The data type variety of stack algebras

J A Bergstra
Programming Research Group, University of Amsterdam,
Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands

J V Tucker
Department of Computer Science, University College of Swansea,
Singleton Park, Swansea, SA2 8PP, Wales

ABSTRACT
We define and study the class of all stack algebras as the class of all minimal algberas in a variety defined by an infinite recursively enumerable set of equations. Among a number of results, we show that the initial model of the variety is computable, that its equational theory is decidable, but that its equational deduction problem is undecidable. We show that it cannot be finitely axiomatised by equations, but it can be axiomatised by equations with a hidden sort and functions. This class of all stack algebras, together with its specifications, can be used to survey the many models in the literature on stacks in a systematic way, and hence give the study of the stack some mathematical coherence.

KEY WORDS & PHRASES :
stacks, equational specifications with hidden functions, complete term rewriting systems, many sorted algebras, computable algebras, equational theories, equational deduction problem.

To Dirk van Dalen

1 INTRODUCTION

A stack is a structure that stores data. Its state or memory contains data and may be changed by storing a new datum or retrieving a previously stored datum. It has an empty state when it is not storing data. In formalising the stack using sets and functions, the data and stack states are modelled by sets D and S, and the storage procedures are modelled by constant ϕ and functions push: $D \times S \to S$, top: $S \to D$ and pop: $S \to S$. Taken together the sets and functions form a many sorted algebra which is minimal, i.e. finitely generated by constants named in its signature.

There is a large literature concerned with the semantic modelling of the stack which is scattered and appears to be theoretically incoherent and inconclusive. There is a consensus about the initialisation of the stack using ϕ and the construction of all possible stack states using push, and about the normal operation of top and pop when the stack is not empty. However, there are many ways to handle

$$\text{top}(\phi) \text{ and } \text{pop}(\phi),$$

which give rise to many stack algebras, many axiomatic specifications, and many algebraic and logical problems in the literature. In this paper we address the following questions:

What is the class of algebras that is the class of *all* stack algebras? What are its mathematical properties? Can it be equationally specified? Are all stack algebras computable? Are the logical theories of the stack well behaved? Can the class be used to organise the semantic models of the stack in the literature into a coherent mathematical theory in which, for example, different stack models can be compared by means of homomorphisms?

We give an equational specification (Σ_{st}, E_{st}) whose axioms express the consensus properties of the stack. The class $Alg_m(\Sigma_{st}, E_{st})$ of all minimal Σ_{st} algebras satisfying the equations in E_{st} is then taken to be the class of all stack algebras. The specification (Σ_{st}, E_{st}) is an infinite recursive orthogonal complete term rewriting system with computable initial model; a family of finite subsets E^k_{st}, for k = 1, 2, ... of the specification E_{st} represent the consensus properties of the behaviour of stacks with the capacity to store k data. (A compete term rewriting system is one whose reductions or rewrites satisfy the Church-Rosser property and are strongly terminating.)

First, we examine its logical properties.

Theorem The equational theory is ω-complete. The provability problem for equations over Σ_{st}, i.e.

$$\text{given equation e over } \Sigma_{st}, \text{ is } E_{st} \mid\!- \text{ e?},$$

is decidable. However, the deduction problem for equations over Σ_{st}, i.e.

$$\text{given equations } e_1, \ldots, e_k, \text{ e over } \Sigma_{st}, \text{ is } E_{st} \cup \{e_1, \ldots, e_k\} \mid\!- \text{ e?}$$

is undecidable.

A consequence is that the first order theory of the stack is undecidable. Then we examine the axiomatic specification of the class of stack algebras.

Theorem The class $Alg_m(\Sigma_{st}, E_{st})$ and the initial model $T(\Sigma_{st}, E_{st})$ do not possess a finite equational specification (Σ_{st}, E).

There is a finite equational specification (Σ_{ast}, E_{ast}) with a hidden sort and two hidden operations that specifies the class $Alg_m(\Sigma_{st}, E_{st})$ of all stack algebras and its initial algebra $T(\Sigma_{st}, E_{st})$. Furthermore, the specification is an orthogonal complete term rewriting system.

We give a finite conditional equation specification (Σ_{st}, C_{st}) of the initial algebra $T(\Sigma_{st},E_{st})$ but which fails to specify the class $Alg_m(\Sigma_{st}, E_{st})$ of all stack algebras.

To study the stack, we must consider the data type to be modelled by a *class* of non-isomorphic minimal algebras, and develop methods of specifying such classes using equations and conditional equations. In particular, of primary interest here is the class $Alg_m(\Sigma, E)$ of all minimal algebras in the variety $Alg(\Sigma, E)$, which we call an *MA variety*, or *data type variety*, in the context of data type theory. The specification of the class and the initial algebra is a new object worthy of general study. In algebraic data type theory, data types are specified by means of equational or conditional equational axioms, most commonly up to isomorphism using initial algebra semantics for the specification. The relationship between the specification of the initial algebra of a class and that of the whole class can be complicated.

The basic concepts are carefully defined in Section 2, on equational specifications and term rewriting systems. In Section 3, we introduce the specification (Σ_{st}, E_{st}) and the class $Alg_m(\Sigma_{st}, E_{st})$ and examine its term rewriting properties. In Sections 4-7 we prove the theorems: results about the equational theory and deduction problem are in 4 and 5, and results about finite specifications are in 6 and 7. In section 8 we summarise the open problems.

This paper is associated with our attempts to survey and make a coherent theory of the stack, beginning in Bergstra and Tucker [1988,1990] and currently under revision. It is also relevant 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 especially Bergstra and Tucker [1987] and our other references, and the survey Meseguer and Goguen [1985]).

The reader is assumed well versed in universal algebra (see Wechler [1991] and Meinke and Tucker [1992]) and initial algebra specification methods (see Goguen, Thatcher and Wagner [1978], Ehrig and Mahr [1985] and Wirsing [1990]). An elementary knowledge of logic and the theory of the recursive functions is necessary for the computability results (see Enderton [1972] and Cutland [1980], for instance).

2. ALGEBRAS AND EQUATIONAL SPECIFICATIONS

We summarise the usual concepts we need. Appropriate references are: Meinke and Tucker [1992] and Wechler [1992], for basic universal algebra; Ehrig and Mahr [1985] and Wirsing [1990], for algebraic specifications; and Klop [1992], for term rewriting.

2.1 Preliminaries on algebras and specifications

2.1.1 Signatures and Algebras A *signature* Σ consists of a non-empty set S of *sorts*, and a family of sets of constant *symbols* and *function symbols*.We assume all signatures are finite.

Let Σ_1 and Σ_2 be signatures. If Σ_1 is a *subsignature* of Σ_2 then we write $\Sigma_1 \subseteq \Sigma_2$.

Let Σ be a signature. A Σ–*algebra* A consists of a family of non-empty sets, called the *carriers* of A, that interpret the sorts, together with families of elements and families of functions that interpret the symbols in the signature. If $\sigma \in \Sigma$ is of type $w(1) \times ... \times w(k) \to s$ then it is interpreted by function $\sigma_A : A_{w(1)} \times ... \times A_{w(k)} \to A_s$.

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

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

2.1.2 Term Algebras and Equations Let X be a set the elements of which are called *variable names*. We define the Σ term algebra $T(\Sigma, X)$ of terms over Σ in variables of X in the usual way. We write $T(\Sigma)$ when $X = \varnothing$; the terms of $T(\Sigma)$ are called *closed.*

2.1.3 Lemma Let ψ: $T(\Sigma)/\equiv_1 \to T(\Sigma)/\equiv_2$ be a homomorphism. Then $\psi([t]_1) = [t]_2$ for all $t \in T(\Sigma)$. Thus,

$$T(\Sigma)/\equiv_1 \cong T(\Sigma)/\equiv_2 \iff T(\Sigma)/\equiv_1 = T(\Sigma)/\equiv_2$$

Let Σ be a signature and X a set of variable names. An equation is an expression e of the form

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

where $t(X), t'(X) \in T(\Sigma, X)$ are of the same sort. Let $Eqn(\Sigma, X)$ be the set of equations.

We say that two equations e and e' of the same sort are *α-equivalent* if there is a permutation of the set X of variables that transforms e to e'. We write $e \equiv_\alpha e'$ if e and e' are α-equivalent. The *α-closure* of a set E of equations is the set $\acute{E} = \{e' : e' \equiv_\alpha e$ for some $e \in E\}$; we say that E is *α-closed* if $E = \acute{E}$.

Given a state $\sigma : X \to A$ of the variables, we define the *term evaluation map* $eval_{A,\sigma} : T(\Sigma, X) \to A$ in the usual way, by induction on terms; the map is a Σ homomorphism. From this we can define *validity* or *satisfaction* $A \models t(X) = t'(X)$.

2.1.4 Varieties and MA varieties Let $Alg(\Sigma, E)$ denote the class of all Σ algebras satisfying the equations in E; such a class is called a *variety*.

