CHARACTERIZING SPECIFICATION LANGUAGES
WHICH ADMIT INITIAL SEMANTICS

by

B. Mahr* and J.A. Makowsky**

Technical Report #232
February 1982

* Visiting from TU Berlin and supported by the MINERVA Foundation for German-Israeli collaboration.

** Supported by the Swiss National Science Foundation, Grant No. 82.820.0.80.
ABSTRACT

The paper proposes a unified approach to study and compare specification techniques and languages. It devides into two parts:

(1) A general notion of specification language is given which provides an axiomatic framework for the various known specification techniques.

(2) Algebraic specification languages which admit initial semantics are characterized up to equivalence. They are shown to be exactly the implicational languages.
INTRODUCTION

The problem of specifying abstract data types has been widely discussed in the literature of the last decade and a great variety of specification techniques have been proposed. Abstract data types are generally considered to be many sorted algebras and since the work of Liskov and Zilles [LZ74], Guttag [Gu75] and ADJ [ADJ75], algebraic methods have entered the field and have led to the concept of algebraic specifications. Specifications are given by sets of equations or equation implications, and the specified data type is identified up to isomorphism as the derived initial algebra (see [ADJ78] for more details). The usefulness of this concept is widely accepted because the semantics of a specification is simple and initiality has great advantages. Mainly to meet the criticism that writing equations is cumbersome and too restrictive in expressive power, many proposals have been made to extend the initial semantics approach to more powerful languages such as first order logic or various programming logics. Practical experience, mainly in software engineering, often shows the need for concepts that can only be expressed clearly in higher order languages (well foundedness, termination, fairness, finiteness, etc.). However, it is not clear that the initial semantics is possible with such specification languages. The initial semantics approach seems, on the contrary, inherently tied to equational logic. The results of this paper give evidence to this.

The purpose of this paper is twofold:
We want to create an appropriate framework to speak about specification languages, to compare their expressive power and to axiomatize their semantic behaviour. This framework is meant to capture only the basic properties which are satisfied by any specification language that is
reasonable. Explicitly we demand closure under isomorphism and re-naming, and compatibility with reducts. In contrast to various logics we do not demand closure under conjunction, negation and quantification. A list of examples is given to show how this framework applies to concrete specification languages that are known in the literature.

Our second purpose is to single out all those specification languages which are algebraic and admit initial semantics. Such will include not only "equations" or "universal Horn formulas", but also the specification language consisting of all infinitary implications of the form

$$\bigwedge_{i \in I} e_i \rightarrow e$$

where $e_i (i \in I)$ and $e$ are equations of terms and $I$ is an arbitrary set.

We call a specification language implicational, if it is powerful enough to specify all varieties and if all specifications consist of a set of (infinitary) implications. Our main result states:

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

The proof of this and of other results in this paper mainly consists in an adaption of two important results in universal algebra (Mal'cev [Ma56], Banaschewski, Herrlich [BH76]) to our framework.
SPECIFICATION LANGUAGES

Roughly speaking, specification languages provide a syntactical and semantical framework to specify classes of algebras as structures. Single structures may also be looked at as a class, the one with only one element. We give an abstract definition of specification language which is based on this idea. We do not incorporate a notion of correctness, which in some examples differs from the semantics of the formalism provided by the language. But in any case a specification language may have a notion of correctness attached. However, as far as comparison of languages is concerned, this additional feature is covered by our reducibility concept in the next section.

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

We assume signatures ordered by $(S, C, F, R) \leq (S', C', F', R')$ if $S \subseteq S'$, $C_s \subseteq C'_s$, $F_w, s \subseteq F'_w, s$ and $R_v, w \subseteq R'_v, w$ for all $s \in S$ and $v, w \in S^*$.

For signatures $\sigma$ and $\tau$ renaming $r: \sigma \rightarrow \tau$ denotes the bijective assignment of $\tau$ to $\sigma$ which is compatible with the ordering of the components of $\tau$ and $\sigma$.

