BUILDING SPECIFICATIONS IN AN ARBITRARY INSTITUTION

Donald Sannella and Andrzej Tarlecki*

Department of Computer Science
University of Edinburgh

Abstract

A set of operations for constructing algebraic specifications in an arbitrary logical system is presented. This builds on the framework provided by Goguen and Burstall's work on the notion of an *institution* as a formalisation of the concept of a logical system for writing specifications. We show how to introduce free variables into the sentences of an arbitrary institution and how to add quantifiers which bind them. We use this foundation to define a set of primitive operations for building specifications in an arbitrary institution based loosely on those in the ASL kernel specification language. We examine the set of operations which results when the definitions are instantiated in an institution of first-order logic and compare these with the operations found in existing specification languages. The result of instantiating the operations in an institution of partial first-order logic is also discussed.

1 Introduction

Much work has been done on algebraic specifications in the past ten years. Although much has been accomplished, there is still no general agreement on the definitions of many of the basic concepts, e.g. signature and algebra, and on which kinds of axioms should be used. The disagreement arises partly because different definitions are required to treat various special issues in specification, such as errors [Gog 77, GDLE 82], coercions [Gog 78] and partial operations [BrW 82]; partly because some specification methods such as the initial algebra approach [ADJ 76] only work under certain restrictions on e.g. the form of axioms in specifications; and partly because of disagr*ements over matters of style or taste. These fundamental differences lead to difficulty in comparing the results achieved by different approaches and in building upon the work of others.

The notion of an *institution* [GB 83] provides a tool for unifying all these different approaches to specification by formalising the concept of a logical system for writing specifications. An institution comprises definitions of signature, model, sentence (i.e. axiom) and satisfaction which obey a few internal consistency conditions (details in section 2). Although it is often not obvious, much of the work which has been done on algebraic specification turns out to be independent of the particular definitions of these four notions. In such cases it would be highly desirable to make the generality explicit by basing everything on an arbitrary institution. This was done in the semantics of the Clear specification language [BG 80] (where an institution was called a "language"). Sometimes additional assumptions about the base institution are necessary, as in Clear where use of the initial algebra approach requires the assumption that the institution is liberal (forgetful functors induced by theory morphisms have left adjoints). Instantiating the base institution in different ways (and changing the low-level syntax accordingly) yields a family of specification languages: equational Clear, error Clear, continuous Clear and so on.

In early work on algebraic specification (e.g. [ADJ 76]) it was shown how a collection of

*On leave from Institute of Computer Science, Polish Academy of Sciences, Warsaw.

algebras could be specified by a *theory*, i.e. a signature together with a set of axioms. For small specifications such an approach is adequate, but it is more convenient to build large and complex specifications in a structured way by putting together small specifications. Several specification languages in addition to Clear support such a structured approach to specification. These include CIP-L [Bau 81], LOOK [ZLT 82, ETLZ 82], ASL [Wir 82, SW 83, Wir 83] and the constraint language of [EWT 83]. None of these other languages were based on an arbitrary institution (although the possibility of a similar such generalisation was considered in [SW 83] and [EWT 83]) and so they are not general in the sense that Clear is. However, since they include features which seem desirable but which are not included in Clear, they are more useful as tools for writing specifications in the particular institutions they treat. Most useful of all would be an institution-based specification tool which incorporates the good ideas of all these languages. That is the goal of this paper. We define a set of general specification-building operations based loosely (but not exclusively) on those in ASL.

One novel feature of ASL is a specification-building operation **abstract** which can be used to *behaviourally abstract* from a specification, closing its collection of models under behavioural equivalence [GGM 76, Rei 81]. This allows *abstract model specifications* [LB 77], cf. [Suf 82] in which the desired behaviour is described in some concrete way, e.g. by giving a simple model which exhibits it. Such an operation is a necessary ingredient in an algebraic specification language (as discussed in [San 83]) since the specification of e.g. an abstract data type is supposed to describe a behaviour (an input/output relation) without regard to the particular representation used and therefore *all* algebras which realise the desired behaviour should be permitted as models. Furthermore, using conventional specification languages which lack operations like **abstract**, it is in general difficult (as in Clear) or impossible (as in the initial algebra approach of [ADJ 76] and the final algebra approach of [Wand 79]) to describe collections of models which are closed under behavioural equivalence since such a collection may contain a wide range of non-isomorphic algebras. However, this operation was defined in [SW 83] in such a way that it is not obvious how to generalise it to an arbitrary institution. (There are some remarks in [SW 83] which suggest how this might be done, but the proposed generalisation does not fit smoothly into the institutional framework and anyway the technical details are wrong.) The discussion of **abstract** in the general case is the main contribution of this paper.

The key to the institution-based definition of **abstract** turns out to be the introduction of free variables into the sentences of the institution. We show in section 3 how this may be accomplished. Free variables are necessary because they provide a way of naming unreachable elements of models which cannot be referred to using the operations of the model alone. Such elements play an important role in the definition of behavioural abstraction. Having introduced free variables into the sentences of an institution, we digress in the second part of section 3 and show how to add quantifiers which bind them. This gives a construction for introducing quantified variables into the sentences of an arbitrary institution.

Building on this foundation, we then define a set of primitive operations for building specifications in an arbitrary institution (section 4). The set of operations we provide is based on those present in ASL -- however, there are a number of significant differences. These derive both from difficulties in generalising some of the operations of ASL to an arbitrary institution (for example, since we cannot easily form the union of signatures in this general setting the + operation is not generalised directly) and from extensions which arose naturally in the process of generalisation. One gap here (mainly due to space limitations) is the absence of a mechanism for defining or applying parameterised specifications, although an appropriate parameterisation mechanism should not be difficult to generalise to an arbitrary institution. A feature of ASL which remains is the expressive power and flexibility necessary to provide a kernel for building high-level specification languages. The convenient-to-use specification-building operations of the high-level language would be defined by composing these low-level operations. It would be natural for such a high-level language to hide some of the raw power of the primitives from the user.

In section 5 we examine the set of operations which results when the general definitions are instantiated in an institution of first-order logic with equality as the only predicate. These operations are compared with those found in existing specification languages. In particular, the operations of ASL can be expressed easily in terms of the operations we obtain, but not vice versa. The result of instantiating the operations in an institution of partial first-order logic is considered in section 6. The resulting set of operations is compared with the operations of the early version of ASL in [Wir 82] which also used partial algebras.

We assume some familiarity with a few notions from basic category theory, although no use is made of any deep results. See [AM 75] or [MacL 71] for the definitions which we omit here.

2 Institutions

Following [GB 83] we introduce institutions to formalise the notion of a logical system for writing specifications. An institution consists of a collection of signatures together with for any signature Σ a set of Σ-sentences, a collection of Σ-models and a satisfaction relation between Σ-models and Σ-sentences. Note that signatures are arbitrary abstract objects in this approach, not necessarily the usual "algebraic" signatures used in many standard approaches to algebraic specification (see e.g. [ADJ 76]). The only "semantic" requirement is that when we change signatures, the induced translations of sentences and models preserve the satisfaction relation. This condition expresses the intended independence of the meaning of a specification from the actual notation. Formally:

Def [GB 83]: An *institution* INS consists of:

- A category $\mathrm{Sign}_{\mathrm{INS}}$ (of signatures)
- A functor $\mathbf{Sen}_{\mathrm{INS}}: \mathrm{Sign}_{\mathrm{INS}} \to \mathbf{Set}$ (where Set is the category of all sets; $\mathbf{Sen}_{\mathrm{INS}}$ gives for any signature Σ the set of Σ-sentences and for any signature morphism $\sigma: \Sigma \to \Sigma'$ the function $\mathrm{Sen}_{\mathrm{INS}}(\sigma): \mathrm{Sen}_{\mathrm{INS}}(\Sigma) \to \mathrm{Sen}_{\mathrm{INS}}(\Sigma')$ translating Σ-sentences to Σ'-sentences)
- A functor $\mathbf{Mod}_{\mathrm{INS}}: \mathrm{Sign}_{\mathrm{INS}} \to \mathrm{Cat}^{\mathrm{op}}$ (where Cat is the category of all categories;* $\mathrm{Mod}_{\mathrm{INS}}$ gives for any signature Σ the category of Σ-models and for any signature morphism $\sigma: \Sigma \to \Sigma'$ the σ-reduct functor $\mathrm{Mod}_{\mathrm{INS}}(\sigma): \mathrm{Mod}_{\mathrm{INS}}(\Sigma') \to \mathrm{Mod}_{\mathrm{INS}}(\Sigma)$ translating Σ'-models to Σ-models)
- A satisfaction relation $\models_{\Sigma,\mathrm{INS}} \subseteq |\mathrm{Mod}_{\mathrm{INS}}(\Sigma)| \times \mathrm{Sen}_{\mathrm{INS}}(\Sigma)$ for each signature Σ,

such that for any signature morphism $\sigma: \Sigma \to \Sigma'$ the translations $\mathrm{Mod}_{\mathrm{INS}}(\sigma)$ of models and $\mathbf{Sen}_{\mathrm{INS}}(\sigma)$ of sentences preserve the satisfaction relation, i.e. for any $\phi \in \mathrm{Sen}_{\mathrm{INS}}(\Sigma)$ and $M' \in |\mathbf{Mod}_{\mathrm{INS}}(\Sigma')|$

$$M' \models_{\Sigma',\mathrm{INS}} \mathrm{Sen}_{\mathrm{INS}}(\sigma)(\phi) \quad \text{iff} \quad \mathrm{Mod}_{\mathrm{INS}}(\sigma)(M') \models_{\Sigma,\mathrm{INS}} \phi \qquad \text{(Satisfaction condition)}$$

To be useful as the underlying institution of a specification language, an institution must provide some tools for "putting things together". Thus, in this paper we additionally require

*Of course, some foundational difficulties are connected with the use of this category, as discussed in [MacL 71]. We do not discuss this point here, and we disregard other such foundational issues in this paper; in particular, we use the term "collection" throughout to denote "sets" which may be too large to really be sets.

that the category Sign has pushouts and initial objects (i.e. is finitely cocomplete) and moreover that **Mod** preserves pushouts and initial objects (and hence finite colimits), i.e. that **Mod** translates pushouts and initial objects in **Sign** to pullbacks and terminal objects (respectively) in **Cat**.