Let $Alg_m(\Sigma, E)$ denote the class of all Σ minimal algebras satisfying the equations in E. We call such a class a *MA variety*.

2.1.5 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. We can formalise the use of equations in derivations of terms in $T(\Sigma, X)$ 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'.

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

The term rewriting system (Σ, E) is *complete* if the reduction system $\to_E$ on $T(\Sigma,X)$ 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 $\to_E$.

2.1.6 Lemma $T(\Sigma, E) = T(\Sigma)/\equiv_E$ is the initial algebra of $Alg(\Sigma, E)$.

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

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

2.1.7 Lemma If (Σ, E) is an orthogonal TRS then it is Church-Rosser.

2.1.8 Algebraic Specifications Let A be an algebra of signature Σ. Then A is said to have a *finite equational specification* (Σ, E) if E is a finite set of equations over Σ such that $T(\Sigma, E) \cong A$.

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

$$T(\Sigma', E')|_{\Sigma} \cong A.$$

2.1.9 Lemma Let A be a Σ algebra and (Σ, E) an initial algebra specification of A. Let E' be a set of equations over Σ that are valid in A. Then $(\Sigma, E \cup E')$ is an initial algebra specification of A.

2.1.10 Lemma Let E, E_1, and E_2 be sets of equations over Σ. Suppose that

$$E \cup E_1 \vdash E_2 \text{ and } E \cup E_2 \vdash E_1.$$

Then $T(\Sigma, E \cup E_1) \cong T(\Sigma, E \cup E_2)$.

2.1.11 ω-completeness An equational specification (Σ, E) is *ω-complete* if for any equation e with variables from X,

$$T(\Sigma, E) \models e \text{ if, and only if, } Alg(\Sigma, E) \models e.$$

2.2 Specification of Classes

Let K be any class of minimal Σ algebras.

2.2.1 Definition The class K has an *equational specification* (Σ, E) if E is a set of equations over Σ and

$$K = Alg_m(\Sigma, E),$$

i.e. the class K is a MA variety. The concept of a *conditional equational specification* is defined similarly.

2.2.2 Lemma Let K have an initial algebra I and be closed under homomorphisms. Then the following are equivalent:

(i) (Σ, E) is an equational specification for I under initial algebra semantics;

(ii) (Σ, E) is an equational specification for K.

Proof Suppose (i). Then $K = Hom(I)$ implies $K = Hom(T(\Sigma, E))$ and, since $Alg_m(\Sigma, E) = Hom(T(\Sigma, E))$,

$$K = Alg_m(\Sigma, E).$$

Conversely, if $K = Alg_m(\Sigma, E)$ then $I \cong T(\Sigma, E)$, by the uniqueness of initial objects.

The problem we address has the following form: we have a class $K = Alg_m(\Sigma, E)$ for which we seek an alternative specification. In these circumstances, we deduce the following from the lemma:

2.2.3 Lemma Let (Σ, E) and (Σ, E') be equational specifications with common signature Σ. Then

$$Alg_m(\Sigma, E) = Alg_m(\Sigma, E') \quad \Leftrightarrow \quad T(\Sigma, E) \cong T(\Sigma, E').$$

2.2.4 Definition The class K has an *equational specification* (Σ', E') *with hidden sorts and functions* if $\Sigma \subseteq \Sigma$, E' is a set of equations over Σ', and

$$K = Alg_m(\Sigma', E')|_{\Sigma}.$$

When $K = Alg_m(\Sigma, E)$, we have the following complement to Lemma 2.2.3.

2.2.5 Lemma Let (Σ, E) and (Σ', E') be equational specifications and $\Sigma \subseteq \Sigma'$. Then

$$Alg_m(\Sigma', E')|_{\Sigma} = Alg_m(\Sigma, E) \quad \text{implies} \quad T(\Sigma', E')|_{\Sigma} \cong T(\Sigma, E).$$

Proof Using the premise, we show that $T(\Sigma', E')|_{\Sigma}$ is initial in $Alg_m(\Sigma', E')|_{\Sigma}$. By the uniqueness up to isomorphism and the premise we deduce the isomorphism of the algebras.

Let $A \in Alg_m(\Sigma', E')|_{\Sigma}$. Choose any $B \in Alg_m(\Sigma', E')$ such that

$$A = B|_{\Sigma}.$$

By initiality, there is a Σ' homomorphism $\phi : T(\Sigma', E') \to B$, and hence ϕ is a Σ homomorphism $T(\Sigma', E')|_{\Sigma} \to B|_{\Sigma}$. However, we must check ϕ is unique: since the classes are equal, $A = B|_{\Sigma}$ is Σ minimal. Thus ϕ is unique.

QED

2.2.6 Lemma Let $\Sigma \subseteq \Sigma'$ and B be a minimal Σ' algebra. The following are equivalent:

(i) B is a minimal Σ algebra.

(ii) For each $t' \in T(\Sigma')$ there is a $t \in T(\Sigma)$ such that

$$B \models t' = t.$$

If E is a set of equations or conditional equations over Σ then

$$B \models E \quad \text{implies} \quad B|_{\Sigma} \models E.$$

2.3 Computability

The definitions of a computable and semicomputable algebra are taken from Bergstra and Tucker [1987] and derive from Rabin [1960] and Mal'cev [1961], independent papers devoted to founding a general theory of computable algebraic systems and their computable morphisms.

2.3.1 Computable algebras Let Σ be an S sorted signature. A many sorted algebra A of signature Σ is said to be *effective* if for each sort $s \in S$ there exists a recursive set Ω_s of natural numbers and a surjection $\alpha_s : \Omega_s \to A_s$ such that for each operation symbol $\sigma \in \Sigma$ and corresponding operation $\sigma_A : A_{w(1)} \times \ldots \times A_{w(k)} \to A_s$ of A, there corresponds a recursive *tracking function* $\bar{\sigma}: \Omega_{w(1)} \times \ldots \times \Omega_{w(k)} \to \Omega_s$ which commutes the following diagram,

$$\begin{tikzcd}
A_{w(1)} \times \ldots \times A_{w(k)} \arrow[r, "\sigma"] & A_s \\
\Omega_{w(1)} \times \ldots \times \Omega_{w(k)} \arrow[r, "\bar{\sigma}"] \arrow[u, "\alpha^w"] & \Omega_s \arrow[u, "\alpha"']
\end{tikzcd}$$

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

Taken together the sets of numbers and the tracking functions form an algebra Ω and $\alpha : \Omega \to A$ is an epimorphism. We refer to α as an *effective numbering* or *coordinate system.*

Consider the S sorted relation $\equiv_\alpha$ on the number algebra Ω, defined for $x,y \in \Omega_s$ by

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

The relation is a Σ congruence on Ω.

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

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

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

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

is recursive or r.e.

2.3.2 Lemma 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.

2.3.3 Lemma 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.

2.3.4 Computable Term Algebras The algebras $T(\Sigma,X)$ are computable under any standard gödel numbering γ. If $A \cong T(\Sigma)/\equiv_A$ then A is computable or semicomputable if, and only if, $\equiv_A$ is computable or semicomputable on $T(\Sigma)$, respectively.

From γ we can construct a computable numbering γ_e of the set $Eqn(\Sigma,X)$ of equations in an obvious way.

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

We note that if E is recursively enumerable then the closure É is recursively enumerable.

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

2.3.5 Lemma Let (Σ,E) be a finite equational term rewriting system specification. Then

(i) the reduction system $\rightarrow_E$ and the congruence $\equiv_E$ are semicomputable; and

(ii) the set of normal forms $NF(\Sigma,E)$ is computable.

In particular, $T(\Sigma,E)$ is a semicomputable algebra. If (Σ,E) is recusive 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 $\equiv_E$.

The *rule application problem* is the following: given any term t, is it the case that there is a LHS of a rule in E which is α-equivalent to t?

The rule application problem is equivalent to the decidability of the set of normal forms.

2.3.6 Proposition Let (Σ,E) be a r.e. equational term rewriting system specification which is complete. Suppose that the rule application problem is decidable. Then $T(\Sigma,E)$ is a computable algebra.

2.3.7 Some logical decision problems Let (Σ,E) be an equational theory. The *provability problem* for equations over Σ is the following: given any equation e over Σ, is $E \vdash e$?

The *deduction problem* for equations over Σ is the following: given equations $e_1, \ldots, e_k$, e over Σ, is $E \cup \{e_1, \ldots, e_k\} \vdash e$?

2.4 Application in theory of data types

Consider the concept of an MA variety and its use in data type theory. Suppose that a concrete implementation of a data type is modelled by a many sorted minimal algebra A. Then a data type is modelled 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 modelled as a class K of minimal algebras closed under isomorphism, i.e if $A \in K$ and $B \cong A$ then $B \in K$.

An abstract data type K is equationally specified by (Σ,E) if $K = Alg_m(\Sigma,E)$, i.e. is an MA variety.

