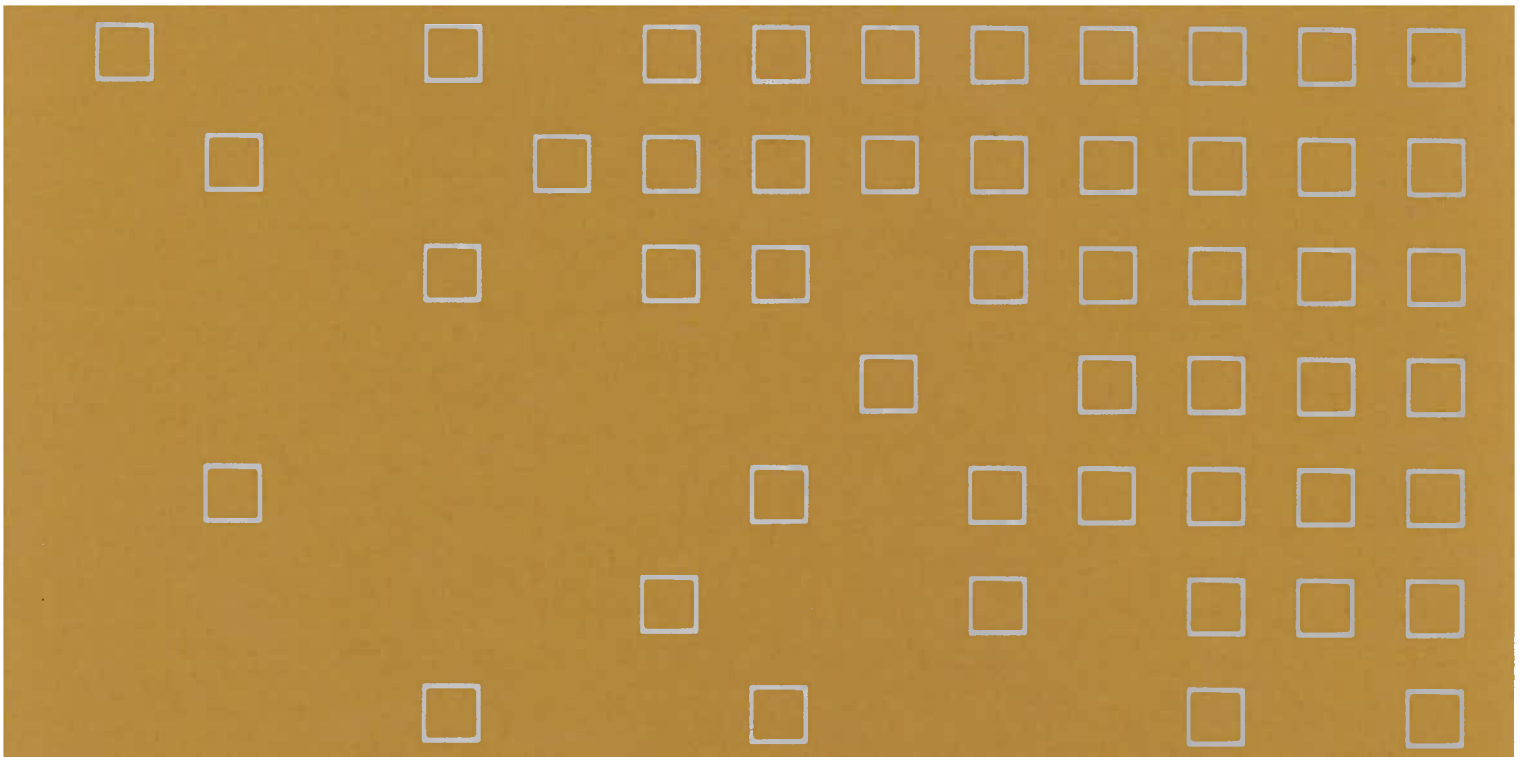

Centrum voor Wiskunde en Informatica

Centre for Mathematics and Computer Science

J.A. Bergstra, J. Heering

Which data types have ω-complete initial algebra specifications?

Computer Science/Department of Software Technology Report CS-R8958 December

1989

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.A. Bergstra, J. Heering

Which data types have ω-complete initial algebra specifications?

Computer Science/Department of Software Technology Report CS-R8958 December

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Which Data Types Have ω-Complete Initial Algebra Specifications?

J.A. Bergstra
Department of Computer Science, University of Amsterdam
Department of Philosophy, University of Utrecht

J. Heering
Department of Software Technology, Centre for Mathematics and Computer Science

An algebraic specification is called *ω-complete* or *inductively complete* if all (open as well as closed) equations valid in its initial model are equationally derivable from it, i.e., if the equational theory of the initial model is identical to the equational theory of the specification. As the latter is recursively enumerable, the initial model of an ω-complete algebraic specification is a data type with a recursively enumerable equational theory. We show that if hidden sorts and functions are allowed in the specification, the converse is also true: every data type with a recursively enumerable equational theory has an ω-complete initial algebra specification with hidden sorts and functions. We also show that in the case of finite data types the hidden sorts can be dispensed with.

Key Words & Phrases: algebraic specification, equational logic, initial algebra semantics, inductive completeness, ω-completeness, functional completeness, Post algebra, finite axiomatizability, hidden sort, hidden function, algebraic semantics of partial evaluation.

1989 CR Categories: F.3.2 **[Logics and meanings of programs]:** Semantics of programming languages - *Algebraic approaches to semantics.*

1985 Mathematics Subject Classification: 03C05 **[Model theory]:** Equational classes, universal algebra; 08B05 **[Varieties]:** Equational logic; 68Q55 **[Theory of computing]:** Semantics; 68Q65 **[Theory of computing]:** Abstract data types.

Note: Partial support received from the European Communities under ESPRIT project 432 (An Integrated Formal Approach to Industrial Software Development - METEOR) and ESPRIT project 2177 (Generation of Interactive Programming Environments II - GIPE II).

Note: This paper will be submitted for publication elsewhere.

1. INTRODUCTION

1.1. Initial algebra specification of data types

Algebraic specifications of data types are often interpreted in terms of initial algebra semantics [MG85]. The data type specified is taken to be the initial algebra of the specification. The latter is characterized by the following two properties:

(i) each of its elements corresponds to at least one closed term (term without variables) over the visible signature of the specification ("no junk");

(ii) two of its elements are never equal unless the corresponding closed terms can be proved equal by means of equational reasoning from the equations given in the specification ("no confusion").

Every algebraic specification whose hidden functions do not generate any "new" elements of visible sorts has an initial algebra satisfying (i) and (ii). It is determined uniquely up to isomorphism.

