Donald Sannella and Andrzej Tarlecki

Foundations of Algebraic Specification and Formal Software Development

September 29, 2010

Springer

Contents

Chapter 4
Working within an arbitrary logical system

Several approaches to specification were discussed in Chapter 2. Each approach involved a different *logical system* as a part of its mathematical underpinnings. We encountered different definitions of:

Signatures: "ordinary" many-sorted signatures, signatures containing *bool*, *true* and *false* (for final and reachable semantics), error signatures, order-sorted signatures;
Algebras (on a signature Σ): "ordinary" Σ-algebras, error Σ-algebras, partial Σ-algebras, order-sorted Σ-algebras;
Logical sentences (on a signature Σ): Σ-equations, conditional Σ-equations, error Σ-equations (with safe and unsafe variables), Σ-definedness formulae, order-sorted Σ-equations; and
Satisfaction (of a Σ-sentence by a Σ-algebra): of a Σ-equation by a (total) Σ-algebra, of an error Σ-equation by an error Σ-algebra, of a Σ-equation by a partial Σ-algebra, of a Σ-definedness formula by a partial Σ-algebra, of an order-sorted Σ-equation by an order-sorted Σ-algebra.

All of these choices can be combined to obtain many different logical systems and hence different approaches to specification, e.g. partial error specifications with conditional axioms. Not only that, but there are several alternative approaches to the specification of partial algebras and at least half a dozen to the specification of error handling. Furthermore, there are many other variations that have not been considered, including the following (some of them briefly mentioned in Section 2.7.6):

- polymorphic signatures which permit polymorphic type constructors (rather than just sorts) and operations having polymorphic types;
- continuous algebras to handle infinite data objects such as streams;
- higher-order algebras to handle higher-order functions (i.e. functions taking functions as arguments and/or yielding functions as results);
- relational structures to model specifications containing predicates;
- inequations and conditional inequations;
- first-order formulae, with and without equality;

- various modal logics, including algorithmic, dynamic, and temporal logics, for formulating properties of (possibly non-functional) programs.

Some of these variations depart quite considerably from the usual algebraic framework presented in Chapters 1 and 2. But none of them (and very few of the others considered in the literature) are artificial, resulting merely from a theoretician's toying with formal definitions. All of them arise from the practical need to specify different aspects of software systems, often reflected by diverse features of different programming languages.

The resulting wealth of choice of definitions of the basic concepts is not a bad thing. None of the logical systems used in specifications is clearly better than all the others — and we should not expect that such a "best" system will ever be developed. In theory, we can imagine putting all of the above concepts together, producing a single logical system where signatures, algebras, sentences and the satisfaction relation would cover as special cases all we have considered up to now. But the result would be so huge and complex as to make it unmanageable. Moreover, what would we do if one day somebody points out that yet another view of software is important and should be reflected in specifications, and hence included in the logical system we use? Scrap everything and start again?

Different specification tasks may call for different systems to express most conveniently the properties required. Moreover, different logical systems may be appropriate for describing different aspects of the same software system, and so a number of logical systems may be useful in a single specification task. It is thus important that the designer of a software system be able to choose which logical system(s) to use.

An unfortunate effect of this necessary wealth of choice is that research on specification sometimes appears to be a confused mess, where everybody adopts a different combination of basic definitions. This makes it difficult to build on the work of others, to compare the results obtained for different logical systems, and to transfer results from one system to another. This is even more disturbing when one realises that such results include not only mathematical definitions and theorems, but also practically useful tools supporting software specification, development and verification produced at great expense of effort, time and money.

In fact, much of the work done turns out to be independent of the particular choice of the basic definitions, although this is often not obvious. The main objective of this chapter, and one of the main objectives of this book, is to lay out the mathematical foundations necessary to make this independence explicit. We achieve this using the notion of an *institution* which formalises the informal concept of a logical system devised to fit the purposes of specification theory; see Section 4.1 below for the definition. Our thesis is that building as much as possible on the notion of an institution brings important benefits for both the theory and the practice of software specification and development. On one hand, this allows much work on theories, results, and practical tools to be done just once for many different specific logical systems; on the other hand it forces, via abstraction, a better understanding of and deeper insight into the real problems.

A first example of this general approach is given in Section 4.2, where we recast the fundamental ideas of the standard approach to specification from Chapter 2 in the framework of an arbitrary institution.

It should be stressed that the notion of an institution captures only certain aspects of the informal concept of a logical system. In particular, it takes a model-theoretic view of logical systems, and no direct attempt is made to accommodate proof-theoretic concepts. See Section 9.1 for a discussion of how proof fits into the picture.

When discussing different approaches to specification in Chapter 2, apart from various basic notions of signature, algebra, sentence and satisfaction, we also considered different kinds of models (algebras satisfying a set of axioms) as particularly interesting:

- the initial models;
- the reachable models satisfying $\forall\varnothing\bullet \mathit{true} \neq \mathit{false}$;
- the final models in the category of reachable models satisfying $\forall\varnothing\bullet \mathit{true} \neq \mathit{false}$.

These options, although important for the overall style of specification, are of a different nature than the choice of the basic definitions embodied in the particular institution used. We show in Section 4.3 how such "interesting models" may be singled out in an arbitrary institution, thus suggesting that the choice here is in a sense orthogonal to the choice of the underlying institution.

Our general programme is to strive to work in an arbitrary institution as much as possible. However, the concepts involved in the basic theory of institutions are often too general, and hence too weak, to express all that is necessary. When this happens, it would be premature to give up, and switch to working in a particular institution. The "game" is then to identify a (hopefully) minimal set of additional assumptions under which the job can be done, covering most or all of the logical systems of interest. This gives rise to an enriched notion of institution with some additional structure that is relevant to the particular purpose we have in mind. A few examples of this are given in Sections 4.4 and 4.5.

Before proceeding we should warn the reader that although working in an arbitrary institution is very important, it is only one side of the story. The other side is to define an institution appropriate for the needs of the particular task at hand, and quite often this is far from trivial. Indeed, in many areas of Computer Science, the fundamental problem yet to be satisfactorily solved is the development of a logical system appropriate for the aspects of computing addressed. An example of an area for which a satisfactory, commonly accepted solution still seems to be outstanding (despite numerous proposals and active research) is the theory of concurrency.

4.1 Institutions

Following Goguen and Burstall [GB92], we introduce the notion of an *institution*, capturing some essential aspects of the informal concept of a "logical system". The basic ingredients of an institution are: a notion of a signature in the system, and then for each signature, notions of an algebra with this signature, of a logical sentence over this signature, and finally a satisfaction relation between algebras and sentences.

In contrast to classical logic and model theory, we are not content with considering logical systems "pointwise", for an "arbitrary but fixed" signature. To capture the process of building a specification and designing a software system, some means of moving from one signature to another is required, that is, some notion of signature morphism. These typically enable signatures to be extended by new components, renaming and/or identifying others, as well as hiding some components used "internally" but not intended to be visible "externally". Any signature morphism should give rise to a translation of sentences and a translation of algebras determined by the change of names involved. Furthermore, these translations must be consistent with one another, preserving the satisfaction relation. As usual, when we switch from syntax (signatures, sentences) to semantics (algebras), the direction of translation is reversed.

The language of category theory is used in the definition to express the above ideas. This concisely and elegantly captures structure arising from signature morphisms, as well as forcing an appropriate level of generality and abstraction.

Definition 4.1.1 (Institution). An *institution* **INS** consists of:

- a category $\mathbf{Sign_{INS}}$ of *signatures*;
- a functor $\mathbf{Sen_{INS}}: \mathbf{Sign_{INS}} \to \mathbf{Set}$, giving a set $\mathbf{Sen}(\Sigma)$ of Σ-*sentences* for each signature $\Sigma \in |\mathbf{Sign_{INS}}|$ and a function $\mathbf{Sen_{INS}}(\sigma): \mathbf{Sen_{INS}}(\Sigma) \to \mathbf{Sen_{INS}}(\Sigma')$ translating Σ-sentences to Σ'-sentences for each signature morphism $\sigma: \Sigma \to \Sigma'$;
- a functor $\mathbf{Mod_{INS}}: \mathbf{Sign}^{op}_{\mathbf{INS}} \to \mathbf{Cat}$, giving a category $\mathbf{Mod}(\Sigma)$ of Σ-*models* for each signature $\Sigma \in |\mathbf{Sign_{INS}}|$ and a functor $\mathbf{Mod_{INS}}(\sigma): \mathbf{Mod_{INS}}(\Sigma') \to \mathbf{Mod_{INS}}(\Sigma)$ translating Σ'-models to Σ-models (and Σ'-morphisms to Σ-morphisms) for each signature morphism $\sigma: \Sigma \to \Sigma'$; and
- for each $\Sigma \in |\mathbf{Sign_{INS}}|$, a *satisfaction relation* $\models_{\mathbf{INS},\Sigma} \subseteq |\mathbf{Mod_{INS}}(\Sigma)| \times \mathbf{Sen_{INS}}(\Sigma)$

such that for any signature morphism $\sigma: \Sigma \to \Sigma'$ the translations $\mathbf{Mod_{INS}}(\sigma)$ of models and $\mathbf{Sen_{INS}}(\sigma)$ of sentences preserve the satisfaction relation, that is, for any $\varphi \in \mathbf{Sen_{INS}}(\Sigma)$ and $M' \in |\mathbf{Mod_{INS}}(\Sigma')|$:

$$M' \models_{\mathbf{INS},\Sigma'} \mathbf{Sen_{INS}}(\sigma)(\varphi) \quad \text{iff} \quad \mathbf{Mod_{INS}}(\sigma)(M') \models_{\mathbf{INS},\Sigma} \varphi$$

[*Satisfaction condition*]

□

We will freely use standard terminology, and for example say that a Σ-model M *satisfies* a Σ-sentence φ, or that φ *holds* in M, whenever $M \models_{\mathbf{INS},\Sigma} \varphi$.

The term "model" (which we use following [GB92]) thereby becomes overloaded: it is used to refer both to objects in the category $\mathbf{Mod_{INS}}(\Sigma)$ and to the algebras which satisfy a given set of axioms (we will soon extend the latter terminology to an arbitrary institution in Section 4.2, and then to an arbitrary structured specification in Chapter 5). Hopefully, this will not lead to confusion as the context will always determine which of the two meanings is meant. If in doubt, we will use "a Σ-model" (where Σ is a signature) for the former, and "a model of Φ" (where Φ is a set of sentences) for the latter meaning of the word.

Notation.

- When there is no danger of confusion, we will omit the subscript **INS** when referring to the components of an institution **INS**. Similarly, the subscript Σ on the satisfaction relations will often be omitted.
- For any signature morphism $\sigma\colon\Sigma \to \Sigma'$, the function $\mathbf{Sen}(\sigma)\colon\mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$ will be denoted simply by $\sigma\colon\mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$ and the functor $\mathbf{Mod}(\sigma)\colon\mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$ by $\_|_\sigma\colon\mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$. Thus for any Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$, $\sigma(\varphi) \in \mathbf{Sen}(\Sigma')$ is its *σ-translation* to a Σ'-sentence, and for any Σ'-model $M' \in |\mathbf{Mod}(\Sigma')|$, $\mathbf{M}'|_\sigma \in |\mathbf{Mod}(\Sigma)|$ is its *σ-reduct* to a Σ-model. We will also refer to M' as a *σ-expansion* of $M'|_\sigma$. Using this notation, the satisfaction condition of Definition 4.1.1 may be expressed as follows: $M' \models \sigma(\varphi) \iff M'|_\sigma \models \varphi$.
- For any signature Σ, the satisfaction relation extends naturally to sets of Σ-sentences and classes[1] of Σ-models. Namely, for any set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences and model $M \in |\mathbf{Mod}(\Sigma)|$, $M \models \Phi$ means $M \models \varphi$ for all $\varphi \in \Phi$. Then, for any Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$ and class $\mathscr{M} \subseteq |\mathbf{Mod}(\Sigma)|$ of Σ-models, $\mathscr{M} \models \varphi$ means $M \models \varphi$ for all $M \in \mathscr{M}$. Finally, we will also write $\mathscr{M} \models \Phi$ with the obvious meaning.
- For any signature Σ, we will sometimes write $Mod(\Sigma)$ for the class $|\mathbf{Mod}(\Sigma)|$ of all Σ-models. □

The definition of an institution as given above is very general and covers many logical systems of interest, as illustrated by the examples below. Nevertheless, it does impose some restrictions which should be made explicit before we proceed further.

First, the assumption that the translations of sentences and models induced by signature morphisms are functors may seem overly restrictive. In some situations it would be natural to relax the requirement of functoriality and assume that **Sen** (and perhaps **Mod** as well) is a functor only "up to some appropriate equivalence". For example, given two signature morphisms $\sigma\colon\Sigma \to \Sigma'$ and $\sigma'\colon\Sigma' \to \Sigma''$, for any sentence $\varphi \in \mathbf{Sen}(\Sigma)$ it follows from the functoriality of **Sen** that $\mathbf{Sen}(\sigma;\sigma')(\varphi) = \mathbf{Sen}(\sigma')(\mathbf{Sen}(\sigma)(\varphi))$ (or using the notational convention introduced above, $(\sigma;\sigma')(\varphi) = \sigma'(\sigma(\varphi))$). This seems overly restrictive when, for example, local identifiers or bound variables are used in sentences. All we really care about here is that the two translations of φ to a Σ''-sentence are *semantically equivalent*: that $(\sigma;\sigma')(\varphi)$ and $\sigma'(\sigma(\varphi))$ hold in the same Σ''-models. A solution

[1] We will be somewhat more careful about the set-theoretical foundations than in our presentation of the basics of category theory in Chapter 3: we will refer to collections of sentences as "sets" and to collections of models as "classes", as in Chapter 2. This is consistent with the formal definition of an institution above, and satisfactory for the logical systems formalised as institutions given as examples (but see Example 4.1.46, footnote 16).

is to consider sentences up to this semantic equivalence, and work in an institution where sentences simply *are* the corresponding equivalence classes. This solution would resemble the usual practice in λ-calculi, where terms are considered "up to α-conversion" (renaming of bound variables), meaning that terms are really classes of mutually α-convertible syntactic terms.

The only explicit requirement in the definition of an institution is that the satisfaction condition holds. Speaking informally, this deals with the situation where a "small" signature Σ and a "big" signature Σ' are related by a signature morphism $\sigma: \Sigma \to \Sigma'$, and we have a model $M' \in |\mathbf{Mod}(\Sigma')|$ over the "big" signature, and a sentence $\varphi \in \mathbf{Sen}(\Sigma)$ over the "small" signature. There are then two ways to check whether M' "satisfies" φ: we can either reduce the model M' to the "small" signature and check whether the reduct satisfies the sentence φ, or translate the sentence φ to the "big" signature and check whether the translated sentence holds in the model M'.

$$
\begin{array}{lccccc}
\text{“big”} & \Sigma' & & M' & \models_{\Sigma'} & \sigma(\varphi) \\
& \Big\uparrow \sigma & & \Big\downarrow & & \Big\uparrow \\
\text{“small”} & \Sigma & & M'|_\sigma & \models_{\Sigma} & \varphi
\end{array}
$$

The satisfaction condition states that these two alternatives are equivalent. This embodies two fundamental assumptions. One is that the meaning of a sentence depends only on the components used in the sentence, and does not depend on the context in which the sentence is considered. The other is that the meaning of a sentence is preserved under translation; as [GB92] say:

Truth is invariant under change of notation.

The latter requirement does not raise much doubt — we are not aware of any natural system in which it would not hold. The former, however, is sometimes violated. There are natural logical systems where the meaning of a sentence depends on the context in which it is used, or in other words on the signature over which the sentence is considered. For instance, in logical systems involving quantifiers, the range of quantification may implicitly depend on the signature, with quantified variables ranging only over reachable values, so that "$\exists x \bullet \ldots$" is interpreted as "there exists an element x which is the value of a ground term, such that $\ldots$" and similarly for universal quantification. For such a logic the satisfaction condition does not hold unless very strong restrictions are placed on signature morphisms.

Exercise 4.1.2. Give a concrete counterexample to the satisfaction condition for a logical system similar to equational logic, but with the universally quantified variables in equations ranging only over reachable values. Show how the logical system you give may be modified to make the satisfaction condition hold. HINT: The satisfaction condition failed because the interpretation of universal quantification over reachable values implicitly depends on the signature; try to make this dependence explicit! □

4.1.1 Examples of institutions

Example 4.1.3 (Ground equational logic GEQ). The institution **GEQ** of ground equational logic is defined as follows:

- The category $\mathbf{Sign}_{\mathbf{GEQ}}$ is just **AlgSig**, the usual category of algebraic signatures.
- The functor $\mathbf{Sen}_{\mathbf{GEQ}}\colon \mathbf{AlgSig} \to \mathbf{Set}$ gives:
  - the set of ground Σ-equations for each $\Sigma \in |\mathbf{AlgSig}|$; and
  - the σ-translation function taking ground Σ-equations to ground Σ'-equations for each signature morphism $\sigma\colon \Sigma \to \Sigma'$.
- The functor $\mathbf{Mod}_{\mathbf{GEQ}}\colon \mathbf{AlgSig}^{op} \to \mathbf{Cat}$ is the functor $\mathbf{Alg}\colon \mathbf{AlgSig}^{op} \to \mathbf{Cat}$ as defined in Example 3.4.29, that is, $\mathbf{Mod}_{\mathbf{GEQ}}$ gives:
  - the category $\mathbf{Alg}(\Sigma)$ of Σ-algebras and Σ-homomorphisms for each $\Sigma \in |\mathbf{AlgSig}|$; and
  - the reduct functor $\_|_{\sigma}\colon \mathbf{Alg}(\Sigma') \to \mathbf{Alg}(\Sigma)$ mapping Σ'-algebras and Σ'-homomorphisms to Σ-algebras and Σ-homomorphisms for each signature morphism $\sigma\colon \Sigma \to \Sigma'$.
- For each $\Sigma \in |\mathbf{AlgSig}|$, the satisfaction relation $\models_{\mathbf{GEQ},\Sigma} \subseteq |\mathbf{Alg}(\Sigma)| \times \mathbf{Sen}_{\mathbf{GEQ}}(\Sigma)$ is the usual relation of satisfaction of a ground Σ-equation by a Σ-algebra.

The Satisfaction Lemma (Lemma 2.1.8) ensures that the required satisfaction condition holds and so that the above definition indeed yields an institution. □

Example 4.1.4 (Equational logic EQ). The institution **EQ** of (ordinary) equational logic is defined as follows:

- The category $\mathbf{Sign}_{\mathbf{EQ}}$ is just **AlgSig**.
- The functor $\mathbf{Sen}_{\mathbf{EQ}}\colon \mathbf{AlgSig} \to \mathbf{Set}$ gives:
  - the set of Σ-equations for each $\Sigma \in |\mathbf{AlgSig}|$; and
  - the σ-translation function taking Σ-equations to Σ'-equations for each signature morphism $\sigma\colon \Sigma \to \Sigma'$.[2]
- The functor $\mathbf{Mod}_{\mathbf{EQ}}$ is $\mathbf{Alg}\colon \mathbf{AlgSig}^{op} \to \mathbf{Cat}$, just like $\mathbf{Mod}_{\mathbf{GEQ}}$ for ground equational logic.

[2] The exact treatment of variables in equations requires special care to ensure that the translation of equations along possibly non-injective signature morphisms is indeed functorial. The use of disjoint union in the translation of many-sorted sets of variables in Definition 1.5.10 causes problems here. The simplest way to make this work is to assume that, in each equation, variables of different sorts are distinct. See [GB92] for details.

- For each $\Sigma \in |\mathbf{AlgSig}|$, the satisfaction relation $\models_{\mathbf{EQ},\Sigma} \subseteq |\mathbf{Alg}(\Sigma)| \times \mathbf{Sen_{EQ}}(\Sigma)$ is the usual relation of satisfaction of a Σ-equation by a Σ-algebra.

The Satisfaction Lemma (Lemma 2.1.8) again ensures that the required satisfaction condition holds and so that the above definition indeed yields an institution. □

There is an obvious sense in which **GEQ** can be regarded as a "subinstitution" of **EQ**. We will encounter further such cases below. We refrain from formulating a notion of subinstitution because the concept turns out to be more subtle than it might appear at first. We postpone a proper treatment of relationships between institutions to Chapter 10 (in particular, see Exercise 10.4.8).

Exercise 4.1.5 (Reachable ground equational logic RGEQ). Define an institution **RGEQ** of ground equational logic on reachable algebras, by modifying the definion of **GEQ** so that only reachable algebras are considered as models. Do not forget to adjust the definition of reduct functors!

Try to extend this to an institution **REQ** of equational logic on reachable algebras — and notice that the satisfaction condition cannot be ensured without modifying the notion of an equation to include "data constructors" to determine the reachable values for which the equation is to be considered, as already hinted at in Exercise 4.1.2. □

Example 4.1.6 (Partial equational logic PEQ). The institution **PEQ** of partial equational logic is defined as follows (cf. Section 2.7.4):

- $\mathbf{Sign_{PEQ}}$ is **AlgSig** again.
- $\mathbf{Sen_{PEQ}}\colon \mathbf{AlgSig} \to \mathbf{Set}$ gives:
  - the set of Σ-equations and Σ-definedness formulae for each $\Sigma \in |\mathbf{AlgSig}|$; and
  - the σ-translation function taking Σ-equations and Σ-definedness formulae to Σ'-equations and Σ'-definedness formulae for each signature morphism $\sigma\colon \Sigma \to \Sigma'$.[3]
- $\mathbf{Mod_{PEQ}}\colon \mathbf{AlgSig}^{op} \to \mathbf{Cat}$ gives:
  - the category $\mathbf{PAlg}(\Sigma)$ of partial Σ-algebras and weak Σ-homomorphisms for each $\Sigma \in |\mathbf{AlgSig}|$ (cf. Example 3.3.13); and
  - the reduct functor $\_|_\sigma\colon \mathbf{PAlg}(\Sigma') \to \mathbf{PAlg}(\Sigma)$ defined similarly as in the total case for each signature morphism $\sigma\colon \Sigma \to \Sigma'$.
- For each $\Sigma \in |\mathbf{AlgSig}|$, the satisfaction relation $\models_{\mathbf{PEQ},\Sigma} \subseteq |\mathbf{PAlg}(\Sigma)| \times \mathbf{Sen_{PEQ}}(\Sigma)$ is the satisfaction of Σ-equations (with strong equality) and Σ-definedness formulae by partial Σ-algebras.

Exercise. Proceeding similarly as in the proof of Satisfaction Lemma (Lemma 2.1.8), show that the satisfaction condition holds for **PEQ**. □

[3] As in Example 4.1.4, care is needed with the treatment of variables and their translation under signature morphisms, see footnote 2.

Example 4.1.7 (Ground partial equational logic PGEQ). The institution **PGEQ** of ground partial equational logic is defined just like the institution **PEQ** of partial equational logic above, except that only ground equations and ground definedness formulae are considered. □

Exercise 4.1.8. Recalling the notion of existential equality for partial algebras from Section 2.7.4, define institutions $\mathbf{PEQ}^e$ and $\mathbf{PGEQ}^e$ of partial existence equational logic and ground partial existence equational logic, respectively, modifying the definitions in Examples 4.1.6 and 4.1.7 by using existential equations of the form $\forall X . t \stackrel{e}{=} t'$ and their ground versions only. □

Example 4.1.9 (Propositional logic PROP). The institution **PROP** of propositional logic is defined as follows:

- $\mathbf{Sign_{PROP}}$ is **Set**, the usual category of sets. In this context, for each "signature" $P \in |\mathbf{Set}|$, we call elements of P *propositional variables*.
- $\mathbf{Sen_{PROP}}\colon \mathbf{Set} \to \mathbf{Set}$ gives
  - For each $P \in |\mathbf{Set}|$, $\mathbf{Sen_{PROP}}(P)$ is the least set that contains P, sentences true and false, and is closed under the usual propositional connectives, that is, if $\varphi, \varphi' \in \mathbf{Sen_{PROP}}(\Sigma)$ then also $\varphi \vee \varphi' \in \mathbf{Sen_{PROP}}(\Sigma)$, $\neg\varphi \in \mathbf{Sen_{PROP}}(\Sigma)$, $\varphi \wedge \varphi' \in \mathbf{Sen_{PROP}}(\Sigma)$, and $\varphi \Rightarrow \varphi' \in \mathbf{Sen_{PROP}}(\Sigma)$. [4]
  - For each function $\sigma\colon P \to P'$, $\mathbf{Sen_{PROP}}(\sigma)$ extends σ to the translation of arbitrary propositional sentences with propositional variables in P to propositional sentences with propositional variables in P', preserving the propositional connectives in the obvious way.
- $\mathbf{Mod_{PROP}}\colon \mathbf{Set}^{op} \to \mathbf{Cat}$ gives:
  - For each set of propositional variables $P \in |Set|$, P-models are all functions from P to $\{\mathit{ff}, \mathit{tt}\}$. These functions can be identified with subsets of P, where $M\colon P \to \{\mathit{ff}, \mathit{tt}\}$ yields $\{p \in P \mid M(p) = \mathit{tt}\}$). Model morphisms are just inclusions of these sets, i.e., given two P-models $M_1, M_2\colon P \to \{\mathit{ff}, \mathit{tt}\}$, we have a (unique) morphism from M_1 to M_2 if for all $p \in P$, $M_2(p) = \mathit{tt}$ whenever $M_1(p) = \mathit{tt}$.
  - For each signature morphism $\sigma\colon P \to P'$, the reduct functor $\mathbf{Mod_{PROP}}(\sigma)\colon \mathbf{Mod_{PROP}}(P') \to \mathbf{Mod_{PROP}}(P)$ maps any model $M'\colon P' \to \{\mathit{ff}, \mathit{tt}\}$ to $\sigma;M'\colon P \to \{\mathit{ff}, \mathit{tt}\}$.
- For each $P \in |\mathbf{Set}|$, the satisfaction relation $\models_{\mathbf{PROP},P} \subseteq |\mathbf{Mod_{PROP}}(P)| \times \mathbf{Sen_{PROP}}(P)$ is the usual relation of satisfaction of propositional sentences, that is, for any P-model $M\colon P \to \{\mathit{ff}, \mathit{tt}\}$, $p \in P$ and $\varphi, \varphi' \in \mathbf{Sen_{PROP}}(P)$:
  - $M \models_{\mathbf{PROP},P} p$ if and only if $M(p) = \mathit{tt}$,
  - $M \models_{\mathbf{PROP},P} \varphi \vee \varphi'$ if and only if $M \models_{\mathbf{PROP},P} \varphi$ or $M \models_{\mathbf{PROP},P} \varphi'$,
  - $M \models_{\mathbf{PROP},P} \neg\varphi$ if and only if $M \not\models_{\mathbf{PROP},P} \varphi$,
  - $M \models_{\mathbf{PROP},P} \varphi \wedge \varphi'$ if and only if $M \models_{\mathbf{PROP},P} \varphi$ and $M \models_{\mathbf{PROP},P} \varphi'$.

[4] We tacitly assume here that true, false, $\vee$, $\wedge$, $\Rightarrow$, $\neg$ are new symbols (not in P), and rely on the usual precedence rules and parentheses to make sure that no ambiguities in their "parsing" arise.

– $M \models_{\mathbf{PROP},P} \varphi \Rightarrow \varphi'$ if and only if $M \models_{\mathbf{PROP},P} \varphi'$ or $M \not\models_{\mathbf{PROP},P} \varphi$
– $M \models_{\mathbf{PROP},P} \mathsf{true}$, and
– $M \not\models_{\mathbf{PROP},P} \mathsf{false}$. □

Exercise 4.1.10. Recall the specification of Boolean algebras in Example 2.2.4.

Note that one way to view the definitions in Example 4.1.9 is to define the set of P-sentences as Boolean terms with variables from P. Then, one can consider the two-element Boolean algebra $\mathbb{B}$ with the carrier $\{ff, tt\}$ (with $\mathsf{true}_{\mathbb{B}} = tt$ and $\mathsf{false}_{\mathbb{B}} = ff$). Furthermore, any propositional model $M{:}P \to \{ff, tt\}$ induces evaluation of terms $M^\sharp{:}\mathbf{Sen_{PROP}}(P) \to |\mathbb{B}|$, with $M^\sharp(\varphi) = tt$ if and only if $M \models_{\mathbf{PROP},P} \varphi$ as defined above.

Define another institution of propositional logic, $\mathbf{PROP^{BA}}$, where signatures and sentences are as in **PROP**, but models use arbitrary Boolean algebras rather than just $\mathbb{B}$. That is, for any set $P \in |\mathbf{Set}|$ of propositional variables, a P-model in $\mathbf{PROP^{BA}}$ consists of a Boolean algebra B together with valuation $M{:}P \to |B|$, where we define $\langle B, M\rangle \models_{\mathbf{PROP^{BA}},P} \varphi$ if and only if $\varphi_B(M) = \mathsf{true}_B$ (where $\varphi_B(M)$ is the value of term φ in B under valuation M).

Prove now that the semantic consequence relation (Definition 2.3.6, cf. Definition 4.2.5 below) in **PROP** and $\mathbf{PROP^{BA}}$ coincide.

HINT: Clearly, if $\Psi \models_{\mathbf{PROP^{BA}},P} \varphi$ then also $\Psi \models_{\mathbf{PROP},P} \varphi$ for any set P of propositional variables, $\Psi \subseteq \mathbf{Sen_{PROP}}(P)$ and $\varphi \in \mathbf{Sen_{PROP}}(P)$. Suppose now that $\Psi \not\models_{\mathbf{PROP^{BA}},P} \varphi$. Use the following lemma[5]:

Lemma. *Given any Boolean algebra B and element $b \in |B|$ such that $b \neq \mathsf{true}_B$, there exists a homomorphism $h{:}B \to \mathbb{B}$ from B to the two-element Boolean algebra $\mathbb{B}$ such that $h(b) = \mathsf{false}_{\mathbb{B}}$.*

Now, given any Boolean algebra B and valuation $M{:}P \to |B|$ such that for all $\psi \in \Psi$, $\psi_B(M) = \mathsf{true}_B$ and $\varphi_B(M) \neq \mathsf{true}_B$, conclude using the above lemma that $(M;h)^\sharp(\psi) = tt$ for all $\psi \in \Psi$, while $(M;h)^\sharp(\varphi) = ff$. □

Exercise 4.1.11. Define the institution of intuitionistic propositional logic, $\mathbf{PROP^{I}}$, following the pattern of $\mathbf{PROP^{BA}}$ in Exercise 4.1.10, but using arbitrary Heyting algebras (see Example 2.7.6) rather than just Boolean algebras.

Show that if $\Psi \models_{\mathbf{PROP^{I}},P} \varphi$ then also $\Psi \models_{\mathbf{PROP},P} \varphi$ for any set P of propositional variables, $\Psi \subseteq \mathbf{Sen_{PROP}}(P)$ and $\varphi \in \mathbf{Sen_{PROP}}(P)$, and give a counterexample to show that the opposite implication fails in general. □

Example 4.1.12 (First-order predicate logic with equality FOPEQ). The institution **FOPEQ** of first-order predicate logic with equality is defined as follows:

- $\mathbf{Sign_{FOPEQ}}$, from now on denoted by **FOSig**, is the category of *first-order signatures* where we define:

[5] The proof of this lemma is beyond the scope of this book, but see e.g. [RS63], I,8.5 and II,5.2,(a)⇒(e).

– A *first-order signature* Θ is a triple $\langle S, \Omega, \Pi \rangle$, where S is a set (of *sort names*), $\Omega = \langle \Omega_{w,s} \rangle_{w \in S^*, s \in S}$ is a family of sets (of *operation names* with their arities and result sorts indicated — just as in algebraic signatures) and $\Pi = \langle \Pi_w \rangle_{w \in S^*}$ is a family of sets (of *predicate* or *relation names* with their arities indicated).
– A *first-order signature morphism* $\theta : \langle S, \Omega, \Pi \rangle \to \langle S', \Omega', \Pi' \rangle$ consists again of three components: a function $\theta_{sorts} : S \to S'$, an $S^* \times S$-indexed family of functions $\theta_{ops} = \langle (\theta_{ops})_{w,s} : \Omega_{w,s} \to \Omega'_{\theta^*_{sorts}(w), \theta_{sorts}(s)} \rangle_{w \in S^*, s \in S}$ (these are as in algebraic signature morphisms) and $\theta_{preds} = \langle (\theta_{preds})_w : \Pi_w \to \Pi'_{\theta^*_{sorts}(w)} \rangle_{w \in S^*}$. (As with algebraic signature morphisms, all the components of a first-order signature morphism θ will be denoted by θ when there is no danger of ambiguity.)

- $\mathbf{Sen_{FOPEQ}} : \mathbf{FOSig} \to \mathbf{Set}$ gives:
  – For each first-order signature $\Theta = \langle S, \Omega, \Pi \rangle$, $\mathbf{Sen_{FOPEQ}}(\Theta)$ is the set of all closed (i.e. without unbound occurrences of variables) *first-order formulae* built out of atomic formulae using the standard propositional connectives ($\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$, $\neg$) and quantifiers ($\forall$, $\exists$). The *atomic formulae* are equalities of the form $t = t'$, where t and t' are $\langle S, \Omega \rangle$-terms (possibly with variables) of the same sort, atomic predicate formulae of the form $p(t_1, \ldots, t_n)$, where $p \in \Pi_{s_1 \ldots s_n}$ and $t_1, \ldots, t_n$ are terms (possibly with variables) of sorts $s_1, \ldots, s_n$, respectively, and the logical constants `true` and `false`.
  – For each first-order signature morphism $\theta : \Theta \to \Theta'$, $\mathbf{Sen_{FOPEQ}}(\theta)$ is the translation of first-order Θ-sentences to first-order Θ'-sentences determined in the obvious way by the renaming θ of sort, operation and predicate names in Θ to the corresponding names in Θ'.[6]

- $\mathbf{Mod_{FOPEQ}} : \mathbf{FOSig}^{op} \to \mathbf{Cat}$, from now on denoted by $\mathbf{FOStr}$, gives:
  – For each first-order signature $\Theta = \langle S, \Omega, \Pi \rangle$, the category $\mathbf{FOStr}(\Theta)$ of *first-order Θ-structures* is defined as follows:
    · A *first-order Θ-structure* $A \in |\mathbf{FOStr}(\Theta)|$ consists of a carrier set $|A|_s$ for each sort name $s \in S$, a function $f_A : |A|_{s_1} \times \ldots \times |A|_{s_n} \to |A|_s$ for each operation name $f \in \Omega_{s_1 \ldots s_n, s}$ (these are the same as in $\langle S, \Omega \rangle$-algebras) and a relation $p_A \subseteq |A|_{s_1} \times \ldots \times |A|_{s_n}$ for each predicate name $p \in \Pi_{s_1 \ldots s_n}$. In the following we write $p_A(a_1, \ldots, a_n)$ for $\langle a_1, \ldots, a_n \rangle \in p_A$.
    · For any first-order Θ-structures A and B, a *first-order Θ-morphism* between them, $h : A \to B$, is a family of functions $h = \langle h_s : |A|_s \to |B|_s \rangle_{s \in S}$ which preserves the operations (as ordinary $\langle S, \Omega \rangle$-homomorphisms do) and predicates (i.e., for $p \in \Pi_{s_1 \ldots s_n}$ and $a_1 \in |A|_{s_1}, \ldots, a_n \in |A|_{s_n}$, if $p_A(a_1, \ldots, a_n)$ then $p_B(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$ as well). A Θ-morphism is *strong* if it reflects predicates as well, so that for $p \in \Pi_{s_1 \ldots s_n}$ and $a_1 \in |A|_{s_1}, \ldots, a_n \in |A|_{s_n}$, $p_A(a_1, \ldots, a_n)$ if and only if $p_B(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$.

[6] As in Example 4.1.4, some care is needed with the exact treatment of quantified variables and their translation under signature morphisms (cf. footnote 2) — again, the simplest solution is to assume that, in each formula, variables of different sorts are distinct. See [GB92] for a careful presentation.

  - For each first-order signature morphism $\theta{:}\Theta \to \Theta'$, we have the θ*-reduct functor* $\mathbf{FOStr}(\theta){:}\mathbf{FOStr}(\Theta') \to \mathbf{FOStr}(\Theta)$ defined similarly as reduct functors corresponding to algebraic signature morphisms.

- For each $\Theta \in |\mathbf{FOSig}|$, the satisfaction relation $\models_{\mathbf{FOPEQ},\Theta} \subseteq |\mathbf{FOStr}(\Theta)| \times \mathbf{Sen}_{\mathbf{FOPEQ}}(\Theta)$ s the usual relation of satisfaction of first-order sentences in first-order structures, determined by the usual interpretation of $\vee$, $\wedge$, $\Rightarrow$ and $\neg$ as disjunction, conjunction, implication and negation, respectively, of $\forall$ and $\exists$ as universal and existential quantifiers, respectively, of equalities $t = t'$ as identity of the values of t and t', of atomic predicate formulae $p(t_1,\ldots,t_n)$ as the value of the predicate named p in the structure on the values of the terms $t_1, \ldots, t_n$, and of true and false.

Exercise. Work out all the details omitted from the above definition. Then, generalising the proof of the Satisfaction Lemma, show that the satisfaction condition holds for **FOPEQ**. □

Exercise 4.1.13 (First-order predicate logic FOP, first-order logic with equality FOEQ). First-order predicate logic with equality contains some standard "sublogics". Define the institution **FOP** of first-order predicate logic (without equality), by referring to the same signatures and models as in **FOPEQ**, but limiting the sentences to those that do not contain equality.

Define also the institution **FOEQ** with signatures and models as in the institution **EQ** of equational logic, but with first-order sentences (without predicates). □

Exercise 4.1.14 (Infinitary logics). Define an institution of so-called $L_{\omega_1\omega}$ logic, which extends first-order predicate logic with equality by allowing conjunctions and disjunctions of *countable* families of formulae (but still only finitary quantification). Extend this further by allowing quantification over countable sets of variables, obtaining an institution of $L_{\omega_1\omega_1}$ logic. You may also want to define institutions of $L_{\alpha\beta}$ logics, for any infinite cardinal numbers α and β such that $\beta \leq \alpha$, with conjunctions and disjunctions of sets of formulae of cardinality smaller than α and quantification over sets of variables of cardinality smaller than β. □

Exercise 4.1.15 (Higher-order logics). Define an institution of *second-order logic*, which extends first-order logic by introducing variables ranging over predicates (which in a model denote subsets of a product of the carrier sets) and quantification over such (first-order) predicates. Then generalise this further to an institution of *higher-order logic*, which introduces variables that range over (second-order) predicates with arities that may include arities of first-order predicates, and predicates with arities that may include arities of second-order predicates, etc., and allows for quantification over such higher-order predicates. Without much additional effort, you may want to extend this further, by allowing variables that range over functions of an arbitrary higher-order type, and quantification over such functions. Note though that this will be different from first-order logic for higher-order algebras as sketched in Example 2.7.56, where quantification over higher-order function types does not necessarily coincide with quantification over *all* functions of this type. □

Exercise 4.1.16 (First-order equational logic with boolean values FOEQBool). Define an institution **FOEQBool** which differs from **FOEQ** by considering only signatures that contain a subsignature Σ_{bool} of *truth values* (Σ_{bool} has a special, distinguished sort *bool* and two constants $true, false{:}\, bool$) and assuming that signature morphisms preserve and reflect symbols in Σ_{bool} and that algebras interpret them in the standard way (the carrier of sort *bool* has exactly two distinct elements that are values of *true* and *false*, respectively).