The results of 2.2 concern the relationship between the specification of the class K and that of its initial algebra. In particular, note that Lemmas 2.2.2 and 2.2.3 state that the notions are equivalent, if the specification does not involve hidden functions. However, our work on the stack shows that the specification of a class by hidden sorts and functions is a necessary activity, and is one that may be more difficult than that of the specification of its initial algebra (Lemma 2.2.5).

3 STACK ALGEBRAS

A stack is modelled by a minimal algebra of signature Σ_{st} satisfying a set E_{st} of equations, defined as follows:

signature	Σ_{st}		
sorts	data, edata, stack		
constants	$d_1, \ldots, d_n$:		$\rightarrow$ data
	$\varnothing$	:	$\rightarrow$ stack
operations	i	: data	$\rightarrow$ edata
	push	: edata $\times$ stack	$\rightarrow$ stack
	top	: stack	$\rightarrow$ edata
	pop	: stack	$\rightarrow$ stack

We assume there are at least $n \geq 2$ constants of sort data, and ignore operations involving data only. The sort edata is for *extended data* which allows the data to be augmented by error elements, unspecified elements etc. The operation i maps data into edata. *The stack is conceived as a stack of extended data.*

To define the set E_{st} of equations over Σ_{st} that we assume true of all stacks we must avoid specifying the behaviour of the operations in "debatable" circumstances. First, we define a sequence of standard stack polynomials.

Let $a_1, a_2, \ldots$ be a fixed list of variables of type data. The *standard stack polynomials* over these variables are inductively defined by

$$T_0 = \varnothing$$

$$T_{k+1} = push(i(a_{k+1}), T_k)$$

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

Thus, the sequence begins

$$\varnothing,\ push(i(a_1), \varnothing),\ push(i(a_2), push(i(a_1), \varnothing)), \ldots$$

These polynomials represent standard stack states containing data and no extended data. Notice that T_k represents a standard stack state containing k elements of sort data.

The equations in E_{st} are for $k \in \mathbf{N}$,

Top equations	$top(T_{k+1})$	=	$i(a_{k+1})$	$t\text{-}eqn_{k+1}$
Pop equations	$pop(T_{k+1})$	=	T_k	$p\text{-}eqn_{k+1}$

Thus E_{st} is an infinite equational specification and the sequence begins

$top(push(i(a_1), \varnothing))$	=	$i(a_1)$	$t\text{-}eqn_1$
$pop(push(i(a_1), \varnothing))$	=	$\varnothing$	$p\text{-}eqn_1$
$top(push(i(a_2), push(i(a_1), \varnothing)))$	=	$i(a_2)$	$t\text{-}eqn_2$
$pop(push(i(a_2), push(i(a_1), \varnothing)))$	=	$push(i(a_1), \varnothing)$	$p\text{-}eqn_2$
	.		
	.		

The equations leave unspecified the effects of the operations on $\varnothing$ and on any extended data. Notice that the first 2k equations for any $k = 1, 2, \ldots$ in the list specify that the top and pop operations function as expected on all standard stack states containing k elements of sort data.

3.1 Definition A *stack algebra* is a minimal algebra in the variety $Alg(\Sigma_{st}, E_{st})$; thus the data type variety of stacks is the MA-variety $Alg_m(\Sigma_{st}, E_{st})$.

3.2 Theorem The infinitary equational specification (Σ_{st}, E_{st}) is an orthogonal complete TRS. The set $N(\Sigma_{st}, E_{st})$ of normal forms of the TRS is decidable, and there is a unique total computable function n that computes normal forms. The initial algebra

$$T(\Sigma_{st}, E_{st})$$

of the variety is computable.

Proof The specification is orthogonal by inspection. Since the application of any equation in E_{st} reduces the length of terms in a sequence of rewrites, i.e. for $t, t' \in T(\Sigma_{st}, X)$

$$t \to t' \quad \text{implies} \quad |\, t \,| > |\, t' \,|,$$

the TRS is strongly terminating. Hence (Σ_{st}, E_{st}) is complete.

Given $t \in T(\Sigma_{st}, X)$, to decide whether or not

$$t \in N(\Sigma_{st}, E_{st})$$

it is sufficient to observe that t can be reduced only by means of equations from E_{st} whose LHS is smaller than $|\, t \,|$. Thus, we can compare t with the first $2\,|\, t \,|$ equations in the list and decide if a rewrite is possible. The existence and uniqueness of the total function n that computes normal forms follows from the completeness of the specification; its computability follows from the decidability of normal forms.

Finally, $T(\Sigma_{st}, E_{st}) = T(\Sigma_{st})/\equiv_{E_{st}}$. For $t, t' \in T(\Sigma_{st})$,

$$\begin{aligned} t \equiv_{E_{st}} t' \quad &\Leftrightarrow \quad E_{st} \mid\!- t = t' \\ &\Leftrightarrow \quad n(t) \equiv n(t') \end{aligned}$$

which is a decidable relation. Hence, $T(\Sigma_{st}, E_{st})$ is computable.

QED

We consider the algebras of finite and unbounded stacks. For $k \in \mathbf{N}$, we define

$$E^0_{st} = \varnothing$$

$$E^k_{st} = \{t\text{-}eqn_1, p\text{-}eqn_1, \ldots, t\text{-}eqn_k, p\text{-}eqn_k\}$$

Clearly, $E_{st} = \cup_{k \in \mathbf{N}} E^k_{st}$.

3.3 Definition A *stack algebra of depth* k is a minimal algebra in the variety $Alg(\Sigma_{st}, E^k_{st})$ for $k \in \mathbf{N}$; thus the data type variety of stacks of depth k is the MA-variety $Alg_m(\Sigma_{st}, E^k_{st})$.

3.4 Theorem The finite equational specification (Σ_{st}, E^k_{st}) is an orthogonal complete TRS. The set $N(\Sigma_{st}, E^k_{st})$ of normal forms of the TRS is decidable, and there is a unique total computable function n that computes normal forms. The initial algebra

$$T(\Sigma_{st}, E^k_{st})$$

of the variety is computable.

3.5 Theorem Let $i, j \in \mathbf{N}$ and $i \geq j$. There is an epimorphism

$$\phi_{ij} : T(\Sigma_{st}, E^j{}_{st}) \to T(\Sigma_{st}, E^i{}_{st})$$

but if $i \neq j$ then

$$T(\Sigma_{st}, E^i{}_{st}) \text{ and } T(\Sigma_{st}, E^j{}_{st}).$$

are not isomorphic. Furthermore,

$$\lim_{i \in \mathbf{N}} T(\Sigma_{st}, E^i{}_{st}) \cong T(\Sigma_{st}, E_{st})$$

and for each $i \in \mathbf{N}$

$$T(\Sigma_{st}, E^i{}_{st}) \text{ and } T(\Sigma_{st}, E_{st})$$

are not isomorphic.

Proof The ϕ_{ij} are natural homomorphisms obtained from the fact that $E^j{}_{st} \subseteq E^i{}_{st}$. Suppose $i \neq j$. The term

$$pop([d_1, \ldots , d_i / a_1, \ldots , a_i]\ T_i)$$

is closed and a normal form of $E^j{}_{st}$ but not of $E^i{}_{st}$, because it reduces under p-eqn$_i$. So the equation

$$pop([d_1, \ldots , d_i / a_1, \ldots , a_i]\ T_i) = i(d_i)$$

is not valid in $T(\Sigma_{st}, E^j{}_{st})$ but is valid in $T(\Sigma_{st}, E^i{}_{st})$.

The direct limit property follows from $E_{st} = \cup_{k \in \mathbf{N}} E^k{}_{st}$.

Finally, suppose for a contradiction that

$$T(\Sigma_{st}, E^i{}_{st}) \cong T(\Sigma_{st}, E_{st})$$

under isomorphism $\varphi_i : T(\Sigma_{st}, E_{st}) \to T(\Sigma_{st}, E^i{}_{st})$. Let

$$\phi_{i,\infty} : T(\Sigma_{st}, E^{i+1}{}_{st}) \to T(\Sigma_{st}, E_{st})$$

be an epimorphism. Thus

$$\varphi_i \circ \phi_{i,\infty} : T(\Sigma_{st}, E^{i+1}{}_{st}) \to T(\Sigma_{st}, E^i{}_{st})$$

is an epimorphism. Since $\phi_{i,i+1} : T(\Sigma_{st}, E^i{}_{st}) \to T(\Sigma_{st}, E^{i+1}{}_{st})$ is an epimorphism we have

$$T(\Sigma_{st}, E^{i+1}{}_{st}) \cong T(\Sigma_{st}, E^i{}_{st})$$

which contradicts a previous part of the lemma. QED

4. THE EQUATIONAL THEORY OF STACKS

4.1 Theorem The specification (Σ_{st}, E_{st}) is ω-complete and its equational theory is decidable.

This is proved using the following analysis of the valid equations:

4.2 Lemma Let $n = n(\Sigma_{st}, E_{st})$ compute normal forms of terms for (Σ_{st}, E_{st}). For any equation $t = t'$ over Σ_{st},

(i) $\quad T(\Sigma_{st}, E_{st}) \models t = t' \quad \Leftrightarrow \quad n(t) \equiv n(t')$;

and hence,

(ii) $\quad (\Sigma_{st}, E_{st}) \models t = t' \quad \Leftrightarrow \quad n(t) \equiv n(t')$.

