Equational Specifications, Complete Term Rewriting Systems, and Computable and Semicomputable Algebras

J. A. BERGSTRA

University of Amsterdam, Amsterdam, and University of Utrecht, Utrecht, The Netherlands

AND

J. V. TUCKER

University of Wales Swansea, Swansea, Wales

Abstract. We classify the computable and semicomputable algebras in terms of finite equational initial algebra specifications and their properties as term term rewriting systems, such as completeness. Further results on properties of these specifications, such as on their size and orthogonality, are provided which show that our main results are the best possible.

Categories and Subject Descriptors: D.3.3 [**Programming Languages**]: Language Constructs and Features; F.1.1 [**Computation by Abstract Devices**]: Models of Computation; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages; F.3.3 [**Logics and Meanings of Programs**]: Studies of Programming Constructs; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic

General Terms: Languages, Theory

Additional Key Words and Phrases: Abstract data types, complete term rewriting systems, computable and semicomputable algebras, equational specifications with hidden functions, many sorted algebras, term rewriting systems

1. *Introduction*

In algebraic data type theory, data types are modelled semantically by many sorted universal algebras, and are specified by means of equational or conditional equational axioms, most commonly up to isomorphism using initial algebra semantics for the specification. Algebras of particular interest are the

Authors' addresses: J. A. Bergstra, Programming Research Group, University of Amsterdam, Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands and Department of Philosophy, University of Utrecht, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands; J. V. Tucker, Department of Computer Science, University of Wales Swansea, Singleton Park, Swansea SA2 8PP Wales.

computable and semicomputable algebras. The algebras that can be algebraically specified using initial algebra semantics are precisely the semicomputable algebras [Bergstra and Tucker 1987].

We examine the term rewriting properties of finite equational hidden function specifications of data types and we give an algebraic characterization of the computable data types in terms of complete term rewriting systems. A complete term rewriting system is one whose reductions or rewrites satisfy the Church–Rosser property and are strongly terminating. The first theorem we prove is this:

THEOREM 1.1. *Let A be a finitely generated minimal Σ algebra. Then, the following are equivalent*:

(1) *A is computable.*
(2) *There is a finite equational specification (Σ_0, E_0) such that*
  (i) $Sort(\Sigma) = Sort(\Sigma_0)$ and $\Sigma \subseteq \Sigma_0$;
  (ii) (Σ_0, E_0) is a complete term rewriting system;
  (iii) $I(\Sigma_0, E_0)|_\Sigma \cong A$.

Furthermore, the (Σ_0, E_0) may be taken to be an orthogonal term rewriting system, and the sizes $|\Sigma_0|$ and $|E_0|$ depend upon the algebra A.

The fact that (2) implies (1) is straightforward and is a principal reason for the usefulness of complete term rewriting systems. The fact that (1) implies (2) is more difficult.

The specific properties of the equational specification (Σ_0, E_0) constructed in the theorem suggest a number of questions. Of special interest is the fact that the size of the specification (Σ_0, E_0) is large and depends on the algebra A. In our previous studies, we showed that any computable algebra could be given a very small equational specification: for example, in Bergstra and Tucker [1983b], we constructed a finite equational specification (Σ_1, E_1) in which the number of hidden functions needed is $3n$ and the number of equations is $2n$, where n is the number of sorts in the signature Σ of the algebra A. *In particular, we note that the size of (Σ_1, E_1) is independent of A and, indeed, independent of the number of constants and operations in the signature Σ of A.* Can any computable algebra be given such a small finite equational specification that is also a complete term rewriting system? We show that there is a finite data type such that any equational specification which is a complete term rewriting system must be dependent on the algebra and must be large (Theorem 6.1.1). Thus, the specification in the main theorem above cannot be improved in this regard.

To further complement the theorem and to provide a comprehensive picture of its scope we examine the term rewriting properties of computable and semicomputable algebras.

We show that a computable algebra can have an *infinite* recursive equational specification *without hidden functions* that is an orthogonal complete term rewriting system (Theorem 4.3.1). From this construction, we obtain a finite equational specification without hidden functions for a finite algebra that is an orthogonal complete term rewriting system (Corollary 4.3.2). We construct an algebra of sets to show that the hidden functions in an equational specification that is a complete term rewriting system can be essential for the construction of normal forms (Theorem 6.2.1).

Finally, we consider semicomputable algebras and, in contrast to the theorem for computable algebras, show the following characterization (Theorems 7.1.1, 7.1.2, and 7.2.1):

THEOREM 1.2. *Let A be a finitely generated minimal Σ algebra. Then, the following are equivalent*:

(1) *A is semicomputable.*
(2) *There is an infinite recursively enumerable equational specification (Σ, E) such that*
   (i) (Σ, E) is a complete term rewriting system;
   (ii) $I(\Sigma, E) \cong A$.

However, the recursively enumerable complete term rewriting specification (Σ, E) cannot be replaced by a recursive complete term rewriting specification, nor can it be replaced by a recursively enumerable orthogonal complete term rewriting specification.

The basic concepts on equational specifications and term rewriting systems are carefully defined in Section 3, and on computable and semicomputable algebras in Section 4. In Sections 5–7, we prove the theorems: results about computable algebras are in 5 and 6, and results about semicomputable algebras are in Section 7. Section 2 explains briefly some theoretical issues concerning data types which the theorems address.

This paper belongs to our series of studies on the adequacy and power of algebraic specification methods for data types which we began in Bergstra and Tucker [1979] (see also Bergstra and Tucker [1987]), which is summarised in Section 2.

A single sorted form of Theorem 1.1 was first announced in Bergstra and Tucker [1980b] and its proof outlined. Here, the theorem is reformulated using current term rewriting theory and a full proof in the many sorted case is given. The other theorems are new. For applications of the theorems, it is the many sorted case that is important. We have found that it is not satisfactory to transform the proof for single sorted algebras to a proof for many sorted algebras, because an informal account of the extension is unable to get at details we consider to be important.

As in our previous studies, we exploit proof methods based on recursion schemes on the natural numbers. Algebraic aspects of the μ-recursive functions and the primitive recursive Kleene T-predicate are used to provide arguments that are algebraically detailed and natural. Related to the proof of Theorem 1.1 are methods for simulating the behavior of Turing machines using rewrite systems (mentioned in, e.g., Dershowitz [1987]). In particular, Dauchet [1989] indicates that a Turing machine computation can be encoded in a single orthogonal rewrite rule that gives rise to a terminating TRS. These Turing machine methods offer an approach different from the recursion scheme methods we have used here and in the past. In either case, the main technical problem in the paper is to obtain a specification of the target algebra without the use of hidden sorts.

The reader is assumed well versed in initial algebra specification methods: see Goguen et al. [1978], Ehrig and Mahr [1985], and Wirsing [1990]; knowledge of our earlier work is desirable but not strictly necessary. Knowledge of the theory of the recursive functions is necessary: see Cutland [1980], for instance.

Finally, we thank the referees for their many valuable comments that have helped us improve the paper.

2. *Initial Algebra Semantics and Data Types*

We will summarize the context for the theorems in the theory of data types.

2.1. ALGEBRAIC SEMANTICS. A concrete implementation of a data type is modelled by a many sorted algebra A finitely generated from initial values $a_1, \ldots, a_n \in A$ named in its finite signature Σ, that is, A is a finitely generated minimal algebra. A data type is modeled by some class K of minimal algebras of common signature. Two concrete data types are equivalent if they are isomorphic as algebras. An abstract data type can be modeled as a class K of minimal algebras closed under isomorphism, that is, if $A \in K$ and $B \cong A$, then $B \in K$. Often, an abstract data type is modeled as an isomorphism type, that is, a class closed under isomorphism in which *all* algebras are isomorphic. Further semantic considerations involve the computability of the many sorted algebras used: A may be a finite, computable, semicomputable or, perhaps, cosemicomputable algebra. These notions are based on theoretical ideas about the implementation of algebras on computers and, to be semantic properties, they must also be isomorphism invariants.

The semantics K which model a data type can be investigated using the construction of an initial algebra $I(K)$ for K with the result that every $A \in K$ is uniquely definable, up to isomorphism, as a unique epimorphic image of $I(K)$ or, equivalently, as a factor algebra $I(K)/\equiv_A$ for a unique congruence $\equiv_A$. If $I(K) \in K$, then this $I(K)$ can be used as a canonical algebra modelling the data type; if, in addition, the class K is an isomorphism type then $I(K)$ "is" the data type. The idea is that $I(K)$ models the data type semantics in a way that is independent of methods for its implementation or specification.

2.2. IMPLEMENTATION USING TERMS. The basic method for constructing this initial algebra $I(K)$, up to isomorphism, is as a factor algebra of the syntactic algebra $T(\Sigma)$ of all closed terms over Σ, because $T(\Sigma)$ is initial for the class $ALG(\Sigma)$ of all Σ algebras. In fact, $I(K) \cong T(\Sigma)/\equiv_K$, where $\equiv_K$ is a congruence for which $t \equiv_K t'$ means that the terms t and t' are equivalent syntactic expressions in all the algebras of K, that is, $K \models t = t'$. The problem of implementing and specifying the data type K via $I(K)$ can be investigated through the corresponding problem of implementing or specifying the congruence $\equiv_K$. A central technical idea is that of a transversal for the equivalence relation $\equiv_K$, which is a set of terms containing one and only one term for each equivalence class.

This algebra $T(\Sigma)|\equiv_K$ is of importance for implementing the data type, using symbolic computation techniques for representing terms and algorithms for term manipulation, especially algorithms for computing $\equiv_K$. In these circumstances, we may say that a data type K is *computable*, or *semicomputable*, when $\equiv_K$ is a decidable or semidecidable relation on $T(\Sigma)$.

2.3. ALGEBRAIC SPECIFICATIONS. To specify the data type, an axiomatic theory (Σ, E) is used so that $K \subseteq ALG(\Sigma, E)$. In particular, if E is a set of equations or conditional equations and $K = ALG(\Sigma, E)$, then $I(K) \in K$, and

$$E \vdash t = t' \Leftrightarrow t \equiv_K t',$$

where the proof system is that for equational or conditional equational logic applied to closed terms. Thus, for any recursively enumerable E we know that $\equiv_K$ is recursively enumerable. However, the specification (Σ, E) is desired to be finite.

We know from Bergstra and Tucker [1987] that this finite equational or conditional equational method will not define all computable (and hence all semicomputable) data types. Enriching the method to allow the use of *a finite number of hidden sorts and functions* does enable it to specify *any* semicomputable (and hence any computable) data type. The question addressed here is:

What finite algebraic specification methods exist that specify all and only the computable data types?

As noted in the Introduction, one answer to this question in Bergstra and Tucker [1983b] is that an algebra A is computable if, and only if, it can be given an equational specification (Σ, E), using a small number of hidden functions and equations, such that A is isomorphic with the initial and final algebras of (Σ, E). In Bergstra and Tucker [1983a] the cosemicomputable algebras where shown to be precisely the algebras defined by algebraic specifications under final algebra semantics; see also Bergstra and Tucker [1980] and Meseguer et al. [1992].

Here the properties of the specification (Σ, E) that determine the decidability of the congruences for initial algebras are analyzed. The equations E may be interpreted as left-to-right rewrite rules for transforming terms of $T(\Sigma)$. Thus, (Σ, E) serves both as an axiomatic specification for a variety of models, and as a *term rewriting system*, intended to formalize a system of deductions, governed by simple algebraic substitution rules, within which a deduction

$$t \to t' \qquad \text{implies} \qquad t \equiv_K t'$$

(but not conversely). The choice of (Σ, E) leads to a transversal or set J of normal forms for $\equiv_K$: given $t, t' \in T(\Sigma)$, to decide $t \equiv_E t'$ one uses E to calculate their prescribed "normal forms" $n, n' \in J$ and on completing the deductions $t \to_E n$, $t' \to_E n'$ one checks $n = n'$.

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, however canonical an implementation. The decidability of $\equiv_K$ implies the computability of the algebra $T(\Sigma)/\equiv_K$ under our definition and, in particular, since the concept of a computable algebra is an isomorphism invariant, we can remove 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 Theorem 1.1, the ease with which statement (2) implies (1) is proved is evidence for the usefulness of complete term rewriting system specifications, whereas the implication that (1) implies (2) is considered as the affirmative answer to the question about adequacy: *Do these finite complete term rewriting system equational specification methods define all the data types one wants*?

The examples and Theorem 1.2 differentiate the computable and semicomputable algebras using related properties of term rewriting systems.

3. *Reduction Systems, Term Rewriting, and Equational Specifications*

We describe many sorted algebraic reduction systems that are meant to abstract the essential structure of term rewriting systems. Then we describe the concepts and notations concerning many sorted term rewriting systems and algebraic specifications that we will need. Supporting references are: Meinke and Tucker [1992] and Wechler [1992], for basic algebra; Ehrig and Mahr [1985] and Wirsing [1990], for algebraic specifications; and Klop [1992], for single sorted term rewriting.

3.1. SIGNATURES AND ALGEBRAS. The notations for signatures and algebras are as follows:

3.1.1. *Signatures.* A *signature* Σ consists of a nonempty set S, whose elements are called *sorts*, and a family

$$\langle \Sigma_{w,s} : w \in S^*, s \in S \rangle$$

of sets, where the elements of $\Sigma_{\lambda,s}$ for λ the empty string, are called *constant names* or *symbols* of sort s; and the elements of $\Sigma_{w,s}$ are called *function names* or *symbols* of type $w \to s$.

