INFORMATION AND COMPUTATION 119, 119–123 (1995)

Homomorphism Preserving Algebraic Specifications Require Hidden Sorts*

J. A. BERGSTRA

Faculty of Mathematics and Computer Science, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
E-mail: janb@fwi.uva.nl

AND

J. HEERING

CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
E-mail: jan@cwi.nl

Although every computable data type has an initial algebra specification with hidden functions, it may happen that some of the homomorphic images of the data type are not models of the specification. The latter are reducts of algebras that would be models of the specification if all its functions were visible, whereas the homomorphic images of the data type are independent of the specification and need not be compatible with the hidden functions used in it. A hidden function specification that does not exclude any of the homomorphic images of its initial model from its model class will be called *homomorphism preserving*. It turns out that, unlike unrestricted initial algebra specification, homomorphism preserving initial algebra specification of computable data types requires both hidden sorts and hidden functions.

1. INTRODUCTION

1.1 *Homomorphism Preserving Hidden Function Specifications*

The first example of a data type whose algebraic specification requires hidden (auxiliary) functions was given by Majster (1977). It was subsequently shown by Bergstra and Tucker (1982) that allowing hidden functions is sufficient in the sense that every *computable* data type has an initial algebra specification with hidden functions. The initial algebra of a hidden function specification is obtained by first taking the initial algebra of the specification as if none of its functions were hidden, and then restricting this algebra to the functions that are actually visible. More generally, a model of a hidden function specification is the reduct of an algebra that would be a model of the specification if all its functions were visible.

It may happen that, given a computable data type and an initial algebra specification with hidden functions for it, some of the algebras that are homomorphic images of the data type are no longer models (in the above sense) of the specification. The reason is that, whereas the homomorphic images of the data type are independent of the hidden functions of the specification, the models of the specification are not. A simple example of a non-homomorphism preserving specification is the single-sorted specification with constants a, b, c, d, hidden unary function f, and equations $f(a)=c$, $f(b)=d$, $f(c)=c$, $f(d)=d$. Its initial algebra has a homomorphic image which identifies a and b, but not c and d. This is not a model of the specification, however, since the addition of f would result in identification of c and d as well.

More generally, it was shown by Bergstra and Tucker (1982) that every computable data type has an algebraic specification with hidden functions that defines it under initial and final algebra semantics simultaneously. All non-trivial minimal models of such a specification are isomorphic to the data type, so none of the non-trivial homomorphic images of the data type (if any) is a model of the specification.

If the homomorphic images of the initial algebra of an algebraic specification are among its models, we shall call it *homomorphism preserving*. According to Birkhoff's theorem an equational variety is closed under homomorphic images. As a consequence, algebraic specifications without conditional equations and without hidden functions are always homomorphism preserving.

Positive conditional specifications without hidden functions need not be homomorphism preserving since a

* Partial support was received from the European Communities under ESPRIT Basic Research Action 7166 (CONCUR II) and ESPRIT Project 2177 (Generation of Interactive Programming Environments II—GIPE II) and from the Netherlands Organization for Scientific Research (NWO) under the *Generic Tools for Program Analysis and Optimization* project.

0890-5401/95 $6.00

homomorphism may cause the conditions of an equation to become true without causing its conclusion to become true at the same time. Hence, a conditional equational variety (quasivariety) need not be closed under homomorphic images. Cf. Section 3 of Taylor's survey (1979). In this paper specifications are assumed to be unconditional.

Obviously, from the viewpoint of pure initial algebra specification homomorphism preserving specifications are of no importance. Only if a wider class of algebras has to be specified does this property become an issue. For instance, the equational axiom system ACP for process algebra in Bergstra and Klop (1984) uses the left-merge and communication merge operators as auxiliary functions, but it is not at all obvious that these are compatible with (can be added to) all algebras one would like ACP to capture.

