Theoretical Computer Science 31 (1984) 49–59
North-Holland

CHARACTERIZING SPECIFICATION LANGUAGES WHICH ADMIT INITIAL SEMANTICS

B. MAHR*
Department of Computer Science, Pennsylvania State University, University Park, PA 16802, U.S.A.

J.A. MAKOWSKY**
Department of Computer Science, Technion—Israel Institute of Technology, Haifa, Israel

Communicated by E. Engeler
Received April 1981
Revised September 1983

Abstract. The paper proposes an axiomatic approach to specification languages, and introduces notions of reducibility and equivalence as tools for their study and comparison. Algebraic specification languages are characterized up to equivalence. They are shown to be limited in expressive power by implicational languages.

1. Introduction

Specification of abstract data types has been widely discussed in the literature of the last decade and a great variety of specification languages and techniques has been proposed. In this paper we present a most general axiomatic framework to speak about specification languages, to compare their expressive power and to axiomatize their semantic behavior. This framework is meant to capture only the basic properties which are satisfied by reasonable specification languages. Properties concerning syntactic representation of specifications or modularization techniques are not considered in this framework which may be viewed as an abstract logic for specification. Unlike for various other logics we do not demand closure under conjunction, negation and quantification, but closure under isomorphism and renaming, and compatibility with reducts. In this respect our framework is closely related to the concept of 'institution', which was independently developed by Goguen and Burstall [4, 9].

We use this framework to characterize those specification languages which are 'strongly algebraic' and 'admit initial semantics'. Our main result states:

> *An algebraic specification language admits initial semantics if and only if it is fully equivalent to an implicational language.*

* The first author was supported by the MINERVA Foundation for German–Israeli collaboration. Present address: Fachbereich Informatik, Technische Universität Berlin, 1000 Berlin 12, Fed. Rep. Germany.
** The second author was supported by the Swiss National Science Foundation, Grant No. 82.820.0.80.

Here, strongly algebraic ensures that the initial objec s in K are exactly the free objects over the empty set of generators. A specificatic n language is called 'implicational' if it is powerful enough to specify all varieties and if all specifications consist of a set of implications with possibly infinitary premises. Under additional assumptions 'finite' implications are sufficient.

The proofs of this and other results in this paper mainly consist in an adaptation of two important results in universal algebra to our framework: Mal'cev's [18], and Cudnovskii's [8] (independently also found by Andreka and Nemeti [2] and Banachewski and Herrlich [5]). Our results shows that in a sense only the use of implications (or equivalent formulas) is 'safe' for specifications with initial semantics. This confirms some long-held prejudices and 'defends' the use of equations and implications of equations in the ADJ-approach to data type specification (see, e.g., [1]). Note that inequalities are a special case of this of the form 'equation implies **false**'. For a detailed discussion of the significance of such a characterization theorem the reader is referred to [16]. This paper is based on our presentation given in [15].

2. The specification language framework

Roughly speaking, specification languages provide syntactical and semantical means to define classes of algebras or structures. Here also a single structure may be viewed as a class, the one with only one element. We give an axiomatic definition of specification languages which is based on just this idea, and discuss some examples in the light of this framework.

2.1. *Basic notions*

We refer to a signature σ as a quadruple (S, C, F, R) consisting of a set S of sorts, an S-sorted family C of sets of constant symbols, and (S^*, S)-sorted family F of sets of function (or operation) symbols, and an S^*-sorted family R of sets of relation (or predicate) symbols; (we sometimes identify constant symbols with 0-ary operation symbols). The class of signatures is denoted by Σ. We call a signature algebraic if it has *no relation symbols*.

We assume signatures partially ordered by the relation of inclusion:

$$(S', C', F', R') \subseteq (S, C, F, R) \text{ if } S' \subseteq S,\ C'_s \subseteq C_s,\ F'_{w,s} \subseteq F_{w,s} \text{ and } R'_w \subseteq R_w$$
$$\text{for all } s \in S \text{ and } w \in S^*.$$

For signatures σ and τ we define *renaming* to be a function $r: \sigma \to \tau$ which is bijective and compatible with the sorting of the components of τ and σ. *Structures* with respect to a signature σ are defined in the usual way, consisting of *non-empty*

domains, constants, operations and relations, named by the symbols in σ. *Renaming carries over to structures*, and we denote by $A^{(r)}$, with respect to a renaming r, the structure which is identical to A, except that its domains, constants, operations and relations are renamed according to r.