Structures (often called algebras) with respect to a signature $\sigma$ are defined in the usual way. We denote signatures by small greek letters $\sigma, \tau$, and structures by capital letters, A, B, ...
Renaming carries over to structures, and we denote \( A(r) \), with respect to a renaming \( r \), the structure which is identical to \( A \) except that its universe, constants, functions and relations are renamed according to \( r \).

For signatures \( \tau \subseteq \sigma \) we define \( A|_{\tau} \) to be the reduct of the \( \sigma \)-structure \( A \) to signature \( \tau \). By \( \text{Alg}(\tau) \) we denote the class of all \( \tau \)-structures and by \( \text{Alg}_T \) the class \( \bigcup_{\tau \in T} \text{Alg}(\tau) \) for some class of signatures \( T \).

Cardinals we denote by \( \alpha \), the cardinality of the natural numbers by \( \omega \) and we use \( F^*_\tau(\alpha) \) to denote the word algebra with \( \alpha \) generators of signature \( \tau \). Its elements we call terms.

Other notions are standard in universal algebra or logic and we refer to Grätzer [Gr79] for further information.

**Definition:** A specification language \( L = (T,K,L,\vdash) \) consists of the following items:

- a class of signatures \( T \);
- a mapping \( K: T \to 2 \) with \( K(\tau) \subseteq \text{Alg}(\tau) \);
- a mapping \( L: T \to 2^S \) where \( S \) is a class of so called \( L \)-sentences, containing for each \( \tau \in \Sigma \) a sentence \( \perp_\tau \);
- a family \( \models = (\models_{\tau})_{\tau \in T} \) with \( \models_{\tau} \subseteq K(\tau) \times L(\tau) \), such that the following properties hold:

1. \( A \cong B \) implies \( A \in K(\tau) \) iff \( B \in K(\tau) \)
2. \( \gamma: \tau \to \sigma \) renaming implies \( A \in K(\tau) \) iff \( A(r) \in K(\sigma) \)
3. \( \tau \subseteq \sigma \) implies \( L(\tau) \subseteq L(\sigma) \)
4. \( A \cong B \) implies \( A \vdash \varphi \) iff \( B \vdash \varphi \)
5. \( r: \tau \to \sigma \) renaming and \( \varphi \in L(\tau) \) implies existence of a sentence \( \varphi(r) \in L(\sigma) \) such that for all \( A \in K(\tau) \) \( A \vdash \varphi \) iff \( A(r) \vdash \varphi(r) \)
(6) \( \tau \subseteq \sigma, \varphi \in L(\tau) \) and \( A \in K(\sigma) \) implies if \( A|_{\tau} \in K(\tau) \) then

\[ A \models \varphi \iff A|_{\tau} \models \varphi \]

(7) for all \( \tau \in \Sigma \) no \( A \in Alg(\tau) \) satisfies \( A \models \bot \).

We define for a set of \( L \)-sentences \( \phi \subseteq L(\tau) \) that \( A \models \phi \) is \( A \models \varphi \) for all \( \varphi \in \phi \). By \( Mod(\phi) \) we denote the class of structures \( A \) such that \( A \models \phi \). If \( \varphi \subseteq K(\tau) \) then \( \varphi \) is \( L \)-definable if there is \( \phi \subseteq L(\tau) \) such that \( \varphi = Mod(\phi) \). We say \( \varphi \) is \( L, \alpha \)-definable, if there is \( \phi \) with cardinality less than \( \alpha \).

Axioms (1), (2), (4), (5) say that a specification language makes no distinction between isomorphic or renamed structures. Of course this meets the widely accepted representation-free definition of data types or classes of such. Axioms (3) and (6) say that validity of sentences is completely determined in connection with those operations and relations which are named in the signature to which they belong.

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 as even functors - just any object in some formalism which talks about structures. Most naturally, sentences or sets of sentences can be specifications of existing specification languages like CLEAR ([BG80]). Before we illustrate the definition by a number of examples, we continue with some definitions.

