Relating CASL with Other Specification Languages: the Institution Level [1]

Till Mossakowski

Bremen Institute of Safe Systems, Department of Computer Science, Universität Bremen, Germany.

Abstract

In this work, we investigate various specification languages and their relation to CASL, the recently developed Common Algebraic Specification Language. In particular, we consider the languages Larch, OBJ3 and functional CafeOBJ, ACT ONE, ASF, and HEP-theories, as well as various sublanguages of CASL. All these languages are translated to an appropriate sublanguage of CASL.

The translation mainly concerns the level of specification in-the-small: the logics underlying the languages are formalized as institutions, and representations among the institutions are developed. However, it is also considered how these translations interact with specification in-the-large.

Thus, we obtain on the one hand translations of any of the abovementioned specification languages to an appropriate sublanguage of CASL. This allows us to take libraries and case studies that have been developed for other languages and re-use them in CASL.

On the other hand, we set up institution representations going from the CASL institution (and some of its subinstitutions) to simpler subinstitutions. Given a theorem proving tool for such a simpler subinstitution, with the help of a representation, it can also be used for a more complex institution. Thus, first-order theorem provers and conditional term rewriting tools become usable for CASL.

Key words: Specification languages, Institutions, Logics, Translations of logics.

Contents

[1] This research was supported by the ESPRIT-funded CoFI Working Group 29432 and by the DFG project MULTIPLE.

1 Introduction

CASL is a specification language that has been designed by CoFI, the international *Common Framework Initiative for algebraic specification and development* [65,21], with the goal to subsume many previous algebraic specification languages and to provide a standard language for the specification and development of modular software systems. The design of CASL is explained in another paper in this volume [4], see also [22]. Actually, CASL is a central language in a whole family of languages. CASL concentrates on the specification of abstract data types and (first-order) functional requirements, while some (currently still prototypical) *extensions* of CASL also consider the specification of higher-order functions [60,72] and of reactive [9,68,69] and object-oriented [3,43] behaviour. Several *restrictions* of CASL to sublanguages make it possible to use specialized tool support.

The aim of the present paper is to substantiate the claim that CASL subsumes many existing specification languages. We will consider the relation of both specification languages (Larch [40], OBJ3 and functional CafeOBJ [31,26], ACT ONE [20], ASF [11], HEP-theories [70]) and common tool-supported logics (e.g. first-order logic, conditional equational logic) to CASL. As a first step, we deal with specification in-the-small, i.e. unstructured specifications of individual software modules, formulated in a specific logic. We also study how these translations interact with the CASL concepts of specification in-the-large (although we do not study translations among *different* concepts of specification in-the-large). A clean separation of specification in-the-small and specification in-the-large is possible, because in CASL, both levels are treated separately: the semantics of CASL in-the-large is orthogonal to the underlying logic that is used for specification in-the-small.

For any practically usable specification language, there is a distinction between underlying mathematical concepts on the one hand, and language constructs on the other hand. Typically, the language constructs provide a concise and user-friendly syntax for writing specifications, and the semantics interprets the constructs in terms of the concepts. At the level of specification in-the-small, the essential mathematical concept is that of the *logic* underlying the language.

Here, we mainly address the problem of relating the underlying logics of the specification languages. To this end, we formalize the logics as *institutions* in the sense of Goguen and Burstall [33]. For some specification languages, the recognition of the underlying logic is not obvious, and the formalization as an institution is a non-trivial task. Once this has been done, the institutions can be related using *institution representations* [47,74]. In some cases, there is an obvious subinstitution of the CASL institution that closely corresponds to the institution underlying the specification language in question. We therefore single out a number of subinstitutions of CASL and develop a uniform naming scheme for them. In other cases, the relation is not so obvious, and there are several good choices for the institution representation.

We also address the translation of (subinstitutions of) CASL to other, simpler subinstitutions. Typically, the target of such a translation will be an institution with good tool support, and the translation will have the property that the tool support can be lifted against the translation (this has been called "borrowing" [18]). We also present some new results concerning such lifting properties.

The level of languages constructs is not discussed formally in this paper. This would require detailed language definitions, which would be too lengthy to be presented here (apart from the difficulty that not all of the languages are precisely defined, e.g. w.r.t. their static semantic conditions). Rather, we informally discuss how the institution representations lift to translations of

the corresponding language constructs. In many cases, this is straightforward. But there remain some situations where it is not advisable to first expand a construct into a theory of the underlying institution and then translate this theory. Instead, it is better to use a more direct translation of the constructs.

The structure of this paper is as follows: In Section 2, we recall the notions of institution and institution representation. We study various properties of institution representations that lead to a good interaction of flat and structured specifications with institution representations. In this context, we also develop some new results concerning structured specifications. In Section 3, we introduce the institution underlying CASL and a number of its subinstitutions, for which we develop a uniform naming scheme. Section 4 discusses representations of CASL in some of its subinstitutions such as first-order logic, and also representations among subinstitutions of CASL. We obtain three graphs of institutions and representations: one at the first-order level without sort generation constraints, one at the first-order level with sort generation constraints, and one at the positive conditional level. The graphs contain different kinds of edges corresponding to the different properties of institution representations introduced in Section 2. By combining these graphs with the results about interaction of institution representations with flat and structured specifications, we obtain a number of theorems about re-use of theorem provers and about the existence of free objects (liberality). In particular, we show that a theorem prover for first-order logic plus induction suffices to prove theorems within the CASL institution, even at the level of structured specifications including certain free specifications. Further, conditional term rewriting and paramodulation can be applied to the positive conditional fragment of CASL. Moreover, we show the latter to be liberal. Next, sections 6, 7, 8, 9, 10 are devoted to the definition and translation of the institutions underlying Larch, ACT ONE, ASF, HEP-theories, and OBJ3 and functional CafeOBJ, respectively. In each case, a separate subsection is devoted to an informal discussion of the level of language constructs. Section 11 contains some general remarks about boolean-valued functions versus predicates and about the empty carrier problem. Throughout the paper, only little knowledge about category theory is assumed, except from some parts of Section 2. These parts rely on some purely category theoretic results, which we have relegated to the appendix.

2 Institutions and Institution Representations

In this section, we introduce the notion of institution and different types of institution representations known from the literature. We summarize some known properties of these and prove new results concerning their interaction with specification in-the-large.

A specification formalism is usually based on some notion of signature, model, sentence and satisfaction. These are the usual ingredients of Barwise's abstract model theory [7]. Contrary to Barwise's notions, *institutions* of Goguen and Burstall [33] do not assume that signatures are algebraic signatures and thus cover a much larger variety of logics. Indeed, the theory of institutions assumes nothing about signatures except that they form a class and that there are *signature morphisms*, which can be composed in some way. This amounts to stating that signatures form a *category*.

There is also nothing special assumed about the form of the *sentences* and *models*. Given a signature Σ, the Σ-sentences form just a set, while the Σ-models form a category (taking into account that there may be *model morphisms*).

Signature morphisms lead to *translations* of sentences and of models (thus, the assignments of sentences and of models to signatures are functors). There is a contravariance between the sentence and the model translation: sentences are translated *along* signature morphisms, while models are translated *against* signature morphisms.

Informally, this can be motivated as follows. Forget for a moment the above generality and think of signatures as of sets of certain symbols. Think of sentences over a signature Σ as derivation trees over some grammar, decorated at the nodes with the symbols from Σ. Then sentence translation along a signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$ keeps the structure of the derivation tree, but replaces the symbols decorating the nodes, using σ. This explains why sentences are translated *along* signature morphisms.

Concerning models over a signature: they have to interpret the symbols from the signature somehow. Thus, a Σ-model can be seen as a map m going from the symbols of Σ to some semantical domain. Now given a Σ'-model m' and a signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$, by composing the interpretation map m' with σ we get a new interpretation map, let us call it $m'|_\sigma$, which is a Σ-model! ($m'|_\sigma$ is also called the *σ-reduct* of m'.) This explains why models are translated *against* signature morphisms.

Of course, these explanations just have motivating purpose: there can be institutions with a completely different view of signatures, models and sentences. However, they shed some light on how many typical institutions work. [2]

Finally, institutions have a *satisfaction relation* between models and sentences, which has to be invariant under the simultaneous translation of sentences and

[2] Indeed, the above explanation has been formalized as so-called *parchments* [55].

models w.r.t. a given signature morphism.

This leads to the following formal definition. Let $\mathcal{CAT}$ be the category of categories and functors. [3]

Definition 2.1 An *institution* $I = (\mathbf{Sign}^I, \mathbf{Sen}^I, \mathbf{Mod}^I, \models^I)$ consists of

- a category $\mathbf{Sign}^I$ of *signatures*,
- a functor $\mathbf{Sen}^I: \mathbf{Sign}^I \longrightarrow \mathbf{Set}$ giving, for each signature Σ, the set of *sentences* $\mathbf{Sen}^I(\Sigma)$, and for each signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$, the *sentence translation map* $\mathbf{Sen}^I(\sigma): \mathbf{Sen}^I(\Sigma) \longrightarrow \mathbf{Sen}^I(\Sigma')$, where often $\mathbf{Sen}^I(\sigma)(\varphi)$ is written as $\sigma(\varphi)$,
- a functor $\mathbf{Mod}^I: (\mathbf{Sign}^I)^{op} \longrightarrow \mathcal{CAT}$ giving, for each signature Σ, the category of *models* $\mathbf{Mod}^I(\Sigma)$, and for each signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$, the *reduct functor* $\mathbf{Mod}^I(\sigma): \mathbf{Mod}^I(\Sigma') \longrightarrow \mathbf{Mod}^I(\Sigma)$, where often $\mathbf{Mod}^I(\sigma)(M')$ is written as $M'|_\sigma$,
- a satisfaction relation $\models^I_\Sigma \subseteq |\mathbf{Mod}^I(\Sigma)| \times \mathbf{Sen}^I(\Sigma)$ for each $\Sigma \in \mathbf{Sign}^I$,

such that for each $\sigma: \Sigma \longrightarrow \Sigma'$ in $\mathbf{Sign}^I$ the following *satisfaction condition* holds:

$$M' \models^I_{\Sigma'} \sigma(\varphi) \Leftrightarrow M'|_\sigma \models^I_\Sigma \varphi$$

for each $M' \in \mathbf{Mod}^I(\Sigma')$ and $\varphi \in \mathbf{Sen}^I(\Sigma)$. □

We will omit the index I when it is clear from the context.

We now informally present some examples. They will be formally introduced in Section 3.

Example 2.2 The institution $Eq^=$ of equational logic. Signatures are many-sorted algebraic signatures consisting of a set of sorts and a set of function symbols (where each function symbol has a string of argument sorts and a result sort). Signature morphisms map sorts and function symbols in a compatible way. Models are just many-sorted algebras, i.e. each sort is interpreted as a carrier set, and each function symbol is interpreted as a function between the carrier sets specified by the argument and result sorts. Reducts are constructed as sketched above. Sentences are equations between many-sorted terms, and sentence translation means replacement of the translated symbols. Finally, satisfaction is the usual satisfaction of an equation in an algebra. □

Example 2.3 The institution $FOL^=$ of many-sorted first-order logic with equality. Signatures are many-sorted first-order signatures, i.e. many-sorted algebraic signatures enriched with predicate symbols. Models are many-sorted first-order structures. Sentences are first-order formulas, and again sentence

[3] Strictly speaking, $\mathcal{CAT}$ is not a category but only a so-called quasicategory, which is a category that lives in a higher set-theoretic universe [42].

translation means replacement of the translated symbols. Satisfaction is the usual satisfaction of a first-order sentence in a first-order structure. □

Example 2.4 The institution $PFOL^=$ of partial first-order logic with equality. Signatures are many-sorted first-order signatures enriched by partial function symbols. Models are many-sorted partial first-order structures. Sentences are first-order formulas containing existential equations, strong equations, definedness statements and predicate applications as atomic formulas. Satisfaction is defined using total valuations of variables, while valuation of terms is partial due to the existence of partial functions. An existential equation holds if both sides are defined and equal, whereas a strong equation also holds if both sides are undefined. A definedness statement holds if the term is defined. A predicate application holds if the terms contained in it are defined, and the corresponding tuple of values is in the interpretation of the predicate. This is extended to first-order formulas as usual. □

Within an arbitrary but fixed institution, we can easily define the usual notion of *logical consequence* or *semantical entailment*: Given a set of Σ-sentences Γ and a Σ-sentence φ, we say

$$\Gamma \models_\Sigma \varphi \text{ (φ follows from Γ)}$$

iff for all Σ-models M, we have

$$M \models_\Sigma \Gamma \text{ implies } M \models_\Sigma \varphi.$$

Here, $M \models_\Sigma \Gamma$ means that $M \models_\Sigma \psi$ for each $\psi \in \Gamma$.

We will also freely use other standard logical terminology when working within an arbitrary but fixed institution.

2.2 Specifications Over an Arbitrary Institution

This paper mainly concentrates on translating specification languages at the level of specification in-the-small. However, when translating the underlying logics of specification languages, the question arises how this interacts with specification in-the-large, for a given *fixed* set of structuring operations for specification in-the-large. Therefore, in this section we recall a popular set of institution-independent structuring operations, which seems to be quite universal and which can also be seen as a kernel language for the CASL structuring constructs. The question how this would relate to other sets of structuring operations is beyond the scope of the present paper.

In the sequel, let us fix an arbitrary institution $I = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$.

The simplest specifications over an arbitrary institution are just *theories* $T = \langle \Sigma, \Gamma \rangle$, where $\Sigma \in \mathbf{Sign}$ and $\Gamma \subseteq \mathbf{Sen}(\Sigma)$ (we set $\mathbf{Sig}(T) = \Sigma$ and $\mathbf{Ax}(T) = \Gamma$). *Theory morphisms* $\sigma\colon \langle \Sigma, \Gamma \rangle \longrightarrow \langle \Sigma', \Gamma' \rangle$ are those signature morphisms $\sigma\colon \Sigma \longrightarrow \Sigma'$ for which $\Gamma' \models_{\Sigma'} \sigma(\Gamma)$, that is, axioms are mapped to logical consequences. By inheriting composition and identities from **Sign**, we obtain a category **Th** of theories. It is easy to extend **Sen** and **Mod** to start from **Th** by putting $\mathbf{Sen}(\langle \Sigma, \Gamma \rangle) = \mathbf{Sen}(\Sigma)$ and letting $\mathbf{Mod}(\langle \Sigma, \Gamma \rangle)$ be the full subcategory of $\mathbf{Mod}(\Sigma)$ induced by the class of those models M satisfying Γ. The category **Pres** of *presentations* (also called *flat specifications*) is just the full subcategory of theories having finite sets of axioms.

Concerning more complex specifications, in [71], the following kernel language for specifications in an arbitrary institution has been proposed. Simultaneously with the notion of specification, we define functions **Sig** and **Mod** yielding the signature and the model class of a specification.

***presentations*:** For any signature $\Sigma \in |\mathbf{Sign}|$ and finite set $\Gamma \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences, the *presentation* $\langle \Sigma, \Gamma \rangle$ is a specification with:

$\mathbf{Sig}(\langle \Sigma, \Gamma \rangle) := \Sigma$
$\mathbf{Mod}(\langle \Sigma, \Gamma \rangle) := \{M \in \mathbf{Mod}(\Sigma) \mid M \models \Gamma\}$

***union*:** For any signature $\Sigma \in |\mathbf{Sign}|$, given Σ-specifications SP_1 and SP_2, their *union* $SP_1 \cup SP_2$ is a specification with:

$\mathbf{Sig}(SP_1 \cup SP_2) := \Sigma$
$\mathbf{Mod}(SP_1 \cup SP_2) := \mathbf{Mod}(SP_1) \cap \mathbf{Mod}(SP_2)$

***translation*:** For any signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$ and Σ-specification SP, **translate** SP **by** σ is a specification with:

$\mathbf{Sig}(\textbf{translate } SP \textbf{ by } \sigma) := \Sigma'$
$\mathbf{Mod}(\textbf{translate } SP \textbf{ by } \sigma) := \{M' \in \mathbf{Mod}(\Sigma') \mid M'|_\sigma \in \mathbf{Mod}(SP)\}$

***hiding*:** For any signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$ and Σ'-specification SP', **derive from** SP' **by** σ is a specification with:

$\mathbf{Sig}(\textbf{derive from } SP' \textbf{ by } \sigma) := \Sigma$
$\mathbf{Mod}(\textbf{derive from } SP' \textbf{ by } \sigma) := \{M'|_\sigma \mid M' \in \mathbf{Mod}(SP')\}$

The above *specification-building operations*, although extremely simple, already provide flexible mechanisms for expressing basic ways of putting specifications together and thus building specifications in a structured manner. Hence, they can be considered to be a kernel language for structured specification. The specification language CASL provides more sophisticated structuring constructs, but it is possible to translate the CASL constructs (except the **free** construct, which will be examined in the next section) to the above kernel language, see [59].

A specification SP is said to be *consistent*, if $\mathbf{Mod}(SP)$ is not empty.

Given two structured specifications SP_1 and SP_2, a *specification morphism* $\sigma: SP_1 \longrightarrow SP_2$ is a signature morphism $\sigma: \mathbf{Sig}(SP_1) \longrightarrow \mathbf{Sig}(SP_2)$ such that $M|_\sigma \in \mathbf{Mod}(SP_1)$ for each $M \in \mathbf{Mod}(SP_2)$. By the satisfaction condition, each theory morphism between presentations also is a specification morphism.

A structured Σ-specification SP_2 *refines* a structured Σ-specification SP_1 (written $SP_1 \rightsquigarrow SP_2$), if $\mathbf{Mod}(SP_2) \subseteq \mathbf{Mod}(SP_1)$.

A specification not containing **derive** is called *flattenable*. This is because it is easy to normalize flattenable specifications into flat specifications [10]:

Definition 2.5 Given a flattenable specification SP over an institution I, its *normal form* $NF(SP)$ is inductively defined as follows:

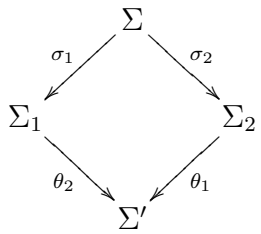
- $NF(\langle \Sigma, \Gamma \rangle) := \langle \Sigma, \Gamma \rangle$
- If $SP = SP_1 \cup SP_2$, let $NF(SP_i) = \langle \Sigma, \Gamma_i \rangle$ $(i = 1, 2)$. Then $NF(SP) := \langle \Sigma, \Gamma_1 \cup \Gamma_2 \rangle$.
- If $SP =$ **translate** SP_1 **by** $\sigma: \Sigma_1 \longrightarrow \Sigma$, let $NF(SP_1) = \langle \Sigma_1, \Gamma_1 \rangle$. Then $NF(SP) := \langle \Sigma, \sigma(\Gamma_1) \rangle$. $\square$

Fact 2.6 If SP is a flattenable specification, then

$$\mathbf{Mod}(SP) = \mathbf{Mod}(NF(SP)). \qquad \square$$

To be able to introduce normal forms of non-flattenable specifications, we need the notion of weak amalgamation:

Definition 2.7 An institution $I = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$ admits *weak amalgamation*, if **Sign** has pushouts and, moreover, given any pushout



any Σ_1-model M_1 and any Σ_2-model M_2 with $M_1|_{\sigma_1} = M_2|_{\sigma_2}$, there exists some Σ'-model M' with $M'|_{\theta_2} = M_1$ and $M'|_{\theta_1} = M_2$. $\square$

It is well known that in institutions with weak amalgamation, there is a normal form for specifications [29,10,12], having the format

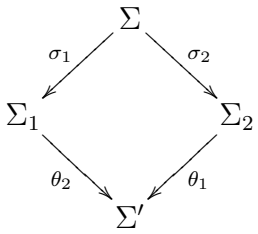
$$\textbf{derive from } \langle \Sigma, \Gamma \rangle \textbf{ by } \sigma.$$

Since the construction of a normal form involves pushouts and these are only determined up to isomorphism, also the normal form is only determined up to

signature isomorphism. We therefore define a relation "SP' is a normal form of SP".

Definition 2.8 The relation "is a normal form of" is the least relation satisfying:

- **derive from** $\langle \Sigma, \Gamma \rangle$ **by** id is a normal form of $\langle \Sigma, \Gamma \rangle$.
- If $SP = SP_1 \cup SP_2$, let **derive from** $\langle \Sigma_i, \Gamma_i \rangle$ **by** σ_i be a normal form of SP_i $(i = 1, 2)$, and let



  be a pushout. Then **derive from** $\langle \Sigma', \theta_2(\Gamma_1) \cup \theta_1(\Gamma_2) \rangle$ **by** $\theta_2 \circ \sigma_1$ is a normal form of SP.
- If $SP =$ **translate** SP' **by** $\sigma_1 \colon \Sigma \longrightarrow \Sigma_1$, let **derive from** $\langle \Sigma_2, \Gamma_2 \rangle$ **by** σ_2 be a normal form of SP', and take the pushout of σ_1 and σ_2 as above. Then **derive from** $\langle \Sigma', \theta_1(\Gamma_2) \rangle$ **by** θ_2 is a normal form of SP.
- If $SP =$ **derive from** SP' **by** $\sigma_1 \colon \Sigma \longrightarrow \Sigma_1$, let **derive from** $\langle \Sigma_2, \Gamma_2 \rangle$ **by** σ_2 be a normal form of SP'. Then **derive from** $\langle \Sigma_2, \Gamma_2 \rangle$ **by** $\sigma_2 \circ \sigma_1$ is a normal form of SP. $\square$

Fact 2.9 Assume that we work with specifications over an institution with weak amalgamation. Let SP' be a normal form of SP. Then

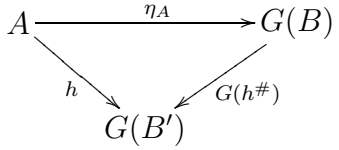
$$\mathbf{Mod}(SP) = \mathbf{Mod}(SP'). \qquad \square$$

2.3 Free extensions and liberality

Another institution independent structuring construct is that of *free extensions*. We treat it separately because in many structuring languages, it is not included, and moreover, it is not preserved so well along institution representations (see Section 2.4).

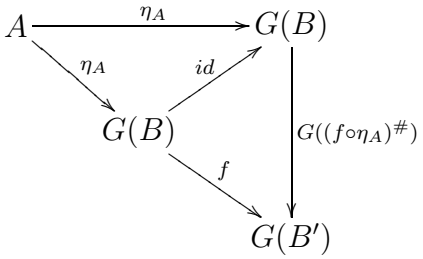
Free extensions can be defined w.r.t. an arbitrary functor: Given categories $\mathbf{A}$ and $\mathbf{B}$ and a functor $G \colon \mathbf{B} \longrightarrow \mathbf{A}$, an object $B \in \mathbf{B}$ is called *G-free (with unit $\eta_A \colon A \longrightarrow G(B)$) over $A \in \mathbf{A}$*, if for any object $B' \in \mathbf{B}$ and any morphism $h \colon A \longrightarrow G(B')$, there is a unique morphism $h^{\#} \colon B \longrightarrow B'$ such that $G(h^{\#}) \circ$

$\eta_A = h$.



In this case, the unit η_A is called a *G-universal arrow*. We will mostly omit the specification of the unit. An object $B \in \mathbf{B}$ is called *persistently G-free*, if it is G-free over some $A \in \mathbf{A}$ with the unit being an isomorphism. It is called *strongly persistently G-free* if it is G-free with unit id over $G(B)$ (id denotes the identity).

Proposition 2.10 Given a functor $G\colon \mathbf{B} \longrightarrow \mathbf{A}$, an object $B \in \mathbf{B}$ is persistently G-free if and only if it is strongly persistently G-free.



Proof. The "if" direction is clear. For the "only if" direction, let $\eta_A\colon A \longrightarrow G(B)$ be a G-universal isomorphism. If $f\colon G(B) \longrightarrow G(B')$ is a morphism, $(f \circ \eta_A)^\#$ is the unique morphism $g\colon B \longrightarrow B'$ with $id \circ G(g) = f$. Hence, $id\colon G(B) \longrightarrow G(B)$ is G-universal as well. □

We now extend the kernel language of the previous section as follows. For any signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$ and Σ'-specification SP', **free** SP' **along** σ is a specification with:

$$\begin{aligned}&\mathbf{Sig}(\textbf{free } SP' \textbf{ along } \sigma) = \Sigma'\\ &\mathbf{Mod}(\textbf{free } SP' \textbf{ along } \sigma) = \{M' \in \mathbf{Mod}(SP') \mid\\ &\qquad M' \text{ is strongly persistently } (\mathbf{Mod}(\sigma)\colon \mathbf{Mod}(SP') \longrightarrow \mathbf{Mod}(\Sigma))\text{-free } \}\end{aligned}$$

These specifications are called **data** SP' **over** σ in [71]. Since the unit has to be the identity, the freeness condition for M' means that for any model $N' \in \mathbf{Mod}(SP')$ and any model morphism $h\colon M'|_\sigma \longrightarrow N'|_\sigma$, there is a unique morphism $h^\#\colon M' \longrightarrow N'$ such that $(h^\#)|_\sigma = h$. By Proposition 2.10, we can equally use just persistent freeness instead of strongly persistent freeness.