A signature Σ is *finite* if S is finite, each $\Sigma_{w,s}$ is finite, and all but finitely many $\Sigma_{w,s} = \emptyset$. *We assume all signatures are finite.*

Let Σ^1 and Σ^2 be signatures. Then Σ^1 is a *subsignature* of Σ^2 if $S^1 \subseteq S^2$ and for each $w \in S^{1*}$ and $s \in S^1$

$$\Sigma^1_{w,s} \subset \Sigma^2_{w,s}.$$

We write $\Sigma^1 \subseteq \Sigma^2$.

3.1.2. *Algebras.* Let Σ be a signature. A Σ algebra A consists of a family

$$\langle A_s : s \in S \rangle$$

of nonempty sets A_s, called the *carriers* of A, together with families of elements

$$\Sigma^A_{\lambda,s} = \langle c_A \in A_s : c \in \Sigma_{\lambda,s} \rangle$$

and families of functions

$$\Sigma^A_{w,s} = \langle \sigma_A : A_{w(1)} \times \cdots \times A_{w(k)} \to A_s : \sigma \in \Sigma_{w,s} \rangle$$

A Σ algebra is *minimal* if it contains no proper subalgebras.

If A is Σ algebra and $\Sigma_0 \subseteq \Sigma$, then $A|_{\Sigma_0}$ is the algebra obtained from A by deleting the constants and operations of A not named in Σ_0 and $\langle A \rangle_{\Sigma_0}$ is the smallest Σ_0 algebra contained in A.

3.2. REDUCTION SYSTEMS AND ALGEBRAIC REDUCTION SYSTEMS. We begin with a sequence of basic definitions.

Let $A = \langle A_s : s \in S \rangle$ be an S sorted family of nonempty sets, and $\equiv = \langle \equiv_s : s \in S \rangle$ be an S sorted family of equivalence relations on A. A *transversal* for $\equiv$ is a set $J = \langle J_s : s \in S \rangle$ where for each $s \in S$, $J_s \subseteq A_s$ and for each $a \in A_s$ there is one and only one $t \in J_s$ such that $t \equiv_s a$.

Let $A = \langle A_s : s \in S \rangle$ be a S sorted family of nonempty sets. A *reduction system* or a *replacement system* on the many sorted set A is an S sorted family

$$\to_R = \langle \to_{R_s} : s \in S \rangle$$

of reflexive and transitive binary relations on A. For $a, b \in A_s$, if $a \to_{R_s} b$ we say *a reduces* to b (under $\to_R$), or that b is a *reduct* of a (under $\to_R$).

An element $a \in A_s$ is a *normal form* for $\to_R$ if there is no $b \in A_s$ so that $a \to_{R_s} b$; the set of all normal forms for $\to_R$ is denoted

$$NF(\to_R) = \langle NF_s(\to_R) : s \in S \rangle.$$

The reduction system $\to_R$ is *confluent* or *Church–Rosser* if for each $s \in S$ and for any $a \in A_s$ if there are $b_1, b_2 \in A_s$ so that $a \to_{R_s} b_1$ and $a \to_{R_s} b_2$, then there is $c \in A_s$ so that $b_1 \to_{R_s} c$ and $b_2 \to_{R_s} c$.

The reduction system $\to_R$ is *weakly terminating* or *weakly normalizing* if, for each $s \in S$ and for each $a \in A_s$, there is some normal form $b \in A_s$ so that $a \to_{R_s} b$.

The reduction system $\to_R$ is *strongly terminating* or *strongly normalizing* if, for each $s \in S$, there does not exist an infinite chain

$$a_0 \to_{R_s} a_1 \to_{R_s} \cdots \to_{R_s} a_n \to_{R_s} \cdots$$

of reductions in A_s wherein for $i \in \mathbf{N}$, $a_i \neq a_{i+1}$.

The reduction system $\to_R$ is *complete* if it is Church–Rosser and strongly terminating.

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

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

$$a \equiv_{R_s} a' \Leftrightarrow \text{there is a sequence } a = b_1, \ldots, b_k = a' \text{ such that for each pair } b_i, b_{i+1} \in A_s \text{ there exists a common reduct } c_i \in A_s \text{ for } 1 \leq i \leq k-1.$$

Using this characterization of $\equiv_R$, it is straightforward to prove this fact:

LEMMA 3.2.1. *The reduction system $\to_R$ on A is Church–Rosser if, and only if, for each $s \in S$ and for any $a, a' \in A_s$ if $a \equiv_{R_s} a'$, then there is $c \in A_s$ so that $a \to_{R_s} c$ and $a' \to_{R_s} c$.*

LEMMA 3.2.2. *Let $\to_R$ be a Church–Rosser weakly terminating reduction system on A. Then the set $NF(\to_R)$ of normal forms is a transversal for $\equiv_R$.*

PROOF. Since every element $a \in A_s$ reduces to some normal form $n \in NF_s(\to_R)$, the set $NF_s(\to_R)$ contains representatives for each equivalence class of $\equiv_{R_s}$. To check uniqueness, let $n, m \in NF_s(\to_R)$ and assume $n \equiv_{R_s} m$. By Lemma 3.2.1, there is $c \in A_s$ so that $n \to_{R_s} c$ and $m \to_{R_s} c$, but since n, m are normal forms $n = c, m = c$ and so $n = m$. Q.E.D.

Definition 3.2.3. Suppose now that A is a many sorted algebra. Then by an *algebraic reduction system* $\to_R$ on the algebra A we mean a reduction system $\to_R$ on the family of carriers of A, which is closed under the operations of A in the sense that for each operation

$$\sigma : A_{w(1)} \times \cdots \times A_{w(k)} \to A_s$$

of A, for all $a_i, b_i \in A_{w(i)}$ and $1 \leq i \leq k$,

$$a_1 \to_{R_{w(1)}} b_1, \ldots, a_k \to_{R_{w(k)}} b_k \Rightarrow \sigma(a_1, \ldots, a_k) \to_{R_s} \sigma(b_1, \ldots, b_k).$$

LEMMA 3.2.4. *If $\to_R$ is an algebraic reduction system on an algebra A, then $\equiv_R$ is a congruence on A.*

We next explain how a reduction system is generated by a set of one-step reductions and how these sets of one-step reductions can be determined from quite arbitrary sets.

Let $\to_R$ be a reduction system on an S sorted set $A = \langle A_s : s \in S \rangle$. Let X be an S sorted subset of $A \times A$, that is, $X = \langle X_s : s \in S \rangle$ where $X_s \subseteq A_s \times A_s$. Then X is said to *generate* $\to_R$ *as a set of one-step reductions* if X is reflexive and $\to_R$ is the smallest transitive set containing X, that is, the *transitive closure* of X.

Let $\to_R$ be an algebraic reduction system on an S sorted algebra A. Then $X \subseteq A \times A$ is said to *generate* $\to_R$ *as a set of algebraic one-step reductions* if X is reflexive; X is closed under *unit substitutions* in the following sense: writing $(a, b) \in X_{w(i)}$ as $a \to_{X_{w(i)}} b$, for any operation

$$\sigma : A_{w(1)} \times \cdots \times A_{w(k)} \to A_s$$

of A, for any $1 \leq i, j \leq k$ and $j \neq i$, $a_j \in A_{w(j)}$, and $a \to_{X_{w(i)}} b$, it follows that

$$\sigma(a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_k) \to_{R_s} \sigma(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_k);$$

and $\to_R$ is the transitive closure of X.

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

3.3. TERM ALGEBRAS. We will apply these ideas to specify algebraic reduction systems on the term algebra $T(\Sigma)$ and connect them with the initial algebras of equational theories. First, we give definitions of the primary notions of terms, equations, term evaluation etc.

3.3.1. *Terms.* Let X be a set of variable names. For $s \in S$ and $x \in X$, we call x^s an *s sorted variable*. Let X_s be the set of all s sorted variables.

Let $V = \langle V_s : s \in S \rangle$ where $V_s \subset X_s$. We define the Σ term algebra $T(\Sigma, V)$ of terms over Σ in variables of V as follows.

The family of carriers of $T(\Sigma, V)$ is

$$\langle T_s(\Sigma, V) : s \in S \rangle$$

where the $T_s(\Sigma, V)$ are inductively defined simultaneously over S:

(i) for $x^s \in V_s$, $x^s \in T_s(\Sigma, V)$;
(ii) for $c \in \Sigma_{\lambda, s}$, $c^s \in T_s(\Sigma, V)$; and
(iii) for $\sigma \in \Sigma_{w, s}$, where $w = w(1) \cdots w(k)$, and $t_i \in T_{w(i)}(\Sigma, V)$;
$\sigma^s(t_1, \ldots, t_k) \in T_s(\Sigma, V)$.

Notice that the type can be determined from each term.

The constants of $T(\Sigma, V)$ are defined for $c \in \Sigma_{\lambda, s}$,

$$c_{T(\Sigma,V)} = c^s$$

and the operations are defined for $\sigma \in \Sigma_{w,s}$,

$$\sigma_{T(\Sigma,V)}(t_1, \ldots, t_k) = \sigma^s(t_1, \ldots, t_k)$$

for $t_i \in T_{w(i)}(\Sigma, V)$.

This makes $T(\Sigma, V)$ a Σ-algebra.

We write $T(\Sigma, X)$ for $V_s = X_s$, $s \in S$, the complete family of sorted variables named by X. $T(\Sigma, V)$ is a subalgebra of $T(\Sigma, X)$. We write $T(\Sigma)$ when $V_s = \varnothing$, $s \in S$. The terms of $T(\Sigma)$ are called *closed*.

A signature Σ is (fully) *instantiated* or *nonvoid* if for all $s \in S$

$$T_s(\Sigma) \neq \varnothing.$$

Definition 3.3.1.1. A *term rewriting system* or *TRS* is any algebraic reduction system on a term algebra.

3.3.2. *Equations.* Let Σ be a signature and X a set of variable names. An equation of sort $s \in S$ is an expression e of the form

$$t(X) = t'(X),$$

where $t(X), t'(X) \in T_s(\Sigma, X)$. Let $Eqn_s(\Sigma, X)$ be the set of such equations and

$$Eqn(\Sigma, X) = \langle Eqn_s(\Sigma, X) | s \in S \rangle.$$

Let $E = \langle E_s | s \in S \rangle$ be a set of equations with $E_s \subset Eqn_s(\Sigma, X)$. Then E is finite if S is finite and E_s is finite for each $s \in S$.

Let $e, e' \in T_s(\Sigma, X)$. We say e and e' are *α-equivalent* if there is a permutation of the set X of variable names that transforms e to e'. We write $e \equiv_\alpha e'$ if e and e' are equivalent.

Let $[e]_\alpha = \{e' \in Eqn_s(\Sigma, X) | e' \equiv_\alpha e\}$.

Let $E = \langle E_s | s \in S \rangle$ be a set of equations. Define the *α-closure* $\acute{E}$ of E to be

$$\acute{E} = \langle \acute{E}_s | s \in S \rangle,$$

where

$$\acute{E}_s = \{e' \in Eqn_s(\Sigma, X) | e' \equiv_\alpha e \text{ for some } e \in E_s\} = \bigcup_{e \in E_s} [e]_\alpha,$$

We say that E is *α-closed* if $E = \acute{E}$.

3.3.3. *Term Evaluation.* We define the term evaluation Σ homomorphism $val_A : T(\Sigma) \to A$. The map $val_A = \langle val_A^s : s \in S \rangle$ is defined by induction on terms simultaneously over S.

For $c \in \Sigma_{\lambda, s}$,

$$val_A^s(c^s) = c_A^s.$$

For $\sigma \in \Sigma_{w,s}$,

$$val_A^s(\sigma^s(t_1, \ldots, t_k)) = \sigma_A^s\big(val_{w(1)}(t_1), \ldots, val_{w(k)}(t_k)\big).$$

Then val_A is clearly a homomorphism (by definition of constants and operations on $T(\Sigma)$). We often write val for val_A when the A is understood.

LEMMA 3.3.3.1. *The map* $val_A: T(\Sigma) \to A$ *is a* Σ *homomorphism and is unique as such.*

The congruence derived from val_A we write

$$\equiv_A = \langle \equiv_A^s \mid s \in S \rangle$$

and $\equiv_A^s$ on A_s is defined by

$$t \equiv_A^s t' \quad \text{iff} \quad val_A^s(t) = val_A^s(t').$$

Thus, if val_A is an epimorphism,

$$T(\Sigma)/\equiv_A \cong A$$

Note A is Σ minimal if val_A is a surjection.

Consider $\equiv_A$ on $T(\Sigma)$.

A *term transversal* for A is a family $T = \langle T_A^s : s \in S \rangle$ such that for each $s \in S$, $T_A^s \subseteq T_s(\Sigma)$ and for each $a \in A_s$ there is exactly one $t \in T_A^s$ such that $val_A^s(t) = a$.

A *canonical term transversal* for A is a term transversal T such that $\sigma^s(t_1, \ldots, t_k) \in T_A^s$ for each subterm $t_i \in T_A^{w(i)}$.