There is now an obvious equivalence between the categories of signatures of **FOPEQ** and **FOEQBool** obtained by mapping each first-order signature to the algebraic signature with the sort *bool* and constants $true, false{:}\, bool$ added, and with new operation name $f_p{:}\, s_1 \times \ldots \times s_n \to bool$ for each predicate $p{:}\, s_1 \times \ldots \times s_n$. First-order structures give raise to algebras with the standard interpretation of Σ_{bool} and with functions f_p that yield the value of *true* exactly on those arguments for which the predicate p holds. Clearly, this yields a one-to-one correspondence between first-order structures and algebras over the corresponding signatures. However, this does not extend to model morphisms in general. (**Exercise:** Find a counterexample. Notice though that every *strong* morphism between first-order structures extends to a homomorphism between their corresponding algebras.) We then consider translation of atomic sentences $p(t_1,\ldots,t_n)$ to equalities $p(t_1,\ldots,t_n) = true$, and extend it further to arbitrary first-order sentences with predicates and equality in the obvious way.

Prove that such translations of sentences and models preserve and reflect satisfaction. □

It is not much more difficult to define, for example, the institution **PFOPEQ** of partial first-order predicate logic with equality, or any other institution formalising one of the many standard variants of the classical notions.

Exercise 4.1.17 (Partial first-order predicate logic with equality PFOPEQ). Define the institution **PFOPEQ** of partial first-order predicate logic with equality according to the following sketch:

- $\mathbf{Sign_{PFOPEQ}} = \mathbf{FOSig}$.
- For each $\Theta \in |\mathbf{FOSig}|$, partial first-order Θ-sentences are defined in the same way as usual first-order Θ-sentences on atomic formulae which here include *atomic definedness formulae* $def(t)$ for any Θ-term t, in addition to equalities and atomic predicate formulae. The translation of sentences along signature morphisms is defined in the obvious way.
- For each $\Theta \in |\mathbf{FOSig}|$, the models in $\mathbf{Mod_{PFOPEQ}}(\Theta)$ are like first-order Θ-structures except that the operations may be partial. Morphisms in $\mathbf{Mod_{PFOPEQ}}(\Theta)$ are like first-order Θ-morphisms but are required to preserve definedness of operations, as weak homomorphisms of partial algebras do. The reduct functors are defined similarly as for first-order structures.
- For each signature $\Theta \in |\mathbf{FOSig}|$, the satisfaction relation $\models_{\mathbf{PFOPEQ},\Theta}$ is defined like the usual first-order satisfaction relation, building on the interpretation of atomic equalities and definedness formulae which follows the interpretation of

(strong) equations and definedness formulae in partial algebras as defined in the institution **PEQ** of partial equational logic and on the usual interpretation of atomic predicate formulae $p(t_1,\ldots,t_n)$ which yields *false* when any of t_1, ..., t_n is undefined. □

Exercise 4.1.18 (Partial first-order logic with equality PFOEQ). Following Exercise 4.1.13, define the institution **PFOEQ** of partial first-order logic with equality with signatures and models inherited from the institution **PEQ** of partial equational logic, but with first-order sentences (without predicates). Similarly, define the institution **PFOP** of partial first-order predicate logic (without equality). □

Exercise 4.1.19 (Partial first-order equational logic with truth PFOEQTruth). As in Exercise 4.1.16, define now an institution **PFOEQBool** of partial first-order logic with equality and built-in boolean values.

However, using partial functions predicates may be modelled differently (and more faithfully when model morphisms are considered). Define an institution **PFOEQTruth** which differs from **PFOEQ** by assuming that the signatures contain a subsignature Σ_{truth} (which has a special, distinguished sort *truth* with a single constant *true*: *truth*), that signature morphisms preserve and reflect symbols in Σ_{truth}, and that algebras interpret them in the standard way: the carrier of sort *truth* has exactly one element that is the value of *true*.

The equivalence of categories of signatures and the translation of sentences between **PFOPEQ** and **PFOEQTruth** can now be given in essentially the same way as in Exercise 4.1.16. Moreover, first-order partial structures are in one-to-one correspondence with algebras over the corresponding algebraic signature, and this correspondence may be described exactly as in Exercise 4.1.16 as well. The difference is that now for arguments for which predicates do not hold, their corresponding operations are undefined instead of yielding a non-*true* value. This allows us to extend this correspondence to model morphisms as well.

Prove that such translations of sentences and models preserve and reflect satisfaction. □

Exercise 4.1.20. Recall the notion of a strong homomorphism between partial algebras (Definition 2.7.31) and between first-order structures (given in Example 4.1.12). For each of the institutions above with models that involve partial operations or predicates (**FOPEQ**, **FOP**, **PFOPEQ**, **PEQ**, etc.) define a variant in which all morphisms are strong. We will refer to these institutions as $\mathbf{FOPEQ_{str}}$, $\mathbf{FOP_{str}}$, $\mathbf{PFOPEQ_{str}}$, $\mathbf{PEQ_{str}}$, etc. In particular, model morphisms in $\mathbf{PFOPEQ_{str}}$ preserve and reflect predicates as well as definedness of operations. □

Exercise 4.1.21. Using the material in Sections 2.7.1, 2.7.3 and 2.7.5, respectively, define institutions: $\mathbf{EQ}^{\Rightarrow}$ of conditional equations with signatures and models as in **EQ**; **Horn** of Horn formulae built over signatures and models of **FOPEQ**, where sentences have the form $\forall X \bullet \varphi_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi$ for atomic formulae φ_1, ..., φ_n, φ; **ErrEQ** of error equational logic; and **OrdEQ** of order-sorted equational logic; □

Example 4.1.22 (The institution CEQ of equational logic for continuous algebras). We need some auxiliary definitions. Let $\Sigma = \langle S, \Omega \rangle$ be an algebraic signature.

Recall (cf. Example 3.3.14) that a continuous Σ-algebra $A \in |\mathbf{CAlg}(\Sigma)|$ consists of carriers, which are complete partial orders $\langle |A|_s, \leq_s \rangle$ for $s \in S$, and operations, which are continuous functions $f_A: |A|_{s_1} \times \ldots \times |A|_{s_n} \to |A|_s$ for $f: s_1 \times \ldots \times s_n \to s$ in Σ.

For any S-sorted set X (of variables), the (S-sorted) set $|T^\infty_\Sigma(X)|$ of *infinitary Σ-terms* is the least set such that[7]:

- $X \subseteq |T^\infty_\Sigma(X)|$;
- for each $f: s_1 \times \ldots \times s_n \to s$ in Σ, if $t_1 \in |T^\infty_\Sigma(X)|_{s_1}, \ldots, t_n \in |T^\infty_\Sigma(X)|_{s_n}$ then $f(t_1, \ldots, t_n) \in |T^\infty_\Sigma(X)|_s$; and
- for each $s \in S$, if for $k \geq 0$, $t_k \in T^\infty_\Sigma(X)_s$, then $\bigsqcup \langle t_k \rangle_{k \geq 0} \in |T^\infty_\Sigma(X)|_s$.

Intuitively, $|T^\infty_\Sigma(X)|$ contains all the usual finitary Σ-terms and in addition is closed under formal "least upper bounds" of countable sequences of terms. Notice, however, that we do not provide $|T^\infty_\Sigma(X)|$ with the structure of a continuous Σ-algebra; in particular, a term $\bigsqcup \langle t_k \rangle_{k \geq 0}$ is just a formal expression here, not a least upper bound.

Then, for any continuous Σ-algebra A and valuation of variables $v: X \to |A|$, we define a *partial* function $v^\#: |T^\infty_\Sigma(X)| \to |A|$ which for any term $t \in |T^\infty_\Sigma(X)|$ yields the *value* $v^\#(t)$ *of* t (if defined):

- for $x \in X$, $v^\#(x) = v(x)$;
- for $f: s_1 \times \ldots \times s_n \to s$ and $t_1 \in |T^\infty_\Sigma(X)|_{s_1}, \ldots, t_n \in |T^\infty_\Sigma(X)|_{s_n}$, $v^\#(f(t_1, \ldots, t_n))$ is defined if and only if $v^\#(t_1), \ldots, v^\#(t_n)$ are all defined, and then $v^\#(f(t_1, \ldots, t_n)) = f_A(v^\#(t_1), \ldots, v^\#(t_n))$; and
- for $t_k \in T^\infty_\Sigma(X)_s$, $k \geq 0$, $v^\#(\bigsqcup \langle t_k \rangle_{k \geq 0})$ is defined if and only if all $v^\#(t_k)$, $k \geq 0$, are defined and form a chain $v^\#(t_0) \leq_s v^\#(t_1) \leq_s \ldots$, and then $v^\#(\bigsqcup \langle t_k \rangle_{k \geq 0}) = \bigsqcup_{k \geq 0} v^\#(t_k)$ (where $\bigsqcup$ on the right hand side stands for the least upper bound in the cpo $\langle |A|_s, \leq_s \rangle$).

As usual, we write $t_A(v)$ for $v^\#(t)$.

Finally, an *infinitary Σ-equation* is a triple $\langle X, t, t' \rangle$, written $\forall X \bullet t = t'$, where X is an S-sorted set of variables[8] and $t, t' \in |T^\infty_\Sigma(X)|_s$ for some $s \in S$. A continuous Σ-algebra A *satisfies* an infinitary Σ-equation $\forall X \bullet t = t'$, written $A \models_{\mathbf{CEQ},\Sigma} \forall X \bullet t = t'$, if for all valuations $v: X \to |A|$, $t_A(v)$ and $t'_A(v)$ are both defined and equal.

We are now ready to define the institution **CEQ** of equational logic for continuous algebras:

- $\mathbf{Sign}_{\mathbf{CEQ}}$ is **AlgSig** again.
- $\mathbf{Sen}_{\mathbf{CEQ}}: \mathbf{AlgSig} \to \mathbf{Set}$ gives:
  - the set of infinitary Σ-equations for each $\Sigma \in |\mathbf{AlgSig}|$; and

[7] For simplicity, we omit the decoration of terms by their target sorts. Formally, to avoid any potential ambiguities, the definition should follow the pattern of Definition 1.4.1.

[8] For $s \in S$, the sets $X_s \subseteq \mathscr{X}$ come from a fixed vocabulary of variables as in Definition 2.1.1 and are mutually disjoint as in footnote 2.

– the σ-translation function, mapping infinitary Σ-equations to infinitary Σ'-equations in the obvious way, for each signature morphism $\sigma\colon\Sigma \to \Sigma'$.

- $\mathbf{Mod_{CEQ}}\colon \mathbf{AlgSig}^{op} \to \mathbf{Cat}$ gives:
  – the category $\mathbf{CAlg}(\Sigma)$ of continuous Σ-algebras and continuous Σ-homomorphisms for each $\Sigma \in |\mathbf{AlgSig}|$; and
  – the reduct functor $\_|_\sigma\colon \mathbf{CAlg}(\Sigma') \to \mathbf{CAlg}(\Sigma)$ defined similarly as in the case of usual (discrete) algebras for each signature morphism $\sigma\colon\Sigma \to \Sigma'$.
- For each $\Sigma \in |\mathbf{AlgSig}|$, the satisfaction relation $\models_{\mathbf{CEQ},\Sigma} \subseteq |\mathbf{CAlg}(\Sigma)| \times \mathbf{Sen_{CEQ}}(\Sigma)$ is the relation of satisfaction of infinitary Σ-equations by continuous Σ-algebras.

Exercise. Proceeding similarly as in the proof of the Satisfaction Lemma, show that the satisfaction condition holds for **CEQ**.

Exercise. Show that even though we have introduced only infinitary equations as sentences in **CEQ**, infinitary inequalities of the form $\forall X \bullet t \leq t'$ are expressible here as well. (HINT: $a \leq b$ iff $a \sqcup b = b$.) □

Exercise 4.1.23. For each of the institutions **INS** defined above, define formally its version $\mathbf{INS}^{der}$ based on the category of signatures with derived signature morphisms as presented in Section 1.5.2 (cf. Exercises 3.1.12 and 3.4.30). □

Example 4.1.24 (Three-valued first-order predicate logic with equality 3FOPEQ). We sketch here the institution **3FOPEQ** of three-valued first-order predicate logic with equality as an example of how the notion of an institution can cope with logical systems based on multiple truth values, where the interpretation of sentences may yield a number of values rather than just being true or false.

- $\mathbf{Sign_{3FOPEQ}}$ is the category **FOSig** of first-order signatures.
- $\mathbf{Sen_{3FOPEQ}}\colon \mathbf{Sign_{3FOPEQ}} \to \mathbf{Set}$ gives:
  – For each $\Theta \in |\mathbf{FOSig}|$, $\mathbf{Sen_{3FOPEQ}}(\Theta)$ is the set of sentences of the form φ **is** *tt*, φ **is** *ff*, or φ **is** *undef*, where φ is a Θ-sentence of partial first-order predicate logic with equality **PFOPEQ** (see Exercise 4.1.17).
  – For each first-order signature morphism $\theta\colon\Theta \to \Theta'$, we define the translation function $\mathbf{Sen_{3FOPEQ}}(\theta)\colon \mathbf{Sen_{3FOPEQ}}(\Theta) \to \mathbf{Sen_{3FOPEQ}}(\Theta')$ in the obvious way using the translation of first-order Θ-sentences to Θ'-sentences induced by the morphism θ.
- $\mathbf{Mod_{3FOPEQ}}\colon \mathbf{Sign}^{op}_{\mathbf{3FOPEQ}} \to \mathbf{Cat}$ is defined as usual for first-order logic, except that operations in structures are partial functions and predicates are interpreted as *partial relations*, which for any tuple of arguments may yield one of three logical values: *tt* (for truth), *ff* (for falsity) and a “third truth value” *undef* (for undefinedness).
- Atomic formulae, propositional connectives and quantifiers may be interpreted over the three-element set of truth values $\{tt, ff, undef\}$ in a number of ways, see for example [KTB91] and references there for a discussion. Here, we adopt the following interpretation:

– Atomic definedness formulae have the expected meaning: $def(t)$ is tt if the value of t is defined, and is ff otherwise.
– Equalities are interpreted as *strict equalities*: $t = t'$ is tt if the values of t and t' are defined and equal, is ff if they are defined and different, and is *undef* otherwise.
– The propositional connectives and quantifiers are interpreted as in Kleene's calculus (cf. [KTB91]). For example, $\varphi \vee \varphi'$ is *true* if either φ or φ' is tt, is ff if both φ and φ' are ff and is *undef* otherwise.

For any $\varphi \in \mathbf{Sen_{PFOPEQ}}(\Theta)$ and $M \in |\mathbf{Mod_{3FOPEQ}}(\Theta)|$, this gives the *interpretation of φ in M*, $[\![\varphi]\!]_M \in \{tt, ff, undef\}$.
For each signature $\Theta \in |\mathbf{FOSig}|$, the satisfaction relation $\models_{\mathbf{3FOPEQ},\Theta} \subseteq |\mathbf{Mod_{3FOPEQ}}(\Theta)| \times \mathbf{Sen_{3FOPEQ}}(\Theta)$ is now defined in the obvious way: for any $M \in |\mathbf{Mod_{3FOPEQ}}(\Theta)|$ and $\varphi \in \mathbf{Sen_{3FOPEQ}}(\Theta)$:

– $M \models_{\mathbf{3FOPEQ},\Theta} \varphi$ **is** tt holds if and only if $[\![\varphi]\!]_M = tt$;
– $M \models_{\mathbf{3FOPEQ},\Theta} \varphi$ **is** ff holds if and only if $[\![\varphi]\!]_M = ff$; and
– $M \models_{\mathbf{3FOPEQ},\Theta} \varphi$ **is** *undef* holds if and only if $[\![\varphi]\!]_M = undef$.

Exercise. Work out all the details omitted from the above definition; notice that, in particular, model morphisms may be defined in a number of sensible ways. Then show that the satisfaction condition holds. □

Example 4.1.25 (The institution FPL of a logic for functional programs). The institution **FPL** of a logic for a simple functional programming language with a first-order monomorphic type system is defined as follows:

- A signature $\mathsf{SIG} = \langle S, \Omega, D \rangle$ consists of a set S of sort names, a family of sets of operation names $\Omega = \langle \Omega_{w,s} \rangle_{w \in S^*, s \in S}$, and a set D of *sorts with value constructors*. Elements of D have the form $\langle d, \mathscr{F} \rangle$ with $d \in S$ and $\mathscr{F} = \langle F_{w,d} \rangle_{w \in S^*}$, where $F_{w,d} \subseteq \Omega_{w,d}$ for $w \in S^*$, with no sort given more than one set of value constructors, i.e. $\langle d, \mathscr{F} \rangle, \langle d, \mathscr{F}' \rangle \in D$ implies $\mathscr{F} = \mathscr{F}'$. So SIG consists of an ordinary algebraic signature $\langle S, \Omega \rangle$ together with a set of *value constructors* for some of the sorts. Sorts with value constructors correspond to algebraic datatypes in functional programming languages. In examples we use a CASL-like notation[9], for instance:

  sort *nat* **free with** 0 **|** *succ*(*nat*)

  adds *nat* to S, $0{:}\,nat$ and $succ{:}\,nat \to nat$ to Ω, and $\langle nat, \{0{:}\,nat, succ{:}\,nat \to nat\} \rangle$ to D. We assume for convenience that each **FPL** signature SIG contains the sort *bool* with value constructors *true* and *false*:

  sort *bool* **free with** *true* **|** *false*

[9] CASL notation: this would be written **free type** *nat* ::= 0 **|** *succ*(*nat*) in CASL.

- A model over a signature $\mathsf{SIG} = \langle S, \Omega, D\rangle$ is a partial $\langle S, \Omega\rangle$-algebra A such that for each set[10] of sorts with value constructors $\{\langle d_1, \mathscr{F}_1\rangle, \ldots, \langle d_n, \mathscr{F}_n\rangle\} \subseteq D$, for $1 \leq i \leq n$, each value constructor in $\mathscr{F}_i$ is total and each element $a \in |A|_{d_i}$ is uniquely constructed from the values in $|A|$ of sorts other than $d_1, \ldots, d_n$ using the value constructors in $\mathscr{F}_1 \cup \cdots \cup \mathscr{F}_n$; that is, $\langle |A|_{d_i}\rangle_{1\leq i\leq n}$ is freely generated by $\mathscr{F}_1 \cup \cdots \cup \mathscr{F}_n$ from the carriers of the other sorts in A.
  We assume that all **FPL**-models interpret the sort *bool* and its constructors *true* and *false* in some standard way.
  A SIG-morphism between SIG-models A and B is an $\langle S, \Omega\rangle$-homomorphism between A and B viewed as partial $\langle S, \Omega\rangle$-algebras. It is *strong* if it is strong when viewed as a homomorphism between partial algebras, see Definition 2.7.31.
- The set $|T_{\mathsf{SIG}}(X)|$ of **FPL**-terms over $\mathsf{SIG} = \langle S, \Omega, D\rangle$ with variables X and their interpretation in an **FPL**-model A are defined by extending the usual definition of terms over $\langle S, \Omega\rangle$ and their interpretation by the following additional functional programming constructs (local recursive function definitions and pattern-matching case analysis, respectively):
  - **let fun** $f(x_1{:}s_1, \ldots, x_n{:}s_n){:}s' = t'$ **in** t is an **FPL**-term of sort s with variables in X if:
    - $s_1, \ldots, s_n, s' \in S$;
    - t' is an **FPL**-term of sort s' over SIG extended by $f{:}s_1 \times \cdots \times s_n \to s'$ with variables in $X \cup \{x_1{:}s_1, \ldots, x_n{:}s_n\}$; and
    - t is an **FPL**-term of sort s over SIG extended by $f{:}s_1 \times \cdots \times s_n \to s'$ with variables in X.

    The value of such a term under a valuation $v{:}X \to |A|$ is determined as follows:
    - extend A to give an algebra $\widehat{A}$ by interpreting $f{:}s_1 \times \cdots \times s_n \to s'$ as the least-defined partial function $f_{\widehat{A}}$ such that for all $a_1 \in |A|_{s_1}, \ldots, a_n \in |A|_{s_n}$, the value of $f_{\widehat{A}}(a_1, \ldots, a_n)$ is the same as the value of t' in $\widehat{A}$ under v modified by mapping x_1 to a_1 and $\ldots$ and x_n to a_n, whenever the latter is defined.[11]
    - the resulting value is then the value of t in $\widehat{A}$ under v.
  - **case** t **of** pat_1**=>**t_1 **|** $\cdots$ **|** pat_n**=>**t_n is an **FPL**-term of sort s with variables in X if:
    - t is an **FPL**-term of some sort s' over SIG with variables in X;
    - for each $1 \leq j \leq n$, pat_j is a *pattern* over SIG of sort s', where a pattern is an $\langle S, \Omega\rangle$-term containing only variables and value constructors, with no repeated variable occurrences; and

---

[10] This definition is complicated because of the possible presence of mutually dependent sorts with value constructors. **Exercise:** Check that imposing the same requirement for each sort with value constructors separately is more permissive and would not capture the intended meaning. Check also that it would be sufficient to consider only maximal sets of sorts with values constructors that are mutually dependent.

[11] The fact that this unambiguously defines $f_{\widehat{A}}$, and that $f_{\widehat{A}}$ can be equivalently given via the natural operational semantics of recursively-defined functions, is a standard result of denotational semantics, see for instance [Sch86].

- for each $1 \leq j \leq n$, t_j is an **FPL**-term of sort s with variables in the set X extended by the variables of pat_j.

The value of such a term under a valuation $v\colon X \to |A|$ is determined as follows:

- obtain the value a of t in A under v;
- find the least j such that *a matches pat_j* yielding a valuation v' of the variables in pat_j, where matching a value against a pattern proceeds as follows:
  - a variable x is matched by any value a, yielding a valuation $\{x \mapsto a\}$;
  - a pattern $f(p_1,\dots,p_m)$ is matched by a yielding v' iff[12] $a = f_A(a_1,\dots,a_m)$ and each p_i $(1 \leq i \leq m)$ is matched by a_i yielding v'_i, with $v' = v'_1 \cup \dots \cup v'_m$;
- the resulting value is that of t_j in A under the extension of v by v' if such a j exists; otherwise, the resulting value is undefined.

- Sentences over SIG are first-order sentences built over atomic formulae which are equalities between **FPL**-terms over SIG of the same sort and definedness assertions for such terms. Interpretation of **FPL**-terms in a model determines satisfaction of such sentences as in **PFOEQ**, see Exercises 4.1.17 and 4.1.18. (Recall that **PFOEQ** uses *strong* equality, see Section 2.7.4.)
  For convenience, we introduce *function definitions* of the form

  $$\textbf{fun } f(x_1{:}s_1,\dots,x_n{:}s_n){:}s = t$$

  to abbreviate the formula

  $$\forall x_1{:}s_1,\dots,x_n{:}s_n \bullet f(x_1,\dots,x_n) = \textbf{let fun } f(x_1{:}s_1,\dots,x_n{:}s_n){:}s = t \textbf{ in } f(x_1,\dots,x_n).$$

  To make the scopes of identifiers clearer, this can be rewritten using a new operation name g as

  $$\forall x_1{:}s_1,\dots,x_n{:}s_n \bullet f(x_1,\dots,x_n) = \textbf{let fun } g(x_1{:}s_1,\dots,x_n{:}s_n){:}s = t' \textbf{ in } g(x_1,\dots,x_n)$$

  where t' is the result of replacing f by g in t. Such a recursive function definition is different from the equality $f(x_1,\dots,x_n) = t$: for instance, $f(x_1,\dots,x_n) = f(x_1,\dots,x_n)$ always holds while $\textbf{fun } f(x_1{:}s_1,\dots,x_n{:}s_n){:}s = f(x_1,\dots,x_n)$ holds only when f is totally undefined.
- Given $\mathsf{SIG} = \langle S, \Omega, D\rangle$ and $\mathsf{SIG}' = \langle S', \Omega', D'\rangle$, an **FPL** signature morphism $\delta\colon \mathsf{SIG} \to \mathsf{SIG}'$ is a derived signature morphism $\delta\colon \langle S, \Omega\rangle \to \langle S', \Omega'\rangle$ (using **FPL**-terms in place of ordinary terms in Definition 1.5.13), such that for each $\langle d, \mathscr{F}\rangle \in D$, we have $\langle \delta(d), \mathscr{F}'\rangle \in D'$ such that δ restricted to $\mathscr{F}$ is determined by a bijection from $\mathscr{F}$ to $\mathscr{F}'$.
  We require all **FPL** signature morphisms to preserve the sort *bool* and its constructors *true* and *false*.
  Such signature morphisms go well beyond the usual renaming of sort and operation names; here we allow (non-constructor) operations to be mapped to

[12] This uniquely determines a result because non-variable patterns are of sorts that are freely generated by the value constructors and there are no repeated occurrences of variables in patterns.

complicated terms involving programming constructs like recursion and pattern-matching case analysis. This will be used in Chapters 6–9 to give examples, starting with Example 6.1.6, that suggest how programs fit into the overall specification and development framework.
Such a signature morphism determines a translation of SIG-sentences to SIG′-sentences in the usual manner,[13] and the same for the reduct from SIG′-models to SIG-models. Moreover, the satisfaction condition holds.

Exercise. Complete the above definition and prove the satisfaction condition. □

Exercise 4.1.26. The functional programming constructs used above are inspired by those in Standard ML [Pau96]. Add more constructs from Standard ML to the definition of **FPL**. Try adding type definitions, polymorphism, higher-order functions, exceptions.

It is easy to add built-in types other than *bool* by basing the definition of **FPL** on an arbitrary algebra DT as in **IMP** (Example 4.1.32 below). □

Exercise 4.1.27. Mutual recursion need not be added explicitly since it is already expressible using local definitions of recursive functions. Show how. HINT: It may be necessary to resort to copying function definitions, to make each function available for the definitions of the others. □

Exercise 4.1.28. Consider an **FPL**-signature SIG containing a sort s that is freely generated by value constructors from other such sorts. Show how an equality operation $eq_s\colon s \times s \to bool$ may be defined using a recursive function definition with pattern-matching case analysis. Use this to view conditionals of the form

$$\textbf{if } t_1 = t_2 \textbf{ then } t \textbf{ else } t'$$

(where t_1, t_2 are SIG-terms of sort s, and t, t' have the same sort) as an abbreviation for

$$\textbf{let fun } eq_s(x{:}s, y{:}s){:}bool = \ldots \textbf{ in case } eq_s(t_1, t_2) \textbf{ of } true\texttt{=>}t \,|\, false\texttt{=>}t'$$ □

Exercise 4.1.29. One could also introduce a conditional of the form **if** φ **then** t **else** t' where φ is a formula. Spell out the details. This would be unusual as a programming construct because branching is controlled by an arbitrary logical formula, allowing terms that would be problematic from a programming point of view, such as **if** $def(t)$ **then** t' **else** t'' and **if** $\forall x{:}s \bullet t_1 = t_2$ **then** t' **else** t''. Note that the meaning of such a conditional would be different from the one introduced in Exercise 4.1.28 when the check for equality involves a term with no defined value. □

---

[13] Care is required to avoid unintended clashes of **let**-bound operation names in SIG-terms with operation names in SIG′. To avoid consequent problems with functoriality of sentence translation, we can regard **FPL**-terms as being defined up to renaming of **let**-bound operation names.

Moreover, as in **FOPEQ** (see Example 4.1.12), care is needed with the treatment of bound variables (which now also include variables in patterns and formal parameters in **let**-bound operation definitions), cf. footnote 6.

Exercise 4.1.30. While **FPL** involves constructs borrowed from functional programming languages, it puts them in a logical context involving equality, logical connectives and quantifiers, which results in sentences capable not only of defining functions, but also of specifying their properties. Identify the "programming part" of **FPL** by defining its "subinstitution" **FProg** with the same signatures and models, but with sets of sentences restricted to function definitions (with satisfaction relations inherited from **FPL** as well). As function definitions may not be closed under translation along arbitrary (derived) signature morphisms in **FPL**, restrict the class of signature morphisms in **FProg** to the standard morphisms, where operation names are mapped to operation names rather than to arbitrary terms. □

Exercise 4.1.31. **FPL**, and its programming part **FProg**, relate to eager functional programming languages like Standard ML because partial functions are required to be strict. Formulate an analogous institution for lazy functional programming as in Haskell. □

The institutions **FPL** and **FProg** will be used in the sequel to present examples that are meant to appeal to the reader's programming intuition. Later on, the connection with functional programming will be further enhanced by introducing notations for defining ML-style modules in **FPL** (see Example 6.1.9 and Exercise 7.3.5 below).

Example 4.1.32 (The institution IMP of a simple imperative language). The institution **IMP** of an imperative programming language with simple type definitions is parameterised by an algebra DT on a signature Σ_{DT} of primitive (built-in) data types and functions of the language. The components of $\mathbf{IMP}_{DT}$ are defined as follows:

- A signature $\Pi = \langle T, P \rangle$ consists of a set T of type names and a set P of functional procedure names with types of the form $s_1, \ldots, s_n \to s$, where each of $s_1, \ldots, s_n, s$ is either a sort in Σ_{DT} or a type name in T. The names in T and P are distinct from those in Σ_{DT}. Thus $\Pi \cup \Sigma_{DT}$ is an algebraic signature — we will denote it by Π_{DT}. Signature morphisms map type names to type names and procedure names to procedure names preserving their types.
- There are two kinds of sentences over a signature $\Pi = \langle T, P \rangle$.
  First, sentences can be type definitions of the form

  type $s = \textit{type-expr}$

  where $s \in T$ is a type name and *type-expr* is a type expression in a simple language of types built over the sorts in Σ_{DT} and a unit type `unit` using the operators $+$ (disjoint union) and $\times$ (Cartesian product). The type expression *type-expr* may contain the type name s as well, which provides for recursive type definitions.[14]
  Second, sentences can be procedure definitions of the form

[14] Other type names from T are excluded, to prevent mutual recursion in type definitions — with some extra work this restriction can be removed.

proc $p(x_1{:}s_1,\ldots,x_n{:}s_n) = \textit{while-program};$ **result** $\textit{expr}{:}s$

where $p{:}s_1,\ldots,s_n \to s$ is a procedure name in P, *expr* is a Π_{DT}-term (with variables) of sort s, and *while-program* is a statement in a deterministic programming language over the built-in data types and functions given in DT (*while-program* may be empty, and so the program part of a procedure body may be omitted). We assume that the usual iterative program constructions are provided: sequential statements, conditionals and while loops. This requires that Σ_{DT} contains the sort *bool* with $|DT|_{bool} = \{tt, ff\}$. The basic statements are well-typed assignments (of expression values to formal parameters or variables scoped within each procedure body).

Expressions may use projections $\mathtt{proj}_1(v)$ and $\mathtt{proj}_2(v)$ for values v of product types of the form $s_1 \times s_2$, and pairing $\langle v_1, v_2\rangle$ to build values of product types, as well as boolean tests $\mathtt{is\text{-}in}_1(v)$ and $\mathtt{is\text{-}in}_2(v)$ for values v of union types of the form $s_1 + s_2$ and the constant $\langle\rangle$ of type $\mathtt{unit}$ denoting the only element of this type. The usual coercions between union types and their component types may also be used. With a bit of additional complication we can also allow expressions to contain (recursive) procedure calls.

- A model M over a signature $\Pi = \langle T, P\rangle$ has a carrier set $|M|_s$ for each $s \in T$. We write $|M|_s$ for $|DT|_s$ if s is a sort name in Σ_{DT}.

  We have the usual notion of *state*, where each state maps formal parameters and variables to values of their sorts in M, or marks them as undefined. An obvious operational semantics may be given that determines, for each statement and state, a sequence of states that formally captures the execution of that statement starting in that state.

  Then, M assigns to each procedure name $p{:}s_1,\ldots,s_n \to s$ in P and every sequence $v_1 \in |M|_{s_1},\ldots,v_n \in |M|_{s_n}$ of (actual parameter) values a formal execution which has one of the following forms:

  (*Successful termination*): a finite sequence of states and a value $v \in |M|_s$;
  (*Unsuccessful termination*): a finite sequence of states; or
  (*Divergence*): an infinite sequence of states.

  Given any such model M, for any procedure name $p{:}s_1,\ldots,s_n \to s$ in P we get a partial function $p_M{:}|M|_{s_1} \times \cdots \times |M|_{s_n} \to |M|_s$.

  The models defined in this way form a discrete category.
- For any signature $\Pi = \langle T, P\rangle$ and Π-model M:
  - M satisfies a Π-sentence of the form

    type $s = \textit{type-expr}$

    if $|M|_s$ is the least set $\mathbb{D}$ such that $\mathbb{D}$ is the value of the type expression *type-expr* in which the type name s is interpreted as $\mathbb{D}$ and sort names s' in Σ_{DT} are interpreted as $|DT|_{s'}$.
  - M satisfies a Π-sentence of the form

    proc $p(x_1{:}s_1,\ldots,x_n{:}s_n) = \textit{while-program};$ **result** $\textit{expr}{:}s$

if for all $v_1 \in |M|_{s_1}, \ldots, v_n \in |M|_{s_n}$, $M(p)(v_1,\ldots,v_n)$ is the formal execution of the statement *while-program* starting in the state $\{x_1 \mapsto v_1,\ldots,x_n \mapsto v_n\}$, and if the execution terminates successfully in a state in which *expr* has a defined value then $M(p)(v_1,\ldots,v_n)$ contains this value as well.

Exercise. Complete the above definition and prove the satisfaction condition. □

Exercise 4.1.33. Sentences in **IMP** are essentially programs; they provide no means of writing loose specifications. Add sentences of **PFOPEQ** for specifying properties of the procedures of **IMP** viewed as partial functions. A different way of achieving a similar effect will be presented in Examples 10.1.9, 10.1.14 and 10.1.17. □

Example 4.1.34 (The institution CDIAG of commutative diagrams). The following example is of a rather non-standard character. We present a simple logical system for stating that certain diagrams in a category with named objects and morphisms commute. Sentences of the logical system allow one to require that morphisms produced by composition of series of (named) morphisms coincide.

- The category of signatures in **CDIAG** is the category **Graph** of graphs (see Definition 3.2.36).
- A *path equation* in a graph G is a pair of paths in G with the same sources and targets, respectively. For any graph G (a signature in $\mathbf{Sign_{CDIAG}}$), G-sentences in **CDIAG** are sets of path equations in G.
- A model over a graph G is a (small) category $\mathbf{C}$ with a diagram D of "shape" G, i.e. (via Exercise 3.4.21) a functor $D\colon \mathbf{Path}(G) \to \mathbf{C}$. For any two G-models $D1\colon \mathbf{Path}(G) \to \mathbf{C1}$ and $D2\colon \mathbf{Path}(G) \to \mathbf{C2}$, a G-*morphism* in $\mathbf{Mod_{CDIAG}}(G)$ from $D1$ to $D2$ is a functor $\mathbf{F}\colon \mathbf{C1} \to \mathbf{C2}$ such that $D1;\mathbf{F} = D2$.
- For any G-model $D\colon \mathbf{Path}(G) \to \mathbf{C}$, a path p from s to t in G determines a morphism $D(p)\colon D(s) \to D(t)$ in $\mathbf{C}$. We say that a G-model $D\colon \mathbf{Path}(G) \to \mathbf{C}$ *satisfies* a path equation $\langle p,q\rangle$ if $D(p) = D(q)$. A G-model satisfies a G-sentence Φ if it satisfies all path equations $\varphi \in \Phi$.

Exercise. Complete the definition and prove the satisfaction condition for **CDIAG**.

Exercise. Reformulate the above definitions so that a sentence over a graph G would be a subdiagram of G used to denote the set of path equations in G which make the subdiagram commute. □

The last few examples show that the notion of institution covers much more than what one usually connects with the concept of a logical system.

The next two examples are perhaps even more unusual: we show that the definition of an institution does not restrict the sentences of a logic to be syntactic objects, and does not force models to provide semantic domains and operations used to determine the meanings of the syntactic objects. Thus, the notion of an institution covers systems in which such a distinction is entirely blurred.

Example 4.1.35. Consider an arbitrary category **Sign** and functor $\mathbf{Mod}\colon \mathbf{Sign}^{op} \to \mathbf{Cat}$. We think of **Sign** as a category of signatures and of **Mod** as yielding categories of models and reduct functors. To be cautious about foundations, we should make sure that **Mod** yields only small categories.

We can now define an institution $\mathbf{INS}^{\mathrm{Sen}(\mathbf{Mod})}$ where “sentences” are classes of models:

- The category of signatures of $\mathbf{INS}^{\mathrm{Sen}(\mathbf{Mod})}$ is **Sign**.
- The “sentence” functor of $\mathbf{INS}^{\mathrm{Sen}(\mathbf{Mod})}$ is defined as follows:
  - For any signature $\Sigma \in |\mathbf{Sign}|$, a Σ-“sentence” of $\mathbf{INS}^{\mathrm{Sen}(\mathbf{Mod})}$ is a collection $\mathscr{M} \subseteq |\mathbf{Mod}(\Sigma)|$ of Σ-models.
  - For any signature morphism $\sigma\colon \Sigma \to \Sigma'$, the σ-translation of any Σ-“sentence” $\mathscr{M} \subseteq |\mathbf{Mod}(\Sigma)|$ to a Σ'-“sentence” $\sigma(\mathscr{M}) \subseteq |\mathbf{Mod}(\Sigma')|$ is defined as the coimage of $\mathscr{M}$ w.r.t. the σ-reduct functor, i.e. $\sigma(\mathscr{M}) = \{M' \in |\mathbf{Mod}(\Sigma')| \mid \mathbf{Mod}(\sigma)(M') \in \mathscr{M}\}$.
- The model functor of $\mathbf{INS}^{\mathrm{Sen}(\mathbf{Mod})}$ is **Mod**.
- For each signature Σ, the Σ-satisfaction relation of $\mathbf{INS}^{\mathrm{Sen}(\mathbf{Mod})}$ is just the membership relation: for any Σ-model $M \in |\mathbf{Mod}(\Sigma)|$ and Σ-“sentence” $\mathscr{M} \subseteq |\mathbf{Mod}(\Sigma)|$, $M \models_{\mathbf{INS}^{\mathrm{Sen}(\mathbf{Mod})},\Sigma} \mathscr{M}$ if and only if $M \in \mathscr{M}$.

Exercise. Complete the definition and check the satisfaction condition. □

Example 4.1.36. Consider an arbitrary category **Sign** and functor $\mathbf{Sen}\colon \mathbf{Sign} \to \mathbf{Set}$. We think of **Sign** as a category of signatures and of **Sen** as yielding sets of sentences and their translations.

We can now define an institution $\mathbf{INS}^{\mathrm{Mod}(\mathbf{Sen})}$ where “models” are sets of sentences:

- The category of signatures of $\mathbf{INS}^{\mathrm{Mod}(\mathbf{Sen})}$ is **Sign**.
- The sentence functor of $\mathbf{INS}^{\mathrm{Mod}(\mathbf{Sen})}$ is **Sen**.
- The “model” functor of $\mathbf{INS}^{\mathrm{Mod}(\mathbf{Sen})}$ is defined as follows:
  - For any signature $\Sigma \in |\mathbf{Sign}|$, a Σ-“model” of $\mathbf{INS}^{\mathrm{Mod}(\mathbf{Sen})}$ is a set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences. The category of Σ-“models” is just the preorder category where the set of all such subsets is ordered by inclusion.
  - For any signature morphism $\sigma\colon \Sigma \to \Sigma'$, the σ-reduct functor of $\mathbf{INS}^{\mathrm{Mod}(\mathbf{Sen})}$ from the category of Σ'-“models” to the category of Σ-“models” maps any Σ'-“model” $\Phi' \subseteq \mathbf{Sen}(\Sigma')$ to its coimage $\{\varphi \in \mathbf{Sen}(\Sigma) \mid \mathbf{Sen}(\sigma)(\varphi) \in \Phi'\} \subseteq \mathbf{Sen}(\Sigma)$; this obviously extends to a functor between the preorder categories of Σ'- and Σ-“models”.
- For each signature Σ, the Σ-satisfaction relation of $\mathbf{INS}^{\mathrm{Mod}(\mathbf{Sen})}$ is (the inverse of) the membership relation: for any Σ-“model” $\Phi \subseteq \mathbf{Sen}(\Sigma)$ and Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$, $\Phi \models_{\mathbf{INS}^{\mathrm{Mod}(\mathbf{Sen})},\Sigma} \varphi$ if and only if $\varphi \in \Phi$.