Free extensions allow one to express certain inductive properties in a concise way. For example, the transitive closure of an arbitrary relation can be specified using the **free** construct in CASL as follows:

Example 2.11

spec BINARYRELATION =
  sort *Elem*
  pred $\_\_ \sim \_\_ : Elem \times Elem$
end

spec TRANSITIVECLOSURE [BINARYRELATION] =
  free
  { **pred** $\_\_ \sim^* \_\_ : Elem \times Elem$
    $\forall\ x, y, z : Elem$
    • $x \sim y \Rightarrow x \sim^* y$
    • $x \sim^* y \wedge y \sim^* z \Rightarrow x \sim^* z$
  }
end □

The corresponding structured specification is constructed as follows: Let $\langle \Sigma', \Gamma' \rangle$ be the presentation consisting of all sorts, predicates and axioms declared in either of BINARYRELATION and TRANSITIVECLOSURE, and let Σ be the signature of BINARYRELATION. Then as denotation of the above specification, we get

$$\textbf{free}\ \langle \Sigma', \Gamma' \rangle\ \textbf{along}\ \sigma$$

where σ is the inclusion of Σ into Σ'.

Another use of the free construct is in the generation of datatypes. For examples, consider the specification of finite sets over arbitrary elements in CASL:

Example 2.12

spec GENERATEFINITESET [**sort** *Elem*] =
free
  { **type** $FinSet[Elem] ::= \{\} \mid \{\_\_\}(Elem) \mid \_\_ \cup \_\_(FinSet[Elem]; FinSet[Elem])$
    op $\_\_ \cup \_\_ : FinSet[Elem] \times FinSet[Elem] \to FinSet[Elem]$,
      assoc, **comm**, **idem**, **unit** $\{\}$
  }
end

This expands to the following:

spec GENERATEFINITESET [**sort** *Elem*] =
free
  { **sort** $FinSet[Elem]$
    ops $\{\} : FinSet[Elem]$;
      $\{\_\_\} : Elem \to FinSet[Elem]$;
      $\_\_ \cup \_\_ : FinSet[Elem] \times FinSet[Elem] \to FinSet[Elem]$
    forall $x, y, z : Elem$
    • $x \cup (y \cup z) = (x \cup y) \cup z$

- $x \cup y = y \cup x$
- $x \cup x = x$
- $x \cup \{\} = x$

}
end

The question whether free models actually exist leads to the notion of liberality [33]:

Definition 2.13 Given an institution $I = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$,

- a theory morphism $\sigma: T \longrightarrow T'$ is said to be *liberal* if, for each T-model M, there is a T'-model M' that is $\mathbf{Mod}(\sigma)$-free over M; it is called *strongly persistently liberal* if, moreover, M' is strongly persistently $\mathbf{Mod}(\sigma)$-free;
- the institution I is called liberal if each of its theory morphisms is liberal;
- given a class $\mathcal{M}$ of theory morphisms in I, I is called $\mathcal{M}$-liberal if each theory morphism in $\mathcal{M}$ is liberal. □

The notion of (strongly persistent) liberality can easily be extended from theory morphisms to specification morphisms.

One could guess that in a liberal institution, **free** SP **along** σ is consistent whenever SP is. However, this is not the case, as the following counterexample shows:

Example 2.14 The institution $Eq^{=}$ of equational logic is known to be liberal[4] (see [33] and Theorem 4.16 below). Let Σ consist of one sort s and one constant $c : s$, let Σ' be Σ plus one unary function $f: s \longrightarrow s$, and let $\sigma: \Sigma \longrightarrow \Sigma'$ be the inclusion. Clearly $\langle \Sigma', \emptyset \rangle$ is consistent. However, **free** $\langle \Sigma', \emptyset \rangle$ **along** σ is inconsistent: any Σ-model is freely extended by adding an ω-chain

$$\{\, f(a),\ f(f(a)),\ f(f(f(a))),\ \ldots \}$$

over each of its elements a. Thus, a Σ'-model can never be free over its own σ-reduct. □

To show consistency of **free** SP **along** σ, we need strongly persistent liberality:

Proposition 2.15 If $\sigma: \langle \Sigma, \emptyset \rangle \longrightarrow \langle \Sigma', \Gamma' \rangle$ is strongly persistently liberal and $\langle \Sigma', \Gamma' \rangle$ is consistent, then also

$$\mathbf{free}\ \langle \Sigma', \Gamma' \rangle\ \mathbf{along}\ \sigma$$

[4] One has to allow empty carrier sets or restrict oneself to strict theory morphisms to get this result, see Section 11.2.

is consistent.

Proof. Let $M'_1 \in \mathbf{Mod}(\langle \Sigma', \Gamma' \rangle)$ by consistency, and let M' be $\mathbf{Mod}(\sigma)$-free over $M'_1|_\sigma$ with M' also being strongly persistently $\mathbf{Mod}(\sigma)$-free. Then $M' \in \mathbf{Mod}(\textbf{free}\ \langle \Sigma', \Gamma' \rangle\ \textbf{along}\ \sigma)$, and hence, $\textbf{free}\ \langle \Sigma', \Gamma' \rangle\ \textbf{along}\ \sigma$ is consistent. □

For specifications containing **free**, it is not so easy to obtain a normal form. This is because in general **free** does not commute with the other specification building operations.

2.4 Institution Representations

In order to relate sublanguages of CASL, we relate their underlying institutions. We therefore use the notion of institution representation (also called simple map of institutions) [47,74]. The idea behind an institution representation is to *encode* an institution I within an institution J. A *simple institution representation* from an institution I to an institution J consists of the following components:

- a translation Φ of I-signatures to J-presentations. Given an I-signature Σ, the task is to find a J-encoding $\Phi(\Sigma)$ of Σ in some way. In particular, the model category of $\Phi(\Sigma)$ should approximate the model category of Σ somehow. Consider, for example, the translation of $PFOL^=$ to $FOL^=$ (cf. Examples 2.3 and 2.4). Here, a partial function symbol can be encoded by a total function symbol, assuming that undefinedness is encoded by extra error values in the models. Thus, we would also need definedness predicates (one for each sort) which distinguish the "defined" values from the "undefined" values. Now let us assume that our partial logic is strict. Then we have to assume that in the encoding, the total functions return a "defined" result only for "defined" arguments. This can be expressed by an axiom, which should be included into $\Phi(\Sigma)$. This explains why $\Phi(\Sigma)$ should be a *presentation*, and not just a signature.
- a translation α of I-sentences to J-sentences. The reason why the sentence translation goes *along* with the signature translation is similar to the reason why the sentence translation *within* an institution goes along with the signature morphism. Namely, if a signature Σ in I is encoded by the presentation $\Phi(\Sigma)$ in J, it is expected that each symbol in Σ is translated to some corresponding symbol in $\Phi(\Sigma)$. Now if we assume that a Σ-sentence φ is a derivation tree decorated with some symbols from Σ, the translation $\alpha_\Sigma(\varphi)$ just keeps the structure of the tree and translates the symbols according to the correspondence of symbols in Σ and $\Phi(\Sigma)$. In the example above, α_Σ would just replace partial function symbols by total function symbols.

- a translation β of J-models to I-models, giving the above mentioned relation between Σ-models in I and $\Phi(\Sigma)$-models in J. Here, we again have the contravariance of the model translation, as in the definition of institution. For example, we can extract a Σ-model $\beta_\Sigma(M')$ with strict partial functions out of a $\Phi(\Sigma)$-model M' with total functions reflecting "defined" values as sketched above: we just take as carriers of the partial model the interpretations of the definedness predicates in the total model, while the total functions are restricted to these new carriers, yielding partial functions. Often it happens that there is also a model translation γ in the opposite direction. In the example, one would totalize a partial model by adding one element to each carrier, representing "undefined". However, while β can be formalized as a natural transformation, γ is not always natural (see [44] for a counterexample). But naturality of β is essential if we want to translate structured specifications along the institution representation (see Theorems 2.31, 2.35, 2.36, 2.43, 2.45, 2.49 and 2.51 below). This explains the direction of the model translation.

We also have a condition analogous to the satisfaction condition: we require that a translated model satisfies a sentence iff the original model satisfies the translated sentence. This is called the *representation condition*.

More formally, given institutions I and J, a *simple institution representation* [74,54] (also called *simple map of institutions* [47]) $\mu = (\Phi, \alpha, \beta): I \longrightarrow J$ consists of

- a functor $\Phi: \mathbf{Sign}^I \longrightarrow \mathbf{Pres}^J$ [5],
- a natural transformation $\alpha: \mathbf{Sen}^I \longrightarrow \mathbf{Sen}^J \circ \Phi$,
- a natural transformation $\beta: \mathbf{Mod}^J \circ \Phi^{op} \longrightarrow \mathbf{Mod}^I$

such that the following *representation condition* is satisfied for all $\Sigma \in \mathbf{Sign}^I$, $M' \in \mathbf{Mod}^J(\Phi(\Sigma))$ and $\varphi \in \mathbf{Sen}^I(\Sigma)$:

$$M' \models^J_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\varphi) \Leftrightarrow \beta_\Sigma(M') \models^I_\Sigma \varphi.$$

In more detail, this means that each signature $\Sigma \in \mathbf{Sign}^I$ is translated to a presentation $\Phi(\Sigma) \in \mathbf{Pres}^J$, and each signature morphism $\sigma: \Sigma \longrightarrow \Sigma' \in \mathbf{Sign}^I$ is translated to a presentation morphism $\Phi(\sigma): \Phi(\Sigma) \longrightarrow \Phi(\Sigma') \in \mathbf{Pres}^J$. Moreover, for each signature $\Sigma \in \mathbf{Sign}^I$, we have a sentence translation map $\alpha_\Sigma: \mathbf{Sen}^I(\Sigma) \longrightarrow \mathbf{Sen}^J(\Phi(\Sigma))$ and a model translation functor $\beta_\Sigma : \mathbf{Mod}^J(\Phi(\Sigma)) \longrightarrow \mathbf{Mod}^I(\Sigma)$. Naturality of α and β means that for any signature morphism

[5] Meseguer [47] requires $\Phi: \mathbf{Th}^I \longrightarrow \mathbf{Th}^J$, but since μ is simple, both formulations are equivalent using Meseguer's α-extension (except for the fact that we use presentations instead of theories).

$\sigma: \Sigma \longrightarrow \Sigma' \in \mathbf{Sign}^I$,

$$\begin{array}{ccc} \mathbf{Sen}^I(\Sigma) & \xrightarrow{\alpha_\Sigma} & \mathbf{Sen}^J(\Phi(\Sigma)) \\ \downarrow{\scriptstyle \mathbf{Sen}^I(\sigma)} & & \downarrow{\scriptstyle \mathbf{Sen}^J(\Phi(\sigma))} \\ \mathbf{Sen}^I(\Sigma') & \xrightarrow{\alpha_{\Sigma'}} & \mathbf{Sen}^J(\Phi(\Sigma')) \end{array}$$

and

$$\begin{array}{ccc} \mathbf{Mod}^I(\Sigma) & \xleftarrow{\beta_\Sigma} & \mathbf{Mod}^J(\Phi(\Sigma)) \\ \uparrow{\scriptstyle \mathbf{Mod}^I(\sigma)} & & \uparrow{\scriptstyle \mathbf{Mod}^J(\Phi(\sigma))} \\ \mathbf{Mod}^I(\Sigma') & \xleftarrow{\beta_{\Sigma'}} & \mathbf{Mod}^J(\Phi(\Sigma')) \end{array}$$

commute.

We illustrate this definition with some informal examples. They will be formally introduced in Section 4.

Example 2.16 There is a simple institution representation going from equational logic to first-order logic with equality. An algebraic signature is translated to a first-order signature by just taking the set of predicates to be empty. Sentence translation is just inclusion of equations into first-order sentences. A first-order model with empty set of predicates is translated by just considering it as an algebra. □

Example 2.17 Institution representations capture the encoding of a richer institution into a poorer one, as the following example (that has already been sketched above) shows: Define a simple institution representation going from partial first-order logic with equality to first-order logic with equality as follows: A partial first-order signature is translated to a total one by encoding each partial function symbol as a total one, plus a (new) unary predicate D ("definedness") and a (new) function symbol $\bot$ ("undefined") for each sort (this means that $\bot$ and D are heavily overloaded). Furthermore, we add axioms stating that D does not hold on $\bot$, and that (encoded) total functions preserve ("totality") and reflect ("strictness") D, while partial functions only reflect D (and the holding of predicates implies D to hold on the arguments). Sentence translation is done by replacing all partial function symbols by the total functions symbols encoding them, replacing strong equations $t = u$ by $(D(t) \vee D(u)) \Rightarrow t = u$, existence equations by conjunctions of the equation and the definedness (using D) of one of the sides of the equation, replacing definedness with D, and leaving predicate symbols as they are. For a given total model of the translated signature, we just take as carriers of the partial model the interpretations of the definedness predicates in the total model,

while the total functions are restricted to these new carriers, yielding partial functions. □

Definition 2.18 Given a simple institution representation $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$, we can extend it to a translation $\hat{\mu}$ of structured specifications, extending the definition in [75,12]:

- If SP is a Σ-specification of form $\langle \Sigma, \Gamma \rangle$, then

$$\hat{\mu}(SP) = \langle \mathbf{Sig}(\Phi(\Sigma)), \mathbf{Ax}(\Phi(\Sigma)) \cup \alpha_\Sigma(\Gamma) \rangle.$$

- If SP is a Σ-specification of form $SP_1 \cup SP_2$, then

$$\hat{\mu}(SP) = \hat{\mu}(SP_1) \cup \hat{\mu}(SP_2).$$

- If SP is a Σ-specification of form **translate** SP' **by** $\sigma\colon \Sigma \longrightarrow \Sigma'$, then

$$\hat{\mu}(SP) = (\textbf{translate } \hat{\mu}(SP') \textbf{ by } \Phi(\sigma)) \cup \Phi(\Sigma').$$

- If SP is a Σ-specification of form **derive from** SP' **by** σ, then

$$\hat{\mu}(SP) = \textbf{derive from } \hat{\mu}(SP') \textbf{ by } \Phi(\sigma).$$

- If SP is a Σ-specification of form **free** SP' **along** σ, then

$$\hat{\mu}(SP') = \textbf{free } \hat{\mu}(SP') \textbf{ along } \Phi(\sigma).$$

□

Note that $\Phi(\sigma)$ as used above is a morphism between presentations, but as such it is also a signature morphism. Further note that the axioms in $\mathbf{Ax}(\Phi(\Sigma))$ are added for presentations and also for translations, since the latter lead to an extension of the signature (while union, **derive** and **free** do not extend the signature). This will be crucial for a good interaction with institution representations (Theorem 2.31).

We now come to the interaction of $\hat{\mu}$ with normal forms.

Proposition 2.19 Let $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ be an institution representation and SP a flattenable specification in I. Then

$$NF(\hat{\mu}(SP)) = \hat{\mu}(NF(SP)).$$

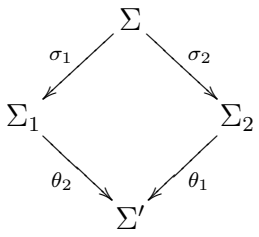
Proof. By induction over the structure of SP.

- $SP = \langle \Sigma, \Gamma \rangle$:
  $NF(\hat{\mu}(SP)) = \hat{\mu}(SP) = \hat{\mu}(NF(SP))$.

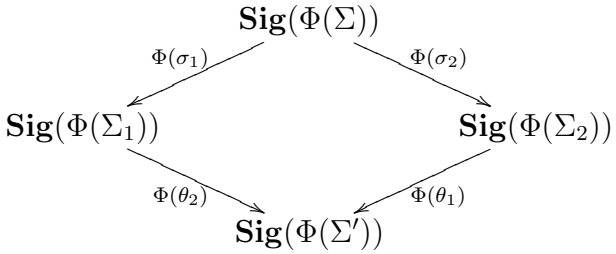
- $SP = SP_1 \cup SP_2$:
  Let $NF(SP_i) = \langle \Sigma, \Gamma_i \rangle$ $(i = 1, 2)$. By induction hypothesis, $NF(\hat{\mu}(SP_i)) = \hat{\mu}(NF(SP_i)) = \langle \mathbf{Sig}(\Phi(\Sigma)), \mathbf{Ax}(\Phi(\Sigma)) \cup \alpha_\Sigma(\Gamma_i) \rangle$. Therefore, $NF(\hat{\mu}(SP)) = \langle \mathbf{Sig}(\Phi(\Sigma)), \mathbf{Ax}(\Phi(\Sigma)) \cup \alpha_\Sigma(\Gamma_1) \cup \alpha_\Sigma(\Gamma_2) \rangle$. But this is $\hat{\mu}(NF(SP))$.
- $SP =$ **translate** SP_1 **by** $\sigma: \Sigma_1 \longrightarrow \Sigma$:
  Let $NF(SP_1) = \langle \Sigma_1, \Gamma_1 \rangle$, then $NF(SP) = \langle \Sigma, \sigma(\Gamma_1) \rangle$. By induction hypothesis, $NF(\hat{\mu}(SP_1)) = \hat{\mu}(NF(SP_1)) = \langle \mathbf{Sig}(\Phi(\Sigma_1)), \mathbf{Ax}(\Phi(\Sigma_1)) \cup \alpha_{\Sigma_1}(\Gamma_1) \rangle$. From this, we get $NF(\hat{\mu}(SP)) = NF($**translate** $\hat{\mu}(SP_1)$ **by** $\Phi(\sigma) \cup \Phi(\Sigma)) = \langle \mathbf{Sig}(\Phi(\Sigma)), \Phi(\sigma)(\mathbf{Ax}(\Phi(\Sigma_1)) \cup \alpha_{\Sigma_1}(\Gamma_1)) \cup \mathbf{Ax}(\Phi(\Sigma)) \rangle$. Since $\Phi(\sigma)(\mathbf{Ax}(\Phi(\Sigma_1))) \subseteq \mathbf{Ax}(\Phi(\Sigma))$ and α is natural, this is equal to $\langle \mathbf{Sig}(\Phi(\Sigma)), \alpha_\Sigma(\sigma(\Gamma_1)) \cup \mathbf{Ax}(\Phi(\Sigma)) \rangle$. But this is just $\hat{\mu}(NF(SP))$. $\square$

In order to extend this result to non-flattenable specifications, we need a preparatory lemma:

Lemma 2.20 Given institutions I and J and a functor $\Phi: \mathbf{Sign}^I \longrightarrow \mathbf{Pres}^J$ preserving pushouts and a pushout



in $\mathbf{Sign}^I$, then



is a pushout in $\mathbf{Sign}^J$, and

$$\mathbf{Ax}(\Phi(\Sigma')) = \Phi(\theta_2)(\mathbf{Ax}(\Phi(\Sigma_1))) \cup \Phi(\theta_1)(\mathbf{Ax}(\Phi(\Sigma_2))).$$

Proof. Follows from the construction of colimits of presentationss, see [27,33]. $\square$

We now generalize a result from [12] to the case of institution representations that map signatures to theories (and not just signatures):

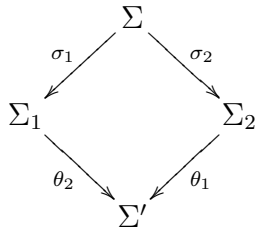
Proposition 2.21 Let I and J be two institutions having weak amalgamation, and let $\mu = (\Phi, \alpha, \beta)$ be an institution representation such that Φ pre-

serves pushouts. Furthermore, let SP be a specification not containing **free**. Then we have:

If SP' is a normal form of SP, then $\hat{\mu}(SP')$ is a normal form of $\hat{\mu}(SP)$.

Proof. By induction over the structure of SP. Since the proof very much resembles the proof of Theorem 5.14 in [12], we only treat the case of translation here.

If $SP = \textbf{translate}\ SP_0\ \textbf{by}\ \sigma_1 \colon \Sigma \longrightarrow \Sigma_1$, let $\textbf{derive from}\ \langle \Sigma_2, \Gamma_2 \rangle\ \textbf{by}\ \sigma_2$ be a normal form of SP_0. Now let



be a pushout. Then $SP' = \textbf{derive from}\ \langle \Sigma', \theta_1(\Gamma_2) \rangle\ \textbf{by}\ \theta_2$ is w.l.o.g. the normal form of SP, and $\hat{\mu}(SP')$ is

$$\textbf{derive from}\ \langle \textbf{Sig}(\Phi(\Sigma')), \textbf{Ax}(\Phi(\Sigma')) \cup \alpha_{\Sigma'}(\theta_1(\Gamma_2)) \rangle\ \textbf{by}\ \Phi(\theta_2).$$

By induction hypothesis, $\hat{\mu}(\textbf{derive from}\ \langle \Sigma_2, \Gamma_2 \rangle\ \textbf{by}\ \sigma_2) =$

$$\textbf{derive from}\ \langle \textbf{Sig}(\Phi(\Sigma_2)), \textbf{Ax}(\Phi(\Sigma_2)) \cup \alpha_{\Sigma_2}(\Gamma_2) \rangle\ \textbf{by}\ \Phi(\sigma_2)$$

is a normal form of $\hat{\mu}(SP_0)$. By Lemma 2.20,

derive from
$\langle \textbf{Sig}(\Phi(\Sigma')), \Phi(\theta_1)(\textbf{Ax}(\Phi(\Sigma_2)) \cup \alpha_{\Sigma_2}(\Gamma_2)) \cup \Phi(\theta_2)(\textbf{Ax}(\Phi(\Sigma_1))) \rangle$
by $\Phi(\theta_2)$

is a normal form of $(\textbf{translate}\ \hat{\mu}(SP_0)\ \textbf{by}\ \Phi(\sigma_1)) \cup \Phi(\Sigma_1) = \hat{\mu}(SP)$. Since we have $\Phi(\theta_1)(\alpha_{\Sigma_2}(\Gamma_2)) = \alpha_{\Sigma'}(\theta_1(\Gamma_2))$ by naturality of α and $\textbf{Ax}(\Phi(\Sigma')) = \Phi(\theta_2)(\textbf{Ax}(\Phi(\Sigma_1))) \cup \Phi(\theta_1)(\textbf{Ax}(\Phi(\Sigma_2)))$ by Lemma 2.20, this normal form is just $\hat{\mu}(SP')$. □

Occasionally, we also need the notion of *conjunctive* institution representation.

Let $\mathcal{P}_{fin} \colon \textbf{Set} \longrightarrow \textbf{Set}$ be the covariant finite powerset functor, mapping each set to the set of its finite subsets, and each function f to the function taking a finite subset to its image along f. Given institutions I and J, a *conjunctive institution representation* [54] $\mu = (\Phi, \alpha, \beta) \colon I \longrightarrow J$ consists of

- a functor $\Phi \colon \textbf{Sign}^I \longrightarrow \textbf{Pres}^J$,

- a natural transformation $\alpha\colon \mathbf{Sen}^I \longrightarrow \mathcal{P}_{fin} \circ \mathbf{Sen}^J \circ \Phi$, and
- a natural transformation $\beta\colon \mathbf{Mod}^J \circ \Phi^{op} \longrightarrow \mathbf{Mod}^I$,

such that the representation condition is satisfied for all $\Sigma \in \mathbf{Sign}^I$, $M' \in \mathbf{Mod}^J(\Phi(\Sigma))$ and $\varphi \in \mathbf{Sen}^I(\Sigma)$:

$$M' \models^J_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\varphi) \Leftrightarrow \beta_\Sigma(M') \models^I_\Sigma \varphi.$$

It is also possible to combine the sentence-theoretic property of being conjunctive with the other, model-theoretic properties of representations that we are going to introduce in the following subsections.

2.4.1 Borrowing

An important use of institution representations is the re-use (also called borrowing) of proof calculi and theorem provers.