A standard construction of a canonical term transversal T is to order $T_s(\Sigma)$ lexically and for each $a \in A$ choose the first (and so shortest) $t \in T_s(\Sigma)$ such that $val_A^s(t) = a$. To see this is canonical, note that if $t = \sigma^s(t_1, \ldots, t_k)$ and t_i is not a shortest term for $a \in A_{w(i)}$ then we can replace t_i with a shorter term that would contradict that t is in the transversal T.

3.4. TERM REWRITING SYSTEMS. Let $T(\Sigma, X)$ be the algebra of terms over Σ in variable names X. Let $E \subseteq Eqn(\Sigma, X)$ be a set of equations such that for each $t = t' \in E$ the LHS t is not a variable and all the variables in the RHS t' also appear in t. We can define a set $D(E) \subseteq T(\Sigma) \times T(\Sigma)$ by

$$D(E)_s = \{(t(r_1, \ldots, r_k), t'(r_1, \ldots, r_k)) : t = t' \in E_s \text{ and } r_i \in T(\Sigma)_{w(i)}\}$$

and so obtain the smallest set $\to_{E(1)}$ of algebraic one-step reductions containing $D(E)$, and the algebraic reduction relation $\to_E$ it generates. This formalizes the use of equations in derivations of terms in $T(\Sigma)$ where the reduction $t \to_E t'$ requires substitutions to be made in some equation $e \in E$ and the LHS of e is replaced by the RHS of e in t to obtain t'.

Definition 3.4.1. We call the pair (Σ, E) an *equational term rewriting system* or *equational TRS*, for short.

The first set of properties of a term rewriting system (Σ, E) is now defined by applying the properties of reduction systems to $\to_E$; for example:

The term rewriting system (Σ, E) is *complete* if the reduction system $\to_E$ on $T(\Sigma)$ is Church–Rosser and strongly terminating.

We denote by $NF(\Sigma, E)$ the set of all normal forms of $\to_E$ and by $\equiv_E$ the congruence associated to $\to_E$.

LEMMA 3.4.2. $T(\Sigma, E) = T(\Sigma)/\equiv_E$ *is the initial algebra of* $ALG(\Sigma, E)$.

LEMMA 3.4.3. *Let A be a minimal Σ algebra. Let (Σ, E) be a weakly terminating TRS. If $A \models E$ and $NF(\Sigma, E)$ is a transversal for $\equiv_A$, then*

$$T(\Sigma, E) = T(\Sigma)/\equiv_A \cong A.$$

PROOF. We shall show that for $t, t' \in T(\Sigma)$,

$$t \equiv_E t' \Leftrightarrow t \equiv_A t'.$$

Case (i). Suppose $t \equiv_E t'$. Then, by Birkhoff's Completeness Theorem, $E \vdash t = t'$ and since $A \in ALG(\Sigma, E)$ we have $E \models t = t'$, by soundness. This means that $val_A(t) = val_A(t')$ in A, that is, $t \equiv_A t'$.

Case (ii). Suppose $t \equiv_A t'$. Since the TRS (Σ, E) is weakly terminating we can calculate its normal forms for all terms using a function nf_E. We know that

$$t \equiv_E nf_E(t) \quad \text{and} \quad t' \equiv_E nf_E(t'). \tag{*}$$

By the result of case (i), using $A \models E$,

$$t \equiv_A nf_E(t) \quad \text{and} \quad t' \equiv_A nf_E(t').$$

Since $t \equiv_A t'$,

$$nf_E(t) \equiv_A nf_E(t'),$$

but since $NF(\Sigma, E)$ is a transversal for $\equiv_A$, this implies

$$nf_E(t) \equiv nf_E(t').$$

Thus,

$$nf_E(t) \equiv_E nf_E(t')$$

and by (*)

$$t \equiv_E t'. \qquad \text{Q.E.D.}$$

Definition 3.4.4. The term rewriting system (Σ, E) is *left linear* if for all $t = t' \in E$, each variable that appears in t does so only once.

The term rewriting system (Σ, E) is *nonoverlapping* if

(i) for any pair of different equations $t = t'$, $r = r' \in E$, the terms t and r do not overlap in the following sense: there exist closed substitutions τ, ρ of t,r such that $\rho(r)$ is a subterm of $\tau(t)$ and the outermost function symbol of $\rho(r)$ occurs as a part of t.

(ii) for any rule $t = t' \in E$, t does not overlap with itself in the following sense: there exist closed substitutions τ, ρ of t such that $\tau(t)$ is a proper subterm of $\rho(t)$ and the outermost function symbol of $\tau(t)$ occurs as a part of t.

The term rewriting system (Σ, E) is *orthogonal* if it is left linear and nonoverlapping.

The following is a basic fact about orthogonality; for a proof, see Section 3 of Klop [1992, Theorem 3.1.2].

LEMMA 3.4.5. *If (Σ, E) is an orthogonal TRS, then it is Church–Rosser.*

3.5. ALGEBRAIC SPECIFICATIONS. Let A be an algebra of signature Σ_A. Then A is said to have a *finite equational specification* (Σ, E) if $\Sigma = \Sigma_A$ and E is a finite set of equations over $T(\Sigma, X)$ such that $T(\Sigma, E) \cong A$.

An algebra A of signature Σ_A is said to have a *finite equational hidden enrichment specification* (Σ, E) if $\Sigma_A \subseteq \Sigma$ and E is a finite set of equations over $T(\Sigma, X)$ such that

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

The structural properties of a specification (Σ, E), such as the Church–Rosser and normalization properties, are taken from those of its reduction relation $\rightarrow_E$. For example:

Definition 3.5.1. An algebra A of signature Σ_A is said to have a *finite equational complete TRS hidden enrichment specification* (Σ, E) if $\Sigma_A \subseteq \Sigma$ and E is a finite set of equations over $T(\Sigma, X)$ such that $\rightarrow_E$ is a complete TRS and

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

4. *Computable and Semicomputable Algebra*

The definitions of a computable and semicomputable algebra have their origins in Rabin [1960] and Mal'cev [1961/1971], independent papers devoted to founding a general theory of computable algebraic systems and their computable morphisms. We will mention only those ideas and facts that contribute to the understanding or proofs of the theorems. In fact, the definitions and associated basic results (e.g., about invariance and the word problem) that are standard in computable algebra, and that we need here, can be found in Mal'cev [1961/1971].

For further information about the theory from the point of view of logic see the articles Mal'cev [1961/1971] and Ershov [1977]. For a view from computer science, where the subject is treated in the many-sorted case, see Meseguer and Goguen [1985] and Bergstra and Tucker [1987]. A survey of computable algebra, which includes details of its historical development, is Stoltenberg-Hansen and Tucker [1995].

4.1. ALGEBRAS AND NUMBERINGS. Let Σ be an S sorted signature. A many sorted algebra A of signature Σ is said to be *effective* if for each $s \in S$ there exists a recursive set Ω_s of natural numbers and a surjection $\alpha_s: \Omega_s \rightarrow A$ such that for each operation symbol $\sigma \in \Sigma_{w,s}$ for $w = w(1) \cdots w(k)$, and corresponding operation σ_A of A, there corresponds a recursive *tracking function* $\bar{\sigma}: \Omega_{w(1)} \times \cdots \times \Omega_{w(k)} \rightarrow \Omega_s$ which commutes the following diagram,

$$\begin{array}{ccc} A_{w(1)} \times \ldots \times A_{w(k)} & \xrightarrow{\sigma_A} & A_s \\ \uparrow{\scriptstyle \alpha^w} & & \uparrow{\scriptstyle \alpha} \\ \Omega_{w(1)} \times \ldots \times \Omega_{w(k)} & \xrightarrow{\bar{\sigma}} & \Omega_s \end{array}$$

wherein $\alpha^w(x_1, \ldots, x_k) = (\alpha_{w(1)}(x_1), \ldots, \alpha_{w(k)}(x_k))$.

We combine the sets into the S sorted family

$$\Omega = \langle \Omega_s : s \in S \rangle$$

which, along with the tracking functions, constitutes a recursive Σ algebra R of numbers. We combine the surjections into the S sorted map

$$\alpha = \langle \alpha_s : s \in S \rangle$$

which is a Σ epimorphism $\alpha: R \to A$. We refer to α as an *effective numbering* or *coordinate system*.

Consider the S sorted relation $\equiv_\alpha = \langle \equiv_{\alpha_s} : s \in S \rangle$ on the number algebra R, defined for $x, y \in \Omega_s$ by

$$x \equiv_{\alpha_s} y \quad \text{if, and only if,} \quad \alpha_s(x) = \alpha_s(y) \text{ in } A.$$

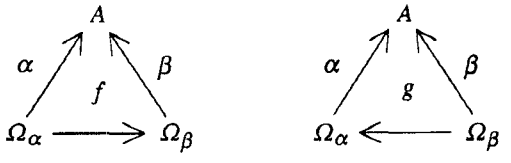
The relation is a Σ congruence on R.

Definition 4.1.1. Suppose the relation $\equiv_\alpha$ is recursive, that is, for each $s \in S$, $\equiv_{\alpha_s}$ is recursive on Ω_s. Then, we say A is *computable* under effective numbering α.

Suppose the relation $\equiv_\alpha$ is recursively enumerable, that is, for each $s \in S$, $\equiv_{\alpha_s}$ is recursively enumerable on Ω_s. Then, we say A is *semicomputable* under effective numbering α.

Both concepts are *finiteness conditions*, that is, isomorphism invariants possessed of all finite structures. Also noteworthy is this other invariance property first observed in Mal'cev [1961/1971].

LEMMA 4.1.2. *If A is a finitely generated algebra computable or semicomputable, under both $\alpha: \Omega_\alpha \to A$ and $\beta: \Omega_\beta \to A$, then α and β are recursively equivalent in the sense that there exists recursive functions f, g which commute the diagram:*



Let A be computable under α. Using the recursiveness of $\equiv_\alpha$ and the observation that α is an epimorphism from R to A, it is easy to prove this useful fact (see Bergstra and Tucker [1987]):

LEMMA 4.1.3. *Every computable algebra A is isomorphic to a recursive number algebra R whose carrier R_s is the set N of natural numbers, if A_s is infinite, and is the set $\{0, 1, \ldots, m-1\}$ of the first m natural numbers, if A_s is finite of cardinality m.*

Obviously, no such isomorphic representation is possible for the semicomputable algebras for otherwise they would be computable.

If A is computable under α, then an S sorted set $X \subseteq A^w$ is (α-)*computable* or (α-)*semicomputable* accordingly as

$$\alpha^{-1}(X) = \{(x_1, \ldots, x_k) \in \Omega^w : \alpha^w(x_1, \ldots, x_k) \in X\}$$

is recursive or recursively enumerable.

LEMMA 4.1.4. *Let A be a computable algebra and $\equiv$ a congruence on A. If $\equiv$ is computable or semicomputable then the factor algebra $A/\equiv$ is computable or semicomputable accordingly.*

4.2. COMPUTABLE TERM ALGEBRAS. The algebras $T(\Sigma)$ are computable under any of their standard gödel numberings. This was implicit in Section 1 where we spoke of a data type K being computable when its defining congruence $\equiv_K$ is decidable on $T(\Sigma)$. By Lemma 4.1.4 and isomorphism invariance, we can define equivalently a data type to be computable when its initial algebra is computable.

We will describe the computability properties of term algebras that we will need.

Definition 4.2.1. Let Σ be an S sorted signature and X a countable set of variable names used to make S sorted variables. Let $T(\Sigma, X)$ be the Σ term algebra over X. Then $T(\Sigma, X)$ is a computable algebra.

Let γ be a computable numbering for $T(\Sigma, X)$. Without loss of generality, we assume that γ has the following properties.

(i) Let G be a recursive Σ algebra of numbers with $G_s = N$ for all $s \in S$, and let

$$\gamma: G \to T(\Sigma, X) \qquad \text{and} \qquad \gamma^{-1}: T(\Sigma, X) \to G$$

be Σ-isomorphisms.

(ii) Let γ be *standard* in the sense of Mal'cev [1961/1971]; this means that we can decide when a term is a variable, compute subterms of a term, etc.

(iii) Let γ satisfy the following monotonic property: for each $s \in S$, and context $t(x^u) \in T_s(\Sigma, X)$ with single variable x^u of sort u, then for all $r_1, r_2 \in T_u(\Sigma, X)$,

$$\gamma_u^{-1}(r_1) < \gamma_u^{-1}(r_2) \qquad \text{implies} \qquad \gamma_u^{-1}(t(r_1)) < \gamma_u^{-1}(t(r_2)).$$

This means that the numbering induces an ordering on terms and that substitution into terms is monotonic with respect to this ordering. In a uniform way, for every minimal Σ algebra, a transversal can be found by taking the smallest terms in this ordering to represent the elements of A.

All standard numberings are recursively equivalent, by Lemma 4.1.2. Recall that $A \cong T(\Sigma)/\equiv_A$. The following result states the equivalence between the classes of computable and semicomputable algebras and those of algebras with decidable or semidecidable generalised word problems, respectively. A proof of this can be found in Mal'cev [1961/1971].