Exercise. Complete the definition and check the satisfaction condition. □

Let us complete this list of examples by pointing out that the definition of institution admits a number of trivial situations:

Example 4.1.37 (Trivial institutions).

- Recall that $\mathbf{0}$ is the empty category. Hence, there is a unique (empty) functor from $\mathbf{0}$ to **Set** and a unique (empty) functor from $\mathbf{0}^{op} = \mathbf{0}$ to **Cat**. Together with the empty family of relations, they form an empty institution (no signatures, hence no sentences and no models).
- Given any category **Sign** and functor $\mathbf{Mod}\colon \mathbf{Sign}^{op} \to \mathbf{Cat}$, a trivial institution with signatures **Sign**, with models given by **Mod**, and with no sentences may be constructed. Formally, the sentences of this institution are given by the functor $\mathbf{Sen}_{\varnothing}\colon \mathbf{Sign} \to \mathbf{Set}$ which yields the empty set for each signature.
- Given any category **Sign** and functor $\mathbf{Sen}\colon \mathbf{Sign} \to \mathbf{Set}$, a trivial institution with signatures **Sign**, with sentences given by **Sen**, and with no models may be constructed. Formally, the models of this institution are given by the functor $\mathbf{Mod}_{\mathbf{0}}\colon \mathbf{Sign}^{op} \to \mathbf{Cat}$ which yields the empty category for each signature.
- Given any category **Sign** and functors $\mathbf{Sen}\colon \mathbf{Sign} \to \mathbf{Set}$ and $\mathbf{Mod}\colon \mathbf{Sign}^{op} \to \mathbf{Cat}$, two trivial institutions with signatures **Sign**, with sentences given by **Sen**, and with models given by **Mod** may be constructed. One is obtained by making all sentences false in all models, that is by defining each satisfaction relation to be empty. The other is obtained by making all sentences hold in all models, that is by definining each satisfaction relation to be total (i.e., for each $\Sigma \in |\mathbf{Sign}|$, $\models_{\Sigma} = |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$). □

4.1.2 *Constructing institutions*

In the examples of the previous subsection, each of the institutions was constructed "from scratch" by explicitly defining its signatures, sentences, models and satisfaction relations. This is often a rather tedious task (we have simplified it in many cases by referring to the standard definitions) and then checking the satisfaction condition is not always easy. In this subsection we will give some examples of constructions leading from an institution to a more complex one. The complexity added by the construction does not necessarily imply that the institution so obtained has any extra "expressive power". We start with some examples of "formal juggling" with institution components, very much in the spirit of Examples 4.1.35 and 4.1.36, and only then show how adding propositional connectives to a logic may be viewed as a construction of a new institution from an existing one.

Example 4.1.38. Sets of sentences of any institution may be regarded as single sentences (with the obvious "conjunctive" interpretation).

For any institution **INS** define the institution $\mathbf{INS}^{\wedge}$ of sets of **INS**-sentences as follows:

- The category of $\mathbf{INS}^{\wedge}$-signatures is the same as the category **Sign** of **INS**-signatures.
- The sentence functor $\mathbf{Sen}_{\mathbf{INS}^{\wedge}}$ is defined as follows:

  – For any signature $\Sigma \in |\mathbf{Sign}|$, $\mathbf{Sen}_{\mathbf{INS}^\wedge}(\Sigma)$ is the set of all sets $\Phi \subseteq \mathbf{Sen}_{\mathbf{INS}}(\Sigma)$ of Σ-sentences in **INS**.
  – For any signature morphism $\sigma : \Sigma \to \Sigma'$, the translation of a Σ-sentence Φ in $\mathbf{INS}^\wedge$ is its image w.r.t. the σ-translation function in **INS**: $\mathbf{Sen}_{\mathbf{INS}^\wedge}(\sigma)(\Phi) = \{\mathbf{Sen}_{\mathbf{INS}}(\sigma)(\varphi) \mid \varphi \in \Phi\} \subseteq \mathbf{Sen}_{\mathbf{INS}}(\Sigma')$.

- The model functor of $\mathbf{INS}^\wedge$ is the same as the model functor $\mathbf{Mod} : \mathbf{Sign}^{op} \to \mathbf{Cat}$ of **INS**.
- For any signature $\Sigma \in |\mathbf{Sign}|$, the satisfaction relation of $\mathbf{INS}^\wedge$ gives the conjunctive interpretation of (sets of) sentences: for any Σ-model $M \in |\mathbf{Mod}(\Sigma)|$ and Σ-sentence $\Phi \subseteq \mathbf{Sen}_{\mathbf{INS}}(\Sigma)$, $M \models_{\mathbf{INS}^\wedge,\Sigma} \Phi$ if and only if for all $\varphi \in \Phi$, $M \models_{\mathbf{INS},\Sigma} \varphi$. □

Example 4.1.39. Signatures of any institution may be enriched to incorporate sentences which restrict the class of models considered over the given signature.

For any institution **INS** define the institution $\mathbf{INS}^{\mathbf{Sign}^+}$ with signatures enriched by sentences as follows:

- Signatures of $\mathbf{INS}^{\mathbf{Sign}^+}$ are pairs $\langle \Sigma, \Phi \rangle$, where $\Sigma \in |\mathbf{Sign}_{\mathbf{INS}}|$ is an **INS**-signature and $\Phi \subseteq \mathbf{Sen}_{\mathbf{INS}}(\Sigma)$ is a set of Σ-sentences. Then, an $\mathbf{INS}^{\mathbf{Sign}^+}$-signature morphism $\sigma : \langle \Sigma, \Phi \rangle \to \langle \Sigma', \Phi' \rangle$ is a signature morphism $\sigma : \Sigma \to \Sigma'$ in $\mathbf{Sign}_{\mathbf{INS}}$ such that for all $\varphi \in \Phi$, $\sigma(\varphi) \in \Phi'$. This defines a category $\mathbf{Sign}_{\mathbf{INS}^{\mathbf{Sign}^+}}$ of $\mathbf{INS}^{\mathbf{Sign}^+}$-signatures (with composition inherited from $\mathbf{Sign}_{\mathbf{INS}}$).
- Sentences of $\mathbf{INS}^{\mathbf{Sign}^+}$ are the same as **INS**-sentences: for any $\mathbf{INS}^{\mathbf{Sign}^+}$-signature $\langle \Sigma, \Phi \rangle$, $\mathbf{Sen}_{\mathbf{INS}^{\mathbf{Sign}^+}}(\langle \Sigma, \Phi \rangle) = \mathbf{Sen}_{\mathbf{INS}}(\Sigma)$, with the translation functions inherited from **INS** as well.
- Models of $\mathbf{INS}^{\mathbf{Sign}^+}$ are again the same as models of **INS**; we consider, however, only those models that satisfy the sentences in the given signature. For any $\mathbf{INS}^{\mathbf{Sign}^+}$-signature $\langle \Sigma, \Phi \rangle$, $\mathbf{Mod}_{\mathbf{INS}^{\mathbf{Sign}^+}}(\langle \Sigma, \Phi \rangle)$ is the full subcategory of $\mathbf{Mod}_{\mathbf{INS}}(\Sigma)$ consisting of all Σ-models (in **INS**) that satisfy (according to $\models_{\mathbf{INS},\Sigma}$) all the sentences in Φ. The reduct functors are again inherited from **INS**.
- The satisfaction relations of $\mathbf{INS}^{\mathbf{Sign}^+}$ are inherited from **INS**.

Exercise. Spell out all the details of the above definition. In particular, check that the reduct functors of the new institution $\mathbf{INS}^{\mathbf{Sign}^+}$ are well-defined (cf. Fact 4.2.24 below). □

Example 4.1.40. For any institution, we can enlarge its categories of models by considering models over extended signatures.

For any institution **INS**, define the institution $\mathbf{INS}^{\mathbf{Mod}^+}$ with categories of models containing models over extended signatures as follows:

- The category of $\mathbf{INS}^{\mathbf{Mod}^+}$-signatures is the category **Sign** of **INS**-signatures.
- The sentence functor of $\mathbf{INS}^{\mathbf{Mod}^+}$ is the sentence functor $\mathbf{Sen} : \mathbf{Sign} \to \mathbf{Set}$ of **INS**.
- The model functor $\mathbf{Mod}_{\mathbf{INS}^{\mathbf{Mod}^+}} : \mathbf{Sign}^{op} \to \mathbf{Cat}$ is defined as follows:

– For any signature $\Sigma \in |\mathbf{Sign}|$, a Σ-model of $\mathbf{INS}^{\mathbf{Mod}^+}$ is an **INS**-model over an extension of the signature Σ. Formally: a Σ-model in $\mathbf{INS}^{\mathbf{Mod}^+}$ is a pair $\langle \sigma{:}\Sigma \to \Sigma', M' \in |\mathbf{Mod}_{\mathbf{INS}}(\Sigma')|\rangle$. A Σ-model morphism between two such Σ-models is again a pair $\langle \sigma', f\rangle{:}\langle \sigma_1{:}\Sigma \to \Sigma'_1, M'_1 \in |\mathbf{Mod}_{\mathbf{INS}}(\Sigma'_1)|\rangle \to \langle \sigma_2{:}\Sigma \to \Sigma'_2, M'_2 \in |\mathbf{Mod}_{\mathbf{INS}}(\Sigma'_2)|\rangle$, which consists of an **INS**-signature morphism $\sigma'{:}\Sigma'_1 \to \Sigma'_2$ such that $\sigma_1;\sigma' = \sigma_2$ and a model morphism $f{:}M'_1 \to \mathbf{Mod}_{\mathbf{INS}}(\sigma')(M'_2)$ in $\mathbf{Mod}_{\mathbf{INS}}(\Sigma'_1)$.
– For any signature morphism $\sigma{:}\Sigma_1 \to \Sigma_2$, the σ-reduct functor $\mathbf{Mod}_{\mathbf{INS}^{\mathbf{Mod}^+}}(\sigma)$ maps any Σ_2-model $\langle \sigma_2{:}\Sigma_2 \to \Sigma'_2, M'_2 \in |\mathbf{Mod}_{\mathbf{INS}}(\Sigma'_2)|\rangle$ to the Σ_1-model $\langle \sigma;\sigma_2{:}\Sigma_1 \to \Sigma'_2, M'_2 \in |\mathbf{Mod}_{\mathbf{INS}}(\Sigma'_2)|\rangle$. On model morphisms, $\mathbf{Mod}_{\mathbf{INS}^{\mathbf{Mod}^+}}(\sigma)$ is the identity.

- For each signature $\Sigma \in |\mathbf{Sign}|$, the Σ-satisfaction relation of $\mathbf{INS}^{\mathbf{Mod}^+}$ is determined by the Σ-satisfaction relation of **INS**: for any Σ-model $\langle \sigma{:}\Sigma \to \Sigma', M' \in |\mathbf{Mod}_{\mathbf{INS}}(\Sigma')|\rangle$ and Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$, $\langle \sigma, M'\rangle \models_{\mathbf{INS}^{\mathbf{Mod}^+},\Sigma} \varphi$ if and only if $M' \models_{\mathbf{INS},\Sigma'} \mathbf{Sen}(\sigma)(\varphi)$, which by the satisfaction condition for **INS** is equivalent to $\mathbf{Mod}_{\mathbf{INS}}(\sigma)(M') \models_{\mathbf{INS},\Sigma} \varphi$.

Exercise. Complete the definition and check the satisfaction condition. Try to express the construction of the categories of models of $\mathbf{INS}^{\mathbf{Mod}^+}$ using the flattening construction for indexed categories (Definition 3.4.58) and the machinery of comma categories (Definition 3.4.49). □

Example 4.1.41. For any institution **INS** define the institution $\mathbf{INS}^{prop}$ by closing the sets of its sentences under propositional connectives (with the usual interpretation) as follows:

- The category of signatures of $\mathbf{INS}^{prop}$ is just the category **Sign** of **INS**-signatures.
- The sentence functor $\mathbf{Sen}_{\mathbf{INS}^{prop}}{:}\mathbf{Sign} \to \mathbf{Set}$ is defined as follows:

– For any signature $\Sigma \in |\mathbf{Sign}|$, $\mathbf{Sen}_{\mathbf{INS}^{prop}}(\Sigma)$ is the least set that contains all of the Σ-sentences of **INS** and two special sentences true and false, and is closed under the usual propositional connectives as introduced in Example 4.1.9, that is, if $\varphi, \varphi' \in \mathbf{Sen}_{\mathbf{INS}^{prop}}(\Sigma)$ then also $\varphi \vee \varphi' \in \mathbf{Sen}_{\mathbf{INS}^{prop}}(\Sigma)$, $\neg\varphi \in \mathbf{Sen}_{\mathbf{INS}^{prop}}(\Sigma)$, $\varphi \wedge \varphi' \in \mathbf{Sen}_{\mathbf{INS}^{prop}}(\Sigma)$, and $\varphi \Rightarrow \varphi' \in \mathbf{Sen}_{\mathbf{INS}^{prop}}(\Sigma)$.[15]
– For any signature morphism $\sigma{:}\Sigma \to \Sigma'$, the σ-translation function $\mathbf{Sen}_{\mathbf{INS}^{prop}}(\sigma)$ coincides with $\mathbf{Sen}_{\mathbf{INS}}(\sigma)$ on $\mathbf{Sen}_{\mathbf{INS}}(\Sigma)$ and preserves the propositional connectives in the new sentences in the obvious way.

- The model functor of $\mathbf{INS}^{prop}$ is the model functor $\mathbf{Mod}{:}\mathbf{Sign}^{op} \to \mathbf{Cat}$ of **INS**.
- For each signature $\Sigma \in |\mathbf{Sign}|$, the Σ-satisfaction relation of $\mathbf{INS}^{prop}$ is just the same as the Σ-satisfaction relation of **INS** for sentences in $\mathbf{Sen}_{\mathbf{INS}}(\Sigma)$ and then, for any Σ-model $M \in |\mathbf{Mod}(\Sigma)|$, for the sentences built using the propositional connectives, the satisfaction of such sentences in M is defined inductively as in Example 4.1.9.

[15] The remarks in footnote 4 apply as appropriate.

Exercise. Show how **PROP**, the institution of propositional logic (see Example 4.1.9) arises as the propositional closure of a simple institution with propositional variables as the only sentences. □

In Section 4.4.2 below we define yet another similar construction on institutions by showing how quantifiers may be introduced.

The constructions described in the examples above may naturally be viewed as extensions of the original institution — this will be made formal in Section 10.2, cf. Example 10.2.5. In Section 10.3 we will discuss how such extensions may be combined using the limit construction in a suitable category of institutions.

These examples are about adding new sentences built using logical connectives to an institution. The new sentences are added, even if the connectives were already expressible in the following sense:

Definition 4.1.42. The institution **INS** *has negation* if for every signature $\Sigma \in |\mathbf{Sign}|$ and Σ-sentence φ, there exists a Σ-sentence ψ such that for every Σ-model M, $M \models_\Sigma \varphi$ iff $M \not\models_\Sigma \psi$. Any such ψ may be referred to as $\neg\varphi$.

The properties of *having conjunction*, *having disjunction* and *having implication* are defined in the analogous way, and similarly for *having truth*, *having falsity*, *having infinitary conjunction* etc. □

Exercise 4.1.43. Suppose that the institution **INS** has negation. Using the satisfaction condition, show that for every signature morphism $\sigma: \Sigma \to \Sigma'$ and Σ-sentence φ, $\neg\sigma(\varphi)$ may be taken to be $\sigma(\neg\varphi)$. Show a similar property for the other connectives. □

Example 4.1.44. For any institutions $\mathbf{INS}_1 = \langle \mathbf{Sign}_1, \mathbf{Sen}_1, \mathbf{Mod}_1, \langle \models_{1,\Sigma_1} \rangle_{\Sigma_1 \in |\mathbf{Sign}_1|} \rangle$ and $\mathbf{INS}_2 = \langle \mathbf{Sign}_2, \mathbf{Sen}_2, \mathbf{Mod}_2, \langle \models_{2,\Sigma_2} \rangle_{\Sigma_2 \in |\mathbf{Sign}_2|} \rangle$, their *sum* $\mathbf{INS}_1 + \mathbf{INS}_2$ puts $\mathbf{INS}_1$ and $\mathbf{INS}_2$ side by side without any "interaction". Formally, $\mathbf{INS}_1 + \mathbf{INS}_2$ is defined as follows:

- The category of signatures of $\mathbf{INS}_1 + \mathbf{INS}_2$ is the disjoint union $\mathbf{Sign}_1 + \mathbf{Sign}_2$ of the categories of signatures of $\mathbf{INS}_1$ and of $\mathbf{INS}_2$.
- The sentence functor $\mathbf{Sen}_{\mathbf{INS}_1+\mathbf{INS}_2}: \mathbf{Sign}_1 + \mathbf{Sign}_2 \to \mathbf{Set}$ acts as $\mathbf{Sen}_1$ on $\mathbf{Sign}_1$ and as $\mathbf{Sen}_2$ on $\mathbf{Sign}_2$ (that is, $\mathbf{Sen}_{\mathbf{INS}_1+\mathbf{INS}_2}$ is determined by $\mathbf{Sen}_1$ and $\mathbf{Sen}_2$ according to the coproduct property of $\mathbf{Sign}_1 + \mathbf{Sign}_2$).
- The model functor $\mathbf{Mod}_{\mathbf{INS}_1+\mathbf{INS}_2}: (\mathbf{Sign}_1 + \mathbf{Sign}_2)^{op} \to \mathbf{Cat}$ acts as $\mathbf{Mod}_1$ on $\mathbf{Sign}_1$ and as $\mathbf{Mod}_2$ on $\mathbf{Sign}_2$ (that is, $\mathbf{Mod}_{\mathbf{INS}_1+\mathbf{INS}_2}$ is determined by $\mathbf{Mod}_1$ and $\mathbf{Mod}_2$ according to the coproduct property of $\mathbf{Sign}_1 + \mathbf{Sign}_2$).
- The family of satisfaction relations of $\mathbf{INS}_1 + \mathbf{INS}_2$ is the union of the families of satisfaction relations of $\mathbf{INS}_1$ and of $\mathbf{INS}_2$ (that is, for $\Sigma_1 \in |\mathbf{Sign}_1|$, $\models_{\mathbf{INS}_1+\mathbf{INS}_2,\Sigma_1}$ is $\models_{1,\Sigma_1}$, and for $\Sigma_2 \in |\mathbf{Sign}_2|$, $\models_{\mathbf{INS}_1+\mathbf{INS}_2,\Sigma_2}$ is $\models_{2,\Sigma_2}$). □

Example 4.1.45. Given institutions $\mathbf{INS}_1 = \langle \mathbf{Sign}_1, \mathbf{Sen}_1, \mathbf{Mod}_1, \langle \models_{1,\Sigma_1} \rangle_{\Sigma_1 \in |\mathbf{Sign}_1|} \rangle$ and $\mathbf{INS}_2 = \langle \mathbf{Sign}_2, \mathbf{Sen}_2, \mathbf{Mod}_2, \langle \models_{2,\Sigma_2} \rangle_{\Sigma_2 \in |\mathbf{Sign}_2|} \rangle$, their *product* $\mathbf{INS}_1 \times \mathbf{INS}_2$ is defined as follows:

- The category of signatures of $\mathbf{INS}_1 \times \mathbf{INS}_2$ is the product $\mathbf{Sign_1} \times \mathbf{Sign_2}$ of the categories of signatures of $\mathbf{INS}_1$ and of $\mathbf{INS}_2$; thus a signature in $\mathbf{INS}_1 \times \mathbf{INS}_2$ is a pair consisting of one signature from $\mathbf{INS}_1$ and one from $\mathbf{INS}_2$, and similarly for signature morphisms.
- The sentence functor $\mathbf{Sen}_{\mathbf{INS}_1 \times \mathbf{INS}_2} \colon \mathbf{Sign_1} \times \mathbf{Sign_2} \to \mathbf{Set}$ is defined as follows:
  - For any signature $\langle \Sigma_1, \Sigma_2 \rangle \in |\mathbf{Sign}_1 \times \mathbf{Sign}_2|$, $\mathbf{Sen}_{\mathbf{INS}_1 \times \mathbf{INS}_2}(\langle \Sigma_1, \Sigma_2 \rangle) = \mathbf{Sen}_1(\Sigma_1) + \mathbf{Sen}_2(\Sigma_2)$ is the disjoint union of the sets of $\mathbf{INS}_1$-sentences over Σ_1 and of $\mathbf{INS}_2$-sentences over Σ_2.
  - For any signature morphism $\langle \sigma_1, \sigma_2 \rangle \colon \langle \Sigma_1, \Sigma_2 \rangle \to \langle \Sigma'_1, \Sigma'_2 \rangle$, $\mathbf{Sen}_{\mathbf{INS}_1 \times \mathbf{INS}_2}(\langle \sigma_1, \sigma_2 \rangle) = \mathbf{Sen}_1(\sigma_1) + \mathbf{Sen}_2(\sigma_2)$ acts as $\mathbf{Sen}_1(\sigma_1)$ on $\mathbf{INS}_1$-sentences and as $\mathbf{Sen}_2(\sigma_2)$ on $\mathbf{INS}_2$-sentences over the signature $\langle \Sigma_1, \Sigma_2 \rangle$.
- The model functor $\mathbf{Mod}_{\mathbf{INS}_1 \times \mathbf{INS}_2} \colon (\mathbf{Sign_1} \times \mathbf{Sign_2})^{op} \to \mathbf{Cat}$ is defined as follows:
  - For any signature $\langle \Sigma_1, \Sigma_2 \rangle \in |\mathbf{Sign}_1 \times \mathbf{Sign}_2|$, $\mathbf{Mod}_{\mathbf{INS}_1 \times \mathbf{INS}_2}(\langle \Sigma_1, \Sigma_2 \rangle) = \mathbf{Mod}_1(\Sigma_1) \times \mathbf{Mod}_2(\Sigma_2)$ is the product of the categories of $\mathbf{INS}_1$-models over Σ_1 and of $\mathbf{INS}_2$-models over Σ_2; thus a model in $\mathbf{INS}_1 \times \mathbf{INS}_2$ is a pair consisting of one model from $\mathbf{INS}_1$ and one from $\mathbf{INS}_2$, and similarly for model morphisms.
  - For any signature morphism $\langle \sigma_1, \sigma_2 \rangle \colon \langle \Sigma_1, \Sigma_2 \rangle \to \langle \Sigma'_1, \Sigma'_2 \rangle$, $\mathbf{Mod}_{\mathbf{INS}_1 \times \mathbf{INS}_2}(\langle \sigma_1, \sigma_2 \rangle) = \mathbf{Mod}_1(\sigma_1) \times \mathbf{Mod}_2(\sigma_2)$ acts as $\mathbf{Mod}_1(\sigma_1)$ on the $\mathbf{INS}_1$-components of $\langle \Sigma'_1, \Sigma'_2 \rangle$-models and model morphisms and as $\mathbf{Mod}_2(\sigma_2)$ on the $\mathbf{INS}_2$-components of $\langle \Sigma'_1, \Sigma'_2 \rangle$-models and model morphisms.
- For any signature $\langle \Sigma_1, \Sigma_2 \rangle \in |\mathbf{Sign_1} \times \mathbf{Sign_2}|$, model $\langle M_1, M_2 \rangle \in |\mathbf{Mod}_{\mathbf{INS}_1 \times \mathbf{INS}_2}(\langle \Sigma_1, \Sigma_2 \rangle)|$ and sentences $\varphi_1 \in \mathbf{Sen}_1(\Sigma_1)$ and $\varphi_2 \in \mathbf{Sen}_2(\Sigma_2)$, $\langle M_1, M_2 \rangle \models_{\mathbf{INS}_1 \times \mathbf{INS}_2, \langle \Sigma_1, \Sigma_2 \rangle} \varphi_1$ if and only if $M_1 \models_{1,\Sigma_1} \varphi_1$, and $\langle M_1, M_2 \rangle \models_{\mathbf{INS}_1 \times \mathbf{INS}_2, \langle \Sigma_1, \Sigma_2 \rangle} \varphi_2$ if and only if $M_2 \models_{2,\Sigma_2} \varphi_2$. That is, satisfaction in $\mathbf{INS}_1 \times \mathbf{INS}_2$ is defined as $\mathbf{INS}_1$-satisfaction for $\mathbf{INS}_1$-sentences (extracting the $\mathbf{INS}_1$-components of $\mathbf{INS}_1 \times \mathbf{INS}_2$-models) and as $\mathbf{INS}_2$-satisfaction for $\mathbf{INS}_2$-sentences (extracting the $\mathbf{INS}_2$-components of $\mathbf{INS}_1 \times \mathbf{INS}_2$-models). □

The next example indicates a technically correct but intuitively somewhat artificial way of dealing with the translation of sentences along signature morphisms. The simple idea is that instead of actually translating sentences from one signature to another, we can always keep the original sentence over its original signature together with a morphism "fitting" it to another signature.

Example 4.1.46. Consider an institution $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ and a function $NewSen \colon |\mathbf{Sign}| \to |\mathbf{Set}|$ together with a family of relations $\langle \models_{NewSen,\Sigma} \subseteq |\mathbf{Mod}(\Sigma)| \times NewSen(\Sigma) \rangle_{\Sigma \in |\mathbf{Sign}|}$. Intuitively, for any signature Σ, $NewSen(\Sigma)$ is a set of new sentences over Σ with the satisfaction relation $\models_{NewSen,\Sigma}$. We define an institution $\mathbf{INS} + NewSen$ by adding these new sentences fitted to other signatures via signature morphisms:

- The category of signatures of $\mathbf{INS} + NewSen$ is just the category $\mathbf{Sign}$ of $\mathbf{INS}$-signatures.
- The sentence functor $\mathbf{Sen}_{\mathbf{INS}+NewSen} \colon \mathbf{Sign} \to \mathbf{Set}$ is defined as follows:

– For any signature $\Sigma \in |\mathbf{Sign}|$, $\mathbf{Sen}_{\mathbf{INS}+\mathit{NewSen}}(\Sigma)$ is the (disjoint) union of the "old" sentences $\mathbf{Sen}(\Sigma)$ and the set[16] of "new" sentences fitted to the signature Σ by a signature morphism. The latter are pairs $\langle \varphi', \theta \rangle$, written as φ' **through** θ, with $\theta : \Sigma' \to \Sigma$ and $\varphi' \in \mathit{NewSen}(\Sigma')$ for an arbitrary signature Σ'.
– For any signature morphism $\sigma : \Sigma \to \Sigma_1$, $\mathbf{Sen}_{\mathbf{INS}+\mathit{NewSen}}(\sigma)$ works as $\mathbf{Sen}(\sigma)$ on the **INS**-sentences; for $\theta : \Sigma' \to \Sigma$ and $\varphi' \in \mathit{NewSen}(\Sigma')$, $\mathbf{Sen}_{\mathbf{INS}+\mathit{NewSen}}(\sigma)(\varphi'$ **through** $\theta) = \varphi'$ **through** $\theta;\sigma$.

- The model functor of $\mathbf{INS} + \mathit{NewSen}$ is the model functor $\mathbf{Mod} : \mathbf{Sign}^{op} \to \mathbf{Cat}$ of **INS**.
- For each signature $\Sigma \in |\mathbf{Sign}|$, the Σ-satisfaction relation of $\mathbf{INS} + \mathit{NewSen}$ is just the same as the Σ-satisfaction relation of **INS** for the "old" Σ-sentences and then, for any Σ-model $M \in |\mathbf{Mod}(\Sigma)|$, $\theta : \Sigma' \to \Sigma$ and $\varphi' \in \mathit{NewSen}(\Sigma')$, $M \models_{\mathbf{INS}+\mathit{NewSen}} \varphi'$ **through** θ if and only if $M|_\theta \models_{\mathit{NewSen},\Sigma'} \varphi'$.

Exercise. Check the satisfaction condition. □

We conclude this list of constructions on institutions with a sketch of how various modal (and temporal) logics may be built over an arbitrary institution.

Example 4.1.47. Let $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ be an institution. We define the institution $\mathbf{LTL_{INS}}$ of linear-time temporal logic over **INS**, using sequences of models from **INS** as models and sentences from **INS** as "state sentences", that is:

- The category of signatures of $\mathbf{LTL_{INS}}$ is **Sign**, the same as in **INS**.
- For each signature Σ, a Σ-model in $\mathbf{LTL_{INS}}$ is a countably infinite sequence $\overline{M} = \langle M_n \rangle_{n \geq 0}$ of models $M_n \in |\mathbf{Mod}(\Sigma)|$ for $n \geq 0$. Reducts of such models w.r.t. a signature morphism σ are defined componentwise, using the reduct w.r.t. σ as defined in **INS**. (We disregard model morphisms here, taking $\mathbf{Mod}_{\mathbf{LTL_{INS}}}(\Sigma)$ to be the discrete category.)
- For each signature Σ, the set of Σ-sentences in $\mathbf{LTL_{INS}}$ is the least set that contains `true` and all the sentences in $\mathbf{Sen}(\Sigma)$ (called *state sentences* in this context) and is closed under negation, written $\neg\varphi$, conjunction, $\varphi \wedge \psi$, and two modal operators: *next time*, $\mathsf{X}\varphi$, and *until*, $\varphi \,\mathsf{U}\, \psi$.
- For each signature Σ, satisfaction is defined in terms of an auxiliary relation of satisfaction at a given position in the temporal sequence; for each model $\overline{M} = \langle M_n \rangle_{n \geq 0}$, and $j \geq 0$ we define:

  – for any state sentence φ, $\overline{M} \models^j \varphi$ if $M_j \models \varphi$ (in **INS**);
  – $\overline{M} \models^j \neg\varphi$ if it is not the case that $\overline{M} \models^j \varphi$;
  – $\overline{M} \models^j \varphi \wedge \psi$ if $\overline{M} \models^j \varphi$ and $\overline{M} \models^j \psi$;

[16] This may lead to some foundational difficulties, since the collection of all signature morphisms into Σ, and hence the collection of all new Σ-sentences, need not form a set. One argument for ignoring these problems here is that we can typically limit the size of the category of signatures of the institution we start with, for example assuming that the category **Sign** is small.

– $\overline{M} \models^j \mathsf{X}\varphi$ if $\overline{M} \models^{j+1} \varphi$; and
– $\overline{M} \models^j \varphi \,\mathsf{U}\, \psi$ if for some $k \geq j$, $\overline{M} \models^k \psi$ and for all $j \leq i < k$, $\overline{M} \models^i \varphi$.

We put now $\overline{M} \models_{\mathbf{LTL}_{\mathbf{INS}},\Sigma} \varphi$ if $\overline{M} \models^0 \varphi$.

Exercise. Complete the definition and check the satisfaction condition.

Exercise. Add other temporal modalities, like "eventually/finally" and "henceforth/globally", either by defining them explicitly, or as abbreviations, for instance: $\mathsf{F}\varphi \equiv \mathsf{true}\,\mathsf{U}\,\varphi$, $\mathsf{G}\varphi \equiv \neg(\mathsf{F}(\neg\varphi))$, etc.

Also, add "past" temporal modalities (previous, since, sometimes in the past, always in the past, etc). □

Exercise 4.1.48. Proceeding similarly as in Example 4.1.47, given an institution **INS**, define the institution $\mathbf{MDL}_{\mathbf{INS}}$ of modal logic:

- The category of signatures of $\mathbf{MDL}_{\mathbf{INS}}$ is **Sign**, the same as in **INS**.
- For each signature Σ, a Σ-model in $\mathbf{MDL}_{\mathbf{INS}}$ is a Kripke structure, i.e., a triple $\langle W, \rightsquigarrow, \overline{M}\rangle$, which consists of a set W (of "possible worlds" or "state names") and a relation $\rightsquigarrow \subseteq W \times W$ ("transition relation") together with a family $\overline{M} = \langle M_w \rangle_{w \in W}$ of Σ-models in **INS**, $M_w \in |\mathbf{Mod}(\Sigma)|$ for $w \in W$. Again, we disregard model morphisms.
- For each signature Σ, the set of Σ-sentences in $\mathbf{MDL}_{\mathbf{INS}}$ is the least set that contains true and all the sentences in $\mathbf{Sen}(\Sigma)$ and is closed under negation $\neg\varphi$, conjunction $\varphi \wedge \psi$, and the modal operator $\Box\varphi$.
- For each signature Σ, satisfaction is defined in terms of an auxiliary relation of satisfaction at a given world in a Kripke structure; here is the crucial clause:

  – $\langle W, \rightsquigarrow, \overline{M}\rangle \models^w \Box\varphi$ if for all $v \in W$ such that $w \rightsquigarrow v$, $\langle W, \rightsquigarrow, \overline{M}\rangle \models^v \varphi$.

  Then a model satisfies a sentence in $\mathbf{MDL}_{\mathbf{INS}}$ if the sentence holds in the above sense at each of its possible worlds.

Complete the definition and check the satisfaction condition.

To keep the definition closer to $\mathbf{LTL}_{\mathbf{INS}}$, you may want to define a somewhat different version of modal logic, where Kripke structures in addition indicate an initial world, and then the satisfaction of a sentence in a model is determined by its satisfaction at this initial world. You may also want to impose requirements on the transition relation (for instance, that it is transitive, or that all possible worlds can be reached from the initial world, etc.).

Combining the ideas behind $\mathbf{MDL}_{\mathbf{INS}}$ and $\mathbf{LTL}_{\mathbf{INS}}$, define the institution $\mathbf{CTL}^*_{\mathbf{INS}}$ of branching-time temporal logic, where signatures and models are as in $\mathbf{MDL}_{\mathbf{INS}}$, but sentences are closed under a variety of temporal operators used to quantify (separately) over paths in the Kripke structure and over worlds in these paths. HINT: Distinguish two kinds of sentences: path sentences that are evaluated for a given path in the Kripke structure; and state sentences that are evaluated for a given world in the Kripke structure — or see [Eme90].

You may also start by defining a simpler institution $\mathbf{CTL}_{\mathbf{INS}}$ where the use of temporal operators is limited by requiring that quantification over paths and over worlds in these paths always happen together, so in fact we have only bundled path/state temporal operators, like “for some path, always in this path”, “for some path, eventually in this path”, etc. □

Exercise 4.1.49. Consider an institution $\mathbf{MDL_{FOPEQ}}$ of modal logic built over first-order predicate logic with equality. Note that this is *not* the institution of first-order modal logic, since quantification is internal to state sentences only and cannot be interleaved with the modal operator. Define an institution of first-order modal logic in which such an arbitrary interleaving of quantifiers, propositional connectives and the modal operator is allowed. HINT: The trouble here is with moving valuations of variables from one world to another in the definition of satisfaction. At least, define such an institution assuming that the carriers of all models in any Kripke structure coincide. Discuss possible generalisations.

Carry out similar constructions of first-order temporal logics that extend $\mathbf{LTL_{FOPEQ}}$, $\mathbf{CTL^{*}_{FOPEQ}}$ and $\mathbf{CTL_{FOPEQ}}$, respectively. □

4.2 Flat specifications in an arbitrary institution

Throughout this section we will deal with an arbitrary but fixed institution. This means that we will be working with a logical system about which we know nothing beyond what is given in the definition of an institution. For example, we will not be able to refer to any particular components of signatures, any particular syntax of sentences, any particular internal structure of models, or any particular definition of satisfaction. Indeed, we cannot even be sure that signatures have components, that sentences are syntactic entities in any sense, or that models have any internal structure at all.

Given these limitations, one may think that there is very little that can be done. However, the structure of an institution is rich enough to allow us to recast in these terms the material on simple equational specifications presented in Sections 2.2 and 2.3 (this will be done in the present section, without repeating the discussion and motivation) and then to proceed further into the theory of specifications and software development.

Let us then fix an arbitrary institution $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$. We start with the basic concepts built around the notion of satisfaction.

Definition 4.2.1 ($Mod_\Sigma(\Phi)$, $Th_\Sigma(\mathscr{M})$, $Cl_\Sigma(\Phi)$ and $Cl_\Sigma(\mathscr{M})$). Let Σ be an arbitrary signature.

- For any set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences, the class $Mod_\Sigma(\Phi) \subseteq |\mathbf{Mod}(\Sigma)|$ of *models of* Φ is defined as the class of all Σ-models that satisfy all the sentences in Φ.[17]

[17] Note the overloading of the term “model” as discussed after Definition 4.1.1. We continue to follow the terminology of [GB92], hoping that this will not lead to any confusion.

- For any class $\mathscr{M} \subseteq |\mathbf{Mod}(\Sigma)|$ of Σ-models, the *theory of* $\mathscr{M}$ is the set $Th_\Sigma(\mathscr{M}) \subseteq \mathbf{Sen}(\Sigma)$ of all the Σ-sentences that are satisfied by all the models in $\mathscr{M}$.
- A set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences is *closed* if $\Phi = Th_\Sigma(Mod_\Sigma(\Phi))$. We will write $Cl_\Sigma(\Phi)$ for $Th_\Sigma(Mod_\Sigma(\Phi))$ and refer to $Cl_\Sigma(\Phi)$ as the *closure of* Φ.
- A class $\mathscr{M} \subseteq |\mathbf{Mod}(\Sigma)|$ of Σ-models is *closed* if $\mathscr{M} = Mod_\Sigma(Th_\Sigma(\mathscr{M}))$. Closed classes of models will be called *definable*. The *closure* of $\mathscr{M}$ is the class $Mod_\Sigma(Th_\Sigma(\mathscr{M}))$. □

The basic properties of the above notions follow from the fact that Th_Σ and Mod_Σ form a Galois connection:

Proposition 4.2.2. *For any signature Σ, the mappings Th_Σ and Mod_Σ form a Galois connection between sets of Σ-sentences and classes of Σ-models ordered by inclusion.*

Proof. The proof is just the same (and just as easy) as in the equational case, cf. Proposition 2.3.2. □

Corollary 4.2.3. *For any signature Σ, set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences, and class $\mathscr{M} \subseteq |\mathbf{Mod}(\Sigma)|$ of Σ-models:*

$$\Phi \subseteq Th_\Sigma(\mathscr{M}) \quad \textit{iff} \quad Mod_\Sigma(\Phi) \supseteq \mathscr{M}$$ □

Exercise 4.2.4. Construct counterexamples that show that under the assumptions of Corollary 4.2.3 neither of the following implications holds:

$$Mod_\Sigma(\Phi) \subseteq \mathscr{M} \text{ implies } Th_\Sigma(\mathscr{M}) \subseteq \Phi$$
$$Th_\Sigma(\mathscr{M}) \subseteq \Phi \text{ implies } Mod_\Sigma(\Phi) \subseteq \mathscr{M}.$$

Prove that the former implication holds if Φ is closed, and the latter if $\mathscr{M}$ is closed (i.e., is definable). □

The satisfaction relation determines in the obvious way a consequence relation between sentences of the institution:

Definition 4.2.5 (Semantic consequence). Let Σ be an arbitrary signature. A Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$ is a *semantic consequence* of a set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences, written $\Phi \models_\Sigma \varphi$, if $\varphi \in Cl_\Sigma(\Phi)$, or equivalently, if $Mod_\Sigma(\Phi) \models_\Sigma \varphi$. □

As usual, the subscript Σ will often be omitted.