Definition 2.22 Let $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ be an institution representation and $\mathcal{SP}$ a class of I-specifications. We say that μ *admits borrowing of entailment for* $\mathcal{SP}$, if for any Σ-specification $SP \in \mathcal{SP}$ and any Σ-sentence φ in I, we have

$$SP \models^I_\Sigma \varphi \text{ iff } \hat{\mu}(SP) \models^J_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\varphi).$$

Moreover, we say that μ *admits borrowing of refinement for* $\mathcal{SP}$, if for any Σ-specifications $SP_1, SP_2 \in \mathcal{SP}$, we have

$$SP_1 \leadsto SP_2 \text{ iff } \hat{\mu}(SP_1) \leadsto \hat{\mu}(SP_2). \qquad \square$$

The importance of this definition lies in the following: If we have a sound proof calculus for entailment in J, and if we have an institution representation $\mu\colon I \longrightarrow J$ admitting borrowing of entailment for $\mathcal{SP}$, we can use the proof calculus also for proving entailment concerning I-specifications in $\mathcal{SP}$: we just have to translate our proof goals using $\hat{\mu}$ and α. If, moreover, the proof calculus is complete for proving entailment in J, then also its re-use for proving entailment in I is complete. A similar remark holds for proof calculi for refinement.

Since

$$SP \models_\Sigma \varphi \text{ iff } \langle \Sigma, \{\varphi\} \rangle \leadsto SP,$$

μ admits borrowing of entailment for $\mathcal{SP}$ if μ admits borrowing of refinement for $\mathcal{SP}$ (provided that $\mathcal{SP}$ contains all specifications of form $\langle \Sigma, \{\varphi\} \rangle$).

In the next subsections, we will study conditions under which an institution representation admits borrowing. We therefore introduce various properties of institution representations. A first property is the model expansion property:

Definition 2.23 A simple institution representation (Φ, α, β) *admits model expansion* if β is pointwise surjective on objects (i.e., each β_Σ is surjective on objects). $\square$

Example 2.24 The institution representations from Examples 2.16 and 2.17 admit model expansion. For the former one, this is trivial. For the latter one, any partial model can be completed to a total model by adding one element to each carrier (as interpretation of $\bot$), representing "undefined", which is a fixpoint of all functions, while predicates do not hold on it. Then, this one-point completion just generates the original model via the model translation. $\square$

Proposition 2.25 Let $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ be an institution representation and $\mathcal{SP}$ a class of I-specifications.

(1) Assume that for each $SP \in \mathcal{SP}$, $\mathbf{Mod}^I(SP) = \beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP)))$ [6]. Then μ admits borrowing of entailment for $\mathcal{SP}$.
(2) Assume that μ admits model expansion and that for each $SP \in \mathcal{SP}$, $\beta^{-1}_{\mathbf{Sig}(SP)}(\mathbf{Mod}^I(SP)) = \mathbf{Mod}^J(\hat{\mu}(SP))$ [7]. Then μ admits borrowing of entailment and of refinement for $\mathcal{SP}$.
(3) The assumption in (1) can be weakened to $\beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP))) \subseteq \mathbf{Mod}^I(SP)$, if additionally the simultaneous restriction and corestriction $\beta_{\mathbf{Sig}(SP)}\colon \mathbf{Mod}^J(\hat{\mu}(SP)) \longrightarrow \mathbf{Mod}^I(SP)$ is isomorphism-dense, and satisfaction in I is closed under isomorphism.
(4) The assumption of model expansion in (2) can be replaced by assuming that β is pointwise isomorphism-dense and for each $SP \in \mathcal{SP}$, $\mathbf{Mod}^I(SP)$ is isomorphism-closed.

Proof. (1) Let $SP \in \mathcal{SP}$ be a Σ-specification and φ be a Σ-sentence. Then

$SP \models^I_\Sigma \varphi$
iff (by definition) $M \in \mathbf{Mod}^I(SP)$ implies $M \models^I_\Sigma \varphi$
iff (by the assumption) $M' \in \mathbf{Mod}^J(\hat{\mu}(SP))$ implies $\beta_\Sigma(M') \models^I_\Sigma \varphi$
iff (by the representation condition)
$\quad M' \in \mathbf{Mod}^J(\hat{\mu}(SP))$ implies $M' \models^J_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\varphi)$
iff (by definition) $\hat{\mu}(SP) \models^J_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\varphi)$.

(2) Concerning borrowing of entailment, by surjectivity of β_Σ, we obtain $\beta_\Sigma(\beta^{-1}_\Sigma(\mathcal{M})) = \mathcal{M}$ for any $\mathcal{M} \subseteq \mathbf{Mod}^I(\Sigma)$. Thus, we obtain that the assumption of (1) is fulfilled, and the result follows.

[6] This includes the condition that $\beta_{\mathbf{Sig}(SP)}$ is defined on $\mathbf{Mod}^J(\hat{\mu}(SP))$.
[7] This precisely means $\beta_{\mathbf{Sig}(SP)}(M')$ is defined and a member of $\mathbf{Mod}^I(SP)$ if and only if $M' \in \mathbf{Mod}^J(\hat{\mu}(SP))$.

Concerning borrowing of refinement, let $SP_1, SP_2 \in \mathcal{SP}$ be Σ-specifications. Assume that $SP_1 \approx\!\!\gg SP_2$. If now $M' \in \mathbf{Mod}^J(\hat{\mu}(SP_2))$, by the assumption of the proposition, we get $\beta_\Sigma(M') \in \mathbf{Mod}^I(SP_2)$ and therefore $\beta_\Sigma(M') \in \mathbf{Mod}^I(SP_1)$. Again by the assumption of the proposition, we obtain $M' \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$. Hence, $\hat{\mu}(SP_1) \approx\!\!\gg \hat{\mu}(SP_2)$.

Conversely, assume that $\hat{\mu}(SP_1) \approx\!\!\gg \hat{\mu}(SP_2)$. If now $M \in \mathbf{Mod}^I(SP_2)$, by the assumptions of the proposition, we get some $M' \in \mathbf{Mod}^J(\hat{\mu}(SP_2))$ with $\beta_\Sigma(M') = M$. Since then also $M' \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$, by the assumption of the proposition also $\beta_\Sigma(M') = M \in \mathbf{Mod}^I(SP_1)$. Hence, $SP_1 \approx\!\!\gg SP_2$.

(3) In the step of the proof of (1) where we use the assupmtion, we now only get some $M' \in \mathbf{Mod}^J(\hat{\mu}(SP))$ with $\beta_{\mathbf{Sig}(SP)}(M') \cong M$, instead of $\beta_{\mathbf{Sig}(SP)}(M') = M$. But this does no harm since satisfaction in I is closed under isomorphism.

(4) Similarly as (3). $\square$

Borrowing of entailment is strictly weaker than borrowing of refinement, see [12] for an example.

2.4.2 Borrowing For Flat and Flattenable Specifications

For representations admitting model expansion, the well-known "Borrowing theorem" [18,74] holds:

Theorem 2.26 Let I and J be two institutions and $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ be an institution representation admitting model expansion. Then μ admits borrowing of entailment and refinement for flat specifications.

Proof. Let $SP = \langle \Sigma, \Gamma \rangle$ be a flat specification. Then $\beta_\Sigma(M')$ is defined and satisfies SP iff $\beta_\Sigma(M')$ is defined and satisfies Γ iff (by the representation condition) M' satisfies $\mathbf{Ax}(\Phi(\Sigma)) \cup \alpha_\Sigma(\Gamma)$ iff M' satisfies $\hat{\mu}(SP)$. Thus $\beta_\Sigma^{-1}\mathbf{Mod}^I(SP) = \mathbf{Mod}^J(\hat{\mu}(SP))$. The result now follows from Proposition 2.25 (2). $\square$

Corollary 2.27 Let I and J be two institutions and $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ be an institution representation admitting model expansion. Then μ admits borrowing of entailment and refinement for flattenable specifications.

Proof. By Proposition 2.19 and Fact 2.6.

That is, if we have a sound (and complete) theorem prover for flat(tenable) specifications in the target institution of an institution representation admitting model expansion, we can re-use it as a sound (and complete) theorem prover for flat(tenable) specifications in the source institution. In a word:

Institution representations admitting model expansion also admit borrowing of entailment and refinement for flat(tenbale) specifications.

2.4.3 Borrowing For Structured Specifications (Excluding **free***)*

If we want to re-use theorem provers not only for flat, but also for structured specifications (excluding **free**, which will be studied in the next section), we have to assume some additional property:

Definition 2.28 [12] Let $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ be a simple institution representation and let $\mathcal{D}$ be a class of signature morphisms in I. Then μ is said to have the *weak $\mathcal{D}$-amalgamation property*, if for each signature morphism $\sigma\colon \Sigma_1 \longrightarrow \Sigma_2 \in \mathcal{D}$, the diagram

$$\begin{array}{ccc} \mathbf{Mod}^I(\Sigma_2) & \xleftarrow{\beta_{\Sigma_2}} & \mathbf{Mod}^J(\Phi(\Sigma_2)) \\ \downarrow{\scriptstyle \mathbf{Mod}^I(\sigma)} & & \downarrow{\scriptstyle \mathbf{Mod}^J(\Phi(\sigma))} \\ \mathbf{Mod}^I(\Sigma_1) & \xleftarrow{\beta_{\Sigma_1}} & \mathbf{Mod}^J(\Phi(\Sigma_1)) \end{array}$$

admits weak amalgamation, i.e. any for any two models $M_2 \in \mathbf{Mod}^I(\Sigma_2)$ and $M_1' \in \mathbf{Mod}^J(\Phi(\Sigma_1))$ with $M_2|_\sigma = \beta_{\Sigma_1}(M_1')$, there is some $M_2' \in \mathbf{Mod}^J(\Phi(\Sigma_2))$ with $\beta_{\Sigma_2}(M_2') = M_2$ and $M_2'|_{\Phi(\sigma)} = M_1'$. □

Example 2.29 The institution representation from Example 2.16 trivially satisfies the weak $\mathcal{D}$-amalgamation property, where $\mathcal{D}$ is the class of all signature morphisms in $Eq^=$, since the model translations β_Σ are isomorphisms. □

Example 2.30 Let $\mathcal{D}$ be the class of all injective signature morphisms in $PFOL^=$. Weak $\mathcal{D}$-amalgamation for the institution representation from Example 2.17 can be seen as follows: Let $\sigma\colon \Sigma_1 \longrightarrow \Sigma_2 \in \mathcal{D}$, let M_2 be a Σ_2-model and M_1' be a $\Phi(\Sigma_1)$-model such that $M_2|_\sigma = \beta_{\Sigma_1}(M_1')$. Extend M_1' to a $\Phi(\Sigma_2)$-model M_2' as follows: For any sort s not in the image of σ, let the carrier for sort $\sigma(s)$ in M_2' just be $(M_2)_s \uplus \{*\}$, and let $D_{M_2'}$ hold everywhere except on $*$. $\bot$ is interpreted as $*$ in M_2'. Given a function symbol f in Σ outside the image of σ, $\sigma(f)$ is interpreted in M_2' to be f_{M_2}, except that the interpretation of $\bot$ is delivered if the argument is outside M_2 or the result is not defined due to partiality of the function. Given a predicate symbol p in Σ outside the image of σ, $\sigma(p)$ is interpreted in M_2' to be p_{M_2}, except that it is false if the argument is outside M_2. Then, we have that $\beta_{\Sigma_2}(M_2') = M_2$ and $M_2'|_{\Phi(\sigma)} = M_1'$, showing weak $\mathcal{D}$-amalgamation. □

The following theorem has been proved in [12]:

Theorem 2.31 Let I and J be two institutions, and let $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ be an institution representation. Let $\mathcal{SP}$ be the set of structured specifications in I not containing **free**, and containing **derive**s only along morphisms in $\mathcal{D}$. Then

(1) For any specification SP not containing **free**,

$$\beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP))) \subseteq \mathbf{Mod}^I(SP).$$

(2) If μ has the weak $\mathcal{D}$-amalgamation property, then for $SP \in \mathcal{SP}$,

$$\beta^{-1}_{\mathbf{Sig}(SP)}(\mathbf{Mod}^I(SP)) = \mathbf{Mod}^J(\hat{\mu}(SP)).$$

(3) If μ admits model expansion and has the weak $\mathcal{D}$-amalgamation property, then for $SP \in \mathcal{SP}$,

$$\beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP))) = \mathbf{Mod}^I(SP).$$

(4) If μ admits model expansion and has the weak $\mathcal{D}$-amalgamation property, then μ admits borrowing of entailment and refinement for $\mathcal{SP}$. □

The combination of (2) and (3) means that model classes (loose semantics) are preserved and reflected. This is especially important since CASL (like many other specification languages) has a model-theoretic semantics: a specification denotes a signature together with a class or category of models.

Loose semantics for structured specifications without **free** *and with hiding only along $\mathcal{D}$-morphisms can be lifted along and against institution representations admitting model expansion and weak $\mathcal{D}$-amalgamation.*

Loose semantics in this context means taking just the class of *all* models of a specification as its semantics. Of course, whether these are, e.g., all first-order models or just the finitely generated ones, depends on the model functor of the institution.

(4) means that tools for theorem proving within structured specifications in the target institution can be re-used for theorem proving within structured specifications in the source institution. That is:

Institution representations admitting model expansion and weak $\mathcal{D}$-amalgamation admit borrowing of entailment and refinement for structured specifications without **free** *and with hiding only along $\mathcal{D}$-morphisms.*

Definition 2.32 A simple institution representation $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ is said to be *model-bijective* if for each $\Sigma \in \mathbf{Sign}^I$, β_Σ is a bijection on objects. □

Example 2.33 The institution representation from Example 2.16 is model-bijective, while that from Example 2.17 is not: for a given partial model, it is

possible to add any number of "undefined" elements in a model representing it. Hence, the model translation is not bijective. □

Proposition 2.34 Any model-bijective institution representation both admits model expansion and has the weak $\mathcal{D}$-amalgamation property for arbitrary $\mathcal{D}$.

Proof. Admitting of model expansion is immediate. Concerning weak $\mathcal{D}$-amalgamation, let $\sigma: \Sigma_1 \longrightarrow \Sigma_2$ be a signature morphism and let $M_2 \in \mathbf{Mod}^I(\Sigma_2)$ and $M_1' \in \mathbf{Mod}^J(\Phi(\Sigma_1))$ such that $M_2|_\sigma = \beta_{\Sigma_1}(M_1')$. Then $\beta_{\Sigma_2}^{-1}(M_2)$ is the desired model $M_2' \in \mathbf{Mod}^J(\Phi(\Sigma_2))$ with $\beta_{\Sigma_2}(M_2') = M_2$ and $M_2'|_{\Phi(\sigma)} = M_1'$. □

The above slogans can now be shortened:

> *Loose semantics for structured specifications without* **free** *can be lifted along and against model-bijective institution representations.*
>
> *Model-bijective institution representations admit borrowing of entailment and refinement for structured specifications without* **free**.

In [12], it is shown that admitting model expansion and weak $\mathcal{D}$-amalgamation are necessary to get Theorem 2.31. However, if we restrict ourselves to borrowing of entailment, we can replace the requirement that **derive**s in the specification are along morphisms in $\mathcal{D}$-amalgamation by the corresponding requirement for **translate**s.

Theorem 2.35 Let I and J be two institutions, and let $\mu = (\Phi, \alpha, \beta): I \longrightarrow J$ be an institution representation admitting both model expansion and weak $\mathcal{D}$-amalgamation. Let $\mathcal{SP}$ be the set of structured specifications SP in I satisfying the following properties:

- SP does not contain **free**,
- all **translate**s in SP are either flattenable (i.e. do not contain **derive**), or the translation is along a morphism in $\mathcal{D}$, and
- all unions in SP are flattenable (i.e. do not contain **derive**).

Then

(1) For $SP \in \mathcal{SP}$,

$$\mathbf{Mod}^I(SP) = \beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP))).$$

(2) μ admits borrowing of entailment for $\mathcal{SP}$.

Proof. (1) By induction over the structure of SP. Note that by Theorem 2.31(1), we need only prove the inclusion from left to right.

- $SP = \langle \Sigma, \Gamma \rangle$: From the proof of Theorem 2.26, we get $\beta_\Sigma^{-1}\mathbf{Mod}^I(SP) = \mathbf{Mod}^J(\hat{\mu}(SP))$. By model expansion, we get $\mathbf{Mod}^I(SP) = \beta_\Sigma(\mathbf{Mod}^J(\hat{\mu}(SP)))$.
- $SP = SP_1 \cup SP_2$: Since SP is flattenable, by Proposition 2.19 and Fact 2.6, we can refer to the case of presentations.
- $SP =$ **translate** SP_1 **by** $\sigma: \Sigma_1 \longrightarrow \Sigma$: If SP is flattenable, by Proposition 2.19 and Fact 2.6, we can refer to the case of presentations.
  Otherwise, $\sigma \in \mathcal{D}$. For $M \in \mathbf{Mod}^I(SP)$, we have $M|_\sigma \in \mathbf{Mod}^I(SP_1)$. By the induction hypothesis, there is some $M_1' \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$ with $\beta_{\Sigma_1}(M_1') = M|_\sigma$. By the weak $\mathcal{D}$-amalgamation property, there is some $M' \in \mathbf{Mod}^J(\Phi(\Sigma))$ with $M'|_{\Phi(\sigma)} = M_1'$ and $\beta_\Sigma(M') = M$. By $M'|_{\Phi(\sigma)} = M_1'$, $M' \in \mathbf{Mod}^J(\mathbf{translate}\ \hat{\mu}(SP_1)\ \mathbf{by}\ \Phi(\sigma) \cup \Phi(\Sigma)) = \mathbf{Mod}^J(\hat{\mu}(SP))$. Hence, $M \in \beta_\Sigma(\mathbf{Mod}^J(\hat{\mu}(SP)))$.
- $SP =$ **derive from** SP_1 **by** $\sigma: \Sigma \longrightarrow \Sigma_1$: For $M \in \mathbf{Mod}^I(SP)$, we have some $M_1 \in \mathbf{Mod}^I(SP_1)$ with $M_1|_\sigma = M$. By induction hypothesis, we get some $M_1' \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$ with $\beta_{\Sigma_1}(M_1') = M_1$. This implies that $M_1'|_{\Phi(\sigma)} \in \mathbf{Mod}^J(\hat{\mu}(SP))$. But $\beta_\Sigma(M_1'|_{\Phi(\sigma)}) = \beta_{\Sigma_1}(M_1')|_\sigma = M_1|_\sigma = M$. Hence, $M \in \beta_\Sigma(\mathbf{Mod}^J(\hat{\mu}(SP)))$.

(2) Follows from (1) by Proposition 2.25(1). □

Theorem 2.36 Let I and J be two institutions having weak amalgamation, and let $\mu = (\Phi, \alpha, \beta): I \longrightarrow J$ be an institution representation admitting model expansion such that Φ preserves pushouts. Let $\mathcal{SP}$ be the set of structured specifications in I not containing **free**. Then

(1) For $SP \in \mathcal{SP}$,

$$\mathbf{Mod}^I(SP) = \beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP))).$$

(2) μ admits borrowing of entailment for $\mathcal{SP}$.

Proof. (1) Let SP' be a normal form of SP according to Definition 2.8. By Proposition 2.21, $\hat{\mu}(SP')$ is a normal form for $\hat{\mu}(SP)$. Hence by Fact 2.9 it suffices to show $\mathbf{Mod}^I(SP') = \beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP')))$. Since normal forms neither contain **translate** nor unions, this follows from Theorem 2.35.

(2) Follows from (1) by Proposition 2.25(1). □

2.4.4 Borrowing For Structured Specifications (Including **free***)*

It is easy to show borrowing for structured specifications including **free** under the very strong assumption of a *subinstitution representation*:

Definition 2.37 A simple institution representation $\mu = (\Phi, \alpha, \beta): I \longrightarrow J$ is said to be a *subinstitution representation* if Φ is an embedding of categories,

α is a pointwise injection, and β is a natural isomorphism.

I is said to be a *subinstitution* of J if there is a subinstitution representation from I to J. $\square$

Example 2.38 The institution representation from Example 2.16 is a subinstitution representation. $\square$

Theorem 2.39 Let $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ be a subinstitution representation. Then

- For any specification SP,

$$\beta^{-1}_{\mathbf{Sig}(SP)}(\mathbf{Mod}^I(SP)) = \mathbf{Mod}^J(\hat{\mu}(SP)).$$

- μ admits borrowing of entailment and refinement for all specifications.

Proof. (1) is straightforward. (2) follows with Proposition 2.25(2). $\square$

Thus

> *Subinstitution representations admit borrowing of entailment and refinement for all structured specifications.*

The subinstitution property is a rather strong property. Is there a weaker property that suffices to ensure a good interaction with liberality and the **free** construct? The answer is the following notion, which we have introduced in [44] in a slightly stronger form under the name of categorical retractive simulation:

Definition 2.40 A simple institution representation $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ is called *persistently liberal* if for each $\Sigma \in \mathbf{Sign}^I$, β_Σ has a left adjoint[8] γ_Σ such that also $\beta_\Sigma \circ \gamma_\Sigma \cong id$. If we have even $\beta_\Sigma \circ \gamma_\Sigma = id$, then μ is called *strongly persistently liberal*. We write (μ, γ) if we want to chose a particular γ. $\square$

We here use the term liberal (in accordance with [24]) since it stresses the connection with liberality of institutions. Meseguer [49] has introduced persistently liberal representations under the name of *extensions*. He additionally requires that the isomorphism $id \cong \beta_\Sigma \circ \gamma_\Sigma$ is the unit of the adjunction; however, in the light of Proposition A.1, this requirement is superfluous.

Example 2.41 The institution representation from Examples 2.16 is strongly

[8] Given a functor $G\colon \mathbf{B} \longrightarrow \mathbf{A}$, a functor $F\colon \mathbf{A} \longrightarrow \mathbf{B}$ is called left adjoint to G, if for each $A \in \mathbf{A}$, $F(A)$ is G-free over A, and, moreover, the universal arrows form a natural transformation $\eta\colon Id \longrightarrow G \circ F$.

persistently liberal: γ_Σ is just the inverse of the isomorphism β_Σ. $\square$

Example 2.42 The institution representation from Example 2.17 is strongly persistently liberal: γ_Σ totalizes a partial model by adding "undefined" values freely (this is the free completion [14,15]). $\square$

Let us now study how persistently liberal institution representations interact with liberality, strengthening a result of [44] (we can now drop the assumption of the existence of $\mathbf{Mod}^I(\sigma)$-free models).

Theorem 2.43 Let $(\mu,\gamma) = ((\Phi,\alpha,\beta),\gamma): I \longrightarrow J$ be a persistently liberal institution representation such that additionally either satisfaction in I is closed under isomorphism or (μ,γ) is even strongly persistently liberal. Then free constructions can be lifted against (μ,γ) in the following sense:

(1) If $\sigma: \langle\Sigma_1,\Gamma_1\rangle \longrightarrow \langle\Sigma_2,\Gamma_2\rangle$ is a theory morphism in I, $M_1 \in \mathbf{Mod}^I(\langle\Sigma_1,\Gamma_1\rangle)$, and if $M_2' \in \mathbf{Mod}^J(\hat{\mu}(\langle\Sigma_2,\Gamma_2\rangle))$ is $\mathbf{Mod}^J(\Phi(\sigma))$-free over $\gamma_{\Sigma_1}(M_1)$, then $\beta_{\Sigma_2}(M_2')$ is $\mathbf{Mod}^I(\sigma)$-free over M_1.
(2) If $\sigma: \langle\Sigma_1,\Gamma_1\rangle \longrightarrow \langle\Sigma_2,\Gamma_2\rangle$ is a theory morphism in I, then σ is liberal if $\Phi(\sigma)$ is liberal.
(3) I is liberal if J is liberal.
(4) Let $\mathcal{M}$ be a class of signature morphisms in $\mathbf{Sign}^I$. I is $\mathcal{M}$-liberal if J is $\Phi(\mathcal{M})$-liberal.

In a word:

Free constructions can be lifted against persistently liberal institution representations.