For signatures $\tau \subset \sigma$ we define $A\restriction \tau$ to be the *reduct* of the σ-structure A to signature τ. By $Struct(\tau)$ we denote the *class of all τ-structures* and by $Struct_T$ the class $\bigcup_{\tau \in T} Struct(\tau)$ for some class of signatures T.

If τ is algebraic then $A \in Struct(\tau)$ is also called *algebra*, and $Struct(\tau)$ is sometimes denoted by $Alg(\tau)$. Also Alg_T may be used for $Struct_T$ if T is a class of algebraic signatures.

Cardinals are denoted by α, the *cardinality of the natural numbers* by ω, and we use $F_\tau(\alpha)$ to denote the *word algebra* with α generators of signature τ (neglecting predicate symbols). Elements of $F_\tau(\alpha)$ are called *terms*.

Other notions are standard in universal algebra or logic and we refer to [10, 19] for further background.

2.2. Definition (*specification language*)

A specification language $\boldsymbol{L} = (T, K, L, \models)$ consists of the following items:

(i) a class of signatures T;

(ii) a mapping $K: T \to 2^{Struct_T}$ with $K(\tau) \subset Struct(\tau)$;

(iii) a mapping $L: T \to 2^S$ where S is a class of so called $\boldsymbol{L}$-*sentences* containing a distinct sentence **false**;

(iv) a family $\models = (\models_\tau)_{\tau \in T}$ with $\models_\tau \subset K(\tau) \times L(\tau)$.

We require that the following properties hold:

(1) $A \cong B$ implies $A \in K(\tau)$ iff $B \in K(\tau)$.

(2) $A \cong B$ implies $A \models \varphi$ iff $B \models \varphi$.

(3) *If* $r: \tau \to \sigma$ is a renaming, then $A \in K(\tau)$ iff $A^{(r)} \in K(\sigma)$.

(4) If $r: \tau \to \sigma$ is a renaming and $\varphi \in L(\tau)$, then there is a sentence $\varphi^{(r)} \in L(\sigma)$ such that for all $A \in K(\tau) A \models \varphi$ iff $A^{(r)} \models \varphi^{(r)}$.

(5) If $\tau \subset \sigma$, then $L(\tau) \subset L(\sigma)$.

(6) If $\tau \subset \sigma$, $\varphi \in L(\tau)$, $A \in K(\sigma)$ and $A\restriction \tau \in K(\tau)$, then $A \models \varphi$ iff $A\restriction \tau \models \varphi$.

(7) For all $\tau \in T$ we have $\mathbf{false} \in L(\tau)$ and for no A we have $A \models \mathbf{false}$.

We extend the 'satisfaction relation' to sets of sentences and define for a set $\Phi \subset L(\tau)$ of $\boldsymbol{L}$-sentences $A \models \Phi$ iff $A \models \varphi$ for all $\varphi \in \Phi$. By $Mod(\Phi)$ we denote the class of structures A such that $A \models \Phi$. A *specification in* $\boldsymbol{L}$ is a pair (τ, Φ) with $\tau \in T$ and $\Phi \subset L(\tau)$. A class $K \subset Struct(\tau)$ is called $\boldsymbol{L}$-*definable* if there is a specification (τ, Φ) in $\boldsymbol{L}$ such that $K = Mod(\Phi) \cap K(\tau)$. In this case we call K the *class defined by* (τ, Φ). Axioms (1) and (2) say that specification languages do not make a distinction between isomorphic structures. Axioms (3) and (4) say that names have no impact on definability; and changing of names does not violate validity of sentences in structures. Axioms (5) and (6) assure that validity of sentences is

independent from operations and relations to which they are not related. Axiom (7) is for technical convenience only.