In view of property (ii) we can write

Report CS-R8958
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

$$ClEqTh(S) = ClEqTh(I(S)),$$

where S is the specification, $I(S)$ its initial algebra, $ClEqTh(S)$ the set of closed equations (equations without variables) over the visible signature of S that are provable from S by means of equational reasoning, and $ClEqTh(I(S))$ the set of closed equations valid in $I(S)$. Since we consider only *finite* specifications, $ClEqTh(S)$ is a recursively enumerable set. Hence, $ClEqTh(I(S))$ is recursively enumerable as well, and this is equivalent to saying that $I(S)$ is a semicomputable algebra. So initial algebra specifications give rise to semicomputable data types. Conversely, if hidden sorts and functions are allowed in the specification, every semicomputable data type has an initial algebra specification. This result, which was proved for the single-sorted case in [BT87], will play a crucial role in the proof of our main theorem in Section 4.

1.2. Equational logic, the equational theory of the initial algebra, and ω-completeness

The identity

$$ClEqTh(S) = ClEqTh(I(S))$$

expresses the fact that equational reasoning is complete with respect to the set of closed equations valid in the initial algebra. If the restriction to closed equations is dropped, however, and open equations (that is, equations containing variables) are taken into account as well, completeness is lost. Let $EqTh(S)$ be the set of open as well as closed equations over the visible signature of S that are equationally provable from S, and let $EqTh(I(S))$ be the set of open as well as closed equations valid in the initial algebra $I(S)$. Due to the "no junk" property of the initial algebra, an open equation is valid in $I(S)$ if all closed equations that can be obtained from it by substituting closed terms over the visible signature of S for its variables, are valid in $I(S)$. Clearly, such an equation need not be valid in models of S containing "junk". As a consequence, equational reasoning need not be complete with respect to $EqTh(I(S))$ and

$$EqTh(S) \subseteq EqTh(I(S))$$

is the only thing that can be stated with certainty in the general case. It may occasionally happen, however, that

$$EqTh(S) = EqTh(I(S))$$

and in that case S is called *inductively complete* [Pau84] or *ω-complete* [Hee86]. All equations valid in the initial algebra of an ω-complete specification S can be proved by purely equational means from the equations given in S.

Consider, for example, the following simple initial algebra specification of the natural numbers with addition and multiplication:

module N
begin
  sort *Num*
  functions 0: *Num*
    S: *Num* → *Num*
    $+, .$: *Num*×*Num* → *Num*
  variables x,y: *Num*
  equations $x+0 = x$
    $x+S(y) = S(x+y)$
    $x.0 = 0$
    $x.S(y) = x+(x.y)$
end N.

N is not ω-complete. The commutative, associative and distributive laws for addition and multiplication, for instance, are not equationally derivable from N, but by adding them an ω-complete specification $\overline{N}$ is obtained [Hen77]:

module $\overline{N}$
begin
import N
variables x,y,z: *Num*
equations $x+y = y+x$
$x+(y+z) = (x+y)+z$
$x.y = y.x$
$x.(y.z) = (x.y).z$
$x.(y+z) = (x.y)+(x.z)$
end $\overline{N}$.

The ω-completeness of $\overline{N}$ follows from the fact that, using the equations of $\overline{N}$, every $(0, S, +, .)$-term can be brought in canonical polynomial form. Two such canonical forms represent the same function on the natural numbers only if they are syntactically equal modulo associativity and commutativity of addition and multiplication.

1.3. Which data types have ω-complete initial algebra specifications?

First of all, it should be noted that if a specification S is ω-complete, the corresponding theory $EqTh(I(S))$ is recursively enumerable since it is equal to $EqTh(S)$ and the latter is recursively enumerable (whether S is ω-complete or not). As we explained in Section 1.1, the set of closed equations $ClEqTh(I(S))$ is always recursively enumerable, but this is not true for the full set of equations $EqTh(I(S))$. Even in seemingly very simple cases the latter is not recursively enumerable. Consider, for instance, the following initial algebra specification of the natural numbers with addition, multiplication and cut-off subtraction:

module N'
begin
import N
function $\dot{-}$: $Num \times Num \rightarrow Num$
variables x,y: *Num*
equations $x \dot{-} 0 = x$
$0 \dot{-} x = 0$
$S(x) \dot{-} S(y) = x \dot{-} y$
end N'.

The corresponding set of equations $EqTh(I(N'))$ is not recursively enumerable (Section 8 of [DMR76]). Hence, N' cannot be extended to an ω-complete specification, not even if hidden sorts and functions are allowed [Pau84, Hee86]. The same example was used in [Nou81] to show that equational logic plus structural induction is not necessarily complete with respect to $EqTh(I(S))$.

The extension of N to $\overline{N}$ did not require the introduction of hidden signature elements. Obviously, ω-complete initial algebra specifications without hidden signature elements give rise to algebras whose equational theory is finitely axiomatizable in terms of equations over the original signature. Such algebras are called *finitely based*. The ω-completeness of $\overline{N}$ shows that the set of natural numbers with addition and multiplication is finitely based. Conversely, the ω-complete specification of non-finitely based algebras, if possible, requires hidden signature elements.

1.3.1. Finite data types

One of the simplest non-finitely based algebras is a three-element groupoid constructed by Murskiĭ [Mur65]. We give an ω-complete specification for it using addition and multiplication modulo three as hidden functions. (In [Hee86] the same was done for a somewhat larger non-finitely based groupoid due to Lyndon.) A straightforward initial algebra specification of Murskiĭ's groupoid is

module M
begin
  sort *Num*
  functions $0, 1, 2$: *Num*
    μ: *Num* $\times$ *Num* $\rightarrow$ *Num*
  variable x: *Num*
  equations $\mu(0,x) = \mu(x,0) = \mu(1,1) = 0$
    $\mu(1,2) = 1$
    $\mu(2,1) = \mu(2,2) = 2$
end M.

It is shown in [Mur65] that for each $n \geqslant 3$ the equation

$$\mu(x_1,\mu(x_2, \cdots \mu(x_{n-1},\mu(x_n,x_1)) \cdots)) = \mu(\mu(x_1,x_2),\mu(x_n,\mu(x_{n-1}, \cdots \mu(x_4,\mu(x_3,x_2)) \cdots)))$$

is valid in $I(M)$ but not provable from equations with less than n different variables. Hence, $I(M)$ is not finitely based.

It so happens that the finite field $\mathbf{Z}_3$ of integers modulo three with addition and multiplication is *functionally complete*, that is, every k-ary total function on $\mathbf{Z}_3$ can be represented by a closed or open $(0, 1, 2, +, .)$-term. Furthermore, $\mathbf{Z}_3$ has an ω-complete specification, which can be obtained in an economical way by taking $\overline{N}$ and adding a few equations to it (cf. [Pau84]):

module $\overline{Z}_3$
begin
  import $\overline{N}$
  functions $1, 2$: *Num*
  variable x: *Num*
  equations $1 = S(0)$
    $2 = S(1)$
    $S(S(S(x))) = x.(x.x) = x$
end $\overline{Z}_3$.