Proof.

$$\begin{array}{ccc}
\mathbf{Mod}^I(\langle\Sigma_2,\Gamma_2\rangle) & \underset{\beta_{\Sigma_2}}{\overset{\gamma_{\Sigma_2}}{\rightleftarrows}} & \mathbf{Mod}^J(\hat{\mu}(\langle\Sigma_2,\Gamma_2\rangle)) \\
\Big\downarrow{\scriptstyle \mathbf{Mod}^I(\sigma)} & & \Big\downarrow{\scriptstyle \mathbf{Mod}^J(\Phi(\sigma))} \\
\mathbf{Mod}^I(\langle\Sigma_1,\Gamma_1\rangle) & \underset{\beta_{\Sigma_1}}{\overset{\gamma_{\Sigma_1}}{\rightleftarrows}} & \mathbf{Mod}^J(\hat{\mu}(\langle\Sigma_1,\Gamma_1\rangle))
\end{array}$$

(1) By the representation condition, β_{Σ_i} restricts to $\beta_{\Sigma_i}: \mathbf{Mod}^J(\hat{\mu}(\langle\Sigma_i,\Gamma_i\rangle)) \longrightarrow \mathbf{Mod}^I(\langle\Sigma_i,\Gamma_i\rangle)$ for $i = 1,2$. We now show that similarly, γ_{Σ_i} restricts to $\gamma_{\Sigma_i}: \mathbf{Mod}^I(\langle\Sigma_i,\Gamma_i\rangle) \longrightarrow \mathbf{Mod}^J(\hat{\mu}(\langle\Sigma_i,\Gamma_i\rangle))$: For $M \in \mathbf{Mod}^I(\langle\Sigma_i,\Gamma_i\rangle)$, either $\beta_{\Sigma_i}(\gamma_{\Sigma_i}(M)) = M$, or $\beta_{\Sigma_i}(\gamma_{\Sigma_i}(M)) \cong M$ and satisfaction in I is closed under isomorphism. In both cases, $\beta_{\Sigma_i}(\gamma_{\Sigma_i}(M)) \models_{\Sigma_i} \Gamma_i$, and by the representation condition, $\gamma_{\Sigma_i}(M) \models \alpha_{\Sigma_i}(\Gamma_i)$, hence $\gamma_{\Sigma_i}(M) \in \mathbf{Mod}^J(\hat{\mu}(\langle\Sigma_i,\Gamma_i\rangle))$. We now can apply Proposition A.2 (1) with $U = \mathbf{Mod}^I(\sigma)$, $V = \mathbf{Mod}^J(\Phi(\sigma))$, $R = \beta_{\Sigma_2}$, $L = \gamma_{\Sigma_2}$, $R' = \beta_{\Sigma_1}$, and $L' = \gamma_{\Sigma_1}$, $B = M_2'$ and $X = M_1$.

(2), (3) and (4) directly follow from (1). $\square$

Example 2.55 below shows that the assumption of persistent liberality is needed to get this result.

We now come to borrowing for specifications containing **free**.

Definition 2.44 Given institutions I and J, a persistently liberal institution representation $(\mu,\gamma) = ((\Phi,\alpha,\beta),\gamma)\colon I \longrightarrow J$ and a signature morphism $\sigma\colon \Sigma_1 \longrightarrow \Sigma_2$ in I, γ is called *σ-natural*, if $\gamma_{\Sigma_1} \circ \mathbf{Mod}^I(\sigma) = \mathbf{Mod}^J(\Phi(\sigma)) \circ \gamma_{\Sigma_2}$.

Theorem 2.45 Let $(\mu,\gamma) = ((\Phi,\alpha,\beta),\gamma)\colon I \longrightarrow J$ be a persistently liberal institution representation, such that additionally either satisfaction in I is closed under isomorphism or (μ,γ) is even strongly persistently liberal. Let $\mathcal{SP}$ consist of those I-specifications SP for which

- for each **free** SP' **along** $\sigma\colon \Sigma \longrightarrow \Sigma_1$ occurring in SP, either $\Phi(\sigma)\colon \Sigma \longrightarrow SP'$ is strongly persistently liberal or γ is σ-natural, and moreover, γ_Σ is surjective on objects,
- for each **derive from** SP_1 **by** σ occurring in SP, γ is σ-natural,
- for each **translate** SP_1 **by** σ occurring in SP, either γ is σ-natural, or SP_1 is flattenable, i.e. contains neither **free** nor **derive**.

Then

(1) For $SP \in \mathcal{SP}$,

$$\beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP))) \subseteq \mathbf{Mod}^I(SP).$$

(2) For $SP \in \mathcal{SP}$,

$$\gamma_{\mathbf{Sig}(SP)}(\mathbf{Mod}^I(SP)) \subseteq \mathbf{Mod}^J(\hat{\mu}(SP)).$$

(3) μ admits borrowing of entailment for $\mathcal{SP}$.

Proof. We prove (1) and (2) by simultaneous induction over the structure of SP.

For (1), all cases except from **free** are treated as in the proof of Lemma 8.6 in [12] (cf. our Theorem 2.31).

- $SP = \langle \Sigma, \Gamma \rangle$: (2) Assume that $M \in \mathbf{Mod}^I(SP)$, i.e. $M \models_\Sigma \Gamma$. Now either $M = \beta_\Sigma(\gamma_\Sigma(M))$ or $M \cong \beta_\Sigma(\gamma_\Sigma(M))$ and satisfaction in I is closed under isomorphism. In both cases, $\beta_\Sigma(\gamma_\Sigma(M)) \models_\Sigma \Gamma$. By the representation condition, $\gamma_\Sigma(M) \models \alpha_\Sigma(\Gamma)$. Since $\gamma_\Sigma(M) \in \mathit{dom}\ \beta_\Sigma$, also $\gamma_\Sigma(M) \models \mathbf{Ax}(\Phi(\Sigma))$. Hence, $M \in \mathbf{Mod}^J(\hat{\mu}(SP))$.
- $SP = SP_1 \cup SP_2$: (2) follows directly from the induction hyposthesis.

- $SP = \mathbf{translate}\ SP_1\ \mathbf{by}\ \sigma\colon \Sigma_1 \longrightarrow \Sigma$: (2) In the case that SP_1 (and hence SP) is flattenable, by Proposition 2.19 and Fact 2.6, we can refer to the case of presentations. Otherwise, let $M \in \mathbf{Mod}^I(SP)$, i.e. $M|_\sigma \in \mathbf{Mod}^I(SP_1)$. By induction hypothesis, $\gamma_{\Sigma_1}(M|_\sigma) \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$. By γ-naturality of σ, this is just $\gamma_\Sigma(M)|_{\Phi(\sigma)}$. Since also $\gamma_\Sigma(M) \in dom\ \beta_\Sigma$ and therefore $\gamma_\Sigma(M) \models \mathbf{Ax}(\Phi(\Sigma))$, altogether $\gamma_\Sigma(M) \in \mathbf{Mod}^J(\hat{\mu}(SP))$.
- $SP = \mathbf{derive\ from}\ SP_1\ \mathbf{by}\ \sigma\colon \Sigma \longrightarrow \Sigma_1$: (2) Let $M \in \mathbf{Mod}^I(SP)$. Then there is some $M' \in \mathbf{Mod}^I(SP_1)$ with $M'|_\sigma = M$. By induction hypothesis, $\gamma_{\Sigma_1}(M') \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$. Since by γ-naturality of σ, $\gamma_{\Sigma_1}(M')|_{\Phi(\sigma)} = \gamma_\Sigma(M'|_\sigma) = \gamma_\Sigma(M)$, $\gamma_\Sigma(M) \in \mathbf{Mod}^J(\hat{\mu}(SP))$.
- $SP = \mathbf{free}\ SP'\ \mathbf{along}\ \sigma\colon \Sigma \longrightarrow \Sigma_1$: By induction hypothesis, β_{Σ_1} restricts to $\beta_{\Sigma_1}\colon \mathbf{Mod}^J(\hat{\mu}(SP')) \longrightarrow \mathbf{Mod}^I(SP')$ and γ_{Σ_1} restricts to $\gamma_{\Sigma_1}\colon \mathbf{Mod}^I(SP') \longrightarrow \mathbf{Mod}^J(\hat{\mu}(SP'))$. We thus can apply Proposition A.2 in the sequel, with $U = \mathbf{Mod}^I(\sigma)$, $V = \mathbf{Mod}^J(\Phi(\sigma))$, $R = \beta_{\Sigma_1}$, $L = \gamma_{\Sigma_1}$, $R' = \beta_\Sigma$, and $L' = \gamma_\Sigma$.

  (1) Let $M \in \mathbf{Mod}^J(\hat{\mu}(SP))$, i.e. M is strongly persistently $\mathbf{Mod}^J(\Phi(\sigma))$-free. By the assumption that γ_Σ is surjective on objects, we can apply Proposition A.2(3) to get that $\beta_{\Sigma_1}(M)$ is strongly persistently $\mathbf{Mod}^I(\sigma)$-free, i.e. $\beta_{\Sigma_1}(M) \in \mathbf{Mod}^I(SP)$.

  (2) Let $M \in \mathbf{Mod}^I(SP)$. From M being $\mathbf{Mod}^I(\sigma)$-free over $M|_\sigma$, by Proposition A.2(2) we get that $\gamma_{\Sigma_1}(M)$ is $\mathbf{Mod}^J(\Phi(\sigma))$-free over $\gamma_\Sigma(M|_\sigma)$. According to the assumption, there are two cases: If γ is σ-natural, we can apply Proposition A.2(4) to get that $\gamma_{\Sigma_1}(M)$ is strongly persistently free. If on the other hand, $\Phi(\sigma)\colon \Sigma \longrightarrow SP'$ is strongly persistently liberal, then there is some M' that is $\mathbf{Mod}^J(\Phi(\sigma))$-free over $\gamma_\Sigma(M|_\sigma) = M'|_{\Phi(\sigma)}$ with the identity as unit. Since free objects are unique up to isomorphism, $\gamma_{\Sigma_1}(M) \cong M'$. Hence, the unit witnessing that $\gamma_{\Sigma_1}(M)$ is free is an isomorphism, and so $\gamma_{\Sigma_1}(M)$ is persistently free. In both cases, by Proposition 2.10, $\gamma_{\Sigma_1}(M) \in \mathbf{Mod}^J(\hat{\mu}(SP))$.

(3) Follows from (1) and (2) with Proposition 2.25(1) (in case that (μ, γ) is strongly persistently liberal) or Proposition 2.25(3) (if satisfaction in I is closed under isomorphism). □

The following examples shows that the assumption of σ-naturality of γ resp. strongly persistent liberality of $\Phi(\sigma)$ in Theorem 2.45 cannot be dropped.

Example 2.46 Consider the strongly persistently liberal institution representation going from $PFOL^=$ to $FOL^=$ introduced in Section 4.1.4 below. Let σ be the signature morphism that is the obvious inclusion of a signature Σ_1 with just one sort into a signature Σ_2 with additionally one partial operation symbol. Clearly, σ is strongly persistently liberal: the free construction equips any set with the everywhere undefined function. Consequently, a Σ_2-model is a model of **free** $\langle \Sigma_2, \emptyset \rangle$ **along** σ iff the partial operation is everywhere undefined.

Now $\Phi(\sigma)$ is not persistently liberal: similarly to Example 2.14, any $\Phi(\Sigma_1)$-model is freely extended by adding an ω-chain

$$\{\, f(a),\ f(f(a)),\ f(f(f(a))),\ \dots \}$$

over each of its elements a. Thus, a $\Phi(\Sigma_2)$-model can never be free over its own σ-reduct, and $\hat{\mu}(\textbf{free}\ \langle \Sigma_2, \emptyset \rangle\ \textbf{along}\ \sigma)$ is inconsistent. Since anything is implied by an inconsistent specifcation, obviously borrowing of entailment fails here. $\square$

The following example shows that surjectivity of γ_Σ (for Σ occuring in "parameter position" in a **free** specification, i.e. in as in **free** SP' **along** $\sigma\colon \Sigma \longrightarrow \Sigma_1$) is really needed in Theorem 2.45.

Example 2.47 Let $ColouredFOL^=$ be a variant of $FOL^=$ where each symbol in a signature can be coloured either red or blue. Define a strongly persistently liberal institution representation $((\Phi, \alpha, \beta), \gamma)\colon ColouredFOL^= \longrightarrow FOL^=$ as follows:

Signatures The sorts and operation symbols and non-unary predicate symbols of a $ColouredFOL^=$-signature are left as they are, while unary predicate symbols are translated to binary ones. Moreover, for each blue unary predicate symbol, the corresponding binary relation is axiomatized to be transitive.

Models A binary relation encoding a unary one is mapped to its reflexive support (i.e. $\{x \mid R(x,x)\}$), while everything else is left unchanged by β_Σ. Vice versa, γ_Σ maps a unary relation R to $\{(x,x) \mid R(x)\}$. This easily is seen to be left adjoint to β_Σ.

Sentences Variables are mapped to pairs of variables (also quantification is over both variables in the pair), and hence, terms have two possible translations: one using the first component of the variable pairs, and the other one using the second one. Usually only the first way of translating terms is used, except from the case of binary relations encoding unary ones: here, the binary relation is applied to both the translations.

Satisfaction The representation condition can easily be checked.

Now consider the inclusion σ of Σ_1 (consisting of a sort s and a red unary predicate symbol $p : s$) into SP_2 (consisting additionally of a blue unary predicate symbol $q : s$, and an axiom $\forall x : s \bullet p(x) \Rightarrow q(x)$). Further consider its translation $\Phi(\sigma)\colon \Phi(\Sigma_1) \longrightarrow \Phi(SP_2)$. Consider the $\Phi(\Sigma_1)$-model with carrier $\{1;2\}$ and relation $\{(1,2);(2,1)\}$. Its free extension along σ is the $\Phi(SP_2)$-model with the same carrier and interpretation of p as $\{(1,2);(2,1)\}$ and of q as $\{(1,1);(1,2);(2,1);(2,2)\}$. Obviously, this free extension is strongly persistently liberal, i.e. a model of $\hat{\mu}(\textbf{free}\ SP_2\ \textbf{along}\ \sigma)$. However, this model translated by β gives a model with carrier $\{1;2\}$ and interpretation of p as $\emptyset$ and of q as $\{1;2\}$. But this is not free over its own σ-reduct, and therefore

not a model of **free** SP_2 **along** σ. Still, the (even strongly persistent!) free model over this reduct exists: it interprets both p and q as $\emptyset$. Hence, β does not preserve the semantics of specifications involving **free**. $\square$

The examples show that unfortunately persistent liberality does not suffice for ensuring preservation of all specifications involving **free**[9]. We therefore introduce the following stronger notion from [44]:

Definition 2.48 An institution representation $\mu = (\Phi, \alpha, \beta): I \longrightarrow J$ is called an *institution embedding* if Φ is an embedding and for each $\Sigma \in \mathbf{Sign}^I$, α_Σ is injective and β_Σ is an equivalence of categories. $\square$

Non-trivial examples of institution embeddings (i.e. other than subinstitution representations) are given in Theorems 10.3, 10.8, and 10.9.

Theorem 2.49 Let I and J be institutions such that satisfaction in I is closed under isomorphisms, and let $\mu = (\Phi, \alpha, \beta): I \longrightarrow J$ be an institution embedding. Let $\mathcal{SP}$ consist of those I-specifications having neither **free** nor **derive** occurring within a **translate** or a union. Then

(1) For $SP \in \mathcal{SP}$,

$$\beta_{\mathbf{Sig}(SP)}(\mathbf{Mod}^J(\hat{\mu}(SP))) \subseteq \mathbf{Mod}^I(SP).$$

(2) For $SP \in \mathcal{SP}$, $\beta_{\mathbf{Sig}(SP)}: \mathbf{Mod}^J(\hat{\mu}(SP)) \longrightarrow \mathbf{Mod}^I(SP)$ is isomorphism-dense.
(3) μ admits borrowing of entailment for $\mathcal{SP}$.

Proof. We simultaneously prove (1) and (2) by induction over the structure of SP.

- $SP = \langle \Sigma, \Gamma \rangle$: (1) follows from Theorem 2.31(1). Concerning (2), assume $M \in \mathbf{Mod}^I(SP)$ (i.e. $M \models_\Sigma \Gamma$). Since $\beta_\Sigma: \mathbf{Mod}^J(\Phi(\Sigma)) \longrightarrow \mathbf{Mod}^I(\Sigma)$ is isomorphism-dense, there is some $M' \in \mathbf{Mod}^J(\Phi(\Sigma))$ with $\beta_\Sigma(M') \cong M$. Since satisfaction in I is closed under isomorphism, $\beta_\Sigma(M') \models \Gamma$. By the satisfaction condition, $M' \models_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\Gamma)$. Hence, $M' \in \mathbf{Mod}^J(\hat{\mu}(SP))$.
- $SP = SP_1 \cup SP_2$: By assumption, SP is flattenable. By Proposition 2.19 and Fact 2.6, we can refer to the case of presentations.
- $SP =$ **translate** SP_1 **by** $\sigma: \Sigma_1 \longrightarrow \Sigma$: Here, the same remark as for unions applies.
- $SP =$ **derive from** SP_1 **by** $\sigma: \Sigma \longrightarrow \Sigma_1$: (1) is proved by the same inductive argument as used in the proof of Lemma 8.6 in [12] (cf. our Theorem 2.31).

[9] It seems that instead of left adjoint right-inverses to the model translations, we need right adjoint left-inverses. We also have developed a notion of bi-liberality with useful results. The practical examples of this, however, are all equivalences of categories.

Concerning (2), for $M \in \mathbf{Mod}^I(SP)$, we have some $M_1 \in \mathbf{Mod}^I(SP_1)$ with $M_1|_\sigma = M$. By induction hypothesis, we get some $M_1' \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$ with $\beta_{\Sigma_1}(M_1') \cong M_1$. By definition, $M_1'|_{\Phi(\sigma)} \in \mathbf{Mod}^J(\hat{\mu}(SP))$. Then we get $\beta_\Sigma(M_1'|_{\Phi(\sigma)}) = \beta_{\Sigma_1}(M_1')|_\sigma \cong M_1|_\sigma = M$.

- $SP = \mathbf{free}\ SP'\ \mathbf{along}\ \sigma\colon \Sigma \longrightarrow \Sigma_1$: By induction hypothesis, we have that $\beta_\Sigma\colon \mathbf{Mod}^J(\hat{\mu}(SP')) \longrightarrow \mathbf{Mod}^I(SP')$ is isomorphism-dense. $\mathbf{Mod}^I(SP')$ is a full subcategory of $\mathbf{Mod}^I(\Sigma)$, $\mathbf{Mod}^J(\hat{\mu}(SP'))$ is a full subcategory of $\mathbf{Mod}^J(\Phi(\Sigma))$, and $\beta_\Sigma\colon \mathbf{Mod}^J(\Phi(\Sigma)) \longrightarrow \mathbf{Mod}^I(\Sigma)$ is full and faithful. Therefore $\beta_\Sigma\colon \mathbf{Mod}^J(\hat{\mu}(SP')) \longrightarrow \mathbf{Mod}^I(SP')$ is full and faithful as well. Hence, it is an equivalence of categories (and it has an inverse up to natural isomorphism, which also is an equivalence of categories). The result now follows by the fact that universal arrows (and isomorphisms) are preserved by equivalences of categories, together with Proposition 2.10.

(3) Follows from (1) and (2) with Proposition 2.25(3). □

The following example shows that the restriction to **translate**s not containing **derive**s is really necessary in Theorem 2.49:

Example 2.50 Consider the reduction μ of the institution I of order-sorted algebra to the institution J of many-sorted algebra in Theorem 10.3. While in order-sorted algebra, subsorts are interpreted as set-theoretic inclusions, the coding in many-sorted algebra uses injective functions. The model translation β takes an injection model and uses a colimit of a diagram built up from the injections to generate an inclusion model. This is an institution embedding. Now let Σ_1 be the I-signature

sorts $s1, s2, s3 < t$;
$s1 < s2$;
$s2 < s3$;

ops $f : s2 \to s1$;
$g : s3 \to s2$;

and Σ_2 be

sorts $s1, s2 < t$;
$s1 < s2$;

ops $id1 : s1 \to s1$;
$f : s2 \to s1$;
$id2 : s2 \to s2$;

There are two obvious signature morphisms $\sigma_1, \sigma_2\colon \Sigma_1 \longrightarrow \Sigma_2$. $\sigma_1(t) = \sigma_2(t) = t$, $\sigma_1(s1) = \sigma_1(s2) = \sigma_2(s1) = s1$, $\sigma_1(s3) = \sigma_2(s2) = \sigma_2(s3) = s2$, $\sigma_1(f) = id1$, $\sigma_1(g) = \sigma_2(f) = f$, $\sigma_2(g) = id2$.

Let Γ_2 consist of the Σ_2-axioms $id1(x1) = x1$, $id2(x2) = x2$ and $f(x2) = x2$ in variables $x1 : s1$ and $x2 : s2$ (note that for $f(x2) = x2$, the left sides is

implicitly injected in $s2$). Let SP be the specification

$$\textbf{translate}\ (\textbf{derive from}\ \langle \Sigma_2, \Gamma_2 \rangle\ \textbf{by}\ \sigma_2)\ \textbf{by}\ \sigma_1.$$

The peculiarity is now that for $M \in \mathbf{Mod}^I(SP)$, we have $M_{s1} = M_{s2}$, but due to the use of subsort inclusions, this already holds for any $\langle \Sigma_2, \Gamma_2 \rangle$-model (while it does *not* hold for $\hat{\mu}(\langle \Sigma_2, \Gamma_2 \rangle)$-models). Let $M' \in \mathbf{Mod}^J(\hat{\mu}(\langle \Sigma_2, \Gamma_2 \rangle))$ such that $M'_{s1} \neq M'_{s2}$. Then $M' \notin \mathbf{Mod}^J(\hat{\mu}(SP))$, but $\beta_{\Sigma_2}(M') \in \mathbf{Mod}^I(SP)$, because isomorphic subsorts of the same sort are identified by β. Thus

$$\beta^{-1}_{\mathbf{Sig}(SP)}(\mathbf{Mod}^I(SP)) \neq \mathbf{Mod}^J(\hat{\mu}(SP)),$$

and if we would add sentences that can detect equality of sorts, μ would not admit borrowing of entailment for SP.

Borrowing of refinement does not work either: Let Γ_1 consist of the Σ_1-axioms $f(x2) = x2$ and $g(x3) = x3$ in variables $x2 : s2$ and $x3 : s3$. Then

$$\textbf{derive from}\ \langle \Sigma_2, \Gamma_2 \rangle\ \textbf{by}\ \sigma_1 \approx\!\!\gg \langle \Sigma_1, \Gamma_1 \rangle$$

but

$$\hat{\mu}(\textbf{derive from}\ \langle \Sigma_2, \Gamma_2 \rangle\ \textbf{by}\ \sigma_1) \not\approx\!\!\gg \hat{\mu}(\langle \Sigma_1, \Gamma_1 \rangle).$$

where the latter can be seen by considering a $\hat{\mu}(\langle \Sigma_1, \Gamma_1 \rangle)$-model M with $M_{s1} \neq M_{s2}$. $\square$

The counterexample and the rather intricate conditions of Theorem 2.49 show that model translations which are equivalences of categories do not interact so well with structured specification. This has its reason in the fact that model classes need not be closed under isomorphism. Even if this property should hold for flat specifications, the problem still comes in with **derive**. Namely, in general the image of the reduct functor $\mathbf{Mod}(\sigma)$ is not closed under isomorphism (the reason for this behaviour is that the functor is not transportable [1]). It would make much sense to close up model classes of specifications under isomorphism. (Indeed, this would need to be done only for presentations and hidings — the other structuring operations preserve closedness under isomorphism). Then, Theorem 2.49 would hold for all specifications, and moreover, borrowing of proof calculi also for refinement would become applicable, as it is for subinstitution representations (cf. Theorem 2.39). However, we have refrained from closing up model classes under isomorphism, since this is neither standard in the literature about structured specifications, nor it is implemented in the semantics of CASL [23].

If we omit **derive**, things behave much better:

Theorem 2.51 Let I and J be institutions such that satisfaction in I is closed under isomorphisms, and let $\mu = (\Phi, \alpha, \beta) \colon I \longrightarrow J$ be an embedding. Let $\mathcal{SP}$ consist of those I-specifications with no **derive**. Then

(1) For $SP \in \mathcal{SP}$,

$$\mathbf{Mod}^J(\hat{\mu}(SP)) = \beta^{-1}_{\mathbf{Sig}(SP)}(\mathbf{Mod}^I(SP)).$$

(2) For $SP \in \mathcal{SP}$, $\mathbf{Mod}^I(SP)$ is closed under isomorphism.
(3) μ admits borrowing of entailment and refinement for $\mathcal{SP}$.

Proof. Again, we simultaneously prove (1) and (2) by induction over the structure of SP.