In [GB 83] the category Sign is not required to be cocomplete, but this is required there of any institution to be used as the basis of a specification language (as in Clear [BG 80]). **Mod** is not required there to preserve colimits, however we feel that this is a natural assumption to make the semantics of specification-building operations consistent with our intuitions. A similar but (apparently) stronger condition is required in [EWT 83]. Note that both of these requirements are entirely independent of the "logical" part of the institution, i.e. of sentences and the satisfaction relation, and the fact that all examples of institutions we can think of (including all those in [GB 83]) satisfy them indicates that they are not very restrictive in practice.

The work of [Bar 74] on abstract model theory is similar in intent to the theory of institutions but the notions used and the conditions they must satisfy are more restrictive and rule out many of the examples we would like to deal with.

Notational conventions

- The subscript INS is omitted when there is no danger of confusion.
- We will write $\models$ instead of $\models_\Sigma$ when Σ is obvious.
- For any signature morphism $\sigma:\Sigma\to\Sigma'$, **Sen**(σ) is denoted just by σ and **Mod**(σ) is denoted by $\_|_\sigma$ (i.e. for $\phi\in$**Sen**(Σ), $\sigma(\phi)$ stands for **Sen**$(\sigma)(\phi)$, and e.g. for $M'\in|$**Mod**$(\Sigma')|$, $M'|_\sigma$ stands for **Mod**$(\sigma)(M')$).
- For any signature Σ, $\Phi\subseteq$**Sen**(Σ) and $M\in|$**Mod**$(\Sigma)|$, we write $M\models\Phi$ to denote that $M\models\phi$ for all $\phi\in\Phi$.

Example: the institution GEQ of ground equations

An *algebraic signature* is a pair $\langle S,\Omega\rangle$ where S is a set (of sort names) and Ω is a family of sets $\{\Omega_{w,s}\}_{w\in S^*,s\in S}$ (of operation names). We write $f:w\to s$ to denote $w\in S^*$, $s\in S$, $f\in\Omega_{w,s}$. An *algebraic signature morphism* $\sigma:\langle S,\Omega\rangle\to\langle S',\Omega'\rangle$ is a pair $\langle\sigma_{sorts},\sigma_{opns}\rangle$ where $\sigma_{sorts}:S\to S'$ and σ_{opns} is a family of maps $\{\sigma_{w,s}:\Omega_{w,s}\to\Omega'_{\sigma^*(w),\sigma(s)}\}_{w\in S^*,s\in S}$ where $\sigma^*(s1,\dots,sn)$ denotes $\sigma_{sorts}(s1),\dots,\sigma_{sorts}(sn)$ for $s1,\dots,sn\in S$. We will write $\sigma(s)$ for $\sigma_{sorts}(s)$, $\sigma(w)$ for $\sigma^*(w)$ and $\sigma(f)$ for $\sigma_{w,s}(f)$, where $f\in\Omega_{w,s}$.

The category of algebraic signatures **AlgSig** has algebraic signatures as objects and algebraic signature morphisms as morphisms; the composition of morphisms is the composition of their corresponding components as functions. (This obviously forms a category.)

Let $\Sigma=\langle S,\Omega\rangle$ be an algebraic signature.

A Σ-*algebra* A consists of an S-indexed family of carrier sets $|A| = \{|A|_s\}_{s\in S}$ and for each $f:s1,\dots,sn\to s$ a function $f_A:|A|_{s1}\times\dots\times|A|_{sn}\to|A|_s$. A Σ-*homomorphism* from a Σ-algebra A to a Σ-algebra B, $h:A\to B$, is a family of functions $\{h_s\}_{s\in S}$ where $h_s:|A|_s\to|B|_s$ such that for any $f:s1,\dots,sn\to s$ and $a_1\in|A|_{s1},\dots,a_n\in|A|_{sn}$ $h_s(f_A(a_1,\dots,a_n)) = f_B(h_{s1}(a_1),\dots,h_{sn}(a_n))$.

The category of Σ-algebras **Alg**(Σ) has Σ-algebras as objects and Σ-homomorphisms as morphisms; the composition of homomorphisms is the composition of their corresponding components as functions. (This obviously forms a category.)

For any algebraic signature morphism $\sigma:\Sigma\to\Sigma'$ and Σ'-algebra A', the σ-*reduct* of A' is the Σ-algebra $A'|_\sigma$ defined as follows:

- For $s\in S$, $|A'|_\sigma|_s =_{def} |A'|_{\sigma(s)}$.

- For $f: w \to s$ in Σ, $f_{A'|_\sigma} =_{def} \sigma(f)_{A'}$.

Similarly, for a Σ'-homomorphism $h': A' \to B'$ where A' and B' are Σ'-algebras, the σ-reduct of h' is the Σ-homomorphism $h'|_\sigma: A'|_\sigma \to B'|_\sigma$ defined by $(h'|_\sigma)_s =_{def} h'_{\sigma(s)}$ for $s \in S$.

The mappings $A' \mapsto A'|_\sigma$, $h' \mapsto h'|_\sigma$ form a functor from $Alg(\Sigma')$ to $Alg(\Sigma)$.

For any algebraic signature Σ, $Alg(\Sigma)$ contains an initial object T_Σ which is (to within isomorphism) the algebra of ground Σ-terms, i.e. the carriers $|T_\Sigma|$ contain terms of the appropriate sorts which are constructed using the operation symbols of Σ (without variables) and the operations in T_Σ are defined in the natural way (see e.g. [ADJ 76]). A *ground Σ-equation* is a pair $\langle t, t' \rangle$ (usually written as $t=t'$) where t, t' are ground Σ-terms of the same sort, i.e. $t, t' \in |T_\Sigma|_s$ for some sort s of Σ.

By definition, for any Σ-algebra A there is a unique Σ-homomorphism $h: T_\Sigma \to A$. For any ground term $t \in |T_\Sigma|_s$ (for s in the sorts of Σ) we write t_A rather than $h_s(t)$ to denote the value of t in A. For any Σ-algebra A and ground Σ-equation $t=t'$ we say that $t=t'$ *holds* in A (or A *satisfies* $t=t'$) written $A \models t=t'$, if $t_A = t'_A$.

Let $\sigma: \Sigma \to \Sigma'$ be an algebraic signature morphism. The unique Σ-homomorphism $h: T_\Sigma \to T_{\Sigma'}|_\sigma$ determines a translation of Σ-terms to Σ'-terms. For a ground Σ-term t of sort s we write $\sigma(t)$ rather than $h_s(t)$. This in turn determines a translation (again denoted by σ) of ground Σ-equations to ground Σ'-equations: $\sigma(t=t') =_{def} \sigma(t)=\sigma(t')$.

All the above notions combine to form the institution of ground equations GEQ:

- $Sign_{GEQ}$ is the category of algebraic signatures **AlgSig**.
- For an algebraic signature Σ, $Sen_{GEQ}(\Sigma)$ is the set of all ground Σ-equations; for an algebraic signature morphism $\sigma: \Sigma \to \Sigma'$, $Sen_{GEQ}(\sigma)$ maps any ground Σ-equation $t=t'$ to the ground Σ'-equation $\sigma(t)=\sigma(t')$.
- For an algebraic signature Σ, $Mod_{GEQ}(\Sigma)$ is $Alg(\Sigma)$; for an algebraic signature morphism $\sigma: \Sigma \to \Sigma'$, $Mod_{GEQ}(\sigma)$ is the functor $\_|_\sigma: Alg(\Sigma') \to Alg(\Sigma)$.
- For an algebraic signature Σ, $\models_{\Sigma, GEQ}$ is the satisfaction relation as defined above.

It is easy to check that GEQ is an institution (the satisfaction condition is a special case of the Satisfaction Lemma of [BG 80]). The category **AlgSig** is finitely cocomplete (see [GB 78] Prop. 5) and $Mod_{GEQ}: \mathbf{AlgSig} \to \mathbf{Cat}^{op}$ translates finite colimits in **AlgSig** to finite limits in **Cat** (see [BW 82]).

For some further examples of institutions see [GB 83].

3 Free variables in institutions

In logic, formulae may contain free variables (such formulae are called *open*). To interpret an open formula, we have to provide not only an interpretation for the symbols of the underlying signature (a model) but also an interpretation for the free variables (a valuation of variables into the model). This provides a natural way to deal with quantifiers. The need for open formulae also arises in the study of specification languages. In fact, we will need them to define one of the specification-building operations (**abstract**) in the next section. But for this we need institutions in which sentences may contain free variables.

Fortunately we do not have to change the notion of institution -- we can provide open formulae in the present framework (this idea was influenced by the treatment of variables in [Bar 74]). Note that we use here the term "formula" rather than "sentence", which is reserved for the sentences of the underlying institution.

Consider the institution GEQ of ground equations. Let $\Sigma = \langle S, \Omega \rangle$ be an algebraic signature.

For any S-indexed family of sets, $X=(X_s)_{s\in S}$, define $\Sigma(X)$ to be the extension of Σ by the elements of X as new constants of the appropriate sorts.

Now, any sentence over $\Sigma(X)$ may be viewed as an open formula over Σ with free variables X. Given a Σ-algebra A, to determine whether an open Σ-formula with variables X holds in A we have first to fix a valuation of variables X into |A|. Such a valuation corresponds exactly to an extension of A to a $\Sigma(X)$-algebra, which additionally contains an interpretation of the constants X.