The proof of Theorem 4.1 using Lemma 4.2 is straight-forward. First, for any equation $t = t'$ over Σ suppose that

$$(\Sigma_{st}, E_{st}) \models t = t'.$$

Then by combining (ii) and (i) of Lemma 4.2

$$T(\Sigma_{st}, E_{st}) \models t = t'$$

and hence (Σ_{st}, E_{st}) is ω-complete.

Secondly, for any equation $t = t'$ over Σ we ask

$$E_{st} \vdash t = t'$$

or, equivalently,

$$E_{st} \models t = t'.$$

Then by Lemma 4.2 (ii) this is equivalent with

$$n(t) \equiv n(t')$$

which is decidable because n is a computable total function (Lemma 3.2).

Notice that the equational theory $Th(T(\Sigma_{st}, E_{st}))$ of the initial model is decidable.

Proof of Lemma 4.2 First we show that statement (ii) follows from statement (i). Let $t = t'$ be an equation over Σ_{st}.

If $n(t) \equiv n(t')$ then $(\Sigma_{st}, E_{st}) \vdash t = t'$ and $(\Sigma_{st}, E_{st}) \models t = t'$.

If $E_{st} \models t = t'$ then $T(\Sigma_{st}, E_{st}) \models t = t'$ and, by (i), $n(t) \equiv n(t')$.

We now consider (i). Again, if $n(t) \equiv n(t')$ then $T(\Sigma_{st}, E_{st}) \models t = t'$. So we have to show for all Σ_{st} equations that

$$T(\Sigma_{st}, E_{st}) \models t = t' \quad \text{implies} \quad n(t) \equiv n(t').$$

Contrapositively, we will show for all Σ_{st} equations that

$$n(t) \not\equiv n(t') \quad \text{implies} \quad T(\Sigma_{st}, E_{st}) \not\models t = t'.$$

This is done by constructing a substitution σ such that $\sigma(n(t))$ and $(\sigma(n(t'))$ are closed normal forms and are distinct.

4.3 Lemma Let $t = t(a, p, x)$ be a normal form of (Σ_{st}, E_{st}). Then the following are also normal forms:

(i) $\quad [pop(\emptyset) / x]t$

(ii) $\quad [pop\ pop(\emptyset) / x]t$

(iii) $\quad [d / a]t$ for any constant d of sort data

(iv) $[i(d)]t$ for any constant d of sort data.

Proof We demonstrate (i) and leave the other cases as exercises.

Suppose for a contradiction that $[pop(\phi)/x]t$ is not a normal form. Then t has a subterm r such that $[pop(\phi)/x]r$ can be rewritten. This rewrite must use one of the two kinds of axiom in E_{st}:

(a) a top equation t-eqn$_k$ for some $k \in \mathbf{N}^+$; or

(b) a pop equation p-eqn$_k$ for some $k \in \mathbf{N}^+$.

Consider case (a). The LHS of t-eqn$_k$ is $top(T_k)$ and so $[pop(\phi)/x]r$ is a substitution instance of this term. Notice that x must occur in r for otherwise $r = [pop(\phi)/x]r$ rewrites and t is not a normal form. So $top(T_k)$ has a substitution instance containing $pop(\phi)$ as a subterm. However, this is impossible because only substitution instances by data variables and data constants for data variables are allowed for T_k.

Case (b) is similar and we omit it.

4.4 Lemma Let t and t' be normal forms of (Σ_{st}, E_{st}) such that $t \not\equiv t'$. Then there is a substitution σ, composed of substitutions of the forms (i) - (iv) in Lemma 4.3, which results in closed normal forms $\sigma(t)$ and $\sigma(t')$ of (Σ_{st}, E_{st}) and such that

$$\sigma(t) \not\equiv \sigma(t').$$

Hence because these terms are closed,

$$T(\Sigma_{st}, E_{st}) \not\models \sigma(t) = \sigma(t').$$

Proof Consider a classification of equations $t = t'$ in Table A. There are three main cases determined by the sort of the equation, and 22 subcases in total. To prove the lemma we use induction on $\max(|t|, |t'|)$.

Basis $\max(|t|, |t'|) = 0$. This has 8 subcases as follows: the 4 cases of sort data; subcase 1 of sort edata; and subcase 1 of sort stack. We check two subcases.

In case 1 of sort data $t = t'$ is $d = d'$ for constants d and d'. Since $t \not\equiv t'$ we have $d \not\equiv d'$ and the required substitution σ is the identity, which is equivalent to $[d' / a]$, a substitution of type (iii) in Lemma 4.3.

In case 1 of sort stack $t = t'$ is $x = x'$. Here the required substitution σ is

$$[pop(\varnothing), pop\ pop(\varnothing) / x, x']$$

which is a composition of substitutions of type (i) and (ii) of Lemma 4.3.

The other 6 basis subcases are equally obvious.

Induction step Suppose the lemma is true for all t, t' with $\max(|t|, |t'|) < n$. Here there are 14 cases.

Consider, for example, subcase 5 of sort edata, where t = t' is i(s) = i(s'). By assumption, s $\not\equiv$ s'. So by the induction hypothesis, there is an appropriate substitution σ_0 such that σ_0(s) $\not\equiv$ σ_0(s'). This substitution σ_0 also distinguishes the normal forms σ_0(i(s)) and σ_0(i(s')).

The other cases are similar. Notice that any substitution of the required class works with subcases 4-9 of sort stack.

	t	t'
Sort data		
1	d	d'
2	d	a
3	a	d
4	a	a'
Sort extended data		
1	p	q
2	p	i(s')
3	p	top(s')
4	i(s)	q
5	i(s)	i(s')
6	i(s)	top(s')
7	top(s)	q
8	top(s)	i(s')
9	top(s)	top(s')
Sort stack		
1	x	x'
2	x	push(s'_1, s'_2)
3	x	pop(s')
4	push(s_1, s_2)	x'
5	push(s_1, s_2)	push(s'_1, s'_2)
6	push(s_1, s_2)	pop(s')
7	pop(s)	x'
8	pop(s)	push(s'_1, s'_2)
9	pop(s)	pop(s')

Table A: Classification of Equations

5 THE UNDECIDABILITY OF THE EQUATIONAL DEDUCTION PROBLEM

By Theorem 4.1, the equational theory of (Σ_{st}, E_{st}) is decidable, i.e. it is decidable whether or not

$$E_{st} \vdash t = t'$$

given any equation $t = t'$ over Σ_{st}. Now we prove that the following form of the equational deduction problem for (Σ_{st}, E_{st}) is undecidable, i.e. whether or not

$$E_{st} \cup \{e_1, \ldots, e_k\} \vdash e$$

given any $k \in \mathbf{N}$, equations $e_1, \ldots, e_k$ and closed equation e over Σ_{st}. This follows from the stronger result:

5.1 Theorem There is a finite set E_R of equations over Σ_{st} such that it is undecidable whether or not

$$E_{st} \cup E_R \vdash e$$

given any closed equation e over Σ_{st}. This E_R can be chosen to make the deduction problem a complete r.e. set.

Proof The membership problem for r.e. sets will be reduced to the equational deduction problem for (Σ_{st}, E_{st}) as follows. Let $W \subseteq \mathbf{N}$ be any r.e. set and R any register machine program that defines it, i.e. for $n \in \mathbf{N}$

$$n \in W \iff R(n)\downarrow.$$

To R we associate a finite set E_R of equations over Σ_{st}, and give an algorithm that lists closed equations e(0), e(1), ... such that

$$R(n)\downarrow \iff E_{st} \cup E_R \vdash e(n).$$

Thus, choosing W to be an r.e., non-recursive set shows the problem to be undecidable.

Consider a programming language for the register machine. We assume that each program R involves finitely many variables

$$r_1, \ldots, r_m$$

and consists of finitely many numbered instructions

$$R \equiv I_1, \ldots, I_k.$$

Each instruction I_α has one of the following forms: let $1 \leq \alpha, \beta, \gamma \leq k$, $1 \leq i, j \leq m$,

1	zero	$r_i := 0$
2	successor	$r_i := r_i + 1$
3	predecessor	$r_i := r_i - 1$
4	transfer	$r_i := r_j$
5	jump	goto β
6	conditional jump	if $r_i = 0$ then goto β else goto γ

7 halt H

Three syntactic conditions are placed on R

(i) R contains one halt instruction which is the last instruction I_k.