- $SP = \langle \Sigma, \Gamma \rangle$: (1) follows from the representation condition. (2) follows from satisfaction in I being closed under isomorphism.
- $SP = SP_1 \cup SP_2$: (1) $M' \in \mathbf{Mod}^J(\hat{\mu}(SP))$ iff $M' \in \mathbf{Mod}^J(\hat{\mu}(SP_1)) \cap \mathbf{Mod}^J(\hat{\mu}(SP_1))$ iff (by induction hypothesis) $\beta_{\mathbf{Sig}(SP_1)}(M') \in \mathbf{Mod}^I(SP_1) \cap \mathbf{Mod}^I(SP_2)$ iff $\beta_{\mathbf{Sig}(SP_1)}(M') \in \mathbf{Mod}^I(SP)$.
  (2) The intersection of two isomorphism-closed model classes is isomorphism-closed.
- $SP =$ **translate** SP_1 **by** $\sigma\colon \Sigma_1 \longrightarrow \Sigma$: (1) $M' \in \mathbf{Mod}^J(\hat{\mu}(SP))$ iff $M' \in \mathbf{Mod}^J(\Phi(\Sigma))$ and $M'|_{\Phi(\sigma)} \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$ iff (by induction hypothesis) $\beta_\Sigma(M')|_\sigma = \beta_{\Sigma_1}(M'|_{\Phi(\sigma)}) \in \mathbf{Mod}^I(SP_1)$ iff $\beta_\Sigma(M') \in \mathbf{Mod}^I(SP)$.
  (2) If $M \in \mathbf{Mod}^I(SP)$ and $M' \cong M$, then also $M'|_\sigma \cong M|_\sigma$. By induction hypothesis, $M'|_\sigma \in \mathbf{Mod}^I(SP_1)$. Hence, $M' \in \mathbf{Mod}^I(SP)$.
- $SP =$ **free** SP_1 **along** $\sigma\colon \Sigma \longrightarrow \Sigma_1$: (1) To show that $\beta_{\Sigma_1}\colon \mathbf{Mod}^J(\hat{\mu}(SP_1)) \longrightarrow \mathbf{Mod}^I(SP_1)$ is isomorphism-dense, let $M \in \mathbf{Mod}^I(SP_1)$. By isomorphism-denseness of $\beta_{\Sigma_1}\colon \mathbf{Mod}^J(\Phi(\Sigma_1)) \longrightarrow \mathbf{Mod}^I(\Sigma_1)$, there is $M' \in \mathbf{Mod}^J(\Phi(\Sigma_1))$ with $\beta_{\Sigma_1}(M') \cong M$. By induction hypothesis, $\mathbf{Mod}^I(SP_1)$ is closed under isomorphism, hence $\beta_{\Sigma_1}(M') \in \mathbf{Mod}^I(SP)$. Again by induction hypothesis, $M' \in \mathbf{Mod}^J(\hat{\mu}(SP_1))$. The rest of the proof is as in Theorem 2.49.
  (2) Freeness is a property that is closed under isomorphism. Together with Proposition 2.10, this shows $\mathbf{Mod}^I(SP)$ to be closed under isomorphism.

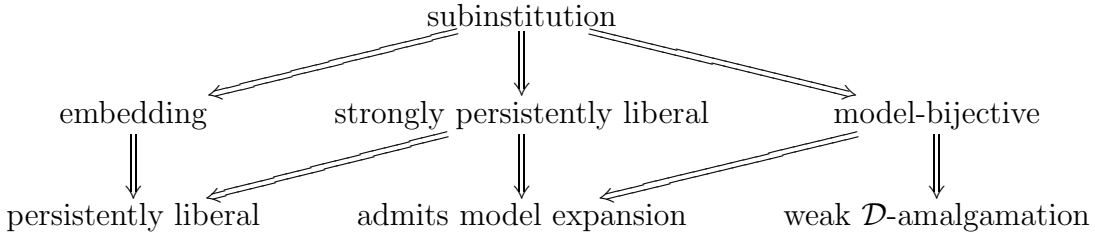
(3) Follows from (1) and (2) with Proposition 2.25(4). □

2.4.5 Comparison Of Properties Of Institution Representations

We now briefly compare the different notions of representations that have been introduced so far.

Proposition 2.52 Any strongly persistently liberal institution representation admits model expansion. Any model-bijective institution representation admits model expansion and weak $\mathcal{D}$-amalgamation for arbitrary $\mathcal{D}$. Any embedding is persistently liberal. Any subinstitution representation is an embedding, strongly persistently liberal and a model-bijective institution represen-

tation.



□

We now present a number of counterexamples showing that the above diagram of implications is optimal.

Example 2.53 The institution representation from Examples 2.17 and 2.42 admits model expansion and it is strongly persistently liberal, but neither model-bijective nor an embedding: for a given partial model, it is possible to add any number of "undefined" elements in a model representing it. Hence, the model translation is neither bijective nor isomorphism-dense. □

Example 2.54 Modify the institution representation from Examples 2.17 and 2.42 by omitting the functions $\bot$ and the axioms involving $\bot$. Then the resulting institution representation is still strongly persistently liberal, but it does not admit weak $\mathcal{D}$-amalgamation, where $\mathcal{D}$ is the class of all signature morphisms adding a partial function symbol to a signature. Let $\sigma\colon \Sigma \longrightarrow \Sigma' \in \mathcal{D}$. Take any Σ'-model M containing a truly partial function and such that all functions in $M|_\sigma$ are total. Then $M|_\sigma$ can be represented by a model M' (i.e. $\beta_\Sigma(M') = M|_\sigma$) that has no "undefined" elements at all. However, it is not possible to represent M with an extension of M', since this would require the presence of some "undefined" element in some carrier of M' (and hence also M). □

Example 2.55 Modify the institution representation from Example 2.17 as follows: Add to $\Phi(\Sigma)$ a sentence $\neg\ x = \bot_s \Rightarrow\ D_s(x)$ (for each sort s in Σ). Thus, $\bot_s$ becomes the *unique* "undefined" element. Now a two-sided inverse of the model translation can be constructed as follows: For a partial Σ-structure M, form its *one-point completion* by just adding one element, $*$ (which is the interpretation of $\bot_s$), to all carriers, let all functions map $*$ to itself and behave as in M otherwise, where undefinedness of partial functions is mapped to $*$. Predicates are false on $*$. The interpretation of the predicates D is fixed by their defining axioms. This shows the representation to be model-bijective (and hence it also admits model expansion).

The above defined inverse to the model translation defines a bijection on model classes, but not an isomorphism of model categories. This is because a Σ-homomorphism need only preserve definedness of partial functions, while

a $\Phi(\Sigma)$-homomorphism has to preserve *and reflect* "definedness" of the representations of partial functions. Thus, the above inverse construction, being the unique inverse construction on models, cannot be extended to a functor. This shows that the above institution representation is neither a subinstitution representation nor persistently liberal.

Now consider the $PFOL^=$-signature Σ with one sort s and one partial function symbol $f\colon s \longrightarrow ?s$, and let σ be the inclusion of the signature consisting just of sort s into Σ. Given a set X, the σ-free Σ-model M over X is just X with f interpreted as the everywhere undefined function. The unique representation M' of M is X with one "undefined" element, and f yielding everywhere the "undefined" element. Now M' is not free over X, since any homomorphism starting from M' has to preserve the undefined element and thus there cannot be a homomorphism from M' into some model where f is defined at some point. This shows that the assumption of persistent liberality is really needed in Theorem 2.43. □

Example 2.56 Let $\mathcal{D}$ be the class of all signature isomorphisms in an institution I. Since functors preserve isomorphisms, any institution representation starting from I admits weak $\mathcal{D}$-amalgamation. Clearly, not every such representation admits model expansion. □

Example 2.57 The institution representation going from $COS\text{-}SubCond^=$ to $COSASC$ described in Theorem 10.4 is an embedding, but it does not admit model expansion (just because there are $COS\text{-}SubCond^=$-models that have non-inclusions as subsort embeddings). □

2.5 Intersections of Subinstitutions

In Section 3.4 below, we want to define subinstitutions of the CASL institution by removing different features from it. Therefore, we need *intersections* of subinstitutions.

Given a family $(\mu_k = (\Phi_k, \alpha^k, \beta^k)\colon J_k \longrightarrow J)_{k\in K}$ of subinstitutions, with $J_k = (\mathbf{Sign}^k, \mathbf{Sen}^k, \mathbf{Mod}^k, \models^k)$, their *intersection* I is defined as follows:

Signatures $\mathbf{Sign}^I := \bigcap_{k\in K} \Phi_k(\mathbf{Sign}^k)$. This is a category, since the Φ_k are embeddings, and categories are closed under taking the image along embeddings and under intersection.
Models $\mathbf{Mod}^I(\Sigma) := \mathbf{Mod}^J(\Sigma)$.
Sentences $\mathbf{Sen}^I(\Sigma) := \bigcap_{k\in K} \alpha^k_{\Phi_k^{-1}(\Sigma)}(\mathbf{Sen}^k(\Phi_k^{-1}(\Sigma)))$, and $\mathbf{Sen}^I(\sigma\colon \Sigma \longrightarrow \Sigma') : \mathbf{Sen}^I(\Sigma) \longrightarrow \mathbf{Sen}^I(\Sigma')$ is the domain-codomain-restriction of $\mathbf{Sen}^J(\sigma)$, which exists by naturality of α. The injection of $\mathbf{Sen}^I(\Sigma)$ into $\mathbf{Sen}^k(\Phi_k^{-1}(\Sigma))$ given

by the appropriate restriction of $(\alpha^k_{\Phi_k^{-1}(\Sigma)})^{-1}$ is denoted by ι^k_Σ.

Satisfaction $M \models^I_\Sigma \varphi$ iff $M \models^J_\Sigma \varphi$ (i.e., the satisfaction condition is inherited from J). $\square$

Note that the intersection is a limit in the category of institutions and simple representations, with limit projections $(\Phi_k^{-1}, \iota, (\beta^k \circ \Phi_k^{-1})^{-1}) \colon I \longrightarrow J_k$ (well, injections would be a better name here, since they inject the intersection I in the J_k).

2.6 Translating language constructs

As stated in the introduction, a specification language typically is based on an institution; and the semantics of the language constructs is defined by mapping the language constructs into mathematical concepts defined in terms of the institution. In this paper, we mainly concentrate on the translation of *basic specifications*, which correspond to *presentations* at the level of concepts.

Given two specification languages L_1 and L_2 with underlying institutions I_1 and I_2, and given an institution representation $\mu \colon I_1 \longrightarrow I_2$, a translation from L_1 to L_2 can be obtained as follows:

For a basic specification SP_1 written in L_1, apply the semantics of L_1 to obtain a presentation T_1 in I_1. Then use μ to obtain a presentation T_2 in I_2. Finally, find a basic specification SP_2 in L_2 having semantics T_2. This assumes that every presentation can be represented in the language. The language CASL fulfills this assumption. We do not know if other languages fulfill this assumption as well, but we would be surprised if not. (The detailed study of language constructs is beyond the scope if this paper.)

For structured specifications, this translation has to be extended inductively according to Definition 2.18. Note that this assumes that L_1 and L_2 have the same structuring mechanisms. The translation of *different* structuring mechanisms into each other is beyond the scope of this paper.

Since the mapping from basic specifications to presentations in the underlying institution is not formally defined for many specification languages (and also because it can become quite complex), we will only informally discuss how an institution representation lifts to a translation of the corresponding language constructs. In some cases, we also indicate how a better translation can be obtained by *not* going the above way from SP_1 to SP_2 through the underlying theories I_1 and I_2, but rather use a direct translation of SP_1 to SP_2. This is advisable e.g. if L_1 and L_2 have similar language constructs for expressing complex presentations in a concise way, while these constructs have rather lengthy expansions in terms of the underlying institution.

3 CASL

CASL is an expressive language combining subsorts, partiality, first-order logic and induction (the latter is expressed using so-called sort generation constraints). The institution underlying CASL is introduced in two steps [23,17]: first, we introduce many-sorted partial first-order logic with sort generation constraints and equality ($PCFOL^=$), and then, subsorted partial first-order logic with sort generation constraints and equality ($SubPCFOL^=$) is described in terms of $PCFOL^=$.

3.1 Partial First-Order Logic

Definition 3.1 The institution $PCFOL^=$.

Signatures A *many-sorted signature* $\Sigma = (S, TF, PF, P)$ in $PCFOL^=$ consists of

- a set S of *sorts*,
- two $S^*\times S$-sorted families $TF = (TF_{w,s})_{w\in S^*,s\in S}$ and $PF = (PF_{w,s})_{w\in S^*,s\in S}$ of *total function symbols* and *partial function symbols*, respectively, such that $TF_{w,s} \cap PF_{w,s} = \emptyset$, for each $(w, s) \in S^*\times S$ (constants are treated as functions with no arguments), and
- a family $P = (P_w)_{w\in S^*}$ of *predicate symbols*.

Note that function and predicate symbols may be overloaded, occurring in more than one of the above sets. To ensure that there is no ambiguity in sentences, however, symbols are always qualified by profiles when used. In the CASL language constructs (see Section 3.3), such qualifications may be omitted when these are unambiguously determined by the context.

We now introduce some notation. We write $f : w \to s \in TF$ for $f \in TF_{w,s}$ (including the special case $f : s$ for empty w), $f : w \to? s \in PF$ for $f \in PF_{w,s}$ (including the special case $f :\to? s$ for empty w) and $p : w \in P$ for $p \in P_w$. For a function symbol $f \in TF_{w,s}$ or $f \in PF_{w,s}$, we call $w \to s$ or $w \to? s$, resp., its *profile*, w its *argument sorts* and s its *result sort*. For predicate symbols $p : w$, we call w both its *profile* and its *argument sorts*. Given a function $f\colon A \longrightarrow B$, let $f^*\colon A^* \longrightarrow B^*$ be its extension to finite strings. Given a finite string $w = s_1 \dots s_n$ and sets $M_{s_1}, \dots, M_{s_n}$, we write M_w for the Cartesian product $M_{s_1} \times \dots \times M_{s_n}$.

Given signatures $\Sigma = (S, TF, PF, P)$ and $\Sigma' = (S', TF', PF', P')$, a signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$ consists of

- a map $\sigma^S\colon S \longrightarrow S'$,
- a map $\sigma^F_{w,s}\colon TF_{w,s} \cup PF_{w,s} \longrightarrow TF'_{\sigma^{S^*}(w),\sigma^S(s)} \cup PF'_{\sigma^{S^*}(w),\sigma^S(s)}$ preserving totality, for each $w \in S^*, s \in S$, and
- a map $\sigma^P_w\colon P_w \longrightarrow P'_{\sigma^{S^*}(w)}$ for each $w \in S^*$.

Identities and composition are defined in the obvious way. This gives us a category of $PCFOL^{=}$-signatures.

Models Given a many-sorted signature $\Sigma = (S, TF, PF, P)$, a *many-sorted Σ-model* M consists of:

- a non-empty carrier set M_s for each sort $s \in S$,
- a partial function $(f_{w,s})_M$ (also written just f_M) from M_w to M_s for each function symbol $f \in TF_{w,s} \cup PF_{w,s}$, the function being total if $f \in TF_{w,s}$, and
- a predicate $(p_w)_M$ (also written just p_M) $\subseteq M_w$ for each predicate symbol $p \in P_w$.

A *many-sorted Σ-homomorphism* $h : M \longrightarrow N$ consists of a family of functions $(h_s : M_s \longrightarrow N_s)_{s \in S}$ with the property that for all $f \in TF_{w,s} \cup PF_{w,s}$ and $(a_1, \ldots, a_n) \in M_w$ with $(f_{w,s})_M(a_1, \ldots, a_n)$ defined, we have

$$h_s((f_{w,s})_M(a_1, \ldots, a_n)) = (f_{w,s})_N(h_{s_1}(a_1), \ldots, h_{s_n}(a_n)),$$

and for all $p \in P_w$ and $(a_1, \ldots, a_n) \in M_w$,

$$(a_1, \ldots, a_n) \in (p_w)_M \text{ implies } (h_{s_1}(a_1), \ldots, h_{s_n}(a_n)) \in (p_w)_N.$$

Identities and composition are defined in the obvious way.

Concerning *reducts*, if $\sigma\colon \Sigma \longrightarrow \Sigma'$ is a signature morphism with $\Sigma = (S, TF, PF, P)$, and M' is a Σ'-model, then $M'|_\sigma$ is the Σ-model M with

- $M_s := M'_{\sigma^S(s)} \quad (s \in S)$
- $(f_{w,s})_M := (\sigma^F_{w,s}(f))_{M'} \quad (f \in TF_{w,s} \cup PF_{w,s})$
- $(p_w)_M := (\sigma^P_w(p))_{M'} \quad (p \in P_w)$

This is well-defined since $\sigma^F_{w,s}$ preserves totality.

Given a Σ'-homomorphism $h'\colon M'_1 \longrightarrow M'_2$, its reduct $h'|_\sigma\colon M'_1|_\sigma \longrightarrow M'_2|_\sigma$ is the homomorphism defined by

$$(h'|_\sigma)_s := h'_{\sigma^S(s)} \quad (s \in S).$$

It is easy to see that the reduct w.r.t. an identity is the identity, and that the reduct w.r.t. a composition is the composition of the reducts w.r.t. the signature morphisms that are composed. Thus, **Mod** is a functor.

Sentences Let a many-sorted signature $\Sigma = (S, TF, PF, P)$ be given. A *variable system over* Σ is an S-sorted, pairwise disjoint family of variables $X = (X_s)_{s \in S}$. Let such a variable system be given.

The sets $T_\Sigma(X)_s$ of *many-sorted Σ-terms* of sort s, $s \in S$, with variables in X are the least sets satisfying the following rules:

(1) $x \in T_\Sigma(X)_s$, if $x \in X_s$,

(2) $f_{w,s}(t_1, \ldots, t_n) \in T_\Sigma(X)_s$, if $t_i \in T_\Sigma(X)_{s_i} (i = 1, \ldots, n), f \in TF_{w,s} \cup PF_{w,s}, w = s_1 \ldots s_n$.

Note that each term has a unique sort.

$T_\Sigma(X)$ can be made into a many-sorted Σ-algebra by putting

- $(f_{w,s})_{T_\Sigma(X)}(t_1, \ldots, t_n) = f_{w,s}(t_1, \ldots, t_n)$ for $f \in TF_{w,s} \cup PF_{w,s}$,

- $(p_w)_{T_\Sigma(X)} = \emptyset$ for $p \in P_w$.

Variable assignments are total, but the value of a term w.r.t. a variable assignment may be undefined, due to the application of a partial function during the evaluation of the term. The evaluation of a term w.r.t. a variable assignment is defined as follows (see [19]):

Given a variable valuation $\nu: X \longrightarrow M$ for X in M, the *term evaluation* $\nu^\#: T_\Sigma(X) \to ? M$ is inductively defined by:

- $\nu^\#_s(x) = \nu(x)$ for all $x \in X_s$ and all $s \in S$,
- $\nu^\#_s(f_{w,s}(t_1, \ldots, t_n)) = \begin{cases} (f_{w,s})_M(\nu^\#_{s_1}(t_1), \ldots, \nu^\#_{s_n}(t_n)), \\ \quad \text{if } \nu^\#_{s_i}(t_i) \text{ is defined } (i = 1, \ldots, n) \text{ and} \\ \quad\quad (f_{w,s})_M(\nu^\#_{s_1}(t_1), \ldots, \nu^\#_{s_n}(t_n)) \text{ is defined} \\ \text{undefined, otherwise} \end{cases}$

  for all $f \in TF_{w,s} \cup PF_{w,s}$, where $w = s_1 \ldots s_n$, and $t_i \in T_\Sigma(X)_{s_i}$, for $i = 1, \ldots, n$.

Given a term t, we say that t is *ν-interpretable*, if $t \in \text{dom } \nu^\#$ and in this case we call $\nu^\#(t) \in M_s$ the *value of t in M under the valuation ν*.

A many-sorted atomic Σ-formula with variables in X is either (1) an application of a qualified predicate symbol to terms of appropriate sorts, (2) an existential equation between terms of the same sort, (3) a strong equation between terms of the same sort, or (4) an assertion about definedness of a term:

The set $AF_\Sigma(X)$ of *many-sorted atomic Σ-formulas* with variables in X is the least set satisfying the following rules:

(1) $p_w(t_1, \ldots, t_n) \in AF_\Sigma(X)$, if $t_i \in T_\Sigma(X)_{s_i}, p \in P_w, w = s_1 \ldots s_n \in S^*$,
(2) $t \stackrel{e}{=} t' \in AF_\Sigma(X)$, if $t, t' \in T_\Sigma(X)_s, s \in S$ (existential equations),
(3) $t = t' \in AF_\Sigma(X)$, if $t, t' \in T_\Sigma(X)_s, s \in S$ (strong equations),
(4) $def\ t \in AF_\Sigma(X)$, if $t \in T_\Sigma(X)_s, s \in S$ (definedness assertions).

The set $FO_\Sigma(X)$ of *many-sorted first-order Σ-formulas* with variables in X is the least set satisfying the following rules:

(1) $AF_\Sigma(X) \subseteq FO_\Sigma(X)$,
(2) $F \in FO_\Sigma(X)$ (read: false),
(3) $(\varphi \wedge \psi) \in FO_\Sigma(X)$ and $(\varphi \Rightarrow \psi) \in FO_\Sigma(X)$ for $\varphi, \psi \in FO_\Sigma(X)$,
(4) $(\forall x : s \bullet \varphi) \in FO_\Sigma(X)$ for $\varphi \in FO_\Sigma(X \cup \{x : s\})$, $s \in S$.

We omit brackets whenever this is unambiguous and use the usual abbreviations: $\neg\varphi$ for $\varphi \Rightarrow F$, $\varphi \vee \psi$ for $\neg(\neg\varphi \wedge \neg\psi)$, T for $\neg\, F$ and $\exists x : s \bullet \varphi$ for $\neg\forall x : s \bullet \neg\varphi$.

A *sort generation constraint* states that some set of sorts is generated by some set of functions. Technically, sort generation constraints also contain a signature morphism component; this is needed to be able to translate them along signature morphisms without sacrificing the satisfaction condition.

Formally, a sort generation constraint over a signature Σ is a triple $(\overset{\bullet}{S}, \overset{\bullet}{F}, \theta)$, where $\theta: \bar{\Sigma} \longrightarrow \Sigma$, $\bar{\Sigma} = (\bar{S}, \bar{TF}, \bar{PF}, \bar{P})$, $\overset{\bullet}{S} \subseteq \bar{S}$ and $\overset{\bullet}{F} \subseteq \bar{TF} \cup \bar{PF}$.

Now a *Σ-sentence* is a closed many-sorted first-order Σ-formula (i.e. a many-sorted first-order Σ-formula in the empty set of variables), or a sort

generation constraint over Σ.

Given a signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$ and variable system X over Σ, we can get a variable system $\sigma(X)$ over Σ' by putting

$$\sigma(X)_{s'} = \bigcup_{\sigma^S(s)=s'} X_s$$

Since the term algebra is total, the inclusion $\zeta_{\sigma,X}: X \longrightarrow T_{\Sigma'}(\sigma(X))|_\sigma$ (construed as a variable valuation) leads to a term evaluation function

$$\zeta^{\#}_{\sigma,X}: T_\Sigma(X) \longrightarrow T_{\Sigma'}(\sigma(X))|_\sigma$$

that is total as well. This can be inductively extended to a translation of Σ-first order formulas along σ:

- $\sigma(t) = \zeta^{\#}_{\sigma,X}(t)$, if t is a Σ-term in variables X,
- $\sigma(p_w(t_1, \ldots, t_n)) = \sigma^P_w(p)_{\sigma^{S*}(w)}(\sigma(t_1), \ldots, \sigma(t_n))$,
- $\sigma(t \stackrel{e}{=} t') = \sigma(t) \stackrel{e}{=} \sigma(t')$,
- $\sigma(t = t') = \sigma(t) = \sigma(t')$,
- $\sigma(def\ t) = def\ \sigma(t)$,
- $\sigma(F) = F$,
- $\sigma(\varphi \wedge \psi) = \sigma(\varphi) \wedge \sigma(\psi)$,
- $\sigma(\forall x : s \bullet \varphi) = \forall x : \sigma^S(s) \bullet \sigma(\varphi)$.

The translation of a Σ-constraint $(\dot{S}, \dot{F}, \theta)$ along σ is the Σ'-constraint $(\dot{S}, \dot{F}, \sigma \circ \theta)$.

It is easy to see that the sentence translation along the identity signature morphism is the identity, and that the sentence translation along a composition of two signature morphisms is is the composition of the sentence translations along the individual signature morphisms. Hence, sentence translation functorial.

Satisfaction Relation Even though the evaluation of a term w.r.t. a variable assignment may be undefined, the evaluation of a formula is always defined (and it is either true or false). That is, we have a two-valued logic. The application of a predicate symbol p to a sequence of argument terms holds w.r.t. a valuation $\nu: X \longrightarrow M$ iff the values of all the terms are defined under $\nu^\#$ and give a tuple belonging to p_M. A definedness assertion concerning a term holds iff the value of the term is defined. An existential equation holds iff the values of both terms are defined and identical, whereas a strong equation holds also when the values of both terms are undefined; thus both notions of equation coincide for defined terms.