Given a translation of sentences along an algebraic signature morphism $\sigma:\Sigma\to\Sigma'$ we can extend it to a translation of open formulae. Roughly, we translate an open Σ-formula with variables X, which is a $\Sigma(X)$-sentence, to the corresponding $\Sigma'(X')$-sentence, which is an open Σ'-formula with variables X'. Here X' results from X by an appropriate renaming of sorts determined by σ (we also have to avoid unintended "clashes" of variables and operation symbols).

The above ideas generalise to an arbitrary institution INS.

Let Σ be a signature.

Any pair $\langle\phi,\theta\rangle$, where $\theta:\Sigma\to\Sigma'$ is a signature morphism and $\phi\in\mathrm{Sen}(\Sigma')$, is an open Σ-formula with variables "$\Sigma'-\theta(\Sigma)$". (Note the quotation marks -- since $\Sigma'-\theta(\Sigma)$ makes no sense in an arbitrary institution, it is only meaningful as an aid to our intuition.) When we use open formulae in specifications we will omit θ if it is obvious from the context.

If M is a Σ-model, $M\in|\mathrm{Mod}(\Sigma)|$, then a valuation of variables "$\Sigma'-\theta(\Sigma)$" into M is a Σ'-model $M'\in|\mathrm{Mod}(\Sigma')|$ which is a θ-extension of M, i.e. $M'|_\theta=M$.

Note that in the standard logical framework there may be no valuation of a set of variables into a model containing an empty carrier. Similarly, here a valuation need not always exist (although there may be more reasons for that). For example, in GEQ if θ is not injective then some models have no θ-extension.

If $\sigma:\Sigma\to\Sigma1$ is a signature morphism and $\langle\phi,\theta\rangle$ is an open Σ-formula then we define the translation of $\langle\phi,\theta\rangle$ along σ as $\sigma(\langle\phi,\theta\rangle)=_{\mathrm{def}}\langle\sigma'(\phi),\theta'\rangle$, where

$$\begin{tikzcd}
\Sigma' \arrow[r, "\sigma'"] & \Sigma1' \\
\Sigma \arrow[u, "\theta"] \arrow[r, "\sigma"'] & \Sigma1 \arrow[u, "\theta'"']
\end{tikzcd}$$

is a pushout in the category of signatures.

There is a rather subtle problem we have to point out here: pushouts are defined only up to isomorphism, so strictly speaking the translation of open formulae is not well-defined. Fortunately, from the definition of an institution one may easily prove that whenever $\iota:\Sigma1'\to\Sigma1''$ is an isomorphism in Sign with inverse ι^{-1} then $\mathrm{Sen}(\iota):\mathrm{Sen}(\Sigma1')\to\mathrm{Sen}(\Sigma1'')$ is a bijection, $\mathrm{Mod}(\iota):\mathrm{Mod}(\Sigma1'')\to\mathrm{Mod}(\Sigma1')$ is an isomorphism in Cat and moreover for any $\Sigma1'$-sentence $\psi\in\mathrm{Sen}(\Sigma1')$ and any $\Sigma1'$-model $M1'\in|\mathrm{Mod}(\Sigma1')|$

$$M1'\models\psi \quad \text{iff} \quad M1'|_{\iota^{-1}}\models\iota(\psi)$$

This shows that (at least for semantic analysis) we can pick out an arbitrary pushout to define the translation of open formulae and so we may safely accept the above definition of translation.

Note that sometimes we want to restrict the class of signature morphisms which may be used (as second components) to construct open formulae. In fact, in the above remark sketching how free variables may be introduced into GEQ we used only algebraic signature inclusions $\iota:\Sigma\to\Sigma'$, where the only new symbols in Σ' were constants. To guarantee that the translation of open formulae is defined under such a restriction, we consider only restrictions to a collection $\mathbb{M}$ of signature morphisms which is closed (at least) under pushing out along arbitrary signature morphisms, i.e. for any signature morphism $\sigma:\Sigma\to\Sigma1$ if $\theta:\Sigma\to\Sigma'\in\mathbb{M}$ then there is a pushout in Sign

$$\begin{tikzcd}
\Sigma' \arrow[r, "\sigma'"] & \Sigma1' \\
\Sigma \arrow[u, "\theta"] \arrow[r, "\sigma"'] & \Sigma1 \arrow[u, "\theta'"']
\end{tikzcd}$$

such that $\theta'\in\mathbb{M}$.

Examples of such collections $\mathbb{M}$ in AlgSig include: the collection of all algebraic signature inclusions, the restriction of this to inclusions $\theta:\Sigma\to\Sigma'$ such that Σ' contains no new sorts, the further restriction of this by the requirement that Σ' contains new constants only (as above), the collection of all algebraic signature morphisms which are onto w.r.t. sorts, the collection of all identities and the collection of all morphisms. Note that most of the above permit variables denoting operations or even sorts.

In the rest of this section we briefly sketch how to universally close the open formulae introduced above (the construction is based on the notion of a syntactic operation in [Bar 74]). It is therefore peripheral to the main concern of this paper but we would like to add some logical meat to our treatment of free variables.

Let $\mathbb{M}$ be a collection of signature morphisms which is closed under pushing out along arbitrary morphisms in Sign. Let Σ be a signature and let $\langle\phi,\theta\rangle$ be an open Σ-formula such that $\theta\in\mathbb{M}$. Consider the universal closure of $\langle\phi,\theta\rangle$, written $\forall\langle\phi,\theta\rangle$, as a new Σ-sentence. The satisfaction relation and the translation of sentences $\forall\langle\phi,\theta\rangle$ along a signature morphism are defined in the expected way:

- A Σ-model satisfies $\forall\langle\phi,\theta\rangle$ if each of its θ-extensions satisfies ϕ, i.e. for any $M\in|\mathrm{Mod}(\Sigma)|$

  $M\models\forall\langle\phi,\theta\rangle$ iff for any $M'\in|\mathrm{Mod}(\Sigma')|$ such that $M'|_\theta=M$, $M'\models\phi$.

- For any signature morphism $\sigma:\Sigma\to\Sigma1$, $\sigma(\forall\langle\phi,\theta\rangle) =_{def} \forall\sigma(\langle\phi,\theta\rangle)$, where $\sigma(\langle\phi,\theta\rangle)=\langle\sigma'(\phi),\theta'\rangle$ is the translation of $\langle\phi,\theta\rangle$ as an open Σ-formula (with $\theta'\in\mathbb{M}$).

Note that in the above we have extended our underlying institution INS. Formally, we can define the extension of INS by universal closure w.r.t. $\mathbb{M}$, $\mathrm{INS}^\forall(\mathbb{M})$, to be the following institution:

- $\mathrm{Sign}_{\mathrm{INS}^\forall(\mathbb{M})}$ is $\mathrm{Sign}_{\mathrm{INS}}$.
- For any signature Σ, $\mathrm{Sen}_{\mathrm{INS}^\forall(\mathbb{M})}(\Sigma)$ is the disjoint union of $\mathrm{Sen}_{\mathrm{INS}}(\Sigma)$ with the collection of all universal closures $\forall\langle\phi,\theta\rangle$ of open Σ-formulae, where $\theta\in\mathbb{M}$; for a signature morphism $\sigma:\Sigma\to\Sigma1$ $\mathrm{Sen}_{\mathrm{INS}^\forall(\mathbb{M})}(\sigma)$ is the function induced by $\mathrm{Sen}_{\mathrm{INS}}(\sigma)$ on $\mathrm{Sen}_{\mathrm{INS}}(\Sigma)$ and by the notion of translation of universally closed open formulae as defined above.
- $\mathrm{Mod}_{\mathrm{INS}^\forall(\mathbb{M})}$ is $\mathrm{Mod}_{\mathrm{INS}}$.
- The satisfaction relation in $\mathrm{INS}^\forall(\mathbb{M})$ is induced by the satisfaction relation of INS for INS-sentences and the notion of satisfaction for universally closed open formulae as defined above.

The following theorem guarantees that $\mathrm{INS}^\forall(\mathbb{M})$ is in fact an institution.

Theorem For any signature morphism $\sigma: \Sigma \to \Sigma 1$, open Σ-formula $\langle \phi, \theta \rangle$ and $\Sigma 1$-model $M1 \in |\mathbf{Mod}(\Sigma 1)|$

$$M1|_{\sigma} \models \forall \langle \phi, \theta \rangle \quad \text{iff} \quad M1 \models \sigma(\forall \langle \phi, \theta \rangle)$$

Example Let $\mathbb{I}$ be the collection of morphisms $\iota: \Sigma \to \Sigma'$ in **AlgSig** such that ι is an algebraic signature inclusion and Σ' contains new constants only. The institution $GEQ^{\forall}(\mathbb{I})$ is the institution of universally quantified equations (cf. [GB 83]). If we additionally allow Σ' to contain new operation names (not just constants) then quantification along morphisms in $\mathbb{I}$ leads to a version of second-order logic.

Obviously, other quantifiers (there exists, there exist infinitely many, there exists a unique, for almost all ...) may be introduced to institutions in the same manner as we have just introduced universal quantifiers. It is also worth mentioning that one may similarly introduce logical connectives (cf. [Bar 74]). Note that by iterating this idea we can, for example, derive the institution of first-order logic from the institution of ground atomic formulae.

4 Specification-building operations

In this section we describe a set of simple operations for building specifications in an arbitrary institution. Our intention is to provide low-level operations which collectively give sufficient power and flexibility to constitute a kernel for building high-level specification languages in any institution. We intentionally do not define a formal specification language but only the specification-building operations behind such a language. The difference is mainly one of syntax; although we provide a suggestive notation for our operations, this is not a complete syntax yet because without fixing a particular institution the syntax of signatures and sentences cannot be fixed. This attitude admits certain informality in the presentation below. However, we do take care to formally define the semantics of all our operations.