Definition: A specification language \( L = (T,K,K,\models) \) is called algebraic, if all \( \tau \in T \) are algebraic and all varieties \( V(\tau) \) which can be specified with less that \( \omega \) equations, are \( L, \omega \)-definable.

An algebraic specification language admits initial semantics, if for all \( \tau \in T \) and \( \phi \subseteq L(\tau) \)

\[ Mod(\phi) \neq \emptyset \] implies \( Mod(\phi) \) has initial object.
where an algebra $A$ is initial in $\text{Mod}(\phi)$, if $A \in \text{Mod}(\phi)$ and for all $B \in \text{Mod}(\phi)$ there is exactly one homomorphism $h : A \rightarrow B$.

If a specification language is algebraic, then for all $\tau \in T$ $\text{Alg}(\tau)$ is definable, since it is the variety specified by the empty set of equations. Thus, in an algebraic specification language $L = (T, K, L, \vdash)$ we have $K(\tau) = \text{Alg}(\tau)$ for all $\tau \in T$.

**Example 1** (many sorted universal algebra):

$L_1 = (T_1, K_1, L_1, \vdash_1)$ with

$T_1$ all algebraic signatures

$K_1(\tau) = \text{Alg}(\tau)$

$L_1(\tau) = (F_\tau(\omega) \times F_\tau(\omega)) \cup \{\bot\}$ equations with variables, where $F_\tau(\omega)$ denotes the free term algebra generated by $\tau$ and $\omega$

$A \vdash_1 \phi$ if $A$ satisfies $\phi$.

In $L_1$ exactly the varieties and the empty class are definable, so $L_1$ is algebraic and admits initial semantics.

**Example 2** (algebraic specification with hidden functions and sorts):

$L_2 = (T_2, K_2, L_2, \vdash_2)$ with

$T_2 = T_1$

$K_2 = K_1$

$L_2(\tau) = (\{\bigcup_{T \in \mathcal{S}} F_\sigma(\omega) \times F_\sigma(\omega)\} \times \{\tau\} \cup \{\bot\})$ a sentence is an equation with arbitrary operation symbols, but marked signature to remember what type of class it is meant to define.

$A \vdash_2 (e, \tau)$ if there is $\sigma$ such that $\tau \subseteq \sigma, e \in F_\sigma(\omega) \times F_\sigma(\omega)$, and there exists $B \in \text{Alg}(\sigma)$ with $B \vdash_1 e$ and $A \cong B|_\tau$.  

Technion - Computer Science Department - Technical Report CS0232 - 1982
Example 3 (first order logic):

$L_3$ is not algebraic and does not admit initial semantics. This follows from the lemma and theorem (Ma) in the next section and the fact that for $\phi \subseteq L(\tau)$ $\text{Mod}(\phi)$ is not necessarily closed under sub-algebras. A counter example is easily constructed:

Let $\phi = \{ (\varphi_1, \tau), \ldots, (\varphi_6, \tau) \}$ be such that $\tau$ is single sorted and contains only one unary operation symbol. Let $\{\varphi_1, \ldots, \varphi_5\}$ define groups with the property $x + x + x = 0$, and $\varphi_6$ the equation $\text{id}(x) = x$. Then $\text{Mod}(\{\varphi_1, \ldots, \varphi_6\})$ contains no algebra with two elements, but an algebra with three elements, say $B$. $B|\tau$ is just the three-element set with identity, which has the two-element set with identity as sub-algebra. But this cannot be reduct of some algebra which satisfies $\{\varphi_1, \ldots, \varphi_6\}$; so it cannot be in $\text{Mod}(\tau)$.

Example 3 (first order logic):

$L_3 = (T_3, K_3, L_3, \models_3)$ with

- $T_3$ all single sorted signatures (thus containing relation symbols!)
- $K_3(\tau) = \text{Alg}(\tau)$
- $L_3(\tau)$ all first order sentences built up from constants, operation and relation symbols of $\tau$ as well as logical constants $\exists, \forall, \sim, \vee, \wedge, \rightarrow$ and $=$
- $A \models_3 \varphi$ if $A$ is model of $\varphi$.