The polynomials $P_{m,n}$ $(m,n = 0,1,2)$ defined by

$$P_{m,n}(x,y) = \prod_{\substack{i=0 \\ i+m\neq 0}}^{2} (x+i) \,.\, \prod_{\substack{j=0 \\ j+n\neq 0}}^{2} (y+j)$$

have the property

$$P_{m,n}(m,n) = 1$$
$$P_{m,n}(x,y) = 0 \quad \text{if } x\neq m \text{ or } y\neq n,$$

so μ can be defined in terms of addition and multiplication modulo three as follows:

$$\mu(x,y) = P_{1,2}(x,y)+2P_{2,1}(x,y)+2P_{2,2}(x,y) = 2x^2y^2+2x^2y+2xy.$$

By adding this definition of μ to $\overline{Z}_3$ and hiding the operators S, $+$ and $.$, an ω-complete specification of $I(M)$ is obtained:

module $\overline{M}$
begin
import $\overline{Z}_3$
hidden functions $S, +, .$
function $\mu: Num \times Num \rightarrow Num$
variables $x,y: Num$
equation $\mu(x,y) = 2x^2y^2 + 2x^2y + 2xy$
end $\overline{M}$.

More generally, $\mathbf{Z}_p$ is functionally complete for every prime p and has an ω-complete specification $\overline{Z}_p$ similar to $\overline{Z}_3$. Hence, the above method of obtaining an ω-complete specification with hidden functions applies to all single-sorted algebras with p elements. If n is not prime, the method breaks down due to the presence of zero divisors in the ring $\mathbf{Z}_n$, but by using the functionally complete Post algebras $\mathbf{P}_n$ rather than $\mathbf{Z}_n$ we will show in Section 3 that *all* finite data types have an ω-complete initial algebra specification with hidden functions.

1.3.2. Infinite data types

The above method does not seem to work for *infinite* data types with a recursively enumerable equational theory. The ω-complete specification $\overline{M}'$ of Murskiĭ's groupoid $I(M)$ shown in Figure 1 is considerably less elegant than the specification $\overline{M}$ given in the previous section, but it illustrates an approach that, apart from equations (6)-(10) which work only in the finite case, does lend itself to generalization. It requires both hidden sorts and hidden functions. It should be emphasized that, in adding hidden elements, we need not bother about equational derivability of open equations containing hidden functions, but only about open equations containing the functions present in $I(M)$.

---

module $\overline{M}'$
begin
import M (Section 1.3.1)
hidden sort *Bool*
hidden functions
$true, false: Bool$
hidden sort *SimVar*
hidden functions
$\xi: SimVar$
$S: SimVar \rightarrow SimVar$
$eq: SimVar \times SimVar \rightarrow Bool$
variables $u,v: SimVar$
equations
(1) $eq(u,u) = true$
(2) $eq(\xi, S(u)) = false$
(3) $eq(S(u), \xi) = false$
(4) $eq(S(u), S(v)) = eq(u,v)$
hidden sort *SimOpenTerm*
hidden functions
$i: Num \rightarrow SimOpenTerm$
$j: SimVar \rightarrow SimOpenTerm$
$m: SimOpenTerm \times SimOpenTerm \rightarrow SimOpenTerm$ (counterpart of μ)
$\sigma: Num \times SimVar \times SimOpenTerm \rightarrow SimOpenTerm$ (substitution)

variables x,y: *Num*
u,v: *SimVar*
t_1,t_2: *SimOpenTerm*

equations

(5) $m(i(x),i(y)) = i(\mu(x,y))$

(6) $\sigma(x,u,i(y)) = i(y)$

(7) $\sigma(x,u,j(u)) = i(x)$

(8) $eq(u,v) = false \Rightarrow \sigma(x,u,j(v)) = j(v)$

(9) $\sigma(x,u,m(t_1,t_2)) = m(\sigma(x,u,t_1),\sigma(x,u,t_2))$

(10) $\sigma(0,u,t_1) = \sigma(0,u,t_2)\ \&\ \sigma(1,u,t_1) = \sigma(1,u,t_2)\ \&\ \sigma(2,u,t_1) = \sigma(2,u,t_2) \Rightarrow t_1 = t_2$

hidden sort *NumList*

hidden functions
nil: *NumList*
list: *Num* × *NumList* → *NumList*
first: *NumList* → *Num*
tail: *NumList* → *NumList*

variables x: *Num*
l: *NumList*

equations

(11) $first(nil) = 0$

(12) $first(list(x,l)) = x$

(13) $tail(nil) = nil$

(14) $tail(list(x,l)) = l$

hidden function
apply: *SimOpenTerm* × *NumList* → *Num*

variables x: *Num*
u: *SimVar*
l: *NumList*
t_1,t_2: *SimOpenTerm*

equations

(15) $apply(i(x),l) = x$

(16) $apply(j(\xi),l) = first(l)$

(17) $apply(j(S(u)),l) = apply(j(u),tail(l))$

(18) $apply(m(t_1,t_2),l) = \mu(apply(t_1,l),apply(t_2,l))$

end $\overline{M}'$

Figure 1

$\overline{M}'$ is an enrichment of the specification M given in the previous section. The hidden machinery of $\overline{M}'$ works as follows. Closed terms of hidden sort *SimOpenTerm* correspond to open terms of sort *Num* of M. The role of variables is played by $j(\xi)$, $j(S(\xi))$, . . . , and the counterparts of the constants 0, 1 and 2 of sort *Num* are $i(0)$, $i(1)$ and $i(2)$. These act as values. Equation (5) establishes m on *SimOpenTerm* as the counterpart of μ on *Num*. The substitution function σ defined by equations (6)-(9) allows substitution of "values" for "variables." It is used in equation (10) to define an equality on closed *SimOpenTerm*-terms corresponding to equality of open *Num*-terms in $I(M)$. This equality is transferred to *Num* by means of the *apply*-function defined in equations (15)-(18). For example, the equation

$$\mu(x,2) = x,$$

which is valid in $I(M)$ but not an equational consequence of M, now obtains the following equational proof: first note that the corresponding *closed* equation of sort *SimOpenTerm*

$$m(j(\xi),i(2)) = j(\xi)$$

follows from equation (10) with $u = j(\xi)$ and equation (5). Next, using *apply* and equations (15)-(18) the left- and right-hand sides can be transformed into the original open terms

$$apply(m(j(\xi),i(2)),list(x,nil)) = \mu(x,2)$$
$$apply(j(\xi),list(x,nil)) = x.$$

Similarly, the equational proof of

$$\mu(x,\mu(y,x)) = \mu(x,y)$$

is

$$\mu(x,\mu(y,x)) = apply(m(j(\xi),m(j(S(\xi)),j(\xi))),list(x,list(y,nil))) \overset{(10)}{=}$$
$$apply(m(j(\xi),j(S(\xi))),list(x,list(y,nil))) = \mu(x,y).$$

For reasons of readability we put equations (8) and (10) in positive conditional form, but this is not strictly necessary. The value 0 assigned to *first*(*nil*) by equation (11) is arbitrary; 1 and 2 would have done equally well.