(ii) R does not contain a jump instruction of type 5 or 6 that points to the first instruction I_1.

(iii) Each instruction I_α in R that has the form of an assignment of type 1-4 is followed by a goto instruction $I_{\alpha+1}$ of type 5.

Semantically, a state of a computation of R on the register machine is represented by an m + 1 tuple

$$\sigma = (n_1, \ldots, n_m, \alpha) \in \mathbf{N}^{m+1}$$

wherein for $1 \leq i \leq m$, n_i is the contents or value of register variable r_i and α is the number of the current instruction in R.

A step of a computation is denoted

$$\sigma \rightarrow_{I_\alpha} \sigma'$$

in which instruction I_α is applied to σ in the usual way.

A computation sequence for R is a sequence of states

$$\sigma_1, \sigma_2, \ldots, \sigma_N$$

such that for all $1 \leq i \leq N - 1$

$$\sigma_i \rightarrow_{I_{\alpha_i}} \sigma_{i+1}$$

where α_i is the number of the current instruction in σ_i.

A computation sequence is terminating if $\alpha_N = k$.

To compute an r.e. set $W \subseteq \mathbf{N}$ using program R we use initial state

$$\sigma_{int}(n) = (n, 0, \ldots, 0, 1)$$

and, if R terminates, we require that its final state is

$$\sigma_{fin}(n) = (n, 0, \ldots, 0, k).$$

It is easy to check that any r.e. set may be computed by a program satisfying these conventions.

We will encode each state of a register machine computation by R using a closed term over Σ_{st}; and each instruction in R as an equation or pair of equations $\varepsilon(I)$ over Σ_{st} such that each step of the computation performed by I corresponds with a rewrite using $\varepsilon(I)$.

First, define polynomials $\tau_L(x_1, \ldots, x_L)$ inductively by

$$\tau_0 \equiv pop(\varnothing)$$

$$\tau_{L+1}(x_1, \ldots, x_L) \equiv push(pop(x_{L+1}), \tau_L(x_1, \ldots, x_L)).$$

Thus, the sequence is

$$pop(\varnothing),\ push(pop(x_1), pop(\varnothing))),\ push(pop(x_2), push(pop(x_1), pop(\varnothing)))),\ \ldots$$

Fix L = m + 1 and let $\tau(x_1, \ldots, x_{m+1}) = \tau_{m+1}(x_1, \ldots, x_{m+1})$. We code a state

$$\sigma(n_1, \ldots, n_m, \alpha)$$

by the closed term

$$\tau(\sigma) = \tau(pop^{n_1}(\varnothing), \ldots, pop^{n_k}(\varnothing), pop^{\alpha}(\varnothing)).$$

Thus, in computing an r.e. set using R we represent the initial state by

$$t_{init}(n) = \tau(\sigma_{int}(n)) = \tau(pop^{n}(\varnothing), \varnothing, \ldots, \varnothing, pop(\varnothing))$$

and the final state, if any, by

$$t_{fin}(n) = \tau(\sigma_{fin}(n)) = \tau(pop^{n}(\varnothing), \varnothing, \ldots, \varnothing, pop^{k}(\varnothing)).$$

For each instruction I_α in R we form an equation (or pair of equations) $\varepsilon(I_\alpha)$, $1 \le \alpha \le k$. There are 6 cases corresponding with the first 6 types of instruction. Let $1 \le \alpha < k$, $1 \le i, j \le m$ and $1 \le \beta, \gamma \le k$,

1. I_α is $r_i := 0$

$$\tau(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_m, pop^{\alpha}(\varnothing)) = \tau(x_1, \ldots, x_{i-1}, \varnothing, x_{i+1}, \ldots, x_m, pop^{\alpha+1}(\varnothing))$$

2. I_α is $r_i := r_i + 1$

$$\tau(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_m, pop^{\alpha}(\varnothing)) = \tau(x_1, \ldots, x_{i-1}, pop(x_i), x_{i+1}, \ldots, x_m, pop^{\alpha+1}(\varnothing))$$

3. I_α is $r_i := r_i - 1$

$$\tau(x_1, \ldots, x_{i-1}, pop(x_i), x_{i+1}, \ldots, x_m, pop^{\alpha}(\varnothing)) = \tau(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_m, pop^{\alpha+1}(\varnothing))$$

$$\tau(x_1, \ldots, x_{i-1}, \varnothing, x_{i+1}, \ldots, x_m, pop^{\alpha}(\varnothing)) = \tau(x_1, \ldots, x_{i-1}, \varnothing, x_{i+1}, \ldots, x_m, pop^{\alpha+1}(\varnothing))$$

4. I_α is $r_i := r_j$

$$\tau(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_m, pop^{\alpha}(\varnothing)) = \tau(x_1, \ldots, x_{i-1}, x_j, x_{i+1}, \ldots, x_m, pop^{\alpha+1}(\varnothing))$$

5. I_α is goto β

$$\tau(x_1, \ldots, x_m, pop^{\alpha}(\varnothing)) = \tau(x_1, \ldots, x_m, pop^{\beta}(\varnothing))$$

6. I_α is if $r_i = 0$ then goto β else goto γ

$$\tau(x_1, \ldots, x_{i-1}, \varnothing, x_{i+1}, \ldots, x_m, pop^{\alpha}(\varnothing)) = \tau(x_1, \ldots, x_{i-1}, \varnothing, x_{i+1}, \ldots, x_m, pop^{\beta}(\varnothing))$$

$$\tau(x_1, \ldots, x_{i-1}, pop(x_i), x_{i+1}, \ldots, x_m, pop^{\alpha}(\emptyset))$$
$$= \tau(x_1, \ldots, x_{i-1}, pop(x_i), x_{i+1}, \ldots, x_m, pop^{\gamma}(\emptyset))$$

We examine the correspondence between computation steps and rewrites. Let σ, σ' be computation states and α be the number of the current instruction of R in σ. Then, by inspection of E_R,

5.2 Lemma $\sigma \rightarrow_{I_\alpha} \sigma' \iff \tau(\sigma) \rightarrow_{\varepsilon(I_\alpha)} \tau(\sigma')$.

By induction on the length N of a computation sequence and Lemma 5.2

$$\sigma_1, \ldots, \sigma_N$$

we can prove that

$$E_R \mid\!- \tau(\sigma_1) = \tau(\sigma_N).$$

5.3 Lemma For all $n \in \mathbf{N}$

$$R(n)\downarrow \iff (\Sigma_{st}, E_{st} \cup E_R) \mid\!- t_{int}(n) = t_{fin}(n)$$

Proof If $R(n)\downarrow$ then there is a terminating computation sequence

$$\sigma_{int}(n) = \sigma_1, \sigma_2, \ldots, \sigma_N = \sigma_{fin}(n).$$

Thus, we have

$$E_R \mid\!- \tau(\sigma_{int}(n)) = \tau(\sigma_{fin}(n))$$

as required, since $\tau(\sigma_{int}(n)) = t_{int}(n)$ and $\tau(\sigma_{fin}(n)) = t_{fin}(n)$. Notice that E_{st} is not needed.

Conversely, suppose that $(\Sigma_{st}, E_{st} \cup E_R) \mid\!- t_{int}(n) = t_{fin}(n)$. The TRS $(\Sigma_{st}, E_{st} \cup E_R)$ is orthogonal. By Lemma 2.1.7, the TRS is confluent so $t_{int}(n)$ and $t_{fin}(n)$ have a common reduct $r(n)$. Since $t_{fin}(n)$ is a normal form we have $r(n) \equiv t_{fin}(n)$ and hence a reduction $t_{int}(n)$ to $t_{fin}(n)$. This reduction path encodes a terminating computation path of R by Lemma 5.2.

An equivalent form of Theorem 5.1 is this:

5.4 Theorem There is a finite set E_R of equations over Σ_{st} such that $T(\Sigma_{st}, E_{st} \cup E_R)$ is a semicomputable but not a computable algebra.

6 FINITE SPECIFICATIONS WITHOUT HIDDEN SORTS AND FUNCTIONS

We consider the absence of a finite equational or conditional equation specification (Σ, E) for the class $Alg_m(\Sigma_{st}, E_{st})$ of stack algebras. Recall from Lemma 2.2.3, if (Σ_{st}, E) is an equational specification then

$$Alg_m(\Sigma_{st}, E) = Alg_m(\Sigma_{st}, E_{st}) \quad \Leftrightarrow \quad T(\Sigma_{st}, E) \cong T(\Sigma_{st}, E_{st}).$$

6.1 Theorem $T(\Sigma_{st}, E_{st})$ does not possess a finite equational specification (Σ_{st}, E) and, hence, neither does the class $Alg_m(\Sigma_{st}, E_{st})$ of stack algebras.

Proof Suppose for a contradiction that (Σ_{st}, E) is a finite equational specification for $T(\Sigma_{st}, E_{st})$. For each equation $t = t' \in E$ we compute the normal forms $n(t)$ and $n(t')$ with respect to (Σ_{st}, E_{st}) and form the equation $n(t) = n(t')$. Let $N(E)$ be the set of all such normalised equations.