$L_3$ is not algebraic and does not admit initial semantics.

Example 4 (c.f. [CMPPV80]):

$L_4 = (T_4, K_4, L_4, \models_4)$ with

- $T_4 = T_3$
- $K_4(\tau)$ all "functionally reachable" structures and "natural" quotients of such
- $L_4 = L_3$
- $A \models_4 \varphi$ if $A$ is model of $\varphi$. 

Technion - Computer Science Department - Technical Report CS0232 - 1982
In [CMPPV80] it is shown that in those cases, where $\phi \subseteq L_4(\tau)$ possesses the "intersection property for Herbrand interpretations", then $\text{Mod}(\phi)$ has initial object. However, this property has to be proved for each specification $\phi$, if one exceeds Horn formulas.

Example 5 (final data types, confer [K80]):

$L_5 = (T_5, K_5, L_5, \models_5)$ with

- $T_5$ all algebraic signatures
- $K_5(\tau) = \text{Alg}(\tau)$
- $L_5(\tau)$ consists of all pairs $(\text{DS}(\tau), \text{FD}(\tau))$, where $\text{DS}(\tau)$ is a "distinguishing" set for all sort and some operations symbols of $\tau$, and $\text{FD}(\tau)$ is the definition of the remaining operations on the representations given by $\text{DS}(\tau)$; short, it consists of all final data type specifications of signature $\tau$.

A $\models_5 \phi$ if $A$ is isomorphic to the final data type, specified by $\phi$.

For $\phi \subseteq L_5(\tau)$ $\text{Mod}_5(\phi)$ consists exactly of one isomorphism class of algebras, and is empty in case where the sentences in $\phi$ are not pairwise equivalent, i.e. specify the same algebra up to isomorphism.

It is worth noting, that final data type specification defines single data types which then are used to define, independently of their specification, a class of data types, called data abstraction. Any type in the data abstraction is considered to be a correct implementation of the specified data type.

In contrast to this method, algebraic specification defines classes of data types, but is interested only in their initial objects or reducts of such. The members of the specified class are not considered to be correct implementations of the initial member.
Example 6 (generalized implication language):

$L_6 = (T_6, K_6, L_6, \models_6)$ with

$T_6$ all algebraic signatures

$K_6(\tau) = \text{Alg}(\tau)$

$L_6(\tau) = \text{Imp}(\tau, \omega)$ which is defined below

$A \models_6 \phi$ if $A$ satisfies $\phi$.

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

Let $F_\tau(\omega) \times F_\tau(\omega)$ denote the set of equations over $\tau$ and $\omega$ as in Example 2. Then

(1) If $\{e_i | i \in I\}$ is an indexed family of equations and $e$ is a single equation, then $\left( \bigwedge_{i \in I} e_i \right) \rightarrow e \in \text{Imp}(\tau, \omega);

(2) If $\{\varphi_j | j \in J\}$ is an indexed family of elements in $\text{Imp}(\tau, \omega)$ then $\bigwedge_{j \in J} \varphi_j \in \text{Imp}(\tau, \omega)$.

Elements of $\text{Imp}(\tau, \omega)$ can be interpreted as sets of implicational equations with set-many equations in the premises. Their satisfiability is defined in the usual way.

$L_6$ is algebraic, since equations are implicational equations with no premises, and $L_6$ admits initial semantics, which is a corollary of Theorem 2 in the next section.

Characterizing Initial Semantics

In this section we characterize those specification languages which are algebraic and admit initial semantics.

