction sequence from t at which equation (0) is applied it must have been reduced to $\underline{0}$ or to some $S(\tau)$. The first possibility does not permit an infinite continuation of the sequence because the next element is some $\underline{h}(r',s')$ where r' and s' are strongly normalised reducts of r and s and this term is strongly normalising by the induction hypothesis since $\underline{h}$ preceeds $\underline{\lambda}$ in Σ_0. Therefore only sequences of the second kind need careful consideration.

Let $\mathrm{val}(\tau)$, for τ a strongly normalising term, be just as in Case 4. Define for any term of the kind $t = \underline{\lambda}(r,s,u)$ the number

$$\chi(r,s) = (\mu z)[g(z,\mathrm{val}(s)) = 0] \mathbin{\dot{-}} \mathrm{val}(r)$$

wherein $\mathrm{val}(s)$ abbreviates $(\mathrm{val}(s_1),\dots,\mathrm{val}(s_k))$.

We do this case by a concise induction on the value $\chi(r,s)$. As basis we have t with $\chi(r,s) = 0$. Consider a reduction sequence from t in which the first application of equation (0) produces $\underline{\lambda}(S(r'),s',\underline{g}(S(r),s'))$ from $\underline{\lambda}(r',s',S(\tau))$. Since $r \to r'$, $s \to s'$ we have $\chi(r',s') = 0$ and

$$\mathrm{val}(r') \ge (\mu z)[g(z,\mathrm{val}(s)) = 0].$$

And, thanks to the main induction hypothesis, we know that all the subterms of $\underline{\lambda}(S(r'),s',\underline{g}(S(r'),s'))$ are strongly normalising. From this information we can dedcue $\mathrm{val}(\underline{g}(S(r'),s')) = 0$ so if a second application of equation

(0) is made in the sequence then we will have a sequence of the kind considered, and proved finite, at the opening of this case; whereas if no second application of (0) is made in the sequence then the reductions must be made to the known strongly normalising subterms and so it must terminate as observed in the opening of the induction argument of lemma 4.3. The calculation required is this

$$\begin{aligned} val(\underline{g}(S(r),S')) &= g(val(r')+1,val(s')) \\ &= g(\mu z)([g(z,val(s')) = 0],val(s')) \\ &= 0. \end{aligned}$$

Consider $t = \underline{\lambda}(r,s,u)$ with $\chi(r,s) = n > 0$ and assume as induction hypothesis that if r_1,s_1,u_1 are strongly normalising and $\chi(r_1,s_1) < n$ then $\underline{\lambda}(r_1,s_1,u_1)$ is strongly normalising. Consider a reduction sequence from t in which the first application of equation (0) produces $\underline{\lambda}(S(r'),s',\underline{g}(S(r'),s'))$ from $\underline{\lambda}(r',s',S(\tau))$. By our assumptions and the main induction hypothesis all subterms of the new reduct are strongly normalising. Moreover, $\chi(S(r'),s') < \chi(r',s') = \chi(r,s) = n$ and therefore by the latest induction hypothesis $\underline{\lambda}(S(r'),s',\underline{g}(S(r'),s'))$ is strongly normalising and the reduction sequence must terminate.

CASE 6. $\lambda(x) = f(x)$.

This is, by now, obvious.

Having concluded the proof of Lemma 4.3 we have also concluded the argument for Lemma 4.2. Q.E.D.

5. THE MANY SORTED CASE

(It may well go without saying, but) we assume the reader thoroughly acquainted with the technical foundations of the algebra of many-sorted structures for which no reference can better substitute for the ADJ's basic paper [3].

In notation consistent with our [1], we assume A to be a many-sorted algebra with domains $A_1,\ldots,A_{n+m}$ and operations of the form

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

where $\lambda_i, \mu \in \{1,\ldots,n+m\}$, $1 \leq i \leq k$.