It is important to notice that we have not assumed sentences to be of a special form. Sentences can be Gödel numbers, equations, second-order sentences or even functors—just any object in some formalism, which defines structures. Most naturally, sentences (or sets of sentences) can be axioms used in existing specification languages. We illustrate this definition of specification language by a number of examples that we will also use in Section 4 of this paper.

2.3. *Examples of specification languages*

(1) *Many sorted equational logic* (= *universal algebra*). $\boldsymbol{L}_1 = (T_1, K_1, L_1, \models_1)$, with T_1 all algebraic signatures; $K_1(\tau) := Alg(\tau)$; $L_1(\tau) := (F_\tau(\omega) \times F_\tau(\omega)) \cup \{\textbf{false}\}$, i.e., equations with variables; and $A \models_1 \varphi$ iff A satisfies the universal closure of the equation φ.

In $\boldsymbol{L}_1$ exactly the many sorted varieties and the empty class are definable. Specifications correspond to algebraic specifications in the sense of [1].

(2) *Algebraic specifications with hidden functions and sorts.* $\boldsymbol{L}_2 = (T_2, K_2, L_2, \models_2)$, with T_2 all algebraic signatures; $K_2(\tau) := Alg(\tau)$; $L_2(\tau) := (\bigcup_{\tau \subset \sigma} F_\sigma(\omega) \times F_\sigma(\omega)) \cup \{\textbf{false}\}$, i.e., equations using arbitrary operation symbols; and $A \models_2 \varphi$ iff there is a signature σ such that $\tau \subset \sigma$, $\varphi \in F_\sigma(\omega) \times F_\sigma(\omega)$ and there exists $B \in Alg(\sigma)$ with $B \models_1 \varphi$ and $B \restriction \tau = A$.

Specifications in $\boldsymbol{L}_2$ correspond to equational algebraic specifications with hidden sorts and operations.

(3) *First-order logic with equality.* $\boldsymbol{L}_3 = (T_3, K_3, L_3, \models_3)$, with T_3 all many-sorted signatures (including predicate symbols); $K_3(\tau) := Struct(\tau)$; $L_3(\tau)$ all first-order sentences built from constant, operation and predicate symbols of τ, as well as logical constants $\forall$, $\exists$, $\bigwedge$, $\bigvee$, $\neg$, $\rightarrow$ and equality $=$, as well as the atomic sentence **false**; and $A \models_3 \varphi$ iff A is a model of φ.

Exactly the axiomatizable classes of many-sorted structures are $\boldsymbol{L}_3$-definable.

(4) *Equational Horn formula language.* $\boldsymbol{L}_4 = (T_4, K_4, L_4, \models_4)$, with T_4 all algebraic signatures; $K_4(\tau) := Alg(\tau)$; $L_4(\tau) := Horn(\tau, \omega) \cup \{\textbf{false}\}$, where $\varphi \in Horn(\tau, \omega)$ iff φ is a *finite* disjunction of equations $e_i \in L_1(\tau)$ and negations of equations, with at most one of them not negated; and $A \models_4 \varphi$ iff A is a model of φ.

Equational Horn formulas with *exactly one* non-negated equation are called *strict (equational) Horn formulas*. (Strict) Horn formulas are the finite case of the following:

(5) *Generalized implication language.* $\boldsymbol{L}_5 = (T_5, K_5, L_5, \models_5)$, with $T_5 :=$ all algebraic signatures; $K_5(\tau) := Alg(\tau)$; $L_5(\tau) := Imp(\tau, \omega) \cup \{\textbf{false}\}$;

We define $Imp(\tau, \omega)$ inductively as follows:

(a) If $\{e_i : i \in I\}$ is an indexed family of equations $e_i \in F_\tau(\omega) \times F_\tau(\omega)$ and e is a single such equation or e is **false**, then

$$\left(\bigwedge_{i \in I} e_i\right) \rightarrow e \in Imp(\tau, \omega);$$

(b) if $\{\varphi_j: j \in J\}$ is an indexed family of elements $\varphi_j \in Imp(\tau, \omega)$ then

$$\bigwedge_{j \in J} \varphi_j \in Imp(\tau, \omega).$$

Elements in $Imp(\tau, \omega)$ can be interpreted as sets of implicational equations with set-many equations in the premises. If e is **false**, we have actually a negation of an infinite conjunction. We call the elements of $Imp(\tau, \omega)$ which do not contain **false** *strict implications.*

The satisfaction relation for $Imp(\tau, \omega)$ is defined accordingly: $A \models_s \varphi$ iff A satisfies φ.