In some cases the static semantics of a programming or application language is a homomorphic image of its dynamic semantics (cf. the notion of *abstract interpretation*—a homomorphism is an algebraically well-behaved abstraction function). In principle, such a homomorphic relationship may be exploited at the specification level by reusing the dynamic semantics specification as part of the static semantics specification. The homomorphism is defined implicitly by the equations added to the dynamic semantics specification. This possibility is blocked, however, if the auxiliary functions (and/or conditional equations—see above) used in the dynamic semantics specification do not preserve the homomorphism in question.

More generally, adding equations to a hidden function specification in order to obtain a specification of a homomorphic image of the original data type may be (and very likely is) impossible if the specification was not explicitly written with this kind of reuse in mind. As will be explained below, this may require the introduction of hidden sorts that would otherwise not be necessary.

1.2. *Hiding vs Textual Deletion as a Modularization Operator*

The fact that the reusability of a specification may be hampered by the presence of hidden functions manifests itself in another, less fundamental, way. If used in a new context in which only a subset of the visible functions of a specification is needed, hiding the unnecessary functions is not the same as textually deleting them from the specification. For instance, if an initial algebra specification of the natural numbers $\geqslant 1$ with addition, multiplication, and exponentiation is reused in a definition of the natural numbers mod k with addition and multiplication, the exponentiation operator has to be deleted from the specification. Hiding it is not sufficient since for most k exponentiation is not well-behaved mod k, that is, for most k there are $m_1, m_2, n_1, n_2 \geqslant 1$ with $m_1 = m_2 \pmod{k}$ and $n_1 = n_2 \pmod{k}$ but $m_1^{n_1} \neq m_2^{n_2} \pmod{k}$. (The exceptional values of k for which exponentiation is well-behaved mod k are 1, 2, 6, 42, 1806, a result due to D. Higgs. Cf. Corollary 3.2 of Burris and Lee, 1993.)

Similarly, an initial algebra specification of lists with a length function cannot be made into a definition of sets unless the length function is deleted. As in the previous case, hiding it is not sufficient since the length function is not compatible with the idempotent law for sets.

The algebraic semantics of the modularization operators $+$ and $\square$ were discussed in Bergstra *et al.* (1990). The latter, which restricts the visible signature of a specification, is directly related to hiding and satisfies several important identities. A delete operator, on the other hand, is not considered, nor is it available in most algebraic specification languages. As a consequence, modules that are "too large" can be reused only after the relevant parts have been put in a separate module, an operation which affects their modular structure. With the textual delete operator no restructuring is necessary.

1.3 *Two Basic Questions*

In this paper we address two basic adequacy questions:

(1) Does every computable data type have a homomorphism preserving initial algebra specification with hidden functions?

We show that the answer to this question is negative (Theorem 2, Section 3). This is our main theorem. It immediately suggests the second question:

(2) Does every computable data type have a homomorphism preserving initial algebra specification with hidden sorts and functions?

The answer to this question is positive (Theorem 3, Section 3). Hence, homomorphism preserving initial algebra specification of computable data types requires both hidden sorts and hidden functions.

2. PRELIMINARIES

For the notions of equational specification, initial algebra, and homomorphism the reader is referred to the surveys by Meseguer and Goguen (1985), Meinke and Tucker (1992), or Wirsing (1990). We consider only *finite* equational specifications. Equations do not have conditions. 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 (see Section 4.3 of Meseguer and Goguen's survey (1985) for a discussion of the effects of allowing models with empty carriers on the rules of equational logic). In the context of a signature *function* always means *n-adic function* ($n \geqslant 0$). Zero-adic functions are sometimes called *constants*.

Signatures never have void (empty) sorts, that is, sorts for which there are no closed terms.

As specifications may contain hidden sorts and functions, we have to define the meaning of hiding at the semantic level. 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. 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 Bergstra and Tucker (1987)), 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. The same convention is adopted in Bergstra and Tucker (1982, 1987).