Let $E_0 \subseteq E_{st}$ be a set of equations sufficient to perform all the reductions of the elements of E to the elements of $N(E)$. We choose $k \in \mathbf{N}$ sufficiently large so that

$$E_0 \subseteq E^k{}_{st} = \{t\text{-}eqn_i,\ p\text{-}eqn_i : i = 1, \dots , k\}.$$

6.2 Lemma $T(\Sigma_{st}, N(E) \cup E^k{}_{st}) \cong T(\Sigma_{st}, E_{st})$.

Proof By Lemma 2.1.9, $(\Sigma_{st}, E \cup E^k{}_{st})$ is a finite equational specification of $T(\Sigma_{st}, E_{st})$ because $E^k{}_{st}$ is valid in $T(\Sigma_{st}, E_{st})$. Now, E and $N(E)$ are logically equivalent in the presence of $E^k{}_{st}$, i.e.

$$E \cup E^k{}_{st} \vdash N(E) \quad \text{and} \quad N(E) \cup E^k{}_{st} \vdash E.$$

Thus, by Lemma 2.1.10, we deduce that

$$T(\Sigma_{st}, E \cup E^k{}_{st}) = T(\Sigma_{st}, N(E) \cup E^k{}_{st})$$

and hence the lemma is proved.

Next we consider the equations of $N(E)$. Since each $n(t) = n(t') \in N(E)$ is valid in $T(\Sigma_{st}, E_{st})$ we know from Lemma 4.2 (i) that $n(t) \equiv n(t')$. Hence each equation in $N(E)$ is trivial and can be removed in Lemma 6.2, i.e.

$$T(\Sigma_{st}, E^k{}_{st}) \cong T(\Sigma_{st}, E_{st}).$$

However, this contradicts Lemma 3.

QED

The question arises: Does there exist a finite conditional equation specification (Σ_{st}, C) for the class of stack algebras. Here we know that if (Σ_{st}, C) is such a specification then

$$Alg_m(\Sigma_{st}, C) = Alg_m(\Sigma_{st}, E_{st}) \quad \text{implies} \quad T(\Sigma_{st}, C) \cong T(\Sigma_{st}, E_{st}).$$

We will prove the converse statement is false.

Consider the set C_{st} of conditional equations over Σ_{st}:

$top(push(i(a), \varnothing)) = i(a)$	t-eqn$_1$
$pop(push(i(a), \varnothing)) = \varnothing$	p-eqn$_1$
$top(x) = i(a) \rightarrow top(push(i(b), x)) = i(b)$	t-ceqn
$top(x) = i(a) \rightarrow pop(push(i(b), x)) = x$	p-ceqn

6.3 Theorem (Σ_{st}, C_{st}) is a finite conditional equation specification for $T(\Sigma_{st}, E_{st})$ and

$$T(\Sigma_{st}, C_{st}) = T(\Sigma_{st}, E_{st}).$$

However, it is not a specification of $Alg_m(\Sigma_{st}, E_{st})$, and we have the proper inclusion

$$Alg_m(\Sigma_{st}, C_{st}) \subset Alg_m(\Sigma_{st}, E_{st}).$$

Proof The axioms of C_{st} are valid in $T(\Sigma_{st}, E_{st})$ by inspection. Hence there is an epimorphism $T(\Sigma_{st}, C_{st}) \rightarrow T(\Sigma_{st}, E_{st})$. To prove the converse, and hence the first part of the theorem by Lemma 2.1.9 and 2.1.10, we prove

$$C_{st} \vdash E_{st}$$

by means of induction on the index k of the equations of E_{st}.

Basis k = 1. This is immediate because t-eqn$_1$ and p-eqn$_1$ of E_{st} are contained in C_{st}.

Induction step k + 1. As induction hypothesis we take

$$C_{st} \vdash E^k_{st}.$$

Consider first equation t-eqn$_{k+1}$

$$top(T_{k+1}) = i(a_{k+1}).$$

By induction, we know

$$top(T_k) = i(a_k)$$

so by conditional equation t-ceqn we have

$$top(push(i(b), T_k)) = i(b).$$

Substituting a_{k+1} for b,

$$top(push(i(a_{k+1}), T_k)) = i(a_{k+1})$$

which is by definition of T_{k+1}

$$top(T_{k+1}) = i(a_{k+1}).$$

The second equation p-eqn$_{k+1}$ can be shown similarly. (Notice it depends on the provability of t-eqn$_{k+1}$.)

To complete the theorem, we construct an algebra

$$A \in Alg_m(\Sigma_{st}, E_{st}) \text{ and } A \notin Alg_m(\Sigma_{st}, C_{st}).$$

Let u be a new constant symbol of sort stack which added to Σ_{st} forms the signature $\Sigma_{st(u)}$. We define a set $E_{st(u)}$ of equations over $\Sigma_{st(u)}$ to be E_{st} with the following added:

(i)	pop(∅)	=	u
(ii)	pop(u)	=	u
(iii)	pop(∅)	=	$i(d_1)$
(iv)	top(u)	=	$i(d_1)$
(v)	push(p, u)	=	u

wherein d_1 is a fixed constant of sort data in Σ_{st}. Because $E_{st} \subseteq E_{st(u)}$ we have that $A = T(\Sigma_{st(u)}, E_{st(u)})|_{\Sigma_{st}}$ is in $Alg(\Sigma_{st}, E_{st})$ using Lemma 2.2.6. Because of equation (i), A is in $Alg_m(\Sigma_{st}, E_{st})$ using Lemma 2.2.6.

We show that t-ceqn of C_{st} is not valid in A. Suppose for a contradiction that this conditional equation is valid. By equation (iv) we deduce that

$$top(push(i(a), u)) = i(a) \qquad (*)$$

But by equation (v),

$$push(i(a), u) = u$$

and so

$$top(push(i(a), u))= top(u)$$

and, by equation (iv),

$$= i(d_1)$$

If we substitute a constant d_2 distinct from d_1 in (*) then we obtain a contradiction.

6.4 Problem Does there exist a finite conditional equation specification (Σ_{st}, C) for the class $Alg_m(\Sigma_{st}, E_{st})$ of stack algebras?

7 FINITE SPECIFICATIONS WITH HIDDEN SORTS AND FUNCTIONS

We define a finite equational specification (Σ_{ast}, E_{ast}), with a hidden or auxiliary sort, for $T(\Sigma_{st}, E_{st})$ and $Alg_m(\Sigma_{st}, E_{st})$. Recall from Lemma 2.2.5, if (Σ, E) is any equational specification with $\Sigma_{st} \subseteq \Sigma$ then

$$Alg_m(\Sigma, E)|_{\Sigma_{st}} = Alg_m(\Sigma_{st}, E_{st}) \quad \text{implies} \quad T(\Sigma, E)|_{\Sigma_{st}} \cong T(\Sigma_{st}, E_{st}),$$

but the converse is not necessarily true.

To Σ_{st} we add a new sort astack for an auxiliary stack, and new constant and operations

$$\begin{array}{lll} \varnothing_{ast} & : & \to \text{astack} \\ \text{push}_{ast} & : \text{data} \times \text{astack} & \to \text{astack} \\ \text{i}_{ast} & : \quad\quad \text{astack} & \to \text{stack} \end{array}$$

The full signature Σ_{ast} is illustrated in Figure 1.

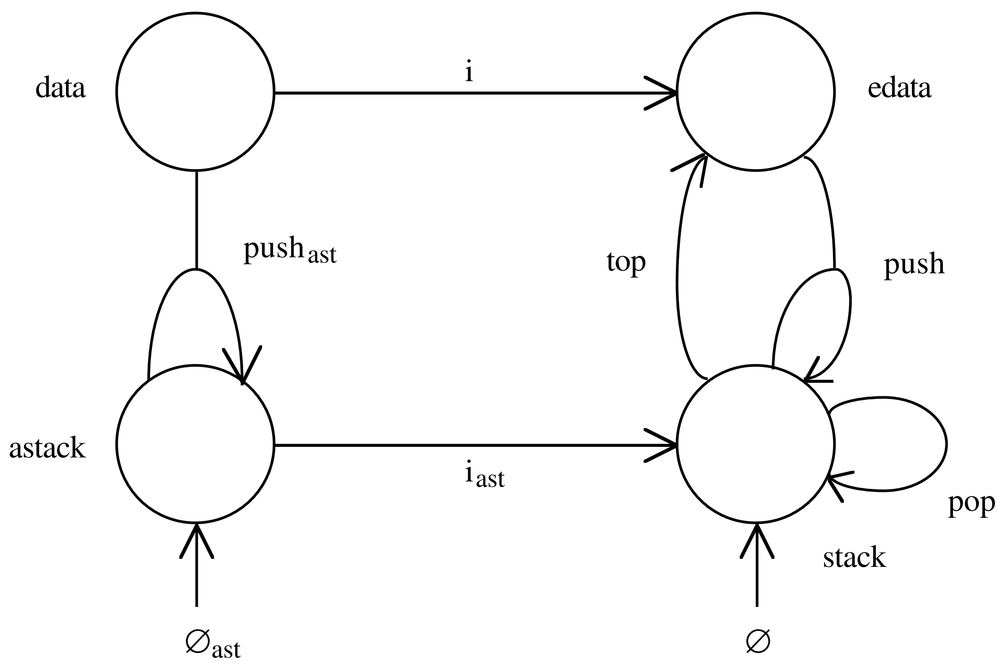

Figure 1.