THEOREM 4.2.2. *Let A be a finitely generated algebra. Then A is computable or semicomputable if, and only if, $\equiv_A$ is computable or semicomputable on $T(\Sigma)$, respectively.*

From γ we can computably number the set $Eqn(\Sigma, X)$ of equations in an obvious way: $\gamma_e: [G_E \to Eqn(\Sigma, X)]$ is obtained from γ by pairing.

We may define $E \subset Eqn(\Sigma, X)$ to be *recursively enumerable* if $\gamma_e^{-1}(E)$ is recursively enumerable.

We note that if E is recursively enumerable then the closure $\acute{E}$ is recursively enumerable.

However, we define $E \subset Eqn(\Sigma, X)$ to be *recursive* if $\gamma_e^{-1}(E)$ is recursive and if $\gamma_e^{-1}(\acute{E})$ is recursive.

A starting point for the investigation of effectivity are these facts:

LEMMA 4.2.3. *Let (Σ, E) be a finite equational term rewriting system specification. Then*

(i) the basis set D_E and the one-step reduction relation $\rightarrow_{E(1)}$ are computable;
(ii) the reduction system $\rightarrow_E$ and the congruence $\equiv_E$ are semicomputable;
(iii) the set $NF(\Sigma, E)$ of normal forms is computable.

In particular, $T(\Sigma, E)$ is a semicomputable algebra. If (Σ, E) is recursive or recursively enumerable, then (i) and (ii) hold, but the set $NF(\Sigma, E)$ is cosemicomputable.

Notice that $NF(\Sigma, E)$ need not be a transversal for $\equiv_E$. We may now trivially prove the following:

LEMMA 4.2.4. *Let (Σ, E) be a finite equational term rewriting system specification which is Church–Rosser and weakly terminating. Then $T(\Sigma, E)$ is a computable algebra.*

PROOF. Given $t \in T(\Sigma)$, we can interleave the algorithms enumerating $NF(\Sigma, E)$ and $\equiv_E$ to seek the normal form of t which is guaranteed to exist from the weak termination 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 Lemma 4.2.4 is also that of this companion lemma to Lemma 4.1.4.

LEMMA 4.2.5. *Let A be a semicomputable algebra with semicomputable congruence $\equiv$. If there exists a semicomputable transversal for $\equiv$, then the factor algebra $A/\equiv$ is a computable algebra.*

4.3. THEOREMS ON RECURSIVE SPECIFICATIONS. We will now prove some first results about specifications of algebras without hidden functions.

THEOREM 4.3.1. *Let A be a minimal Σ algebra. If A is computable, then A possesses a possibly infinite recursive equational specification (Σ, E') that is an orthogonal complete term rewriting system.*

PROOF. Now $A \cong T(\Sigma)/\equiv_A$ and since A is computable we have that $\equiv_A$ is recursive in the numbering γ. Let $nf_\gamma: T(\Sigma) \rightarrow T(\Sigma)$ compute the smallest normal forms for $\equiv_A$ as determined by the enumeration γ; it is defined by

$$nf_\gamma(t) = \gamma((\textit{least } i)[t \equiv_A \gamma(i)]).$$

Define

$$E_A = \{t = nf_\gamma(t) : t \not\equiv nf_\gamma(t) \text{ and for each proper subterm } r \text{ of } t,\ nf_\gamma(r) \equiv r\}.$$

Let $T_\gamma = im(nf_\gamma)$, the image of nf_γ, be a transversal for $\equiv_A$. We claim that T_γ is the set of normal forms for the TRS (Σ, E_A). First, note that the terms in T_γ do not reduce since neither they nor their subterms occur on a LHS. This is because each subterm r has the property that $r \equiv nf_\gamma(r)$, for if not, then there is a contradiction with the enumeration property (iii) of the computable

numbering γ as formulated in 4.2.1 above. Next, note that the terms not in T_γ can be reduced. To see this, let $t \notin T_\gamma$ and let r be the smallest subterm of t such that $r \notin T_\gamma$ (perhaps r is t). Then r occurs on a LHS of an equation in E_A, and we may reduce t.

Thus, $NF(\Sigma, E_A) = T_\gamma$.

Next we show that (Σ, E_A) is strongly terminating. Because the Gödel numbering γ of $T(\Sigma)$ is monotonic, any reduction of $t \to t'$ entails $\gamma^{-1}(t) > \gamma^{-1}(t')$. Thus, an infinitely long reduction cannot occur because $(\mathbf{N}, <)$ is well founded.

We show that (Σ, E_A) is an orthogonal TRS. Clearly, since each rule has no variables, there is no rule with duplicated variables on its LHS, that is, the TRS is left-linear. If two rules overlap then, since the rules have closed terms on the LHS, one of the LHS must be a subterm of the other. By inspection of E_A, we see this is not the case. Thus, the TRS is nonoverlapping.

Thus, (Σ, E_A) is complete since it is strongly terminating and orthogonal.

Now, (Σ, E_A) is recursive since E_A is recursive using the computability of syntactic identity $\equiv$, nf_γ and subterm decomposition, and E_A is trivially closed under the renaming of its variables.

Finally, since $NF(\Sigma, E_A) = T_\gamma$ is a transversal for $=_A$ and trivially $A \in Alg(\Sigma, E_A)$, we deduce that

$$A \cong T(\Sigma, E_A)$$

by Lemma 3.4.3. Q.E.D.

COROLLARY 4.3.2. *Let A be a Σ minimal algebra. If A is finite, then A possesses a finite orthogonal complete TRS specification (Σ, E).*

PROOF. To see this note that $im(nf_\gamma)$ is finite, its cardinality is that of A, and each LHS in E_A is of the form $g(t_1, \ldots, t_k)$ with g a function symbol in Σ and $t_i \in im(nf_\gamma)$ for $1 \leq i \leq k$. Since Σ is finite, we know that E_A is finite. Q.E.D.

Consider the size of (Σ, E) in the case of the corollary. Let

$$l = \max\{|A_s| : s \in S\}$$
$$k = \max\{|w| : \Sigma_{w,s} \neq \varnothing\}$$
$$m = |\Sigma| = |S| + |\cup\{\Sigma_{w,s} : w \in S^*, s \in S\}|.$$

Then $|E| < ml^k$.

5. *Proof of Theorem* 1.1

Let us restate the first theorem.

THEOREM 1.1. *Let A be a finitely generated minimal Σ algebra. Then the following are equivalent*:

(1) *A is computable.*
(2) *There is a finite equational specification (Σ_0, E_0) such that*
  (i) $Sort(\Sigma) = Sort(\Sigma_0)$ and $\Sigma \subseteq \Sigma_0$;
  (ii) (Σ_0, E_0) is a complete term rewriting system;
  (iii) $I(\Sigma_0, E_0)|_\Sigma \cong A$.

Furthermore, the (Σ_0, E_0) *may be taken to be an orthogonal term rewriting system, and the sizes* $|\Sigma_0|$ *and* $|E_0|$ *depend upon the algebra* A.

The fact that statement (2) implies statement (1) was established in Lemma 4.2.4; we prove that (1) implies (2).

Let $S = \{1, 2, \ldots, n\}$ be a set of n sorts. Let Σ be an S sorted signature with p constants and q operations. Let A be any computable minimal Σ algebra.

If A is finite then there exists an appropriate finite equational specification (Σ, E) for A by Corollary 4.3.2.

Here is the idea of the proof in the case that A is infinite.

Suppose A is infinite with l infinite carriers

$$A_1, \ldots, A_l$$

and $n - l$ finite carriers

$$A_{l+1}, \ldots, A_n.$$

We suppose that for $i = l + 1, \ldots, n$

$$|A_i| = b_i + 1.$$

Of course, A need not have finite carriers and $l = n$.

Since A is computable, by Lemma 4.1.3, we can take A to be isomorphic to a recursive algebra R of numbers and we can concentrate on building an appropriate finite equational specification for R, which will also be a specification for A.

First, we build an expansion R_0 of R by adding constants and functions such that

$$R_0|_\Sigma = R.$$

The design of R_0 involves coding all the carriers into one of the infinite carriers and simulating the algebraic operations using recursive functions and the Kleene T-predicate.

Next, we build a finite equational specification (Σ_0, E_0) for R_0 that we will prove is a complete TRS. Here, we exploit algebraic aspects of the definition of the recursive functions by recursion schemes. This equational specification of R_0 does not involve hidden sorts or functions.

The specification (Σ_0, E_0) serves as an appropriate specification for R that does involve hidden functions.

5.1. CONSTRUCTION OF R_0. Using Lemma 4.1.3, let $R \cong A$ and for $i = 1, \ldots, l$

$$R_i = \mathbf{N}$$

and $i = l + 1, \ldots, n$

$$R_i = \{0, 1, \ldots, b_i\} \subset \mathbf{N}.$$

We assume that each of the p constants has the form

$$c \in R_s$$

for some $s \in S$, and that each of the q operations are total recursive functions on $\mathbf{N}$ having the form

$$f: R_{w(1)} \times \cdots \times R_{w(k)} \to R_s$$

for some $w(1), \ldots, w(k), s \in S$.

To make R_0, we add to R the following constants and functions:

5.1.1. *Counting.* For $i = 1, \ldots, n$ we add the constant zero

$$^i0 \in R_i$$

and the successor function

$$^i succ: R_i \to R_i$$

defined as follows: for infinite sorts $i = 1, \ldots, l$

$$^i succ(x) = x + 1,$$

and for finite sorts $i = l + 1, \ldots, n$

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

5.1.2. *Coding.* We add functions that code all the carriers into the first infinite carrier R_1, namely: for $i = 1, \ldots, n$

$$^i fold: R_i \to R_1$$

is defined by

$$^i fold(x) = x$$

for $x \in \mathbf{N}$. We add functions that decode from R_1, namely:

$$^i unfold: R_1 \to R_i$$

defined for $i = 1, \ldots, l$ by

$$^i unfold(x) = x$$

for $x \in \mathbf{N}$, and for $i = l + 1, \ldots, n$ by

$$^i unfold(x) = min(x, b_i).$$

for $x \in \mathbf{N}$.

5.1.3. *Tracking Functions.* For each operation $f: R_{w(1)} \times \cdots \times R_{w(k)} \to R_s$ of R we add a recursive function $f': R_1 \times \cdots \times R_1 \to R_1$ that simulates f on R_1. This f' is defined by

$$f'(x_1, \ldots, x_k) = {}^s fold\Big(f\big({}^{w(1)} unfold(x_1), \ldots, {}^{w(k)} unfold(x_k)\big)\Big)$$

for $x_1, \ldots, x_k \in R_1$.

5.1.4. *Enumeration Functions.* We add new functions designed to compute each tracking function f' on R_1 in a special way as follows: let $\phi: \mathbf{N}^k \to \mathbf{N}$ be any total recursive function. Then, by the *Kleene Normal Form Theorem*, this may be written

$$\phi(x) = U(\mu z \cdot T^k(e, x, z)),$$

where U and T are the so called *Kleene computation function* and *T predicate*, respectively, and e is some index for ϕ, $x \in N^k$, and $z \in N$. Since U and T^k are primitive recursive so are the functions

$$h(z, x) = U(\mu z' \leq z \cdot [z' = z \text{ or } T^k(e, x, z')])$$
$$g(z, x) = 0 \quad \text{if} \quad \exists z' \leq z \cdot T^k(e, x, z')$$
$$= 1 \quad \text{otherwise.}$$

From these functions we can define a recursive function

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

to simulate the least number operator. Thus, ϕ is factorized into t, h, g in the sense that

$$\phi(x) = t(0, x, 1).$$

(The reader may care to consult Cutland [1980].)

We apply this method for ϕ to each tracking function f' on R_1 and add the constructed functions h, g, and t.

5.1.5. *Subfunctions.* Finally, for each tracking map f', we add each primitive recursive subfunction λ of its primitive recursive enumeration functions g, h. Let Δ be this list of all the subfunctions of the enumeration functions g and h. Such a list is added for each of the q tracking maps.

5.1.6. *Size.* Now R_0 is the many sorted algebra obtained by adding all the above functions. Clearly,

$$R_0|_\Sigma = R.$$

It is instructive to count these additions. Since Σ has p constants and q functions, to make the signature Σ_0, we must add the following:

n	constants
n	successors
n	folds
n	unfolds
q	tracking functions
$3q$	enumeration functions
Π	subfunctions

where $\Pi = \Pi_1 + \cdots + \Pi_q$ and Π_i is the number of subfunctions in some primitive recursive definitions of g and h associated with the ith tracking operation f'. Note that Π is dependent on the algebra, whereas the other numbers are independent of the algebra and dependent only on the number of sorts, constants and functions in Σ.

5.2. CONSTRUCTION OF (Σ_0, E_0). We now define specification (Σ_0, E_0) for R_0. Let Σ have the form

constants $\cdots$
$\underline{c}: \to s$
$\cdots$

operations $\cdots$
$F: w(1) \times \cdots \times w(k) \to s$
$\cdots$

where there are p constants and q operations.