An ω-complete specification similar to $\overline{M}'$ can be given for all data types with a recursively enumerable equational theory. The introduction of hidden signature elements such that the corresponding closed terms mimic the open terms over the original signature as well as the use of *apply* to transform closed identities into open identities are generally applicable. For infinite data types equations (6)-(10) have to be replaced by other ones, however, so as to obtain a proper definition of the equational theory of the data type in question (possibly by means of additional hidden sorts and functions). We will prove the corresponding general theorem, which is our main result, in Section 4.

1.4. Related work

Plotkin has shown that the $\lambda K\beta\eta$-calculus is ω-incomplete [Plo74]. Paul introduced the notion of inductive completeness while analyzing possible failure modes of the *inductive completion* (also called inductionless induction or proof by consistency) algorithm [Pau84, LLT87]. The equivalent notion of ω-completeness was used in [Hee86] in an attempt to understand what "making maximal use of incomplete information" might mean in the context of *partial evaluation* or *mixed computation* [Ers82]. The problem of the ω-completability of initial algebra specifications addressed in the current paper arose in that context (Open Question 2.6 of [Hee86]). Moller has studied various axiom systems for *process algebra* from the viewpoint of ω-completeness [Mol89].

The standard reference on initial algebra semantics is the survey by Meseguer and Goguen [MG85]. [BT87] is a systematic discussion of the power of initial algebra specification. For relevant work on (non)finitely based algebras the surveys by Taylor [Tay79] and McNulty [McN89] may be consulted. A survey of results on functional completeness and related matters has been given by Rosenberg [Ros77]. The equational theory of the natural numbers with addition, multiplication and various other functions is discussed by, among others, Davis, Matijasevič and Robinson [DMR76], Henkin [Hen77] and Gurevič [Gur85].

Related results on finite axiomatizability of recursively enumerable first-order theories (with and without equality) using hidden predicates were obtained by Kleene [Kle52] and by Craig and Vaught [CV58].

2. PRELIMINARIES

We consider only *finite* specifications. *Provable* always means *equationally provable*. We do not allow algebras with empty carriers or partial functions as models of a specification, so the usual rules of equational logic apply without reservation (cf. Section 4.3 of [MG85]). In the context of a signature *function* always means *n-adic function* ($n \geq 0$). Zero-adic functions are sometimes called *constants*. Signatures never have void (empty) sorts.

As specifications may contain hidden sorts and functions, it is necessary to define the meaning

of hiding at the semantic level of equational theories and initial algebras. Let S be a specification with visible signature Σ and total signature Σ_T. $\Sigma_T - \Sigma$ consists of the hidden sorts and functions of S. Since hidden functions may be defined on visible sorts, $\Sigma_T - \Sigma$ need not be (and virtually never is) a self-contained signature. Let S_T be the specification obtained from S by making the entire signature Σ_T visible. In keeping with [BT87] and the informal discussion in the previous sections we adopt the following conventions:

(1) The set of equations provable from S consists of the Σ-equations provable from S_T, that is,

$$EqTh(S) = Eq(\Sigma) \cap EqTh(S_T),$$

where $Eq(\Sigma)$ is the set of all Σ-equations. Similarly, the set of closed equations (equations without variables) provable from S consists of the closed Σ-equations provable from S_T, that is,

$$ClEqTh(S) = Eq(\Sigma) \cap ClEqTh(S_T).$$

(2) The initial algebra of S is the Σ-reduct of the initial algebra of S_T, that is,

$$I(S) = \Sigma \square I(S_T).$$

The reduct $\Sigma \square I(S_T)$ can be interpreted in two ways (cf. Section 2 of [BT87]), namely
(*a*) as the algebra $I(S_T)|_\Sigma$ consisting of the carriers and functions of $I(S_T)$ named in Σ (the *usual* interpretation), or
(*b*) as the subalgebra of $I(S_T)|_\Sigma$ generated by the functions named in Σ (the *subalgebra* interpretation).
To avoid any possibility of confusion between the two interpretations we consider only specifications for which they coincide. This is the case if $I(S_T)|_\Sigma$ is Σ-*minimal* ("no junk"), that is, if every closed Σ_T-term of a sort in Σ is equal to a closed Σ-term. The hidden functions of such specifications do not generate any "new" elements of visible sorts.

(3) The set $EqTh(I(S))$ consists of the Σ-equations valid in $I(S)$. According to (2) we assume $I(S)$ to be Σ-minimal, so an open equation is valid in $I(S)$ if and only if all closed equations that can be obtained from it by substituting closed Σ-terms for its variables, are valid in $I(S)$. A closed equation is valid in $I(S)$ if and only if it is provable from S in the sense of (1), that is, $ClEqTh(I(S)) = ClEqTh(S)$ ("no confusion").

Keeping these conventions in mind, we can now give a precise definition of ω-completeness:

Definition 1. An algebraic specification S with hidden sorts and functions is *ω-complete* if $EqTh(S) = EqTh(I(S))$.

The ω-completeness of a specification S does *not* imply the ω-completeness of the specification S_T obtained from S by making the entire signature of S visible. Open equations that are valid in $I(S_T)$ need not be equationally derivable if they contain functions that were hidden in S.

3. FINITE DATA TYPES

Theorem 1. Every finite minimal algebra **A** has an ω-complete initial algebra specification with hidden functions. If **A** is single-sorted, the number of hidden functions required is $\leqslant 1$. Otherwise it is $\leqslant 2N_{sorts} + N_{functions}$, where N_{sorts} is the number of sorts and $N_{functions}$ is the number of functions in the signature of **A**.