We call classes definable by sets of finite Horn formulas *quasi-varieties.* There is some confusion in the literature and what we call strict Horn formulas are simply called Horn formulas. Accordingly, quasi-varieties appear in the literature sometimes as classes definable by strict Horn formulas. We shall call the latter *strict quasi-varieties. The main differences lies in the fact* that strict quasi-varieties always contain the *trivial structure* (which is the same as the empty cartesian product).

Many more examples are possible which show the usefulness and generality of our axiomatic framework. Among such examples are final data type specifications (cf. [15, 21]), specifications as proposed by Maibaum et al. [7] or languages as studied by Kamin [13] and Bergstra and Tucker [6].

The axiomatic framework given above, is closely related to 'institutions' introduced and studied by Burstall and Goguen [4, 9]. Institutions are defined in categorical terminology and are more general by allowing signatures and structures different from those we have defined. Axiomatic frameworks like the above are also studied in abstract model theory, as illustrated in the collection of surveys in the monograph edited by Barwise and Feferman [3], especially the expository articles by Makowsky [17]. But there the emphasis is more on closure operations and definability theory on one side, and on algebraic construction in general on the other side. A discussion of the difference between abstract model theory and axiomatic semantics, as proposed here, may also be found in [16].

3. Reducibility and equivalence

We introduce in this section the notions of *reducibility* between and *equivalence* of specification languages. This allows us to study the relationship between specification languages as well as their expressive power.

3.1. Definition (reducibility and equivalence)

Given specification languages $\boldsymbol{L} = (T, K, L, \models)$ and $\boldsymbol{L}' = (T', K', L', \models)$, a mapping $C: \Sigma \to 2^{Struct_\Sigma}$ with $C(\tau) \subseteq Struct(\tau)$, and a cardinal α. Then we say:

(1) $\boldsymbol{L}$ *is* (C, α)*-reducible to* $\boldsymbol{L}'$, and write $\boldsymbol{L} <^{\alpha}_{C} \boldsymbol{L}'$, if $T \subset T$ and for all $\tau \in T$ and $\varphi \in L(\tau)$ there is $\psi \subset L'(\tau)$ with $card(\psi) < \alpha$ such that

$$Mod(\varphi) \cap K(\tau) \cap C(\tau) = Mod(\psi) \cap K(\tau) \cap C(\tau).$$

We write $\boldsymbol{L} <_{C} \boldsymbol{L}'$, if there is an α such that $\boldsymbol{L} <^{\alpha}_{C} \boldsymbol{L}'$, and we write $\boldsymbol{L} <^{\infty}_{C} \boldsymbol{L}'$, if the cardinality of ψ cannot be bounded by some cardinal.

(2) $\boldsymbol{L}$ *is fully* α*-reducible to* $\boldsymbol{L}'$, written as $\boldsymbol{L} <^{\alpha} \boldsymbol{L}'$, if $\boldsymbol{L}$ is (C, α)-reducible to $\boldsymbol{L}'$ for every C. Analogously, we write $\boldsymbol{L} < \boldsymbol{L}'$ if there is an α such that $\boldsymbol{L} <^{\alpha} \boldsymbol{L}'$, and $\boldsymbol{L} <^{\infty} \boldsymbol{L}'$ if there is no bound α.