Then Σ_0 has the form of Σ with the following adjoined:

constants	${}^i0: \to i$	$1 \leq i \leq n$
operations	${}^i\text{SUCC}: i \to i$	$1 \leq i \leq n$
	${}^i\text{FOLD}: i \to 1$	$1 \leq i \leq n$
	${}^i\text{UNFOLD}: 1 \to i$	$1 \leq i \leq n$
	$\cdots$	
	$F': 1^k \to 1$	
	$G: 1 \times 1^k \to 1$	for each operation F of Σ
	$H: 1 \times 1^k \to 1$	
	$T: 1 \times 1^k \times 1 \to 1$	
	$\underline{\Delta}$	
	$\cdots$	

We will often use $\underline{\lambda}$ for a notation naming a function λ in R_0.

The equations of E_0 are constructed as follows:

5.2.1. *Counting.* For finite sorts $i = l + 1, \ldots, n$

$$ {}^iSUCC({}^iSUCC^{b_i}({}^i0)) = {}^iSUCC^{b_i}({}^i0). \tag{1}$$

5.2.2. *Coding.* For infinite sorts $i = 1, \ldots, l$

$$ {}^iFOLD({}^i0) = {}^10 \tag{2}$$

$$ {}^iFOLD({}^iSUCC({}^iX)) = {}^1SUCC({}^iFOLD({}^iX)). \tag{3}$$

For finite sorts $i = l + 1, \ldots, n$

$$ {}^iFOLD({}^i0) = {}^10 \tag{4}$$

$$ {}^iFOLD({}^iSUCC^k({}^i0)) = {}^1SUCC^k({}^10) \tag{5}$$

and for $k = 1, \ldots, b_i$.

For all sorts, $i = 1, \ldots, n$

$$ {}^iUNFOLD({}^10) = {}^i0 \tag{6}$$

$$ {}^iUNFOLD({}^1SUCC({}^1X)) = {}^iSUCC({}^1UNFOLD({}^1X)). \tag{7}$$

5.2.3. *Tracking.* For each constant $\underline{c}$ of sort s naming $c \in R_s$,

$$ \underline{c} = {}^sSUCC^c({}^s0). \tag{8}$$

For each operation symbol $F \in \Sigma_{w,s}$,

$$ F\left({}^{w(1)}X_1, \ldots, {}^{w(k)}X_k\right) = {}^sUNFOLD\left(F'\left({}^{w(1)}FOLD\left({}^{w(1)}X_1\right), \ldots, {}^{w(k)}FOLD\left({}^{w(k)}X_k\right)\right)\right). \tag{9}$$

5.2.4. *Enumeration and Subfunctions.* For each tracking function f' for each function f we add equations for the g, h, and t, and primitive subfunctions Δ of g and h, as follows:

For t we add

$$T(Z, X, {}^1 0) = H(Z, X) \tag{10}$$

$$T(Z, X, {}^1SUCC(Y)) = T({}^1SUCC(Z), X, G({}^1SUCC(Z), X)) \tag{11}$$

and

$$F'(X) = T({}^1 0, X, {}^1SUCC({}^1 0)) \tag{12}$$

For each primitive recursive function $\lambda \in \Delta \cup \{g, h\}$, we add equations for its name $\underline{\lambda}$, using a case distinction on the defining equations for λ:

If $\lambda(x_1, \ldots, x_k) = x_i$, then add $\underline{\lambda}(X_1, \ldots, X_k) = X_i$ (13i)

If $\lambda(y) = y + 1$, then add $\underline{\lambda}(Y) = {}^iSUCC(Y)$ (13ii)

If $\lambda(x) = \mu(\mu_1(x), \ldots, \mu_n(x))$, then add $\underline{\lambda}(X) = \underline{\mu}\big(\underline{\mu}_1(X), \ldots, \underline{\mu}_n(X)\big)$ (13iii)

where $x = (x_1, \ldots, x_k)$ and $X = (X_1, \ldots, X_k)$.

If $\lambda(0, x) = \mu_1(x)$ and $\lambda(y + 1, x) = \mu_2(y, x, \lambda(y, x))$

then add

$$\underline{\lambda}(\underline{0}, X) = \underline{\mu}_1(X) \tag{13iv}$$

$$\underline{\lambda}({}^iSUCC(Y), X) = \underline{\mu}_2(Y, X, \underline{\lambda}(Y, X)) \tag{13v}$$

where, again, x and X are possibly vectors.

Let us count the equations:

$n - l$	counting
$2l + 2(b_{l+1} + \cdots + b_n) + 2n$	coding
p	tracking constants
q	tracking operations
$3q$	t and tracking functions
Π_e	g, h, and Δ

The method for primitive recursive functions adds at most two equations per function; in previous notation, we note

$$\Pi_e < 2.2q + 2.\ \Pi = 2(2q + \Pi).$$

Thus, the number of equations depends on the size of the finite carriers and the structure of (the enumeration of) the operations.

LEMMA 5.2.4.1. *The finite equational specification* (Σ_0, E_0) *is an initial algebra specification of* R_0, *that is,*

$$R_0 \cong T(\Sigma_0, E_0).$$

As a TRS, (Σ_0, E_0) *has the following properties*:

(*i*) *it is orthogonal*;
(*ii*) *it is strongly terminating*;
(*iii*) *the sets of normal forms are, for infinite sorts* $i = 1, \ldots, l$,

$$NF_i(\Sigma_0, E_0) = \{{}^iSUCC^k({}^i0) : k = 0, 1, \ldots\}$$

and for finite sorts $i = l + 1, \ldots, n$

$$NF_i(\Sigma_0, E_0) = \left\{{}^iSUCC^k({}^i0) : k = 0, 1, \ldots, b_i\right\}.$$

From (*i*) *and* (*ii*), *it follows that* (Σ_0, E_0) *is a complete TRS.*

PROOF. Clearly, R_0 satisfies E_0 by construction. We will prove these statements in the order (i) followed by (iii). Then we will assume (ii) and deduce that $R_0 \cong T(\Sigma_0, E_0)$. Finally, we prove (ii) in the next subsection 5.3.

(i) The TRS (Σ_0, E_0) is left linear and nonoverlapping by inspection.
(iii) For $i = 1, \ldots, l$ define

$$T_i = \{{}^iSUCC^k({}^i0) : k = 0, 1, \ldots\}$$

and for $i = l + 1, \ldots, n$ define

$$T_i = \left\{{}^iSUCC^k({}^i0) : k = 0, 1, \ldots, b_i\right\}.$$

Clearly, for each $i = 1, \ldots, n$,

$$T_i \subset NF_i(\Sigma_0, E_0)$$

because no equation of E_0 has a LHS that matches with the numerals of T_i.

Next we consider the converse inclusion. Let $t \notin T_i$ we show that t reduces and so is not a normal form.

Let r be the smallest subterm of t such that $r \notin T_i$. There are three cases:

(a) If i is a finite sort and $r \equiv {}^iSUCC^k({}^i0)$ for $k > b_i$. Then there is a reduction of r by means of eq. (1) for counting in 4.2.1, and so t is not a normal form.
(b) If $r \equiv {}^i\underline{c}$, then there is a reduction by eq. (8) for constants in 5.2.3.
(c) If $r \equiv \underline{\lambda}(t_1, \ldots, t_k)$ for $\underline{\lambda}$ any function symbol, then $t_1, \ldots, t_k$ must be numerals by the minimality of r. For each choice of $\underline{\lambda}$, there is a reduction of r by means of an equation of E_0 and so t is not a normal form.

We prove isomorphism using 5.2.4.1(ii); we define a map $\phi: R_0 \to T(\Sigma_0, E_0)$ which is a family of maps

$$\phi_i: R_i \to T_i(\Sigma_0, E_0)$$

for $i = 1, \ldots, n$, where

$$\phi_i(x) = [{}^iSUCC^x({}^i0)]$$

for all $x \in R_i$. Since the numerals are normal forms 5.2.4.1(iii); the map is injective. If the strong termination property 5.2.4.1(ii) holds then every term has a normal form that is a numeral and so the map is surjective.

We show ϕ is a homomorphism.

Let $\underline{\lambda}$ be any operation symbol in Σ_0 of type $u(1) \times \cdots \times u(k) \to \nu$ naming the operation $\underline{\lambda}$ in $T(\Sigma_0, E_0)$ and the operation λ in R_0; we show that

$$\phi_\nu(\lambda(x_1, \ldots, x_k)) = \underline{\lambda}\big(\phi_{u(1)}(x_1), \ldots, \phi_{u(k)}(x_k)\big).$$

The LHS is

$$[{}^\nu SUCC^{\lambda(x_1, \ldots, x_k)}({}^\nu 0)].$$

The RHS is

$$\underline{\lambda}([{}^{u(1)}SUCC^{x_1}({}^{u(1)}0)], \ldots, [{}^{u(k)}SUCC^{x_k}({}^{u(k)}0)])$$
$$= \big[\underline{\lambda}({}^{u(1)}SUCC^{x_1}({}^{u(1)}0), \ldots, {}^{u(k)}SUCC^{x_k}({}^{u(k)}0))\big],$$

by definition of $\underline{\lambda}$ in $T(\Sigma_0, E_0)$.

To show that these terms are equivalent in E_0 we reason as follows: Note that since R_0 is an algebra of numbers,

$$R_0 \models {}^\nu SUCC^{\lambda(x_1, \ldots, x_k)}({}^\nu 0) = \underline{\lambda}({}^{u(1)}SUCC^{x_1}({}^{u(1)}0), \ldots, {}^{u(k)}SUCC^{x_k}({}^{u(k)}0)).$$

If the strong termination property 5.2.5(ii) holds, since the numerals are the normal forms, we have

$$E_0 \vdash \underline{\lambda}({}^{u(1)}SUCC^{x}1({}^{u(1)}0), \ldots, {}^{u(k)}SUCC^{x}k({}^{u(k)}0)) = {}^\nu SUCC^{y}({}^\nu 0)$$

for some $y \in \boldsymbol{N}$. Thus, because R_0 satisfies E_0,

$$R_0 \models {}^\nu SUCC^{y}({}^\nu 0) = {}^\nu SUCC^{\lambda(x_1, \ldots, x_k)}({}^\nu 0)$$

and, by uniqueness, $y = \lambda(x_1, \ldots, x_k)$. Therefore,

$$E_0 \vdash \underline{\lambda}({}^{u(1)}SUCC^{x_1}({}^{u(1)}0), \ldots, {}^{u(k)}SUCC^{x_k}({}^{u(k)}0)) = {}^\nu SUCC^{\lambda(x_1, \ldots, x_k)}({}^\nu 0),$$

and the equivalence classes are identified, and ϕ is a homomorphism. Q.E.D.

5.3. PROOF OF STRONG TERMINATION. We prove (ii) of Lemma 5.2.4.1, that each $t \in T(\Sigma_0)$ is strongly terminating, by induction on the complexity of t.

As basis, we consider all constant symbols. If $t \equiv {}^i 0$, then we know that no reduction is possible from E_0 because it is a normal form.

If $t \equiv \underline{c}$ of sort s names the constant $c \in R_i$, then, by inspection of E_0, there is at most one reduction possible, by means of eq. 8, and this leads to a normal form ${}^s SUCC^c({}^s 0)$.

We formulate the induction step as follows:

LEMMA 5.3.1. *Let $s_1, \ldots, s_k \in T(\Sigma_0)$ be strongly terminating and let $\underline{\lambda}$ be a k-ary function symbol of Σ_0. Then $\underline{\lambda}(s_1, \ldots, s_k)$ is strongly terminating.*

PROOF. We prove this by induction on an ordering of the function symbols.

First, we order the signature Σ_0 by ordering the operations of R_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

$${}^1 succ, \ldots, {}^n succ, {}^1 fold, \ldots, {}^n fold, {}^1 unfold, \ldots, {}^n unfold,$$
$$\Delta_1, \ldots, \Delta_q, h_1, \ldots, h_q, g_1, \ldots, g_q, t_1, \ldots, t_q, f_1', \ldots, f_q', f_1, \ldots, f_q$$

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 this ordering of Σ_0.

The proof divides into a basis case, and an induction step involving nine cases, many of which split into subcases. In every case, we argue by contradiction. We assume that an infinite reduction

$$t \to t_1 \to t_2 \to \cdots \to t_N \to t_{N+1} \to \cdots$$

exists for $t \equiv \underline{\lambda}(s_1, \ldots, s_k)$ and $s_1, \ldots, s_k$ strongly terminating. We assume that $t_N \to t_{N+1}$ is the first reduction where a rule is used that involves the outermost function symbol $\underline{\lambda}$. Such an N must exist for otherwise the infinite reduction would contradict the assumption that the $s_1, \ldots, s_k$ are strongly terminating. Thus, $t_1, \ldots, t_N$ have the forms

$$t_i \equiv \underline{\lambda}(r_{i1}, \ldots, r_{ik})$$

for $i = 1, \ldots, N$, and for some $1 \leq \alpha(i) \leq k$

$$r_{ij} = r_{i+1j} \quad \text{for} \quad j \neq \alpha(i)$$
$$r_{ij} \to r_{i+1j} \quad \text{for} \quad j = \alpha(i).$$

Note that $t_N \equiv \underline{\lambda}(r_{N_1}, \ldots, r_{N_k})$ and that the subterms $r_{N_1}, \ldots, r_{N_k}$ are strongly terminating because the $s_1, \ldots, s_k$ are.