In the following we will often implicitly rely on three basic properties of semantic consequence, namely that it is reflexive, closed under weakening, and transitive, in the following sense:

Proposition 4.2.6. *Let Σ be a signature. Consider any Σ-sentences $\varphi, \psi \in \mathbf{Sen}(\Sigma)$, and sets of Σ-sentences $\Phi, \Psi \subseteq \mathbf{Sen}(\Sigma)$, and $\Psi_\varphi \subseteq \mathbf{Sen}(\Sigma)$ for each $\varphi \in \Phi$. Then:*

1. $\{\varphi\} \models_\Sigma \varphi$.
2. If $\Phi \models_\Sigma \varphi$ *then* $\Phi \cup \Psi \models_\Sigma \varphi$.

3. If $\Phi \models_\Sigma \psi$ *and* $\Psi_\varphi \models_\Sigma \varphi$ *for each* $\varphi \in \Phi$ *then* $\bigcup_{\varphi\in\Phi} \Psi_\varphi \models_\Sigma \psi$*.*

Proof. Directly from the definition. □

Another important property of semantic consequence is that it is preserved by translation along signature morphisms:

Proposition 4.2.7. *For any signature morphism* $\sigma : \Sigma \to \Sigma'$*, set* $\Phi \subseteq \mathbf{Sen}(\Sigma)$ *of* Σ*-sentences, and* Σ*-sentence* $\varphi \in \mathbf{Sen}(\Sigma)$*,*

$$\Phi \models_\Sigma \varphi \;\; \textit{implies} \;\; \sigma(\Phi) \models_{\Sigma'} \sigma(\varphi).$$

In other words, $\sigma(Cl_\Sigma(\Phi)) \subseteq Cl_{\Sigma'}(\sigma(\Phi))$*.*

Proof. Let $M' \in Mod_{\Sigma'}(\sigma(\Phi))$. Then by the satisfaction condition $M'|_\sigma \in Mod_\Sigma(\Phi)$, and so by the definition of the consequence relation $M'|_\sigma \models \varphi$. Thus, by the satisfaction condition again, $M' \models \sigma(\varphi)$, which shows that indeed $\sigma(\Phi) \models \sigma(\varphi)$. □

In general, the reverse implication does not hold, that is, the consequence relation is not reflected by translation along signature morphisms.

Exercise 4.2.8. Try to prove the opposite implication, and notice where the proof breaks down. Then construct a counterexample showing that $\sigma(\Phi) \models \sigma(\varphi)$ does not imply that $\Phi \models \varphi$ even in the standard equational institution **EQ**. (HINT: See Proposition 4.2.15 below.) □

Corollary 4.2.9. *Under the assumptions of Proposition 4.2.7,* $Cl_{\Sigma'}(\sigma(Cl_\Sigma(\Phi))) = Cl_{\Sigma'}(\sigma(\Phi))$*.* □

The above corollary implies that when we want to "move" the closure of a set of sentences from one signature to another, it is enough to move only the set itself; all its consequences can be derived over the target signature as well.

Another consequence of Proposition 4.2.7 is that closure of a set of sentences is reflected by translation along signature morphisms:

Corollary 4.2.10. *For any signature morphism* $\sigma : \Sigma \to \Sigma'$ *and set* $\Phi' \subseteq \mathbf{Sen}(\Sigma')$ *of* Σ'*-sentences, if* Φ' *is closed then so is* $\sigma^{-1}(\Phi')$*.*

Proof. Suppose Φ' is closed and let $\varphi \in Cl_\Sigma(\sigma^{-1}(\Phi'))$. First, notice that since $\sigma(\sigma^{-1}(\Phi')) \subseteq \Phi'$, $Cl_{\Sigma'}(\sigma(\sigma^{-1}(\Phi'))) \subseteq Cl_{\Sigma'}(\Phi')$. Now, by Proposition 4.2.7, $\sigma(\varphi) \in Cl_{\Sigma'}(\sigma(\sigma^{-1}(\Phi'))) \subseteq Cl_{\Sigma'}(\Phi') = \Phi'$. Thus, $\varphi \in \sigma^{-1}(\Phi')$. □

Notice that the above does not imply that "closure commutes with inverse image" in general; only one of the required inclusions holds:

Corollary 4.2.11. *For any signature morphism* $\sigma : \Sigma \to \Sigma'$*, set* $\Phi' \subseteq \mathbf{Sen}(\Sigma')$ *of* Σ'*-sentences, and* Σ*-sentence* $\varphi \in \mathbf{Sen}(\Sigma)$*, if* $\sigma^{-1}(\Phi') \models \varphi$ *then* $\Phi' \models \sigma(\varphi)$*. In other words,* $Cl_\Sigma(\sigma^{-1}(\Phi')) \subseteq \sigma^{-1}(Cl_{\Sigma'}(\Phi'))$*.* □

Exercise 4.2.12. Show that the reverse inclusion does not hold in the standard equational institution **EQ**. □

Forming the closure of a set of sentences consists of two phases: first taking the class of models the set defines, and then taking the theory of this class. Separation of these two phases by translation along a signature morphism preserves the closure to some extent only:

Proposition 4.2.13. *For any signature morphism $\sigma:\Sigma \to \Sigma'$ and set $\Phi' \subseteq \mathbf{Sen}(\Sigma')$ of Σ'-sentences,*

$$Cl_\Sigma(\sigma^{-1}(\Phi')) \subseteq Th_\Sigma(Mod_{\Sigma'}(\Phi')|_\sigma) = \sigma^{-1}(Cl_{\Sigma'}(\Phi'))$$

where for any class $\mathscr{M} \subseteq |\mathbf{Mod}(\Sigma')|$, $\mathscr{M}|_\sigma = \{M'|_\sigma \mid M' \in \mathscr{M}\}$.

Proof. For the first part, let $\varphi \in Cl_\Sigma(\sigma^{-1}(\Phi'))$. Then, by Corollary 4.2.11, $\Phi' \models_{\Sigma'} \sigma(\varphi)$. Hence, by the satisfaction condition, $Mod_{\Sigma'}(\Phi')|_\sigma \models_\Sigma \varphi$, and so $\varphi \in Th_\Sigma(Mod_{\Sigma'}(\Phi')|_\sigma)$.

Since $Mod_{\Sigma'}(\Phi') = Mod_{\Sigma'}(Cl_{\Sigma'}(\Phi'))$, this shows $Th_\Sigma(Mod_{\Sigma'}(\Phi')|_\sigma) = Th_\Sigma(Mod_{\Sigma'}(Cl_{\Sigma'}(\Phi'))|_\sigma) \supseteq Cl_\Sigma(\sigma^{-1}(Cl_{\Sigma'}(\Phi'))) \supseteq \sigma^{-1}(Cl_{\Sigma'}(\Phi'))$, and hence also proves one inclusion ("$\supseteq$") of the second part. For the opposite inclusion, consider $\varphi \in Th_\Sigma(Mod_{\Sigma'}(\Phi')|_\sigma)$, that is $Mod_{\Sigma'}(\Phi')|_\sigma \models_\Sigma \varphi$. By the satisfaction condition, $Mod_{\Sigma'}(\Phi') \models_{\Sigma'} \sigma(\varphi)$, which means $\sigma(\varphi) \in Cl_{\Sigma'}(\Phi')$, and so indeed $\varphi \in \sigma^{-1}(Cl_{\Sigma'}(\Phi'))$. □

Corollary 4.2.14. *For any signature morphism $\sigma:\Sigma \to \Sigma'$ and set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences, $Cl_\Sigma(\Phi) \subseteq \sigma^{-1}(Cl_{\Sigma'}(\sigma(\Phi)))$.* □

Just as the implication opposite to the one stated in Proposition 4.2.7 does not hold in general, the inclusion opposite to the one above does not hold in general either. This changes for *surjective* reduct functors.

Proposition 4.2.15. *Consider a signature morphism $\sigma:\Sigma \to \Sigma'$ such that the reduct functor $\_|_\sigma:\mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$ is surjective on models. For any set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences and Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$,*

$$\Phi \models_\Sigma \varphi \quad \mathit{iff} \quad \sigma(\Phi) \models_{\Sigma'} \sigma(\varphi).$$

Proof. We prove only the implication opposite to that of Proposition 4.2.7. Let $M \in |\mathbf{Mod}(\Sigma)|$ be an arbitrary Σ-model, and let $M' \in |\mathbf{Mod}(\Sigma')|$ be a σ-expansion of M, i.e., $M'|_\sigma = M$ (such an M' exists since $\_|_\sigma$ is surjective on models). If $M \models_\Sigma \Phi$ then by the satisfaction condition $M' \models_{\Sigma'} \sigma(\Phi)$, and so $M' \models_{\Sigma'} \sigma(\varphi)$. Thus, by the satisfaction condition again, $M \models_\Sigma \varphi$. □

Corollary 4.2.16. *Under the assumptions of Proposition 4.2.15, $Cl_\Sigma(\Phi) = \sigma^{-1}(Cl_{\Sigma'}(\sigma(\Phi)))$.* □

This shows that the surjectivity of the reduct functor ensures that moving along a signature morphism is "sound" and "complete" as a strategy for deciding if $\Phi \models_\Sigma \varphi$ by checking whether or not $\sigma(\Phi) \models_{\Sigma'} \sigma(\varphi)$ — without this property, such a strategy is still "complete" (the satisfaction condition ensures that no consequences are lost) but is not always "sound" (new consequences between "old" sentences may be added).

Exercise 4.2.17. Provide an example showing that surjectivity of $\_|_\sigma:\mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$ is not a necessary condition for the conclusions of Proposition 4.2.15 and Corollary 4.2.16. □

Exercise 4.2.18. Show that the inclusion $Cl_\Sigma(\Phi) \subseteq \sigma^{-1}(Cl_{\Sigma'}(\sigma(\Phi)))$, for any $\sigma:\Sigma \to \Sigma'$ and $\Phi \subseteq \mathbf{Sen}(\Sigma)$, directly implies (and, in fact, is equivalent to) Corollary 4.2.11. However, the opposite inclusion $Cl_\Sigma(\Phi) \supseteq \sigma^{-1}(Cl_{\Sigma'}(\sigma(\Phi)))$ does not imply the opposite to the inclusion there: even under the assumptions of Proposition 4.2.15 and Corollary 4.2.16, the inclusion $Cl_\Sigma(\sigma^{-1}(\Phi')) \supseteq \sigma^{-1}(Cl_{\Sigma'}(\Phi'))$ may fail for a set $\Phi' \subseteq \mathbf{Sen}(\Sigma')$ of Σ'-sentences. (HINT: One way to construct a counterexample is to add *false* to the set of sentences of **EQ** for some, but not all signatures.)

Show, however, that under the assumptions of Proposition 4.2.15, for any set $\Phi' \subseteq \mathbf{Sen}(\Sigma')$ of Σ'-sentences, $Cl_\Sigma(\sigma^{-1}(\Phi')) = Th_\Sigma(Mod_{\Sigma'}(\Phi')|_\sigma)$ and $Cl_\Sigma(\sigma^{-1}(\Phi')) = \sigma^{-1}(Cl_{\Sigma'}(\Phi'))$ provided that in addition $\sigma : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$ is surjective. Discuss why this fact does not seem very interesting. □

The following generalisation of Proposition 4.2.15 underlies the key corollary below.

Proposition 4.2.19. *Let $\sigma:\Sigma \to \Sigma'$ be a signature morphism. Suppose that a set $\Gamma \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences exactly characterises the σ-reducts of Σ'-models that satisfy a set $\Gamma' \subseteq \mathbf{Sen}(\Sigma')$ of Σ'-sentences, that is, $Mod_\Sigma(\Gamma) = \mathbf{Mod}(\sigma)(Mod_{\Sigma'}(\Gamma'))$. Then for any set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences and Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$, $\Phi \cup \Gamma \models_\Sigma \varphi$ if and only if $\sigma(\Phi) \cup \Gamma' \models_{\Sigma'} \sigma(\varphi)$.*

Proof. For the "if" part, assume that $\sigma(\Phi) \cup \Gamma' \models_{\Sigma'} \sigma(\varphi)$ and let $M \models_\Sigma \Phi \cup \Gamma$. Then, since $M \in Mod_\Sigma(\Gamma)$, there exists $M' \in Mod_{\Sigma'}(\Gamma')$ with $M'|_\sigma = M$. By the satisfaction condition, $M' \models_{\Sigma'} \sigma(\Phi)$, hence $M' \models_{\Sigma'} \sigma(\Phi) \cup \Gamma'$ and so $M' \models_{\Sigma'} \sigma(\varphi)$ as well. Thus, by the satisfaction condition again, $M \models_\Sigma \varphi$.

For the "only if" part, asume that $\Phi \cup \Gamma \models_\Sigma \varphi$ and let $M' \models_{\Sigma'} \sigma(\Phi) \cup \Gamma'$. Then by the satisfaction condition, $M'|_\sigma \models_\Sigma \Phi$ and moreover, by the assumption, $M'|_\sigma \models_\Sigma \Gamma$. Hence, $M'|_\sigma \models_\Sigma \Phi \cup \Gamma$, and so $M'|_\sigma \models_\Sigma \varphi$ as well, which by the satisfaction condition again proves that $M' \models_{\Sigma'} \sigma(\varphi)$. □

Corollary 4.2.20. *Let $\sigma:\Sigma \to \Sigma'$ be a signature morphism. Suppose that a set $\Gamma \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences exactly characterises the σ-reducts of Σ'-models, that is, $Mod_\Sigma(\Gamma) = \mathbf{Mod}(\sigma)(|\mathbf{Mod}(\Sigma')|)$. Then for any set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences and Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$, $\Phi \cup \Gamma \models_\Sigma \varphi$ if and only if $\sigma(\Phi) \models_{\Sigma'} \sigma(\varphi)$.* □

Exercise 4.2.21. Show that Proposition 4.2.15 follows directly from Proposition 4.2.19 (or Corollary 4.2.20). Generalise Corollary 4.2.16 in a similar way. □

Definition 4.2.22 (Presentation). For any signature Σ, a *Σ-presentation* (also known as a *flat specification*) is a pair $\langle \Sigma, \Phi \rangle$ where $\Phi \subseteq \mathbf{Sen}(\Sigma)$. $M \in |\mathbf{Mod}(\Sigma)|$ is a *model* of a Σ-presentation $\langle \Sigma, \Phi \rangle$ if $M \models \Phi$. $Mod[\langle \Sigma, \Phi \rangle]$ denotes the class of all models of the presentation $\langle \Sigma, \Phi \rangle$, and $\mathbf{Mod}[\langle \Sigma, \Phi \rangle]$ the full subcategory of $\mathbf{Mod}(\Sigma)$ with objects in $Mod[\langle \Sigma, \Phi \rangle]$. □

Definition 4.2.23 (The category of theories). For any signature Σ, a Σ-*theory* T is a Σ-presentation $\langle\Sigma,\Phi\rangle$ where Φ is closed. A Σ-presentation $\langle\Sigma,\Psi\rangle$ *presents* the Σ-theory $\langle\Sigma, Cl_\Sigma(\Psi)\rangle$.

For any theories $T=\langle\Sigma,\Phi\rangle$ and $T'=\langle\Sigma',\Phi'\rangle$, a *theory morphism* $\sigma: T\to T'$ is a signature morphism $\sigma:\Sigma\to\Sigma'$ such that $\sigma(\varphi)\in\Phi'$ for every $\varphi\in\Phi$.

The category $\mathbf{Th}_{\mathbf{INS}}$ of theories in **INS** has theories as objects and theory morphisms as morphisms, with identities and composition inherited from the category $\mathbf{Sign}_{\mathbf{INS}}$ of signatures of **INS**. □

The satisfaction condition implies the following important characterisation of theory morphisms, analogous to that given for equational theory morphisms in Proposition 2.3.13.

Proposition 4.2.24. *For any signature morphism $\sigma:\Sigma\to\Sigma'$ and sets $\Phi\subseteq\mathbf{Sen}(\Sigma)$ and $\Phi'\subseteq\mathbf{Sen}(\Sigma')$ of sentences, the following conditions are equivalent:*

1. *σ is a theory morphism $\sigma:\langle\Sigma,Cl_\Sigma(\Phi)\rangle\to\langle\Sigma',Cl_{\Sigma'}(\Phi')\rangle$.*
2. *$\sigma(\Phi)\subseteq Cl_{\Sigma'}(\Phi')$.*
3. *For every $M'\in Mod_{\Sigma'}(\Phi')$, $M'|_\sigma\in Mod_\Sigma(\Phi)$.*

Proof.

$1\Rightarrow 2$: Obvious, since $\Phi\subseteq Cl_\Sigma(\Phi)$.

$2\Rightarrow 3$: Consider $M'\in Mod_{\Sigma'}(\Phi')$. Then also $M'\in Mod_{\Sigma'}(Cl_{\Sigma'}(\Phi'))$, and so for all $\varphi\in\Phi$, $M'\models\sigma(\varphi)$ (since $\sigma(\varphi)\in Cl_{\Sigma'}(\Phi')$). Hence, by the satisfaction condition, $M'|_\sigma\models\varphi$, and thus indeed $M'|_\sigma\in Mod_\Sigma(\Phi)$.

$3\Rightarrow 1$: Consider any $\varphi\in Cl_\Sigma(\Phi)$. We have to show that $\sigma(\varphi)\in Cl_{\Sigma'}(\Phi')$, that is that for all $M'\in Mod_{\Sigma'}(\Phi')$, $M'\models\sigma(\varphi)$. However, if $M'\in Mod_{\Sigma'}(\Phi')$ then $M'|_\sigma\in Mod_\Sigma(\Phi)$. Hence, $M'|_\sigma\models\varphi$, and the conclusion follows from the satisfaction condition. □

Exercise 4.2.25. Define the category $\mathbf{Pres}_{\mathbf{INS}}$ of presentations in **INS**, with morphisms $\sigma:\langle\Sigma,\Phi\rangle\to\langle\Sigma',\Phi'\rangle$ that are signature morphisms $\sigma:\Sigma\to\Sigma'$ such that $\Phi'\models\sigma(\varphi)$ for all $\varphi\in\Phi$. Check that $\mathbf{Th}_{\mathbf{INS}}$ is a full subcategory of $\mathbf{Pres}_{\mathbf{INS}}$ and that the two categories are equivalent. □

Exercise 4.2.26. Show that by Proposition 4.2.24 above, the mapping which to any theory assigns the category of its models extends to a functor $\mathbf{Mod}:\mathbf{Th}_{\mathbf{INS}}^{op}\to\mathbf{Cat}$, where:

- for any theory $T=\langle\Sigma,\Phi\rangle$, $\mathbf{Mod}[T]$ is the full subcategory of $\mathbf{Mod}(\Sigma)$ with objects in $Mod[T]$ as in Definition 4.2.22; and
- for any theory morphism $\sigma: T\to T'$, $\mathbf{Mod}(\sigma)$ is the reduct functor $\_|_\sigma:\mathbf{Mod}[T']\to\mathbf{Mod}[T]$. □

Many standard properties of theories (and presentations) investigated in the realm of classical model theory may be formulated in the framework of an arbitrary institution. For example:

Definition 4.2.27 (Consistency and completeness of a presentation). A presentation $\langle\Sigma,\Phi\rangle$ is *consistent* if it has a model, i.e. if $Mod[\langle\Sigma,\Phi\rangle]\neq\varnothing$.

A presentation $\langle\Sigma,\Phi\rangle$ is *complete* if it is a maximal consistent presentation, i.e. if it is consistent and no presentation $\langle\Sigma,\Phi'\rangle$ such that Φ' properly contains Φ is consistent. □

Proposition 4.2.28. *A presentation $\langle\Sigma,\Phi\rangle$ is consistent if and only if the theory $\langle\Sigma,Cl_\Sigma(\Phi)\rangle$ is consistent. Any complete presentation is a (consistent) theory.* □

Definition 4.2.29 (Conservative theory morphism). For any theories $T=\langle\Sigma,\Phi\rangle$ and $T'=\langle\Sigma',\Phi'\rangle$, a theory morphism $\sigma\colon T\to T'$ is *conservative* if for every Σ-sentence φ, $\varphi\in\Phi$ whenever $\sigma(\varphi)\in\Phi'$.

A theory morphism $\sigma\colon T\to T'$ *admits model expansion* if the corresponding reduct function $\_|_\sigma\colon Mod_{\Sigma'}(\Phi')\to Mod_\Sigma(\Phi)$ is surjective, that is, for every Σ-model M such that $M\models_\Sigma\Phi$, there exists a Σ'-model M' such that $M'\models_{\Sigma'}\Phi'$ and $M'|_\sigma=M$. □

Exercise 4.2.30. As in Proposition 4.2.15, show that a theory morphism $\sigma\colon T\to T'$ is conservative if it admits model expansion. Note that the opposite implication does not hold by Exercise 4.2.17. □

The careful reader has probably realised that in this section we have not even mentioned model morphisms. Indeed, everything above works equally well if we forget about the category structure provided on the collections of models in an institution. But this proves inadequate for some purposes; see for example the next section where the category structure on models is exploited.

4.3 Constraints

As discussed in Section 2.5, the class of all models that satisfy a given presentation often contains some models that intuitively are undesirable realisations of the presentation. Different methods are used to constrain the semantics of presentations so that from among all its models only the ones that are "desirable" are selected: for example, one may take its initial semantics, reachable semantics, final semantics, etc. (cf. Sections 2.5 and 2.7.2). How do these fit into the institutional framework introduced above? Let us consider initiality constraints[18] first.

There is clearly no problem with expressing the basic concept of initial model in an arbitrary institution: models over any signature form a category, hence the class of models satisfying a given presentation determines a full subcategory of this category — and we know what initiality means in any category (cf. Section 3.2.1).

Let $\mathbf{INS}=\langle\mathbf{Sign},\mathbf{Sen},\mathbf{Mod},\langle\models_\Sigma\rangle_{\Sigma\in|\mathbf{Sign}|}\rangle$ be an institution, fixed throughout this section.

[18] We use the term "constraint" here following the terminology of [BG80], [GB92]. Initiality and data constraints as discussed and formally defined below have nothing to do with constraints as used in "constraint logic programming" [JL87].

Definition 4.3.1 (Initial model of a presentation). For any signature $\Sigma \in |\mathbf{Sign}|$ and set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of sentences, the *initial model* of the presentation $\langle \Sigma, \Phi \rangle$ is the (unique up to isomorphism) initial object in $Mod_\Sigma(\Phi)$ considered as a full subcategory of $\mathbf{Mod}(\Sigma)$. □

We might feel tempted to pursue a number of possibilities to incorporate the idea of initiality into the institutional framework:

- We may hope to be able to modify all institutions of interest so that they yield initial semantics directly, by changing the model functor **Mod** to yield only the initial models as models over any signature. Clearly, this fails: requiring initiality only makes sense relative to a presentation. If sentences are not taken into account then for example the only initial models in the institution **EQ** of equational logic are ground term algebras.
- We can attempt to modify the satisfaction relation so that only the initial models of a sentence will be defined to satisfy it. Quite obviously, this does not work, since it would then be impossible to adequately define models of presentations involving more than one sentence. Without modifying the satisfaction relation, we could modify Definitions 4.2.1 and 4.2.22 and consider only initial models of presentations by defining $Mod_\Sigma(\Phi)$ to consist only of the initial models in $\{M \mid M \models \Phi\}$ considered as a full subcategory of $\mathbf{Mod}(\Sigma)$. But this would make the whole theory rather clumsy, and the various definitions would not fit together as neatly as they do now. For example, Propositions 4.2.7 and 4.2.24 would no longer hold. Worse, this would not allow the user to write axioms that are to be interpreted in a loose, non-initial fashion, indicating that only certain parts of a specification are to be interpreted in an initial way. See Example 4.3.2 below.
- We can view the requirement of initiality with respect to a presentation as just another *sentence*. This would be a rather complicated sentence, as it has to contain other sentences within it, but in view of examples like 4.1.38 (not to mention 4.1.35) there is no reason why this should bother us. This is the approach we will take.

It is not sufficient to define initiality constraints simply as sets of sentences over a given signature, and then to define their satisfaction via the notion of an initial model. The problem is that we do not always want to constrain the entire model of a presentation. As the following example illustrates, we need to be able to constrain only a certain part of this model, that is, to impose initiality constraints on its reduct to a certain subsignature.

Example 4.3.2. Recall Exercise 2.5.21 which concerned the specification of a function $ch\colon nat \to nat$ that for each natural number n chooses an arbitrary number that is greater than n. As argued there, we certainly do not want to take the initial model of the entire specification: the initial model would generate "artificial elements" of sort *nat* (as the results of the function *ch*) and then artificial elements of sort *bool* as well (as results of comparisons by $<$ involving the artificial elements of sort *nat*). What one would like is to first interpret the original specification NAT of natural numbers in an initial way, do the same for the specification BOOL, add the operation

$\_\_<\_\_ : nat \times nat \to bool$ (which is defined by its axioms in a sufficiently complete way) — it so happens that this would be the same as taking an initial model of these specifications put together — and only then add an operation $ch : nat \to nat$ with the corresponding axiom interpreted in the underlying logic, with no initiality restrictions intervening in any way at this stage. □

By allowing initiality requirements to be "fitted" to larger signatures by signature morphisms, along the lines of the construction presented in Example 4.1.46, we can impose the initiality requirement on parts of models.

Definition 4.3.3 (Initiality constraint). Let $\Sigma \in |\mathbf{Sign}|$ be a signature. A Σ-*initiality constraint* is a pair $\langle \Phi', \theta \rangle$, written as **initial** Φ' **through** θ, where $\theta : \Sigma' \to \Sigma$ is a signature morphism and $\Phi' \subseteq \mathbf{Sen}(\Sigma')$ is a set of Σ'-sentences. A Σ-model $M \in |\mathbf{Mod}(\Sigma)|$ *satisfies* a Σ-initiality constraint **initial** Φ' **through** θ if its reduct $M|_\theta \in |\mathbf{Mod}(\Sigma')|$ is an initial model of $\langle \Sigma', \Phi' \rangle$. □

Now, such an initiality constraint may be regarded as just another sentence in a presentation, and freely mixed with "ordinary" sentences.

Exercise 4.3.4. Redo Exercise 2.5.21 using initiality constraints. Discuss the possibility of achieving the same effect without the "fitting morphism" component in initiality constraints. □

The specification built in Exercise 4.3.4 is not a presentation in **FOEQ** — we have to extend this institution by adding initiality constraints first. Indeed, given an institution **INS** we can always form a new institution $\mathbf{INS}^{init}$ in which initiality constraints are allowed as additional sentences. Such a construction is implicitly involved whenever initiality constraints are used.

Definition 4.3.5 (Institution with initiality constraints). The institution $\mathbf{INS}^{init}$ with initiality constraints in **INS** is defined as follows:

- The category $\mathbf{Sign}_{\mathbf{INS}^{init}}$ of signatures is just **Sign**, the same as in **INS**.
- The functor $\mathbf{Sen}_{\mathbf{INS}^{init}}$ gives:
  - for each signature Σ, the (disjoint) union of the set $\mathbf{Sen}(\Sigma)$ of Σ-sentences in **INS** and of the set of Σ-initiality constraints;[19] and
  - for each signature morphism $\sigma : \Sigma \to \Sigma_1$, the translation function $\mathbf{Sen}_{\mathbf{INS}^{init}}(\sigma)$ that works as $\mathbf{Sen}(\sigma)$ on all the "old" Σ-sentences in **INS**, and for any Σ-initiality constraint **initial** Φ' **through** θ, where $\theta : \Sigma' \to \Sigma$ and $\Phi' \subseteq \mathbf{Sen}(\Sigma')$, is defined by $\mathbf{Sen}_{\mathbf{INS}^{init}}(\sigma)($**initial** Φ' **through** $\theta) =$ **initial** Φ' **through** $\theta;\sigma$.
- The functor $\mathbf{Mod}_{\mathbf{INS}^{init}}$ is just **Mod**, the same as in **INS**.
- For each signature $\Sigma \in |\mathbf{Sign}_{\mathbf{INS}^{init}}|$, the Σ-satisfaction relation $\models_{\mathbf{INS}^{init},\Sigma}$ is the same as the Σ-satisfaction relation in **INS** for the Σ-sentences from **INS**, and is given by Definition 4.3.3 for Σ-initiality constraints. □

[19] As in Example 4.1.46, this may lead to some foundational difficulties which we disregard here, cf. footnote 16.

Exercise 4.3.6. Present the above definition as an instance of the construction given in Example 4.1.46. Notice that this is sufficient to conclude that $\mathbf{INS}^{init}$ is indeed an institution.

Show (referring for example to Exercise 4.3.4) that in general the translation of an initiality constraint cannot be given without the "fitting morphism" component, and so we would not be able to define an institution where only initiality constraints with trivial (identity) fitting morphisms would be allowed. □

Exercise 4.3.7. Working in the institution **EQ**, follow Definition 4.3.3 and define *reachability constraints* that are satisfied only by algebras having an indicated reduct that is reachable. Note that axioms used in initiality constraints play no role here, so you can adopt a syntax like **reachable through** θ. Following Definition 4.3.5, define an institution $\mathbf{EQ}^{reach}$ extending **EQ** by reachability constraints.

Assuming that each category of models in **INS** comes equipped with a factorisation system (Section 3.3), introduce reachability constraints for **INS** using Definition 3.3.7 and extend **INS** correspondingly. □

The use of initiality constraints as introduced above is not always entirely satisfactory. Often, rather than requiring that a certain part of a model is initial, we want to require it to be a *free extension* of some other part. Natural examples arise when we want to specify data structures built on an arbitrary set of elements, like lists, sets or bags of arbitrary elements. This involves imposing the requirement that an algebra modelling the data structure is a free extension of its reduct to the sort of elements. To formalise this, the concept of a data constraint is introduced below.

Definition 4.3.8 (Data constraint). Let $\Sigma \in |\mathbf{Sign}|$ be a signature.

A Σ-*data constraint* is a triple $\langle \sigma, \Phi', \theta \rangle$, written as **data** Φ' **over** σ **through** θ, where $\sigma : \Sigma_1 \to \Sigma'$ and $\theta : \Sigma' \to \Sigma$ are signature morphisms and $\Phi' \subseteq \mathbf{Sen}(\Sigma')$ is a set of Σ'-sentences.

A Σ-model $M \in |\mathbf{Mod}(\Sigma)|$ *satisfies* the data constraint **data** Φ' **over** σ **through** θ if its reduct $M|_\theta \in |\mathbf{Mod}(\Sigma')|$ to a Σ'-model is a free model of Φ' w.r.t. the reduct functor $\_|_\sigma : \mathbf{Mod}[\langle \Sigma, \Phi' \rangle] \to \mathbf{Mod}(\Sigma_1)$ over $(M|_\theta)|_\sigma$, with the identity as unit. That is, M satisfies **data** Φ' **over** σ **through** θ if:

- $M|_\theta \models_{\Sigma'} \Phi'$; and
- for any $M' \in Mod_{\Sigma'}(\Phi')$ and Σ_1-morphism $f : M|_{\sigma;\theta} \to M'|_\sigma$ there exists a unique Σ'-morphism $f^\# : M|_\theta \to M'$ such that $f^\#|_\sigma = f$. □

Exercise 4.3.9. Using data constraints, give a specification of finite bags of an arbitrary set of elements. □

Exercise 4.3.10. Following the pattern of Definition 4.3.5 (and of Example 4.1.46), define the institution $\mathbf{INS}^{data}$ by adding data constraints as additional sentences to **INS**. □

Note that nowhere in the above has it been assumed that initial models of presentations actually exist in general (nor that the reduct functor used in Definition 4.3.8

has a left adjoint). We do know that in some institutions (for example, in the institution **EQ** of equational logic and in the institution **PEQ** of partial equational logic) any set of sentences over a given signature has an initial model (see Theorem 2.5.14 for the case of **EQ**). On the other hand, there are institutions in which some sets of sentences do not have initial models; the institution **FOEQ** of first-order logic with equality is an example (see Example 2.7.11). Nevertheless, the above definitions work for an arbitrary institution. If a set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences has no initial model, then an initiality constraint **initial** Φ **through** θ based on this set has no model, even if the class $Mod_\Sigma(\Phi)$ of models of this set of sentences is not empty.

Exercise 4.3.11. Any set of sentences in the equational institution **EQ** has a model, and moreover, it has an initial model. Show that neither of these properties carries over to the institution $\mathbf{EQ}^{init}$ of initiality constraints in **EQ**. That is, give a presentation in $\mathbf{EQ}^{init}$ that has no model. □

Exercise 4.3.12. Recall the institution **Horn** of Horn formulae from Exercise 4.1.21 and show that every set of sentences in **Horn** has an initial model. Discuss the interpretation of predicates in initial models: notice that they hold "minimally", meaning that only positive cases need to be explicitly specified. Extend this analysis to data constraints, and use this to specify the transitive and reflexive closure of an arbitrary binary predicate. □

Exercise 4.3.13. Working in the institution **EQ** as in Exercise 4.3.7, follow Definition 4.3.8 and define *generation constraints* **generated over** σ **through** θ that are satisfied by algebras A such that $A|_\theta$ is generated in a suitable sense by $A|_{\sigma;\theta}$. Define an institution $\mathbf{EQ}^{gen}$ extending **EQ** by generation constraints.

Assuming that each category of models in **INS** comes equipped with a factorisation system (Section 3.3), introduce generation constraints for **INS** anticipating Definition 4.5.1 and extend **INS** correspondingly. □

Exercise 4.3.14. Following Exercise 3.5.24, dualise the concept of data constraint. A *co-data constraint* in an institution **INS** can be written as **co-data** Φ' **over** σ **through** θ, where Φ', σ and θ are as in Definition 4.3.8. A Σ-model $M \in |\mathbf{Mod}(\Sigma)|$ *satisfies* **co-data** Φ' **over** σ **through** θ if $M|_\theta$ is a cofree model of Φ' w.r.t. the reduct functor $\_|_\sigma : \mathbf{Mod}[\langle\Sigma',\Phi'\rangle] \to \mathbf{Mod}(\Sigma_1)$ over its σ-reduct, with the identity as counit, that is, if $M|_\theta \models_{\Sigma'} \Phi'$ and for any $M' \in Mod_{\Sigma'}(\Phi')$ and Σ_1-morphism $f : M'|_\sigma \to M|_{\sigma;\theta}$ there exists a unique Σ'-morphism $f^\# : M' \to M|_\theta$ such that $f^\#|_\sigma = f$. Extend this definition to build an institution $\mathbf{INS}^{codata}$ by adding co-data constraints as additional sentences to **INS**.

Discuss the use of co-data constraints in standard institutions like **EQ** and **FOPEQ**. For instance, consider the following simple presentation:

$$\begin{array}{lll} \textbf{spec } \mathrm{STREAM} = & \textbf{sorts} & elem, stream \\ & \textbf{ops} & hd{:}\, stream \to elem \\ & & tl{:}\, stream \to stream \\ & & cons{:}\, elem \times stream \to stream \\ & \multicolumn{2}{l}{\forall x{:}\, elem, \forall s{:}\, stream} \\ & & \bullet\ hd(cons(x,s)) = x \\ & & \bullet\ tl(cons(x,s)) = s \end{array}$$

Check that any model M of STREAM that is cofree over $E = |M|_{elem}$ (w.r.t. the reduct functor given by the obvious signature inclusion) is isomorphic to the algebra E^{ω} of (countably) infinite streams of elements from E, with the operations defined in the standard way.

Much the same effect is achieved even when we remove the operation *cons* and the two axioms from this presentation: check that if Σ is a signature with sorts $elem, stream$ and operations $hd{:}\, stream \to elem$, $tl{:}\, stream \to stream$ then cofree Σ-models over their carrier E of sort *elem* are (up to isomorphism) the algebras E^{ω} of (countably) infinite streams of elements from E, with *hd* and *tl* defined in the standard way. Check then that in any such algebra the two axioms in STREAM define the operation *cons* unambiguously. □

4.4 Exact institutions

As illustrated in Sections 4.2 and 4.3, institutions provide a sufficient basis for much of the standard machinery of specifications without the need for further assumptions. Still, the structure and properties of a logical system exposed by the definition of an institution are very limited, and do not provide an adequate basis for many other aspects of the theory and practice of software specification and development. As discussed in the introduction to this chapter, this should not discourage us from working within the institutional framework. On the contrary, it is worth trying to find some adequately abstract additional assumptions that are sufficient for the purpose at hand. As always in mathematics, the main informal guideline to follow is to keep the additional assumptions to a minimum. Part of the payoff is that this forces us to work at a level of generality and abstraction that ensures a deeper understanding of the essence of the studied phenomena, while at the same time covering as many of the cases of potential interest as possible.

In this section and the next we will illustrate this strategy by presenting some extensions to the notion of an institution by additional structure or properties that are required to support study of more detailed properties of specifications.

The ways in which specifications (or programs, systems, or structures of any kind) are put together is the very essence of the theory and methodology of software specification and development. One of the basic tools for “putting things together” is the categorical notion of colimit (cf. Section 3.2) with pushouts as a particularly important special case; see for instance Section 6.3 below. Putting specifications together then involves taking colimits in the category of theories. It would be rather inconvenient to have to establish the existence of a colimit for each diagram of interest separately, so we normally require the category of theories to be cocomplete (or at least finitely cocomplete). Checking this directly would be tedious — and this is why the following general result is useful.

Theorem 4.4.1. *For any institution* **INS***, if the category* $\mathbf{Sign}_{\mathbf{INS}}$ *of signatures in* **INS** *is cocomplete then so is the category* $\mathbf{Th}_{\mathbf{INS}}$ *of theories in* **INS***.*

Proof. Let D be a diagram in $\mathbf{Th}_{\mathbf{INS}}$ with $|G(D)|_{node} = N$ and $D_n = \langle \Sigma_n, \Phi_n \rangle$ for $n \in N$. Let D' be the corresponding diagram in $\mathbf{Sign}_{\mathbf{INS}}$, hence $D'_n = \Sigma_n$ for $n \in N$. By the assumption of the theorem, D' has a colimit, say $\langle \alpha_n : \Sigma_n \to \Sigma \rangle_{n \in N}$. Let $\Phi = Cl_\Sigma(\bigcup_{n \in N} \alpha_n(\Phi_n))$. Then for each $n \in N$, $\alpha_n : \langle \Sigma_n, \Phi_n \rangle \to \langle \Sigma, \Phi \rangle$ is a theory morphism (this is obvious) and $\langle \alpha_n \rangle_{n \in N}$ is a colimit of D in $\mathbf{Th}_{\mathbf{INS}}$. For: first notice that it is a cocone on D (since it is a cocone on D' in $\mathbf{Sign}_{\mathbf{INS}}$), and then consider another cocone on D, say $\langle \beta_n : \langle \Sigma_n, \Phi_n \rangle \to \langle \Sigma', \Phi' \rangle \rangle_{n \in N}$. By the construction, there exists a unique signature morphism $\sigma : \Sigma \to \Sigma'$ such that for each $n \in N$, $\alpha_n;\sigma = \beta_n$. To complete the proof, it is sufficient to show that $\sigma : \langle \Sigma, \Phi \rangle \to \langle \Sigma', \Phi' \rangle$ is a theory morphism. By Proposition 4.2.24, it is enough to show that $\sigma(\bigcup_{n \in N} \alpha_n(\Phi_n)) \subseteq \Phi'$. This easily follows from the fact that for each $n \in N$, β_n is a theory morphism, and hence $\sigma(\alpha_n(\Phi_n)) = (\alpha_n;\sigma)(\Phi_n) = \beta_n(\Phi_n) \subseteq \Phi'$. □

The above proof shows that in fact a stronger property holds: in any institution, the category of theories has all of the colimits that the category of signatures has: the forgetful functor mapping theories to their underlying signatures *lifts colimits*. So, for instance:

Corollary 4.4.2. *For any institution* **INS***, if the category* $\mathbf{Sign}_{\mathbf{INS}}$ *of signatures in* **INS** *is finitely cocomplete then so is the category* $\mathbf{Th}_{\mathbf{INS}}$ *of theories in* **INS***.* □

Notice that the above theorem applies to *any* institution, regardless of the means used to construct it. Hence, for example, if the category $\mathbf{Sign}_{\mathbf{INS}}$ of signatures in an institution **INS** is cocomplete, then not only is the category $\mathbf{Th}_{\mathbf{INS}}$ of theories in **INS** cocomplete, but so are the categories $\mathbf{Th}_{\mathbf{INS}^{init}}$, $\mathbf{Th}_{\mathbf{INS}^{data}}$ and $\mathbf{Th}_{\mathbf{INS}^{codata}}$ of theories in the corresponding institutions with initiality constraints, data constraints and co-data constraints respectively (cf. Definition 4.3.5, Exercise 4.3.10 and Exercise 4.3.14).