Let INS be an arbitrary institution, fixed throughout this section.

A specification describes a collection of models of the same signature. To formalise this, for any specification SP we define its signature $Sig[SP] \in |\mathbf{Sign}|$ and the collection of its models $Mod[SP] \subseteq |\mathbf{Mod}(Sig[SP])|$. It is more usual to define the semantics of specification-building operations in terms of theories in the underlying (or an extended) institution rather than in terms of collections of models (as in e.g. Clear). But this is not an option here -- most of the operations defined below cannot be naturally viewed on this level. If $Sig[SP] = \Sigma$ then we call SP a Σ-specification.

The operations we provide are the following:

- Form a basic specification given a signature Σ and Σ-sentences Φ. This specifies the collection of Σ-models that satisfy Φ.
- Form the union of two Σ-specifications SP and SP', specifying the collection of Σ-models satisfying both SP and SP'.
- Translate a Σ-specification to another signature Σ' along a signature morphism $\sigma: \Sigma \to \Sigma'$. This together with union allows large specifications to be built from smaller and more or less independent specifications.
- Derive a Σ'-specification from a richer Σ-specification using a signature morphism $\sigma: \Sigma' \to \Sigma$. This allows details of a constructive specification to be hidden while essentially preserving its collection of models.
- Given a Σ-specification restrict models to only those which are minimal extensions of their σ-reducts for a given $\sigma: \Sigma' \to \Sigma$. This imposes on the models of a specification the additional constraint which excludes models which are "larger" than necessary.
- Abstract away from certain details of a specification, admitting any models which

are equivalent to a model of the specification w.r.t. some given set of properties (defined using sentences of the institution).

- Close the collection of models of a specification under isomorphism.

Here is a more formal description of the above operations (we discuss their instantiations in a typical institution at a more intuitive level in section 5):

A basic specification is a pair $\langle\Sigma,\Phi\rangle$, where $\Sigma\in|\mathbf{Sign}|$ is a signature and $\Phi\subseteq\mathrm{Sen}(\Sigma)$ is a set of Σ-sentences. We define:

$$\mathrm{Sig}[\langle\Sigma,\Phi\rangle] = \Sigma$$
$$\mathrm{Mod}[\langle\Sigma,\Phi\rangle] = \{\, M\in|\mathrm{Mod}(\Sigma)| \mid M\models\Phi \,\}$$

Given two Σ-specifications SP and SP′ (i.e. Sig[SP]=Sig[SP′]=Σ) their union SP $\cup$ SP′ is defined as follows

$$\mathrm{Sig}[SP \cup SP'] = \Sigma$$
$$\mathrm{Mod}[SP \cup SP'] = \mathrm{Mod}[SP] \cap \mathrm{Mod}[SP']$$

(where $\cap$ denotes set-theoretic intersection). Note that if SP and SP′ are basic specifications $\langle\Sigma,\Phi\rangle$ and $\langle\Sigma,\Phi'\rangle$ then their union has the same collection of models as $\langle\Sigma,\Phi\cup\Phi'\rangle$ (this time $\cup$ denotes the usual set-theoretic union).

If SP is a Σ-specification and $\sigma\colon\Sigma\rightarrow\Sigma'$ is a signature morphism then we define the translation of SP along σ, **translate** SP **by** σ, by:

$$\mathrm{Sig}[\textbf{translate } SP \textbf{ by } \sigma] = \Sigma'$$
$$\mathrm{Mod}[\textbf{translate } SP \textbf{ by } \sigma] = \{\, M'\in|\mathrm{Mod}(\Sigma')| \mid M'|_\sigma\in\mathrm{Mod}[SP] \,\}$$

If SP is a basic specification $\langle\Sigma,\Phi\rangle$ then **translate** SP **by** σ has the same collection of models as $\langle\Sigma',\sigma(\Phi)\rangle$, where $\sigma(\Phi)$ is the image of Φ under σ (i.e. Sen(σ) here).

Note again that using union we can only "put together" specifications with the same signature. To combine specifications with different signatures we have to form a "union signature" (renaming some of the signature symbols if necessary), translate the specifications into this "union signature" (using **translate** w.r.t. appropriate signature injections) and then form the union of the translated specifications. All this may be combined into one operation using an appropriate category of "signature inclusions" to form the "union signature" as a coproduct (R. Burstall, private communication, cf. also a remark in [GB 83] section 6.1). However, we decided to keep two simple, more elementary operations (which gives slightly more flexibility) rather than provide a single higher-level operation.

If $\sigma\colon\Sigma'\rightarrow\Sigma$ is a signature morphism then from any Σ-specification SP we can derive a Σ'-specification, **derive from** SP **by** σ:

$$\mathrm{Sig}[\textbf{derive from } SP \textbf{ by } \sigma] = \Sigma'$$
$$\mathrm{Mod}[\textbf{derive from } SP \textbf{ by } \sigma] = \{\, M|_\sigma \mid M\in\mathrm{Mod}[SP] \,\}$$

For $\Phi\subseteq\mathrm{Sen}(\Sigma)$, Mod[**derive from** $\langle\Sigma,\Phi\rangle$ **by** σ] $\subseteq$ Mod[$\langle\Sigma',\sigma^{-1}(\Phi)\rangle$], where $\sigma^{-1}(\Phi)$ is the coimage of Φ under σ (i.e. Sen(σ)). Note however that this inclusion may be proper, since sometimes not all the properties of models of the derived specification are expressible using just Σ'-sentences. The right-hand side of this inclusion corresponds to the definition of the **derive** operation in Clear [BG 80].

To define restriction to the minimal models of a specification we need the following notion:

Let $\sigma\colon\Sigma'\rightarrow\Sigma$ be a signature morphism and $C\subseteq|\mathrm{Mod}(\Sigma)|$ be a collection of Σ-models. We say that a model M is *σ-minimal in* C if $M\in C$ and if M contains (to within isomorphism) no proper submodel from C with an isomorphic σ-reduct, which we formalise as follows: for every $M1 \in C$, any monomorphism $m: M1 \rightarrow M$ (in $\mathbf{Mod}(\Sigma)$) such that $m|_\sigma$ is an isomorphism from $M1|_\sigma$ to $M|_\sigma$ (in $\mathbf{Mod}(\Sigma')$) is in fact an isomorphism (in $\mathbf{Mod}(\Sigma)$).

Now, for any signature morphism $\sigma: \Sigma' \rightarrow \Sigma$ and Σ-specification SP, **minimal** SP **wrt** σ specifies the models of SP which are minimal extensions of their σ-reducts, i.e.:

Sig[**minimal** SP **wrt** σ] = Σ
Mod[**minimal** SP **wrt** σ] = { M | M is σ-minimal in Mod[SP] }

To describe the next specification-building operation we need some further definitions:

For any signature Σ, set of Σ-sentences $\Phi \subseteq \mathrm{Sen}(\Sigma)$ and Σ-models $M1, M2 \in |\mathbf{Mod}(\Sigma)|$, we say that M1 is *$\Phi$-equivalent* to M2 if for any $\phi \in \Phi$, $M1 \models \phi$ iff $M2 \models \phi$.

Then, for any signature morphisms $\theta: \Sigma \rightarrow \Sigma'$, $\sigma: \Sigma'' \rightarrow \Sigma'$ and models $M \in |\mathbf{Mod}(\Sigma)|$, $M' \in |\mathbf{Mod}(\Sigma')|$, we say that M' is a *$\sigma$-full θ-extension* of M if it is a θ-extension of M, i.e. $M'|_\theta = M$, and its σ-reduct is reachable, i.e. $M'|_\sigma$ is $\iota_{\Sigma''}$-minimal in $|\mathbf{Mod}(\Sigma'')|$, where for any signature $\Sigma1$ we use the notation $\iota_{\Sigma1}$ to denote the unique morphism from the initial object in Sign to $\Sigma1$ (the "inclusion" of the "empty signature" into $\Sigma1$).

For any signature morphisms $\theta: \Sigma \rightarrow \Sigma'$ and $\sigma: \Sigma'' \rightarrow \Sigma'$, set $\Phi' \subseteq \mathrm{Sen}(\Sigma')$ of open Σ-formulae with variables "$\Sigma' - \theta(\Sigma)$" and Σ-models $M1, M2 \in |\mathbf{Mod}(\Sigma)|$, we say that M1 is *$\Phi'$-equivalent* to M2 *via θ on σ* if there are σ-full θ-extensions $M1', M2' \in |\mathbf{Mod}(\Sigma')|$ of M1 and M2, respectively, such that M1' is Φ'-equivalent to M2'. (For an intuitive description of the meaning of this definition in a typical situation see section 5.)

Now, for any Σ-specification SP, signature morphisms $\theta: \Sigma \rightarrow \Sigma'$ and $\sigma: \Sigma'' \rightarrow \Sigma'$ and set $\Phi' \subseteq \mathrm{Sen}(\Sigma')$ of open Σ-formulae with variables "$\Sigma' - \theta(\Sigma)$", the specification **abstract** SP **wrt** Φ' via θ on σ (intuitively) ignores the properties specified in SP as much as possible without affecting Φ' where σ determines which elements of models must be considered when interpreting Φ', i.e. it admits any model Φ'-equivalent via θ on σ to a model of SP:

Sig[**abstract** SP **wrt** Φ' **via** θ **on** σ] = Σ
Mod[**abstract** SP **wrt** Φ' **via** θ **on** σ] =
{ $M1 \in |\mathbf{Mod}(\Sigma)|$ | M1 is Φ'-equivalent to M2 via θ on σ for some $M2 \in$ Mod[SP] }

Note that a model of SP need not, in general, be a model of **abstract** SP **wrt** Φ' **via** θ **on** σ. In fact, it is if and only if it has a σ-full θ-extension.