We will consider such reduction sequences for every type of operator $\underline{\lambda}$ in the ordered list. For convenience, we would like to assume that $N = 1$ and that the first reduction involves $\underline{\lambda}$; this is possible:

LEMMA 5.3.2. *The following are equivalent*:

(*a*) *For all strongly terminating terms* $r_1, \ldots, r_k$, *each reduction sequence of* $\underline{\lambda}(r_1, \ldots, r_k)$ *is finite and* $\underline{\lambda}(r_1, \ldots, r_k)$ *is strongly terminating.*
(*b*) *For all strongly terminating terms* $r_1, \ldots, r_k$ *each reduction sequence of* $\underline{\lambda}(r_1, \ldots, r_k)$ *that begins with a reduction involving* $\underline{\lambda}$ *is finite.*

PROOF. That (a) implies (b) is immediate. Consider the converse. Consider the arbitrary infinite reduction sequence

$$t \to t_1 \to t_2 \to \cdots \to t_N \mapsto t_{N+1} \to \cdots .$$

If this is infinite, then the reduction sequence

$$t_N \to t_{N+1} \to \cdots$$

is infinite and involves $\underline{\lambda}$ at the first step. By (b), we deduce that this is impossible. Hence, t is strongly terminating. Q.E.D.

Thus, in each argument, we will examine the reduction

$$t \to t_1$$

and show that t_1 is strongly terminating (often it cannot be further reduced).

We now begin the proof by induction on the position of $\underline{\lambda}$ in the list.

Basis 5.3.3. $\underline{\lambda}$ is ${}^1SUCC$ and $t \equiv {}^1SUCC(s)$.

There is no equation in E_0 that allows us to rewrite ${}^1SUCC$, so an infinite reduction sequence from t starting with such a rewrite is impossible.

Induction Steps 5.3.4. We assume the Lemma 5.3.1 is true for all function symbols preceeding $\underline{\lambda}$ in the list. There are nine cases and several subcases.

Case 1. $\underline{\lambda}$ is ${}^{i}SUCC$ for $1 < i \leq n$.

Subcase 1a. The sort $i = 2, \ldots, l$ is infinite. Then, the argument is exactly the same as in the basis case above.

Subcase 1b. The sort $i = l + 1, \ldots, n$ is finite. Consider a $\underline{\lambda}$ first step reduction sequence

$$t \to t_1 \to t_2 \to \cdots .$$

Only eq. (1) allows $\underline{\lambda}$ to be reduced and this leads to a normal form that cannot be reduced further. Thus, no infinite reduction sequence exists.

Case 2. $\underline{\lambda}$ is ${}^{i}FOLD$ for $1 \leq i \leq n$ and $t \equiv {}^{i}FOLD(s)$.

Subcase 2a. The sort $i = 1, \ldots, l$ is infinite.

Only rules (2) and (3) can be applied to the reduction $t \to t_1$ in a $\underline{\lambda}$ first step reduction sequence. We show that t is strongly terminating by induction on the value $val(s)$ of s in R_0.

If $val(s) = 0$, then eq. (3) cannot be applied because otherwise s must have the form ${}^{i}SUCC(z)$, for some z, which contradicts that $val(s) = 0$, by definition of *val*. So eq. (2) is applied which entails $s \equiv {}^{i}0$ and hence $t_1 \equiv {}^{i}0$ and a normal form. Hence, the sequence terminates.

For the induction step, suppose that for all strongly terminating s such that $val(s) = d$, the term $t \equiv {}^{i}FOLD(s)$ is strongly terminating.

Consider some s that is strongly terminating and $val(s) = d + 1$. In the reduction $t \to t_1$ eq. (2) cannot be applied because $val(s) \neq 0$ and so s is not ${}^{i}0$. Thus, eq. (3) is applied whence $s \equiv {}^{i}SUCC(z)$, for some z, and $t_1 \equiv {}^{1}SUCC({}^{i}FOLD(z))$. Now $val(z) = d$ so by the induction hypothesis on d we know that ${}^{i}FOLD(z)$ is strongly terminating. By the basis case of induction on the list, we know t_1 is strongly terminating.

Subcase 2b. The sort $i = l + 1, \ldots, n$ is finite.

Only rules (4) and (5) can be applied to the reduction $t \to t_1$ and in both cases t_1 must be a normal form.

Case 3. $\underline{\lambda}$ is ${}^{i}UNFOLD$ for $1 \leq i \leq n$ and $t \equiv {}^{i}UNFOLD(s)$.

In the reduction $t \to t_1$ only eq. (6) and (7) can be applied. The argument that t is strongly normalising follows that of subcase 2a and uses induction on $val(s)$ in R_0.

Case 4. $\underline{\lambda}$ names a primitive recursive function on R_1 from

$$\Delta_1, \ldots, \Delta_q, h_1, \ldots, h_q, g_1, \ldots, g_q$$

Note that $\underline{\lambda}$ is constructed from functions earlier in the list. There are five subcases corresponding with the five parts of the definition of primitive recursion. We will classify these cases using the function named by $\underline{\lambda}$.

Subcase 4a. *Constant function.* $\underline{\lambda}$ names $\lambda(x_1, \ldots, x_k) = {}^{1}0$.

If $t \equiv \underline{\lambda}(s_1, \ldots, s_k)$, then the only equation that applies is (13i) and so $t_1 \equiv {}^{1}0$ and is a normal form. The sequence halts and t is strongly terminating.

Subcase 4b. *Projection.* $\underline{\lambda}$ names $\lambda(x_1, \ldots, x_k) = x_j$.

If $t \equiv \underline{\lambda}(s_1, \ldots, s_k)$, then the only equation that applied is (13ii). Hence, $t_1 \equiv s_j$, which is strongly terminating by hypothesis.

Subcase 4c. *Successor.* $\underline{\lambda}$ names $\lambda(x) = {}^1SUCC(x)$.

If $t \equiv \underline{\lambda}(s)$, then the only equation that applied is (13iii). Hence, $t_1 \equiv {}^1SUCC(s)$, which is strongly terminating by the earlier basis case of this lemma. Hence, t is strongly terminating.

Subcase 4d. *Composition* $\underline{\lambda}$ names

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

where $x = (x_1, \ldots, x_k)$.

If $t \equiv \underline{\lambda}(s)$, where $s = (s_1, \ldots, s_k)$, then the only equation that applies is (13iv). Then

$$t_1 \equiv \underline{\mu}\big(\underline{\mu}_1(s), \ldots, \underline{\mu}_n(s)\big),$$

where $\underline{\mu}, \underline{\mu}_1, \ldots, \underline{\mu}_n$ name the subfunctions of λ. Since the $\underline{\mu}, \underline{\mu}_1, \ldots, \underline{\mu}_n$ preceed $\underline{\lambda}$ in the list, by the main induction hypothesis we know that $\underline{\mu}_1(s), \ldots, \underline{\mu}_n(s)$ are strongly terminating and, further, that t_1 is strongly terminating, hence so is t.

Subcase 4e. *Primitive recursion.* $\underline{\lambda}$ names the function

$$\lambda(0, x) = \mu_1(x)$$

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

where $x = (x_1, \ldots, x_k)$.

Let $t \equiv \underline{\lambda}(r, s)$ where s is $(s_1, \ldots, s_k)$. The only equations that apply are (13v) and (13vi). We show that t is strongly terminating by induction on the value $val(r)$ in R_0.

If $val(r) = 0$ then the only equation that applies is (13v) and $t_1 \equiv \underline{\mu}_1(s)$. Since $\underline{\mu}_1$ occurs before $\underline{\lambda}$ we know that $\underline{\mu}_1(s)$ is strongly terminating and so is t.

If $val(r) > 0$, then we take as induction hypothesis the following: for all strongly terminating r, s such that $val(r) = d$ the term $\underline{\lambda}(r, s)$ is strongly terminating.

Let $t \equiv \underline{\lambda}(r, s)$ for $val(r) = d + 1$. The only equation that applies is (13vi). Thus, $r \equiv {}^1SUCC(z)$ for some z and

$$t_1 \equiv \underline{\mu}_2(z, s, \underline{\lambda}(z, s)).$$

By the induction hypothesis on $val(r)$, we know that $\underline{\lambda}(z, s)$ is strongly terminating, because $val(z) = d$. Therefore, by the main induction hypothesis of 5.3, since $\underline{\mu}_2$ preceeds $\underline{\lambda}$ in the list, and $z, s, \underline{\lambda}(z, s)$ are strongly terminating, we know that t_1 and hence t are strongly terminating.

Case 5. λ is T_j and $t \equiv T_j(r, s, u)$ where s is $(s_1, \ldots, s_k)$.

The only equations that can be applied to the first reduction are (10) and (11). Define

$$\chi(r, s) = (\mu z)\big[g_j(z, val(s)) = 0\big] - val(r).$$

We argue using induction on $\chi(r, s)$.

Basis. $\chi(r, s) = 0$.

Subcase a. $val(u) = 0$. Then $u \equiv {}^1 0$ and the equation used is (10), and $t_1 \equiv H_j(r, s)$. Since H_j preceeds T_j in the list, by induction, since r and s are strongly terminating so t_1 and (hence) t are strongly terminating.

Subcase b. $val(u) \neq 0$. Then the equation used is (11) and

$$t_1 \equiv T_j\big({}^1SUCC(r), s, G_j({}^1SUCC(r), s)\big).$$

First, note that

$${}^1SUCC(r), s \text{ and } G_j({}^1SUCC(r), s)$$

are strongly normalising by hypothesis and/or by induction on the list. It follows that an infinite reduction of t_1 must involve a further application of eqs. (10) and (11). For this to happen $G_j({}^1SUCC(r), s)$ must reduce to either ${}^1 0$ or to ${}^1SUCC(z)$ for some z.

CLAIM. $val(G_j({}^1SUCC(r), s)) = 0$.

PROOF OF CLAIM. Let $z_0 = (\mu z)[g_j(z, val(s)) = 0]$. Since $\chi(r, s) = 0$, we have

$$z_0 < val(r) < val({}^1SUCC(r)).$$

Now, if $z_0 < z$ and $g_j(z_0, x) = 0$, then $g_j(z, x) = 0$. Thus, $g_j({}^1SUCC(r), s) = 0$ and the claim follows.

By the claim, the subterm of t_1 reduces to ${}^1 0$, and t_1 reduces by (10) to

$$H_j(SUCC(r), s),$$

which is strongly terminating by induction on the list.

Induction step. We suppose as induction hypothesis that for any r, s, and u that are strongly terminating and $\chi(r, s) = d$, $t \equiv T_j(r, s, u)$ is strongly terminating.

Suppose $\chi(r, s) = d + 1$.

Case a. $val(u) = 0$. Then the argument is as for case (a) of the basis of this case 5.

Case b. $val(u) \neq 0$. Then the equation used is (11) and

$$t_1 \equiv T_j\big({}^1SUCC(r), s, G_j({}^1SUCC(r), s)\big).$$

All subterms are strongly terminating and we note that

$$\chi({}^1SUCC(r), s) = d$$

by definition of χ. Thus, by the induction hypothesis on $\chi(r, s)$ we have that t_1 and (hence) t are strongly terminating.

This concludes the case 5.

Case 6. λ is F_j' and $t \equiv F_j'(s_1, \ldots, s_k)$
The only equation that applies is (12) which means that

$$t_1 \equiv T_j\big({}^1 0, s_1, \ldots, s_k, {}^1 SUCC({}^1 0)\big),$$

which is strongly terminating by case 5.

Case 7. λ is F and $t \equiv F(s_1, \ldots, s_k)$
The only equation that applies is (9) which means that

$$t_1 \equiv {}^s UNFOLD\big(F'\big({}^{w(1)} FOLD(s_1), \ldots, {}^{w(k)} FOLD(s_k)\big)\big).$$

By the previous cases 1, 2, and 6, we deduce that t_1 and (hence) t are strongly terminating. Q.E.D.

Up to case 5, where T_j appears, our proof could be replaced by a shorter but more advanced argument based on the recursive path ordering.

6. *Examples of Term Rewriting Specifications*

Theorem 1.1 shows that any computable algebra has a finite equational complete TRS specification, but that the number of equations in it may be large (in relation to earlier results) and involves hidden functions. We will show that these features are necessary.

6.1. An Example of a Family of Finite Algebras such that All Complete TRS Equational Specifications Are Dependent on the Size of the Algebras.

Let $n > 1$ and Σ_n be the signature

sort s
constants $c_1, \ldots, c_n : s$

and consider the Σ_n algebra

$$A_n = (\{a\} : a, \ldots, a)$$

wherein the n constants of Σ_n are identified.

Theorem 6.1.1. *Let (Σ, E) be a finite equational TRS such that $\Sigma_n \subseteq \Sigma$, (Σ, E) is complete and*

$$I(\Sigma, E)|_{\Sigma_n} \cong A_n.$$

Then $|E| > n - 1$.

Proof. Let $t \in T(\Sigma)$ be the normal form of c_1. Then, for $1 \leq j \leq n$, t is the normal form of c_j. To see this note that

$$A_n \models c_1 = c_j$$

and, because A_n is specified by (Σ, E),

$$E \vdash c_1 = c_j$$

so c_1 and c_j have a common reduct because (Σ, E) is confluent. Thus, we deduce that *at most one* of $c_1, \ldots, c_n$ could be the normal form t.