More formally, satisfaction of a formula $\varphi \in FO_\Sigma(X)$ by a variable valuation $\nu: X \longrightarrow M$ is defined inductively over the structure of φ:

- $\nu \Vdash_\Sigma p_w(t_1, \ldots, t_n)$ iff $\nu^\#(t_i)$ is defined ($i = 1, \ldots, n$) and, moreover, $(\nu^\#(t_1), \ldots, \nu^\#(t_n)) \in (p_w)_M$
- $\nu \Vdash_\Sigma t_1 \stackrel{e}{=} t_2$ iff $\nu^\#(t_1)$ and $\nu^\#(t_2)$ are both defined and equal,
- $\nu \Vdash_\Sigma t_1 = t_2$ iff $\nu^\#(t_1)$ and $\nu^\#(t_2)$ are either both defined and equal, or

both undefined,

- $\nu \Vdash_\Sigma def\ t$ iff $\nu^\#(t)$ is defined,
- not $\nu \Vdash_\Sigma F$
- $\nu \Vdash_\Sigma (\varphi \wedge \psi)$ iff $\nu \Vdash_\Sigma \varphi$ and $\nu \Vdash_\Sigma \psi$
- $\nu \Vdash_\Sigma (\varphi \Rightarrow \psi)$ iff $\nu \Vdash_\Sigma \varphi$ implies $\nu \Vdash_\Sigma \psi$
- $\nu \Vdash_\Sigma (\forall x : s \bullet \varphi)$ iff for all valuations $\xi: X \cup \{x : s\} \longrightarrow M$ which extend ν on $X \setminus \{x : s\}$ (i.e., $\xi(x) = \nu(x)$ for $x \in X \setminus \{x : s\}$), we have $\xi \Vdash_\Sigma \varphi$.

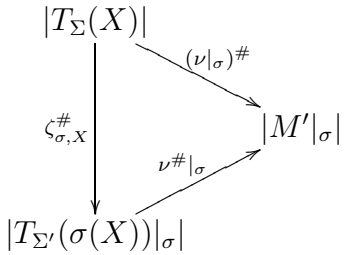
A formula φ is satisfied in a model M (written $M \models \varphi$) iff it is satisfied w.r.t. all variable valuations into M.

A Σ-constraint $(\dot{S}, \dot{F}, \theta)$ satisfied in a Σ-model M, if the carriers of $M|_\theta$ of the sorts in $\dot{S}$ are generated by the function symbols in $\dot{F}$, i.e. for every sort $s \in \dot{S}$ and every value $a \in (M|_\theta)_s$, there is a $\bar{\Sigma}$-term t containing only function symbols from $\dot{F}$ and variables of sorts not in $\dot{S}$ such that $\nu^\#(t) = a$ for some assignment ν into $M|_\theta$.

For a sort generation constraint $(\dot{S}, \dot{F}, \theta)$ we can assume without loss of generality that all the result sorts of function symbols in $\dot{F}$ occur in $\dot{S}$. If not, we can just leave out from $\dot{F}$ those function symbols not satisfying this requirement. The satisfaction of the sort generation constraint in any model will not be affected by this: in the $\bar{\Sigma}$-term t witnessing the satisfaction of the constraint, any application of a function symbol with result sort outside $\dot{S}$ can just be replaced by a variable of that sort, which gets then as assigned value the evaluation of the function application.

Concerning the satisfaction condition, we need the following two lemmas:

Lemma 3.2 Let a signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$, a Σ'-model M', and a variable system X over Σ be given. Then for each valuation $\nu: \sigma(X) \longrightarrow M'$, there is valuation $\nu|_\sigma: X \longrightarrow M'|_\sigma$ such that $\nu^\#|_\sigma \circ \zeta^\#_{\sigma,X} = (\nu|_\sigma)^\#$. Moreover, this is a one-one correspondence.



Proof. We put

$$(\nu|_\sigma)_s(x) := \nu_{\sigma(s)}(x) \text{ for } x \in X_s.$$

The inverse is given by

$$((\_|_\sigma)^{-1}(\nu))_{s'}(x) := \nu_s(x),$$

where s is the unique $s \in S$ with $x \in X_s$ and $\sigma(s) = s'$.

The property $\nu^\#|_\sigma \circ \zeta^\#_{\sigma,X} = (\nu|_\sigma)^\#$ follows by induction over $T_\Sigma(X)$. $\square$

Lemma 3.3 Given a signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$, a Σ'-model M, a variable system X over Σ, and a formula $\varphi \in FO_\Sigma(X)$, we have

$$\nu|_\sigma \Vdash \varphi \text{ iff } \nu \Vdash \sigma(\varphi)$$

Proof. Induction over φ.

E.g. for strong equations, we have $\nu|_\sigma \Vdash t_1 = t_2$
iff $(\nu|_\sigma)^\#(t_1) = (\nu|_\sigma)^\#(t_2)$ (or both sides are undefined)
iff (by Lemma 3.2) $\nu^\# \circ \zeta^\#_{\sigma,X}(t_1) = \nu^\# \circ \zeta^\#_{\sigma,X}(t_2)$ (or both sides are undefined)
iff (by definition) $\nu^\#(\sigma(t_1)) = \nu^\#(\sigma(t_2))$ (or both sides are undefined)
iff $\nu \Vdash \sigma(t_1 = t_2)$.

For quantifications, we have $\nu|_\sigma \Vdash \forall x : s \bullet \psi$ iff for all $\xi: X \cup \{x : s\} \longrightarrow M'|_\sigma$ extending $\nu|_\sigma$ on $X \setminus \{x : s\}$, $\xi \Vdash \psi$ iff (by induction hypothesis) for all $\xi: X \cup \{x : s\} \longrightarrow M'|_\sigma$ extending $\nu|_\sigma$ on $X \setminus \{x : s\}$, $(\_|_\sigma)^{-1}(\xi) \Vdash \sigma(\psi)$ iff (since $\_|_\sigma$ is one-one) for all $\rho: \sigma(X \cup \{x : s\}) \longrightarrow M'$ extending ν on $\sigma(X) \setminus \{x : \sigma(s)\}$, $\rho \Vdash \sigma(\psi)$ iff $\nu \Vdash \forall x : \sigma(s) \bullet \sigma(\psi)$.

The other cases are treated similarly. □

The satisfaction condition for first-order formulas now follows easily from Lemma 3.3 (noting that by Lemma 3.2 σ is surjective on valuations).

The satisfaction condition for sort generation constraints is obvious (indeed, the extra signature morphism component in the sort generation constraints has been introduced to make it work).

This completes the definition of the institution $PCFOL^=$. □

3.2 Subsorted Partial First-Order Logic

Subsorted partial first-order logic is defined in terms of partial first-order logic. The basic idea is to reduce subsorting to injections between sorts. While in the subsorted institution, these injections have to occur explicitly in the sentences, in the CASL language, they may be left implicit. Apart from the injections, one also has partial projection functions (one-sided inverses of the injections) and membership predicates.

Definition 3.4 The institution $SubPCFOL^=$.

Signatures The notion of subsorted signatures extends the notion of order-sorted signatures as given by Goguen and Meseguer [36], by allowing not only total function symbols, but also partial function symbols and predicate symbols:

A *subsorted signature* $\Sigma = (S, TF, PF, P, \leq_S)$ consists of a many-sorted signature (S, TF, PF, P) together with a reflexive transitive *subsort relation* $\leq_S$ on the set S of sorts. Note that $\leq_S$ is not required to be antisymmetric; this allows to declare isomorphic sorts, where the injections correspond to

change of representations.

The relation $\leq_S$ extends pointwise to sequences of sorts. We drop the subscript S when obvious from the context.

For a subsorted signature, $\Sigma = (S, TF, PF, P, \leq_S)$, we define *overloading relations* (also called *monotonicity orderings*), $\sim_F$ and $\sim_P$, for function and predicate symbols, respectively:

Let $f : w_1 \longrightarrow s_1, f : w_2 \longrightarrow s_2 \in TF \cup PF$, then

$$f : w_1 \longrightarrow s_1 \sim_F f : w_2 \longrightarrow s_2$$

iff there exist $w \in S^*$ with $w \leq w_1$ and $w \leq w_2$ and $s \in S$ with $s_1 \leq s$ and $s_2 \leq s$.

Let $p : w_1, p : w_2 \in P$, then $p : w_1 \sim_P p : w_2$ iff there exists $w \in S^*$ with $w \leq w_1$ and $w \leq w_2$.

A *signature morphism* $\sigma : \Sigma \to \Sigma'$ is a many-sorted signature morphism that preserves the subsort relation and the overloading relations. Note that, due to preservation of subsorting, the preservation of the overloading relations can be simplified to:

$$f : w_1 \longrightarrow s_1 \sim_F f : w_2 \longrightarrow s_2 \text{ implies } \sigma^F_{w_1,s_1}(f) = \sigma^F_{w_2,s_2}(f)$$

$$p : w_1 \sim_P p : w_2 \text{ implies } \sigma^P_{w_1}(p) = \sigma^P_{w_2}(p)$$

With each subsorted signature $\Sigma = (S, TF, PF, P, \leq_S)$ we associate a many-sorted signature $\hat{\Sigma}$, which is the extension of the underlying many-sorted signature (S, TF, PF, P) with

- a total *injection* function symbol $\mathtt{inj} : s \to s'$, for each pair of sorts $s \leq_S s'$,
- a partial *projection* function symbol $\mathtt{pr} : s' \to ? s$, for each pair of sorts $s \leq_S s'$, and
- a unary *membership* predicate symbol $\in^s : s'$, for each pair of sorts $s \leq_S s'$.

We assume that the symbols used for injection, projection and membership are not used otherwise in Σ. In formulas, we also write $t \in s$ instead of $\in^s_{s'}(t)$ if s' is clear from the context.

Given a signature morphism $\sigma : \Sigma \longrightarrow \Sigma'$, we can extend it to a signature morphism $\hat{\sigma} : \hat{\Sigma} \longrightarrow \hat{\Sigma}'$ by just mapping the injections, projections and memberships in $\hat{\Sigma}$ to the corresponding injections, projections and memberships in $\hat{\Sigma}'$. This turns $\hat{\_}$ into a functor $\hat{\_} : \mathbf{Sign}^{SubPCFOL^=} \longrightarrow \mathbf{Sign}^{PCFOL^=}$.

Models Subsorted Σ-models are ordinary many-sorted $\hat{\Sigma}$-models satisfying the following set of axioms $\hat{J}(\Sigma)$ (where the variables are all universally quantified):

$\mathtt{inj}_{(s,s)}(x) \stackrel{e}{=} x$ (identity)

$\mathtt{inj}_{(s,s')}(x) \stackrel{e}{=} \mathtt{inj}_{(s,s')}(y) \Rightarrow x \stackrel{e}{=} y$ for $s \leq_S s'$ (embedding-injectivity)

$\mathtt{inj}_{(s',s'')}(\mathtt{inj}_{(s,s')}(x)) \stackrel{e}{=} \mathtt{inj}_{(s,s'')}(x)$ for $s \leq_S s' \leq_S s''$ (transitivity)

$\mathtt{pr}_{(s',s)}(\mathtt{inj}_{(s,s')}(x)) \stackrel{e}{=} x$ for $s \leq_S s'$ (projection)

$\mathtt{pr}_{(s',s)}(x) \stackrel{e}{=} \mathtt{pr}_{(s',s)}(y) \Rightarrow x \stackrel{e}{=} y$ for $s \leq_S s'$ (projection-injectivity)

$\in^s_{s'}(x) \Leftrightarrow def\ \mathtt{pr}_{(s',s)}(x)$ for $s \leq_S s'$ (membership)

$\mathtt{inj}_{(s',s)}(f_{w',s'}(\mathtt{inj}_{(s_1,s'_1)}(x_1),\ldots,\mathtt{inj}_{(s_n,s'_n)}(x_n))) =$
$= \mathtt{inj}_{(s'',s)}(f_{w'',s''}(\mathtt{inj}_{(s_1,s''_1)}(x_1),\ldots,\mathtt{inj}_{(s_n,s''_n)}(x_n)))$
for $f_{w',s'} \sim_F f_{w'',s''}$, where $w \leq w', w''$, $w = s_1 \ldots s_n$, $w' = s'_1 \ldots s'_n$, $w'' = s''_1 \ldots s''_n$ and $s', s'' \leq s$ (function-monotonicity)

$p_{w'}(\mathtt{inj}_{(s_1,s'_1)}(x_1),..,\mathtt{inj}_{(s_n,s'_n)}(x_n)) \Leftrightarrow p_{w''}(\mathtt{inj}_{(s_1,s''_1)}(x_1),..,\mathtt{inj}_{(s_n,s''_n)}(x_n))$
for $p_{w'} \sim_P p_{w''}$, where $w \leq w', w''$, $w = s_1 \ldots s_n$, $w' = s'_1 \ldots s'_n$, and $w'' = s''_1 \ldots s''_n$ (predicate-monotonicity)

Σ-homomorphisms are $\hat{\Sigma}$-homomorphisms.

Lemma 3.5 Let $\sigma\colon \Sigma \longrightarrow \Sigma'$ be a subsorted signature morphism. Then

$$\mathbf{Sen}^{PCFOL^=}(\hat{\sigma})(\hat{J}(\Sigma)) \subseteq \hat{J}(\Sigma')$$

Proof. $\hat{\sigma}$ preserves subsorting, injections, projections, membership and the overloading relations $\sim_F$ and $\sim_P$. Now the sentences in $\hat{J}(\Sigma)$ and $\hat{J}(\Sigma')$ just correspond to these. □

To obtain a *reduct* of a subsorted Σ'-model M' along a subsorted signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, take the the many-sorted reduct $M'|_{\hat{\sigma}} = \mathbf{Mod}^{PCFOL^=}(\hat{\sigma})(M')$ of M' along $\hat{\sigma}\colon \hat{\Sigma} \longrightarrow \hat{\Sigma}'$. By definition of subsorted model, $M' \models^{PCFOL^=}_{\hat{\Sigma}'} \hat{J}(\Sigma')$. By the lemma,

$$M' \models^{PCFOL^=}_{\hat{\Sigma}'} \mathbf{Sen}^{PCFOL^=}(\hat{\sigma})(\hat{J}(\Sigma)).$$

From this, we get by the satisfaction condition for $PCFOL^=$

$$\mathbf{Mod}^{PCFOL^=}(\hat{\sigma})(M') \models^{PCFOL^=}_{\hat{\Sigma}} \hat{J}(\Sigma).$$

Thus, $\mathbf{Mod}^{PCFOL^=}(\hat{\sigma})(M')$ is a subsorted Σ-model, and hence, we can define $\mathbf{Mod}^{SubPCFOL^=}(\sigma)(M')$ to be $\mathbf{Mod}^{PCFOL^=}(\hat{\sigma})(M')$.

Sentences *Subsorted Σ-sentences* are ordinary many-sorted $\hat{\Sigma}$-sentences. Sentence translation along a subsorted signature morphism σ is just sentence translation along the many-sorted signature morphism $\hat{\sigma}$.

Satisfaction Since models and sentences are taken from $PCFOL^=$, satisfaction, as well as the satisfaction condition, can also be inherited from $PCFOL^=$.

This completes the definition of the institution $SubPCFOL^=$. □

3.3 CASL *Language Constructs*

Since the level of constructs will be treated only informally in this work, we just give a brief overview of the constructs for writing basic specifications (i.e. specifications in-the-small) in CASL. A detailed description can be found in the CASL Language Summary [22] and the CASL semantics [23].

The CASL language provides constructs for writing sort, subsort, operation [10] and predicate declarations that contribute to the signature in the obvious way. Operations, predicates and subsorts can also be defined; this leads to a corresponding declaration plus a defining axiom.

Operation and predicate symbols may be overloaded; this may lead to ambiguities in the formulas. A formula is well-formed only if it has a unique fully-qualified expansion up to equivalence w.r.t. the overloading relations $\sim_F$ and $\sim_P$.

For operations and predicates, a mixfix syntax is provided. Precedence and associativity annotations may help to disambiguate terms containing mixfix symbols. There is also a syntax for literals such as numbers and strings, which allows the specification of the usual datatypes purely in CASL, without the need of magic built-in modules.

Binary operations can be declared to be associative, commutative, idempotent, or to have a unit. This leads to a corresponding axiom, and, in the case of associativity, to an associativity annotation.

The **type**, **free type** and **generated type** constructs allow the concise description of datatypes. They are expanded into the declaration of the corresponding constructor and selector operations and axioms relating the selectors and constructors. In the case of generated and free datatypes, also a sort generation constraint is produced. Free datatypes additionally lead to axioms that state the injectivity of the constructors and the disjointness of their images.

A typical CASL specification is shown in Fig. 1. Its translation to a presentation in $SubPCFOL^=$, the institution underlying CASL, is shown in Fig. 2. The translation has been generated with the CASL tool set [51] and uses the CASL notation to display theories in $SubPCFOL^=$. However, the notations are so close to each other that it should be easy to understand Fig. 2. The translation of CASL constructs to the underlying mathematical concepts is formally defined in the CASL semantics [23]. Since such a formal semantics is missing for most other languages, we discuss language constructs only informally in this paper.

3.4 Subinstitutions of CASL

Despite being first-order, CASL is a quite rich and complex language. When relating CASL to other languages, it is quite useful to single out sublanguages of CASL. Here, we define a number of subinstitutions of the CASL institution

[10] At the level of constructs, functions are called operations.

%list $[\_\_], nil, \_\_ :: \_\_$
%prec $\{\_\_ :: \_\_\} < \{\_\_ ++ \_\_\}$

spec LIST [**sort** *Elem*] =
  free type $List[Elem] ::= nil \mid \_\_ :: \_\_(head :? Elem; tail :? List[Elem])$;
  sort $NEList[Elem] = \{L : List[Elem] \bullet \neg L = nil\}$;
  op $\_\_ ++ \_\_ : List[Elem] \times List[Elem] \to List[Elem]$;
  forall $e : Elem; K, L : List[Elem]$
    • $nil ++ L = L$ %(concat_nil)%
    • $(e :: K) ++ L = e :: K ++ L$ %(concat_cons)%
end

Fig. 1. Specification of lists over an arbitrary element sort in CASL.

$SubPCFOL^=$. The corresponding restriction of the CASL language is then straightforward, see [56].

Below, we define subinstitutions of $SubPCFOL^=$ by just imposing restrictions on the signatures and/or axioms. This implicitly means that we take the full subcategory of signatures satisfying the restriction, and restrict the model and sentence functors and the satisfaction relation to this signature subcategory. A restriction on sentences further leads to the replacement of the sentence functor by a subfunctor. In each case, it is quite obvious to construct the corresponding subinstitution representation into $SubPCFOL^=$.

3.4.1 A Number of Features of CASL

In this section, we describe a number of CASL's features *negatively* by specifying, for each feature, the subinstitution of CASL that leaves out exactly that feature. This is possible since CASL is already the combination of all its features. A combination of only some of CASL's features can then be obtained by intersecting all those subinstitutions that exclude exactly one of the undesired features. (Note that the combination of features from scratch is far more complicated [62].)

3.4.1.1 Partiality The institution $SubCFOL^=$ is the restriction of the institution $SubPCFOL^=$ to those signatures with an empty set of partial function symbols and those sentences that do not involve partial projection symbols. Note that $SubCFOL^=$, like $SubPCFOL^=$, is still defined via a reduction to $PCFOL^=$, which involves signatures $\hat{\Sigma}$ containing partial projection symbols. However, these symbols are not used in the sentences, and they are redundant in the models (meaning that leaving them out leads to isomorphic model categories). Thus, it is justified to call $SubCFOL^=$ an institution of *total* algebras.

spec LIST[**sort** Elem] =
  sorts $NEList[Elem] < List[Elem]$
  ops $\_\_ + + \_\_ : List[Elem] \times List[Elem] \to List[Elem]$;
    $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$;
    $head : List[Elem] \to? Elem$;
    $nil : List[Elem]$;
    $tail : List[Elem] \to? List[Elem]$
  generated
    { **sort** $List[Elem]$
    ops $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$;
      $nil : List[Elem]$ }
  forall $X0 : Elem$; $X1 : List[Elem]$
  • $X0 =$
    (**op** $head : List[Elem] \to? Elem$)(
      (**op** $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$)$(X0, X1)$)
    %$(selector\_head)$%
  forall $X0 : Elem$; $X1 : List[Elem]$
  • $X1 : List[Elem] =$
    (**op** $tail : List[Elem] \to? List[Elem]$)(
      (**op** $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$)$(X0, X1)$)
    %$(selector\_tail)$%
  forall $Y0 : Elem$; $Y1 : List[Elem]$
  • $\neg$ (**op** $nil : List[Elem]$) =
    (**op** $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$)$(Y0, Y1)$
    %$(disjoint\_nil\_\_ :: \_\_)$%
  forall $X0 : Elem$; $X1 : List[Elem]$; $Y0 : Elem$; $Y1 : List[Elem]$
  • (**op** $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$)$(X0, X1) =$
    (**op** $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$)$(Y0, Y1)$
    $\Rightarrow X0 = Y0 \wedge X1 = Y1$ %$(injective\_\_ :: \_\_)$%
  forall $L : List[Elem]$
  • $L \in NEList[Elem] \Leftrightarrow \neg L =$ (**op** $nil : List[Elem]$)
    %$(subsort\_defn\_NEList[Elem])$%
  forall $K : List[Elem]$
  • (**op** $\_\_ + + \_\_ : List[Elem] \times List[Elem] \to List[Elem]$)(
    (**op** $nil : List[Elem]$), $L) = L$ %$(concat\_nil)$%
  forall $e : Elem$; $K : List[Elem]$; $L : List[Elem]$
  • (**op** $\_\_ + + \_\_ : List[Elem] \times List[Elem] \to List[Elem]$)(
    (**op** $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$)$(e, K), L) =$
    (**op** $\_\_ :: \_\_ : Elem \times List[Elem] \to List[Elem]$)$(e,$
    (**op** $\_\_ + + \_\_ : List[Elem] \times List[Elem] \to List[Elem]$)$(K, L))$
    %$(concat\_cons)$%
end

Fig. 2. Translation of the specification LIST to a presentation in $SubPCFOL^{=}$.

3.4.1.2 Predicates The institution $SubPCFOAlg^{=}$ is the restriction of $SubPCFOL^{=}$ to those signatures with an empty set of predicate symbols. Note that the signatures $\hat{\Sigma}$ and therefore the sentences in $SubPCFOAlg^{=}$ do involve membership predicate symbols. In the OBJ community, these are called "sort constraints". That is, $SubPCFOAlg^{=}$ includes subsorting with sort constraints. In Section 10, we will also consider subsorting without sort constraints. (Note that sort constraints should not be confused with sort *generation* constraints.)

3.4.1.3 Subsorting The institution $PCFOL^{=}$ has already been defined. It can be made into a subinstitution of $SubPCFOL^{=}$ by extending each signature with the trivial subsort relation (i.e., the subsort relation which is the identity relation on the set of sorts).

3.4.1.4 Sort Generation Constraints The institution $SubPFOL^{=}$ is the restriction of the institution $SubPCFOL^{=}$ to those sentences that are not sort generation constraints.

3.4.1.5 Equality The institution $SubPCFOL$ is the restriction of the institution $SubPCFOL^{=}$ to those sentences that involve neither strong nor existential equality.

3.4.2 A Number of Levels of Axiom Expressiveness

In the sequel, we introduce a number of subinstitutions of $SubPCFOL^{=}$ that correspond to different levels of expressiveness of the axioms. In contrast to the previous subsection, these are not orthogonal features, but rather we get a hierarchy of expressiveness.

3.4.2.1 First-order Logic This is given by $SubPCFOL^{=}$, which trivially is a subinstitution of itself.

3.4.2.2 Positive Conditional Logic Positive conditional logic more precisely means: universally quantified positive conditional logic. Usually this means that formulas are restricted to universally quantified implications that consist of a premise that is a conjunction of atoms, and a conclusion that is an atom:

$$\forall x_1 : s_1 \ldots \forall x_k : s_k \bullet \varphi_1 \wedge \ldots \wedge \varphi_m \Rightarrow \varphi$$

Positive conditional means that the φ_i must not implicitly contain negative parts. Usually, this condition is satisfied by atomic formulas. However, strong equations are implicit implications (*if* one side is defined, *then* so is the other, and they are equal), and the premise of an implication is a negative part of a formula. Hence, strong equations may not occur in the premises of positive conditional axioms (they are harmless in the conclusion, since the implicit premise can be thought of as an additional premise of the whole implication). The main motivation for this is that we want to use proof techniques such as conditional term rewriting and paramodulation [67] and semantical constructions such as initial models. These work only if strong equations are not allowed in the premises (see [16,5]).

Let $SubPCHorn^{=}$ be the restriction of $SubPCFOL^{=}$ to sentences of form

$$\forall x_1 : s_1 \ldots \forall x_k : s_k \bullet \varphi_1 \wedge \ldots \wedge \varphi_m \Rightarrow \varphi$$

where the φ_i and φ are atomic formulas such that none of the φ_i is a strong equation.

3.4.2.3 Generalized Positive Conditional Logic In the following, we generalize the above form of positive conditional formulas. Each formula of this more general kind is equivalent to a set of formulas of the standard conditional kind. Thus, there is an easy transformation from generalized positive conditional logic to plain positive conditional logic.