The four equations of E_{ast} are

$$\begin{array}{lcl} \varnothing & = & i_{ast}(\varnothing_{ast}) \\ push(i(a), i_{ast}(z)) & = & i_{ast}(push_{ast}(a, z)) \\ top(i_{ast}(push_{ast}(a, z))) & = & i(a) \\ top(i_{ast}(push_{ast}(a, z))) & = & i_{ast}(z) \end{array}$$

where a is a variable of sort data, and z is a variable of sort astack

The idea is to enumerate standard stack states (via standard stack polynomials over sort data) within the sort stack using the hidden sort astack.

7.1 Theorem The finite equational specification (Σ_{ast}, E_{ast}) is an orthogonal complete TRS. The set $N(\Sigma_{st}, E_{st})$ of normal forms of the TRS is decidable, and unique normal forms are computable. The initial algebra $T(\Sigma_{ast}, E_{ast})$ is computable.

Proof The TRS (Σ_{ast}, E_{ast}) is orthogonal by inspection. It is terminating because every application of a rule of E_{ast} reduces the number of function symbols from Σ_{st}; the length of any sequence of rewrites of t is bounded by the number of symbols from Σ_{st} appearing in t. The other properties are easy to check.

QED

7.2 Theorem (Σ_{ast}, E_{ast}) is a finite equational specification with hidden sort and functions for the class $Alg_m(\Sigma_{st}, E_{st})$ of all stacks, i.e.

$$Alg_m(\Sigma_{ast}, E_{ast})|_{\Sigma_{st}} = Alg_m(\Sigma_{st}, E_{st})$$

and

$$T(\Sigma_{ast}, E_{ast})|_{\Sigma_{st}} \cong T(\Sigma_{st}, E_{st}).$$

Proof First we show that $Alg_m(\Sigma_{ast}, E_{ast})|_{\Sigma_{st}} \subseteq Alg_m(\Sigma_{st}, E_{st})$. Let $B \in Alg_m(\Sigma_{ast}, E_{ast})$ and $A = B|_{\Sigma_{st}}$. We show that

(i) $A \models E_{st}$, and

(ii) A is Σ_{st} minimal.

We prove $E_{ast} \vdash E_{st}$, thus $B \models E_{st}$ and hence $A \models E_{st}$. Define the standard astack polynomials

$$T^{ast}{}_0 = \varnothing_{ast}$$
$$T^{ast}{}_{k+1} = push_{ast}(a_{k+1}, T^{ast}{}_k)$$

7.3 Lemma $E_{ast} \vdash i_{ast}(T^{ast}{}_k) = T_k$

Proof This is proved by induction on k.

Basis k = 0. The equation is equation 1 of E_{ast}.

Induction step k + 1. We calculate:

$$\begin{aligned} i_{ast}(T^{ast}{}_{k+1}) &= i_{ast}(push_{ast}(a_{k+1}, T^{ast}{}_k)) && \text{by definition} \\ &= push(i(a_{k+1}), i_{ast}(T^{ast}{}_k)) && \text{by equation 2} \\ &= push(i(a_{k+1}), T_k)) && \text{by induction} \\ &= T_{k+1} && \text{by definition} \end{aligned}$$

Consider the equations of E_{st}. First, t-eqn_{k+1}:

$$
\begin{aligned}
top(T_{k+1}) &= top(i_{ast}(T^{ast}{}_{k+1})) && \text{by lemma}\\
&= top(i_{ast}(push_{ast}(a_{k+1}, T^{ast}{}_{k}))) && \text{by definition}\\
&= i(a_{k+1}) && \text{by equation 3}
\end{aligned}
$$

This proves the equation. Next, p-eqn$_{k+1}$:

$$
\begin{aligned}
pop(T_{k+1}) &= pop(i_{ast}(T^{ast}{}_{k+1})) && \text{by lemma}\\
&= pop(i_{ast}(push_{ast}(a_{k+1}, T^{ast}{}_{k}))) && \text{by definition}\\
&= i(T^{ast}{}_{k}) && \text{by equation 4}
\end{aligned}
$$

Now we prove (ii), that A is Σ_{st} minimal. By Lemma 2.2.6, it is sufficient to prove that for any $t' \in T(\Sigma_{ast})$ there is $t \in T(\Sigma_{st})$ such that

$$E_{ast} \vdash t' = t.$$

Let E_{ast2} be a set of equations formed from E_{ast} by changing sides as follows

$$
\begin{aligned}
i_{ast}(\varnothing_{ast}) &= \varnothing\\
i_{ast}(push_{ast}(a, z)) &= push(i(a), i_{ast}(z))\\
top(i_{ast}(push_{ast}(a, z))) &= i(a)\\
pop(i_{ast}(push_{ast}(a, z))) &= i_{ast}(z)
\end{aligned}
$$

Clearly, for any $t' \in T(\Sigma_{ast})$ and $t \in T(\Sigma_{st})$

$$(\Sigma_{ast}, E_{ast}) \vdash t' = t \iff (\Sigma_{ast}, E_{ast2}) \vdash t' = t$$

by initial algebra semantics.

First, we notice that (Σ_{ast}, E_{ast2}) is a terminating TRS because for each rewrite the following number of operation symbols decreases

$$| pop | + | top | + | i_{ast} | + | \varnothing_{ast} |.$$

Notice that E_{ast2} is not orthogonal.

Choose any $t' \in T(\Sigma_{ast})$ and let $n(t')$ be some normal form of t' with respect to (Σ_{ast}, E_{ast2}). We claim that $n(t') \in T(\Sigma_{st})$ and

$$E_{ast} \vdash t' = n(t')$$

shows minimality. Suppose for a contradiction that $n(t') \notin T(\Sigma_{st})$. This implies that there is a maximal subterm r of sort ast. Thus,

$$n(t') = C[r]$$

where C is a non-empty context. We can write

$$C[r] \equiv C_1[C_2[r]]$$

where C_1 may be empty and C_2 involves only one function symbol from Σ_{ast}. Then $C_2 \equiv i_{ast}[.]$ for if this function symbol were $push_{ast}$ then r would not be maximal of sort ast. But $C_2[r]$ is not a normal form hence $n(t')$ is not a normal form, which is the desired contradiction.

Conversely, we show that $Alg_m(\Sigma_{st}, E_{st}) \subseteq Alg_m(\Sigma_{ast}, E_{ast})|_{\Sigma st}$. For each $A \in Alg_m(\Sigma_{st}, E_{st})$ we construct $B \in Alg_m(\Sigma_{ast}, E_{ast})$ such that $A = B|_{\Sigma st}$. To expand the Σ_{st} algebra A to the Σ_{ast} algebra B we add the set

$$D^*$$

of finite sequences over the set D of sort data in A, to interpret the sort astack. The new constant and operations needed are as follows: let $k \in \mathbf{N}$ and $d, d_1, \ldots, d_k \in D$,

(i) $\varnothing_{astB} = \lambda$, the empty sequence

(ii) $push_{astB}: D \times D^* \to D^*$

$$push_{astB}(d, <d_1, \ldots, d_k>) = <d_1, \ldots, d_k, d>$$

(iii) $i_{astB}: D^* \to S$

$$i_{astB}(\lambda) = \varnothing_A$$

$$i_{astB}(<d_1, \ldots, d_k>) = push_A(i_A(d_k), i_{astB}(<d_1, \ldots, d_{k-1}>))$$

Clearly, B is Σ_{ast} minimal and $B|_{\Sigma_{st}} = A$. We must show $B \models E_{ast}$.

Let $k \geq 0$, $d_1, \ldots, d_k, d_{k+1} \in D$ and $<d_1, \ldots, d_k> \in D^*$. Let σ be a state of the variables in B with

$$\sigma(a) = d_{k+1} \quad \text{and} \quad \sigma(z) = <d_1, \ldots, d_k>.$$

We consider the four equations of E_{ast} in turn.

asteqn1 $B \models \varnothing = i_{ast}(\varnothing_{ast})$.

On evaluating for the arbitrary state σ we obtain

$$eval_{B,\sigma}(\varnothing) = \lambda \quad \text{and} \quad eval_{B,\sigma}(i_{ast}(\varnothing_{ast})) = \lambda$$

by definition of the operations in B; so the equation is valid.

asteqn2 $B \models push(i(a), i_{ast}(z)) = i_{ast}(push_{ast}(a, z))$.