The concepts and machinery of section two must be reformulated, but this is not difficult: An *algebraic replacement system* R on A consists of a collection of set-theoretic replacement systems $R_1,\ldots,R_n$ on its domains which satisfy the property that for each operation $\sigma^{\lambda,\mu}$ of A, with arguments $a_{\lambda_1},\ldots,a_{\lambda_k}$ and $b_{\lambda_1},\ldots,b_{\lambda_k}$, where $a_{\lambda_i}, b_{\lambda_i} \in A_{\lambda_i}$, if $a_{\lambda_1} \rightarrow_{R_1} b_{\lambda_1},\ldots,a_{\lambda_k} \rightarrow_{R_k} b_{\lambda_k}$ then $\sigma^{\lambda,\mu}(a_{\lambda_1},\ldots,a_{\lambda_k}) \rightarrow_{R_\mu} \sigma^{\lambda,\mu}(b_{\lambda_1},\ldots,b_{\lambda_k})$. The classification of replacement systems and the definitions of the associated congruence, one-step reductions and so on as *families* of single sorted relations proceed along the lines established for generalising algebraic ideas from single sorted to many-sorted algebras. As do their properties and the mechanisms for specifying replacement systems. Notice that if E is a set of many sorted reduction equations then each $t(X) \geq t'(X) \in E$ has t and t' of the same sort.

To lift section three to computable many-sorted algebras is also quite straight-forward and, in fact, has been virtually written out already in our [1] where Lemma 3.1 appears many-sorted, for example. Those lemmas pertaining to replacement system specifications require only the appropriate introduction of sort indices into their proofs.

Up to and including the proofs that (2) implies (3), and (3) implies (1), for the full theorem in its many-sorted case, it may be truely said that no new ideas or techniques are required.

Consider the proof that (1) implies (2). With the help of a trick (the real subject of this section) we are able to construct this proof with the toolkit of section four. Dispensing with an easy case where all the domains of A are finite, we assume A to be a many-sorted computable algebra with at least one domain infinite.

Without loss of generality we can take these domains to be $A_1,\ldots,A_n$, $B_1,\ldots,B_m$ where the A_i are infinite and the B_i are finite of cardinality $b_i + 1$. The generalised Lemma 3.1 provides us with a recursive many-sorted algebra of numbers R with domains $\Omega_1,\ldots,\Omega_n$ and $\Gamma_1,\ldots,\Gamma_m$ where $\Omega_i = \omega$ for $1 \leq i \leq m$, $\Gamma_i = \{0,1,\ldots,b_i\}$ for $1 \leq i \leq m$, and R is isomorphic to A. When not interested in the cardinality of a domain of R we refer to it as R_i, $1 \leq i \leq n+m$. The aim is to give R a finite equational hidden function replacement system specification.

The first task is to build a recursive number algebra R_0 by adding to R new constants and functions. The main idea is to code the many-sorted algebra R into its first infinite sort Ω_1 by means of functions $R_i \to \Omega_1$ and $\Omega_1 \to R_i$ and recursive tracking functions on Ω_1 associated to the multisorted operations of R. At the same time we shall dissolve the finite sorts by adding them as sets of constants. Here is the formal construction.

For each infinite sort i we add as a new constant of sort i the number $0 \in \Omega_i$ and the successor function x+1. For each finite sort i we add *all* the elements of Γ_i as *new* constants.

Each domain R_i is coded into Ω_1 by adding the function $\mathit{fold}^i(x) = x$, and is recovered by adding the function $\mathit{unfold}^i\colon \Omega_1 \to R_i$, defined for infinite sorts i by $\mathit{unfold}^i(x) = x$, and for finite sorts i by

$$\mathit{unfold}^i(x) = \begin{cases} x & \text{if } x \leq b_i \\ b_i & \text{otherwise} \end{cases}$$

Next we add for each operation $f = f^{\lambda,\mu}$ of R a recursive tracking function $\hat{f}\colon \Omega_1^k \to \Omega_1$ which commutes the following diagram:

$$\begin{tikzcd} R_{\lambda_1} \times \ldots \times R_{\lambda_k} \arrow[r, "f"] \arrow[d, "\mathit{fold}^{\lambda_1} \times \ldots \times \mathit{fold}^{\lambda_k}"'] & R_\mu \\ \Omega_1 \times \ldots \times \Omega_1 \arrow[r, "\hat{f}"] & \Omega_1 \arrow[u, "\mathit{unfold}^\mu"'] \end{tikzcd}$$

And, just as in the single sorted case, we factorise $\hat{f}$ into functions t,h,g and add these along with all the primitive recursive functions arising from the primitive recursive definitions of h and g. That is all. Observe $R_0|_\Sigma = \langle R_0 \rangle_\Sigma = R$, so it remains to give a finite, equational replacement system specification for R_0 which is Church-Rosser and strongly normalising. Let Σ_0 be the signature of R_0 in which ${}^{i}\underline{0}$, ${}^{i}S$, FOLD^i, UNFOLD^i name the zero, successor function, and coding maps associated to sort i; for convenience we drop the sort superscript on zero and successor in case i = 1. We will give the requisite set of equations E_0 beginning with the operations of R.