(3) $\boldsymbol{L}$ *is* (C, α)*-equivalent to* $\boldsymbol{L}'$, denoted by $\boldsymbol{L} \equiv^{\alpha}_{C} \boldsymbol{L}'$ if $\boldsymbol{L} <^{\alpha}_{C} \boldsymbol{L}$ and $\boldsymbol{L}' <^{\alpha}_{C} \boldsymbol{L}$. $\boldsymbol{L}$ *is fully* α*-equivalent to* $\boldsymbol{L}'$, written as $\boldsymbol{L} \equiv^{\alpha} \boldsymbol{L}'$, if $\boldsymbol{L} \equiv^{\alpha}_{C} \boldsymbol{L}'$ for all C. Analogously to (1) and (2) we write $\boldsymbol{L} \equiv_{C} \boldsymbol{L}'$, $\boldsymbol{L} \equiv^{\infty}_{C} \boldsymbol{L}'$ and $\boldsymbol{L} \equiv \boldsymbol{L}'$ and $\boldsymbol{L} \equiv^{\infty} \boldsymbol{L}'$.

The role of the mapping C is to restrict the comparison of specification languages to algebras which are in the domain of C. It turns reducibility into an extremely sensitive tool. The role of the cardinal α is to give additional information about the relation between specification languages. The following facts are easily derived from the definition.

3.2. *Facts*

(1) $\boldsymbol{L} <^{\alpha}_{C} \boldsymbol{L}'$ implies $\boldsymbol{L} <^{\beta}_{D} \boldsymbol{L}'$, for all $\beta \geq \alpha$ and D with $D(\tau) \subset C(\tau)$ for all $\tau \in T$.

(2) $\boldsymbol{L} <^{\alpha} \boldsymbol{L}'$ if and only if $\boldsymbol{L} <^{\alpha}_{C} \boldsymbol{L}'$ for some C with $K(\tau) \cup K(\tau) \subset C(\tau)$ for all $\tau \in T$.

(3) $\boldsymbol{L} < \boldsymbol{L}'$ or $\boldsymbol{L} <^{\infty} \boldsymbol{L}'$ if and only if every $\boldsymbol{L}$-definable class is also $\boldsymbol{L}'$ definable.

(4) For any cardinal $\alpha > \omega$ the relation $<^{\alpha}_{C}$ is reflexive and transitive, and $\equiv^{\alpha}_{C}$ is an equivalence relation.

Had we assumed sentences to be closed under conjunction, this would be true for all α.

4. Initial semantics

In this section we single out those specification languages which are 'algebraic' and 'admit initial semantics', and characterize these languages up to equivalence.

4.1. *Definition (algebraic, admits initial semantics)*

(1) A specification language $\boldsymbol{L} = (T, K, L, \models)$ is called *algebraic* if all signatures $\tau \in T$ are algebraic, and all varieties $V(\tau) \subset Alg(\tau)$ are $\boldsymbol{L}$-definable.

(2) A structure $A \in Alg(\tau)$ is called *reachable*, if every element $a \in A$ is the interpretation of some term over τ. If τ contains at least one constant symbol, then for every $A \in Alg(\tau)$ there is a unique reachable substructure $A_0 \subset A$.

(3) A specification language $\boldsymbol{L}=(T, K, L, \models)$ is called *strongly algebraic*, if all signatures contain at least one constant symbol and all non-empty $\boldsymbol{L}$-definable classes contain a reachable structure. (In [14] strongly algebraic languages are said to 'admit reachable structures'.)

(4) An algebraic specification language $\boldsymbol{L}$ is said to *admit initial semantics*, if all non-empty $\boldsymbol{L}$-definable classes K contain an initial object, i.e., there is an $A \in K$ such that for all $B \in K$ there is exactly one homomorphism $h: A \to B$.

Note, if a specification language $\boldsymbol{L}$ is algebraic, then for every $\tau \in T$ the class $Alg(\tau)$ is $\boldsymbol{L}$-definable, since it is the variety specified by the empty set of equations. Thus $K(\tau) = Alg(\tau)$ in any algebraic specification language.

4.2. Examples and facts

(1) The languages $\boldsymbol{L}_1$ (equational logic), $\boldsymbol{L}_4$ (equational Horn logic) and $\boldsymbol{L}_6$ (generalized implicational logic) defined in Section 2.3 are all algebraic and even strongly algebraic, provided every $\tau \in T$ contains at least a constant symbol. They also admit initial semantics. For $\boldsymbol{L}_1$ this is well known, for the languages $\boldsymbol{L}_4$ and $\boldsymbol{L}_5$, this will also follow from the results below.