Proof. (*a*) Let **A** be a minimal algebra with n elements, signature Σ, and single sort E. If $n = 1$ an ω-complete specification $\bar{S}$ of **A** is obtained by adding the equation $x = y$ to S with x,y variables of sort E. If $n \geqslant 2$ consider the specification $\bar{P}_n$ given in Figure 2. Apart from a few insignificant differences, $\bar{P}_n$ is the equational axiomatization of n-valued Post algebras given by Epstein [Eps73]. Post algebras bear the same relationship to many-valued propositional calculi as do Boolean algebras to ordinary propositional calculus. In fact, two-valued Post algebras are Boolean algebras. In that case C_0 is negation, C_1 is the identity function, and $\vee$ and $\wedge$ are ordinary disjunction and conjunction. We only consider the initial algebra of $\bar{P}_n$, which is the n-valued Post algebra with n elements $\mathbf{P}_n$. It is already fully determined by equations (1)-(4). The other equations are valid in $\mathbf{P}_n$ as is easily verified by substituting constants $e_0, \ldots, e_{n-1}$ for the variables occurring in them.

module $\bar{P}_n$
begin
sort E
functions $e_i\colon E \quad (0 \leqslant i \leqslant n-1)$
$C_i\colon E \to E \quad (0 \leqslant i \leqslant n-1)$
$\vee\colon E \times E \to E$
$\wedge\colon E \times E \to E$
variables $x,y,z\colon E$
equations

(1) $C_i(e_i) = e_{n-1}$
(2) $C_i(e_j) = e_0 \quad (j \neq i)$
(3) $e_i \vee e_j = e_k \quad (k = \max(i,j))$
(4) $e_i \wedge e_j = e_k \quad (k = \min(i,j))$
(5) $x \vee y = y \vee x$
(6) $x \wedge y = y \wedge x$
(7) $x \vee (y \vee z) = (x \vee y) \vee z$
(8) $x \wedge (y \wedge z) = (x \wedge y) \wedge z$
(9) $x \vee x = x$
(10) $x \wedge x = x$
(11) $x \wedge (x \vee y) = x$
(12) $x \vee (x \wedge y) = x$
(13) $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
(14) $e_0 \vee x = x$
(15) $x \wedge e_{n-1} = x$
(16) $C_i(x) \wedge C_j(x) = e_0 \quad (i \neq j)$
(17) $\bigvee_{i=0}^{n-1} C_i(x) = e_{n-1}$
(18) $C_i(x \vee y) = \left[C_i(x) \wedge \bigvee_{j=0}^{i} C_j(y)\right] \vee \left[C_i(y) \wedge \bigvee_{j=0}^{i} C_j(x)\right] \qquad (i > 1)$
(19) $C_0(x \wedge y) = C_0(x) \vee C_0(y)$
(20) $x = \bigvee_{i=1}^{n-1} \left[e_i \wedge C_i(x)\right]$

end $\bar{P}_n$

Figure 2

$\mathbf{P}_n$ is a distributive lattice by equations (5)-(13). $\bar{P}_n$ has the following two properties:

(i) *Its initial algebra* $\mathbf{P}_n$ *is functionally complete.* Indeed, every k-ary function f on $\mathbf{P}_n$ can be represented by a term in disjunctive normal form:

$$f(x_1, \ldots, x_k) = \bigvee_{\substack{0 \leqslant i_j \leqslant n-1 \\ 1 \leqslant j \leqslant k}} \left[f(e_{i_1}, \ldots, e_{i_k}) \wedge C_{i_1}(x_1) \wedge \cdots \wedge C_{i_k}(x_k)\right].$$

This is easily verified by noting that $C_{i_1}(e_{i_1}) \wedge \cdots \wedge C_{i_k}(e_{i_k}) = e_{n-1}$ and $C_{i_1}(x_1) \wedge \cdots \wedge C_{i_k}(x_k) = e_0$ otherwise, and by applying (15), (14) and the equation $x \wedge e_0 = e_0$, which is valid in $\mathbf{P}_n$.

(ii) $\bar{P}_n$ *is* ω*-complete.* By virtue of Theorem 13 of [Eps60], equations (1)-(20) are sufficient to bring any term $t[x_1, \ldots, x_k]$ $(k \geqslant 1)$ in disjunctive normal form

$$\bigvee_{\substack{0\leqslant i_j\leqslant n-1\\ 1\leqslant j\leqslant k}} \left[e_{i_1,\ldots,i_k} \wedge C_{i_1}(x_1) \wedge \cdots \wedge C_{i_k}(x_k)\right].$$

Next, all *non-essential* variables, that is, all variables whose value does not affect the value of t if viewed as a function on $\mathbf{P}_n$, can be removed by means of equation (17). This yields the reduced normal form

$$\bigvee_{\substack{0\leqslant i_j\leqslant n-1\\ 1\leqslant j\leqslant l}} \left[e_{i_1,\ldots,i_l} \wedge C_{i_1}(y_1) \wedge \cdots \wedge C_{i_l}(y_l)\right], \qquad (1\leqslant l\leqslant k)$$

where $\{y_1, \ldots, y_l\}$ is the subset of $\{x_1, \ldots, x_k\}$ consisting only of the essential variables of t, or the reduced normal form

$$e_i$$

if all variables of t are non-essential ($l=0$). Two such reduced normal forms are equal in $\mathbf{P}_n$ if and only if they are syntactically identical modulo associativity and commutativity of $\vee$ and $\wedge$ (equations (7)-(8) and (5)-(6)). Hence, $\bar{P}_n$ is ω-complete.

An ω-complete specification of $\mathbf{A}$ can now be obtained by taking $\bar{P}_n$, hiding all its functions, adding Σ, and adding for each function f of Σ its representation as an $(e_0, \ldots, e_{n-1}, C_0, \ldots, C_{n-1}, \vee, \wedge)$-term t_f. (We assume that the functions of $\bar{P}_n$ are not in Σ. Otherwise they have to be renamed first.) This yields the following specification $\bar{S}$:

module $\bar{S}$
begin
  import $\bar{P}_n$
  hidden functions $e_0, \ldots, e_{n-1}, C_0, \ldots, C_{n-1}, \vee, \wedge$
  import Σ
  variables $x_1, \ldots, x_m$: E $\quad (m = \text{maxarity}(\Sigma))$
  equations
    $f(x_1, \ldots, x_k) = t_f[e_0, \ldots, e_{n-1}, C_0, \ldots, C_{n-1}, \vee, \wedge, x_1, \ldots, x_k]$
    $(f\in\Sigma,\ k = \text{arity}(f)\geqslant 0)$
end $\bar{S}$.

To obtain an ω-complete specification with only a single hidden function, we replace all functions of $\bar{P}_n$ except e_0 with the generalized Sheffer function [Web36]

$$x \mid y = \sim(x \vee y),$$

where

$$\sim x = \bigvee_{i=0}^{n-2} \left[e_{i+1} \wedge C_i(x)\right]$$

is a single step rotation, using the identities

$$C_i(x) = \sim(\sim(\sim^{n-1-i} x \vee e_{n-2}) \vee e_{n-2})$$
$$x \wedge y = -(-x \vee -y)$$
$$-x = \bigvee_{j=0}^{n-2} \sim(\sim(\sim^{n-1-j} x \vee e_{n-2}) \vee e_{n-2-j}) \qquad (-e_i = e_{n-1-i})$$
$$e_i = \sim^i e_0 \quad (i\neq 0)$$
$$x \vee y = \sim^{n-1}(x \mid y)$$
$$\sim x = x \mid x.$$

Apart from its smaller signature, the resulting specification $\bar{P}_n'$ has the same properties as $\bar{P}_n$. Since $\mathbf{A}$ is minimal, each of its elements corresponds to a closed Σ-term, so Σ contains at least one constant. Without loss of generality we may assume this constant to be e_0. Using $\bar{P}_n'$, the desired

specification of **A** becomes:

module $\bar{S}'$
begin
  import $\bar{P}_n'$
  hidden function |
  import Σ
  variables $x_1, \ldots, x_m$: E $(m = \text{maxarity}(\Sigma))$
  equations $f(x_1, \ldots, x_k) = t_f[e_0, |, x_1, \ldots, x_k]$ $(f \in \Sigma, f \neq e_0, k = \text{arity}(f) \geqslant 0)$
end $\bar{S}'$.