As a tool for comparison of specification languages we introduce the notions of reducibility and equivalence.
Definition: Given specification languages \( L = (T, K, L, \vdash) \) and \( L' = (T', K', L', \vdash') \), a mapping \( C: \Sigma \to 2 \) with \( C(\tau) \subseteq \text{Alg}(\tau) \), and a cardinal \( \alpha \). Then we say

1. \( L \) is \((C, \alpha)\)-reducible to \( L' \), denoted by \( L \prec^\alpha_C L' \) if \( T \subseteq T' \) and for all \( \tau \in T \) and \( \psi \in L_1(\tau) \) there is \( \psi' \in L_2(\tau) \) with \( \text{card}(\psi') < \alpha \) such that
   \[
   \text{Mod}(\tau) \cap C(\tau) = \text{Mod}(\psi') \cap C(\tau)
   \]
   We write \( L \preceq C L' \) if there is an \( \alpha \) such that \( L \prec^\alpha_C L' \) and write \( L \prec^\alpha_C L' \) if the cardinality of \( \psi' \) cannot be bounded by some cardinal.

2. \( L \) is fully \( \alpha \)-reducible to \( L' \), denoted by \( L \prec^\alpha C L' \), if \( L \) is \((C, \alpha)\)-reducible to \( L' \) for all \( C \).
   Analogously, we write \( L \prec L' \) if there is an \( \alpha \) such that \( L \prec^\alpha C L' \) and \( L \prec^\infty C L' \) if there is no bound \( \alpha \).

3. \( L \) is \((C, \alpha)\)-equivalent to \( L' \), denoted by \( L \equiv^\alpha_C L' \), if \( L \prec^\alpha C L' \) and \( L' \prec^\alpha C L \).
   \( L \) is fully \( \alpha \)-equivalent to \( L' \), denoted by \( L \equiv^\alpha C L' \), if \( L \equiv^\alpha C L' \) for all \( C \).
   Analogously, to (1) and (2) we write \( L \equiv C L' \), \( L \equiv^\infty C L' \) or \( L \equiv C L' \) and \( L \equiv^\infty C L' \).

The role of the mapping \( C \) is to restrict comparison of specification languages to algebras which are admitted in \( C \). It turns reducibility into an extremely sensitive tool. The role of the cardinal \( \alpha \) is to give additional information about the relation between specification languages.

Facts: (1) \( L \prec^\alpha_C L' \) implies \( L \prec^\beta_D L' \), for all \( \beta \geq \alpha \) and \( D \) such that \( D(\tau) \subseteq C(\tau) \) for all \( \tau \in T \).
(2) \( L \prec^\alpha L' \) if and only if \( L \prec_C^\alpha L' \), with \( C \) such that 
\[ C(\tau) \supseteq K(\tau) \cup K'(\tau) \] 
for all \( \tau \in T \).

(3) For any cardinality \( \alpha > \omega \), \( \leq^\alpha_C \) is a reflexive and transitive relation, and \( \equiv^\alpha_C \) is an equivalence relation. If \( \alpha \leq \omega \) this is true if sentences are closed under conjunction.

(4) \( L \prec^\alpha L' \) and \( L \) algebraic implies \( L' \) algebraic.

(5) \( L \prec^\alpha L' \) and \( L' \) admits initial semantics implies \( L \) admits initial semantics.

Let us call \( L \) initially \( \alpha \)-reducible to \( L' \), if \( L \prec_C^\alpha L' \) with 
\[ C(\tau) = \{ A \in \text{Alg}(\tau) | A \text{ is initial in } \{A\} \} \]

Note, this definition makes sense because of the following theorem (stated in [Gr79], p.165 for single sorted algebras).

Theorem ([Gr79]): Let \( A \) be an algebra. Then the following three conditions are equivalent:

(1) \( A \) is free algebra in some class \( K \)

(2) \( A \) is free algebra in an equational class \( K \)

(3) \( A \) is free algebra in the class \( \{A\} \).

Using this theorem one easily sees the following:

Theorem 1: Any algebraic specification language which admits initial semantics is initially equivalent to the language \( L_1 \) (see Example 1).

