Algebraic Properties of Interfaces

Michael Löwe, Harald König, and Christoph Schulz

Fachhochschule für die Wirtschaft (FHDW),
Hannover, Germany
michael.loewe@fhdw.de

Abstract. Interfaces became one of the most important features in modern software systems, especially in object-oriented systems. They provide for abstraction and detail hiding and, therefore, contribute to readable and reusable software construction. Formal specification methods have also addressed interfaces and provided formal semantics to them. These semantics are always based on some forgetful and/or restriction constructions. The main focus has been laid on integrating these constructions with free or behaviourable constructions in order to provide some combined semantics for specification modules.
In this paper, we investigate the algebraic properties of "forgetting" and "restricting". We define two different notions of model category for algebraic specification with explicit interface signatures, so-called interface specifications. The first one uses forgetful constructions only, the second one integrates the restriction to the part reachable by the interface.
We investigate closure properties of these categories w.r.t. subalgebras, products and directed limits. This analysis provides the first result, namely that interface specifications with Horn-axioms do not exceed the expressiveness of Horn-axiom specifications without interfaces. The inverse is more interesting. We show that each category specified by Horn-axioms can be specified by simple equations on possibly hidden operations. The constructive proof for this theorem leads to the next main result that every class specified by an interface specification with restriction semantics possesses an interface specification using forgetful semantics only.

1 Introduction and Preliminaries

Since the early days of algebraic specification research (see [4, 6]), it is well-known that some specification problems cannot be solved using a finite number of equations or Horn-axioms only ([2, 8, 9, 11]). There is no way to avoid some hidden operators for a finite solution. These results gave rise to a theory of implementation relations between abstract data types ([1]). The main mechanism here is a forgetful and/or restriction semantics in order to semantically model the realisation of target or export data types by some source or import data types. In these frameworks, the sorts and operations of the source types are considered hidden. In [3], specification and implementation methods are combined within one concept: algebraic module specifications. Semantics of these models consist of free constructions together with some forgetful and restriction steps. Algebraic

H.-J. Kreowski et al. (Eds.): Formal Methods (Ehrig Festschrift), LNCS 3393, pp. 190–203, 2005.

module specifications are able to provide semantics to access control concepts in object oriented languages as they are defined by "private"- or "protected"-keywords in Java ([5]) or UML ([10]) for example.

In this paper, we investigate forgetful and restriction semantics, as well. But we use a somewhat different approach. We are not interested in point-wise construction of export models from import models as it is done in [3]. We put the main emphasis on categories of algebras which are designed as model classes for algebraic specification with explicit export interfaces, so-called interface specifications. We show that, by the right choice of these model classes, data generating, forgetful, and restriction steps can be combined into a single free construction. Thus, the semantics of algebraic data types with hidden operations can be presented as a simple free construction, which possesses the well-known categorical compositional properties [7]. The paper is organised as follows.

The section "Interface Specifications" introduces syntax and semantics for algebraic specifications with explicit interfaces. We provide two kinds of semantics: forgetful semantics and restriction semantics. We show that forgetful model categories are not closed with respect to subalgebras. Therefore, they are no candidate for initial semantics. Restriction semantics is derived from forgetful semantics just by repairing this deficiency.

The section "Closure Properties" investigates algebraic closure properties of model classes with restriction semantics in the spirit of [12]. We address closure w.r.t. products, directed limits, and subalgebras using Horn-axioms as formulas[1]. The results of this section characterise restriction classes as classes which can be specified by Horn-axioms without hidden operators. They also guarantee the existence of initial semantics, which are explicitly constructed in section "Free Construction". Here the name restriction semantics is justified by showing that free constructions into restriction classes are composed of term-generating, equivalence factorisation, forgetful, and restriction steps.

Section "Interface Specifications with Equations" investigates the subset of interface specifications with restriction semantics where all axioms are equations. We show that for each Horn-specification without hidden operators there is one with hidden operators using equations only. The constructive proof provides another interesting corollary, namely that for each restriction class, there is an interface specification with forgetful semantics.

The "Summary and Outlook"-section provides an overview of all results contained in this paper and an outlook on parametric interface specifications as a future research topic.

We use the following notions and notations. A *signature* $SIG = (S, OP)$ consists of a set S of sorts and a family $OP = (OP_{w,s})_{w \in S^*, s \in S}$ of sets of operator symbols. With $op : s_1 s_2 \cdots s_n \to s$ we denote domain and codomain for operators $op \in OP_{s_1 s_2 \cdots s_n, s}$.

[1] We restrict ourselves to Horn-axioms, since they are the most expressive type of formulas that (1) are finite in all components and (2) provide an iterative access to the specified equivalence. Hence, they are the right choice on the design level of software systems, which is addressed by interface specifications, too.