(2) The language $\boldsymbol{L}_2$ (equational logic with hidden functions) is algebraic, but does not admit initial semantics. This can be seen from the following counter-example: Let (τ, Φ) be a specification in $\boldsymbol{L}_2$ such that τ is single-sorted and contains only one unary operation symbol *id*. Let $\Phi = \{\varphi_1, \ldots, \varphi_6\}$ such that $\{\varphi_1, \ldots, \varphi_5\}$ is a set of equations defining the variety of groups of order 3 (i.e., satisfying the equation $x + x + x = 0$), and φ_6 is the equation $id(x) = x$. Then $Mod(\Phi)$ contains no algebra with two elements, but an algebra with three elements, say B. $B\restriction \tau$ is just the 3-element set with the identity function, which has the two-element set with identity as a sub-algebra. But this two-element set cannot be the reduct of some algebra which satisfies Φ; so it cannot be $\boldsymbol{L}_2$-definable.

(3) The language $\boldsymbol{L}_6$ (first-order logic) is not algebraic, even if restricted to algebraic signatures only; though in the latter case $\boldsymbol{L}_6$ is an algebraic language, it does not admit initial semantics.

The following are some useful facts:

(4) Let $\boldsymbol{L} <^{*} \boldsymbol{L}'$. If $\boldsymbol{L}$ is (strongly) algebraic, then $\boldsymbol{L}'$ is also (strongly) algebraic.

(5) Let $\boldsymbol{L} <^{*} \boldsymbol{L}'$, and $\boldsymbol{L}$ and $\boldsymbol{L}'$ be (strongly) algebraic. If $\boldsymbol{L}$ admits initial semantics, then $\boldsymbol{L}'$ admits initial semantics.

If we restrict our attention to (reachable) initial algebras (i.e., algebras without endomorphisms), then the language $\boldsymbol{L}_1$ of simple equations provides full expressive power. This stems from the fact that an algebra A is initial in some class of τ-algebras iff it is initial in the class $\{A\}$ iff A is endomorphism-free. Therefore we have the following theorem.

Theorem 4.1. *An algebraic specification language $\boldsymbol{L}$ admits initial semantics if and only if $\boldsymbol{L} \equiv_c \boldsymbol{L}_1$ with $C(\tau) := \{A \in Alg(\tau): A \text{ is initial}\}$.*

Proof. The proof is simple: We only have to build a specification in $\boldsymbol{L}_1$ from all the equations true in the initial object of the $\boldsymbol{L}$-defined class under consideration. The converse is trivial by assumption that $\boldsymbol{L}$ is algebraic. □

A characterization of strongly algebraic specification languages which admit initial semantics, even if we allow arbitrary algebras, is given in the next theorem. Here we use the concept of full equivalence.

We call a non-empty class $K \subset Alg(\tau)$ *implicational* if there is a specification (τ, Φ) in the language $\boldsymbol{L}_5$ (cf. Section 2.3(5)) which defines K. We call K *strictly implicational*, if there is a specification (τ, Φ) in the language $\boldsymbol{L}_6$, consisting of strict implications only, which defines K. A specification language $\boldsymbol{L}$ is called *implicational* (*strictly implicational*) if it is strongly algebraic and if all specifications in $\boldsymbol{L}$ are also specifications in $\boldsymbol{L}_5$ (consisting of strict implications only).

Theorem 4.2. *A strongly algebraic specification language $\boldsymbol{L}$ admits initial semantics if and only if it is fully equivalent to an implicational language $\boldsymbol{L}'$, i.e., $\boldsymbol{L} \equiv \boldsymbol{L}'$.*

The proof of this theorem makes use of two theorems in universal algebra, adapted to the many-sorted case, and one lemma.

The first theorem is due to Mal'cev [18] and characterizes free classes of algebras. A class of algebras K is called *free* if it is non-empty, closed under isomorphic images, and if the intersection of K with any variety V of the same type contains free algebras of arbitrary rank, provided $K \cap V \neq \emptyset$.