If an algebraic specification language which admits initial semantics, only serves the purpose to define initial algebras, then nothing in the expressive power is lost, if it is replaced by \( L_1 \); this is the interpretation of Theorem 1. Of course, in practice, a specification language intends to give more information, namely in regard to
correctness and implementations. In general, it defines classes of algebras and not just a single initial algebra. The following theorem, indeed, takes also this point of view into consideration. It is based on two theorems, adapted to the many sorted case, and one lemma:

**Theorem ([Ma56]):** A class \( K \) of algebras is free if and only if \( K \) contains all subalgebras and direct products of its members.

Classes of algebras are generally assumed to be nonempty and closed under isomorphic images. A class is free, if the intersection of \( K \) with any variety \( V \) of the same type contains free objects of arbitrary rank, provided \( K \cap V \neq \emptyset \).

**Theorem ([BH76]):** A class \( K \) of algebras is implicational if and only if \( K \) contains all subalgebras and direct products of its members.

A class \( K \) is **implicational** if it is definable by a sentence of \( L_6 \) (see Example 6).

**Lemma:** Let \( L \) be an algebraic specification language. Then \( L \) admits initial semantics if and only if every \( L \)-definable class is free, provided it is nonempty.

**Proofidea:** If \( \text{Mod}(\phi) \) is nonempty and is free, then it contains an initial object.

If \( L \) admits initial semantics, then any nonempty \( \text{Mod}(\phi) \) is free: construct the free algebras as reducts of initial algebras in appropriate classes \( K(\text{Mod}(\phi)) \) by interpreting the generating elements as zeroary operations.

Putting these theorems and the lemma together, one obtains easily

**Theorem 2:** Any algebraic specification language which admits initial semantics is fully equivalent to an implicational specification language.
We call a specification language \( L = (T, K, L, \to) \) implicational if the following properties are satisfied:

1. \( L \) algebraic
2. \( L(\tau) \subseteq \text{Imp}(\tau, \omega) \) (c.f. Example 6)
3. \( A \models \varphi \) if \( \varphi \) is valid in \( A \).

Using the classical characterization, adapted to the many sorted case:

**Theorem:** If \( K \) is an axiomatic class of algebras, then \( K \) is definable by a set of basic Horn formulas if and only if \( K \) contains all sub-algebras and direct products of its members.

Where \( K \) is axiomatic if it is definable in \( L_3 \) (see Example 3, c.f. [Se72]), we obtain:

**Theorem 3:** Any algebraic specification language which admits initial semantics and is fully \( \omega \)-reducible to \( L_3 \), is fully equivalent to a Horn specification language.

We call a specification language \( L = (T, K, L, \to) \) Horn if the following properties are satisfied:

1. \( L \) algebraic
2. \( L(\tau) \) contains only basic Horn formulas
3. \( A \models \varphi \) if \( \varphi \) is model of \( \varphi \).

Theorem 3 uses in contrast to Theorem 2 an additional assumption but concludes that finitary formulas are enough for specifications (basic Horn formulas are finite conjunctions of atomic formulas or their negation, where at most one is atomic).

As a consequence of Theorem 2 we have that an implicational specification language admits initial semantics, which from what was
known about implications, is certainly not surprising. But the theorem also states that no other algebraic language which admits initial semantics, has more expressive power. This, in fact, shows that any attempt to increase the expressive power beyond implicational languages, but with the desired property of admitting initial semantics preserved, must fail.

Finally, in this section, we state the following theorem which can be concluded from Mal'cev's theorem and a closer look at the proofs of Theorems 1 and 2.

**Theorem 4:** The specification languages which admit initial semantics form with respect to union and intersection, up to equivalence, a complete lattice which has \( L_1 \) as smallest and \( L_6 \) as largest element.

\[ \square \]

Remark: We have concentrated in this section on the characterization of algebraic specification languages which admit initial semantics. The tool of reducibility, however, also allows us to relate concrete languages in the way Kamin ([K79]) and Bergstra & Tucker ([BT79]) have done it, for example.

---

**ACKNOWLEDGEMENT**

We thank John T. Baldwin for pointing out to us [Ma56] and Brian Mayoh for fruitful discussions.
REFERENCES