This proves the theorem for the single-sorted case.

(*b*) Let **A** be a finite many-sorted minimal algebra with signature Σ such that $N_{sorts} \geqslant 2$. Let S be an initial algebra specification of **A** without hidden sorts or functions. Such a specification can be obtained simply by giving an appropriate table of values for each fundamental operation of **A**. Since **A** is minimal, each of its elements corresponds to a closed Σ-term, so this does not require the introduction of hidden items. Let E be a sort in Σ such that the number of elements n of the corresponding carrier of **A** is at least as large as the number of elements of any of the other carriers and consider the following specification:

module S'
begin
  import S
  functions i_s: $s \to E$ $(s \in \Sigma)$
    j_s: $E \to s$ $(s \in \Sigma)$
    τ_f: $E \times \cdots \times E \to E$ $(f \in \Sigma, \text{arity}(\tau_f) = \text{arity}(f) \geqslant 0)$
  variables x_s: s $(s \in \Sigma)$
    $x_1, \ldots, x_m$: E $(m = \text{maxarity}(\Sigma))$
  equations $\cdots$ (equations defining injections i_s for all sorts $s \neq E$)
    $\cdots$ (equations defining surjections j_s for all sorts $s \neq E$)
(1) $i_E(x_E) = x_E$
(2) $j_s(i_s(x_s)) = x_s$
(3) $\tau_f(x_1, \ldots, x_k) = i_{s_0}(f(j_{s_1}(x_1), \ldots, j_{s_k}(x_k)))$ $(\text{type}(f) = s_1 \times \cdots \times s_k \to s_0)$
end S'.

S' takes the original specification S of **A** as its point of departure and adds a pair of functions i_s: $s \to E$ and j_s: $E \to s$ for each sort $s \in \Sigma$ in such a way that equations (1)-(2) hold and **A** is equal to $\Sigma \square I(S')$, the Σ-reduct of $I(S')$ (cf. Section 2). Furthermore, for each $f \in \Sigma$ S' adds a function τ_f of the same arity as f but operating entirely within the confines of E and defined by equation (3). Let Σ_E consist of E and the functions τ_f. To each Σ-term u corresponds a Σ_E-term τ_u which is obtained by replacing each function symbol f in u by τ_f and each variable x of sort s by a variable τ_x of sort E. Repeated application of equations (3) and (2) immediately yields

(3′) $\tau_u[x_1, \ldots, x_k] = i_{s_0}(u[j_{s_1}(x_1), \ldots, j_{s_k}(x_k)])$

for suitable sorts $s_0, \ldots, s_k$. A Σ-equation $u = v$ and its associated Σ_E-equation $\tau_u = \tau_v$ hold simultaneously. Indeed, if $u = v$ holds, then $i_{s_0}(u) = i_{s_0}(v)$ and by substituting $j_s(\tau_x)$ for each s-sorted variable x throughout u and v and applying (3′) $\tau_u = \tau_v$ follows. Conversely, if $\tau_u = \tau_v$ holds for some equation $u = v$, then $u = v$ itself holds as well by (3′), (2) and substitution of $i_s(x)$ for τ_x throughout u and v. As a consequence, an ω-complete specification of $I(S')$ can be obtained from an ω-complete specification of $\Sigma_E \square I(S')$. The latter is a single-sorted minimal algebra so part (*a*) of the proof applies. With $\mathbf{A} = \Sigma \square I(S')$ this yields the following ω-complete specification of **A**:

module $\bar{S}$
begin
import S'
hidden functions i_s, j_s, τ_f
import $\bar{P}_n'$ (with n the cardinality of carrier E of A)
hidden functions $e_0, |$
variables $x_1, \ldots, x_m$: E $(m = \text{maxarity}(\Sigma))$
equations
(4) $\tau_f(x_1, \ldots, x_k) = t_{\tau_f}[e_0, |, x_1, \ldots, x_k]$
end $\bar{S}$.

The number of hidden functions of $\bar{S}$ is $2N_{sorts} + N_{functions} + 2$, but the identity functions i_E and j_E were introduced only for reasons of convenience and can be omitted. This proves the theorem for the many-sorted case. ■

Remarks. (i) The functional completeness of $\mathbf{P}_n$ (with fundamental operations $\sim_n = \sim^{n-1}$ and $\vee_n = \vee$) was first pointed out by Post (Section 11 of [Pos21]).

(ii) A somewhat different equational axiomatization of Post algebras was given by Traczyk [Tra64]. It is based on the fundamental operations C and D_i $(1 \leqslant i \leqslant n-1)$ with $C = C_0$ and

$$D_i(x) = \bigvee_{j=i}^{n-1} C_j(x).$$

The latter are used as auxiliary functions in [Eps60]. They obey the simple laws

$$D_i(x \vee y) = D_i(x) \vee D_i(y)$$
$$D_i(x \wedge y) = D_i(x) \wedge D_i(y).$$

(iii) In the terminology of [BT87] Theorem 1 says that every finite data type has an ω-complete (FIN, EQ, HE) specification. Due to the existence of finite algebras that are not finitely based the hidden functions cannot in general be dispensed with (Section 1.3.1), so (FIN, EQ, HE) cannot be improved to (FIN, EQ).

4. THE MAIN THEOREM

Theorem 2. Every minimal algebra **A** whose equational theory is recursively enumerable has an ω-complete initial algebra specification with hidden sorts and functions.

Proof. (*a*) Let **A** be a minimal algebra with signature Σ and single sort E such that $EqTh(\mathbf{A})$ is recursively enumerable. We first define an algebra **A′** such that $ClEqTh(\mathbf{A}')$ is similar to $EqTh(\mathbf{A})$. Consider the following specification without equations:

module S_0
begin
sort *SimVar*
functions ξ: *SimVar*
S: *SimVar* → *SimVar*
sort *SimOpenTerm*
functions j: *SimVar* → *SimOpenTerm*
ϕ_f: *SimOpenTerm* × ⋯ × *SimOpenTerm* → *SimOpenTerm*
(counterpart of f, $f \in \Sigma$, $\text{arity}(\phi_f) = \text{arity}(f) \geqslant 0$)
end S_0.