Finally, for any Σ-specification SP, the specification **iso close** SP is defined by:

Sig[**iso close** SP] = Σ
Mod[**iso close** SP] = { $M \in |\mathbf{Mod}(\Sigma)|$ | M is isomorphic to some model $M1 \in$ Mod[SP] }

Observe that there is no guarantee in the definition of an institution that the satisfaction relation is preserved under isomorphism of models. Thus, even the collection of models of a basic specification need not be closed under isomorphism. Also note (see section 5) that the collection of models of **derive from** SP **by** σ need not be closed under isomorphism even if the collection of models of SP is. However, the remaining operations do preserve closure under isomorphism.

5 A standard case

The definitions of the specification-building operations we gave in the last section were so general that they may be difficult to understand. We will now consider what the operations do in the familiar context -- the institution FOEQ of first-order logic with equality as the only predicate symbol -- and compare them with operations in existing specification languages.

We define this institution as follows:

- **Sign**$_{FOEQ}$ is AlgSig (i.e. Sign$_{GEQ}$, the category of algebraic signatures and their morphisms).
- **Mod**$_{FOEQ}$ is Mod$_{GEQ}$ (i.e. for any algebraic signature Σ, Mod$_{FOEQ}(\Sigma)$ is the category of Σ-algebras and for any algebraic signature morphism $\sigma: \Sigma \rightarrow \Sigma'$, **Mod**$_{FOEQ}(\sigma)$ is the σ-reduct functor from Mod$_{FOEQ}(\Sigma')$ to Mod$_{FOEQ}(\Sigma)$).
- For any algebraic signature Σ, Sen$_{FOEQ}(\Sigma)$ is the set of closed first-order formulae with operation symbols from Σ and the equality as the only predicate symbol; for any algebraic signature morphism $\sigma: \Sigma \rightarrow \Sigma'$, Sen$_{FOEQ}(\sigma)$ is the translation of Σ-formulae to Σ'-formulae defined in the natural way.
- The satisfaction relation is determined by the standard notion of satisfaction of first-order sentences.

This clearly forms an institution (details in [GB 83]). Moreover, our assumptions that the category of signatures is finitely cocomplete and that Mod$_{FOEQ}$ translates finite colimits in **Sign**$_{FOEQ}$ to limits in Cat obviously hold here too: in fact, these parts of the institution are exactly the same as in GEQ.

In the following we analyse the specification-building operations defined in section 4 in the framework of the above institution of first-order logic.

There is hardly anything to be said about basic specifications. All specification languages provide a syntactic tool for listing a set of axioms over a given signature, although usually they differ in which formulae are acceptable. First-order equational axioms are relatively powerful compared with e.g. equations in [ADJ 76] or universal Horn axioms in [ADJ 80].

In examples we use a syntax corresponding to that of Clear:

```
Bool = sorts  bool
       opns   true, false: →bool
              not: bool→bool
              or: bool, bool→bool
       axioms ∀x. true or x = true       not(true) = false
              ∀x. false or x = x         not(false) = true
              ∀x. x=true ∨ x=false
```

(Of course, *or* and ∨ are formally not the same here.)

The union operation differs from the corresponding operation in other specification languages (e.g. + in Clear or ASL) in that it works only for specifications of the same signature, and so it provides no direct way for putting together specifications over different signatures. To do this, we have to use union together with the translate operation, which introduces new sorts and operation symbols to a specification (and renames old ones).

The sum of two specifications (as defined in ASL) may now be expressed as follows:

SP + SP' $=_{def}$ (**translate** SP **by** ι) $\cup$ (**translate** SP' **by** ι')

where $\iota: \Sigma \rightarrow \Sigma \cup \Sigma'$ and $\iota': \Sigma' \rightarrow \Sigma \cup \Sigma'$ are the inclusions of Σ and Σ', respectively, into their set-theoretic union $\Sigma \cup \Sigma'$. To avoid unintended confusion of symbols with the same names in Σ and Σ', instead of using the inclusions ι and ι' we need injections which rename the common symbols as required (as in Clear).

An operation similar to enrich in Clear (identical when there are no symbol clashes) may be defined in terms of the union and the basic specification operations:

enrich SP by sorts S opns Ω axioms $\Phi =_{def}$ SP + $\langle\langle$sorts(SP) $\cup$ S, opns(SP) $\cup \Omega\rangle, \Phi\rangle$

Note that the **translate** operation corresponds directly to the TRA operator of [EWT 83].

The **derive** operation is, in a sense, dual to **translate**. It may be used to rename and to hide some of the sorts and operation symbols of a specification. It is exactly the same as **derive** in ASL [SW 83, short version only] and corresponds directly to the reflection (REF) operator in [EWT 83]. The intention is the same as that of **derive** in Clear, but the meaning is slightly different as mentioned in section 4.

Note that the collection of models of **derive from** SP **by** σ need not to be closed under isomorphism even if Mod[SP] is. This phenomenon occurs when σ is not injective on sorts. When for two sorts s and s' $\sigma(s)=\sigma(s')$, **derive from** SP **by** σ requires the carriers of sorts s and s' to be identical rather than only isomorphic. (See below for some further discussion on this point.)

The **minimal** operation restricts the models of a specification SP to only those algebras which contain (to within isomorphism) no proper subalgebra which is a model of SP with the same σ-reduct. In particular, in the institution of first-order logic the definition of **minimal** as given in section 4 states that if an algebra A is a model of the specification **minimal** SP **wrt** σ then A is a model of SP and whenever B is a model of SP which is a subalgebra of A such that $B|_\sigma=A|_\sigma$, then A=B. Moreover, if Mod[SP] is closed under isomorphism then the converse of this implication is true as well. In general, however, this need not be the case.

The **minimal** operation is similar to the GEN operator of [EWT 83] rather than to the **reachable** operation of ASL [SW 83] or the use of finitely generated algebras in CIP-L [Bau 81]. In fact, minimality does not guarantee reachability (and hence, for example, the induction principle need not hold in minimal algebras) although reachability does imply minimality:

```
NN = sorts   nat
     opns    zero: →nat
             succ: nat→nat
     axioms ∃x. succ(x) = x
```

Nat_ω = **minimal** NN **wrt** $\iota_{\mathrm{Sig[NN]}}$

(Recall that $\iota_{\mathrm{Sig[NN]}}$ is the inclusion of the empty signature into Sig[NN].) Models of NN contain (up to isomorphism) either a finite segment of natural numbers $\mathbb{N}$, $\{0,\ldots,n\}$ with succ(n)=n and an arbitrary unreachable part or else $\mathbb{N}$ together with an arbitrary unreachable part containing at least one element x such that succ(x)=x. The only models of Nat_ω are (up to isomorphism) finite segments of $\mathbb{N}$, $\{0,\ldots,n\}$ with succ(n)=n and all elements reachable, or else $\mathbb{N}$ together with exactly one unreachable element ω such that $\mathrm{succ}(\omega)=\omega$.

An operation which is like **reachable** in ASL [SW 83] may be defined in terms of **minimal** as follows:

reachable SP **wrt** $\sigma =_{\mathrm{def}}$ SP + **minimal** <Sig[SP], $\emptyset$> **wrt** σ

The **reachable** operation of ASL is in fact a special case of the above:

reachable SP **on** S $=_{\mathrm{def}}$ **reachable** SP **wrt** ι

where ι is the inclusion of the signature <sorts(SP)-S, $\emptyset$> into Sig[SP].

Nat-seg = **reachable** NN **wrt** $\iota_{\mathrm{Sig[NN]}}$ = **reachable** NN **on** {nat}

Now, the only models of Nat-seg are (up to isomorphism) finite segments of $\mathbb{N}$, $\{0,\ldots,n\}$ with succ(n)=n and all elements reachable.

Our specification-building operations do not provide the possibility to require initiality or freeness (unless axioms like data constraints [GB 83] are already present in the underlying institution). We could easily add such an operation. In practice, however, this requires a serious restriction on the underlying institution which in the standard case excludes axioms more powerful than universal Horn formulae (see [MM 83], also [Tar 83]) although note that formally it is possible to give a semantics for data constraints without this restriction [Tar 84]. Anyway, we do not consider such an operation necessary; see [SW 83] for further discussion on this point.

The derive operation allows one to hide some of the sorts and operation symbols of a specification. This also causes some of the properties of its models to be hidden, since they cannot be expressed using the remaining operations. However, this is not real abstraction yet since the structure induced by the hidden operations remains. To do real abstraction we can pick out a set of properties we would like to preserve and then use the **abstract** operation.

The properties we would like to preserve must be expressed as sentences of the underlying institution. However, to deal properly with unreachable elements of models (dubbed "junk" in [BG 81]) we have to use open formulae rather than (closed) sentences. Why not just forbid junk? Although unreachable elements seem to be of no consequence, there is an example (Infinite-Set) in [SW 83] which shows how an unreachable element in a model of SP can become reachable and useful in **enrich** SP **by opns** .... Furthermore, junk naturally arises when we "forget" operations using **derive**, which corresponds to the situation where an algebra which is reachable when viewed from a low level becomes non-reachable when viewed from a higher level of abstraction.