Exercise 4.4.3. Assume that the category of signatures of a certain institution has an initial object. What is then an initial object in the category of theories? □

Example 4.4.4. Working in the institution **EQ** of equational logic, recall Example 3.2.35 of a simple pushout of algebraic signatures, and the set $\Phi\mathrm{N{\scriptstyle AT}}$ of equational axioms over the signature $\Sigma\mathrm{N{\scriptstyle AT}}$ given in Exercise 2.5.4. Let $T\mathrm{N{\scriptstyle AT}}$ be the $\Sigma\mathrm{N{\scriptstyle AT}}$-theory presented by $\Phi\mathrm{N{\scriptstyle AT}}$. Let $T\mathrm{N{\scriptstyle AT}}_{fib}$ be the $\Sigma\mathrm{N{\scriptstyle AT}}_{fib}$-theory presented by the axioms $\Phi\mathrm{N{\scriptstyle AT}}_{fib}$ that include $\Phi\mathrm{N{\scriptstyle AT}}$ plus the following:

$$fib(0) = succ(0)$$
$$fib(succ(0)) = succ(0)$$
$$\forall n{:}nat \bullet fib(succ(succ(n))) = fib(succ(n)) + fib(n)$$

Finally, let $T\mathrm{NAT}_{mult}$ be the $\Sigma\mathrm{NAT}_{mult}$-theory presented by the axioms $\Phi\mathrm{NAT}_{mult}$ that include $\Phi\mathrm{NAT}$ plus the following:

$$\forall n{:}nat \bullet mult(0,n) = 0$$
$$\forall n,m{:}nat \bullet mult(succ(n),m) = mult(n,m) + m$$

Now, we have theory inclusions:

$$T\mathrm{NAT}_{fib} \longleftarrow T\mathrm{NAT} \longrightarrow T\mathrm{NAT}_{mult}$$

with the corresponding signature inclusions given in Example 3.2.35. Their pushout is the $\Sigma\mathrm{NAT}_{fib,mult}$-theory $T\mathrm{NAT}_{fib,mult}$ presented by the union of $\Phi\mathrm{NAT}$, $\Phi\mathrm{NAT}_{fib}$ and $\Phi\mathrm{NAT}_{mult}$.

As in Example 3.2.35, this is deceptively simple, as only single-sorted theory inclusions that introduce different operation names are involved.

Exercise. Give examples of pushouts in the category of equational theories with signatures involving more than one sort, extensions with overlapping sets of operation names, and theory morphisms that are not injective on sort and/or on operation names. Notice however that the extra complications come only from the construction of signature pushouts; the theories are defined in much the same way.

Exercise. Obviously, when giving the set of axioms for $T\mathrm{NAT}_{fib,mult}$, $\Phi\mathrm{NAT}$ may be omitted, as it is already included in the other sets of axioms. Try to generalise this remark to "optimise" the construction of the colimit in the category of theories given in the proof of Theorem 4.4.1. □

We have seen how the assumption that the category of signatures of an institution is (finitely) cocomplete ensures that the institution provides means for "putting theories together". It is also interesting to investigate how this relates to "putting models together", which is what structured programming in the large is all about. There is an important difference here: in the above, and in general when dealing with specifications, we were interested in combining theories, i.e., sets of sentences. In model-theoretic terms, this corresponds to combining classes of models. However, when the specified system is being built, we are interested in expanding and combining *individual* models.

Example 4.4.5. Recall Example 4.4.4 of a simple pushout in the category of theories of the institution **EQ** of equational logic. Consider an arbitrary model N of $T\mathrm{NAT}$, any $\Sigma\mathrm{NAT}_{mult}$-algebra N_2 built by adding to N an interpretation of *fib* such that the axioms in $\Phi\mathrm{NAT}_{fib}$ are satisfied, and any $\Sigma\mathrm{NAT}_{mult}$-algebra N_2 built by adding to N an interpretation of *mult* such that the axioms in $\Phi\mathrm{NAT}_{mult}$ are satisfied. Then, much as in Example 3.4.35 where specific such algebras were considered, N_1 and N_2 may be uniquely combined to a $\Sigma\mathrm{NAT}_{fib,mult}$-algebra N' that expands them both. The key property now is that the algebras built in this way are models of the theory $T\textsc{Nat}_{fib,mult}$, and moreover, that all its models may be built in this way. □

It turns out that the crucial link which ensures that constructions to combine theories and to combine models work together smoothly, as in the above example, is the continuity of the model functor in the underlying institution.

Definition 4.4.6 (Exact institution). An institution **INS** is (*finitely*) *exact* if its category of signatures $\mathbf{Sign_{INS}}$ is (finitely) cocomplete and its model functor $\mathbf{Mod_{INS}}\colon \mathbf{Sign}_{\mathbf{INS}}^{op} \to \mathbf{Cat}$ is (finitely) continuous, mapping (finite) colimits in $\mathbf{Sign_{INS}}$ to limits in **Cat**. □

Example 4.4.7. All of the institutions defined in the examples and sketched in the exercises in Section 4.1.1, with the major exception of **FPL** (Example 4.1.25) and perhaps those given in Examples 4.1.35, 4.1.36 and 4.1.37 where we know nothing about the signature categories, are exact. See Exercises 3.2.53 and 3.4.33 for the standard algebraic case of the equational institution **EQ** — all of the other cases require a similar argument. □

Exercise 4.4.8. The abstract formulation of exactness above may somewhat hide the role of this property in "putting models together". Consider an exact institution **INS** and a diagram D in $\mathbf{Sign_{INS}}$ with colimit signature Σ'. Anticipating the crucial case of preservation of signature pushouts treated in Definition 4.4.12, show that (up to isomorphism of categories) $\mathbf{Mod_{INS}}(\Sigma')$ can be defined as follows, where N is the set of nodes in D:

- Σ'-models are families $\langle M_n \in |\mathbf{Mod_{INS}}(D_n)|\rangle_{n\in N}$ that are compatible with signature morphisms in D in the sense that $M_n = M_m|_{D_e}$ for each edge $e\colon n \to m$ in the graph of D; and
- Σ'-morphisms between any such Σ'-models $\langle M_n\rangle_{n\in N}$ and $\langle M'_n\rangle_{n\in N}$ are families $\langle h_n\colon M_n \to M'_n\rangle_{n\in N}$ of morphisms in $\mathbf{Mod_{INS}}(D_n)$, $n \in N$, that are compatible with signature morphisms in D in the sense that $h_n = h_m|_{D_e}$ for each edge $e\colon n \to m$ in the graph of D.

Moreover, for each $n \in N$, the reduct functor w.r.t. the colimit injection from D_n to Σ' is just the projection of such families on the n-th component.
HINT: Use Exercise 3.4.32 (and indirectly Exercise 3.2.53). □

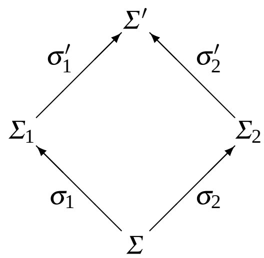
Exercise 4.4.9. Consider a finitely exact institution. Present initiality constraints (Definition 4.3.3) as a special case of data constraints (Definition 4.3.8). Is the assumption that the institution is finitely exact essential? □

Exercise 4.4.10. An interesting standard institution with a cocomplete category of signatures and a model functor that preserves "nearly all" finite colimits of signatures is the institution **SSEQ** of single-sorted equational logic. Give a precise definition of this institution and indicate which colimits of signature diagrams are not preserved by the model functor. HINT: Consider the initial single-sorted signature. □

Definition 4.4.11 (Semi-exact institution). An institution **INS** is *semi-exact* if all pushouts exist in its category of signatures $\mathbf{Sign_{INS}}$ and its model functor $\mathbf{Mod_{INS}}: \mathbf{Sign}_{\mathbf{INS}}^{op} \to \mathbf{Cat}$ preserves pushouts, mapping them to pullbacks in **Cat**. □

A consequence of the assumption that the model functor of an institution preserves signature pushouts is the well-known *Amalgamation Lemma*.

Definition 4.4.12 (Amalgamation property). Let $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ be an institution and consider the following diagram in **Sign**:



This diagram *admits amalgamation* if:

- for any two models $M_1 \in |\mathbf{Mod}(\Sigma_1)|$ and $M_2 \in |\mathbf{Mod}(\Sigma_2)|$ such that $M_1|_{\sigma_1} = M_2|_{\sigma_2}$, there exists a unique model $M' \in |\mathbf{Mod}(\Sigma')|$ such that $M'|_{\sigma_1'} = M_1$ and $M'|_{\sigma_2'} = M_2$ (we call such M' the *amalgamation* of M_1 and M_2); and
- for any two model morphisms $f_1: M_{11} \to M_{12}$ in $\mathbf{Mod}(\Sigma_1)$ and $f_2: M_{21} \to M_{22}$ in $\mathbf{Mod}(\Sigma_2)$ such that $f_1|_{\sigma_1} = f_2|_{\sigma_2}$, there exists a unique model morphism $f': M_1' \to M_2'$ in $\mathbf{Mod}(\Sigma')$ such that $f'|_{\sigma_1'} = f_1$ and $f'|_{\sigma_2'} = f_2$ (we call such f' the *amalgamation* of f_1 and f_2).

The institution **INS** *has the amalgamation property* if all pushouts in **Sign** exist and every pushout diagram in **Sign** admits amalgamation. □

Exercise 4.4.13. Show that if a diagram as in Definition 4.4.12 admits amalgamation and is commutative then all models and morphisms in $\mathbf{Mod}(\Sigma')$ are amalgamations of pairs of (compatible) models and morphisms from $\mathbf{Mod}(\Sigma_1)$ and $\mathbf{Mod}(\Sigma_2)$, respectively. □

Lemma 4.4.14 (Amalgamation Lemma). *Any semi-exact institution has the amalgamation property.* □

The proof of the Amalgamation Lemma is based on the construction of pullbacks in **Cat**, cf. Exercise 3.4.32; see also Exercise 3.4.34, which is the same result in the standard algebraic framework. Note that the opposite implication also holds, so semi-exactness is equivalent to the amalgamation property.

Clearly, every exact institution is finitely exact, and every finitely exact institution is semi-exact. However, the last property is strictly weaker: for example, the institution **SSEQ** of single-sorted equational logic is semi-exact, but not finitely exact (see Exercise 4.4.10). In semi-exact institutions coproducts of signatures need not exist, or if they exist, need not be preserved by the model functor. However, if signature coproducts exist, the colimits for a large interesting class of signature diagrams (exist and) are preserved:

Proposition 4.4.15. *In any semi-exact institution, if the category of signatures has an initial object then it is finitely cocomplete and the model functor maps colimits of all finite non-empty connected diagrams of signatures to limits in* **Cat***.*

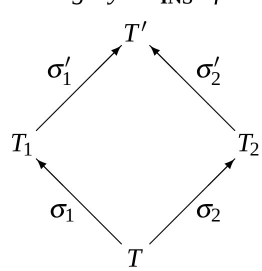
Proof sketch. The first part (existence of colimits of finite signature diagrams) follows as usual, by dualising Exercise 3.2.48; the second part (preservation of limits of finite non-empty connected signature diagrams) follows by Exercise 3.4.55. □

Exercise 4.4.16. Define institutions: **SSFOPEQ** of single-sorted first-order predicate logic with equality, **SSPFOPEQ** of single-sorted partial first-order predicate logic with equality, **SSCEQ** of single-sorted equational logic for continuous algebras, etc. Check that all of these institutions have cocomplete categories of signatures and are semi-exact. However, check that their model functors do not map coproducts of their signatures to products of the corresponding model categories, so these institutions are not (finitely) exact. □

Exercise 4.4.17. Let **INS** be a (finitely) exact institution. Recall that there is a functor $\mathbf{Mod_{Th}}\colon \mathbf{Th}_{\mathbf{INS}}^{op} \to \mathbf{Cat}$ mapping theories to their model categories and theory morphisms to the corresponding reduct functors (cf. Exercise 4.2.26). Prove that $\mathbf{Mod_{Th}}$ preserves (finite) limits.

HINT: First use the satisfaction condition for **INS** and the Amalgamation Lemma for signatures (Lemma 4.4.14) to prove the following generalisation of the Amalgamation Lemma:

Lemma (Amalgamation Lemma for theories). *Let* **INS** *be a semi-exact institution. Consider a pushout in the category* $\mathbf{Th_{INS}}$ *of theories:*

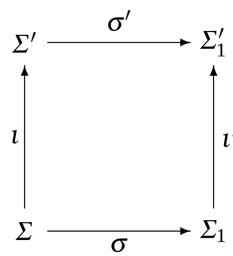


Then, for any two models $M_1 \in Mod[T_1]$ and $M_2 \in Mod[T_2]$ such that $M_1|_{\sigma_1} = M_2|_{\sigma_2}$, there exists a unique model $M' \in Mod[T']$ such that $M'|_{\sigma_1'} = M_1$ and $M'|_{\sigma_2'} = M_2$, and similarly for morphisms.

To complete the proof that $\mathbf{Mod_{Th}}$ is finitely continuous, by Exercise 3.2.48 it is enough to consider the initial theory and its category of models. To show that it is continuous, by Exercise 3.4.23 it is enough to consider coproducts of arbitrary families of theories and their categories of models. □

The trouble with **FPL** and with other institutions based on derived signature morphisms (see Exercise 4.1.23) is more severe than with single-sorted institutions: they are not semi-exact since not all pushouts exist in their signature categories, see Exercise 3.2.54. This motivates the following relaxation of semi-exactness, which is important for applications later on.

Definition 4.4.18 (I-semi-exact institution). For any institution **INS**, we say that a collection **I** of signature morphisms in **INS** is *closed under pushouts* if **I** contains all the identities, is closed under composition (so that **I** is a wide subcategory of $\mathbf{Sign}_{\mathbf{INS}}$) and for any signature morphism $\sigma\colon\Sigma \to \Sigma_1$ and "**I**-extension of Σ" $\iota\colon\Sigma \to \Sigma'$ in **I**, there is a pushout in **Sign**



such that $\iota' \in \mathbf{I}$.

Moreover, if all such pushouts with $\iota, \iota' \in \mathbf{I}$ admit amalgamation (i.e., the model functor maps them to pullbacks in **Cat**) we say that **INS** is *semi-exact w.r.t.* **I** (or **I**-*semi-exact*). □

Exercise 4.4.19. As mentioned above, institutions with derived signature morphisms do not have cocomplete signature categories. Check, however, that for example the institution $\mathbf{GEQ}^{der}$ is semi-exact w.r.t. the class of all inclusions (where inclusions are derived signature morphisms that map any *n*-ary operation name f to the term $f(\boxed{1},\ldots,\boxed{n})$, cf. Definition 1.5.14). Similarly, check that $\mathbf{GEQ}^{der}$ is semi-exact w.r.t. the class of inclusions that introduce only new constants. (Notice that in general an institution may be **I**-semi-exact without being $\mathbf{I}'$-semi-exact for some $\mathbf{I}' \subseteq \mathbf{I}$.)

For **FPL**, consider the class $\mathbf{I}_{\mathbf{FPL}}$ of signature morphisms $\delta\colon \mathsf{SIG} \to \mathsf{SIG}'$ that are injective renamings of sort and operation names such that no new value constructors are added for "old" sorts (i.e. sorts in $\delta(\mathsf{SIG})$). Show that **FPL** is $\mathbf{I}_{\mathbf{FPL}}$-semi-exact. Notice that both parts of the assumption on these morphisms are essential. Give an example of a non-injective renaming that does not have a pushout with another **FPL** signature morphism. Give an example of an injective renaming that adds value constructors for an old sort and does not have a pushout with another **FPL** signature morphism. Finally, give an example of a pushout in the the category of **FPL**-signatures that is not mapped by the **FPL**-model functor to a pullback in **Cat**. HINT: Consider two morphisms that add a new sort and a new unary value constructor for a previously unconstrained sort, with the new sort as its argument sort. □

Exercise 4.4.20. To complete the formal picture, note that the category of theories in **FPL** is cocomplete even though its category of signatures is not. Discuss why this is not useful for combining models over different signatures. HINT: Consider a simple signature with one sort and one binary operation, and two morphisms which map this operation to the projections on the first and second argument respectively. Then these two morphisms do not have a coequaliser in $\mathbf{Sign_{FPL}}$ while in $\mathbf{Th_{FPL}}$ their coequaliser is obtained by adding an equation to assert that the two projections coincide. □

We have introduced and studied amalgamation, exactness and semi-exactness as purely technical properties of institutions. However, as hinted at by Example 4.4.5 and the examples it builds on, amalgamation, and hence semi-exactness and exactness, provide a fundamental tool for combining models over different signatures. The point is easiest to see in institutions with standard signatures, like **FOPEQ** or **EQ**, when all the morphisms are inclusions. In that case, generalising the simple example of natural numbers and their extensions by the Fibonacci function and multiplication in Example 3.2.35, given signatures Σ_1 and Σ_2 with $\Sigma = \Sigma_1 \cap \Sigma_2$, we get $\Sigma' = \Sigma_1 \cup \Sigma_2$ as the pushout signature. Now, the amalgamation property ensures that, given a Σ_1-model M_1 and a Σ_2-model M_2 which give the same interpretation to all of the common symbols (in Σ), we can put them together in the obvious way (generalising Example 4.4.5) to interpret all of the symbols in the combined signature Σ'. In the institutional context, this intuition applies as well, but the sharing requirement is expressed by insisting on a common reduct along the indicated signature morphisms, and the combined signature is obtained using the pushout.

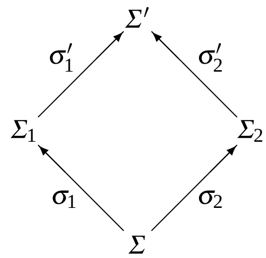
4.4.1 Abstract model theory

One of the ideas behind the definition of institution is that it is important to indicate over which signature one is working. In classical logic, there are a number of theorems in which the signature (or *language*, as logicians would say) over which formulae are constructed must be considered. Here is an example (for this, and for a classical formulation of the Robinson consistency theorem mentioned below, see e.g. [CK90]):

Theorem (Craig interpolation theorem). *In first-order logic, for any two formulae φ_1 and φ_2, if $\varphi_1 \models \varphi_2$ then there exists a formula θ using only the common symbols of φ_1 and φ_2 — that is, those symbols that occur in both formulae — such that $\varphi_1 \models \theta$ and $\theta \models \varphi_2$.* □

In our view, this standard formulation is not very elegant: referring to “the common symbols of φ_1 and φ_2” feels rather clumsy, even though it is easy enough to make it precise in the case of first-order logic. In the institutional framework this can be expressed in a more general and abstract way using colimits in the category of signatures.

Definition 4.4.21 (Craig interpolation property). Let **INS** be an institution with a finitely cocomplete category **Sign** of signatures. **INS** satisfies the *Craig interpolation property* if for any pushout

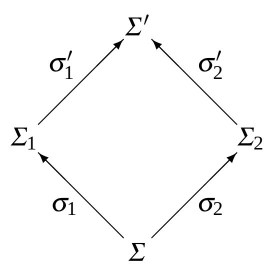


in **Sign**, and for any Σ_1-sentence $\varphi_1 \in \mathbf{Sen}(\Sigma_1)$ and Σ_2-sentence $\varphi_2 \in \mathbf{Sen}(\Sigma_2)$, if $\sigma_1'(\varphi_1) \models_{\Sigma'} \sigma_2'(\varphi_2)$ then there exists a Σ-sentence $\theta \in \mathbf{Sen}(\Sigma)$ (called an *interpolant* for φ_1 and φ_2) such that $\varphi_1 \models_{\Sigma_1} \sigma_1(\theta)$ and $\sigma_2(\theta) \models_{\Sigma_2} \varphi_2$. □

Not only has "the common symbols of φ_1 and φ_2" been captured by the simple categorical concept of a pushout here, but we were also forced to identify the signatures over which the individual consequence relations are considered. In our view, this is a much improved statement of the Craig interpolation property! Not only does it seem more clear (of course, any comparison should be made with a fully formal statement of the Craig interpolation theorem in the classical framework, not with the presentation given above), it is also more abstract and may be used for any logical system formalised as an institution, not just for first-order logic.

Here is another example, which states that consistent extensions of a complete theory (cf. Definition 4.2.27) combine safely:

Definition 4.4.22 (Robinson consistency property). Let **INS** be an institution with a finitely cocomplete category **Sign** of signatures. **INS** satisfies the *Robinson consistency property* if for any pushout



in **Sign**, and for any complete Σ-theory $T = \langle \Sigma, \Phi \rangle$ and consistent theories $T_1 = \langle \Sigma_1, \Phi_1 \rangle$ and $T_2 = \langle \Sigma_2, \Phi_2 \rangle$ such that $\sigma_1 \colon T \to T_1$ and $\sigma_2 \colon T \to T_2$ are theory morphisms, the Σ'-presentation $\langle \Sigma', \sigma_1'(\Phi_1) \cup \sigma_2'(\Phi_2) \rangle$ is consistent. □

Exercise 4.4.23. Adapt any standard proof of the Craig interpolation theorem to show that **FOPEQ** has the Craig interpolation property for those pushouts where at least one of σ_1 or σ_2 is injective on sorts. Construct a counterexample which shows that the proof must break down if neither σ_1 nor σ_2 is injective on sort names (injectivity on operation and predicate names does not have to be required). HINT: See [Bor05].

Show also that the Craig interpolation theorem for **FOPEQ** implies the analogous result for some of the subinstitutions of **FOPEQ** (see Exercise 4.1.13), for instance for **FOEQ**. Note though that your argument will not work for **FOP**, first-order predicate logic without equality — in fact, Craig interpolation may fail in **FOP** when one of the morphisms involved is non-injective on operation names, even if all the morphisms are injective on sort names. Of course, the standard proofs of Craig interpolation easily adapt to **FOP** when the morphisms involved are injective (on sort names as well as on operation names). □

It is well known that equational logic does not have the interpolation property:

Counterexample 4.4.24. In **EQ**, consider the signature Σ with three sorts s, s_1 and s_2, and two constants $a,b{:}s$. Let Σ_1 and Σ_2 extend Σ by a constant $e{:}s_1$ and by a unary operation $f{:}s_1 \to s_2$ respectively. Let Σ' be the union of Σ_1 and Σ_2 (this is the pushout signature for the two signature inclusions). Consider the sentences $\forall x{:}s_2 \bullet a = b \in \mathbf{Sen}_{\mathbf{EQ}}(\Sigma_1)$ and $a = b \in \mathbf{Sen}_{\mathbf{EQ}}(\Sigma_2)$. Clearly, over Σ' we have $\forall x{:}s_2 \bullet a = b \models a = b$ (since all Σ'-algebras have non-empty carriers for all sorts).

Suppose that we have an interpolant $\theta \in \mathbf{Sen}_{\mathbf{EQ}}(\Sigma)$ for $\forall x{:}s_2 \bullet a = b$ and $a = b$, so that $\forall x{:}s_2 \bullet a = b \models \theta$ over Σ_1 and $\theta \models a = b$ over Σ_2. Consider a Σ_1-algebra A_1 with the carrier of sort s_2 empty and with $a_{A_1} \neq b_{A_1}$. Clearly, $A_1 \models_{\Sigma_1} \forall x{:}s_2 \bullet a = b$, and so also $A_1 \models_{\Sigma_1} \theta$. Hence, $A_1|_\Sigma \models_\Sigma \theta$. Take a subalgebra of $A_1|_\Sigma$ with the empty carrier of sort s_1, which satisfies θ, and consider its expansion A_2 to a Σ_2-algebra. Then $A_2 \models_{\Sigma_2} \theta$ but $A_2 \not\models_{\Sigma_2} a = b$. Contradiction. □

Exercise 4.4.25. It is often stated that equational logic has interpolation (at least for pushouts w.r.t. injective signature morphisms) if one admits a *set of interpolants*, rather than just a single interpolant sentence θ as in Definition 4.4.21. Spell out this property following Definition 4.4.21, but using a set of sentences $\Theta \subseteq \mathbf{Sen}(\Sigma)$ in place of a single sentence $\theta \in \mathbf{Sen}(\Sigma)$. It also makes sense then to replace the single sentence $\varphi_1 \in \mathbf{Sen}(\Sigma_1)$ by a set $\Phi_1 \subseteq \mathbf{Sen}(\Sigma_1)$.

Unfortunately, equational logic has this property only if we restrict attention to algebras with non-empty carriers for all sorts. Carry out the proof for this case assuming that the signature morphisms considered are injective (HINT: see [Rod91]) and note where the assumption that the carriers are non-empty is important. Give a counterexample which shows that in general no single interpolant can be sufficient here. Extend this proof to the case where only one of the signature morphisms is injective on sorts (HINT: see [RG00], [PŞR09]).

Check that Counterexample 4.4.24 shows that the institution **EQ** of equational logic (with models that admit empty carriers) does not have the interpolation property, not even when sets of interpolants are allowed (and the morphisms involved are signature inclusions).

Go through other examples of institutions in Section 4.1.1 and check which of them have the interpolation property, either with a single interpolant, or with a set of interpolants (at least for pushouts involving signature inclusions, where this notion makes sense). □

Of course, we cannot expect to be able to prove that either the Craig interpolation or Robinson consistency properties are satisfied by an arbitrary institution

— they simply do not hold for some logics. However, one may attempt to identify other conditions on the underlying institution which imply the two properties. Along these lines, under some further technical assumptions, the two properties are equivalent: an institution satisfying certain technical assumptions satisfies the Craig interpolation property if and only if it satisfies the Robinson consistency property. This reflects what is well-known in classical model theory, where the two properties are indeed derivable from one another.

4.4.2 Free variables and quantification

In logic, formulae may contain free variables; such formulae are called *open*, as opposed to *closed* formulae which have no free variables. To interpret an open formula, one needs 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 in 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 use them to abstractly express the basic notion of behavioural equivalence in Section 8.5.3, see Exercise 8.5.61.

Fortunately we do not have to change the notion of an institution to cope with free variables — we can provide open formulae in the present framework. Note that we use here the term "formula" rather than "sentence", which is reserved for the sentences of the underlying institution, corresponding to closed formulae.

Consider the institution **GEQ** of ground equational logic (Example 4.1.3). Let $\Sigma = \langle S, \Omega \rangle$ be an algebraic signature. For any S-indexed family of sets, $X = \langle X_s \rangle_{s \in S}$, define $\Sigma(X)$ to be the extension of Σ by the elements of X as new constants of the appropriate sorts. 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 to first fix a valuation of variables X into $|A|$. Such a valuation corresponds exactly to an expansion of A to a $\Sigma(X)$-algebra.

Given a translation of sentences along an algebraic signature morphism $\sigma : \Sigma \to \Sigma'$ we can extend it to a translation of open formulae: 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 any semi-exact institution $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$.

Definition 4.4.26 (Open formula). Let $\Sigma \in |\mathbf{Sign}|$ be a signature in **INS**. Any pair $\langle \varphi, \theta \rangle$, where $\theta : \Sigma \to \Sigma'$ is a signature morphism and $\varphi \in \mathbf{Sen}(\Sigma')$, is an *open Σ-formula* with variables "$\Sigma' \setminus \theta(\Sigma)$". For any Σ-model $M \in |\mathbf{Mod}(\Sigma)|$, a *valuation* of variables "$\Sigma' \setminus \theta(\Sigma)$" into M is a Σ'-model $M' \in |\mathbf{Mod}(\Sigma')|$ which is a θ-expansion of M, i.e., such that $M'|_\theta = M$. We say that $\langle \varphi, \theta \rangle$ *holds in M under valuation M'* iff

$M' \models_{\Sigma'} \varphi$. If $\sigma{:}\Sigma \to \Sigma_1$ is a signature morphism then we define the translation of $\langle \varphi, \theta \rangle$ along σ as $\langle \sigma'(\varphi), \theta' \rangle$, where

$$\begin{array}{ccc} \Sigma' & \xrightarrow{\sigma'} & \Sigma'_1 \\ \uparrow{\scriptstyle\theta} & & \uparrow{\scriptstyle\theta'} \\ \Sigma & \xrightarrow[\sigma]{} & \Sigma_1 \end{array}$$

is a pushout in **Sign**. □

Note the quotation marks around the "set of variables" $\Sigma' \setminus \theta(\Sigma)$ in the above definition: since $\Sigma' \setminus \theta(\Sigma)$ makes no sense in an arbitrary institution, it is only meaningful as an aid to our intuition.

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. For example, in **GEQ** if a signature morphism $\theta{:}\Sigma \to \Sigma'$ is not injective then some Σ-models have no θ-expansion.

There is a rather subtle problem with the above definition: pushouts are defined only up to isomorphism, so strictly speaking the translation of open formulae is not well-defined. The following exercise shows that (at least for semantic analysis) an arbitrary pushout may be selected and so we may safely accept the above definition of translation.

Exercise 4.4.27. Consider an isomorphism $\iota{:}\Sigma'_1 \to \Sigma''_1$ in **Sign**, with inverse ι^{-1}. Since functors preserve isomorphisms, $\mathbf{Sen}(\iota){:}\mathbf{Sen}(\Sigma'_1) \to \mathbf{Sen}(\Sigma''_1)$ is a bijection and $\mathbf{Mod}(\iota){:}\mathbf{Mod}(\Sigma''_1) \to \mathbf{Mod}(\Sigma'_1)$ is an isomorphism in **Cat**. Show that moreover, for any $\psi \in \mathbf{Sen}(\Sigma'_1)$ and $M'_1 \in |\mathbf{Mod}(\Sigma'_1)|$, $M'_1 \models_{\Sigma'_1} \psi \iff M'_1|_{\iota^{-1}} \models_{\Sigma''_1} \iota(\psi)$. □

Sometimes we want to restrict the class of signature morphisms that may be used to construct open formulae. In fact, in the above remarks sketching how free variables may be introduced into **GEQ** we used just algebraic signature inclusions $\iota{:}\Sigma \hookrightarrow \Sigma'$ 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 **I** of signature morphisms that is closed under pushouts (see Definition 4.4.18).

Examples of such collections **I** in **AlgSig** include: the collection of all algebraic signature inclusions, the restriction of this to inclusions $\theta{:}\Sigma \hookrightarrow \Sigma'$ 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 surjective on sorts, the collection of all identities, and the collection of all morphisms. Note that most of these permit variables denoting operations or even sorts.

4.4.2.1 Universal quantification

In the rest of this section we briefly sketch how to universally close the open formulae introduced above.

Let **I** be a collection of signature morphisms that is closed under pushouts. Let Σ be a signature and let $\langle \varphi, \theta \rangle$ be an open Σ-formula such that $\theta \in \mathbf{I}$. Consider the universal closure of $\langle \varphi, \theta \rangle$, written $\forall \theta \bullet \varphi$, as a new Σ-sentence. The satisfaction relation and the translation of a sentence $\forall \theta \bullet \varphi$ along a signature morphism are defined in the expected way:

- A Σ-model satisfies the Σ-sentence $\forall \theta \bullet \varphi$ if $\langle \varphi, \theta \rangle$ holds in this model under any valuation of the variables "$\Sigma' \setminus \theta(\Sigma)$", that is, for any $M \in |\mathbf{Mod}(\Sigma)|$, $M \models_\Sigma \forall \theta \bullet \varphi$ if for all $M' \in |\mathbf{Mod}(\Sigma')|$ such that $M'|_\theta = M$, $M' \models_{\Sigma'} \varphi$.
- For any signature morphism $\sigma\colon \Sigma \to \Sigma_1$, $\sigma(\forall \theta \bullet \varphi)$ is $\forall \theta' \bullet \sigma'(\varphi)$, where

$$\begin{array}{ccc} \Sigma' & \xrightarrow{\sigma'} & \Sigma'_1 \\ \uparrow{\scriptstyle \theta} & & \uparrow{\scriptstyle \theta'} \\ \Sigma & \xrightarrow[\sigma]{} & \Sigma_1 \end{array}$$

  is a pushout in **Sign** such that $\theta' \in \mathbf{I}$.

Note that in the above we have extended our underlying institution **INS**. Formally:

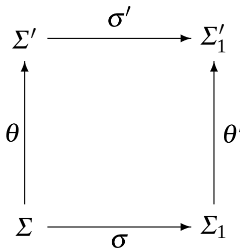
Definition 4.4.28 (Institution with universally closed formulae). Let **INS** be an institution, and let **I** be a collection of signature morphisms in **INS** that is closed under pushouts such that **INS** is **I**-semi-exact. The *extension of* **INS** *by universal closure w.r.t.* **I** is the following institution $\mathbf{INS}^{\forall(\mathbf{I})}$:

- $\mathbf{Sign}_{\mathbf{INS}^{\forall(\mathbf{I})}}$ is $\mathbf{Sign}_{\mathbf{INS}}$.
- For any signature Σ, $\mathbf{Sen}_{\mathbf{INS}^{\forall(\mathbf{I})}}(\Sigma)$ is the disjoint union of $\mathbf{Sen}_{\mathbf{INS}}(\Sigma)$ with the collection[20] of all universal closures $\forall \theta \bullet \varphi$ of open Σ-formulae, where $\theta \in \mathbf{I}$; for any signature morphism $\sigma\colon \Sigma \to \Sigma_1$, $\mathbf{Sen}_{\mathbf{INS}^{\forall(\mathbf{I})}}(\sigma)$ is the function induced by $\mathbf{Sen}_{\mathbf{INS}}(\sigma)$ on $\mathbf{Sen}_{\mathbf{INS}}(\Sigma)$ and by the notion of translation defined above on universally closed open Σ-formulae.
- $\mathbf{Mod}_{\mathbf{INS}^{\forall(\mathbf{I})}}$ is $\mathbf{Mod}_{\mathbf{INS}}$.
- The satisfaction relation in $\mathbf{INS}^{\forall(\mathbf{I})}$ 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 $\mathbf{INS}^{\forall(\mathbf{I})}$ is in fact an institution, modulo the above remark about the definition of the translation of open formulae.

[20] As usual, we disregard here the foundational problems which may arise if **I** is not a set.

Theorem 4.4.29 (Satisfaction condition for $\mathbf{INS}^{\forall(\mathbf{I})}$). *Let* **INS** *and* **I** *be as in Definition 4.4.28. For any signature morphism $\sigma\colon\Sigma \to \Sigma_1$, open Σ-formula $\langle\varphi,\theta\rangle$ (where $\theta \in \mathbf{I}$), Σ_1-model $M_1 \in |\mathbf{Mod}(\Sigma_1)|$, and pushout*



in **Sign** *such that $\theta' \in \mathbf{I}$,*

$$M_1|_\sigma \models_\Sigma \forall\theta\bullet\varphi \quad \textit{iff} \quad M_1 \models_{\Sigma_1} \forall\theta'\bullet\sigma'(\varphi)$$

Proof.

$(\Rightarrow)$: Assume that $M_1|_\sigma \models_\Sigma \forall\theta\bullet\varphi$ and let M_1' be a θ'-expansion of M_1. Put $M' = M_1'|_{\sigma'}$. Obviously, $M'|_\theta = M_1'|_{\theta;\sigma'} = M_1'|_{\sigma;\theta'} = M_1|_\sigma$. Thus, since $M_1|_\sigma \models_\Sigma \forall\theta\bullet\varphi$, $M' \models_{\Sigma'} \varphi$. Hence, by the satisfaction condition of **INS**, $M_1' \models_{\Sigma_1'} \sigma'(\varphi)$, which proves $M_1 \models_{\Sigma_1} \forall\theta'\bullet\sigma'(\varphi)$.

$(\Leftarrow)$: Assume that $M_1 \models_{\Sigma_1} \forall\theta'\bullet\sigma'(\varphi)$ and let M' be a θ-expansion of $M_1|_\sigma$. Since **INS** is **I**-semi-exact, there exists a θ'-expansion M_1' of M_1 such that $M_1'|_{\sigma'} = M'$. Then, since $M_1 \models_{\Sigma_1} \forall\theta'\bullet\sigma'(\varphi)$, $M_1' \models_{\Sigma_1'} \sigma'(\varphi)$. Thus, by the satisfaction condition, $M' \models_{\Sigma'} \varphi$, which proves $M_1|_\sigma \models_\Sigma \forall\theta\bullet\varphi$. □

Example 4.4.30. Let **I** be the collection of algebraic signature inclusions $\iota\colon\Sigma \hookrightarrow \Sigma'$ in **AlgSig** such that $\Sigma' \setminus \Sigma$ contains new constants only. The institution $\mathbf{GEQ}^{\forall(\mathbf{I})}$ essentially coincides with the institution **EQ** of equational logic (modulo the details of the notation used for sentences), as suggested already in Exercise 2.1.6. If $\Sigma' \setminus \Sigma$ is allowed to contain new operation names (not just constants), then quantification along morphisms in **I** leads to a version of second-order logic. □

Other quantifiers (there exists, there exists a unique, there exist infinitely many, for almost all, ...) may be introduced in the same manner as we have just introduced universal quantifiers. Example 4.1.41 illustrates how one may introduce logical connectives. By iterating these constructions one can, for example, derive the institution of first-order logic from the institution of ground atomic formulae.

4.5 Institutions with reachability structure

An alternative to the standard initial algebra approach to specifications is to take the reachable semantics of presentations, as discussed in Section 2.7.2, where from among all the algebras satisfying a presentation only the *reachable* algebras are selected. In Section 4.3 we argued that it is important to consider not just initial algebras, but more generally, algebras that are free extensions of a specified part; similarly, it is important here to consider not just reachable algebras, but more generally, algebras that are generated by some specified part. Given an algebraic signature Σ and a subsignature $\Sigma' \subseteq \Sigma$, a Σ-algebra A is *reachable from* Σ' if it has no proper subalgebra with the same Σ'-reduct. (**Exercise:** Show that this is the same as to require that the algebra is generated by the set of all its elements in the carriers of the sorts in Σ', as defined in Exercise 1.2.6.) To generalise this notion to the framework of an arbitrary institution we will proceed along the lines suggested by the "categorical theory of reachability" presented in Section 3.3 based on factorisation systems.

Definition 4.5.1 (Reachable model). Let $\langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ be an institution. Assume that for each signature $\Sigma \in |\mathbf{Sign}|$, we have a factorisation system $\langle \mathbf{E}_\Sigma, \mathbf{M}_\Sigma \rangle$ for the category $\mathbf{Mod}(\Sigma)$ of Σ-models.