Theorem 4.3 (Mal'cev's Theorem). *A class K of algebras is free if and only if K contains all subalgebras and nonempty direct products of its members.*

In the literature there is some confusion concerning this theorem. Mal'cev also allows the empty product and therefore one has to assume that the class K always contains the trivial structure. However, his proof can be modified to yield the above version of his theorem. In fact, this corresponds to two version of his theorem: one characterizing quasi-varieties and one characterizing strict quasi-varieties in terms of the existence of sufficiently many free structures.

The second theorem, which was given independently by several authors (see, e.g., [2, 5, 8]) characterizes implicational classes:

Theorem 4.4. *A class K of algebras is (strictly) implicational if and only if K contains all subalgebras and (non-empty) direct products of its members.*

Both theorems remain true if adopted to the many-sorted case. They are related to our framework by the following lemma.

Lemma 4.5. *Let $\boldsymbol{L}$ be a strongly algebraic specification language, then $\boldsymbol{L}$ admits initial semantics if and only if every non-empty $\boldsymbol{L}$-definable class is free.*

Proof (idea). If an **L**-definable class K is free, it contains an initial algebra, the free algebra of rank 0. Here we use our reachability assumption. Conversely, if $\boldsymbol{L}$ admits initial semantics, then any non-empty $\boldsymbol{L}$-definable class K is free. Construct the free algebras as reducts of initial algebras in appropriate classes $Exp(K)$ by interpreting the generating elements as zero-ary operations. □

Proof of Theorem 4.2. Putting together Theorem 4.3, Theorem 4.4 and Lemma 4.5, one easily obtains the result stated in Theorem 4.2. □

The characterization of initial semantics in terms of equivalence to sets of infinitary formulas is certainly not a result very appealing from a practical point of view. Indeed, it would be desirable to find characterizations which use only finitary formulas, and additionally imply specifications which are recursively enumerable or even finite. A first step into this direction is based on the following classical characterization.

Let us call a non-empty class $K \subset Alg(\tau)$ *universal Horn* (*strictly universal Horn*), if there is specification (τ, Φ) in the language $\boldsymbol{L}_4$ (cf. Section 2.3(4)) which defines K (and consists of strict Horn formulas only).

Theorem 4.6. (McKinsey, 1943). *Let K be a class definable by a set of first-order sentences, then K is (strictly) universal Horn if and only if K contains all subalgebras and (non-empty) direct products of its members.*

For a proof one may consult, e.g., [10]. For an axiomatization of the consequence relation of finite Horn sentences one may consult [20].

We call a specification language $\boldsymbol{L}$ *Horn* if it is strongly algebraic and if all specifications in $\boldsymbol{L}$ are also specifications in $\boldsymbol{L}_4$.

Theorem 4.7. *Let $\boldsymbol{L}$ be an algebraic specification language which is fully ω-reducible to the language $\boldsymbol{L}_3$, i.e., $\boldsymbol{L} \leqslant^{\omega} \boldsymbol{L}_3$, then $\boldsymbol{L}$ admits initial semantics if and only if it is fully equivalent to a specification language $\boldsymbol{L}'$ which is Horn.*

This result is a direct consequence of the previous theorems and the lemma mentioned above.

We have seen that specification languages which are implicational or Horn are strongly algebraic and admit initial semantics. From what is known about implications, this is not new. But Theorems 4.2 and 4.7 also state, in some sense, the converse and show that no other strongly algebraic specification language which admits initial semantics has more expressive power than Horn or implicational languages. This, in fact, makes clear, that any attempt to increase the expressive power beyond implicational or Horn languages, but with the desired property of admitting initial semantics preserved, must fail.

Strongly algebraic specification languages which admit initial semantics are bounded in their expressive power: From below by the language $\boldsymbol{L}_1$ and from above

by the language L_5. They form, with respect to union and intersection, up to equivalence, a complete lattice between L_1 and L_5

What happens if we weaken our assumption and drop the reachability requirement? The characterization Theorems 4.2 and 4.7 become false for trivial reasons. But from the point of view of data structures it seems rather unnatural to deal with algebras (structures) where there are elements which are not the interpretation of some term. If such a situation occurs, one can usually achieve reachability by changing the signature.

