1. Introduction

During recent years the concept of algebraic specification (cf. e.g. /Guttag 75/, /Burstall, Goguen 77/, /Goguen et al. 78/, /Wirsing et al. 80/) has proved to be a powerful and flexible tool for the formal definition of data structures. Algebraic concepts have also been employed for the specification of programming language semantics, e.g. first-order identities (/Wand 77/) or continuous algebras (/Courcelle, Nivat 78/, /Goguen et al. 77/). In contrast to these "explicit" constructions of semantics, /Broy, Wirsing 80a/ have introduced a technique for characterizing the semantic models of a language by the axioms of an algebraic type without resorting to (the isomorphism class of) a fixed model. This approach is characterized by the following peculiarities (cf. /Wirsing et al. 80/, /Broy, Wirsing 80c/):

(1) As semantic models, finitely generated (cf. /Bauer, Wössner 81/) heterogeneous algebras with partial operations are considered.

(2) Their properties are specified in algebraic types using positive conditional equations and a definedness predicate \( D \) on the terms of the type.

(3) Within the equations, the metasymbol \( = \) is interpreted as strong equality, and \( D \) is total, so that the underlying logic remains two-valued.

(4) In general, the types are hierarchical, i.e. a type may be based on a subtype which is considered as the specification of primitive objects and operations. Models of hierarchical types are required to preserve the properties of the primitive type.

*) This research was carried out within the Sonderforschungsbereich 49, Programmiertechnik, Munich.
(5) All models of a type are considered. In order to compare them, especially with respect to the definedness of operations, partial and strong homomorphisms are used.

In the sequel we illustrate this approach with the specifications of two layers of a programming language. The first layer comprises simple statements based on integer arithmetic; its programs always terminate. Whereas this language layer can also be treated in the conventional framework of total algebras, the second layer containing general jumps introduces the possibility of non-termination and thus needs the generalized setting of partial algebras.

For each of the layers we analyse the class of semantic models, especially with respect to extremal (initial and terminal) algebras. Each semantic model is characterized by a special equivalence between the terms denoting programs; these equivalences range from textual identity over different operational equivalences to functional (mathematical) equivalence. Models corresponding to this latter equivalence (fully abstract models in the sense of /Milner 77/) can be constructed using the techniques of denotational semantics (cf. /Scott, Strachey 71/).

Due to lack of space we present only a short analysis of each specification and omit proofs. For an extended treatment we refer to /Broy et al. 81/.

2. Basic Definitions

In the sequel we give the most important definitions; for a detailed treatment compare e.g. /Wirsing et al. 80/, /Broy, Wirsing 80a, c/.

2.1. Partial Algebras

For a signature \( \Sigma = (S, F) \) a partial heterogeneous \( \Sigma \)-algebra \( A \) is defined analogously to a total one; however, the operation symbols in \( F \) may also be interpreted by partial functions. The term algebra is denoted by \( W_\Sigma \) and the interpretation of a term or an operation symbol \( t \) in \( A \) is denoted by \( t^A \). A \( \Sigma \)-algebra is called finitely generated if all elements of its carrier sets can be obtained as interpretations of terms in \( W_\Sigma \). Given two \( \Sigma \)-algebras \( A, B \), a family \( \varphi = (\varphi_s : s^A \to s^B) \in S \) of partial functions is called a partial \( \Sigma \)-homomorphism if for all operation symbols \( f : s_1 \times \ldots \times s_n \to s \) in \( F \) and all arguments \( a_i \in s^A_i \) (\( i = 1, \ldots, n \))

\[ f^A(a_1, \ldots, a_n) \text{ defined} \Rightarrow [f^B(\varphi_{s_1}(a_1), \ldots, \varphi_{s_n}(a_n)) \text{ defined and} \]

\[ \varphi_{s_i}(f^A(a_1, \ldots, a_n)) = f^B(\varphi_{s_1}(a_1), \ldots, \varphi_{s_n}(a_n)) \]