Let $\sigma : \Sigma' \to \Sigma$ be a signature morphism. A Σ-model $M \in |\mathbf{Mod}(\Sigma)|$ is σ-*reachable* if M has no proper submodel with an isomorphic σ-reduct, that is, if any factorisation monomorphism $m : N \to M$ in $\mathbf{M}_\Sigma$ such that $m|_\sigma$ is an isomorphism in $\mathbf{Mod}(\Sigma')$ is in fact an isomorphism in $\mathbf{Mod}(\Sigma)$. □

Example 4.5.2. Recall that for any algebraic signature $\Sigma \in \mathbf{AlgSig}$, the categories $\mathbf{Alg}(\Sigma)$, $\mathbf{PAlg}(\Sigma)$ and $\mathbf{CAlg}(\Sigma)$ of total, partial and continuous algebras come equipped with factorisation systems (Examples 3.3.3, 3.3.13 and 3.3.14, respectively). Hence, the above definition makes sense in the institutions **EQ** of equational logic, **PEQ** of partial equational logic and **CEQ** of equational logic for continuous algebras, yielding the expected notions. □

Exercise 4.5.3. Recall that by Definition 3.3.7 a Σ-model is reachable if it has no proper submodel. Show that if **INS** is finitely exact then reachability is a special case of σ-reachability as defined above. (HINT: Use the fact that there is an initial signature with the singleton category **1** of models.) □

In Section 3.3 it was shown how the notion of reachability introduced there may be related to an equivalent definition stated in terms of quotients of initial models (Theorem 3.3.8(1)). In the standard algebraic case, an algebra is reachable if and only if it is isomorphic to a quotient of the algebra of ground terms (Exercise 1.4.14). To give an analogous result for σ-reachability we have to be able to build terms over a specified reduct of the given algebra (cf. Exercise 3.5.11). Given such a construction, a Σ-algebra A is reachable from $\Sigma' \subseteq \Sigma$ if and only if evaluation in A of Σ-terms over the Σ'-reduct of A is surjective, or equivalently, if A is a natural quotient of the algebra of Σ-terms built over $A|_{\Sigma'}$. We introduce a generalisation of the construction of term algebras to an arbitrary institution by requiring that reduct functors induced by signature morphisms have left adjoints. Notice that only signatures are involved in this definition, no sentences, and so this requirement indeed corresponds to the mild assumption that free models (term algebras) may be built along arbitrary signature morphisms.

Definition 4.5.4 (Institution with reachability structure). An *institution with reachability structure* is an institution $\langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ together with:

- for each signature $\Sigma \in |\mathbf{Sign}|$, a factorisation system $\langle \mathbf{E}_\Sigma, \mathbf{M}_\Sigma \rangle$ for the category $\mathbf{Mod}(\Sigma)$ of Σ-models; and
- for each signature morphism $\sigma : \Sigma' \to \Sigma$, a *$\sigma$-free functor* $\mathbf{F}_\sigma : \mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$ which is left adjoint to the σ-reduct functor $\_|_\sigma : \mathbf{Mod}(\Sigma) \to \mathbf{Mod}(\Sigma')$ with unit $\eta^\sigma : \mathbf{Id}_{\mathbf{Mod}(\Sigma')} \to \mathbf{F}_\sigma(\_)|_\sigma$.

(As usual, sub- and superscripts will be omitted when convenient.) □

Example 4.5.5. The institution **EQ** of equational logic equipped with factorisation systems for categories of algebras (cf. Example 3.3.3) has reachability structure — the free functors are given by Exercise 3.5.11. □

Exercise 4.5.6. Show that the institution **PEQ** of partial equational logic with the factorisation systems given by Example 3.3.13 for categories of partial algebras forms an institution with reachability structure. (HINT: Free functors are rather trivial here.)

Similarly, show that the institution **CEQ** of equational logic for continuous algebras with the factorisation systems given by Example 3.3.14 for categories of continuous algebras forms an institution with reachability structure. (HINT: The construction of free functors is much more difficult here — follow the construction for ordinary algebras in Exercise 3.5.11, but when defining the new operations in a free way remember that you have to extend the complete partial order to cover the new values as well, ensuring continuity of the operations.) □

Exercise 4.5.7. Let **INS** be a finitely exact institution. Prove that if every reduct functor in **INS** has a left adjoint, then for every signature Σ the category $\mathbf{Mod}_{\mathbf{INS}}(\Sigma)$ of Σ-models has an initial object. (HINT: Use the fact that there is an initial signature with the singleton category **1** of models.) □

The following theorem generalises well-known facts from the standard algebraic setting. Just like its "predecessor" Theorem 3.3.8, it confirms our confidence in the abstract definitions by showing how their different versions "click together" nicely.

Theorem 4.5.8. *Let* $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ *be an institution with reachability structure. Consider a signature morphism* $\sigma : \Sigma' \to \Sigma$.

1. *A Σ-model $M \in |\mathbf{Mod}(\Sigma)|$ is σ-reachable if and only if it is a natural quotient of the free object over its σ-reduct, that is, the counit morphism* $\varepsilon_M = (id_{M|_\sigma})^\# : \mathbf{F}_\sigma(M|_\sigma) \to M$ *belongs to* $\mathbf{E}_\Sigma$ *(cf. Exercise 3.5.24).*
2. *For any σ-reachable model $M \in |\mathbf{Mod}(\Sigma)|$, any model $N \in |\mathbf{Mod}(\Sigma)|$ and Σ'-model morphism $f' : M|_\sigma \to N|_\sigma$, there exists at most one Σ-model morphism $f : M \to N$ that extends f' (i.e., such that $f|_\sigma = f'$).*
3. *Every Σ-model has a unique (up to isomorphism) σ-reachable submodel with an isomorphic σ-reduct.*

4. *If $M \in |\mathbf{Mod}(\Sigma)|$ is σ-reachable then for any Σ-model morphism $f{:}N \to M$ such that $f|_\sigma$ is an isomorphism, f is a factorisation epimorphism (i.e., $f \in \mathbf{E}_\Sigma$).*

Proof.

1. $(\Rightarrow)$: Let $\mathbf{F}_\sigma(M|_\sigma) \xrightarrow{e} N \xrightarrow{m} M$ be a factorisation of $\varepsilon_M{:}\mathbf{F}_\sigma(M|_\sigma) \to M$. Arguing dually to Exercise 3.5.18 we can show that $m|_\sigma{:}N|_\sigma \to M|_\sigma$ is an isomorphism. Hence, by the σ-reachability of M, m is an isomorphism, which proves that $\varepsilon_M \in \mathbf{E}_\Sigma$.
   $(\Leftarrow)$: Let $m{:}N \to M$, $m \in \mathbf{M}_\Sigma$, with $m|_\sigma$ being an isomorphism. Define $f{:}\mathbf{F}_\sigma(M|_\sigma) \to N$ by $f = ((m|_\sigma)^{-1})^\#$. Then $\eta_{M|_\sigma};(f;m)|_\sigma = id_{M|_\sigma}$. By the freeness of $\mathbf{F}_\sigma(M|_\sigma)$, this implies that $f;m = \varepsilon_M$. Thus, by the assumption that $\varepsilon_M \in \mathbf{E}_\Sigma$ and by Exercise 3.3.5, m is an isomorphism.
2. Suppose that $f_1, f_2{:}M \to N$ are such that $f_1|_\sigma = f_2|_\sigma = f'$. Then $\eta_{M|_\sigma};(\varepsilon_M;f_1)|_\sigma = f' = \eta_{M|_\sigma};(\varepsilon_M;f_2)|_\sigma$, and so $\varepsilon_M;f_1 = \varepsilon_M;f_2$. Thus, we also have $f_1 = f_2$, since by (1) above ε_M is an epimorphism.
3. Consider an arbitrary Σ-model M. Let $\mathbf{F}_\sigma(M|_\sigma) \xrightarrow{e} N \xrightarrow{m} M$ be a factorisation of $\varepsilon_M{:}\mathbf{F}_\sigma(M|_\sigma) \to M$. Again, arguing dually to Exercise 3.5.18 we can show that $m|_\sigma{:}N|_\sigma \to M|_\sigma$ is an isomorphism. Moreover, by the naturality of ε, $\mathbf{F}_\sigma(m|_\sigma);\varepsilon_M = \varepsilon_N;m$, that is $\mathbf{F}_\sigma(m|_\sigma);e;m = \varepsilon_N;m$, and so (since m is a monomorphism) $\varepsilon_N = \mathbf{F}_\sigma(m|_\sigma);e \in \mathbf{E}_\Sigma$. Thus, by (1) again, N is a σ-reachable submodel of M.
   To prove uniqueness up to isomorphism, consider a subobject $m_1{:}N_1 \to M$ with $m_1|_\sigma$ being an isomorphism and $\varepsilon_{N_1}{:}F_\sigma(N_1|_\sigma) \to N_1$ in $\mathbf{E}_\Sigma$. Then $\mathbf{F}_\sigma(m_1|_\sigma);\varepsilon_M = \varepsilon_{N_1};m_1$, and since $\mathbf{F}_\sigma(m_1|_\sigma)$ is an isomorphism, we have two factorisations of $\varepsilon_M{:}\mathbf{F}_\sigma(M|_\sigma) \to M$, $\langle \mathbf{F}_\sigma(m_1|_\sigma)^{-1};\varepsilon_{N_1}, m_1\rangle$ and $\langle e, m\rangle$, which by the uniqueness of factorisations implies that N and N_1 are isomorphic.
4. Let $N \xrightarrow{e} \cdot \xrightarrow{m} M$ be a factorisation of $f{:}N \to M$. Then, by naturality of ε, $\varepsilon_N;e;m = \mathbf{F}_\sigma(f|_\sigma);\varepsilon_M$. Now, since $f|_\sigma$ (and hence $\mathbf{F}_\sigma(f|_\sigma)$) is an isomorphism, by σ-reachability of M and (1) above, $\varepsilon_N;e;m \in \mathbf{E}_\Sigma$. Thus, by Exercise 3.3.5, m is an isomorphism, and so $f \in \mathbf{E}_\Sigma$. □

4.5.1 The method of diagrams

In the standard algebraic framework, reachable algebras enjoy a number of useful properties which make them especially easy to deal with. As a consequence of the fact that we are able to “name” (using ground terms) all their elements, reachable algebras are easy to describe using the most elementary logical sentences, ground equations. To be more precise: for any algebraic signature Σ and reachable Σ-algebra A, the class

$$Ext(A) = \{B \in |\mathbf{Alg}(\Sigma)| \mid \text{there exists a } \Sigma\text{-homomorphism } h{:}A \to B\}$$

is definable by the ground Σ-equations that hold in A, that is, $Ext(A) = Mod_{\mathbf{GEQ}}(Th_{\mathbf{GEQ}}(\{A\}))$, and moreover, A is initial in $Ext(A)$. (We will refer to classes of algebras of the form $Ext(A)$ for a reachable algebra A as *ground varieties*.) This gives a one-to-one correspondence between ground equational theories and isomorphism classes of reachable algebras (and furthermore, congruences on ground term algebras by Exercise 1.4.14).

Unfortunately, not all algebras are reachable, and it is clear that this correspondence does not carry over to arbitrary algebras: there are algebras that cannot be characterised as initial models of equational theories. But there is a technical trick that may help: if a Σ-algebra A is not reachable, then consider the signature $\Sigma(A)$ obtained by adding to Σ the elements of $|A|$ as constants of the appropriate sorts. Now, the algebra A has an obvious expansion to a reachable $\Sigma(A)$-algebra $E(A)$, where the new constants are interpreted as the elements they correspond to. This expansion has a number of useful properties:

- Any Σ-homomorphism $h{:}A \to B$ determines unambiguously an expansion of B to a $\Sigma(A)$-algebra $E_h(B)$ where each new constant in $\Sigma(A)$ is interpreted as the value of h on the corresponding element of $|A|$. Moreover, this expansion is independent from any decomposition of h: for any Σ-homomorphisms $h_1{:}A \to C$ and $h_2{:}C \to B$ such that $h = h_1;h_2$, the homomorphism h_2 (or more precisely, its underlying map) is a $\Sigma(A)$-homomorphism from $E_{h_1}(C)$ to $E_h(B)$.
- Intuitively, the expansion does not introduce more structure than necessary to make A reachable; in particular, no new elements are added.

Putting all these together, any Σ-algebra A may be characterised by the set of ground equations on the signature $\Sigma(A)$ that hold in $E(A)$. This technique, known as *the method of diagrams*, is one of the basic tools of classical model theory (cf. e.g. [CK90]). We have already suggested its use in the construction of the free functor corresponding to a signature morphism in Exercise 3.5.11.

In the following the method of diagrams is formulated in the context of an arbitrary institution with reachability structure. We will assume that the institution is finitely exact in order to be able to deal with reachability (not just reachability relative to signature morphisms, cf. Exercises 4.5.3 and 4.5.7).

Definition 4.5.9 (The method of diagrams). Let $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ be a finitely exact institution with reachability structure. **INS** *admits the method of diagrams* if:

- (*Definability of ground varieties*)
  for every signature $\Sigma \in |\mathbf{Sign}|$ and reachable Σ-model $M \in |\mathbf{Mod}(\Sigma)|$, the class

  $$Ext(M) = \{N \in |\mathbf{Mod}(\Sigma)| \mid \text{there exists a } \Sigma\text{-model morphism } h{:}M \to N\}$$

  of extensions of M is definable, that is, $Ext(M) = Mod_\Sigma(\Phi)$ for some set $\Phi \subseteq \mathbf{Sen}(\Sigma)$.
- (*Existence of diagrams*)
  for every signature $\Sigma \in |\mathbf{Sign}|$ and Σ-model $M \in |\mathbf{Mod}(\Sigma)|$, there exists a signature $\Sigma(M) \in |\mathbf{Sign}|$ and signature morphism $\iota{:}\Sigma \to \Sigma(M)$ such that:

– M has a reachable ι-expansion $E(M)$: there exists $E(M)$ which is a reachable $\Sigma(M)$-model such that $E(M)|_\iota = M$;
– ι-reduct is an isomorphism of the slice categories $\mathbf{Mod}(\Sigma(M))\uparrow E(M)$ and $\mathbf{Mod}(\Sigma)\uparrow M$ (see Exercise 3.1.30), that is, for any Σ-model morphism $f: M \to N$, there exists a unique ι-expansion of N, $E_f(N)$, such that f has an ι-expansion $E(f): E(M) \to E_f(N)$ and such that any Σ-model morphism $h: N \to N_1$ has a unique ι-expansion $E(h): E_f(N) \to E_{f;h}(N_1)$; and
– ι-reduct preserves the factorisation system on $\mathbf{Mod}(\Sigma(M))\uparrow E(M)$ as inherited from $\mathbf{Mod}(\Sigma(M))$, that is, for any $f: E(M) \to N'$ and $h: N' \to N''$, if $h \in \mathbf{E}_{\Sigma(M)}$ then $h|_\iota \in \mathbf{E}_\Sigma$ and if $h \in \mathbf{M}_{\Sigma(M)}$ then $h|_\iota \in \mathbf{M}_\Sigma$.

Then, $\Sigma(M)$ is called the *diagram signature for M* (with *signature inclusion* ι), $E(M)$ is called the *diagram expansion of M*, and finally the theory $\Delta^+(M) = Th_{\Sigma(M)}(Ext(E(M)))$ is called the (*positive*) *diagram of M*. □

Example 4.5.10. The institutions **EQ** of equational logic, **PEQ** of partial equational logic, and **CEQ** of equational logic for continuous algebras admit the method of diagrams. Ground varieties in **EQ** are definable by sets of ground equations; ground varieties of **PEQ** are definable by sets of ground equations and ground definedness formulae; ground varieties in **CEQ** are definable by sets of ground infinitary equations. For any (total, partial, or continuous) Σ-algebra A, the diagram signature for A is formed by adding constants corresponding to all the elements of $|A|$. The diagram expansion of a partial algebra is formed by requiring that the new constants are defined and have the expected values. □

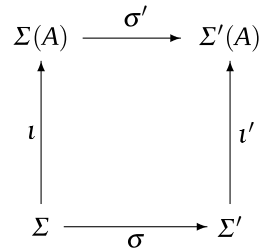
Exercise 4.5.11. Show that in any institution that admits the method of diagrams, and for any model M, the class of models of the positive diagram of M is the class of all extensions of the diagram expansion of M: $Mod_{\Sigma(M)}(\Delta^+(M)) = Ext(E(M))$. □

4.5.2 Abstract algebraic institutions

In Exercise 3.5.11 we suggested the use of the method of diagrams to prove that in the standard algebraic framework, the reduct functor induced by a signature morphism has a left adjoint. With some more effort, one can generalise this result and prove that in the standard equational institution the reduct functor induced by a *theory* morphism has a left adjoint:

Exercise 4.5.12. Prove that in the equational institution **EQ**, for any theory morphism $\sigma: T \to T'$, the reduct functor $\_|_\sigma: \mathbf{Mod}[T'] \to \mathbf{Mod}[T]$ has a left adjoint.

HINT: Formalise and complete the following construction: Let $T = \langle \Sigma, \Phi \rangle$ and $T' = \langle \Sigma', \Phi' \rangle$. For any Σ-algebra $A \in Mod[T]$, let $\Sigma(A)$ be its diagram signature, and let



be a pushout in the category of signatures. Then, let $\Delta^+(A) \subseteq \mathbf{Sen}_{\mathbf{EQ}}(\Sigma(A))$ be the positive diagram of A. Consider the presentation $\langle \Sigma'(A), \sigma'(\Delta^+(A)) \cup \iota'(\Phi') \rangle$. By Theorem 2.5.14, this has an initial model. Its ι'-reduct is a free object over A. (See also Exercise 3.5.11 for a slightly different line of reasoning.) □

We will come back to a careful, more abstract analysis of this construction later (cf. Theorem 4.5.18 below). For now, just notice that the construction not only uses the fact that the equational institution admits the method of diagrams, but also relies (directly or indirectly) on a number of simple facts about the reachability structure of the equational institution. We capture some of these additional properties in the following abstract definition:

Definition 4.5.13 (Abstract algebraic institution). An *abstract algebraic institution* is a finitely exact institution $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ with reachability structure that admits the method of diagrams, for which the following conditions hold:

- For any signature $\Sigma \in |\mathbf{Sign}|$, the category $\mathbf{Mod}(\Sigma)$ has all products (of sets of models) and is $\mathbf{E}_\Sigma$-co-well-powered (Definition 3.3.10).
- For any signature morphism $\sigma : \Sigma \to \Sigma'$, the σ-reduct functor preserves submodels (i.e., for all $m' \in \mathbf{M}_{\Sigma'}$, $m'|_\sigma \in \mathbf{M}_\Sigma$) and products.
- (*Abstraction condition*) For any signature Σ and Σ-models $M, N \in |\mathbf{Mod}(\Sigma)|$, if M and N are isomorphic then they satisfy exactly the same Σ-sentences. □

Example 4.5.14. The institutions **EQ** of equational logic, **PEQ** of partial equational logic, and **CEQ** of equational logic for continuous algebras are abstract algebraic institutions. □

Exercise 4.5.15. There is a certain asymmetry in the above definition: reduct functors in abstract algebraic institutions are required to preserve submodels but are not required to preserve quotients. Prove that in **EQ**, reduct functors preserve quotients as well: for all $\sigma : \Sigma \to \Sigma'$ and $e' \in \mathbf{E}_{\Sigma'}$, $e'|_\sigma \in \mathbf{E}_\Sigma$. Show, however, that this is not true in general in **PEQ**. □

4.5.3 Liberal abstract algebraic institutions

In Section 4.3 we have shown that it is possible to restrict attention to initial models of specifications written in an arbitrary institution, even if theories in the institution are not guaranteed to have initial models in general. Similarly, data constraints make sense in an arbitrary institution even if reduct functors induced by theory morphisms are not guaranteed to have left adjoints. This flexibility is useful, but nevertheless it may be important to know whether or not a theory used in an initiality constraint has an initial model, or whether a theory morphism used in a data constraint has a corresponding free functor. In some institutions this is always the case: the equational institution **EQ** is one example (cf. Theorem 2.5.14 and Exercise 4.5.12). In the rest of this section we present a characterisation of institutions that have this property. Of course, very little can be done in the framework of an arbitrary institution: however, abstract algebraic institutions as introduced above provide a sufficiently rich background.

Definition 4.5.16 (Liberal institution). An institution **INS** *admits initial models* if every theory in **INS** has an initial model. **INS** is *liberal* if for every theory morphism $\sigma : T \to T'$ in **INS**, the σ-reduct functor $\_|_{\sigma} : \mathbf{Mod}[T'] \to \mathbf{Mod}[T]$ has a left adjoint.

Then, an abstract algebraic institution **INS** *admits reachable initial models* if every theory in **INS** has an initial model which is reachable. **INS** is *strongly liberal* if for every theory morphism $\sigma : T \to T'$ in **INS**, the σ-reduct functor $\_|_{\sigma} : \mathbf{Mod}[T'] \to \mathbf{Mod}[T]$ has a left adjoint $\mathbf{F}_{\sigma} : \mathbf{Mod}[T] \to \mathbf{Mod}[T']$ such that for any $M \in Mod[T]$, $\mathbf{F}_{\sigma}(M) \in Mod[T']$ is σ-reachable. □

In the last part of the definition we have slightly abused notation by using σ as both a *theory* morphism and a *signature* morphism (which in fact it is). It is important that the notion of σ-reachability used here is taken w.r.t. signature morphisms (cf. Definition 4.5.1) without taking into account the theory context.

Exercise 4.5.17. Find an institution that admits initial models but does not admit reachable initial models. HINT: Consider an algebraic signature Σ with a unary operation symbol $f : s \to s$. Show that the class of Σ-algebras satisfying the axiom $\exists! x{:}s \bullet f(x) = x$ has an initial model which is not reachable, where $\exists!$ reads "there exists a unique", that is, $\exists! x{:}s \bullet f(x) = x$ stands for $\exists x{:}s \bullet f(x) = x \wedge \forall x_1, x_2{:}s \bullet f(x_1) = x_1 \wedge f(x_2) = x_2 \Rightarrow x_1 = x_2$. □

For abstract algebraic institutions, the requirements introduced in Definition 4.5.16 are pairwise equivalent.

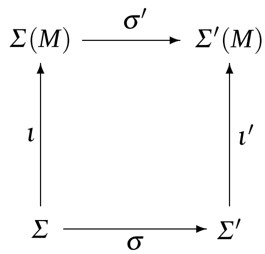
Theorem 4.5.18. *Let* **INS** *be an abstract algebraic institution.* **INS** *is liberal if and only if it admits initial models.*

Proof.

($\Rightarrow$): Let $T = \langle \Sigma, \Phi \rangle$ be a theory. Let $\iota_{\Sigma} : \Sigma_{\varnothing} \to \Sigma$ be the only signature morphism from the initial signature $\Sigma_{\varnothing}$ to Σ. Then $\iota_{\Sigma} : T_{\varnothing} \to T$ is a theory morphism, where $T_{\varnothing} = \langle \Sigma_{\varnothing}, Cl_{\Sigma_{\varnothing}}(\varnothing) \rangle$ is the initial theory, and so the reduct functor $\_|_{\iota_{\Sigma}} : \mathbf{Mod}[T] \to \mathbf{Mod}[T_{\varnothing}]$ has a left adjoint $\mathbf{F}_{\iota_{\Sigma}} : \mathbf{Mod}[T_{\varnothing}] \to \mathbf{Mod}[T]$. Now, there is exactly one $\Sigma_{\varnothing}$-model, say $M_{\varnothing} \in |\mathbf{Mod}[T_{\varnothing}]|$, and moreover, $\mathbf{F}_{\iota_{\Sigma}}(M_{\varnothing})$ is an initial model of T.

($\Leftarrow$): We follow the proof for the equational institution **EQ** sketched in Exercise 4.5.12. For any theory morphism $\sigma: T \to T'$, where $T = \langle \Sigma, \Phi \rangle$ and $T' = \langle \Sigma', \Phi' \rangle$, and model $M \in Mod[T]$, we construct a model $\mathbf{F}_\sigma(M) \in Mod[T']$ with unit $\eta_M: M \to \mathbf{F}_\sigma(M)|_\sigma$ that is free over M w.r.t. $\_|_\sigma: \mathbf{Mod}[T'] \to \mathbf{Mod}[T]$.

Let $\Sigma(M)$ be the diagram signature for M with signature inclusion $\iota: \Sigma \hookrightarrow \Sigma(M)$, and let



be a pushout in the category of signatures. Then, let $\Delta^+(M) \subseteq \mathbf{Sen}(\Sigma(M))$ be the positive diagram of M. Consider the presentation $\langle \Sigma'(M), \sigma'(\Delta^+(M)) \cup \iota'(\Phi') \rangle$. By the assumption, it has an initial model, say I. Put $\mathbf{F}_\sigma(M) = I|_{\iota'}$. Then, since by the satisfaction condition $I|_{\sigma'} \models_{\Sigma(M)} \Delta^+(M)$, $I|_{\sigma'} \in Ext(E(M))$ (cf. Exercise 4.5.11). Hence, there exists a (unique, since $E(M)$ is reachable) $\Sigma(M)$-model morphism $\widehat{\eta_M}: E(M) \to I|_{\sigma'}$. Put $\eta_M = \widehat{\eta_M}|_\iota: M \to \mathbf{F}_\sigma(M)|_\sigma$.

First, notice that since $I \models_{\Sigma'(M)} \iota'(\Phi')$, $\mathbf{F}_\sigma(M) \in Mod[T']$. Then, consider an arbitrary model $N \in Mod[T']$ and a Σ-model morphism $f: M \to N|_\sigma$.

By the definition of the diagram signature for M, $N|_\sigma$ has a unique ι-expansion to a $\Sigma(M)$-model $E_f(N|_\sigma)$ such that there exists a $\Sigma(M)$-model morphism $E(f): E(M) \to E_f(N|_\sigma)$ with $E(f)|_\iota = f$. Amalgamation yields a unique $\Sigma'(M)$-model $E^\sigma_f(N|_\sigma) \in |\mathbf{Mod}(\Sigma'(M))|$ with $E^\sigma_f(N|_\sigma)|_{\sigma'} = E_f(N|_\sigma)$ and $E^\sigma_f(N|_\sigma)|_{\iota'} = N$. Since $N \models_{\Sigma'} \Phi'$, $E^\sigma_f(N|_\sigma) \models_{\Sigma'(M)} \iota'(\Phi')$. Then, since $E_f(N|_\sigma) \in Ext(E(M))$, $E_f(N|_\sigma) \models_{\Sigma(M)} \Delta^+(M)$, and so $E^\sigma_f(N|_\sigma) \models_{\Sigma'(M)} \sigma'(\Delta^+(M))$. Consequently, we get a unique $\Sigma'(M)$-model morphism $\widehat{f'}: I \to E^\sigma_f(N|_\sigma)$. Put $f' = \widehat{f'}|_{\iota'}: \mathbf{F}_\sigma(M) \to N$. Notice that $\widehat{\eta_M};\widehat{f'}|_{\sigma'}: E(M) \to E_f(N|_\sigma)$. Hence, since $E(M)$ is reachable, $\widehat{\eta_M};\widehat{f'}|_{\sigma'} = E(f)$, and so we obtain $\eta_M;f'|_\sigma = f$. Moreover, f' is the only morphism with this property. To see this, suppose that for some $f'': \mathbf{F}_\sigma(M) \to N$, $\eta_M;f''|_\sigma = f$. Then, by the amalgamation property (this time for model morphisms) there exists a $\Sigma'(M)$-model morphism $\widehat{f''}: I \to E^\sigma_f(N|_\sigma)$ such that $\widehat{f''}|_{\iota'} = f''$ (and $\widehat{f''}|_{\sigma'} = E(f''|_\sigma): I|_{\sigma'} \to E_f(N|_\sigma)$). By initiality of I, $\widehat{f''} = \widehat{f'}$, and so $f'' = f'$, which completes the proof. □

Theorem 4.5.19. *Let* **INS** *be an abstract algebraic institution.* **INS** *is strongly liberal if and only if it admits reachable initial models.*

Proof. We extend the proof of the previous theorem, relying on the notation introduced there.

($\Rightarrow$): The only additional remark needed is that $\mathbf{F}_{\iota_\Sigma}(M_\varnothing)$ is reachable if it is ι_Σ-reachable (cf. Exercise 4.5.3).

($\Leftarrow$): We have to additionally prove that $\mathbf{F}_\sigma(M) = I|_{\iota'}$ is σ-reachable whenever I is reachable. To see this, consider an arbitrary submodel of $I|_{\iota'}$ with an isomorphic σ-reduct, say $m{:}N \to I|_{\iota'}$, where $m \in \mathbf{M}_{\Sigma'}$ and $m|_\sigma{:}N|_\sigma \to I|_{\sigma;\iota'}$ is an isomorphism. Put $f = \eta_M;(m|_\sigma)^{-1}{:}M \to N|_\sigma$. Then $f;m|_\sigma = \eta_M$, and so $m|_\sigma$ has an expansion to a $\Sigma(M)$-model morphism $E(m|_\sigma){:}E_f(N|_\sigma) \to E_{\eta_M}(I|_{\sigma;\iota'}) = I|_{\sigma'}$. Then, as in the corresponding part of the proof of Theorem 4.5.18, we get a unique $\Sigma'(M)$-model $E^\sigma_f(N|_\sigma) \in |\mathbf{Mod}(\Sigma'(M))|$ such that $E^\sigma_f(N|_\sigma)|_{\sigma'} = E_f(N|_\sigma)$ and $E^\sigma_f(N|_\sigma)|_{\iota'} = N$, and a $\Sigma'(M)$-model morphism $\widehat{f'}{:}I \to E^\sigma_f(N|_\sigma)$. On the other hand, by the amalgamation property again, there exists a unique $\Sigma'(M)$-model morphism $\widehat{m}{:}E^\sigma_f(N|_\sigma) \to I$ such that $\widehat{m}|_{\sigma'} = E(m|_\sigma)$ and $\widehat{m}|_{\iota'} = m$. By the initiality of I, $\widehat{f'};\widehat{m}$ is the identity, and so is $(\widehat{f'};\widehat{m})|_{\iota'} = \widehat{f'}|_{\iota'};m$. Thus, by Exercise 3.3.5, m is an isomorphism — which completes the proof. □

4.5.4 *Characterising abstract algebraic institutions that admit reachable initial models*

From the very beginning of work on algebraic specifications it has been known that the standard equational institution **EQ** admits reachable initial models (cf. Theorem 2.5.14). Moreover, the proof of this property generalises readily to the situation where conditional equations (even with infinite sets of premises) are permitted as axioms. On the other hand, Example 2.7.11 shows that if disjunction is permitted, the property is lost. Indeed, in the standard algebraic framework the infinitary conditional axioms, which define all non-empty quasi-varieties, form in some sense a borderline beyond which one cannot be sure of the existence of reachable initial models. We generalise this result to the framework of abstract algebraic institutions.

Theorem 4.5.20. *Let* **INS** *be an abstract algebraic institution.* **INS** *admits reachable initial models if and only if every class of models definable in* **INS** *is closed under products (of sets of models) and under submodels.*

Proof.

($\Leftarrow$): This follows directly by Lemma 3.3.12; just notice that any class of models closed under products and submodels is a *non-empty* quasi-variety (cf. Definition 3.3.11).

($\Rightarrow$): Let $\langle \Sigma, \Phi \rangle$ be a presentation in **INS**. We show the required closure properties of $Mod_\Sigma(\Phi)$.

(*Submodels*): Consider a model $M \in Mod_\Sigma(\Phi)$ and its submodel $m{:}N \to M$, $m \in \mathbf{M}_\Sigma$. Let $\Sigma(N)$ be a diagram signature for N with signature inclusion $\iota{:}\Sigma \to \Sigma(N)$, and let $\Delta^+(N) \subseteq \mathbf{Sen}(\Sigma(N))$ be the positive diagram of N. Recall that $Mod_{\Sigma(N)}(\Delta^+(N)) = Ext(E(N))$, where $E(N) \in \mathbf{Mod}(\Sigma(N))$ is the diagram expansion of N. The presentation $\langle\Sigma(N),\Delta^+(N)\cup\iota(\Phi)\rangle$ has a reachable initial model, say I. We show that $I|_\iota$ is isomorphic to N, which in particular implies $N\in Mod_\Sigma(\Phi)$.

Since $I\models_{\Sigma(N)}\Delta^+(N)$, there exists a $\Sigma(N)$-model morphism $f{:}E(N)\to I$. Moreover, since I is reachable, $f\in\mathbf{E}_{\Sigma(N)}$ (by Theorem 3.3.8(4)) and hence also $f|_\iota\in\mathbf{E}_\Sigma$. Then, let $E_m(M)$ be the unique expansion of M to a $\Sigma(N)$-model with $E(m){:}E(N)\to E_m(M)$ such that $E(m)|_\iota=m$. Since $M\models\Phi$, $E_m(M)\models_{\Sigma(N)}\iota(\Phi)$, and, since $E_m(M)\in Ext(E(N))$, $E_m(M)\models_{\Sigma(N)}\Delta^+(N)$. Hence, there is a (unique) morphism $g{:}I\to E_m(M)$. Now, since $E(N)$ is reachable, there exists at most one morphism from $E(N)$ to $E_m(M)$, and so we have $f;g=E(m)$, which implies $f|_\iota;g|_\iota=m\in\mathbf{M}_\Sigma$. Since $f|_\iota\in\mathbf{E}_\Sigma$, it follows from Exercise 3.3.5 that $f|_\iota{:}N\to I|_\iota$ is indeed an isomorphism.

(*Products*): Consider any family $M_i\in Mod_\Sigma(\Phi)$, $i\in J$, where J is any set (of indices). Let N with projections $\pi_i{:}N\to M_i$, $i\in J$, be the product of $\langle M_i\rangle_{i\in J}$. We proceed similarly as in the previous case: let $\Sigma(N)$ be a diagram signature for N with signature inclusion $\iota{:}\Sigma\to\Sigma(N)$, and let $\Delta^+(N)\subseteq\mathbf{Sen}(\Sigma(N))$ be the positive diagram of N. The presentation $\langle\Sigma(N),\Delta^+(N)\cup\iota(\Phi)\rangle$ has a reachable initial model, say I. We show that $I|_\iota$ is isomorphic to N, which implies that $N\in Mod_\Sigma(\Phi)$.

Just as in the previous case, there exists $f{:}E(N)\to I$ with $f|_\iota\in\mathbf{E}_\Sigma$.

Then, for $i\in J$, let $E_{\pi_i}(M_i)$ be the unique $\Sigma(N)$-model such that there is an expansion of π_i to a $\Sigma(N)$-model morphism $E(\pi_i){:}E(N)\to E_{\pi_i}(M_i)$. $E_{\pi_i}(M_i)$ satisfies both $\Delta^+(N)$ and $\iota(\Phi)$, and so there exists a morphism $h_i{:}I\to E_{\pi_i}(M_i)$. Hence, by the definition of a product, there exists a (unique) Σ-model morphism $g{:}I|_\iota\to N$ such that for $i\in J$, $h_i|_\iota=g;\pi_i$. Moreover, for $i\in J$, since $E(N)$ is reachable and so there is at most one morphism from $E(N)$ to $E_{\pi_i}(M_i)$, $f;h_i=E(\pi_i)$. Consequently, $(f|_\iota;g);\pi_i=f|_\iota;h_i|_\iota=(f;h_i)|_\iota=E(\pi_i)|_\iota=\pi_i$. It follows that $f|_\iota;g$ is an isomorphism, and thus $f|_\iota\in\mathbf{E}_\Sigma$ implies that $f|_\iota{:}N\to I|_\iota$ is an isomorphism as well. □

Exercise 4.5.21. As we have mentioned earlier, institutions of single-sorted logics, like those in Exercises 4.4.10 and 4.4.16, are only semi-exact, rather than finitely exact.

Call an institution **INS** *almost abstract algebraic* if it satisfies all the assumptions imposed on abstract algebraic institution except for the requirement of finite exactness, instead of which we require that:

- **INS** is semi-exact; and
- for each signature $\Sigma\in|\mathbf{Sign_{INS}}|$, the category $\mathbf{Mod_{INS}}(\Sigma)$ of Σ-models has an initial object.

The above characterisation theorems nearly hold for almost abstract algebraic institutions:

- By direct inspection of their proofs, check that Theorem 4.5.20 as well as the "if" parts of Theorems 4.5.18 and 4.5.19 hold for almost abstract algebraic institutions.
- Prove that the "only if" part of Theorem 4.5.18 holds for almost abstract algebraic institutions. HINT: To show that a Σ-theory T has an initial model, consider the identity signature morphism as a morphism from the empty Σ-theory to T. Then use Exercise 3.5.17.
- Show that the "only if" part of Theorem 4.5.19 does not hold for almost abstract algebraic institutions. HINT: In **SSEQ**, the requirement of σ-reachability is trivial for any signature morphism σ. Consider the extension of **SSEQ** by sentences involving the quantifier "there exists a unique". □

4.6 Bibliographical remarks

This chapter has its origins in the seminal work of Goguen and Burstall on institutions. The reader may have noticed that the main paper on institutions [GB92] appeared later than many of its applications. The first appearance of institutions was in the semantics of Clear [BG80], under the name "language", and early versions of [GB92] were widely circulated, with [GB84a] as an early published version. Most of our terminology (signature, sentence, model, liberal institution, etc.) comes from [GB92]. There is a minor technical difference with respect to the definition given in [GB92]: we take the contravariant functor $\mathbf{Mod}_{\mathbf{INS}}$ to be $\mathbf{Mod}_{\mathbf{INS}} \colon \mathbf{Sign}_{\mathbf{INS}}^{op} \to \mathbf{Cat}$ rather than $\mathbf{Mod}_{\mathbf{INS}} \colon \mathbf{Sign}_{\mathbf{INS}} \to \mathbf{Cat}^{op}$. This is consistent with the further refinement of this definition in Chapter 10 as well as with the notion of an indexed category (cf. Section 3.4.3 and [TBG91]).

A large number of variants, generalisations and extensions of the notion of institution have been considered. In some work where model morphisms are not important, institutions were considered with classes (rather than categories) of models, e.g. [BG80]. Somewhat dually, one way to bring deduction into the realm of institutions is by considering categories (rather than sets) of sentences, where morphisms capture proofs. These variants were present in some unpublished versions of [GB92]; see also [MGDT07] for some elaboration on these possibilities.

One line of generalisation is to allow a space of truth values other than just the standard two-valued set, leading to proposals like galleries [May85] or generalised institutions [GB86]. General logics [Mes89] add an explicit notion of entailment and proof to institutions, see Chapter 9 for developments in this direction. Foundations [Poi88] include a similar idea, in addition imposing a rich indexed category structure on sentences. Context institutions [Paw96] offer an explicit notion of context and hence of open formulae and valuation as a part of the institution structure. There have also been attempts to relax the satisfaction condition, with for instance pre-institutions [SS93], [SS96], where the equivalence in the satisfaction conditions is split into two separately-imposed implications. This captures logical systems in which one or both of the directions of the satisfaction condition fail, as discussed before Exercise 4.1.2. This applies to the so-called ultra-loose approach to algebraic specification [WB89], Extended ML [KST97] and various notions of behavioural satisfaction, see Chapter 8. (In [Gog91a], the satisfaction condition is satisfied for behavioural satisfaction but at the cost of restricting the notion of signature morphism.) Overall though, in spite of all these proposed variants and generalisations, most research has been based on the original notion, as we present it here.

The theory of institutions adopts a primarily model-theoretic view of logical systems. This does not preclude proof-theoretic investigation, see Chapter 9, but it does exclude logical systems that are inherently not based on the Tarskian notion of satisfaction of a sentence in a model. Typically such systems are centred around a notion of logical consequence that is defined via deduction, in contrast to our Definition 4.2.5. One such example would be non-monotonic logics [MT93], where increasing the set of premises can render consequences invalid. Other examples include substructural logics such as linear logic [Gir87], where changing the number of occurrences of premises, or their order, may affect deduction and change the set of valid consequences. Clearly, such logics cannot be directly represented as institutions, but see for instance [CM97] which indicates how an institution for linear logic can be defined by taking linear logic sequents (statements about consequence) as individual sentences. A view of logic based on proof rules and deduction underlies so-called "general logical frameworks", with Edinburgh LF [HHP93] as a prime example. For proposals in this direction related to institutions, see π-institutions [FS88] and also entailment systems [Mes89], [HST94], which re-emerge in Definition 9.1.2 below.