Let the variables of sort E be $x_1, x_2, \ldots$ and let the signature of S_0 be Σ_0. With every Σ-term t we associate a *closed* Σ_0-term ϕ_t of sort *SimOpenTerm* which is obtained by replacing each function symbol f in t by ϕ_f and each variable x_k by $j(S^{k-1}(\xi))$. Using ϕ, we define a congruence $\equiv$ on the free term algebra $I(S_0)$:

(i) (u,v closed terms of sort *SimOpenTerm*) $u \equiv v$ if and only if there is an equation $s = t \in EqTh(\mathbf{A})$ such that $\phi_s = u$ and $\phi_t = v$;
(ii) (u,v closed terms of sort *SimVar*) $u \equiv v$ if and only if u and v are syntactically identical.
Let $\mathbf{A}' = I(S_0)/_{\equiv}$. Obviously,

$$s = t \in EqTh(\mathbf{A}) \text{ if and only if } \phi_s = \phi_t \in ClEqTh(\mathbf{A}').$$

Since $EqTh(\mathbf{A})$ is recursively enumerable, $ClEqTh(\mathbf{A}')$ is recursively enumerable as well, so $\mathbf{A}'$ is a semicomputable minimal algebra. Hence, according to Theorem 5.3 of [BT87] there is a specification S' with hidden sorts* and functions such that

$$I(S') = \mathbf{A}'$$

and $I(S')$ is Σ_0-minimal if I is interpreted in the usual way (cf. point (2) of Section 2). Hence, with $ClEqTh(\mathbf{A}') = ClEqTh(I(S')) = ClEqTh(S')$, we have

$$s = t \in EqTh(\mathbf{A}) \text{ if and only if } \phi_s = \phi_t \in ClEqTh(S').$$

To obtain an ω-complete specification $\bar{S}$ of $\mathbf{A}$ we use S' as hidden component and add some further hidden machinery linking *SimOpenTerm* to E:

module $\bar{S}$
begin
import Σ
import S'
hidden sorts *SimVar*,*SimOpenTerm*
hidden functions ξ,S,j,ϕ_f
hidden sort *EList*
hidden functions
nil: *EList*
list: $E \times EList \rightarrow EList$
first: $EList \rightarrow E$
tail: $EList \rightarrow EList$
variables x: E
l: *EList*
equations
(1) *first*(*nil*) = e_0 (e_0 is an arbitrary constant of sort E)
(2) *first*(*list*(x,l)) = x
(3) *tail*(*nil*) = *nil*
(4) *tail*(*list*(x,l)) = l
hidden function
apply: *SimOpenTerm* $\times$ *EList* $\rightarrow E$
variables u: *SimVar*
l: *EList*
$t_1, \ldots, t_m$: E (m = maxarity(Σ))
equations
(5) *apply*($j(\xi),l$) = *first*(l)
(6) *apply*($j(S(u)),l$) = *apply*($j(u)$,*tail*(l))
(7) *apply*($\phi_f(t_1, \ldots, t_k),l$) = f(*apply*(t_1,l), . . . , *apply*(t_k,l))
($f \in \Sigma$, k = arity(f) ≥ 0)
end $\bar{S}$.

* If $\mathbf{A}$ has a *recursive* equational theory, $\mathbf{A}'$ is computable. In that case, S' can do without hidden sorts according to Theorem 5.1 of [BT87].

Without loss of generality we assume that the hidden names of $\bar{S}$ do not occur in Σ. $\bar{S}$ has the following two properties:

(i) *Its initial algebra $I(\bar{S})$ is Σ-minimal if I is interpreted in the usual way*. Indeed, $I(S')$ is Σ_0-minimal if I is interpreted in the usual way, so we need not bother about the hidden functions of S', but may concentrate on the hidden functions introduced in $\bar{S}$. Let t be a *closed* term of sort E not containing any of the hidden functions of S'. If t does not contain *first* or *apply* it is syntactically a Σ-term. If t is of the form *first*(l) with l not containing *first* or *apply* it is equal to a closed Σ-term by (1)-(4). (Without equation (1) this would not be true. The presence of a constant e_0 in Σ is guaranteed by the minimality of **A**.) Finally, if t is of the form *apply*(t',l) with l not containing *apply*, then it is equal to a closed Σ-term by (5)-(7) and (1)-(4).

(ii) $EqTh(\bar{S}) = EqTh(\mathbf{A})$.
For any Σ-term t we have by virtue of the definition of ϕ and equations (1)-(7)

$$apply(\phi_t, list(x_1, list(x_2, \cdots list(x_M, l) \cdots))) = t$$

with M the largest index of any variable x_k occurring in t (0 if t is a closed term) and arbitrary l. Now, if $s = t \in EqTh(\mathbf{A})$, then $\phi_s = \phi_t \in ClEqTh(S')$ and therefore

$$s = apply(\phi_s, list(x_1, list(x_2, \cdots list(x_M, l) \cdots))) \overset{(S')}{=} apply(\phi_t, list(x_1, list(x_2, \cdots list(x_M, l) \cdots))) = t$$

with M the largest index of any variable occurring in s or t. Hence, $s = t \in EqTh(\bar{S})$ and $EqTh(\bar{S}) \supseteq EqTh(\mathbf{A})$.
Conversely, to show that $EqTh(\bar{S}) \subseteq EqTh(\mathbf{A})$ it is sufficient to show that $ClEqTh(\bar{S}) \subseteq EqTh(\mathbf{A})$. If $s = t \in ClEqTh(\bar{S})$, then $M = 0$ and

$$apply(\phi_s, l) = apply(\phi_t, l)$$

with l a variable of sort *EList*. But this implies $\phi_s = \phi_t \in ClEqTh(S')$ as application of equations (4) or (7) is useless and the other ones do not apply. Hence, $s = t \in EqTh(\mathbf{A})$.

We conclude from (i) and (ii) that $\bar{S}$ is an ω-complete initial algebra specification with hidden sorts and functions of **A**. This proves the single-sorted case.

(*b*) We omit the proof of the many-sorted case. It is a straightforward generalization of (*a*). ■

Remarks. (i) The proof of Theorem 2 is a generalization of the construction of $\bar{M}'$ in Section 1.3.2.

(ii) In the terminology of [BT87] Theorem 2 says that every data type with a recursively enumerable equational theory has an ω-complete (FIN, EQ, HES) specification.

5. OPEN PROBLEMS

(1) Whereas hidden functions are in general indispensable, we do not know whether hidden sorts are ever really necessary. The results obtained by Marongiu and Tulipani [MT87, MT89] for ordinary (not necessarily ω-complete) initial algebra specification of semicomputable data types may have consequences for the ω-complete case as well, but these remain to be investigated.