Within generalized positive conditional formulas, we also allow

- conjunctions of atoms in the conclusion (they can be removed by writing, for each conjunct, an implication with the original premise and the conjunct as conclusion), and
- equivalences instead of implications (an equivalence is equivalent to two implications).

Thus, let $SubPCGHorn^{=}$ be the restriction of $SubPCFOL^{=}$ to sentences of form

$$\forall x_1 : s_1 \ldots \forall x_k : s_k \bullet \varphi_1 \wedge \ldots \wedge \varphi_m \Rightarrow \psi_1 \wedge \ldots \wedge \psi_n$$

where the φ_i and ψ_j are atomic formulas such that none of the φ_i is a strong equation, or of form

$$\forall x_1 : s_1 \ldots \forall x_k : s_k \bullet \varphi_1 \wedge \ldots \wedge \varphi_m \Leftrightarrow \psi_1 \wedge \ldots \wedge \psi_n$$

where the φ_i and ψ_j are atomic formulas that are not strong equations.

3.4.2.4 Atomic Logic Let $SubPCAtom^{=}$ be the restriction of $SubPCFOL^{=}$ to sentences of form

$$\forall x_1 : s_1 \ldots \forall x_n : s_n \bullet \varphi$$

where φ is an atomic formula not being a strong equation.

This is the restriction of conditional logic to unconditional formulas. Strong equations are removed due to their conditional nature: in [53] it is proved that strong equations can simulate positive conditional formulas.

3.4.3 A Terminology for Naming CASL *Subinstitutions*

We now give a two-component name to the various subinstitutions that can be obtained by combining CASL's features. The first component is a vector of tokens. The presence (or absence) of a token denotes the presence (or absence) of a corresponding feature (cf. Section 3.4.1). The second component determines the level of expressiveness due to Section 3.4.2.

We assign the following tokens to the features:

- *Sub* stands for subsorting,
- *P* stands for partiality,
- *C* stands for sort generation constraints, and
- an equality symbol (=) stands for equality.

There is a naming problem with the predicate feature. Firstly, the letter P already stands for partiality. Secondly, FOL for first-order logic or $Horn$ for Horn clause logic have become quite standard, but do not contain a token corresponding to predicates. Therefore, we deal with the predicate feature together with the levels of expressiveness from Section 3.4.2:

With predicates, we have the following endings:

- FOL stands for the unrestricted form of axioms (first-order logic),
- $GHorn$ stands for the restriction to generalized positive conditional logic,
- $Horn$ stands for the restriction to positive conditional logic,
- *Atom* stands for the restriction to atomic logic.

Without predicates, we have the following endings:

- $FOAlg$ stands for the unrestricted form of axioms (first-order logic),
- $GCond$ stands for the restriction to generalized positive conditional logic,
- $Cond$ stands for the restriction to positive conditional logic,
- Eq stands for the restriction to atomic logic.

Any subset of the set of the four tokens Sub, P, C and $=$, followed by any of

the eight above introduced endings now denotes the subinstitution obtained by intersecting

- the subinstitution of $SubPCFOL^=$ corresponding to the ending with
- the intersection of all the subinstitutions of $SubPCFOL^=$ associated to those letters *not* occurring in the set of tokens.

We finally adopt the convention that the equality sign = is always put at the end, as a superscript.

3.4.4 Some Interesting Subinstitutions of CASL

This section shall help to understand what the above naming scheme means in practice.

SubPCFOL$^=$ (read: subsorted partial constraint first-order logic with equality). This is the logic of CASL itself!

SubPFOL$^=$ (read: subsorted partial first-order logic with equality). CASL without sort generation constraints. This is described in [17].

FOL$^=$ Standard many-sorted first-order logic with equality.

PFOL$^=$ Partial many-sorted first-order logic with equality.

FOAlg$^=$ First-order algebra (i.e., no predicates).

SubPHorn$^=$ This is the positive conditional fragment of CASL. It has two important properties:
(1) Initial models and free extensions exist (see Theorem 4.16).
(2) Using a suitable encoding of subsorting and partiality, one can use conditional term rewriting or paramodulation [67] for theorem proving.

SubPCHorn$^=$ The positive conditional fragment plus sort generation constraints. Compared with $SubPHorn^=$, one has to add induction techniques to the theorem proving tools.

PCond$^=$ These are Burmeister's partial quasi-varieties [14] modulo the fact the Burmeister does not have total function symbols. But total function symbols can be easily simulated by partial ones, using totality axioms, as in the partly total algebras of [15]. A suitable restriction leads to Reichel's HEP-theories [70]. Meseguer's Rewriting Logic [48] can be embedded into $PCond^=$.

Horn$^=$ This is Eqlog [34,67]. By further restricting this we get Membership Equational Logic *MEqtl* [49], Equational Type Logic [46] and Unified Algebras [63]. Of course, Membership Equational Logic, Equational Type Logic and Unified Algebras are not just restrictions of $Horn^=$, but all have been invented in order to represent more complex logics within a subset of $Horn^=$.

Horn Logic Programming (Pure Prolog) [45].

SubCond$^=$ Subsorted conditional logic. This is similar but not equal to OBJ [38], see Section 10.

Cond$^{=}$ This is many-sorted conditional equational logic [76] .
SubPAtom The atomic subset of CASL. Unconditional term rewriting becomes applicable.
SubPCAtom The atomic subset plus sort generation constraints.
Eq$^{=}$ This is the classical equational logic [37].
CEq$^{=}$ Equational logic plus sort generation constraints.

In the literature, some of the above institutions are typically defined in a way allowing empty carrier sets, while CASL excludes empty carriers. This problem is discussed in Section 11.2.

4 Representations Among Subinstitutions of CASL

In order to relate subinstitutions of CASL, we relate their underlying institutions. We therefore use the notion of institution representations (also called simple maps of institutions) [47,74] introduced in Section 2.

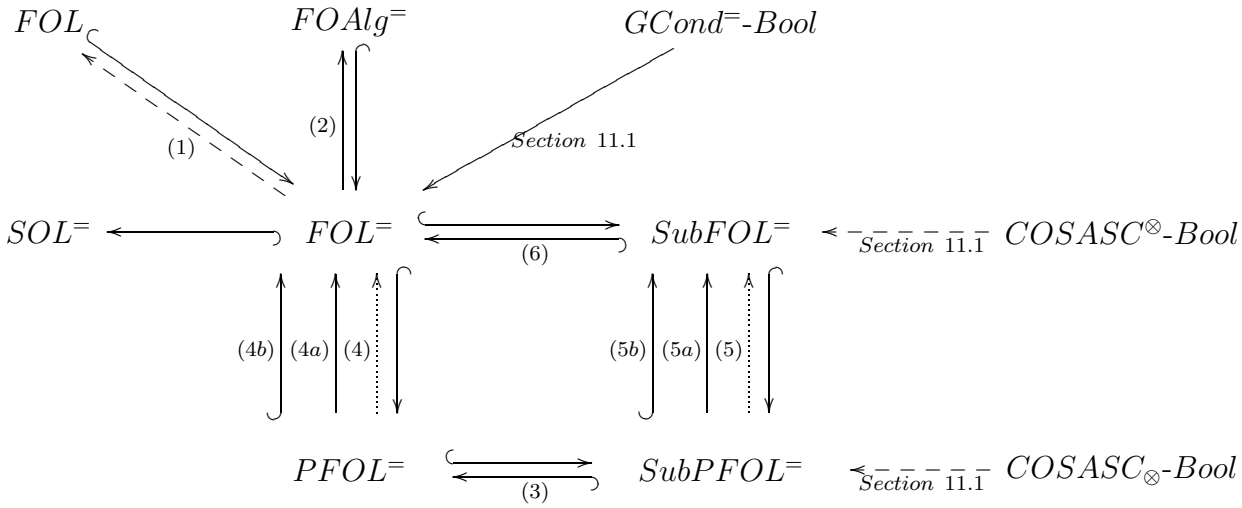
4.1 The First-Order Level

The diagram in Fig. 3 shows that the first-order subinstitutions of CASL have all the same expressiveness, except from FOL, which is a bit weaker ($FOL^{=}$ can be represented in FOL only with a representation admitting model expansion, but not as a subinstitution). Some of the arrows are labeled with numbers in brackets; this refers to later subsections where the corresponding representations are described in detail. Obvious subinstitution representations are not labeled.

Another diagram, shown in Fig. 4, can be obtained by adding a "C" to each institution in Fig. 3. We also have added the institution $SOL^{=}$ (second-order logic with equality), a *superinstitution* of $FOL^{=}$, which is strong enough to express sort generation constraints directly. The institutions in Fig. 4 are strictly more expressive than those in Fig. 3, since sort generation constraints cannot be expressed within first-order logic (since sort generation constraints can be used to specify the natural numbers up to isomorphism, this follows from Gödel's incompleteness theorem).

The institution $SOL^{=}$ of second-order logic can be described as follows:

Signatures Signatures and signature morphisms are those of $FOL^{=}$.
Models Models, model homomorphisms and reducts are that of $FOL^{=}$.
Sentences Σ-sentences may contain variables that may be typed not only with sorts, but also with function types $w \longrightarrow s$ or predicate types $pred(w)$,



Index for arrow types:

- 1. Admits model expansion: Borrowing of entailment and refinement for flat specifications
- 2. Strongly persistently liberal: Lifting of free constructions, implies 1.
- 3. Embedding: Borrowing of entailment for certain structured specifications including **free**
- 4. Model-bijective: Borrowing of entailment and refinement for structured specifications excluding **free**, implies 1.
- 5. Subinstitution: Borrowing of entailment and refinement for all structured specifications

Fig. 3. The first-order level

where $w \in S^*$, $s \in S$. Quantification within sentences over variables of these higher types is also allowed. Variables of function or predicate types may be applied to arguments, like function and predicate symbols in $FOL^=$.

Satisfaction The satisfaction is defined much as in $FOL^=$, with the exception that valuations map variables of a function type to functions of that type, and variables of a predicate type to predicates of that type. □

4.1.1 (1)*: Mapping $FOL^=$ to FOL*

The idea here is simply to replace equality by a congruence relation.

Signatures A $FOL^=$-signature Σ is mapped to the $FOL-$presentation $\Phi(\Sigma)$ that extends Σ by adding predicate symbols $\equiv$: $s \times s$ (overloaded for each sort s) that are axiomatized to be a congruence (also w.r.t. the predicates).

Models A model M is translated by factoring its carriers w.r.t. $\equiv$. The functions and predicates can act on the equivalence classes because $\equiv$ is a congru-

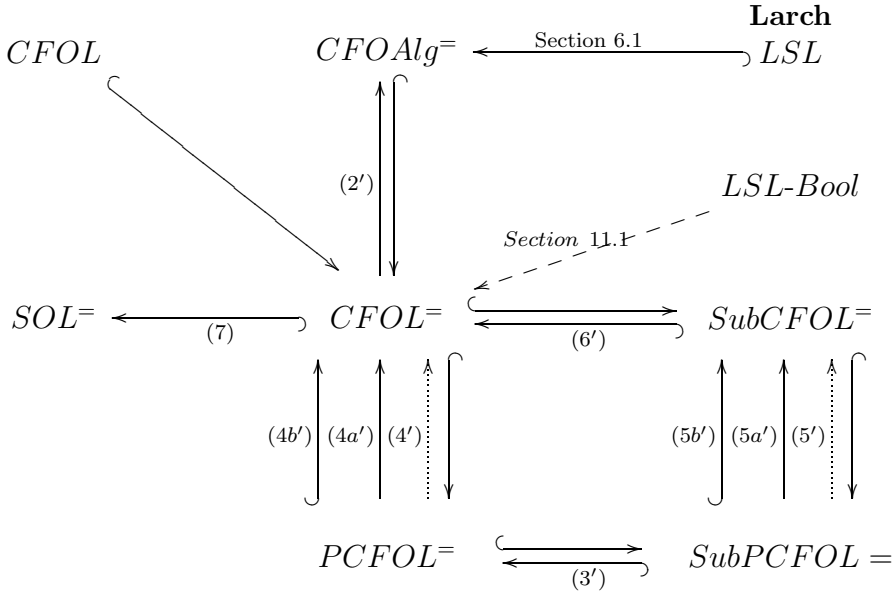


Fig. 4. The first-order level with sort generation constraints (index: see Fig. 3).

ence. A homomorphism $h\colon M \longrightarrow M'$ is translated to $\bar{h}$ with $\bar{h}([a]) = [h(a)]$ (where $[a]$ is the congruence class of a). This is well-defined since $a \equiv_M a'$ implies $h(a) \equiv_{M'} h(a')$ (predicates are preserved by homomorphisms).

Sentences Sentence translation is done by replacing $=$ by $\equiv$.

Satisfaction To prove the representation condition, define a mapping on valuations as follows: Given a $\Phi(\Sigma)$-model M' and a valuation $\nu\colon X \longrightarrow M'$, define $\beta(\nu)\colon X \longrightarrow \beta_\Sigma(M')$ by

$$\beta(\nu)_s(x) := [\nu(x)] \text{ for } x \in X_s.$$

We then have

$$\beta(\nu)^\#(t) = [\nu^\#(t)],$$

which can be easily proved by induction over t, using the congruence axioms. By induction over $\varphi \in \mathbf{Sen}(\Sigma)$, we can now show that

$$\nu \Vdash \alpha_\Sigma(\varphi) \text{ iff } \beta(\nu) \Vdash \varphi.$$

Concerning e.g. strong equations, we have $\nu \Vdash \alpha_\Sigma(t_1 = t_2)$ iff $\nu \Vdash t_1 \equiv t_2$ iff $\nu^\#(t_1) \equiv_{M'} \nu^\#(t_2)$ iff $[\nu^\#(t_1)] = [\nu^\#(t_2)]$ iff $\beta(\nu)^\#(t_1) = \beta(\nu)^\#(t_2)$ iff $\beta(\nu) \Vdash t_1 = t_2$.

Concerning predicate applications, $\nu \Vdash \alpha_\Sigma(p(t_1, \ldots, t_n))$ iff $\nu \Vdash p(t_1, \ldots, t_n)$ iff $(\nu^\#(t_1), \ldots, \nu^\#(t_n)) \in P_{M'}$ iff (by the congruence axiom for p) $([\nu^\#(t_1)], \ldots, [\nu^\#(t_n)]) \in P_{\beta_\Sigma(M')}$ iff $((\beta(\nu))^\#(t_1), \ldots, (\beta(\nu))^\#(t_n)) \in P_{\beta_\Sigma(M')}$ iff $\beta(\nu) \Vdash p(t_1, \ldots, t_n)$.

Concerning quantification, $\nu \Vdash \forall x : s \bullet \alpha_\Sigma(\psi)$ iff for all $\xi\colon X \cup \{x : s\} \longrightarrow M'$ extending ν on $X \setminus \{x : s\}$, $\xi \Vdash \alpha_\Sigma(\psi)$ iff (by induction hypothesis)

for all $\xi: X \cup \{x : s\} \longrightarrow M'$ extending ν on $X \setminus \{x : s\}$, $\beta(\xi) \Vdash \varphi$ iff for all $\rho: X \cup \{x : s\} \longrightarrow \beta_\Sigma(M')$ extending $\beta(\nu)$ on $X \setminus \{x : s\}$, $\rho \Vdash \varphi$ iff $\beta(\nu) \Vdash \forall x : s \bullet \varphi$.

The other cases are treated similarly. The representation condition now follows by noting that β is surjective on valuations. $\square$

The representation can be seen to admit model expansion as follows: Given a Σ-model M in $FOL^=$, turn it into a $\Phi(\Sigma)$-model by letting $\equiv$ be interpreted as the identity relation. This model is a pre-image of M under the model translation. However, the representation is not persistently liberal: for example, let Σ consist of just one sort, let M be the Σ-model consisting of two points, and let M' be the $\Phi(\Sigma)$-model consisting of two equivalent points. Then there is just one homomorphism from M to $\beta_\Sigma(M')$. If the representation were persistently liberal, there would have to be just one homomorphism from $\gamma_\Sigma(M)$ to M' as well. But this is impossible, since then $\gamma_\Sigma(M)$ would have to be empty (even if empty carriers were allowed, this still would contradict the requirement $\beta_\Sigma(\gamma_\Sigma(M)) = M$).

Note that the representation cannot be generalized to the level of sort generation constraints in a straightforward way, because these are not preserved along taking quotients: Let Nat be the signature consisting of a sort Nat and two total function symbols $0 : Nat$ and $suc: Nat \longrightarrow Nat$. Let M' be the $\Phi(Nat)$-model consisting of two copies of the natural numbers, which are identified by the congruence relation. Since the constant 0 yields the zero only of one copy of the naturals, M is not term-generated. However $\beta_\Sigma(M)$ consist of just one copy of the naturals and therefore is term-generated.

4.1.2 (2) and (2′): Mapping $(C)FOL^=$ to $(C)FOAlg^=$

Here, the idea is to replace predicates by Boolean-valued functions. Note that the Booleans can be axiomatized monomorphically in $FOAlg^=$.

Signatures A signature is mapped by adding the presentation BOOL and replacing each predicate symbol $p : w$ by a total function symbol $f_p: w \longrightarrow Bool$

spec BOOL =
sort *Bool*
ops *True*, *False* : *Bool*
forall b : *Bool*
- $b = True \vee b = False$
- $\neg\ True = False$

end

Strictly speaking, BOOL has to be renamed in order to become disjoint with the signature.

Models A model is translated by replacing each *Bool*-valued function by the corresponding predicate.
Sentences Sentence translation is done by replacing atomic formulas

$$p_w(t_1, \ldots, t_n)$$

by

$$(f_p)_{w,Bool}(t_1, \ldots, t_n) = True_{Bool}.$$

Sort generation constraints are left unchanged.
Satisfaction The representation condition is proved in a straightforward way. □

It is also straightforward to show that this gives a model-bijective institution representation. Note, however, that we do *not* get a subinstitution representation. This is because homomorphisms in $FOL^{=}$ need to preserve just truth (but not falsehood) of predicates, while homomorphisms in the representation in $FOAlg^{=}$ need to preserve both truth *and* falsehood of (represented) predicates.

Indeed, we conjecture that there cannot be a subinstitution representation from $FOL^{=}$ to $FOAlg^{=}$ since $FOL^{=}$ is strictly more expressive than $FOAlg^{=}$ w.r.t. a construct that actually exploits the homomorphisms: with the **free** construct, it is possible to express the transitive closure of an arbitrary (parameter) relation, while we conjecture that this is not possible in $FOAlg^{=}$.

4.1.3 (3) *and* (3′)*: Mapping* $SubP(C)FOL^{=}$ *to* $P(C)FOL^{=}$

The translation of $SubP(C)FOL =$ to $P(C)FOL =$ is trivial, since $SubPCFOL^{=}$ is defined in terms of $PCFOL^{=}$:

Signatures A signature Σ is mapped to the presentation $(\hat{\Sigma}, \hat{J}(\Sigma))$.
Models Model translation is the identity.
Sentences Sentence translation is the identity.
Satisfaction The representation condition follows immediately. □

This trivially gives a subinstitution representation.

4.1.4 (4) *and* (4*a*)*: Mapping* $PFOL^{=}$ *to* $FOL^{=}$

A translation of $PFOL^{=}$ into $FOL^{=}$ is described in [19]. We refine this translation here. The main idea is to use a definedness predicate to divide each carrier into "defined" and "undefined" elements. The "defined" elements represent ordinary values, while the "undefined" elements all represent the undefined.

Partial functions thus can be totalized: they possibly yield an "undefined" element. We specify that there is at least one "undefined" element $\bot$; however, it may be not the only one.

Signatures A $PFOL^=$-signature $\Sigma = (S, TF, PF, P)$ is translated to a $FOL^=$-presentation having the signature

$$\mathbf{Sig}(\Phi(\Sigma)) = (S, TF \uplus PF \uplus \{\bot : s \mid s \in S\}, P \uplus \{D : s \mid s \in S\})$$

and the set of axioms $\mathbf{Ax}(\Phi(\Sigma))$:

$\exists x : s \bullet D_s(x)$	$s \in S$	(1)
$\neg D_s(\bot_s)$	$s \in S$	(2)
$D_s(f(x_1, \ldots, x_n)) \Leftrightarrow \bigwedge_{i=1..n} D_{s_i}(x_i)$	$f : s_1, \ldots, s_n \to s \in TF$	(3)
$D_s(g(x_1, \ldots, x_n)) \Rightarrow \bigwedge_{i=1..n} D_{s_i}(x_i)$	$g : s_1 \ldots s_n \to ? \; s \in PF$	(4)
$p(x_1, \ldots, x_n) \Rightarrow \bigwedge_{i=1..n} D_{s_i}(x_i)$	$p\colon s_1 \ldots s_n \in P$	(5)

D plays the role of a definedness predicate: the elements inside D are called "defined", those outside D are called "undefined". The axioms in the signature translation state that there is at least one "defined" element (1), that $\bot$ is an "undefined" element (2), total functions are indeed total (3) and all functions ((3), (4)) and predicates (5) are strict.

A signature morphism σ is translated to a presentation morphism $\Phi(\sigma)$ which acts as σ on those parts of $\Phi(\Sigma)$ being included from Σ, while it maps the added structure for a signature component to the added structure for a mapped component.

Models A $\Phi(\Sigma)$-structure M (in $FOL^=$) is translated to the partial Σ-structure $\beta_\Sigma(M) = M'$ (in $PFOL^=$) with

- $M'_s = (D_s)_M$ for $s \in S$,
- $f_{M'}$ is f_M restricted to M'_w for $f : w \to s \in TF$ (this is a total function by (3)),
- $g_{M'}(a_1, \ldots, a_n) = \begin{cases} g_M(a_1, \ldots, a_n), & \text{if } g_M(a_1, \ldots, a_n) \in M'_s \\ \text{undefined}, & \text{otherwise} \end{cases}$ for $g : w \to ? \; s \in PF$,
- $p_{M'} = p_M \cap M'_w$ for $p\colon w \in P$.

Homomorphisms are translated by restricting them to the carriers of M'. This is well-defined since predicates are preserved by homomorphisms.

Sentences The sentence translation keeps the structure of the sentences and maps strong and existential equality to appropriate circumscriptions using the definedness predicate D. Definedness is mapped to D, and quantifiers are relativized to the set of all defined elements.

Formally, a Σ-sentence φ (in $PFOL^=$) is translated to the $\Phi(\Sigma)$-sentence $\alpha_\Sigma(\varphi)$:

$\alpha_\Sigma(def(t)) = D_s(t)$	$\alpha_\Sigma(t_1 = t_2) = ((D_s(t_1) \vee D_s(t_2)) \Rightarrow t_1 = t_2)$
$\alpha_\Sigma(\varphi \wedge \psi) = \alpha_\Sigma(\varphi) \wedge \alpha_\Sigma(\psi)$	$\alpha_\Sigma(t_1 \stackrel{e}{=} t_2) = t_1 = t_2 \wedge D_s(t_1)$
$\alpha_\Sigma(p(t_1, \ldots, t_n)) = p(t_1, \ldots, t_n)$	$\alpha_\Sigma(F) = F$
$\alpha_\Sigma(\varphi \Rightarrow \psi) = \alpha_\Sigma(\varphi) \Rightarrow \alpha_\Sigma(\psi)$	$\alpha_\Sigma(\forall x : s \bullet \varphi) = \forall x : s \bullet D_s(x) \Rightarrow \alpha_\Sigma(\varphi)$

Satisfaction To prove the representation condition, we need to talk about partial variable valuations. This is necessary since a valuation in a $\Phi(\Sigma)$-model M' may assign an "undefined" element to a variable. In the Σ-model $M = \beta_\Sigma(M')$, this should correspond to a truly undefined variable. Thus, we consider *partial* variable valuations $\nu\colon X \to ?\, M$. The evaluation $\nu^\#$ of terms and the satisfaction $\nu \Vdash$ of formulas is defined as before (see Section 3.1), with the following two exceptions:

- $\nu_s^\#(x) = \begin{cases} \nu(x), & \text{if this is defined} \\ \text{undefined}, & \text{otherwise} \end{cases}$ for all $x \in X_s$ and all $s \in S$;
- $\nu \Vdash_\Sigma (\forall x : s \bullet \varphi)$ iff for all valuations $\xi\colon X \cup \{x : s\} \longrightarrow M$ which extend ν on $X \setminus \{x : s\}$ *and are defined on* $x : s$, we have $\xi \Vdash_\Sigma \varphi$.

Note that the new definitions of $\nu^\#$ and $\Vdash$ coincide with the old ones for total valuations. To achieve this, it is crucial to require the extended valuation ξ to be defined on x in the clause for satisfaction of quantified formulas above.