On evaluating the RHS for the arbitrary state σ

$$\begin{aligned} eval_{B,\sigma}(i_{ast}(push_{ast}(a, z))) &= i_{astB}(push_{astB}(d_{k+1,} <d_1, \ldots, d_{k-1}>)) \\ &= i_{astB}(<d_1, \ldots, d_k, d_{k+1}>) \\ &= push_A(i_A(d_{k+1,} i_{astB}(<d_1, \ldots, d_{k,}>))) \end{aligned}$$

by definitions of $push_{astB}$ and of i_{astB} for B;

$$= eval_{B,\sigma}(push(i(a), i_{ast}(z)))$$

by evaluation.

Before the next equation, we prove a lemma.

7.4 Lemma Let $d_1, \ldots, d_k \in D$ and let σ be a state with $\sigma(a_i) = d_i$, $1 \leq i \leq k$, and $\sigma(z) = <d_1, \ldots, d_k>$.

Then

$$(B, \sigma) \models i_{ast}(z) = T_k.$$

Proof This is done by induction on k.

Basis k = 0. This is by definition of i_{astB}, etc:

$$eval_{B,\sigma}(i_{ast}(z)) = i_{astB}(\lambda) = \emptyset_A = eval_{B,\sigma}(\emptyset) = eval_{B,\sigma}(T_0).$$

Induction step k + 1. Suppose the lemma is true for k and consider state σ with $\sigma(a_i) = d_i$, $1 \le i \le k + 1$, and $\sigma(z) = \langle d_1, \ldots, d_k, d_{k+1}\rangle$. Now,

$$eval_{B,\sigma}(i_{ast}(z)) = eval_{B,\sigma'}(i_{ast}(push_{ast}(a_{k+1}, z)))$$

by definition of i_{astB}, where

$$\sigma'(z) = \langle d_1, \ldots, d_k\rangle \quad \text{and} \quad \sigma'(a_{k+1}) = d_{k+1};$$

$$= eval_{B,\sigma'}(push(i(a_{k+1}), i_{ast}(z)))$$

by validity of asteqn2 proved earlier;

$$= eval_{B,\sigma'}(push(i(a_{k+1}), T_k))$$

by induction hypothesis;

$$= eval_{B,\sigma'}(T_{k+1})$$

by definition of T_{k+1};

$$= eval_{B,\sigma}(T_{k+1})$$

since z is not free in T_{k+1}.

asteqn3 $B \models top(i_{ast}(push_{ast}(a, z))) = i(a)$

To apply the lemma, we choose, without loss of generality, the equation with variable a_{k+1} in place of a. We evaluate the LHS on state σ in B:

$$eval_{B,\sigma}(top(i_{ast}(push_{ast}(a_{k+1}, z))) = eval_{B,\sigma}(top(push(i(a_{k+1}), i_{ast}(z))))$$

since asteqn2 is valid in B;

$$= eval_{B,\sigma}(top(push(i(a_{k+1}), T_k)))$$

by lemma 7.4;

$$= eval_{B,\sigma}(top(T_{k+1}))$$

by definition of T_{k+1};

$$= eval_{B,\sigma}(i(a_{k+1}))$$

by equation t-eqn$_{k+1}$ of E_{st} and $A \models E_{st}$. Hence the equation is valid at any state σ.

asteqn4 $B \models pop(i_{ast}(push_{ast}(a, z))) = i_{ast}(a)$.

The argument follows that of asteqn3, with pop replacing top.

7.5 Problem Does there exist a finite equational specification involving hidden functions but no hidden sorts for the class $Alg_m(\Sigma_{st}, E_{st})$ of all stacks?

8. CONCLUDING REMARKS

The diversity of algebraic models of the stack is a reflection of the diversity of ideas for the design of stack mechanisms. Despite the progress of algebraic methods in software design (compare surveys Kutzler and Lichtenberger [1983], Wirsing and Bergstra [1989], and Bidoit *et al* [1991], for example) much remains to be understood theoretically about this diversity, which is a general phenomenon for data types, not one specific to the stack. The stacks form an important class in algebraic data type theory for obvious reasons. Our next task is to re-survey the literature on algebraic models of the stack in the light of the mathematical results about the data type variety in this paper (see Bergstra and Tucker [1988,1990] for previous surveys).

REFERENCES

Bergstra, J A and J V Tucker, *Algebraic specifications of computable and semicomputable data structures,* Mathematical Centre, Department of Computer Science Research Report IW 115/79, Amsterdam, l979.

Bergstra, J A and J V Tucker, A natural data type with a finite equational final semantics specification, but no effective equational initial specification, *Bulletin of the EATCS* 11 (1980) 23-33.

Bergstra, J A and J V Tucker, A characterisation of computable data types by means of a finite equational specification method, in J W de Bakker and J van Leeuwen (eds.) *Automata, Languages and Programming (ICALP), 7th Colloquium, Noordwijkerhout,* 1980, Springer LNCS 81, Springer-Verlag, Berlin, 1980, 76-90.

Bergstra, J A and J V Tucker, Initial and final algebra semantics for data type specifications: two characterisation theorems, *SIAM Journal on Computing*, 12 (1983) 366-387.

Bergstra, J A and J V Tucker, The completeness of the algebraic specification methods for computable data types, *Information and Control,* 54 (1983) 186-200.

Bergstra, J A and J V Tucker, Algebraic specifications of computable and semicomputable data types, *Theoretical Computer Science* 50 (1987) 137-181.

Bergstra, J A and J V Tucker, The inescapable stack: an exercise in algebraic specification with total functions, Programming Research Group Report P8804, University of Amsterdam, 1988. Revised version Report P8804b, 1990.

Bergstra, J A and J V Tucker, Equational specifications, complete term rewriting systems, and computable and semicomputable algebras, Programming Research Group Report P9215, University of Amsterdam, 1992.

Bergstra, J A and J V Tucker, Equational specifications for computable data types: 6 hidden functions suffice and other sufficiency bounds, in J V Tucker and K.Meinke (eds.), *Many Sorted Logic and its Applications*, J Wiley and Sons, 1993, 89-102.

Bergstra, J A and J V Tucker, On bounds for the specification of finite data types by means of equations and conditional equations, in J V Tucker and K.Meinke (eds.), *Many Sorted Logic and its Applications*, J Wiley and Sons, 1993, 103-122.

Bidoit, M, H-J Kreowski, P Lescanne, F Orejas, and D Sannella (eds.), *Algebraic System Specification and Development: A Survey and Bibliography*, Lecture Notes in Computer Science 501, Springer-Verlag, Berlin, 1991.

Cutland, N, *Computability: An Introduction to the Theory of the Recursive Functions*, Cambridge University Press, 1980.

Dershowitz, N, Termination of rewriting, *J Symbolic Computation* 3 (1987) 69-116.

Ehrig, H and B Mahr, *Fundamentals of Algebraic Specifications 1*, EATCS Monographs in Theoretical Computer Science 6, Springer Verlag, Berlin, 1985.

Enderton, H B, *A Mathematical Introduction to Logic*, Academic Press, 1972.

Goguen, J A, J W Thatcher and E G Wagner, An initial algebra approach to the specification, correctness and implementation of abstract data types, in R.T Yeh (ed.) *Current Trends in Programming Methodology. IV. Data Structuring,* Prentice-Hall, Engelwood Cliffs, New Jersey, l978, 80-149.

Klop, J W , Term rewriting systems, in S Abramsky, D Gabbay and T S E Maibaum (eds.), *Handbook of Logic in Computer Science. Volume 2.*, Oxford University Press, 1992.

Kutzler, B, and F Lichtenberger, *Bibliography on Abstract Data Types*, Informatik Fachberichte 68, Springer-Verlag, 1983.

Mal'cev, A I, Constructive algebras, I., *Russian Mathematical Surveys*, 16 (1961) 77-129. Also in: *The Metamathematics of Algebraic Systems. Collected Papers 1936-1967*, translated and edited by B F Wells III, North-Holland, Amsterdam, 1971.

Meinke, K and J V Tucker, Universal algebra, in S Abramsky, D Gabbay and T S E Maibaum (eds.), *Handbook of Logic in Computer Science. Volume 1.*, Oxford University Press, 1992, 189-411.

Meseguer, J and J A Goguen, Initiality, induction and computability, in M Nivat and J Reynolds (eds.), *Algebraic Methods in Semantics*, Cambridge University Press, Cambridge, 1985, 459-541.

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

Wirsing, M, Algebraic specifications, in J van Leeuwen (ed.), *Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics,* North-Holland, 1990, 675-788.

Wirsing, M and J A Bergstra, *Algebraic Methods: Theory, Tools and Applications*, Lecture Notes in Computer Science, Springer-Verlag, 1989.