Let $f = f^{\lambda,\mu}$ be an operation of R named by function symbol $\underline{f} \in \Sigma \subset \Sigma_0$ and let $\hat{f}$ be its associated tracking map on Ω_1 named by $\underline{\hat{f}} \in \Sigma_0$. First, following the procedure of section four write out all the equations assigned to $\hat{f}$ and its factorisation. Secondly, add to this equation to "eliminate" $\underline{f}$

$$\underline{f}(X_{\lambda_1},\ldots,X_{\lambda_k}) \geq \mathit{UNFOLD}^{\mu}(\underline{\hat{f}}(\mathit{FOLD}^{\lambda_1}(X_{\lambda_1}),\ldots,\mathit{FOLD}^{\lambda_k}(X_{\lambda_k})))$$

where X_{λ_i} is a variable of sort λ_i. Do this for every operation of R.

Turning to the coding machinery, consider first the folding functions. For each infinite sort i add the equations,

$$\begin{array}{lll} \mathit{FOLD}^{i}({}^{i}\underline{0}) & \geq & \underline{0} \\ \mathit{FOLD}^{i}({}^{i}S(X_i)) & \geq & S(\mathit{FOLD}^{i}(X_i)) \end{array}$$

where X_i is a variable of sort i.

For each finite sort i, if ${}^{i}\underline{\underline{c}} \in \Sigma_0 - \Sigma$ is a new constant of sort i denoting number $c \in \Gamma_i$ then add

$$\mathit{FOLD}^{i}({}^{i}\underline{\underline{c}}) \geq S^{c}(\underline{0})$$

Secondly consider the unfolding functions. For each infinite sort i add the equations,

$$\begin{array}{lll} \mathit{UNFOLD}^{i}(\underline{0}) & \geq & {}^{i}\underline{0} \\ \mathit{UNFOLD}^{i}(S(X)) & \geq & {}^{i}S(\mathit{UNFOLD}^{i}(X)) \end{array}$$

where X is a variable of sort 1.

For each finite sort i, if ${}^{i}\underline{\underline{c}}$ is as before then add the equations

$$\begin{array}{lll} \mathit{UNFOLD}^{i}(S^{c}(\underline{0})) \geq {}^{i}\underline{\underline{c}} & \qquad & \text{if } c < b_i \\ \mathit{UNFOLD}^{i}(S^{c}(X)) \geq \underline{\underline{b}}_i & \qquad & \text{if } c \geq b_i \end{array}$$

where b_i is the last element of Γ_i and is named in $\Sigma_0 - \Sigma$ by $\underline{\underline{b}}_i$; and X is a variable of sort 1.

And finally we consider the equations for the constants. For each infinite sort i, if ${}^{i}\underline{c} \in \Sigma$ denotes the number $c \in \Omega_i$ then add

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

For each finite sort, i, if ${}^{i}\underline{c} \in \Sigma$ denotes the number $c \in \Gamma_i$ and ${}^{i}\underline{\underline{c}} \in \Sigma_0 - \Sigma$ is its new constant symbol then we remove the duplication by adding ${}^{i}\underline{c} \geq {}^{i}\underline{\underline{c}}$.

This completes the construction of E_0.

What remains of the proof follows closely the arguments of section four. Here the sets of normal forms are, of course, $\{{}^{i}S^{c}({}^{i}\underline{0}) : c\in\omega\}$ when i is an infinite sort, and $\{{}^{i}\underline{\underline{c}} : c\in\Gamma_i\}$ when i is a finite sort. And the arguments which lift Lemmas 4.1 and 4.2 are *in all essential respects the same.* (So, for instance, to lift Lemma 4.3 one reads the ordering of Σ_0 off the replacement equations in E_0 and finds the bulk of one's proof a fair copy of Lemma 4.3 modulo a sort index.) Given, then, that (Σ_0, E_0) is Church-Rosser and strongly normalising, the normal forms being a traversal for $\equiv_{E_0}$, we can prove $R_0 \cong T(\Sigma_0, E_0)$ by using the mappings ϕ^i defined $\phi^i(c) = [{}^{i}S^{c}({}^{i}\underline{0})]$ for i an infinite sort and $\phi^i(c) = {}^{i}\underline{\underline{c}}$ for i a finite sort.

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 115/79, Amsterdam, 1979.

[2] ————, *On the adequacy of finite equational methods for data type specification,* SIGPLAN Notices, to appear.

[3] 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.

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

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

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

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