The basic adequacy result for initial algebra specification of computable data types says that hidden sorts are not necessary:

THEOREM 1 (Bergstra and Tucker, 1982). *For every computable data type (computable minimal algebra) A with signature Σ there is an equational specification S with visible signature Σ and total signature Σ_T such that $\Sigma_T - \Sigma$ contains no sorts and $I(S) \cong A$.*

Let A be an algebra with signature Σ and let $\mathrm{Hom}(A)$ be the class of homomorphic images of A, that is, the class of Σ-algebras B such that $\alpha(A) = B$ for some Σ-homomorphism α. Furthermore, let

$$\Sigma' \square \mathrm{Hom}(A) = \{\Sigma' \square B \mid B \in \mathrm{Hom}(A)\}.$$

Keeping these conventions in mind, we define:

DEFINITION 1. An initial algebra specification S with visible signature Σ and total signature Σ_T is *homomorphism preserving* if

$$\Sigma \square \mathrm{Hom}(I(S_T)) = \mathrm{Hom}(I(S))(= \mathrm{Hom}(\Sigma \square I(S_T))).$$

3. HOMOMORPHISM PRESERVING SPECIFICATIONS REQUIRE HIDDEN SORTS

THEOREM 2. *Not every computable data type has a homomorphism preserving initial algebra specification with hidden functions but without hidden sorts.*

Remark. This result should be contrasted with Theorem 1.

Remark. In the terminology of Bergstra and Tucker (1987) the theorem says that not every computable data type has a homomorphism preserving (FIN, EQ, HE) specification.

Sketch of Proof. By taking an infinite computable data type A_ϕ whose structure depends on a computable function ϕ of sufficiently high asymptotic time complexity, and assuming that A_ϕ has a homomorphism preserving initial algebra specification without hidden sorts, we derive a procedure for computing ϕ that, using finite homomorphic images of A_ϕ, computes ϕ more quickly than is warranted by its time complexity, thus yielding a contradiction.

Proof. Let ω denote the set of natural numbers and let $\phi: \omega \to \{0, 1\}$ be a computable function of sufficiently high asymptotic time complexity (to be determined). Let A_ϕ be the minimal algebra with constant 0 and unary generators s and g satisfying the equations

$$s(g(s^k(0))) = g(g(s^k(0))) \Leftrightarrow \phi(k) = 0 \ (k \in \omega) \qquad (1)$$

$$s(s(g(s^k(0)))) = s(g(s^k(0)))$$

$$g(s(g(s^k(0)))) = s(g(s^k(0)))$$

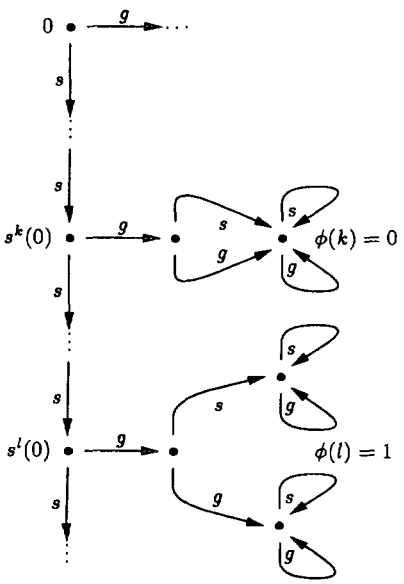

FIG. 1. The algebra A_ϕ.

$$s(g(g(s^k(0)))) = g(g(s^k(0)))$$
$$g(g(g(s^k(0)))) = g(g(s^k(0))),$$

but otherwise free (Fig. 1).

Since A_ϕ is a computable minimal algebra, it is the initial algebra of some hidden function specification S according to Theorem 1. Let $\Sigma = \{0, s, g\}$ be the visible signature of S and let Σ_T be its total signature.

Consider the finite homomorphic image $A_{\phi,n}$ of A_ϕ obtained by factoring A_ϕ with respect to the congruence generated by the equation $s^n(0) = s^{n+1}(0)$ (Fig. 2). $A_{\phi,n}$ has $\leqslant 4(n+1)$ elements.