The most natural way one may view **abstract** in the institution of first-order logic is, we think, the following (this gives a direct generalisation of **abstract** in ASL -- see below):

Given a Σ-specification SP, extend Σ by as many variables X as necessary to name all the elements of algebras you would like to deal with. Then give the set Φ of properties which are to be preserved under abstraction. These properties must be expressed as $\Sigma(X)$-sentences. The abstraction of SP with respect to Φ is given by the specification **abstract** SP **wrt** Φ **via** ι **on** ι' where $\iota: \Sigma \to \Sigma(X)$ is the algebraic signature inclusion and $\iota': \mathcal{X} \to \Sigma(X)$ is the inclusion into $\Sigma(X)$ of the algebraic signature $\mathcal{X}$ with sorts $\{s \in sorts(\Sigma) \mid X_s \text{ is non-empty}\}$ and constants X as the only operations. This specifies (roughly) the collection of Σ-algebras which satisfy the same formulae of Φ as models of SP. More formally, a Σ-algebra A satisfies **abstract** SP **wrt** Φ **via** ι **on** ι' if and only if there is a Σ-algebra B which satisfies SP and variable valuations $v_A: X \to |A|$ and $v_B: X \to |B|$ which are surjective on sorts in which X is non-empty such that for any formula $\phi \in \Phi$, ϕ holds in A under the valuation v_A if and only if ϕ holds in B under the valuation v_B.

Consider the following example:

```
Nat = minimal  < Σ, { ∀x. 0≠succ(x) , ∀x,y. (succ(x)=succ(y) ⟹ x=y) } >  wrt ι_Σ
where Σ = sorts nat      opns 0: →nat    succ: nat→nat
Nat-even = enrich Bool+Nat by opns    even: nat→bool
                              axioms  even(0) = true        even(succ(0)) = false
                                      ∀x: nat. even(succ(succ(x)) = even(x)
```

All models of Nat are isomorphic to the standard model of the natural numbers. (Note that for this specification minimality guarantees reachability.) Each model of Nat-even is the combination of a model of Nat with a model of Bool (see above) with an extra operation *even*. We can abstract from Nat-even preserving only the properties of booleans and the behaviour of *even* as follows:

Nat-mod = **abstract** Nat-even **wrt** Φ **via** ι **on** ι'

where: X is a set of variables with $X_{nat}=\emptyset$ and at least two elements of sort bool, $\iota : \Sigma \rightarrow \Sigma(X)$ and $\iota' : \mathcal{X} \rightarrow \Sigma(X)$ are algebraic signature inclusions, and $\Phi = \{ t=t' \mid t, t'$ are Σ-terms of sort bool with variables X $\}$, where Σ = Sig[Nat-even] and $\mathcal{X}$ is derived from X as above.

All models of Nat-mod are isomorphic either to the natural numbers modulo n, for some $n \in \{2, 4, 6, \ldots\}$ or to $\mathbb{N}$ itself with arbitrary junk of sort *nat* in both cases.

Observe that the above condition means that there are "corresponding parts" of A and B in which exactly the same formulae of Φ hold. This is not the same as the requirement that exactly the same formulae of Φ hold in all of A and B. Namely, if two algebras are Φ-equivalent via ι on ι' then (assuming that Φ is closed under renaming of variables) they are equivalent w.r.t. the set of formulae which results from universally closing all $\phi \in \Phi$, but not vice versa; here is an example:

Suppose Σ = **sorts** s **opns** $f: s \rightarrow s$ and A, B are Σ-algebras such that $|A|_s = |B|_s = \{0, 1, 2\}$, $f_A(0) = f_B(0) = 1$, $f_A(1) = f_B(1) = 0$, $f_A(2) = 2$ but $f_B(2) = 1$.

$$\begin{tikzcd} A: & 0 \arrow[r, shift left] & 1 \arrow[l, shift left] & 2 \arrow[loop above] & B: & 0 \arrow[r, shift left] & 1 \arrow[l, shift left] & 2 \arrow[l] \end{tikzcd}$$

Then A and B are equivalent w.r.t. the formula $\forall x. f(x)=x$ because neither A nor B satisfies it, but they are not equivalent w.r.t. the set of formulae $\{f(x1)=x1, f(x2)=x2, \ldots\}$ because for any surjective variable valuation A satisfies at least one of the formulae in this set while B satisfies none of them.

The idea of comparing algebras w.r.t. a set of formulae also appeared in [Pep 83]. The difference is that there only closed formulae were considered. The two approaches are equivalent if one allows his closed formulae to be infinitary. In fact, two Σ-algebras are equivalent in our sense w.r.t. a set Φ of $\Sigma(X)$-sentences (with ι and ι' as above) if and only if they are equivalent w.r.t. the following closed Σ-sentence:

$$\exists X. (\bigwedge \{ \forall y{:}s. \bigvee \{y=x \mid x \in X_s\} \mid s \in sorts(\Sigma) \ \& \ X_s \neq \emptyset \} \ \& \ \bigwedge \Phi))$$

where $\bigvee$ and $\bigwedge$ denote infinitary disjunction and conjunction, respectively. Note that the size of X depends on the cardinality of the algebras we would like to deal with, so even in the standard case of countable algebras $L_{\omega_1 \omega}$ logic may not be sufficient.

We can further specialise our **abstract** operation to get the **abstract** operation of ASL. Namely, whenever $W \subseteq |T_\Sigma(X)|$ is a set of terms ($T_\Sigma(X)$ is the Σ-algebra of Σ-terms with variables X, see e.g. [ADJ 76]) then the ASL specification **abstract** SP **wrt** W is equivalent to our **abstract** SP **wrt** EQ(W) **via** ι **on** ι', where EQ(W) is the set of all equations t=t' such that t and t' are terms of the same sort which belong to W and ι and ι' are as above.

The **abstract** operation may be used to relax the interpretation of a specification to all models which are behaviourally equivalent to a model of the specification (this is called *behavioural abstraction* in ASL [SW 83] -- see this paper for examples).

Suppose that Σ is an algebraic signature and IN and OUT are subsets of the sorts of Σ. Now, consider all computations which take input from sorts IN and give output in sorts OUT; this set of computations corresponds to the set $|T_\Sigma(X_{IN})|_{OUT}$ of Σ-terms of sorts OUT with variables of sorts IN. Two algebras are equivalent in our sense with respect to the set of equations $EQ(|T_\Sigma(X_{IN})|_{OUT})$ if they are behaviourally equivalent, that is they have matching input/output relations. Note that this covers the notions of behavioural equivalence with respect to a single set OBS of *observable* sorts which appear in the literature. For example, in [Rei 81] and [GM 82] we have IN=sorts(Σ), OUT=OBS; in [Sch 82], [SW 83] and [GM 83] IN=OUT=OBS; and in [GGM 76], [BM 81] and [Kam 83] IN=$\emptyset$ and OUT=OBS.

The **abstract** operation usually does not appear explicitly in specification languages (the only exception we know about is ASL) ; instead, it is somehow included in the notion of implementation of one specification by another. The inclusion of **abstract** as an explicit specification-building operation allows us to use a very simple and elegant definition of implementation (see [SW 83] for details). On the other hand, **abstract** makes inference more complex because it is not monotone (at the level of theories) in the sense that things true in SP need not be true in **abstract** SP **wrt** . . . .

The **iso close** operation closes the collection of models of a specification under isomorphism. The only situation which the collection of models of a specification may not be closed under isomorphism already is when the specification contains a use of **derive from** . . . by σ where σ is not injective on sorts. It would be easy to "fix" **derive** by changing the definition so that the result is automatically closed under isomorphism (this was the alternative adopted in ASL [SW 83, long version]). Another possible "solution", which turns out to yield exactly the same expressive power, is to restrict **derive** by allowing only signature morphisms which are injective on sorts. We prefer, however, to adopt neither solution, retaining both **derive** (as it is defined now) and **iso close**. This is consistent with our policy of providing operations which are as elementary as possible. It also leaves open the possibility of specifying collections of models which are not closed under isomorphism; despite the well-known arguments that closure under isomorphism is natural, we feel that there is no harm in providing such flexibility.

6 A partial case

A good test for the general definitions in section 4 is to consider their instantiation in several significantly different institutions. In this section we discuss the result of instantiating in an institution of partial first-order logic. This is an interesting case to examine because the category of partial Σ-algebras as defined below is sufficiently different from the category of total Σ-algebras discussed in sections 2 and 5 that the definitions of operations (like **minimal** and **abstract**) which depend on the structure of this category are put to a non-trivial test.

Let $\Sigma=\langle S,\Omega\rangle$ be an algebraic signature. A *partial Σ-algebra* is just like a (total) Σ-algebra except that some of its operations may be partial. Formally, a partial Σ-algebra consists of an S-indexed family of sets $|A| = \langle |A|_s\rangle_{s\in S}$ and for each $f: s1,\ldots,sn\rightarrow s$ a possibly partial function $f_A: |A|_{s1}\times\ldots\times|A|_{sn}\rightarrow|A|_s$. A *(weak) Σ-homomorphism* from a partial Σ-algebra A to a partial Σ-algebra B, $h: A\rightarrow B$, is a family of (total) functions $\langle h_s\rangle_{s\in S}$ where $h_s: |A|_s\rightarrow|B|_s$ such that for any $f: s1,\ldots,sn\rightarrow s$ and $a_1\in|A|_{s1},\ldots,a_n\in|A|_{sn}$

$$\begin{aligned} f_A(a_1,\ldots,a_n) \text{ defined } \Rightarrow\ & f_B(h_{s1}(a_1),\ldots,h_{sn}(a_n)) \text{ defined and} \\ & h_s(f_A(a_1,\ldots,a_n)) = f_B(h_{s1}(a_1),\ldots,h_{sn}(a_n)) \end{aligned}$$

([BrW 82] would call this a *total* Σ-homomorphism). If moreover h satisfies the condition

$$f_B(h_{s1}(a_1),\ldots,h_{sn}(a_n)) \text{ defined } \Rightarrow f_A(a_1,\ldots,a_n) \text{ defined}$$

then h is called a *strong Σ-homomorphism*.

The category of partial Σ-algebras $\mathrm{PAlg}(\Sigma)$ has partial Σ-algebras as objects and strong Σ-homomorphisms as morphisms; the composition of homomorphisms is the composition of their corresponding components as functions. (This obviously forms a category.)