(2) Let S_T be the specification obtained from S by making all its hidden sorts and functions visible (cf. Section 2). Does every minimal algebra **A** with a recursively enumerable equational theory have an ω-complete specification $\bar{S}$ such that $\bar{S}_T$ is ω-complete as well? For this to be true, each **A** should at least have an ω-complete specification $\bar{S}$ such that $EqTh(\bar{S}_T)$ is recursively enumerable. This is a question we have not yet addressed.

(3) Perhaps the main problem is *automatic ω-enrichment* of algebraic specifications, that is, the mechanical addition of identities that are valid in the initial model (cf. Section 4 of [Hee86]). Even rudimentary automatic ω-enrichment will have applications in inductive completion (see Section 1.4), type checking of incomplete programs in the style of Snelting and Henhapl [SH86], unification in equational theories, and the automatic derivation of partial evaluators from standard evaluators. Hannan and Miller [HM89] have made inroads into the last-mentioned problem in a structural operational semantics setting by showing how a specification for the standard evaluation of a simple

functional language can be systematically extended to a specification for partial evaluation.

Acknowledgement. Paul Hendriks and Paul Klint suggested various improvements in the presentation.

REFERENCES

[BT87] J.A. Bergstra & J.V. Tucker, Algebraic specifications of computable and semicomputable data types, *Theoretical Computer Science*, **50** (1987), pp. 137-181.

[CV58] W. Craig & R.L. Vaught, Finite axiomatizability using additional predicate symbols, *The Journal of Symbolic Logic*, **23** (1958), pp. 289-308.

[DMR76] M. Davis, Y. Matijasevič & J. Robinson, Hilbert's tenth problem: positive aspects of a negative solution, in: F.E. Browder (ed.), *Mathematical Developments Arising from Hilbert Problems*, American Mathematical Society, 1976, pp. 323-378.

[Eps60] G. Epstein, The lattice theory of Post algebras, *Transactions of the American Mathematical Society*, **95** (1960), pp. 300-317. Also in: D.C. Rine (ed.), *Computer Science and Multiple-Valued Logic*, North-Holland, 1977/1984, Chapter 2.

[Eps73] G. Epstein, An equational axiomatization for the disjoint system of Post algebras, *IEEE Transactions on Computers*, **C-22** (1973), pp. 422-423.

[Ers82] A.P. Ershov, Mixed computation: Potential applications and problems for study, *Theoretical Computer Science*, **18** (1982), pp. 41-67.

[Gur85] R. Gurevič, Equational theory of positive numbers with exponentiation, *Proceedings of the American Mathematical Society*, **94** (1985), pp. 135-141.

[HM89] J. Hannan & D. Miller, Deriving mixed evaluation from standard evaluation for a simple functional language, in: J.L.A. van de Snepscheut (ed.), *Mathematics of Program Construction*, Lecture Notes in Computer Science, Vol. 375, Springer-Verlag, 1989, pp. 239-255.

[Hee86] J. Heering, Partial evaluation and ω-completeness of algebraic specifications, *Theoretical Computer Science*, **43** (1986), pp. 149-167.

[Hen77] L. Henkin, The logic of equality, *The American Mathematical Monthly*, **84** (1977), pp. 597-612.

[Kle52] S.C. Kleene, Finite axiomatizability of theories in the predicate calculus using additional predicate symbols, *Memoirs of the American Mathematical Society*, No. 10 (1952), pp. 27-68. Second printing, with revisions, American Mathematical Society, Providence, Rhode Island, 1967.

[LLT87] A. Lazrek, P. Lescanne & J.-J. Thiel, Proving inductive equalities: algorithms and implementation, Report No. 682, INRIA-Lorraine, June 1987.

[McN89] G.F. McNulty, An equational logic sampler, in: N. Dershowitz (ed.), *Rewriting Techniques and Applications*, Lecture Notes in Computer Science, Vol. 355, Springer-Verlag, 1989, pp. 234-262.

[MG85] J. Meseguer & J.A. Goguen, Initiality, induction, and computability, in: M. Nivat & J.C. Reynolds (eds.), *Algebraic Methods in Semantics*, Cambridge University Press, 1985, pp. 459-541.

[Mol89] F. Moller, *Axioms for Concurrency*, PhD Thesis, Report CST-59-89, University of Edinburgh, July 1989.

[MT87] G. Marongiu & S. Tulipani, Finite algebraic specifications of semicomputable data types, in: H. Ehrig et al. (eds.), *TAPSOFT '87*, Vol. I, Lecture Notes in Computer Science, Vol. 249, Springer-Verlag, 1987, pp. 111-122.

[MT89] G. Marongiu & S. Tulipani, On a conjecture of Bergstra and Tucker, *Theoretical Computer Science*, **67** (1989), pp. 87-97.

[Mur65] V.L. Murskiĭ, The existence in three-valued logic of a closed class with finite basis, not

having a finite complete system of identities, *Doklady Akademii Nauk SSSR*, **163** (1965), pp. 815-818 [Russian]. English translation in: *Soviet Mathematics Doklady*, **6** (1965), pp. 1020-1024.

[Nou81] F. Nourani, On induction for programming logic: syntax, semantics, and inductive closure, *Bulletin of the European Association for Theoretical Computer Science*, No. 13 (February 1981), pp. 51-64.

[Pau84] E. Paul, Proof by induction in equational theories with relations between constructors, in: B. Courcelle (ed.), *Ninth Colloquium on Trees in Algebra and Programming*, Cambridge University Press, 1984, pp. 211-225.

[Plo81] G.D. Plotkin, The λ-calculus is ω-incomplete, *Journal of Symbolic Logic*, **39** (1974), pp. 313-317.

[Pos21] E.L. Post, Introduction to a general theory of elementary propositions, *American Journal of Mathematics*, **43** (1921), pp. 163-185. Also in: J. van Heijenoort (ed.), *From Frege to Gödel*, Harvard University Press, 1967, pp. 264-283.

[Ros77] I.G. Rosenberg, Completeness properties of multiple-valued logic algebras, in: D.C. Rine (ed.), *Computer Science and Multiple-Valued Logic*, North-Holland, 1977/1984, Chapter 6.

[SH86] G. Snelting & W. Henhapl, Unification in many-sorted algebras as a device for incremental semantic analysis, in: *Conference Record of the Thirteenth ACM Symposium on Principles of Programming Languages*, ACM, 1986, pp. 229-235.

[Tay79] W. Taylor, Equational logic, *Houston Journal of Mathematics*, Survey 1979. Abridged version in: G. Grätzer, *Universal Algebra*, Second ed., Springer-Verlag, 1979, Appendix 4.

[Tra64] T. Traczyk, An equational definition of a class of Post algebras, *Bulletin de l'Académie Polonaise des Sciences, Série des sciences math., astr. et phys.*, **XII** (1964), pp. 147-149.

[Web36] D.L. Webb, Definition of Post's generalized negative and maximum in terms of one binary operation, *American Journal of Mathematics*, **58** (1936), pp. 193-194.