Assuming that S is homomorphism preserving, there is an effective procedure for computing $\phi(n-1)$ as follows:

for all Σ-algebras $B = A_{\theta,n}$ with θ some function $\omega \to \{0, 1\}$ **do**
  for all Σ_T-algebras B' such that $\Sigma \square B' = B$ **do**
    if B' satisfies the equations of S
    then comment B is a homomorphic image of $A_{\phi,n}$
      if $\theta(n-1) = 1$
      then comment The inequality $s(g(s^{n-1}(0))) \neq g(g(s^{n-1}(0)))$ holds in B' according to (1). Since B' satisfies the equations of S, this inequality must hold in the initial model A_ϕ of S as well.
        $\phi(n-1) := 1$; **halt**
      else comment The equation $s(g(s^{n-1}(0))) = g(g(s^{n-1}(0)))$ holds in B' according to (1). It need not hold in A_ϕ, however, unless it holds in all B' satisfying the equations of S.
        continue
      fi
    fi
  od
od
comment The equation $s(g(s^{n-1}(0))) = g(g(s^{n-1}(0)))$ holds in all B' satisfying the equations of S. Note that there is at least one such B': since S is homomorphism preserving and $A_{\phi,n}$ is a homomorphic image of A_ϕ, there is a B' with $\Sigma \square B' = A_{\phi,n}$ satisfying the equations of S.
$\phi(n-1) := 0$; **halt**

We estimate the time complexity of this procedure by noting that the outer loop ranges over

$$2^{n+1}$$

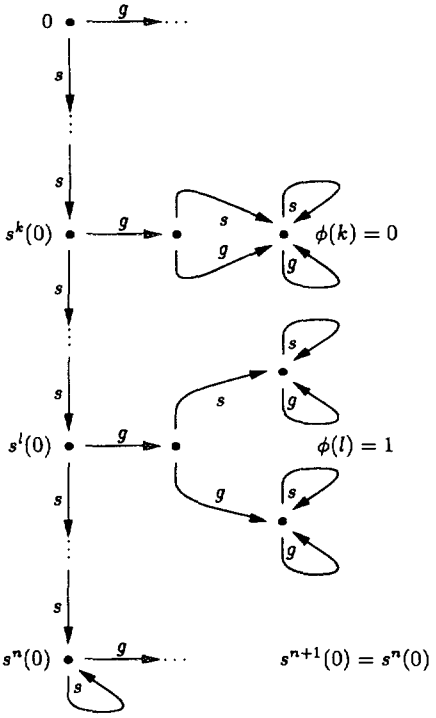

FIG. 2. The finite homomorphic image $A_{\phi,n}$.

possible finite structures B. Each B in turn can be enriched to a Σ_T-algebra B' in at most

$$N^{hN^r}$$

different ways, where N is the number of elements of B, h is the number of hidden functions of S, and r is their maximal arity. Recall that $N \leqslant 4(n+1)$. This provides an upper bound on the range of the inner loop. Finally, the test for whether B' satisfies the equations of S in the body of the inner loop requires $p(N)$ table lookups, where p is a polynomial depending on the number of equations of S and their structure.

Hence the above procedure has asymptotic time complexity

$$O(n^{q(n)}),$$

where q is a polynomial depending on S. This yields a contradiction if ϕ has complexity

$$\Omega(n^{n^n})$$

in the same model of computation in which the complexity of the procedure is measured. Such a ϕ exists by the Time Complexity Hierarchy Theorem (see, e.g., Theorem 7.7.1 of Lewis and Papadimitriou (1981)). Hence, S cannot be homomorphism preserving for such a ϕ. ∎

THEOREM 3. *Every computable data type has a homomorphism preserving initial algebra specification with hidden sorts and functions.*

Remark. In the classification of Bergstra and Tucker (1987) this means that every computable data type has a homomorphism preserving (FIN, EQ, HES) specification.

Proof. According to Theorem 5.3 of Bergstra and Tucker (1987) every semicomputable data type has an initial algebra specification with hidden sorts and functions. Its proof actually shows a somewhat stronger result: every semicomputable data type has a specification with hidden sorts and functions such that the hidden functions do not have visible argument sorts, but only hidden ones. Since this particular type of hidden structure is not affected by identification of visible elements, it can be added to any homomorphic image of the data type. Hence, the specification is homomorphism preserving. ∎

Remark. In view of the proof, the theorem remains valid for semicomputable data types.