The definition of the *σ-reduct* functor $\_|_\sigma: \mathrm{PAlg}(\Sigma')\rightarrow\mathrm{PAlg}(\Sigma)$ where $\sigma: \Sigma\rightarrow\Sigma'$ is an algebraic signature morphism is exactly the same as in the total case; see section 2.

A *partial first-order Σ-sentence* is a closed first-order formula built from Σ-terms using the logical connectives $\neg$, $\wedge$, $\vee$ and $\Rightarrow$, the quantifiers $\forall$ and $\exists$, and the atomic formulae

$D_s(t)$ and $t=t'$ (strong equality [BrW 82]) for each sort s in Σ and terms $t, t' \in |T_\Sigma(X)|_s$ (i.e. t, t' are Σ-terms of sort s with variables X).

Suppose A is a partial Σ-algebra. Then A *satisfies* an atomic formula $D_s(t)$ under a valuation $v: X \to |A|$, written $A \models_v D_s(t)$, iff the value of t in A under v is defined (we omit the definition of the value of a term in a partial algebra under a valuation; see [Bur 82] or [Rei 84] for details). A partial Σ-algebra A satisfies an atomic formula $t=t'$ (where $t, t' \in |T_\Sigma(X)|_s$ for some sort s in Σ) under a (total) valuation $v: X \to |A|$, written $A \models_v t=t'$, iff

$A \not\models_v D_s(t)$ and $A \not\models_v D_s(t')$, or
$A \models_v D_s(t)$ and $A \models_v D_s(t')$ and the values of t and t' in A under v are the same.

Satisfaction of (closed) partial first-order Σ-sentences is defined as usual, but note that $\forall$ and $\exists$ quantify only over defined values.

The above definitions amount in the obvious way to an institution PFOEQ of partial first-order logic. The satisfaction condition follows from the fact that FOEQ is an institution and that definedness of terms is preserved under change of signature. Moreover, $\mathbf{Sign}_{PFOEQ}$ is finitely cocomplete (as mentioned in sections 2 and 5) and $\mathbf{Mod}_{PFOEQ}$ translates finite colimits in $\mathbf{Sign}_{PFOEQ}$ to limits in **Cat**.

The result of instantiating the general definitions of section 4 in PFOEQ gives a set of operations which in some respects resemble those in the early version of ASL described in [Wir 82] defined in the context of partial algebras (call this language "partial ASL", but note that it is significantly different from the ASL described in [SW 83]). One difference, however, is that in partial ASL the collection of models of any specification was closed under renaming of sorts and operations, i.e. if $Sig[SP]=\Sigma$ and $\Sigma \cong \Sigma'$, then Mod[SP] contains partial Σ'-algebras as well as partial Σ-algebras. This feature could be obtained by changing the definition of $\mathbf{Mod}_{PFOEQ}$ and $\models_{\Sigma, PFOEQ}$ but we prefer to omit it.

The comments regarding basic specifications and the operations $\cup$, **translate**, **derive** and **iso close** (and how to define + in terms of $\cup$ and **translate**) in section 5 apply without change here. More interesting are the operations **minimal** and **abstract**.

Intuitively speaking, the minimal operation gives rather unexpected results. One would expect that **minimal** SP **wrt** σ should give the least-defined and smallest (wrt σ) models of SP, but instead it gives the smallest (wrt σ) models of SP in each class of equally-defined models. There seems to be no way to restrict to the minimally-defined partial algebras in a collection of models using the operations we have since strong homomorphisms cannot relate algebras unless they are equally defined. This means that there is no way to express the **mdef** operation of partial ASL, which restricts a collection of partial algebras to the ones which are minimally defined and reachable (and which satisfy true$\neq$false). We can define an operation which restricts to reachable models

reachable SP **wrt** σ $=_{def}$ SP $\cup$ **minimal** $\langle Sig[SP], \emptyset \rangle$ **wrt** σ

which at least gives the possibility of performing proofs by structural induction.

Abstract works in a way similar to that described in section 5. The use of **abstract** for behavioural abstraction is slightly different, since the properties to be preserved must include definedness of the results of "observable" computations. If Σ is an algebraic signature and IN, OUT are subsets of the sorts of Σ as in section 5, behavioural equivalence in the context of partial algebras becomes equivalence in our sense with respect to the set of formulae $EQ(|T_\Sigma(X_{IN})|_{OUT}) \cup \{ D_s(t) \mid t \in |T_\Sigma(X_{IN})|_s \text{ for } s \in OUT \}$. Partial ASL includes no operation similar to **abstract**.

We could get **minimal** to work as expected by changing the institution PFOEQ. The change needed is to use weak Σ-homomorphisms as the morphisms of **PAlg**(Σ) in place of strong Σ-homomorphisms. Then **mdef** as in partial ASL can be expressed, albeit in a rather unsatisfactory way:

> **mdef** SP $=_{def}$ (SP $\cup$ **minimal** <Sig[SP], D> **wrt** $\iota_{sig[SP]}$) + Bool
>
> where D = { $D_s(t)$ | t is a ground Sig[SP]-term of sort s and $M \models D_s(t)$ for all $M \in$ Mod[SP] } and Bool is a specification of the booleans including the axiom true$\neq$false.

But now **abstract** does not work as expected (note that its definition uses the notion of Σ-reachability $=_{def}$ minimality in |Mod(Σ)|, which changes when the morphisms of **PAlg**(Σ) are changed -- now only totally undefined partial Σ-algebras are Σ-reachable).

There is yet another possibility which makes both **minimal** and **abstract** work as expected in the partial case. Namely, we can view (some of) the definedness axioms of a specification as a part of its signature. (Although this might seem like a strange mixture of syntax with semantics, similar mixtures have appeared elsewhere -- [BR 83] includes equations in signatures which define the domains of operations, and [GDLE 82] includes information in signatures regarding which operations may produce error values.) More formally, we can use an institution $\mathbb{P}$FOEQ of partial first-order logic where the category of signatures is the category of theories in PFOEQ containing only definedness axioms. Thus, a signature in $\mathbb{P}$FOEQ is a pair $\langle\Sigma, D\rangle$ where Σ is an algebraic signature and D is a set of definedness formulae over Σ; a signature morphism $\sigma: \langle\Sigma, D\rangle \rightarrow \langle\Sigma', D'\rangle$ in $\mathbb{P}$FOEQ is an algebraic signature morphism $\sigma: \Sigma \rightarrow \Sigma'$ which preserves the definedness axioms, i.e. if $d \in D$ then $\sigma(d) \in D'$; for any signature $\langle\Sigma, D\rangle$, $\mathbf{Mod}_{\mathbb{P}FOEQ}(\langle\Sigma, D\rangle)$ is the category of all partial Σ-algebras which satisfy (at least) D with weak homomorphisms as morphisms. $\mathbf{Sen}_{\mathbb{P}FOEQ}$ and satisfaction are as in PFOEQ.

Now the **minimal** operation works in the intended way, as in the institution PFOEQ with weak homomorphisms. Observe that for any signature $\langle\Sigma, D\rangle$, a $\langle\Sigma, D\rangle$-reachable model (i.e. a $\iota_{\langle\Sigma, D\rangle}$-minimal model in $\mathbf{Mod}_{\mathbb{P}FOEQ}(\langle\Sigma, D\rangle)$) is a minimally-defined model with no unreachable elements, as intended. This means that **abstract** as defined in section 4 works nicely too. But note that to abstract with respect to a set of formulae with variables X we now have to abstract via a signature inclusion which extends the signature $\langle\Sigma, D\rangle$ to $\langle\Sigma(X), D \cup D(X)\rangle$, where $D(X) =_{def} \{ D_s(x) \mid s \in sorts(\Sigma),\ x \in X_s \}$ states that all "constants" X have defined values.

Note that in the above we started from the institution of partial first-order logic, identified in it a "subinstitution" of "static" sentences and then used theories of this subinstitution as the signatures of another institution. This seems to be an interesting way of building a new institution from an old one which perhaps deserves a more careful investigation.

7 Concluding remarks

In this paper we attempted to define a set of primitive and general specification-building operations which when instantiated in any institution provide a powerful but low-level tool for specification. We tested the institution-based general definitions of these operations by examining the result of instantiating them in two different ways: in an institution of total first-order equational logic (section 5) and in an institution of partial first-order logic (section 6). In formulating the definitions we also considered the result of instantiating them in two other institutions -- an error institution based on [Gog 77] and an institution of continuous algebras based on [ADJ 77] -- but due to space limitations we are unable to present the details of these investigations here.

The question of whether the definitions we have given are really general naturally arises;

maybe there is some institution which we have not considered in which the operations we have defined work in an unexpected way. Indeed, whenever one generalises on the basis of a small collection of examples one must choose between all the generalisations which are different in general but which coincide in the particular examples one has at hand. For example, in the definition of the **minimal** operation, to represent the concrete notion of injective homomorphisms we used just monomorphisms rather than say equalisers or extremal monomorphisms (or more generally we could parameterise our definition by an image factorisation system as in [Tar 84]). All of these possibilities work equally well in each of our example institutions. We can try to test our generalisations by comparing them with other available general definitions. So for example we can show that -- under certain not very restrictive conditions -- minimal corresponds to "generated" as defined in [GB 83] (note however that the definition of [GB 83] works only in liberal institutions, and this is a strong restriction).

Another natural question concerns our decision to allow the specification of collections of models which are not closed under isomorphism and our careful treatment of models containing unreachable elements. We chose this course because we cannot see any really compelling reason, either pragmatic or technical, for assuming that all useful collections of models are closed under isomorphism or that only reachable models are worth considering. On the other hand, we also know of no compelling reason why these assumptions (especially the former) are unreasonable. By leaving the choice to the specifier (or to the designer of a high-level specification language which builds upon our kernel operations) we provide the freedom to explore all possibilities without unnecessary restrictions.