Sections 4.1.1 gives only the beginning of the long list of examples of logical systems that have been formalised as institutions. Standard examples of institutions (**EQ**, **FOP**, **Horn**, **Horn** without equality, $\mathbf{EQ}^{\Rightarrow}$) were in [GB92] with further standard algebraic variants in [Mos96b], and **CEQ** is from [Tar86b].

Dozens of other logical systems have been formalised as institutions. Some examples: [Bor00] defines an institution of higher-order logic based on HOL; [SML05] defines an institution with type class polymorphism; [Roş94] defines an institution of order-sorted equational logic; [ACEGG91] defines a family of institutions of multiple-valued logics, including logical systems arising from fuzzy set theory; [Dia00] defines an institution of constraint logic; [Cîr02] defines an institution with models that have both coalgebraic and algebraic components, and sentences involving modal formulae; [FC96] defines an institution of temporal logic; [LS00] defines an institution of hybrid systems based on the specification language of HyTech [HHWT97]; and [BH06a] defines the COL constructor-based observational logic institution based on viewing reachability and observability as dual concepts. The semantics of basic specifications in CASL [ST04] defines an institution, the rest of the semantics being defined in an institution-independent fashion. Alternatives to the standard CASL institution include: the institution underlying CoCASL, which includes cogeneration constraints, cofreeness constraints, and modal formulae [MSRR06]; the institution underlying HasCASL, with partial higher order functions, higher-order subtyping, shallow polymorphism, and type classes, designed for specifying functional programs [SM09]; an institution of labelled transition logic for specifying dynamic reactive systems [RAC99]; and the institution underlying CSP-CASL for describing systems of processes [Rog06]. The eight institutions involved in CafeOBJ [DF98] are defined in [DF02], with their combination leading to an institution via a version of the Grothendieck construction (Definition 3.4.58) that is applicable here [Dia02], and the Maude language [CDE$^+$02] is based on rewriting logic [Mes92] and on the institution of membership equational logic [Mes98] (with some technical nuances of their relationship pointed at in [CMRM10]). Institutions for three different UML diagram types are defined in [CK08a, CK08b, CK08c], with the relationships between them given by institution comorphisms (see Section 10.4 below). A spectrum of institutions capturing some aspects of Semantic Web languages are defined and linked with each other in [LLD06]. Different approaches to the specification of objects have led to the definition of a number of institutions, including [SCS94] which defines an institution of temporal logic for specifying object behaviour, [GD94b] which argues that an institution based on hidden-sorted algebra is relevant, and [Zuc99] which shows how to construct an institution with features for specifying dynamic aspects of systems using so-called "d-oids" from an institution for specifying static data. Finally, some slightly non-standard examples include two institutions for graph colouring in [Sco04], a way of viewing a database as an institution [Gog10], and a framework based on institutions for typed object-oriented, XML and other data models [Ala02].

Some of the examples of constructions on institutions in Section 4.1.2 were independently introduced by others. For instance, [Mes89] constructs an institution "out of thin air" starting with theories in an entailment system, the idea of which is presented in Examples 4.1.36 and 4.1.40. Incidentally, a very interesting exercise is to use the method of diagrams (Definition 4.5.9) to show how the construction of models from theories recovers the institution for which the entailment system that generates the theories was built.

Overall though, Section 4.1.2 only hints at the issue of how institutions should be defined. In particular, we do not discuss here the notion of a *parchment* [GB86], which offers one convenient way to present institutions in a concise and uniform style, at the same time ensuring that the satisfaction condition holds. See also [MTP97, MTP98] for variants of this notion and its use for combining presentations of logical systems.

The idea of data constraints originates in [BG80], but has been independently introduced earlier by Reichel [Rei80], cf. [KR71]. Our treatment in Section 4.3 follows [GB92]. Definition 4.3.8 is essentially equivalent to the definition there, although the technicalities are somewhat different; in particular, as in [ST88a], we do not require the institution to be liberal. Hierarchy constraints [SW82], also known as generating constraints [EWT83], are like data constraints but require that some carriers are generated from other carriers rather than freeness, see Exercise 4.3.13. Exercise 4.3.14 introduces a way to specify so-called co-inductive data types involving infinitary data. This has been mixed with algebraic techniques both in specification, see COCASL [MSRR06] and in experimental programming languages, see [Hag87] and Charity [CS92, CF92]. See [Rut00] for an introduction to a comprehensive coalgebraic approach to specification which provides an alternative perspective to the material on behavioural specifications in Chapter 8 below.

Colimits of signatures and theories built over them have been used as a tool for combining theories and specifications at least since [BG77, GB78]. This follows the general ideas of [Gog73] and underlies for instance the semantics of Clear [BG80] and the commercial Specware system [Smi06]; support for the use of colimits to combine theories in a number of institutions is also offered by the HETS system [MML07]. A category-theoretic approach to software engineering which makes extensive use of these ideas is [Fia05]. Theorem 4.4.1 originates with [GB92], generalising a non-institutional version in [GB84b], and Corollary 4.4.2 is from [BG80].

The idea of amalgamation in model theory [CK90] refers to a subtler and deeper property of certain theories than does the notion defined here. The use of amalgamation in algebraic specification, in connection with pushout-style parameterisation mechanisms, originates with [EM85], following its introduction in [BPP85], see also the Extension Lemma in [EKT$^+$80, EKT$^+$83]. In the context of an arbitrary institution, it was first imposed as a requirement and linked with continuity of the model functor in [ST88a], cf. [EWT83].

Limiting the amalgamation property to pushouts along a chosen collection of signature morphisms, as in Definition 4.4.18, is important not only because of examples like those in Exercise 4.4.19. The range of relevant cases includes systems emerging in practice. For instance, the institution of CASL [Mos04] admits amalgamation for pushouts along most, but not all, CASL signature morphisms, due to problems with the required unique interpretation of subsorting coercions, see [SMT$^+$05].

There has been some confusion with the terminology surrounding exactness of institutions in the literature. The term was first used in [Mes89], although for preservation of signature pushouts (the amalgamation property) only. It became widely used after [DGS93], where it meant that the model functor maps finite colimits of signatures to limits in **Cat**, so that neither infinite colimits nor existence of colimits were covered (the latter also applies to semi-exactness as introduced there). This was sometimes missed in the literature, leading to subtle mistakes in the presentation of some results. We decided to put all of these assumptions together under the single requirement of "exactness". The notion of an institution "with composable signatures" was used in early versions of this chapter and in [Tar99] to mean the same thing as exactness, and this terminology was adopted by other authors in a few papers. The notion of exactness as used in category theory is different, although for functors between so-called Abelian categories it implies preservation of finite colimits.

The consequences of semi-exactness for preservation of finite connected colimits of signature diagrams stated in Proposition 4.4.15 appear to be new in the literature concerning institutions; they had not been clear to us until we were pointed to [CJ95] and a result there which we give as Exercise 3.4.55.

Institutions with extra structure have been used as the basis for the definition of the semantics of a number of specification languages, beginning with ASL [ST88a] which required an exact institution. In [ST86], an institution-independent semantics for the Extended ML specification language is sketched in terms of an "institution with syntax"; this requires an additional functor which gives concrete syntactic representations of sentences, together with a natural transformation which associates these concrete objects with the "abstract" sentences they represent. In [ST04], the semantics of CASL is based on an "institution with qualified symbols" [Mos00] which requires considerable additional structure in order to support the operations on signatures used in the semantics; these include union of signatures and generation of signature morphisms from maps between symbols. Similar constructions on signatures are available when the category of signatures is equipped with a so-called inclusion system, which leads to the concept of an inclusive institution [DGS93], [GR04] (see also Exercise 5.2.1 below).

Although the theory of institutions emerged originally in the context of algebraic specification theory, it shares ideas and broad goals with abstract model theory as pursued within mathematical logic, see [Bar74, BF85], which concentrates on the study of definable classes of algebras (or rather first-order structures), abstracting away from the structure of sentences and from proof-theoretic mechanisms. The idea of developing an institutional version of abstract model theory, which also abstracts away from the nature of models, was first put forward in [Tar86a], where for instance the equivalence of the Craig interpolation and Robinson consistency properties, mentioned in Section 4.4.1, was shown.

The Craig interpolation property (Definition 4.4.21) will be used frequently in the sequel. In this formulation, it originates in [Tar86a]. Interpolation for first-order logic is a standard result in model theory [CK90] but the delicacy of its status in many-sorted first-order logic (see Exercise 4.4.23) was first pointed out in [Bor05]. There are several variants of the formulation of interpolation [DM00]. the generalisation to arbitrary commuting squares of signature morphisms [Dia08] and sets of interpolants (see the discussion in [DGS93]) is especially important. In particular, sets of interpolants may always be found in the case of equational logic under the assumption that carriers are non-empty [Rod91], but the necessity of this assumption has been widely disregarded, see Exercise 4.4.25.

Our treatment of variables, open formulae and quantification in an arbitrary institution comes from [Tar86b, ST88a]; see the concept of syntactic operator in [Bar74] for an earlier related idea. Section 4.5 is based on [Tar85], following [MM84] which is in an institutional style but based on the standard notion of logical structure. In [Tar86b], infinitary conditional "equations" were defined for an arbitrary abstract algebraic institution and it was shown that sets of these sentences define quasi-varieties, see [Mal71], thus obtaining a "syntactic" version of Theorem 4.5.20. Further developments in institutional abstract model theory, with results and ideas that refine those in Sections 4.4 and 4.5 and reach much further into classical model theory than we have done here, are in [Dia08].

References

AC89. Egidio Astesiano and Maura Cerioli. On the existence of initial models for partial (higher-order) conditional specifications. In Josep Díaz and Fernando Orejas, editors, *Proceedings of the International Joint Conference on Theory and Practice of Software Development, TAPSOFT'89*, Barcelona, *Lecture Notes in Computer Science*, volume 351, pages 74–88. Springer, 1989.

AC01. David Aspinall and Adriana B. Compagnoni. Subtyping dependent types. *Theoretical Computer Science*, 266(1–2):273–309, 2001.

ACEGG91. Jaume Agustí-Cullell, Francesc Esteva, Pere Garcia, and Lluis Godo. Formalizing multiple-valued logics as institutions. In Bernadette Bouchon-Meunier, Ronald R. Yager, and Lotfi A. Zadeh, editors, *Proceedings of the 3rd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'90*, Paris, *Lecture Notes in Computer Science*, volume 521, pages 269–278. Springer, 1991.

AF96. Mário Arrais and José Luiz Fiadeiro. Unifying theories in different institutions. In Magne Haveraaen, Olaf Owe, and Ole-Johan Dahl, editors, *Recent Trends in Data Type Specification. Selected Papers from the 11th Workshop on Specification of Abstract Data Types*, Oslo, *Lecture Notes in Computer Science*, volume 1130, pages 81–101. Springer, 1996.

AG97. Robert Allen and David Garlan. A formal basis for architectural connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213–249, 1997.

AH05. David Aspinall and Martin Hofmann. Dependent types. In Benjamin Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 2, pages 45–86. MIT Press, 2005.

AHS90. Jiři Adámek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories: The Joy of Cats*. Wiley, 1990.

Ala02. Suad Alagic. Institutions: Integrating objects, XML and databases. *Information and Software Technology*, 44(4):207–216, 2002.

AM75. Michael A. Arbib and Ernest G. Manes. *Arrows, Structures and Functors: The Categorical Imperative*. Academic Press, 1975.

Asp95. David Aspinall. Subtyping with singleton types. In Leszek Pacholski and Jerzy Tiuryn, editors, *Proceedings of the 8th International Workshop on Computer Science Logic, CSL'94*, Kazimierz, *Lecture Notes in Computer Science*, volume 933, pages 1–15. Springer, 1995.

Asp97. David Aspinall. *Type Systems for Modular Programming and Specification*. PhD thesis, University of Edinburgh, Department of Computer Science, 1997.

Asp00. David Aspinall. Subtyping with power types. In Peter Clote and Helmut Schwichtenberg, editors, *Proceedings of the 14th International Workshop on Computer Science*

Logic, Fischbachau, *Lecture Notes in Computer Science*, volume 1862, pages 156–171. Springer, 2000.

Avr91. Arnon Avron. Simple consequence relations. *Information and Computation*, 92:105–139, 1991.

Awo06. Steve Awodey. *Category Theory*. Oxford University Press, 2006.

Bar74. Jon Barwise. Axioms for abstract model theory. *Annals of Mathematical Logic*, 7:221–265, 1974.

BBB$^+$85. Friedrich L. Bauer, Rudolf Berghammer, Manfred Broy, Walter Dosch, Franz Geiselbrechtinger, Rupert Gnatz, E. Hangel, Wolfgang Hesse, Bernd Krieg-Brückner, Alfred Laut, Thomas Matzner, Bernd Möller, Friederike Nickl, Helmut Partsch, Peter Pepper, Klaus Samelson, Martin Wirsing, and Hans Wössner. *The Munich Project CIP: Volume 1: The Wide Spectrum Language CIP-L*, *Lecture Notes in Computer Science*, volume 183. Springer, 1985.

BBC86. Gilles Bernot, Michel Bidoit, and Christine Choppy. Abstract data types with exception handling: An initial approach based on a distinction between exceptions and errors. *Theoretical Computer Science*, 46(1):13–45, 1986.

BC88. Val Breazu-Tannen and Thierry Coquand. Extensional models for polymorphism. *Theoretical Computer Science*, 59(1–2):85–114, 1988.

BCH99. Michel Bidoit, María Victoria Cengarle, and Rolf Hennicker. Proof systems for structured specifications and their refinements. In Egidio Astesiano, Hans-Jörg Kreowski, and Bernd Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, chapter 11, pages 385–433. Springer, 1999.

BD77. R.M. Burstall and J. Darlington. A transformational system for developing recursive programs. *Journal of the Association for Computing Machinery*, 24(1):44–67, 1977.

BDP$^+$79. Manfred Broy, Walter Dosch, Helmut Partsch, Peter Pepper, and Martin Wirsing. Existential quantifiers in abstract data types. In Hermann A. Maurer, editor, *Proceedings of the 6th International Colloquium on Automata, Languages and Programming*, Graz, *Lecture Notes in Computer Science*, volume 71, pages 73–87. Springer, 1979.

Bén85. Jean Bénabou. Fibred categories and the foundations of naïve category theory. *Journal of Symbolic Logic*, 50:10–37, 1985.

Ber87. Gilles Bernot. Good functors ... are those preserving philosophy! In David H. Pitt, Axel Poigné, and David E. Rydeheard, editors, *Proceedings of the 2nd Summer Conference on Category Theory and Computer Science*, Edinburgh, *Lecture Notes in Computer Science*, volume 283, pages 182–195. Springer, 1987.

BF85. Jon Barwise and Solomon Feferman, editors. *Model-Theoretic Logics*. Springer, 1985.

BG77. R.M. Burstall and J.A. Goguen. Putting theories together to make specifications. In *Fifth International Joint Conference on Artificial Intelligence*, pages 1045–1058, Boston, 1977.

BG80. R.M. Burstall and J.A. Goguen. The semantics of Clear, a specification language. In Dines Bjørner, editor, *Proceedings of the 1979 Copenhagen Winter School on Abstract Software Specification*, *Lecture Notes in Computer Science*, volume 86, pages 292–332. Springer, 1980.

BG81. R.M. Burstall and J.A. Goguen. An informal introduction to specifications using Clear. In R.S. Boyer and J.S. Moore, editors, *The Correctness Problem in Computer Science*, pages 185–213. Academic Press, 1981. Also in: *Software Specification Techniques* (eds. N. Gehani and A.D. McGettrick), Addison-Wesley, 1986.

BG01. Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 19–99. Elsevier and MIT Press, 2001.

BH96. Michel Bidoit and Rolf Hennicker. Behavioural theories and the proof of behavioural properties. *Theoretical Computer Science*, 165(1):3–55, 1996.

BH98. Michel Bidoit and Rolf Hennicker. Modular correctness proofs of behavioural implementations. *Acta Informatica*, 35(11):951–1005, 1998.

BH06a. Michel Bidoit and Rolf Hennicker. Constructor-based observational logic. *Journal of Logic and Algebraic Programming*, 67(1–2):3–51, 2006.

BH06b. Michel Bidoit and Rolf Hennicker. Proving behavioral refinements of COL-specifications. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning and Computation: Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, *Lecture Notes in Computer Science*, volume 4060, pages 333–354. Springer, 2006.

BHK90. Jan Bergstra, Jan Heering, and Paul Klint. Module algebra. *Journal of the Association for Computing Machinery*, 37(2):335–372, 1990.

BHW94. Michel Bidoit, Rolf Hennicker, and Martin Wirsing. Characterizing behavioural semantics and abstractor semantics. In Donald Sannella, editor, *Proceedings of the 5th European Symposium on Programming*, Edinburgh, *Lecture Notes in Computer Science*, volume 788, pages 105–119. Springer, 1994.

BHW95. Michel Bidoit, Rolf Hennicker, and Martin Wirsing. Behavioural and abstractor specifications. *Science of Computer Programming*, 25(2-3):149–186, 1995.

Bir35. Garrett Birkhoff. On the structure of abstract algebras. *Proceedings of the Cambridge Philosophical Society*, 31:433–454, 1935.

BL69. R.M. Burstall and P.J. Landin. Programs and their proofs: an algebraic approach. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 17–43. Edinburgh University Press, 1969.

BM04. Michel Bidoit and Peter D. Mosses, editors. CASL *User Manual*. Number 2900 in Lecture Notes in Computer Science. Springer, 2004.

BN98. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

Bor94. Francis Borceaux. *Handbook of Categorical Algebra*. Cambridge University Press, 1994.

Bor00. Tomasz Borzyszkowski. Higher-order logic and theorem proving for structured specifications. In Didier Bert, Christine Choppy, and Peter D. Mosses, editors, *Recent Trends in Algebraic Development Techniques. Selected Papers from the 14th International Workshop on Algebraic Development Techniques*, Château de Bonas, *Lecture Notes in Computer Science*, volume 1827, pages 401–418. Springer, 2000.

Bor02. Tomasz Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286(2):197–245, 2002.

Bor05. Tomasz Borzyszkowski. Generalized interpolation in first order logic. *Fundamenta Informaticae*, 66(3):199–219, 2005.

BPP85. Edward K. Blum and Francesco Parisi-Presicce. The semantics of shared submodules specifications. In Hartmut Ehrig, Christiane Floyd, Maurice Nivat, and James W. Thatcher, editors, *Mathematical Foundations of Software Development. Proceedings of the International Joint Conference on Theory and Practice of Software Development. Volume 1: Colloquium on Trees in Algebra and Programming*, *Lecture Notes in Computer Science*, volume 185, pages 359–373. Springer, 1985.

BRJ98. Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998.

BS93. Rudolf Berghammer and Gunther Schmidt. Relational specifications. In C. Rauszer, editor, *Proc. XXXVIII Banach Center Semester on Algebraic Methods in Logic and their Computer Science Applications*, *Banach Center Publications*, volume 28, pages 167–190, Warszawa, 1993. Institute of Mathematics, Polish Academy of Sciences.

BST02. Michel Bidoit, Donald Sannella, and Andrzej Tarlecki. Architectural specifications in CASL. *Formal Aspects of Computing*, 13:252–273, 2002.

BST08. Michel Bidoit, Donald Sannella, and Andrzej Tarlecki. Observational interpretation of CASL specifications. *Mathematical Structures in Computer Science*, 18:325–371, 2008.

BT87. Jan Bergstra and John Tucker. Algebraic specifications of computable and semicomputable data types. *Theoretical Computer Science*, 50(2):137–181, 1987.

BT96. Michel Bidoit and Andrzej Tarlecki. Behavioural satisfaction and equivalence in concrete model categories. In Hélène Kirchner, editor, *Proceedings of the 21st International Colloquium on Trees in Algebra and Programming*, Linköping, *Lecture Notes in Computer Science*, volume 1059, pages 241–256. Springer, 1996.

Bur86. Peter Burmeister. *A Model Theoretic Oriented Approach to Partial Algebras*. Akademie-Verlag, 1986.

BW82a. Friedrich L. Bauer and Hans Wössner. *Algorithmic Language and Program Development*. Springer, 1982.

BW82b. Manfred Broy and Martin Wirsing. Partial abstract data types. *Acta Informatica*, 18(1):47–64, 1982.

BW85. Michael Barr and Charles Wells. *Toposes, Triples and Theories*. Number 278 in Grundlehren der mathematischen Wissenschaften. Springer, 1985.

BW95. Michael Barr and Charles Wells. *Category Theory for Computing Science*. Prentice Hall, second edition, 1995.

BWP84. Manfred Broy, Martin Wirsing, and Claude Pair. A systematic study of models of abstract data types. *Theoretical Computer Science*, 33(2–3):139–174, 1984.

Car88. Luca Cardelli. Structural subtyping and the notion of power type. In *Proceedings of the 15th ACM Symposium on Principles of Programming Languages*, San Diego, pages 70–79, 1988.

CDE+02. Manuel Clavela, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002. See also `http://maude.cs.uiuc.edu/`.

Cen94. María Victoria Cengarle. *Formal Specifications with Higher-Order Parameterization*. PhD thesis, Ludwig-Maximilians-Universität München, Institut für Informatik, 1994.

CF92. Robin Cockett and Tom Fukushima. About Charity. Technical Report No. 92/480/18, Department of Computer Science, University of Calgary, 1992.

CGR03. Carlos Caleiro, Paula Gouveia, and Jaime Ramos. Completeness results for fibred parchments: Beyond the propositional base. In Martin Wirsing, Dirk Pattinson, and Rolf Hennicker, editors, *Recent Trends in Algebraic Development Techniques. Selected Papers from the 16th International Workshop on Algebraic Development Techniques*, Frauenchiemsee, *Lecture Notes in Computer Science*, volume 2755, pages 185–200. Springer, 2003.

Chu56. Alonzo Church. *Introduction to Mathematical Logic, Volume 1*. Princeton University Press, 1956.

Cîr02. Corina Cîrstea. On specification logics for algebra-coalgebra structures: Reconciling reachability and observability. In *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures. European Joint Conferences on Theory and Practice of Software (ETAPS 2002)*, Grenoble, *Lecture Notes in Computer Science*, volume 2303, pages 82–97. Springer, 2002.

CJ95. Aurelio Carboni and Peter Johnstone. Connected limits, familial representability and Artin glueing. *Mathematical Structures in Computer Science*, 5(4):441–459, 1995.

CK90. Chen-Chung Chang and H. Jerome Keisler. *Model Theory*. North-Holland, third edition, 1990.

CK08a. María Victoria Cengarle and Alexander Knapp. An institution for OCL 2.0. Technical Report I0801, Institut für Informatik, Ludwig-Maximilians-Universität München, 2008.

CK08b. María Victoria Cengarle and Alexander Knapp. An institution for UML 2.0 interactions. Technical Report I0808, Institut für Informatik, Ludwig-Maximilians-Universität München, 2008.

CK08c. María Victoria Cengarle and Alexander Knapp. An institution for UML 2.0 static structures. Technical Report I0807, Institut für Informatik, Ludwig-Maximilians-Universität München, 2008.

CKTW08. Maria-Victoria Cengarle, Alexander Knapp, Andrzej Tarlecki, and Martin Wirsing. A heterogeneous approach to UML semantics. In Pierpaolo Degano, Rocco de Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*, *Lecture Notes in Computer Science*, volume 5065, pages 383–402. Springer, 2008.

CM97. Maura Cerioli and José Meseguer. May I borrow your logic? (Transporting logical structures along maps). *Theoretical Computer Science*, 173(2):311–347, 1997.

CMRM10. Mihai Codescu, Till Mossakowski, Adrían Riesco, and Christian Maeder. Integrating Maude into Hets. In Mike Johnson and Dusko Pavlovic, editors, *AMAST 2010*, Lecture Notes in Computer Science. Springer, 2010.

CMRS01. Carlos Caleiro, Paulo Mateus, Jaime Ramos, and Amílcar Sernadas. Combining logics: Parchments revisited. In Maura Cerioli and Gianna Reggio, editors, *Recent Trends in Algebraic Development Techniques. Selected Papers from the 15th Workshop on Algebraic Development Techniques joint with the CoFI WG Meeting*, Genova, *Lecture Notes in Computer Science*, volume 2267, pages 48–70. Springer, 2001.

Coh65. Paul M. Cohn. *Universal Algebra*. Harper and Row, 1965.

CS92. Robin Cockett and Dwight Spencer. Strong categorical datatypes I. In R.A.G. Seely, editor, *International Meeting on Category Theory 1991*, Canadian Mathematical Society Proceedings. American Mathematical Society, 1992.

CSS05. Carlos Caleiro, Amílcar Sernadas, and Cristina Sernadas. Fibring logics: Past, present and future. In Sergei N. Artemov, Howard Barringer, Artur S. d'Avila Garcez, Luís C. Lamb, and John Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay, Volume One*, pages 363–388. College Publications, 2005.

DF98. Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, *AMAST Series in Computing*, volume 6. World Scientific, 1998.

DF02. Răzvan Diaconescu and Kokichi Futatsugi. Logical foundations of CafeOBJ. *Theoretical Computer Science*, 285:289–318, 2002.

DGS93. Răzvan Diaconescu, Joseph Goguen, and Petros Stefaneas. Logical support for modularisation. In Gérard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130. Cambridge University Press, 1993.

Dia00. Răzvan Diaconescu. Category-based constraint logic. *Mathematical Structures in Computer Science*, 10(3):373–407, 2000.

Dia02. Răzvan Diaconescu. Grothendieck institutions. *Applied Categorical Structures*, 10(4):383–402, 2002.

Dia08. Răzvan Diaconescu. *Institution-independent Model Theory*. Birkhäuser, 2008.

DJ90. Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B (Formal Models and Semantics)*, pages 244–320. North-Holland and MIT Press, 1990.

DLL62. Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

DM00. Theodosis Dimitrakos and Tom Maibaum. On a generalised modularisation theorem. *Information Processing Letters*, 74(1–2):65–71, 2000.

DMR76. Martin Davis, Yuri Matiyasevich, and Julia Robinson. Hilbert's tenth problem. Diophantine equations: Positive aspects of a negative solution. In *Mathematical Developments Arising from Hilbert Problems, Proceedings of Symposia in Pure Mathematics*, volume 28, pages 323–378, Providence, Rhode Island, 1976. American Mathematical Society.

DP90. B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

Ehr78. Hans-Dieter Ehrich. Extensions and implementations of abstract data type specifications. In Józef Winkowski, editor, *Proceedings of the 7th Symposium on Mathematical Foundations of Computer Science*, Zakopane, *Lecture Notes in Computer Science*, volume 64, pages 155–164. Springer, 1978.

Ehr81. Hans-Dieter Ehrich. On realization and implementation. In Jozef Gruska and Michal Chytil, editors, *Proceedings of the 10th Symposium on Mathematical Foundations of Computer Science*, Štrbské Pleso, *Lecture Notes in Computer Science*, volume 118, pages 271–280. Springer, 1981.

Ehr82. Hans-Dieter Ehrich. On the theory of specification, implementation and parametrization of abstract data types. *Journal of the Association for Computing Machinery*, 29(1):206–227, 1982.

EKMP82. Hartmut Ehrig, Hans-Jörg Kreowski, Bernd Mahr, and Peter Padawitz. Algebraic implementation of abstract data types. *Theoretical Computer Science*, 20:209–263, 1982.

EKT$^+$80. Hartmut Ehrig, Hans-Jörg Kreowski, James Thatcher, Eric Wagner, and Jesse Wright. Parameter passing in algebraic specification languages. Technical report, Technische Universität Berlin, 1980.

EKT$^+$83. Hartmut Ehrig, Hans-Jörg Kreowski, James Thatcher, Eric Wagner, and Jesse Wright. Parameter passing in algebraic specification languages. *Theoretical Computer Science*, 28(1–2):45–81, 1983.

EM85. Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specification 1*, *EATCS Monographs on Theoretical Computer Science*, volume 6. Springer, 1985.

Eme90. E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B (Formal Models and Semantics)*, pages 995–1072. North-Holland and MIT Press, 1990.

End72. Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

EPO89. Hartmut Ehrig, Peter Pepper, and Fernando Orejas. On recent trends in algebraic specification. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *Proceeding of the 16th International Colloquium on Automata, Languages and Programming*, Stresa, *Lecture Notes in Computer Science*, volume 372, pages 263–288. Springer, 1989.

EWT83. Hartmut Ehrig, Eric G. Wagner, and James W. Thatcher. Algebraic specifications with generating constraints. In *Proceeding of the 10th International Colloquium on Automata, Languages and Programming*, Barcelona, *Lecture Notes in Computer Science*, volume 154, pages 188–202. Springer, 1983.

Far89. Jordi Farrés-Casals. Proving correctness of constructor implementations. In Antoni Kreczmar and Grazyna Mirkowska, editors, *Proceedings of the 14th Symposium on Mathematical Foundations of Computer Science*, Porabka-Kozubnik, *Lecture Notes in Computer Science*, volume 379, pages 225–235. Springer, 1989.

Far90. Jordi Farrés-Casals. Proving correctness wrt specifications with hidden parts. In Hélène Kirchner and Wolfgang Wechler, editors, *Proceedings of the 2nd International Conference on Algebraic and Logic Programming*, Nancy, *Lecture Notes in Computer Science*, volume 463, pages 25–39. Springer, 1990.

Far92. Jordi Farrés-Casals. *Verification in ASL and Related Specification Languages*. PhD thesis, University of Edinburgh, Department of Computer Science, 1992.

FC96. José Luiz Fiadeiro and José Félix Costa. Mirror, mirror in my hand: A duality between specifications and models of process behaviour. *Mathematical Structures in Computer Science*, 6(4):353–373, 1996.

Fei89. Loe M. G. Feijs. The calculus $\lambda\pi$. In Martin Wirsing and Jan A. Bergstra, editors, *Proceedings of the Workshop on Algebraic Methods: Theory, Tools and Applications*, *Lecture Notes in Computer Science*, volume 394, pages 307–328. Springer, 1989.

FGT92. William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. Little theories. In Deepak Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction*, *Lecture Notes in Artificial Intelligence*, volume 607, pages 567–581, Saratoga Springs, 1992. Springer.

Fia05. José Luiz Fiadeiro. *Categories for Software Engineering*. Springer, 2005.

Fit08. John S. Fitzgerald. The typed logic of partial functions and the Vienna Development Method. In Dines Bjørner and Martin Henson, editors, *Logics of Specification Languages*, pages 453–487. Springer, 2008.

FJ90. J. Fitzgerald and C.B. Jones. Modularizing the formal description of a database system. In *Proceedings of the 3rd International Symposium of VDM Europe: VDM and Z, Formal Methods in Software Development*, Kiel, *Lecture Notes in Computer Science*, volume 428, pages 189–210. Springer, 1990.

FS88. José Luiz Fiadeiro and Amílcar Sernadas. Structuring theories on consequence. In Donald Sannella and Andrzej Tarlecki, editors, *Recent Trends in Data Type Specification. Selected Papers from the 5th Workshop on Specification of Abstract Data Types*, Gullane, *Lecture Notes in Computer Science*, volume 332, pages 44–72. Springer, 1988.

Gab98. Dov M. Gabbay. *Fibring Logics*, *Oxford Logic Guides*, volume 38. Oxford University Press, 1998.

Gan83. Harald Ganzinger. Parameterized specifications: Parameter passing and implementation with respect to observability. *ACM Transactions on Programming Languages and Systems*, 5(3):318–354, 1983.

GB78. J.A. Goguen and R.M. Burstall. Some fundamental properties of algebraic theories: a tool for semantics of computation. Technical Report 53, Department of Artificial Intelligence, University of Edinburgh, 1978. Revised version appeared as [GB84b] and [GB84c].

GB80. J.A. Goguen and R.M. Burstall. CAT, a system for the structured elaboration of correct programs from structured specifications. Technical Report CSL-118, Computer Science Laboratory, SRI International, 1980.

GB84a. J.A. Goguen and R.M. Burstall. Introducing institutions. In Edmund Clarke and Dexter Kozen, editors, *Proceedings of the Workshop on Logics of Programs*, Pittsburgh, *Lecture Notes in Computer Science*, volume 164, pages 221–256. Springer, 1984.

GB84b. J.A. Goguen and R.M. Burstall. Some fundamental algebraic tools for the semantics of computation. Part 1: Comma categories, colimits, signatures and theories. *Theoretical Computer Science*, 31:175–209, 1984.

GB84c. J.A. Goguen and R.M. Burstall. Some fundamental algebraic tools for the semantics of computation. Part 2: Signed and abstract theories. *Theoretical Computer Science*, 31:263–295, 1984.

GB86. Joseph A. Goguen and Rod M. Burstall. A study in the functions of programming methodology: Specifications, institutions, charters and parchments. In David H. Pitt, Samson Abramsky, Axel Poigné, and David E. Rydeheard, editors, *Proceedings of the Tutorial and Workshop on Category Theory and Computer Programming*, Guildford, *Lecture Notes in Computer Science*, volume 240, pages 313–333. Springer, 1986.

GB92. J.A. Goguen and R.M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

GD94a. Joseph Goguen and Răzvan Diaconescu. An Oxford survey of order sorted algebra. *Mathematical Structures in Computer Science*, 4(3):363–392, 1994.

GD94b. Joseph A. Goguen and Răzvan Diaconescu. Towards an algebraic semantics for the object paradigm. In Hartmut Ehrig and Fernando Orejas, editors, *Recent Trends in Data Type Specification. Selected Papers from the 9th Workshop on Specification of Abstract Data Types joint with the 4th* COMPASS *Workshop*, Caldes de Malavella, *Lecture Notes in Computer Science*, volume 785, pages 1–29. Springer, 1994.

GDLE84. Martin Gogolla, Klaus Drosten, Udo Lipeck, and Hans-Dieter Ehrich. Algebraic and operational semantics of specifications allowing exceptions and errors. *Theoretical Computer Science*, 34(3):289–313, 1984.

GG89. Stephen J. Garland and John V. Guttag. An overview of LP, the Larch Prover. In *Third International Conference on Rewriting Techniques and Applications*, Chapel Hill, *Lecture Notes in Computer Science*, volume 355, pages 137–151. Springer, 1989. See also `http://nms.lcs.mit.edu/larch/LP/all.html`.

GGM76. V. Giarratana, F. Gimona, and Ugo Montanari. Observability concepts in abstract data type specifications. In Antoni Mazurkiewicz, editor, *Proceedings of the 5th Sympo-*

sium on Mathematical Foundations of Computer Science, Gdańsk, *Lecture Notes in Computer Science*, volume 45, pages 567–578. Springer, 1976.

GH78. John Guttag and James Horning. The algebraic specification of abstract data types. *Acta Informatica*, 10:27–52, 1978.

GH93. John V. Guttag and James J. Horning. *Larch: Languages and Tools for Formal Specification*. Springer, 1993.

Gin68. Abraham Ginzburg. *Algebraic Theory of Automata*. Academic Press, 1968.

Gir87. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

Gir89. Jean-Yves Girard. *Proofs and Types*, *Cambridge Tracts in Theoretical Computer Science*, volume 7. Cambridge University Press, 1989. Translated and with appendices by Paul Taylor and Yves Lafont.

GLR00. Joseph Goguen, Kai Lin, and Grigore Roşu. Circular coinductive rewriting. In *Proceedings of the 15th International Conference on Automated Software Engineering*, Grenoble. IEEE Computer Society, 2000.

GM82. Joseph A. Goguen and José Meseguer. Universal realization, persistent interconnection and implementation of abstract modules. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Proceeding of the 9th International Colloquium on Automata, Languages and Programming*, Aarhus, *Lecture Notes in Computer Science*, volume 140, pages 265–281. Springer, 1982.

GM85. Joseph Goguen and José Meseguer. Completeness of many sorted equational deduction. *Houston Journal of Mathematics*, 11(3):307–334, 1985.

GM92. Joseph Goguen and José Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105(2):217–273, 1992.

GM00. Joseph A. Goguen and Grant Malcolm. A hidden agenda. *Theoretical Computer Science*, 245(1):55–101, 2000.

Gog73. Joseph Goguen. Categorical foundations for general systems theory. In F. Pichler and R. Trappl, editors, *Advances in Cybernetics and Systems Research*, London, pages 121–130. Transcripta Books, 1973.

Gog74. J.A. Goguen. Semantics of computation. In Ernest G. Manes, editor, *Proceedings of the 1st International Symposium on Category Theory Applied to Computation and Control*, San Francisco, *Lecture Notes in Computer Science*, volume 25, pages 151–163. Springer, 1974.

Gog78. Joseph Goguen. Abstract errors for abstract data types. In Erich Neuhold, editor, *Formal Description of Programming Concepts*, pages 491–526. North-Holland, 1978.

Gog84. Martin Gogolla. Partially ordered sorts in algebraic specifications. In *Proceedings of the 9th Colloquium on Trees in Algebra and Programming*, pages 139–153. Cambridge University Press, 1984.

Gog85. Martin Gogolla. A final algebra semantics for errors and exceptions. In Hans-Jörg Kreowski, editor, *Recent Trends in Data Type Specification. Selected Papers from the 3rd Workshop on Theory and Applications of Abstract Data Types*, Bremen, *Informatik-Fachberichte*, volume 116, pages 89–103. Springer, 1985.

Gog91a. Joseph Goguen. Types as theories. In G.M. Reed, A.W. Roscoe, and R.F. Wachter, editors, *Topology and Category Theory in Computer Science*, Oxford, pages 357–390. Oxford University Press, 1991.

Gog91b. Joseph A. Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1):49–67, 1991.

Gog96. Joseph A. Goguen. Parameterized programming and software architecture. In Murali Sitaraman, editor, *Proceedings of the Fourth International Conference on Software Reuse*, pages 2–11. IEEE Computer Society Press, 1996.

Gog10. Joseph Goguen. Information integration in institutions. In Larry Moss, editor, *Thinking Logically: a Volume in Memory of Jon Barwise*. CSLI, Stanford University, 2010. To appear.

Gol06. Robert Goldblatt. *Topoi: The Categorial Analysis of Logic*. Dover, revised edition, 2006.

Gor95. Andrew D. Gordon. Bisimilarity as a theory of functional programming. In *Proceedings of the 11th Annual Conference on Mathematical Foundations of Programming Semantics*. *Electronic Notes in Theoretical Computer Science*, 1:232–252, 1995.

GR02. Joseph A. Goguen and Grigore Roşu. Institution morphisms. *Formal Aspects of Computing*, 13(3-5):274–307, 2002.

GR04. Joseph A. Goguen and Grigore Roşu. Composing hidden information modules over inclusive institutions. In *From Object-Orientation to Formal Methods. Essays in Memory of Ole-Johan Dahl*, *Lecture Notes in Computer Science*, volume 2635, pages 96–123. Springer, 2004.

Grä79. George A. Grätzer. *Universal Algebra*. Springer, second edition, 1979.

GS90. Carl Gunter and Dana Scott. Semantic domains. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B (Formal Models and Semantics)*, pages 633–674. North-Holland and MIT Press, 1990.

GTW76. Joseph Goguen, James Thatcher, and Eric Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. Technical Report RC 6487, IBM Watson Research Center, Yorktown Heights NY, 1976. Also in: *Current Trends in Programming Methodology. Volume IV (Data Structuring)* (ed. R.T. Yeh), Prentice-Hall, 80–149, 1978.

GTWW73. Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. A junction between computer science and category theory, I: Basic concepts and examples (part 1). Technical Report RC 4526, IBM Watson Research Center, Yorktown Heights NY, 1973.