Suppose that c_j is not the normal form t. Then, there must be a rule

$$e_j \equiv l(X) = r(X)$$

that applies to c_j for $X = X_1, \ldots, X_l$. Thus, $l(X) \equiv X_i$ or $l(X) \equiv c_j$.

If $l(X) \equiv X_i$, then the TRS (Σ, E) cannot be terminating and indeed according to the definitions (Σ, E) does not qualify as a TRS, so we exclude this case and assume that $l(X) \equiv c_j$.

Now for the $n - 1$ cases, where c_j is not a normal form, we obtain $n - 1$ equations e_j with LHS c_j. Thus, $|E| > n - 1$. Q.E.D.

6.2. An Example of an Algebra such that for All Finite Complete TRS Equational Specifications with Hidden Functions the Normal Forms Must Involve Those Hidden Functions. Let Σ_0 be the signature

sorts	*nat*
	set
constants	0 : *nat*
	$\varnothing$: *set*
operations	S: *nat* $\rightarrow$ *nat*
	INS: *nat* $\times$ *set* $\rightarrow$ *set*

and let E_0 be the set of equations

$$INS(X, INS(Y, Z)) = INS(Y, INS(X, Z))$$
$$INS(X, INS(X, Z)) = INS(X, Z).$$

The initial algebra of (Σ_0, E_0) is isomorphic with

$$F = \left(\mathbf{N}, P_{fin}(\mathbf{N}); 0, \varnothing, n + 1, ins\right)$$

where $P_{fin}(\mathbf{N})$ is the set of finite subsets of $\mathbf{N}$ and

$$ins: \mathbf{N} \times P_{fin}(\mathbf{N}) \rightarrow P_{fin}(\mathbf{N})$$
$$ins(n, A) = \{n\} \cup A.$$

This example is taken from Bergstra et al. [1989].

Theorem 6.2.1. *Let (Σ, E) be a finite equational complete TRS specification for the Σ_0 algebra F, so that*

$$T(\Sigma, E)|_{\Sigma_0} \cong F.$$

Then there is a $t \in T(\Sigma_0)$ whose normal form $t_1 \in T(\Sigma)$ with respect to (Σ, E) does not lie in $T(\Sigma_0)$.

Proof. First consider the terms of $T(\Sigma_0)$. Any $t \in T_{nat}(\Sigma_0)$ has the form $S^l(0)$. Any $t \in T_{set}(\Sigma_0)$ has the following form:

$$t \equiv INS(S^{l(1)}(0), INS(S^{l(2)}(0), \ldots, INS(S^{l(d)}(0), \varnothing) \cdots);$$

we abbreviate t by

$$INS(l(1), \ldots, l(d), \varnothing),$$

or by $INS(l, \varnothing)$ for $l = (l(1), \ldots, l(d))$, when convenient.

Now let (Σ, E) be a complete TRS specification for the algebra F. Notice that E cannot contain any rule $l(X) = r(X)$ where $l(X)$ and $r(X)$ are terms of sort *nat*. This is because $T_{nat}(\Sigma, E)$ is the free term algebra on 0 and S, and any rule that is an equation valid in F would be a syntactic identity that leads to nonterminating derivations using (Σ, E).

We define a term $t \in T_{set}(\Sigma_0)$ that will satisfy the conclusion of the theorem. Let $k = max\{|lhs(e)| : e \in E\} + 1$. Define

$$t \equiv INS(S^k(0), INS(S^{2k}(0), \varnothing)).$$

Let $t_1 \in T(\Sigma)$ be the normal form of t.

Suppose for a contradiction $t_1 \in T(\Sigma_0)$. Let

$$t_1 \equiv INS(l(1), \ldots, l(d), \varnothing).$$

Notice that for $i = 1, \ldots, d$

$$l(i) = k \qquad \text{or} \qquad l(i) = 2k.$$

We now take $t_2 \in T(\Sigma_0)$ to be t_1 with $S^k(0)$ and $S^{2k}(0)$ everywhere interchanged, that is,

$$t_2 \equiv INS(3k - l(1), \ldots, 3k - l(d), \varnothing).$$

We know that

$$F \models t_1 = t_2$$

and so

$$E \vdash t_1 = t_2$$

but t_2 is not a normal form of (Σ, E) because t_1 is, and (Σ, E) is complete.

Hence, there must be a rule that reduces t_2. Let this rule be

$$l(X_1, \ldots, X_p, Z) \to r(X_1, \ldots, X_p, Z)$$

where $X_1, \ldots, X_p$ are variables of sort *nat* and Z is a variable of sort *set*. The term $l(X_1, \ldots, X_p, Z)$ matches with t_2 to make the reduction.

CLAIM 6.2.2. *$l(X, Z) = l(X_1, \ldots, X_p, Z)$ matches t_1 and hence t_1 is not a normal form.*

PROOF. We claim that the term $l(X_1, \ldots, X_p, Z)$ must have the form

$$INS\left(S^{M(1)}(X_{\pi(1)}), \ldots, S^{M(q)}(X_{\pi(q)}), Z\right),$$

where there are q locations where *INS* is applied and

$$\pi : [1, q] \to [1, p]$$

defines which variable $X_{\pi(i)}$ occurs at location i.

To see this structure, note that the alternative is that some $S^{M(i)}(0)$ occurs at location i. If this occurs in $l(X_1, \ldots, X_p, Z)$, then it also occurs in t_2, because of the matching. Now $M(i) < k$ since k bounds the left hand sides of the rules in E. But every subterm of t_2 of type *nat* is either $S^k(0)$ or $S^{2k}(0)$, which is a contradiction.

Now suppose that σ is a substitution such that $\sigma(l(X, Z))$ is a subterm of t_2. Let

$$\sigma(X_i) = S^{m(i)}(0)$$

$$\sigma(Z) = INS(S^{r(1)}(0), \ldots, S^{r(e)}(0), \varnothing)$$

for $i = 1, \ldots, p$. Note that since $\sigma(Z)$ is a subterm of t_2

$$r(i) = k \quad \text{or} \quad r(i) = 2k$$

for $i = 1, \ldots, e$, by the construction of t_2 from t_1.

We will construct a substitution σ' such that $\sigma'(l(X, Z))$ is a subterm of t_1. Define $\lambda: [1, p] \to \mathbf{N}$ by

$$\begin{aligned} \lambda(i) &= m(i) - k \quad \text{if} \quad k < m(i) \leq 2k \\ &= m(i) + k \quad \text{otherwise} \end{aligned}$$

and set

$$\sigma'(X_i) = S^{\lambda(i)}(0)$$

$$\sigma'(Z) = INS(S^{3k-r(1)}(0), \ldots, S^{3k-r(e)}(0), \varnothing).$$

Thus, in the case of $\sigma'(Z)$, the $S^k(0)$ and $S^{2k}(0)$ in $\sigma(Z)$ are interchanged.

We know that $\sigma(l(X, Z))$ is a subterm of t_2. Then we can compare these terms starting from the innermost subterm Z. This leads to the following identities between the powers of the numerals: first by matching the subterm $\sigma(Z)$,

$$\begin{aligned} r(e) &= 3k - l(d) \\ &\vdots \\ r(1) &= 3k - l(d - e + 1) \end{aligned} \qquad (A_1)$$

and next by matching the remainder of $\sigma(l(X, Z))$,

$$\begin{aligned} M(q) + m(\pi(q)) &= 3k - l(d - e) \\ &\vdots \\ M(1) + m(\pi(1)) &= 3k - l(d - e - q + 1). \end{aligned} \qquad (B_1)$$

To prove the claim, we must prove that $\sigma'(l(X, Z))$ matches t_1. This occurs if, and only if, the following identities hold

$$\begin{aligned} 3k - r(e) &= l(d) \\ &\vdots \\ 3k - r(1) &= l(d - e + 1) \end{aligned} \qquad (A_2)$$

and

$$\begin{aligned} M(q) + \lambda(m(\pi(q))) &= l(d - e) \\ &\vdots \\ M(q) + \lambda(m(\pi(1))) &= l(d - e - q + 1). \end{aligned} \qquad (B_2)$$

The set of identities A_2 are immediately derivable from A_1. Consider the first identity of B_2. We will show it is derivable from the first identity of B_1, which is

$$M(q) + m(\pi(q)) = 3k - l(d - e).$$

There are two cases:

$$l(d - e) = k \quad \text{or} \quad l(d - e) = 2k$$

(i) $l(d - e) = k$. Now $M(q) + m(\pi(q)) = 2k$. Since $M(q) < k$ (k bound LHS of equations) we know $2k \geq m(\pi(q)) > k$. Thus, by definition of λ,

$$\lambda(m(\pi(q))) = m(\pi(q)) - k.$$

So substituting,

$$\begin{aligned} M(q) + \lambda(m(\pi(q))) &= M(q) + m(\pi(q)) - k \\ &= 2k - k \\ &= k \\ &= l(d - e). \end{aligned}$$

(ii) $l(d - e) = 2k$. Now $M(q) + m(\pi(q)) = k$. Again $M(q) < k$ implies $m(\pi(q)) < k$. So

$$\lambda(m(\pi(q))) = m(\pi(q)) + k.$$

Substituting

$$\begin{aligned} M(q) + \lambda(m(\pi(q))) &= M(q) + m(\pi(q)) + k \\ &= k + k \\ &= 2k \\ &= l(d - e). \end{aligned}$$

The other identities of B_2 follow similarly from corresponding identities of B_1. Q.E.D.

7. *Results on Semicomputable Algebras*

We now consider the situation for semicomputable algebras.

7.1. Recursive and RE Specifications of Semicomputable Algebras

THEOREM 7.1.1. *Let A be a minimal Σ algebra. Then A is semicomputable if, and only if, A possesses a recursively enumerable complete TRS specification (Σ, E).*

PROOF. If A has a recursively enumerable equational specification, then it is semicomputable (Lemma 4.2.3). The argument for the converse follows that of Theorem 4.3.1. Using similar notation, define

$$E_A = \{t_1 = t_2 | \gamma^{-1}(t_1) > \gamma^{-1}(t_2) \text{ and } t_1 \equiv_A t_2\}.$$

Clearly, E_A is recursively enumerable since A is semicomputable.

Let $T_\gamma = im(nf_\gamma)$ be a transversal for $\equiv_A$. We claim T_γ is the set of normal forms for the TRS (Σ, E_A). First, the terms in T_γ do not reduce, since neither they nor their subterms occur on the LHS since they are minimal in size (with respect to γ). Any term $t \notin T_\gamma$ can be reduced since $t = nf_\gamma(t) \in E_A$.

Thus, $NF(\Sigma, E_A) = T_\gamma$.

The TRS (Σ, E_A) is strongly terminating because the monotonic property of γ implies that any reduction $t \to t'$ entails $\gamma^{-1}(t) > \gamma^{-1}(t')$.

Ground confluence follows immediately from the fact that every closed term t can reduce in one step to its normal form $nf_\gamma(t)$. In order to prove confluence one may use the critical pairs theorem of Knuth and Bendix. Indeed all critical pairs of (Σ, E_A) are convergent and (Σ, E_A) is terminating, because all reductions lead to shorter terms. See the survey [Klop 1992, 2.4.14].

Since (Σ, E_A) is strongly terminating and confluent, it is complete.

Finally, since $NF(\Sigma, E_A) = T_\gamma$ is a transversal for $\equiv_A$ and trivially $A \in ALG(\Sigma, E_A)$ we deduce that

$$A \cong T(\Sigma, E_A)$$

by Lemma 3.4.3. Q.E.D.

THEOREM 7.1.2. *There is a semicomputable single sorted Σ algebra A that does not possess a recursive complete TRS specification.*

PROOF. By Theorem 6.5 in Bergstra and Tucker [1987], there is a semicomputable algebra A that does not possess a recursive equational specification under initial algebra semantics; this A does not possess a recursive equational complete TRS specification (Lemma 4.2.3). Q.E.D.

7.2. ORTHOGONAL SPECIFICATIONS

THEOREM 7.2.1. *There is a semicomputable single sorted Σ algebra A that does not possess a recursively enumerable, complete, orthogonal TRS specification (Σ, E).*

PROOF. Let $\{0,1\}^*$ be the set of finite sequences over $\{0,1\}$ and $\epsilon \in \{0,1\}^*$ be the empty sequence. Let $\perp$ be a new symbol.

Consider the algebra

$$A = (\{0,1\}^* \cup \{\perp\}, \epsilon, \perp, f, g)$$

whose unary operations are defined by

$$f(\perp) = \perp$$
$$f(x) = x \cdot 0 \quad \text{if} \quad x \neq \perp$$

and

$$g(\perp) = \perp$$
$$g(x) = x \cdot 1 \quad \text{if} \quad x \neq \perp .$$

Let $Z \subset \{0,1\}^*$ and assume that

(i) Z is suffix-closed, that is, for any $x, y \in \{0,1\}^*$

$$x \in Z \text{ implies } x \cdot y \in Z;$$

(ii) $Z^c = \{0,1\}^* - Z$ is infinite.

Note that from (i) and (ii) it follows that $\epsilon \notin Z$. If Z is nonempty, then Z is infinite.

We define a congruence $\equiv_Z$ on A by

$$x \equiv_Z y \quad \text{if, and only if,} \quad x, y \in Z \cup \{\perp\} \text{ or } x = y.$$

Let $A_Z = A/\equiv_Z$.