Although the reader might have the impression that we have been carried away in our pursuit of generality, we tried to resist the urge to throw in unnecessary generalisations. So for example, it is clear that **iso close** can be generalised to give an operation which can close under different classes of morphisms, and not just under isomorphism. This generalisation might even be useful; note that closure under (sources of) monomorphisms gives closure under subalgebras, and closure under (targets of) epimorphisms gives closure under quotients. We do not claim to offer every possible operation on collections of models, only a few interesting ones which we know are useful. This is also part of our justification for omitting an operation which restricts to the initial or final elements in a collection of models.

We are not completely satisfied with our definition of **abstract**. It seems just too complicated, although its complexity stems directly from the difficulty of correctly generalising an important detail in the definition of **abstract** in ASL. We are actively investigating alternative definitions; the most promising one at the moment seems to be the following:

$$\mathrm{Sig}[\textbf{abstract}\ SP\ \textbf{wrt}\ \Phi'\ \textbf{via}\ \theta] = \Sigma$$
$$\mathrm{Mod}[\textbf{abstract}\ SP\ \textbf{wrt}\ \Phi'\ \textbf{via}\ \theta] = \{\ M1 \in |\mathrm{Mod}(\Sigma)|\ \mid\ M1 \text{ is } \Phi'\text{-equivalent to } M2 \text{ via } \theta \text{ for some } M2 \in \mathrm{Mod}[SP]\ \}$$

where M1 is *Φ'-equivalent* to M2 *via* θ if for any θ-extension of M1 there is a Φ'-equivalent θ-extension of M2 and vice versa. Unfortunately, we just do not know yet exactly how this definition relates to the definition of **abstract** in ASL and whether it is powerful enough to express behavioural abstraction.

One problem which we have not touched on is the provision of a institution-based parameterisation mechanism. We do not anticipate that this would be very difficult; the macro-like parameterisation mechanism in ASL should generalise without problems. Parameterisation mechanisms based on the idea that specifications are just theories which use the pushout in the category of theories to define application (such as those in Clear, LOOK and other languages) would be more difficult to generalise, although probably this could be done.

It would be interesting to try to build a high-level specification language on top of the kernel which we define here. Such a high-level language most likely would not lend itself to the generality of an arbitrary institution, since there are probably useful operations on specifications which could only be defined in a particular institution or in a restricted class of institutions.

Acknowledgements

Our thanks to Rod Burstall for his support and encouragement, to Martin Wirsing for his work on ASL, and to Gordon Plotkin for help with logic. This research was supported by the Science and Engineering Research Council.

8 References

[ADJ 76] Goguen, J.A., Thatcher, J.W. and Wagner, E.G. An initial algebra approach to the specification, correctness, and implementation of abstract data types. IBM research report RC 6487. Also in: Current Trends in Programming Methodology, Vol. 4: Data Structuring (R.T. Yeh, ed.), Prentice-Hall, pp. 80-149 (1978).

[ADJ 77] Goguen, J.A., Thatcher, J.W., Wagner, E.G. and Wright, J.B. Initial algebra semantics and continuous algebras. JACM 24, 1 pp. 68-95.

[ADJ 80] Ehrig, H., Kreowski, H.-J., Thatcher, J.W., Wagner, E.G. and Wright, J.B. Parameterized data types in algebraic specification languages (short version). Proc. 7th ICALP, Noordwijkerhout, Netherlands. Springer LNCS 85.

[AM 75] Arbib, M.A. and Manes, E.G. Arrow, Structures and Functors: the Categorical Imperative. Academic Press.

[Bar 74] Barwise, K.J. Axioms for abstract model theory. Annals of Math. Logic 7 pp. 221-265.

[Bau 81] Bauer, F.L. *et al* (the CIP Language Group) Report on a wide spectrum language for program specification and development (tentative version). Report TUM-I8104, Technische Univ. München.

[BR 83] Benecke, K. and Reichel, H. Equational partiality. Algebra Universalis 16 pp. 219-232.

[BM 81] Bergstra, J.A. and Meyer, J.J. I/O computable data structures. SIGPLAN Notices 16, 4 pp. 27-32.

[BW 82] Bloom, S.L. and Wagner, E.G. Many-sorted theories and their algebras, with examples from computer science (Working paper). US-French Joint Symp. on the Applications of Algebra to Language Definition and Compilation, Fontainebleau.

[BrW 82] Broy, M. and Wirsing, M. Partial abstract types. Acta Informatica 18 pp. 47-64.

[Bur 82] Burmeister, P. Partial algebras -- survey of a unifying approach towards a two-valued model theory for partial algebras. Algebra Universalis 15 pp. 306-358.

[BG 80] Burstall, R.M. and Goguen, J.A. The semantics of Clear, a specification language. Proc. of Advanced Course on Abstract Software Specifications, Copenhagen. Springer LNCS 86, pp. 292-332.

[BG 81] Burstall, R.M. and Goguen, J.A. An informal introduction to specifications using Clear. In: The Correctness Problem in Computer Science (R.S. Boyer and J.S. Moore eds.), Academic Press, pp. 185-213.

[ETLZ 82] Ehrig, H., Thatcher, J.W., Lucas, P. and Zilles, S.N. Denotational and initial algebra semantics of the algebraic specification language LOOK. Draft report, IBM research.

[EWT 83] Ehrig, H., Wagner, E.G. and Thatcher, J.W. Algebraic specifications with generating constraints. Proc. 10th ICALP, Barcelona. Springer LNCS 154, pp. 188-202.

[GGM 76] Giarratana, V., Gimona, F. and Montanari, U. Observability concepts in abstract data type specification. Proc. 5th MFCS, Gdansk. Springer LNCS 45.

[GDLE 82]	Gogolla, M., Drosten, K., Lipeck, U. and Ehrich, H.D. Algebraic and operational semantics of specifications allowing exceptions and errors. Fb. 140, Abteilung Informatik, Univ. of Dortmund.
[Gog 77]	Goguen, J.A. Abstract errors for abstract data types. Proc. IFIP Working Conf. on the Formal Description of Programming Concepts, New Brunswick, New Jersey.
[Gog 78]	Goguen, J.A. Order sorted algebras: exceptions and error sorts, coercions and overloaded operators. Semantics and Theory of Computation Report No. 14, Dept. of Computer Science, UCLA.
[GB 78]	Goguen, J.A. and Burstall, R.M. Some fundamental algebraic tools for the semantics of computation. Research Report No. 5, Dept. of Artificial Intelligence, Univ. of Edinburgh; to appear in TCS.
[GB 83]	Goguen, J.A. and Burstall, R.M. Introducing institutions. Proc. Logics of Programming Workshop (E. Clarke, ed.), CMU.
[GM 82]	Goguen, J.A. and Meseguer, J. Universal realization, persistent interconnection and implementation of abstract modules. Proc. 9th ICALP, Aarhus, Denmark. Springer LNCS 140, pp. 265-281.
[GM 83]	Goguen, J.A. and Meseguer, J. An initiality primer. Draft report, SRI International.
[Kam 83]	Kamin, S. Final data types and their specification. TOPLAS 5, 1 pp. 97-121.
[LB 77]	Liskov, B.H. and Berzins, V. An appraisal of program specifications. Computation Structures Group memo 141-1, Laboratory for Computer Science, MIT.
[MacL 71]	MacLane, S. Categories for the Working Mathematician. Springer.
[MM 83]	Mahr, B. and Makowsky, J.A. Characterizing specification languages which admit initial semantics. To appear in TCS.
[Pep 83]	Pepper, P. On the correctness of type transformations. Talk at 2nd Workshop on Theory and Applications of Abstract Data Types, Passau.
[Rei 81]	Reichel, H. Behavioural equivalence -- a unifying concept for initial and final specification methods. Proc. 3rd Hungarian Computer Science Conf., Budapest, pp. 27-39.
[Rei 84]	Reichel, H. Structural induction on partial algebras. Akademie-Verlag, Berlin.
[San 83]	Sannella, D.T. Behavioural equivalence and algebraic specification (extended abstract). Workshop on Semantics of Programming Languages, Göteborg, pp. 162-166.
[SW 83]	Sannella, D.T. and Wirsing, M. A kernel language for algebraic specification and implementation. Long version: Report CSR-131-83, Dept. of Computer Science, Univ. of Edinburgh; Extended abstract in: Proc. Intl. Conf. on Foundations of Computation Theory, Borgholm, Sweden. Springer LNCS 158, pp. 413-427.
[Sch 82]	Schoett, O. A theory of program modules, their specification and implementation (extended abstract). Report CSR-155-83, Dept. of Computer Science, Univ. of Edinburgh.
[Suf 82]	Sufrin, B. Formal specification of a display-oriented text editor. Science of Computer Programming 1 pp. 157-202.
[Tar 83]	Tarlecki, A. Free constructions in algebraic institutions. Long version: Report CSR-149-83, Dept. of Computer Science, Univ. of Edinburgh; Extended abstract in: Proc. MFCS 84, Prague. Springer LNCS (to appear).
[Tar 84]	Tarlecki, A. Free constructions in abstract algebraic institutions. Draft report, Univ. of Edinburgh.
[Wand 79]	Wand, M. Final algebra semantics and data type extensions. JCSS 19 pp. 27-44.
[Wir 82]	Wirsing, M. Structured algebraic specifications. Proc. AFCET Symp. on Mathematics for Computer Science, Paris, pp. 93-107.
[Wir 83]	Wirsing, M. Structured algebraic specifications: a kernel language. Habilitation thesis, Technische Univ. München.
[ZLT 82]	Zilles, S.N., Lucas, P. and Thatcher, J.W. A look at algebraic specifications. IBM research report RJ 3568.