Acknowledgment

We wish to thank J.T. Baldwin and B. Mayoh for stimulating discussions during the preparation of this paper. We wish to thank H. Andreka, I. Nemeti and A. Tarlecki for many valuable comments. A. Tarlecki pointed out that in [15] we mistakenly forgotten to state that we only treat the case where the initial structures are reachable. In [14] and [16] this assumption is explicitly stated.

References

[1] J.A. Goguen, J. Thatcher and E. Wagner, Abstract data types as initial algebras and correctness of data representations, in: R. Yeh, ed., *Current Trends in Programming Methodology, Vol. 4* (Prentice-Hall, New York, 1978) pp. 80–149.

[2] H. Andreka and I. Nemeti, Generalization of variety and quasivariety concepts to partial algebras through category theory, *Dissertationes Math. (Rozprawy Mat.)* **204** (1982).

[3] J. Barwise and S. Feferman, *Higher Model Theory: Logic of Mathematical Concepts* (Springer, Berlin, 1983).

[4] R.M. Burstall and J.A. Goguen, The semantics of CLEAR, a specification language, in: *Proc. 1979 Copenhagen Winter School of Abstract Software Specifications* (1980).

[5] B. Banaschewski and H. Herrlich, Subcategories defined by implications, *Houston J. Math.* **2** (2) (1976) 149–171.

[6] J.A. Bergstra and J.V. Tucker, Algebraic specifications of computable and semicomputable data structures, TR IW-115/79, Mathematisch Centrum, Amsterdam, 1979.

[7] R.L. de Carvalho, T.S.E. Maibaum, T.H.C. Pequeno, A.A. Pereda and P.A.S. Veloso, A model theoretic approach to the theory of abstract data types and data structures, Res. Rept. CS-80-22, Waterloo, Ontario, 1980.

[8] G.V. Cudnovskii, Some results in the theory of infinitely long expressions, *Soviet Math. Dokl.* **9** (1968) 556–559.

[9] J.A. Goguen and R.M. Burstall, INSTITUTIONS: Abstract model theory for program specification, Draft version, 1982.

[10] G. Gratzer, *Universal Algebra* (Springer, Berlin, 2nd ed., 1979).

[11] J.V. Guttag, The specification and application to programming of abstract data types, TR CSRG-59, Toronto, 1975.

[12] S. Kamin, Some definitions for algebraic data type specifications, *SIGPLAN Notices* **14** (3) (1979).

[13] S. Kamin, Final data type specifications: A new data type specification method, in: *Proc. 7th POPL-Conference*, 1980.

[14] B. Mahr and J.A. Makowsky, An axiomatic approach to semantics of specification languages, in: *Theoretical Computer Science, 6th GI-Conf. Dortmund 1983*, Lecture Notes in Computer Science **145** (Springer, Berlin, 1983) pp. 211–219.

[15] B. Mahr and J.A. Makowsky, Characterizing specification languages which admit initial semantics, in: *CAAP'83*, Lecture Notes in Computer Science **159** (Springer, Berlin, 1983) pp. 300–316.
[16] J.A. Makowsky, Model theoretic issues in theoretical computer science, in: *Proc. Logic Colloquium '82*, Florence, 1982.
[17] J.A. Makowsky, Compactness, embeddings and definability; and, Abstract embedding relations, *Higher Model Theory: Logic of Mathematical Concepts* (Springer, Berlin, 1983) Chapters 18, 20.
[18] A.I. Mal'cev, Quasiprimitive classes of abstract algebras in the metamathematics of algebraic systems, in: *Studies in Logic* **66** (North-Holland, Amsterdam, 1971) pp. 27–31.
[19] J.D. Monk, *Mathematical Logic* (Springer, Berlin, 1976).
[20] A. Selman, Completeness of calculi for axiomatically defined classes of algebras, *Algebra Universalis* **2** (1) (1972) 20–32.
[21] M. Wand, Final algebra semantics and data type extensions, TR65, Indiana, 1978.