GTWW75. Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. An introduction to categories, algebraic theories and algebras. Technical Report RC 5369, IBM Watson Research Center, Yorktown Heights NY, 1975.

GTWW77. Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. Initial algebra semantics and continuous algebras. *Journal of the Association for Computing Machinery*, 24(1):68–95, 1977.

Gut75. John Guttag. *The Specification and Application to Programming of Abstract Data Types*. PhD thesis, University of Toronto, Department of Computer Science, 1975.

Hag87. Tatsuya Hagino. *A Categorical Programming Language*. PhD thesis, University of Edinburgh, Department of Computer Science, 1987.

Häh01. Reiner Hähnle. Tableaux and related methods. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 100–178. Elsevier and MIT Press, 2001.

Hal70. Paul R. Halmos. *Naive Set Theory*. Undergraduate Texts in Mathematics. Springer, 1970.

Hat82. William Hatcher. *The Logical Foundations of Mathematics*. Foundations and Philosophy of Science and Technology. Pergamon Press, 1982.

Hay94. Susumu Hayashi. Singleton, union and intersection types for program extraction. *Information and Computation*, 109(1/2):174–210, 1994.

Hee86. Jan Heering. Partial evaluation and ω-completeness of algebraic specifications. *Theoretical Computer Science*, 43:149–167, 1986.

Hen91. Rolf Hennicker. Context induction: A proof principle for behavioural abstractions and algebraic implementations. *Formal Aspects of Computing*, 3(4):326–345, 1991.

HHP93. Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

HHWT97. Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.

Hig63. Phillip J. Higgins. Algebras with a scheme of operators. *Mathematische Nachrichten*, 27:115–132, 1963.

HLST00. Furio Honsell, John Longley, Donald Sannella, and Andrzej Tarlecki. Constructive data refinement in typed lambda calculus. In *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures. European Joint Conferences on Theory and Practice of Software (ETAPS 2000)*, Berlin, *Lecture Notes in Computer Science*, volume 1784, pages 161–176. Springer, 2000.

Hoa72. C. A. R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1:271–281, 1972.

HS73. Horst Herrlich and George E. Strecker. *Category Theory: An Introduction*. Allyn and Bacon, 1973.

HS96. Martin Hofmann and Donald Sannella. On behavioural abstraction and behavioural satisfaction in higher-order logic. *Theoretical Computer Science*, 167:3–45, 1996.

HS02. Furio Honsell and Donald Sannella. Prelogical relations. *Information and Computation*, 178:23–43, 2002.

HST94. Robert Harper, Donald Sannella, and Andrzej Tarlecki. Structured presentations and logic representations. *Annals of Pure and Applied Logic*, 67:113–160, 1994.

Hus92. Heinrich Hussmann. Nondeterministic algebraic specifications and nonconfluent term rewriting. *Journal of Logic Programming*, 12(1–4):237–255, 1992.

HWB97. Rolf Hennicker, Martin Wirsing, and Michel Bidoit. Proof systems for structured specifications with observability operators. *Theoretical Computer Science*, 173(2):393–443, 1997.

Jac99. Bart Jacobs. *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. Elsevier Science, 1999.

JL87. Joxan Jaffar and Jean-Louis Lassez. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, Munich, pages 111–119, 1987.

JNW96. André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

JOE95. Rosa M. Jiménez, Fernando Orejas, and Hartmut Ehrig. Compositionality and compatibility of parameterization and parameter passing in specification languages. *Mathematical Structures in Computer Science*, 5(2):283–314, 1995.

Joh02. Peter T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium*. Oxford Logic Guides Series. Clarendon Press, 2002.

Jon80. Cliff B. Jones. *Software Development: A Rigorous Approach*. Prentice-Hall, 1980.

Jon89. Hans B.M. Jonkers. An introduction to COLD-K. In Martin Wirsing and Jan A. Bergstra, editors, *Proceedings of the Workshop on Algebraic Methods: Theory, Tools and Applications*, *Lecture Notes in Computer Science*, volume 394, pages 139–205. Springer, 1989.

JR97. Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science*, 62:222–259, 1997.

KB70. Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

Kir99. Hélène Kirchner. Term rewriting. In Egidio Astesiano, Hans-Jörg Kreowski, and Bernd Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, chapter 9, pages 273–320. Springer, 1999.

KKM88. Claude Kirchner, Hélène Kirchner, and José Meseguer. Operational semantics of OBJ-3. In Timo Lepistö and Arto Salomaa, editors, *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*, Tampere, *Lecture Notes in Computer Science*, volume 317, pages 287–301. Springer, 1988.

Klo92. Jan Klop. Term rewriting systems. In Samson Abramsky, Dov Gabbay, and Tom Maibaum, editors, *Handbook of Logic in Computer Science. Volume 2 (Background: Computational Structures)*, pages 1–116. Oxford University Press, 1992.

KM87. Deepak Kapur and David R. Musser. Proof by consistency. *Artificial Intelligence*, 31(2):125–157, 1987.

KR71. Heinz Kaphengst and Horst Reichel. Algebraische Algorithmentheorie. Technical Report WIB 1, VEB Robotron, Zentrum für Forschung und Technik, Dresden, 1971.

Kre87. Hans-Jörg Kreowski. Partial algebras flow from algebraic specifications. In T. Ottmann, editor, *Proceedings of the 14th International Colloquium on Automata, Languages and Programming*, Karlsruhe, *Lecture Notes in Computer Science*, volume 267, pages 521–530. Springer, 1987.

KST97. Stefan Kahrs, Donald Sannella, and Andrzej Tarlecki. The definition of Extended ML: A gentle introduction. *Theoretical Computer Science*, 173:445–484, 1997.

KTB91. Beata Konikowska, Andrzej Tarlecki, and Andrzej Blikle. A three-valued logic for software specification and validation. *Fundamenta Informaticae*, 14(4):411–453, 1991.

Las98. Sławomir Lasota. Open maps as a bridge between algebraic observational equivalence and bisimilarity. In Francesco Parisi-Presicce, editor, *Recent Trends in Data Type Specification. Selected Papers from the 12th International Workshop on Specification of Abstract Data Types*, Tarquinia, *Lecture Notes in Computer Science*, volume 1376, pages 285–299. Springer, 1998.

Law63. F. William Lawvere. *Functorial Semantics of Algebraic Theories*. PhD thesis, Columbia University, 1963.

LB88. Butler Lampson and Rod Burstall. Pebble, a kernel language for modules and abstract data types. *Information and Computation*, 76(2/3):278–346, 1988.

LEW96. Jacques Loeckx, Hans-Dieter Ehrich, and Markus Wolf. *Specification of Abstract Data Types*. John Wiley and Sons, 1996.

Lin03. Kai Lin. *Machine Support for Behavioral Algebraic Specification and Verification*. PhD thesis, University of California, San Diego, 2003.

Lip83. Udo Lipeck. *Ein algebraischer Kalkül für einen strukturierten Entwurf von Datenabstraktionen*. PhD thesis, Universität Dortmund, 1983.

LLD06. Dorel Lucanu, Yuan-Fang Li, and Jin Song Dong. Semantic Web languages—towards an institutional perspective. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning and Computation: Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, *Lecture Notes in Computer Science*, volume 4060, pages 99–123. Springer, 2006.

LS86. Joachim Lambek and Philip J. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1986.

LS00. Hugo Lourenço and Amílcar Sernadas. An institution of hybrid systems. In Didier Bert, Christine Choppy, and Peter D. Mosses, editors, *Recent Trends in Algebraic Development Techniques. Selected Papers from the 14th International Workshop on Algebraic Development Techniques*, Château de Bonas, *Lecture Notes in Computer Science*, volume 1827, pages 219–236. Springer, 2000.

Luo93. Zhaohui Luo. Program specification and data refinement in type theory. *Mathematical Structures in Computer Science*, 3(3):333–363, 1993.

Mac71. Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

Mac84. David B. MacQueen. Modules for Standard ML. In *Proceedings of the 1984 ACM Conference on LISP and Functional Programming*, pages 198–207, 1984.

MAH06. Till Mossakowski, Serge Autexier, and Dieter Hutter. Development graphs — proof management for structured specifications. *Journal of Logic and Algebraic Programming*, 67(1–2):114–145, 2006.

Mai72. Tom Maibaum. The characterization of the derivation trees of context free sets of terms as regular sets. In *Proceedings of the 13th Annual IEEE Symposium on Switching and Automata Theory*, pages 224–230, 1972.

Maj77. Mila E. Majster. Limits of the “algebraic” specification of abstract data types. *ACM SIGPLAN Notices*, 12(10):37–42, 1977.

Mal71. Anatoly Malcev. Quasiprimitive classes of abstract algebras in the metamathematics of algebraic systems. In *Mathematics of Algebraic Systems: Collected Papers, 1936-67*, number 66 in Studies in Logic and Mathematics, pages 27–31. North-Holland, 1971.

Man76. Ernest G. Manes. *Algebraic Theories*. Springer, 1976.

May85. Brian Mayoh. Galleries and institutions. Technical Report DAIMI PB-191, Aarhus University, 1985.

Mei92. Karl Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100:385–417, 1992.

Mes89. José Meseguer. General logics. In H.-D. Ebbinghaus, editor, *Logic Colloquium '87*, Granada, pages 275–329. North-Holland, 1989.

Mes92. José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

Mes98. José Meseguer. Membership algebra as a logical framework for equational specification. In Francesco Parisi-Presicce, editor, *Recent Trends in Data Type Specification. Selected Papers from the 12th International Workshop on Specification of Abstract Data Types*, Tarquinia, *Lecture Notes in Computer Science*, volume 1376, pages 18–61. Springer, 1998.

Mes09. José Meseguer. Order-sorted parameterization and induction. In Jens Palsberg, editor, *Semantics and Algebraic Specification: Essays Dedicated to Peter D. Mosses on the Occasion of His 60th Birthday*, *Lecture Notes in Computer Science*, volume 5700, pages 43–80. Springer, 2009.

MG85. José Meseguer and Joseph Goguen. Initiality, induction and computability. In Maurice Nivat and John C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–541. Cambridge, 1985.

MGDT07. Till Mossakowski, Joseph Goguen, Răzvan Diaconescu, and Andrzej Tarlecki. What is a logic? In Jean-Yves Beziau, editor, *Logica Universalis: Towards a General Theory of Logic*, pages 111–135. Birkhäuser, 2007.

MHST08. Till Mossakowski, Anne Haxthausen, Donald Sannella, and Andrzej Tarlecki. CASL — the common algebraic specification language. In Dines Bjørner and Martin Henson, editors, *Logics of Specification Languages*, pages 241–298. Springer, 2008.

Mid93. Aart Middeldorp. Modular properties of conditional term rewriting systems. *Information and Computation*, 104(1):110–158, 1993.

Mil71. Robin Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pages 481–489, 1971.

Mil77. Robin Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4(1):1–22, 1977.

Mil89. Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

Mit96. John C. Mitchell. *Foundations of Programming Languages*. MIT Press, 1996.

MM84. Bernd Mahr and Johann Makowsky. Characterizing specification languages which admit initial semantics. *Theoretical Computer Science*, 31:49–60, 1984.

MML07. Till Mossakowski, Christian Maeder, and Klaus Lüttich. The heterogeneous tool set, HETS. In Orna Grumberg and Michael Huth, editors, *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. European Joint Conferences on Theory and Practice of Software (ETAPS 2007)*, Braga, *Lecture Notes in Computer Science*, volume 4424, pages 519–522. Springer, 2007. See also `http://www.informatik.uni-bremen.de/cofi/hets/`.

Mog91. Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

Moo56. Edward F. Moore. Gedanken-experiments on sequential machines. In Claude E. Shannon and John McCarthy, editors, *Annals of Mathematics Studies 34, Automata Studies*, pages 129–153. Princeton University Press, 1956.

Mos89. Peter D. Mosses. Unified algebras and modules. In *Proceedings of the 16th ACM Symposium on Principles of Programming Languages*, Austin, pages 329–343, 1989.

Mos93. Peter Mosses. The use of sorts in algebraic specifications. In Michel Bidoit and Christine Choppy, editors, *Recent Trends in Data Type Specification. Selected Papers from the 8th Workshop on Specification of Abstract Data Types joint with the 3rd* COMPASS *Workshop*, Dourdan, *Lecture Notes in Computer Science*, volume 655, pages 66–91. Springer, 1993.

Mos96a. Till Mossakowski. Different types of arrow between logical frameworks. In Friedhelm Meyer auf der Heide and Burkhard Monien, editors, *Proceedings of the 23rd International Colloquium Automata, Languages and Programming*, Paderborn, *Lecture Notes in Computer Science*, volume 1099, pages 158–169. Springer, 1996.

Mos96b. Till Mossakowski. *Representations, Hierarchies and Graphs of Institutions*. PhD thesis, Universität Bremen, 1996.

Mos00. Till Mossakowski. Specification in an arbitrary institution with symbols. In Didier Bert, Christine Choppy, and Peter D. Mosses, editors, *Recent Trends in Algebraic Development Techniques. Selected Papers from the 14th International Workshop on Algebraic Development Techniques*, Château de Bonas, *Lecture Notes in Computer Science*, volume 1827, pages 252–270. Springer, 2000.

Mos02. Till Mossakowski. Comorphism-based Grothendieck logics. In Krzysztof Diks and Wojciech Rytter, editors, *Proceedings of the 27th Symposium on Mathematical Foundations of Computer Science*, Warsaw, *Lecture Notes in Computer Science*, volume 2420, pages 593–604. Springer, 2002.

Mos03. Till Mossakowski. Foundations of heterogeneous specification. In Martin Wirsing, Dirk Pattinson, and Rolf Hennicker, editors, *Recent Trends in Algebraic Development Techniques.. Selected Papers from the 16th International Workshop on Algebraic Development Techniques*, Frauenchiemsee, *Lecture Notes in Computer Science*, volume 2755, pages 359–375. Springer, 2003.

Mos04. Peter D. Mosses, editor. CASL *Reference Manual*. Number 2960 in Lecture Notes in Computer Science. Springer, 2004.

Mos05. Till Mossakowski. Heterogeneous Specification and the Heterogeneous Tool Set. Habilitation thesis, Universität Bremen, 2005.

MS85. David MacQueen and Donald Sannella. Completeness of proof systems for equational specifications. *IEEE Transactions on Software Engineering*, SE-11(5):454–461, 1985.

MSRR06. Till Mossakowski, Lutz Schröder, Markus Roggenbach, and Horst Reichel. Algebraic-coalgebraic specification in COCASL. *Journal of Logic and Algebraic Programming*, 67(1–2):146–197, 2006.

MSS90. Vincenzo Manca, Antonino Salibra, and Giuseppe Scollo. Equational type logic. *Theoretical Computer Science*, 77(1–2):131–159, 1990.

MST04. Till Mossakowski, Donald Sannella, and Andrzej Tarlecki. A simple refinement language for CASL. In José Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques.. Selected Papers from the 17th International Workshop on Algebraic Development Techniques*, Barcelona, *Lecture Notes in Computer Science*, volume 3423, pages 162–185. Springer, 2004.

MT92. Karl Meinke and John Tucker. Universal algebra. In Samson Abramsky, Dov Gabbay, and Tom Maibaum, editors, *Handbook of Logic in Computer Science. Volume 1 (Background: Mathematical Structures)*, pages 189–409. Oxford University Press, 1992.

MT93. V. Wiktor Marek and Mirosław Truszczyński. *Nonmonotonic Logics: Context-Dependent Reasoning*. Springer, 1993.

MT94. David B. MacQueen and Mads Tofte. A semantics for higher-order functors. In Donald Sannella, editor, *Proceedings of the 5th European Symposium on Programming*, Edinburgh, *Lecture Notes in Computer Science*, volume 788, pages 409–423. Springer, 1994.

MT09. Till Mossakowski and Andrzej Tarlecki. Heterogeneous logical environments for distributed specifications. In Andrea Corradini and Ugo Montanari, editors, *Recent Trends in Algebraic Development Techniques.. Selected Papers from the 19th International Workshop on Algebraic Development Techniques*, Pisa, *Lecture Notes in Computer Science*, volume 5486, pages 266–289. Springer, 2009.

MTD09. Till Mossakowski, Andrzej Tarlecki, and Răzvan Diaconescu. What is a logic translation? *Logica Universalis*, 3(1):95–124, 2009.

MTHM97. Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised)*. MIT Press, 1997.

MTP97. Till Mossakowski, Andrzej Tarlecki, and Wiesław Pawłowski. Combining and representing logical systems. In Eugenio Moggi and Giuseppe Rosolini, editors, *Proceedings of the 7th International Conference on Category Theory and Computer Science*, Santa Margherita Ligure, *Lecture Notes in Computer Science*, volume 1290, pages 177–196. Springer, 1997.

MTP98. Till Mossakowski, Andrzej Tarlecki, and Wiesław Pawłowski. Combining and representing logical systems using model-theoretic parchments. In Francesco Parisi-Presicce, editor, *Recent Trends in Data Type Specification. Selected Papers from the 12th International Workshop on Specification of Abstract Data Types*, Tarquinia, *Lecture Notes in Computer Science*, volume 1376, pages 349–364. Springer, 1998.

MTW88. Bernhard Möller, Andrzej Tarlecki, and Martin Wirsing. Algebraic specifications of reachable higher-order algebras. In Donald Sannella and Andrzej Tarlecki, editors, *Recent Trends in Data Type Specification. Selected Papers from the 5th Workshop on Specification of Abstract Data Types*, Gullane, *Lecture Notes in Computer Science*, volume 332, pages 154–169. Springer, 1988.

Mus80. David Musser. On proving inductive properties of abstract data types. In *Proceedings of the 7th ACM Symposium on Principles of Programming Languages*, Las Vegas, pages 154–162, 1980.

MW98. Alfio Martini and Uwe Wolter. A single perspective on arrows between institutions. In Armando Haeberer, editor, *Proceedings of the 7th International Conference on Algebraic Methodology and Software Technology*, Manaus, *Lecture Notes in Computer Science*, volume 1548, pages 486–501. Springer, 1998.

Nel91. Greg Nelson, editor. *Systems Programming in Modula-3*. Prentice-Hall, 1991.

Nip86. Tobias Nipkow. Non-deterministic data types: Models and implementations. *Acta Informatica*, 22(6):629–661, 1986.

NO88. Pilar Nivela and Fernando Orejas. Initial behaviour semantics for algebraic specifications. In Donald Sannella and Andrzej Tarlecki, editors, *Recent Trends in Data Type Specification. Selected Papers from the 5th Workshop on Specification of Abstract Data Types*, Gullane, *Lecture Notes in Computer Science*, volume 332, pages 184–207. Springer, 1988.

Nou81. Farshid Nourani. On induction for programming logic: Syntax, semantics, and inductive closure. *Bulletin of the European Association for Theoretical Computer Science*, 13:51–64, 1981.

Oka98. Chris Okasaki. *Purely Functional Data Structures*. Cambridge University Press, 1998.

ONS93. Fernando Orejas, Marisa Navarro, and Ana Sánchez. Implementation and behavioural equivalence: A survey. In Michel Bidoit and Christine Choppy, editors, *Recent Trends in Data Type Specification. Selected Papers from the 8th Workshop on Specification of Abstract Data Types joint with the 3rd* COMPASS *Workshop*, Dourdan, *Lecture Notes in Computer Science*, volume 655, pages 93–125. Springer, 1993.

Ore83. Fernando Orejas. Characterizing composability of abstract implementations. In Marek Karpinski, editor, *Proceedings of the 1983 International Conference on Foundations of Computation Theory*, Borgholm, *Lecture Notes in Computer Science*, volume 158, pages 335–346. Springer, 1983.

Pad85. Peter Padawitz. Parameter preserving data type specifications. In Hartmut Ehrig, Christiane Floyd, Maurice Nivat, and James Thatcher, editors, *TAPSOFT'85: Proceedings of the International Joint Conference on Theory and Practice of Software Development. Volume 2: Colloquium on Software Engineering*, Berlin, *Lecture Notes in Computer Science*, volume 186, pages 323–341. Springer, 1985.

Pad99. Peter Padawitz. Proof in flat specifications. In Egidio Astesiano, Hans-Jörg Kreowski, and Bernd Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, chapter 10, pages 321–384. Springer, 1999.

Pau87. Laurence Paulson. *Logic and Computation: Interactive Proof with Cambridge LCF*. Cambridge University Press, 1987.

Pau96. Laurence Paulson. *ML for the Working Programmer*. Cambridge University Press, second edition, 1996.

Paw96. Wiesław Pawłowski. Context institutions. In Magne Haveraaen, Olaf Owe, and Ole-Johan Dahl, editors, *Recent Trends in Data Type Specification. Selected Papers from* *the 11th Workshop on Specification of Abstract Data Types*, Oslo, *Lecture Notes in Computer Science*, volume 1130, pages 436–457. Springer, 1996.

Pet10. Marius Petria. *Generic Refinements for Behavioural Specifications*. PhD thesis, University of Edinburgh, School of Informatics, 2010.

Pey03. Simon Peyton Jones, editor. *Haskell 98 Language and Libraries: The Revised Report*. Cambridge University Press, 2003.

Pho92. Wesley Phoa. An introduction to fibrations, topos theory, the effective topos and modest sets. Technical Report ECS-LFCS-92-208, LFCS, Department of Computer Science, University of Edinburgh, 1992.

Pie91. Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. MIT Press, 1991.

Plo77. Gordon D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977.

Poi86. Axel Poigné. On specifications, theories, and models with higher types. *Information and Control*, 68(1–3):1–46, 1986.

Poi88. Axel Poigné. Foundations are rich institutions, but institutions are poor foundations. In Hartmut Ehrig, Horst Herrlich, Hans-Jörg Kreowski, and Gerhard Preuß, editors, *Proceedings of the International Workshop on Categorical Methods in Computer Science with Aspects from Topology*, Berlin, *Lecture Notes in Computer Science*, volume 393, pages 82–101. Springer, 1988.

Poi90. Axel Poigné. Parametrization for order-sorted algebraic specification. *Journal of Computer and System Sciences*, 40:229–268, 1990.

Poi92. Axel Poigné. Basic category theory. In Samson Abramsky, Dov Gabbay, and Tom Maibaum, editors, *Handbook of Logic in Computer Science. Volume 1 (Background: Mathematical Structures)*, pages 413–640. Oxford University Press, 1992.

Pos47. Emil Post. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, 12:1–11, 1947.

PS83. Helmuth Partsch and Ralf Steinbrüggen. Program transformation systems. *ACM Computing Surveys*, 15(3):199–236, 1983.

PŞR09. Andrei Popescu, Traian Florin Şerbănuţă, and Grigore Roşu. A semantic approach to interpolation. *Theoretical Computer Science*, 410(12–13):1109–1128, 2009.

QG93. Xiaolei Qian and Allen Goldberg. Referential opacity in nondeterministic data refinement. *ACM Letters on Programming Languages and Systems*, 2(1–4):233–241, 1993.

Qia93. Zhenyu Qian. An algebraic semantics of higher-order types with subtypes. *Acta Informatica*, 30(6):569–607, 1993.

RAC99. Gianna Reggio, Egidio Astesiano, and Christine Choppy. CASL-LTL: a CASL extension for dynamic systems — summary. Technical Report DISI-TR-99-34, DISI, Università di Genova, 1999.

RB88. David Rydeheard and Rod Burstall. *Computational Category Theory*. Prentice Hall International Series in Computer Science. Prentice Hall, 1988.

Rei80. Horst Reichel. Initially-restricting algebraic theories. In Piotr Dembiński, editor, *Proceedings of the 9th Symposium on Mathematical Foundations of Computer Science*, *Lecture Notes in Computer Science*, volume 88, pages 504–514, Rydzyna, 1980. Springer.

Rei81. Horst Reichel. Behavioural equivalence — a unifying concept for initial and final specification methods. In *Proceedings of the 3rd Hungarian Computer Science Conference*, pages 27–39, 1981.

Rei85. Horst Reichel. Behavioural validity of equations in abstract data types. In *Proceedings of the Vienna Conference on Contributions to General Algebra*, pages 301–324. Teubner-Verlag, 1985.

Rei87. Horst Reichel. *Initial Computability, Algebraic Specifications, and Partial Algebras*. Oxford University Press, 1987.

RG98. Grigore Roşu and Joseph A. Goguen. Hidden congruent deduction. In Ricardo Caferra and Gernot Salzer, editors, *Proceedings of the 1998 Workshop on First-Order Theorem Proving*, Vienna, *Lecture Notes in Artificial Intelligence*, volume 1761, pages 251–266. Springer, 1998.

RG00. Grigore Roşu and Joseph A. Goguen. On equational Craig interpolation. *Journal of Universal Computer Science*, 6(1):194–200, 2000.

Rod91. Pieter Hendrik Rodenburg. A simple algebraic proof of the equational interpolation theorem. *Algebra Universalis*, 28:48–51, 1991.

Rog06. Markus Roggenbach. CSP-CASL — a new integration of process algebra and algebraic specification. *Theoretical Computer Science*, 354(1):42–71, 2006.

Roş94. Grigore Roşu. The institution of order-sorted equational logic. *Bulletin of the European Association for Theoretical Computer Science*, 53:250–255, 1994.

Roş00. Grigore Roşu. *Hidden Logic*. PhD thesis, University of California at San Diego, 2000.

RRS00. Sergei Romanenko, Claudio Russo, and Peter Sestoft. Moscow ML owner's manual. Technical report, Royal Veterinary and Agricultural University, Copenhagen, 2000. Available from `http://www.itu.dk/people/sestoft/mosml/manual.pdf`.

RS63. Helena Rasiowa and Roman Sikorski. *The Mathematics of Metamathematics*. Number 41 in Monografie Matematyczne. Polish Scientific Publishers, 1963.

Rus98. Claudio Russo. *Types for Modules*. PhD thesis, University of Edinburgh, Department of Computer Science, 1998. Also in: *Electronic Notes in Theoretical Computer Science*, 60, 2003.

Rut00. Jan J.M.M. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

San82. Donald Sannella. *Semantics, Implementation and Pragmatics of Clear, a Program Specification Language*. PhD thesis, University of Edinburgh, Department of Computer Science, 1982.

SB83. Donald Sannella and Rod Burstall. Structured theories in LCF. In Giorgio Ausiello and Marco Protasi, editors, *Proceedings of the 8th Colloquium on Trees in Algebra and Programming*, L'Aquila, *Lecture Notes in Computer Science*, volume 159, pages 377–391. Springer, 1983.

Sch86. David Schmidt. *Denotational Semantics: A Methodology for Language Development*. Allyn and Bacon, 1986.

Sch87. Oliver Schoett. *Data Abstraction and the Correctness of Modular Programs*. PhD thesis, University of Edinburgh, Department of Computer Science, 1987.

Sch90. Oliver Schoett. Behavioural correctness of data representations. *Science of Computer Programming*, 14(1):43–57, 1990.

Sch92. Oliver Schoett. Two impossibility theorems on behaviour specification of abstract data types. *Acta Informatica*, 29(6/7):595–621, 1992.

Sco76. Dana Scott. Data types as lattices. *SIAM Journal of Computing*, 5(3):522–587, 1976.

Sco04. Giuseppe Scollo. An institution isomorphism for planar graph colouring. In Rudolf Berghammer, Bernhard Möller, and Georg Struth, editors, *Relational and Kleene-Algebraic Methods in Computer Science. Selected Papers from the 7th International Seminar on Relational Methods in Computer Science and 2nd International Workshop on Applications of Kleene Algebra*, Bad Malente, *Lecture Notes in Computer Science*, volume 3051, pages 252–264. Springer, 2004.

SCS94. Amílcar Sernadas, José Félix Costa, and Cristina Sernadas. An institution of object behaviour. In Hartmut Ehrig and Fernando Orejas, editors, *Recent Trends in Data Type Specification. Selected Papers from the 9th Workshop on Specification of Abstract Data Types joint with the 4th* COMPASS *Workshop*, Caldes de Malavella, *Lecture Notes in Computer Science*, volume 785, pages 337–350. Springer, 1994.

Sel72. Alan Selman. Completeness of calculi for axiomatically defined classes of algebras. *Algebra Universalis*, 2:20–32, 1972.

SH00. Christopher A. Stone and Robert Harper. Deciding type equivalence in a language with singleton kinds. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages*, Boston, pages 214–227, 2000.

Sha08. Stewart Shapiro. Classical logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. CSLI, Stanford University, fall 2008 edition, 2008. Available from `http://plato.stanford.edu/archives/fall2008/entries/logic-classical/`.

SM09. Lutz Schröder and Till Mossakowski. HASCASL: Integrated higher-order specification and program development. *Theoretical Computer Science*, 410(12–13):1217–1260, 2009.

Smi93. Douglas R. Smith. Constructing specification morphisms. *Journal of Symbolic Computation*, 15(5/6):571–606, 1993.

Smi06. Douglas R. Smith. Composition by colimit and formal software development. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, *Lecture Notes in Computer Science*, volume 4060, pages 317–332. Springer, 2006.

SML05. Lutz Schröder, Till Mossakowski, and Christoph Lüth. Type class polymorphism in an institutional framework. In José Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques.. Selected Papers from the 17th International Workshop on Algebraic Development Techniques*, Barcelona, *Lecture Notes in Computer Science*, volume 3423, pages 234–248. Springer, 2005.

Smo86. Gert Smolka. Order-sorted Horn logic: Semantics and deduction. Technical Report SR-86-17, Universität Kaiserslautern, Fachbereich Informatik, 1986.

SMT$^+$05. Lutz Schröder, Till Mossakowski, Andrzej Tarlecki, Bartek Klin, and Piotr Hoffman. Amalgamation in the semantics of CASL. *Theoretical Computer Science*, 331(1):215–247, 2005.

Spi92. J. Michael Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International Series in Computer Science, second edition, 1992.

SS93. Antonino Salibra and Guiseppe Scollo. A soft stairway to institutions. In Michel Bidoit and Christine Choppy, editors, *Recent Trends in Data Type Specification. Selected Papers from the 8th Workshop on Specification of Abstract Data Types joint with the 3rd* COMPASS *Workshop*, Dourdan, *Lecture Notes in Computer Science*, volume 655, pages 310–329. Springer, 1993.

SS96. Antonino Salibra and Giuseppe Scollo. Interpolation and compactness in categories of pre-institutions. *Mathematical Structures in Computer Science*, 6(3):261–286, 1996.

SST92. Donald Sannella, Stefan Sokołowski, and Andrzej Tarlecki. Toward formal development of programs from algebraic specifications: Parameterisation revisited. *Acta Informatica*, 29(8):689–736, 1992.

ST85. Donald Sannella and Andrzej Tarlecki. Program specification and development in Standard ML. In *Proceedings of the 12th ACM Symposium on Principles of Programming Languages*, New Orleans, pages 67–77, 1985.

ST86. Donald Sannella and Andrzej Tarlecki. Extended ML: An institution-independent framework for formal program development. In David H. Pitt, Samson Abramsky, Axel Poigné, and David E. Rydeheard, editors, *Proceedings of the Tutorial and Workshop on Category Theory and Computer Programming*, Guildford, *Lecture Notes in Computer Science*, volume 240, pages 364–389. Springer, 1986.

ST87. Donald Sannella and Andrzej Tarlecki. On observational equivalence and algebraic specification. *Journal of Computer and System Sciences*, 34:150–178, 1987.

ST88a. Donald Sannella and Andrzej Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76(2/3):165–210, 1988.

ST88b. Donald Sannella and Andrzej Tarlecki. Toward formal development of programs from algebraic specifications: Implementations revisited. *Acta Informatica*, 25:233–281, 1988.

ST89. Donald Sannella and Andrzej Tarlecki. Toward formal development of ML programs: Foundations and methodology. In Josep Díaz and Fernando Orejas, editors, *TAPSOFT'89: Proceedings of the International Joint Conference on Theory and Practice of Software Development. Volume 2: Advanced Seminar on Foundations of Innovative Software Development II and Colloquium on Current Issues in Programming Languages*, Barcelona, *Lecture Notes in Computer Science*, volume 352, pages 375–389. Springer, 1989.

ST97. Donald Sannella and Andrzej Tarlecki. Essential concepts of algebraic specification and program development. *Formal Aspects of Computing*, 9:229–269, 1997.

ST04. Donald Sannella and Andrzej Tarlecki, editors. CASL semantics. In *[Mos04]*. 2004.

ST06. Donald Sannella and Andrzej Tarlecki. Horizontal composability revisited. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning and Computation: Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, *Lecture Notes in Computer Science*, volume 4060, pages 296–316. Springer, 2006.

ST08. Donald Sannella and Andrzej Tarlecki. Observability concepts in abstract data type specification, 30 years later. In Pierpaolo Degano, Rocco de Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models: Essays Dedicated to Ugo Montanari on the Occasion of his 65th Birthday*, Lecture Notes in Computer Science. Springer, 2008.

Str67. Christopher Strachey. Fundamental concepts in programming languages. In *NATO Summer School in Programming, Copenhagen*. 1967. Also in: *Higher-Order and Symbolic Computation* 13(1–2):11–49, 2000.

SU06. Morten H. Sørensen and Paweł Urzyczyn. *Lectures on the Curry-Howard Isomorphism*. Number 149 in Studies in Logic and the Foundations of Mathematics. Elsevier Science, 2006.

SW82. Donald Sannella and Martin Wirsing. Implementation of parameterised specifications. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Proceeding of the 9th International Colloquium on Automata, Languages and Programming*, Aarhus, *Lecture Notes in Computer Science*, volume 140, pages 473–488. Springer, 1982.

SW83. Donald Sannella and Martin Wirsing. A kernel language for algebraic specification and implementation. In Marek Karpinski, editor, *Proceedings of the 1983 International Conference on Foundations of Computation Theory*, Borgholm, *Lecture Notes in Computer Science*, volume 158, pages 413–427. Springer, 1983.

SW99. Donald Sannella and Martin Wirsing. Specification languages. In Egidio Astesiano, Hans-Jörg Kreowski, and Bernd Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, chapter 8, pages 243–272. Springer, 1999.

Tar85. Andrzej Tarlecki. On the existence of free models in abstract algebraic institutions. *Theoretical Computer Science*, 37(3):269–304, 1985.

Tar86a. Andrzej Tarlecki. Bits and pieces of the theory of institutions. In David H. Pitt, Samson Abramsky, Axel Poigné, and David E. Rydeheard, editors, *Proceedings of the Tutorial and Workshop on Category Theory and Computer Programming*, Guildford, *Lecture Notes in Computer Science*, volume 240, pages 334–360. Springer, 1986.

Tar86b. Andrzej Tarlecki. Quasi-varieties in abstract algebraic institutions. *Journal of Computer and System Sciences*, 33(3):333–360, 1986.

Tar87. Andrzej Tarlecki. Institution representation. Unpublished note, Dept. of Computer Science, University of Edinburgh, 1987.

Tar96. Andrzej Tarlecki. Moving between logical systems. In Magne Haveraaen, Olaf Owe, and Ole-Johan Dahl, editors, *Recent Trends in Data Type Specification. Selected Papers from the 11th Workshop on Specification of Abstract Data Types*, Oslo, *Lecture Notes in Computer Science*, volume 1130, pages 478–502. Springer, 1996.

Tar99. Andrzej Tarlecki. Institutions: An abstract framework for formal specification. In Egidio Astesiano, Hans-Jörg Kreowski, and Bernd Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, chapter 4, pages 105–130. Springer, 1999.

Tar00. Andrzej Tarlecki. Towards heterogeneous specifications. In Dov Gabbay and Maarten de Rijke, editors, *Frontiers of Combining Systems 2*, Studies in Logic and Computation, pages 337–360. Research Studies Press, 2000.

TBG91. Andrzej Tarlecki, Rod M. Burstall, and Joseph A. Goguen. Some fundamental algebraic tools for the semantics of computation. Part 3: Indexed categories. *Theoretical Computer Science*, 91(2):239–264, 1991.

Ter03. Terese. *Term Rewriting Systems*, *Cambridge Tracts in Theoretical Computer Science*, volume 55. Cambridge University Press, 2003.

Tho89. Simon Thompson. A logic for Miranda. *Formal Aspects of Computing*, 1(4):339–365, 1989.

TM87. Władysław M. Turski and Thomas S.E. Maibaum. *Specification of Computer Programs*. Addison-Wesley, 1987.

Tra93. Will Tracz. Parametrized programming in LILEANNA. In *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, Indianapolis, pages 77–86, 1993.

TWW82. James Thatcher, Eric Wagner, and Jesse Wright. Data type specification: Parameterization and the power of specification techniques. *ACM Transactions on Programming Languages and Systems*, 4(4):711–732, 1982.

Vra88. Jos L.M. Vrancken. The algebraic specification of semi-computable data types. In Donald Sannella and Andrzej Tarlecki, editors, *Recent Trends in Data Type Specification. Selected Papers from the 5th Workshop on Specification of Abstract Data Types*, Gullane, *Lecture Notes in Computer Science*, volume 332, pages 249–259. Springer, 1988.

Wad89. Philip Wadler. Theorems for free! In *Proceedings of the 4th International ACM Conference on Functional Programming Languages and Computer Architecture*, London, pages 347–359, 1989.

Wan79. Mitchell Wand. Final algebra semantics and data type extensions. *Journal of Computer and System Sciences*, 19:27–44, 1979.

Wan82. Mitchell Wand. Specifications, models, and implementations of data abstractions. *Theoretical Computer Science*, 20(1):3–32, 1982.

WB82. Martin Wirsing and Manfred Broy. An analysis of semantic models for algebraic specifications. In Manfred Broy and Gunther Schmidt, editors, *Theoretical Foundations of Programming Methodology: Lecture Notes of an International Summer School, Marktoberdorf 1981*, pages 351–416. Reidel, 1982.

WB89. Martin Wirsing and Manfred Broy. A modular framework for specification and implementation. In Josep Díaz and Fernando Orejas, editors, *TAPSOFT'89: Proceedings of the International Joint Conference on Theory and Practice of Software Development. Volume 1: Advanced Seminar on Foundations of Innovative Software Development I and Colloquium on Trees in Algebra and Programming*, Barcelona, *Lecture Notes in Computer Science*, volume 351, pages 42–73. Springer, 1989.

WE87. Eric G. Wagner and Hartmut Ehrig. Canonical constraints for parameterized data types. *Theoretical Computer Science*, 50:323–349, 1987.

Wec92. Wolfgang Wechler. *Universal Algebra for Computer Scientists*, *EATCS Monographs on Theoretical Computer Science*, volume 25. Springer, 1992.

Wik. Wikipedia. Hash table. Available from `http://en.wikipedia.org/wiki/Hash_table`.

Wir82. Martin Wirsing. Structured algebraic specifications. In *Proceedings of the AFCET Symposium on Mathematics for Computer Science*, Paris, pages 93–107, 1982.

Wir86. Martin Wirsing. Structured algebraic specifications: A kernel language. *Theoretical Computer Science*, 42(2):123–249, 1986.

Wir90. Martin Wirsing. Algebraic specification. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B (Formal Models and Semantics)*, pages 675–788. North-Holland and MIT Press, 1990.

Wir93. Martin Wirsing. Structured specifications: Syntax, semantics and proof calculus. In Friedrich L. Bauer, Wilfried Brauer, and Helmut Schwichtenberg, editors, *Logic and*

Algebra of Specification: Proceedings of the NATO Advanced Study Institute, Marktoberdorf 1991, pages 411–442. Springer, 1993.

WM97. Michal Walicki and Sigurd Meldal. Algebraic approches to nondeterminism: An overview. *ACM Computing Surveys*, 29(1):30–81, 1997.

Zil74. Steven Zilles. Abstract specification of data types. Technical Report 119, Computation Structures Group, Massachusetts Institute of Technology, 1974.

Zuc99. Elena Zucca. From static to dynamic abstract data-types: An institution transformation. *Theoretical Computer Science*, 216(1–2):109–157, 1999.