Remark. The proof implicitly contains a sufficient (but not a necessary) textual condition for a specification with hidden sorts and functions to be homomorphism preserving. No comparable condition for hidden function specifications is known to us.

Remark. As one of the referees pointed out, under the subalgebra interpretation of the reduct operator $\square$ (cf. Section 2) the hidden sorts can be be replaced with extensions to visible sorts, so in that case Theorem 3 can be strengthened and Theorem 2 no longer holds. Note that these specifications are not correct from the viewpoint of the usual interpretation, however.

4. OPEN QUESTIONS

(1) The axiom system ACP for process algebra in Bergstra and Klop (1984) uses the left-merge and communication merge operators as auxiliary functions. Is ACP homomorphism preserving?

(2) An algebraic specification is called ω-complete if all (open as well as closed) equations valid in its initial algebra are equationally derivable from it, i.e., if the equational theory of the initial algebra is identical to the equational theory of the specification. If ω-completeness of the specification is an issue, hidden functions are necessary even in the case of finite data types. Neither the ω-complete hidden function specifications of finite data types given by Gagliardi and Tulipani (1990) nor those in Bergstra and Heering (1994) are homomorphism preserving, however. In both cases the ω-complete specifications are equationally complete (maximally consistent), so all non-trivial homomorphic images are lost. Hence, it is an open question whether every finite data type has a hidden function specification that is both ω-complete and homomorphism preserving.

Received July 8, 1993; final manuscript received March 2, 1994

REFERENCES

Bergstra, J. A., and Heering, J. (1994), Which data types have ω-complete initial algebra specifications? *Theoret. Comput. Sci.* **124**, 149–168.

Bergstra, J. A., Heering, J., and Klint, P. (1990), Module algebra, *J. Assoc. Comput. Mach.* **37**, 335–372.

Bergstra, J. A., and Klop, J. W. (1984), Process algebra for synchronous communication, *Inform. and Control* **60**, 109–137.

Bergstra, J. A., and Tucker, J. V. (1982), The completeness of the algebraic specification methods for computable data types, *Inform. and Control* **54**, 186–200.

Bergstra, J. A., and Tucker, J. V. (1987), Algebraic specifications of computable and semicomputable data types, *Theoret. Comput. Sci.* **50**, 137–181.

Burris, S., and Lee, S. (1993), Tarski's high school identities, *Amer. Math. Monthly* **100**, 231–236.

Gagliardi, G., and Tulipani, S. (1990), On algebraic specifications of computable algebras with the discriminator technique, *Theoret. Informat. Appl.* **24**, 429–440.

Lewis, H. R., and Papadimitriou, C. H. (1981), "Elements of the Theory of Computation," Prentice–Hall, Englewood Cliffs, NJ.

Majster, M. E. (1977), Limits of the "algebraic" specification of abstract data types, *SIGPLAN Not.* **12** (10), 37–42.

Meinke, K., and Tucker, J. V. (1992), Universal Algebra, *in* "Handbook of Logic in Computer Science" (S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, Eds.), Vol. I, pp. 189–411, Oxford Univ. Press, Oxford.

Meseguer, J., and Goguen, J. A. (1985), Initiality, induction, and computability, *in* "Algebraic Methods in Semantics" (M. Nivat and J. C. Reynolds, Eds.), pp. 459–541, Cambridge Univ. Press, Cambridge, UK.

Taylor, W. (1979), Equational logic, *Houston J. Math. Survey*. Abridged version *in* "Universal Algebra" (G. Grätzer, Ed.), 2nd ed., Appendix 4, Springer-Verlag, Berlin, 1979.

Wirsing, M. (1990), Algebraic specification, *in* "Handbook of Theoretical Computer Science" (J. van Leeuwen, Ed.), Vol. B, Chap. 13, Elsevier, Amsterdam.