LEMMA 7.2.2. *Z is recursively enumerable if, and only if A_Z is semicomputable.*

PROOF. Immediate. Q.E.D.

We want to choose Z satisfying (i) and (ii) such that

(iii) Z is recursively enumerable
(iv) If $V \subset \{0,1\}^*$ is infinite and recursively enumerable, then $V \cap Z \neq \emptyset$.

LEMMA 7.2.3. *A Z satisfying (i)–(iv) exists.*

PROOF. Let $\gamma: \mathbf{N} \to \{0,1\}^*$ be a bijective enumeration of finite strings. Let $\langle W_e, e \in \mathbf{N}\rangle$ be an enumeration of the recursively enumerable subsets of $\mathbf{N}$ and take $\gamma(W_e)$, $e \in \mathbf{N}$ to be an enumeration of the recursively enumerable subsets of $\{0,1\}^*$. Let

$$Z = \cup_n Z_n,$$

where Z_n is defined uniformly in n to be a semicomputable set such that

$$\gamma(W_n) \text{ infinite implies } \gamma(W_n) \cap Z_n \neq \emptyset.$$

The algorithm for Z_n is defined as follows: let

$$c(n) = (\textit{least } k)[k \in W_n \textit{ and } \|\gamma(k)\| > 2n + 1]$$

and set

$$x \in Z_n \Leftrightarrow x \text{ extends the string } \gamma(c(n))$$

for $x \in \{0,1\}^*$.

First note that Z is infinite and recursively enumerable. Next note, by definition of when $c(n)$ is defined, $Z \cap \gamma(W_n) \neq \emptyset$ for all n such that W_n is infinite.

Finally, we show that Z^c is infinite. Recall the measure μ on subsets of $\{0,1\}^*$ defined from

$$\mu(B_x) = \frac{1}{2^{|x|}},$$

where $B_x = \{y \in \{0,1\}^* \mid y \textit{ extends } x\}$.

Then

$$\begin{aligned}\mu(Z^c) &= 1 - \sum_{n=0}^{\infty} \mu(Z_n) \\ &= 1 - \sum_{n=0}^{\infty} \frac{1}{2^{2n+1}} \\ &> 0.\end{aligned}$$

Thus, Z^c is infinite. Q.E.D.

Let Σ be the signature of A:

sort	s
constants	$\epsilon, \perp : s$
operations	$F: s \to s$
	$G: s \to s$

LEMMA 7.2.4. *The algebra A_Z does not possess a recursively enumerable orthogonal TRS equational specification which is strongly terminating for ground terms.*

PROOF. Suppose for a contradiction that such a specification (Σ, E) exists.

First note that E is infinite for if E is finite then (Σ, E) is a finite complete TRS specification of A_Z and this implies A_Z is computable, which contradicts the choice of Z.

Now let $t_\perp$ be a normal form for $\perp$.

We divide the equations into two sets E_1 and E_2: let $E = E_1 \cup E_2$ where the equations of E_2 are precisely those of E whose LHS have the following forms:

$$\perp\ ,\ F(t_\perp) \text{ and } G(t_\perp).$$

Notice that $|E_2| \leq 3$ because E is orthogonal and its equations may not overlap.

Hence, we know that E_1 is infinite. We will construct from E_1 an infinite recursively enumerable subset X of Z^c. This will contradict clause (iv) in the definition of Z.

We define for $s \in \{0,1\}^*$ a term $\tau_s(X) \in T(\Sigma, X)$ in the single variable X as follows:

$$\begin{aligned} \tau_\epsilon(X) &= X \\ \tau_{r\cdot 0}(X) &= F(\tau_r(X)) \qquad \text{if} \quad s = r\cdot 0 \\ \tau_{r\cdot 1}(X) &= G(\tau_r(X)) \qquad \text{if} \quad s = r\cdot 1 \end{aligned}$$

Now each equation $e \in E_1$ has a LHS of one of the following forms

(1) $F(\tau_s(X))$	(3) $F(\tau_s(\epsilon))$	(5) $F(\tau_s(\perp))$
(2) $G(\tau_s(X))$	(4) $G(\tau_s(\epsilon))$	(6) $G(\tau_s(\perp))$

for some s. In the case of closed terms (3)–(6) we note that $\tau_s(\epsilon)$ and $\tau_s(\perp)$ are not syntactically identical to $\perp$. To see this notice that $\tau_s(\epsilon)$ or $\tau_s(\perp)$ can only be syntactically identical to $\perp$ if $s = \epsilon$; so let us assume this. There are two cases.

(i) If $\perp \equiv t_\perp$, then an equation e with LHS $F(\tau_s(\perp))$ or $G(\tau_s(\perp))$ is in E_2 and hence not in E_1.

(ii) If $\perp$ is not $t_\perp$, then suppose that $\tau_\epsilon(\perp) \equiv \perp$. Now let equation e have $F(\tau_\epsilon(\perp)) \equiv F(\perp)$ as a LHS. Then equation e shows an overlap with a rule that allows $\perp$ to reduce which must exist since $\perp$ is not syntactically equal to its normal form $t_\perp$. This contradicts the orthogonality of E.

Further, we note that ϵ cannot be a LHS of $e \in E_1$, since it is a normal form; this follows from the fact that no other term has value $[\epsilon] \in A/\equiv_Z$. Q.E.D.

CLAIM 7.2.5. *If $e \in E_1$ has a form* (1)–(6) *with index $s \in \{0,1\}^*$, then $s \notin Z$.*

Assuming this claim, we can complete the argument of Lemma 7.2.4 as follows: Let

$$X = \{s \in \{0,1\}^* : s \textit{ is an index for an equation } e \in E_1 \textit{ of form } (1)\text{–}(6)\}.$$

Since E_1 is infinite and recursively enumerable, we have that X is infinite and recursively enumerable. By claim, $X \subseteq Z^c$. This is the desired contradiction of clause (iv).

PROOF OF CLAIM 7.2.5. We argue by contradiction: Suppose $e \in E_1$. There are six cases, depending on the form of $e \in E_1$.

First, we show that forms (5) and (6) cannot arise with index s in Z; we consider case (5) only, as (6) is similar. Observe that

$$A_Z \models \tau_s(\perp) = \perp \qquad \text{and} \qquad A_Z \models \perp = t_\perp .$$

Hence, $\tau_s(\perp)$ and $t_\perp$ have a common reduct $t_\perp$; a rewriting is necessary because $\tau_s(\perp)$ is not $\perp$. The equation that performs this rewrite must overlap with the equation e of type (5). Thus, e does not exist.

Now, we show that the forms (3) and (4) cannot arise with index s in Z; we consider case (3) only, as (4) is similar. Since $s \in Z$

$$A_Z \models \tau_s(\epsilon) = \perp \qquad \text{and} \qquad A_Z \models \perp = t_\perp .$$

Hence, $\tau_s(\epsilon)$ and $t_\perp$ have a common reduct. Again the existence of a rewrite implies that an equation overlaps e and contradicts orthogonality. So e with index $s \in Z$ does not exist.

Finally, we consider forms (1) and (2); (2) is similar to (1). If $s \in Z$, then on substituting ϵ in $\tau_s(X)$ we obtain

$$A_Z \models \tau_s(\epsilon) = \perp \qquad \text{and} \qquad A_Z \models \perp = t_\perp .$$

Thus there is a reduction of $\tau_s(\epsilon)$ to $t_\perp$ and, since ϵ is a normal form, this rewrite involves function symbols in $\tau_s(X)$. It follows that the first reduction of $\tau_s(\epsilon)$ is performed by an equation that overlaps with e, which is the desired contradiction. Q.E.D.

8. *Concluding Remarks*

We conclude that complete term rewriting systems are adequate for the equational specification of computable data types. Hidden functions will often be needed and the use of hidden functions in normal forms cannot, in general, be avoided. An open question is this: Under what conditions can a computable data type be specified using a complete equational term rewriting system involving hidden functions and whose normal forms do not use the hidden functions?

We think that the development of many sorted term rewriting is an interesting and useful project. Arising in our work is the topic of the relationship between term rewriting systems over many sorted signatures with common subsignatures. Perhaps, new many sorted modularity results for term rewriting systems could lead to an easy and satisfying proof of many sorted case of Theorem 1.1 from a proof of the single sorted case.

REFERENCES

BERGSTRA, J. A., KLOP, J. W., AND MIDDELDORP, A. 1989. *Termherschrijfsystemen* (in Dutch). Kluwer, Amsterdam, The Netherlands.

BERGSTRA, J. A., AND TUCKER, J. V. 1979. Algebraic specifications of computable and semicomputable data structures. Department of Computer Science Research Report IW 115/79. Mathematical Centre, Amsterdam, The Netherlands.

BERGSTRA, J. A., AND TUCKER, J. V. 1980a. A natural data type with a finite equational final semantics specification, but no effective equational initial specification, *Bull. EATCS* 11, 23–33.

BERGSTRA, J. A., AND TUCKER, J. V. 1980b. A characterisation of computable data types by means of a finite equational specification method, in *Automata, languages and programming (ICALP), 7th Colloquium, Noordwijkerhout*, J. W. de Bakker and J. van Leeuwen, eds. Lecture Notes in Computer Science, vol. 81. Springer-Verlag, Berlin, pp. 76–90.

BERGSTRA, J. A., AND TUCKER, J. V. 1983a. Initial and final algebra semantics for data type specifications: Two characterisation theorems, *SIAM J. Comput. 12*, 366–387.

BERGSTRA, J. A., AND TUCKER, J. V. 1983b. The completeness of the algebraic specification methods for computable data types. *Inf. Cont. 54*, 186–200.

BERGSTRA, J. A., AND TUCKER, J. V. 1987. Algebraic specifications of computable and semicomputable data types. *Theoret. Comput. Sci. 50*, 137–181.

CUTLAND, N. 1980. *Computability*. Cambridge University Press, Cambridge, England.

DAUCHET, M. 1989. Simulation of Turing machines by a left linear rewrite rule. In N. Dershowitz, Ed., *RTA 89*. Lecture Notes in Computer Science, vol. 355. Springer-Verlag, Berlin, Germany, pp. 109–120.

DERSHOWITZ, N. 1987. Termination of rewriting. *J. Symb. Comput. 3*, 69–116.

EHRIG, H., AND MAHR, B. 1985. Fundamentals of algebraic specifications 1 (EATCS Monographs). In *Theoret. Comput. Sci. 6*. Springer-Verlag, Berlin.

ERSHOV, Y. 1977. Theorie der Numerierungen III. *Z. Math Logic 23*, 289–371.

GOGUEN, J. A., THATCHER, J. W., AND WAGNER, E. G. 1978. An initial algebra approach to the specification, correctness and implementation of abstract data types. In *Current Trends in Programming Methodology. IV. Data Structuring*. R. T. Yeh, ed. Prentice-Hall, Engelwood Cliffs, New Jersey, pp. 80–149.

KAMIN, S. 1979. Some definitions for algebraic data type specifications, *ACM SIGPLAN Notices 14*, 3, 28–37.

KLOP, J. W. 1992. Term rewriting systems. In *Handbook of Logic in Computer Science*. Vol. 2. S. Abramsky, D. Gabbay, and T. S. E. Maibaum, eds. Oxford University Press, Oxford, England, pp. 1–116.

MAL'CEV, A. I. 1961/1971. Constructive algebras, I., *Russian Mathematical Surveys, 16* (1961) 77–129. Also in: *The Metamathematics of Algebraic Systems. Collected Papers 1936–1967*, B. F. Wells, III, ed., North Holland, Amsterdam, The Netherlands, pp. 148–212.

MEINKE, K., AND TUCKER, J. V. 1992. Universal algebra. In *Handbook of Logic in Computer Science*. Vol. 1. S. Abramsky, D. Gabbay and T. S. E. Maibaum, Eds., Oxford University Press, Oxford, England, pp. 189–411.

MESEGUER, J., AND GOGUEN, J. A. 1985. Initiality, induction and computability. In *Algebraic Methods in Semantics*. M. Nivat and J. Reynolds, eds. Cambridge University Press, Cambridge, England, pp. 459–541.

MESEGUER, J., MOSS, L., AND GOGUEN, J. A. 1992. Final algebras, cosemicomputable algebras, and degrees of unsolvability. *Theoret. Comput. Sci. 100*, 267–302.

RABIN, M. O. 1960. Computable algebra, general theory and the theory of computable fields. *Trans. Amer. Math. Soc. 95*, 341–360.

STOLTENBERG-HANSEN, V., AND TUCKER, J. V. 1995. Effective algebra. In *Handbook of Logic in Computer Science*, Vol. 4. S. Abramsky, D. Gabbay, and T. S. E. Maibaum, eds. Oxford University Press, Oxford, England.

WECHLER, W., 1992. *Universal algebra for Computer Scientists*. Springer-Verlag, Berlin, Germany.

WIRSING, M. 1990. Algebraic specifications. In *Handbook of Theoretical Computer Science. Vol. B: Formal models and semantics*. J. van Leeuwen, Ed. North-Holland, Amsterdam, The Netherlands, pp. 675–788.

RECEIVED AUGUST 1992; REVISED NOVEMBER 1994; ACCEPTED JUNE 1995