We now have the following lemma:

Lemma 4.1 For each Σ-formula φ and each (total) valuation $\nu\colon X \longrightarrow M'$ into a $\Phi(\Sigma)$-model M', define the partial valuation $\rho\colon X \longrightarrow \beta_\Sigma(M')$ to be

$$\rho(x) = \begin{cases} \nu(x), & \text{if } \nu(x) \in (D_s)_{M'} \\ \text{undefined}, & \text{otherwise} \end{cases} \quad (x \in X_s)$$

Then we have

(a) $\rho^\#(t)$ is defined iff $\nu^\#(t) \in (D_s)_{M'}$ for all Σ-terms t of sort s.

(b) In the case that one of the sides of (1) holds, we have

$$\rho^\#(t) = \nu^\#(t).$$

(c) $\rho \Vdash_\Sigma^{PFOL^=} \varphi$ iff $\nu \Vdash_{\mathbf{Sig}(\Phi(\Sigma))}^{FOL^=} \alpha_\Sigma(\varphi)$.

Proof. (a) and (b): By induction over the structure of t. For variables, we need just use the definition of ρ. For applications of total or partial function symbols, use axioms (3) or (4) in $\Phi(\Sigma)$.

(c): By induction over the structure of φ. Considering existence equations, we have

$\rho \Vdash_\Sigma^{PFOL^=} t_1 \stackrel{e}{=} t_2$
iff $\rho^\#(t_1)$ and $\rho^\#(t_2)$ are defined and equal
iff $\nu^\#(t_1) \in (D_s)_M$ and $\nu^\#(t_2) \in (D_s)_M$ and $\nu^\#(t_1) = \nu^\#(t_2)$
iff $\nu \Vdash_{\mathbf{Sig}(\Phi(\Sigma))}^{FOL} t_1 = t_2 \wedge D_s(t_1)$
iff $\nu \Vdash_{\mathbf{Sig}(\Phi(\Sigma))}^{FOL} \alpha_\Sigma(t_1 \stackrel{e}{=} t_2)$.

Considering universally quantified formulas, we have

$\rho \Vdash^{PFOL}_{\Sigma} \forall x : s \bullet \varphi$

iff for all $\xi : X \cup \{x : s\} \longrightarrow \beta_\Sigma(M')$ extending ρ on $X \setminus \{x : s\}$ and being defined on $\{x : s\}$, $\xi \Vdash^{PFOL}_{\Sigma} \varphi$

iff for all $\tau : X \cup \{x : s\} \longrightarrow M'$ extending ν on $X \setminus \{x : s\}$ for which $x \in (D_s)_{M'}$,
$\tau \Vdash^{FOL}_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\varphi)$

iff $\nu \Vdash^{FOL}_{\mathbf{Sig}(\Phi(\Sigma))} \forall x : s \bullet D_s(x) \Rightarrow \alpha_\Sigma(\varphi)$

iff $\nu \Vdash^{FOL}_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\forall x : s \bullet \varphi)$.

The other cases are treated similarly. □

The representation condition can now be shown as follows. Let φ be a Σ-sentence and $\nu : \emptyset \longrightarrow M'$ and $\rho : \emptyset \longrightarrow \beta_\Sigma(M')$ be the unique empty valuations. Then

$\beta_\Sigma(M') \models^{PFOL^=}_{\Sigma} \varphi$

iff $\rho \Vdash^{PFOL^=}_{\Sigma} \varphi$

iff $\nu \Vdash^{FOL^=}_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\varphi)$

iff $M' \models^{FOL^=}_{\mathbf{Sig}(\Phi(\Sigma))} \alpha_\Sigma(\varphi)$. □

The model translation can be easily shown to be surjective: For a partial Σ-structure M, form its *one-point completion* by just adding one element, $*$, to all carriers, which is the interpretation of $\bot$. The functions map $*$ to itself and behave as in M otherwise, where undefinedness of partial functions is mapped to $*$. Predicates are false on $*$. The predicate D is true everywhere except on $*$. Thus, our institution representation admits model expansion. Moreover, with the argument of Example 2.30, the representation also has the weak $\mathcal{D}$-amalgamation property, where $\mathcal{D}$ is the class of all injective signature morphisms.

Proposition 4.2 The above constructed institution representation is strongly persistently liberal.

Proof. Define γ as follows. Put $(\gamma_\Sigma(M))_s := M_s \uplus \bar{M}_s$, where

- $\bar{M} \subseteq T_{\mathbf{Sig}(\Phi(\Sigma))}(|M|)$ is the least sorted set satisfying
  - $\bot \in \bar{M}_s$
  - $f(a_1, \ldots, a_n) \in \bar{M}_s$ for $f \in TF_{w,s} \cup PF_{w,s}$, $w = s_1 \ldots s_n$, $(a_1 \ldots a_n) \in (M_w \uplus \bar{M}_w) \setminus dom f_M$
- $(D_s)_{\gamma_\Sigma(M)} := M_s$,
- $(\bot_s)_{\gamma_\Sigma(M)} := \bot$,
- $f_{\gamma_\Sigma(M)}(a_1, \ldots, a_n) := \begin{cases} f_M(a_1, \ldots, a_n), & \text{if } (a_1, \ldots, a_n) \in dom f_M \\ f(a_1, \ldots a_n), & \text{otherwise} \end{cases}$
  for $f : w \longrightarrow s \in TF \cup PF$,
- $p_{\gamma_\Sigma(M)} = p_M$ for $p : w \in P$.

Now (1) of $\Phi(\Sigma)$ above is satisfied since the carriers of M are non-empty. (2) holds by definition of $(D_{\gamma_\Sigma(M)})_s$, and (3) and (4) hold by definition of $f_{\gamma_\Sigma(M)}$.

(5) holds by definition of $p_{\gamma_\Sigma(M)}$.

Clearly, we have $\beta_\Sigma(\gamma_\Sigma(M)) = M$. To show the universal property of

$$M \xrightarrow{id} \beta_\Sigma(\gamma_\Sigma(M))\,,$$

let $h\colon M \longrightarrow \beta_\Sigma(N)$ be a homomorphism. Since $\beta_\Sigma(N)_s \subseteq N_s$, we can define $h^\#\colon \gamma_\Sigma(M) \longrightarrow N$ by

$$h^\#_s(a) = \begin{cases} h(a), & \text{if } a \in M_s \\ \nu^\#(a), & \text{if } a \in \bar{M}_s \end{cases}$$

where $\nu\colon |M| \longrightarrow N$ is defined by $\nu(a) = h(a)$. The first case is determined by the requirement that $\beta_\Sigma(h^\#) = h$, while the second case is determined by the requirement that $h^\#$ is a homomorphism. Thus, $h^\#$ is the unique homomorphism from $\gamma_\Sigma(M)$ to N that is mapped to h under β_Σ. $\square$

Concerning model-bijectivity, by adding to $\Phi(\Sigma)$ the sentences $\neg\ x = \bot_s \Rightarrow D_s(x)$ (for each $s \in S$), $\bot$ becomes the *unique* "undefined" element. With this, we get an institution representation $(4a)$ that is model-bijective. Moreover, the translation of strong equations can be simplified by translating them just to ordinary equations. However, the representation then is no longer persistently liberal.

4.1.5 (4b): Mapping $PFOL^=$ to $FOL^=$ via functions as graphs

The representations of the previous section are quite good. However, in some cases, one needs a representation with stronger properties (like being an embedding or a subinstitution representation). Indeed, it is possible to represent $PFOL^=$ within $FOL^=$ as a subinstitution. This is achieved by representing functions by their graphs. The benefit of obtaining a subinstitution representation comes at a high price, however: the translation of sentences will turn out to be rather clumsy.

Signatures A $PFOL^=$-signature (S, TF, PF, P) is translated to the $FOL^=$-presentation $\langle (S, TF, P \uplus G), \Gamma \rangle$, where G contains a predicate symbol

$$G^f : s_1 \ldots s_n\ s$$

and Γ contains an axiom

$$\forall x_1 : s_1, \ldots, x_n : s_n; y, z : s \bullet G^f(x_1, \ldots, x_n, y) \wedge G^f(x_1, \ldots, x_n, z) \Rightarrow y = z$$

for each $f\colon s_1 \ldots s_n \longrightarrow s \in PF$. The predicate G^f shall hold the graph of the partial operation f, and the axiom states that G^f is right-unique, that is, the graph of a partial operation.

Models β_Σ takes a $\Phi(\Sigma)$-model M and replaces each relation G^f_M by the partial operation with graph G^f_M.

Sentences Following Burmeister [13, p. 325], we translate sentences into a relational form. α_Σ is defined inductively as follows:

- $\alpha_\Sigma(f(t_1, \ldots, t_n) \stackrel{e}{=} t_0) =$

  $$\exists y_0 : s_0 \ldots y_n : s_n \bullet G^f(y_1, \ldots, y_n, y_0) \wedge \alpha_\Sigma(t_0 \stackrel{e}{=} y_0) \wedge \cdots \wedge \alpha_\Sigma(t_n \stackrel{e}{=} y_n)$$

  for $f\colon s_1 \ldots s_n \longrightarrow s_0 \in PF$, where $y_0, \ldots, y_n$ are new variables,
- $\alpha_\Sigma(f(t_1, \ldots, t_n) \stackrel{e}{=} t_0) =$

  $$\exists y_0 : s_0 \ldots y_n : s_n \bullet f(y_1, \ldots, y_n) = y_0 \wedge \alpha_\Sigma(t_0 \stackrel{e}{=} y_0) \wedge \cdots \wedge \alpha_\Sigma(t_n \stackrel{e}{=} y_n)$$

  for $f\colon s_1 \ldots s_n \longrightarrow s_0 \in TF$, where $y_0, \ldots, y_n$ are new variables,
- $\alpha_\Sigma(x \stackrel{e}{=} t) = \alpha_\Sigma(t \stackrel{e}{=} x)$, if t is not a variable,
- $\alpha_\Sigma(x \stackrel{e}{=} y) = x = y$,
- $\alpha_\Sigma(t_1 = t_2) = (\alpha_\Sigma(x \stackrel{e}{=} t_1) \Rightarrow \alpha_\Sigma(x \stackrel{e}{=} t_2)) \wedge (\alpha_\Sigma(x \stackrel{e}{=} t_2) \Rightarrow \alpha_\Sigma(x \stackrel{e}{=} t_1))$, where x is a new variable,
- $\alpha_\Sigma(p(t_1, \ldots, t_n)) =$

  $$\exists y_1 : s_1 \ldots y_n : s_n \bullet p(y_1, \ldots, y_n) \wedge (t_1 \stackrel{e}{=} y_1) \wedge \cdots \wedge \alpha_\Sigma(t_n \stackrel{e}{=} y_n)$$

  for $p : s_1 \ldots s_n \in P$, where $y_1, \ldots, y_n$ are new variables,
- $\alpha_\Sigma(def(t)) = \exists x : s \bullet \alpha_\Sigma(t \stackrel{e}{=} x)$, where x is a new variable and t is of sort s,
- $\alpha_\Sigma(F) = F$,
- $\alpha_\Sigma(\varphi \wedge \psi) = \alpha_\Sigma(\varphi) \wedge \alpha_\Sigma(\psi)$,
- $\alpha_\Sigma(\varphi \Rightarrow \psi) = \alpha_\Sigma(\varphi) \Rightarrow \alpha_\Sigma(\psi)$,
- $\alpha_\Sigma(\forall x : s \bullet \varphi) = \forall x : s \bullet \alpha_\Sigma(\varphi)$.

Satisfaction The representation condition essentially is the proposition on page 326 of [13].

4.1.6 (4′), (4a′) and (4b′): Mapping $PCFOL^=$ to $CFOL^=$

The representation $(4a')$ extends the model-bijective institution representation $(4a)$ described above. Due to the need to translate sort generation constraints, we must be able to generate also the "undefined" elements in the $CFOL^=$-models. In order to be able to correctly model generatedness w.r.t. to partial functions, we have to ensure generatedness of the "undefined" elements. This is ensured by the axioms $\neg\ x = \bot_s \Rightarrow\ D_s(x)$ (for each $s \in S$). Since then there is just one "undefined" element, namely $\bot$, the "undefined" elements are now term generated. However, this method leads to a loss of the persistent liberality of the representation.

Signatures Signatures are translated as in $(4a)$.

Models Models are translated as in (4*a*).

Sentences First-order sentences are translated as in (4*a*). A Σ-sort generation constraint $(\overset{\bullet}{S}, \overset{\bullet}{F}, \theta\colon \bar{\Sigma} \longrightarrow \Sigma)$ is translated to $(\overset{\bullet}{S}, \overset{\bullet}{F} \cup\{\bot_s : s | s \in S\}, \theta')$, where $\theta'\colon \Phi(\bar{\Sigma}) \longrightarrow \Phi(\Sigma)$ is the extension of θ to $\Phi(\bar{\Sigma})$ that is defined by mapping the added symbols in $\Phi(\bar{\Sigma})$ to the corresponding added symbols in $\Phi(\Sigma)$.

Satisfaction The representation condition for formulas is proved as in (4a). Concerning sort generation constraints, note that model translation just removes the interpretation of $\bot$ from the carriers. Since $\bot$ cannot occur in $\overset{\bullet}{F}$ and moreover all functions are strict, generatedness w.r.t. $\overset{\bullet}{F}$ in the model $\beta_\Sigma(M')$ (where interpretations of $\bot_s$ have been removed) is the same as generatedness w.r.t. $\overset{\bullet}{F} \cup\{\bot_s : s | s \in S\}$ in the model M' (where interpretations of $\bot_s$ are not removed). $\square$

If one wants to have a strongly persistently liberal representation (call it (4′), since it extends (4)) also translating sort generation constraints, one can restrict sort generation constraints in $PCFOL^=$ to those including total function symbols only: in this case, the introduction of a unique undefined element is not necessary.

The same restriction of sort generation constraints in $PCFOL^=$ also has to be done if we want to extend (4*b*) to (4*b*′): sort generation constraints can just be left as they are in this case.

4.1.7 (5), (5*a*) *and* (5*b*)*: Mapping* $SubPFOL^=$ *to* $SubFOL^=$

The idea here is again to add new elements (among them $\bot$) that represent "undefined", but to retain the old sorts without the undefined elements. That is, the sort set is doubled: each sort s now gets a companion supersort $s^?$, which contains all values of s plus undefined elements. Since we keep the old sorts, the definedness predicate is just membership in the old sorts. Moreover, we add the companion supersort only to those sorts for which it is necessary, i.e. for those where a term involving partial function symbols exists.

In the signature translation, we introduce two versions for each function and predicate symbol: the original one and its (overloaded) strict extension that just propagates the undefined element. The strict extension is necessary if we want to apply a function or predicate symbol to a term involving partial function symbols, since this term can be translated only to a term of sort $s^?$.

Since all partial functions with result sort s are totalized to functions with result sort $s^?$, also the partial projections have to be totalized. For them, we introduce new special function symbols `proj`. We do not need an axiom like $\exists x : s \bullet D_s(X)$ in (4), because subsorts are interpreted with non-empty carriers

anyway.

Signatures Let $\Sigma = (S, TF, PF, P, \leq_S)$ be a $SubPFOL^=$-signature. Let

- $S^? = \{s \in S \mid$ there exists a variable system X and a $\Sigma(X)$-term of sort s built using at least one partial function symbol or projection cast$\}$,
- $S' = S \cup \{s^? \mid s \in S^?\}$ [11] and $Q(s) = \begin{cases} s^?, & \text{if } s \in S^? \\ s, & \text{otherwise} \end{cases}$

Then Σ is translated to the $SubFOL^=$-signature $\mathbf{Sig}(\Phi(\Sigma))$ consisting of the sort set S', ordered by

- $s \leq' s'$ iff $s \leq s'$ $(s, s' \in S)$;
- $s \leq' s'^?$ iff $s \leq s'$ $(s, s' \in S)$;
- $s^? \not\leq' s'$ $(s, s' \in S)$, and
- $s^? \leq' s'^?$ iff $s \leq s'$, $(s, s' \in S)$;

augmented by the following total function symbols:

$\bot : s^?$	for $s \in S^?$
$f\colon s_1, \ldots, s_n \to s$	for $f\colon s_1, \ldots, s_n \to s \in TF$
$f\colon s_1, \ldots, s_n \to s^?$	for $f\colon s_1, \ldots, s_n \to ? s \in PF$,
$f\colon Q(s_1), \ldots, Q(s_n) \to s^?$	for $f\colon s_1, \ldots, s_n \longrightarrow s \in TF \cup PF$, $\{s_1, \ldots, s_n\} \cap S^? \neq \emptyset$
$\mathtt{proj}\colon s \to s'^?$	for $s' \leq s$
$\mathtt{proj}\colon s^? \to s'^?$	for $s' \leq s$

the following predicate symbols:

$p\colon s_1, \ldots, s_n$	for $p\colon s_1, \ldots, s_n \in P$,
$p\colon Q(s_1), \ldots, Q(s_n)$	for $p\colon s_1, \ldots, s_n \in P, \{s_1, \ldots, s_n\} \cap S^? \neq \emptyset$

and the following axioms $\mathbf{Ax}(\Phi(\Sigma))$:

$\neg \bot \in s$ for $s \in S^?$ (1)

$\forall x : s' \bullet \mathtt{proj}_{(s', s^?)}(x) \in s \Leftrightarrow x \in s$ for $s \leq s'$ (2)

$\forall x : s \bullet \mathtt{proj}_{(s', s^?)}(\mathtt{inj}_{(s, s')}(x)) = \mathtt{inj}_{(s, s^?)}(x)$ for $s \leq s'$ (3)

$\forall x, y : s' \bullet \mathtt{proj}_{(s', s^?)}(x) = \mathtt{proj}_{(s', s^?)}(y) \wedge \mathtt{proj}_{(s', s^?)}(x) \in s \Rightarrow x = y$ for $s \leq s'$ (4)

$\forall x_1 : Q(s_1), \ldots, x_n : Q(s_n) \bullet \ f(x_1, \ldots, x_n) \in s \Leftrightarrow \bigwedge_{s_i \in S^?} x_i \in s_i$ for $f\colon s_1, \ldots, s_n \to s \in TF$ (5)

$\forall x_1 : Q(s_1), \ldots, x_n : Q(s_n) \bullet \ f(x_1, \ldots, x_n) \in s \Rightarrow \bigwedge_{s_i \in S^?} x_i \in s_i$ for $f\colon s_1, \ldots, s_n \to ? s \in PF$ (6)

$\forall x_1 : Q(s_1), \ldots, x_n : Q(s_n) \bullet \ p(x_1, \ldots, x_n) \Rightarrow \bigwedge_{s_i \in S^?} x_i \in s_i$ for $p\colon s_1, \ldots, s_n \in P$ (7)

The axioms of form (1) state that $\bot$ is an "undefined" element. (2), (3) and (4) express the membership, projection and projection-injectivity axioms – these are those of the axioms used in the definition of $SubPCFOL^=$

[11] We assume that S does not contain sorts of the form $s^?$. In a higher-order extension of CASL, there could be a type constructor "?" behaving like our "?".

(see Section 3.2) that refer to the partial projections. (5), (6) and (7) state strictness of the functions and predicates, while (5) additionally expresses totality of the total functions.

Signature morphisms are translated in a way analogous to that of Section 4.1.4.

Models A $\Phi(\Sigma)$-structure M (in $SubFOL^=$) is translated to the partial Σ-structure $\beta_\Sigma(M)$ (in $SubPFOL^=$) having carrier sets M_s for $s \in S$ and total functions and predicates inherited from M, while partial functions $g_{\beta_\Sigma(M)}$ are the corestrictions of g_M to M_s (where s is the result sort of g).

Again, we have a straightforward inverse: a Σ-structure M is extended by adding new carriers $M_{s^?} = M_s \uplus \{*\}$, interpreting $\bot$ as $*$ and adding the strict extensions that map $*$ to itself. Thus, we have an institution representation that admits model expansion.

Sentences A Σ-sentence in $SubPFOL^=$ is translated to a $\Phi(\Sigma)$-sentence in $SubFOL^=$ inductively as follows:

- $\alpha_\Sigma(x : s) = x : s$
- $\alpha_\Sigma(f_{(s_1,\ldots,s_n,s)}(t_1,\ldots,t_n)) = \qquad (f\colon s_1,\ldots,s_n \to s \in TF)$
  $\begin{cases} f_{\langle s_1,\ldots,s_n,s\rangle}(\alpha_\Sigma(t_1),\ldots,\alpha_\Sigma(t_n)\rangle \text{ if for } i = 1\ldots n,\ \alpha_\Sigma(t_i) \text{ is of sort } s_i \\ f_{(Q(s_1),\ldots,Q(s_n),(s^?))}(\alpha_\Sigma(t_1),\ldots,\alpha_\Sigma(t_n)),\ \text{otherwise} \end{cases}$
- $\alpha_\Sigma(f_{(s_1,\ldots,s_n,s)}(t_1,\ldots,t_n)) = \qquad (f\colon s_1,\ldots,s_n \to ?\ s \in PF)$
  $\begin{cases} f_{(s_1,\ldots,s_n,(s^?))}(\alpha_\Sigma(t_1),\ldots,\alpha_\Sigma(t_n)) \text{ if for } i = 1\ldots n,\ \alpha_\Sigma(t_i) \text{ is of sort } s_i \\ f_{(Q(s_1),\ldots,Q(s_n),(s^?))}(\alpha_\Sigma(t_1),\ldots,\alpha_\Sigma(t_n)),\ \text{otherwise} \end{cases}$
- $\alpha_\Sigma(p_{s_1,\ldots,s_n}(t_1,\ldots,t_n)) = \qquad (p : s_1,\ldots,s_n \in P)$
  $\begin{cases} p_{s_1,\ldots,s_n}(\alpha_\Sigma(t_1),\ldots,\alpha_\Sigma(t_n)) \text{ if for } i = 1\ldots n,\ \alpha_\Sigma(t_i) \text{ is of sort } s_i \\ p_{Q(s_1),\ldots,Q(s_n)}(\alpha_\Sigma(t_1),\ldots,\alpha_\Sigma(t_n)),\ \text{otherwise} \end{cases}$
- $\alpha_\Sigma(\mathtt{pr}_{(s_1,s_2)}(t)) = \begin{cases} \mathtt{proj}_{(s_1^?,s_2^?)}(\alpha_\Sigma(t)),\ \text{if } \alpha_\Sigma(t) \text{ is of sort } s_1^? \\ \mathtt{proj}_{(s_1,s_2^?)}(\alpha_\Sigma(t)),\ \text{otherwise} \end{cases} \quad (s_2 \leq s_1)$
- $\alpha_\Sigma(def(t)) = \begin{cases} \alpha_\Sigma(t) \in s,\ \text{if } \alpha_\Sigma(t) \text{ is of sort } s^?, s \in S^? \\ T, \qquad \text{otherwise} \end{cases}$
- $\alpha_\Sigma(t \in s_1) = \begin{cases} t \in s_1^?,\ \text{if } \alpha_\Sigma(t) \text{ is of sort } s_2^? \\ t \in s_1,\ \text{otherwise} \end{cases}$
- $\alpha_\Sigma(t_1 = t_2) = \begin{cases} (\alpha_\Sigma(t_1) \in s \vee \alpha_\Sigma(t_2) \in s) \Rightarrow \alpha_\Sigma(t_1) = \alpha_\Sigma(t_2), \\ \qquad\qquad \text{if } \alpha_\Sigma(t_1), \alpha_\Sigma(t_2) \text{ are of sort } s^? \\ \alpha_\Sigma(t_1) = \alpha_\Sigma(t_2), \qquad \text{otherwise} \end{cases}$
- $\alpha_\Sigma(t_1 \stackrel{e}{=} t_2) = \begin{cases} \alpha_\Sigma(t_1) = \alpha_\Sigma(t_2) \wedge \alpha_\Sigma(t_1) \in s, \\ \qquad\qquad \text{if } \alpha_\Sigma(t_1), \alpha_\Sigma(t_2) \text{ are of sort } s^? \\ \alpha_\Sigma(t_1) = \alpha_\Sigma(t_2), \qquad \text{otherwise} \end{cases}$

Note that in the last two cases, if the translation of only one term is of a "question mark" sort, the translation of the other one has to be made to be of "question mark" sort as well (by inserting an appropriate injection). The translation can easily be extended from atomic to all formulas.

Apart from mapping definedness and projection to the special symbols introduced above, the translation mainly checks whether the original version of a function or predicate symbol can be used, or if we have to use its strict

extension that propagates the undefined element. The strict extension is necessary whenever an argument term involves a partial function symbol.

Satisfaction The representation condition can be proved in a way similar to the proof of that of (4). □

With the argument of Example 2.30, the representation has the weak $\mathcal{D}$-amalgamation property, where $\mathcal{D}$ is the class of all injective signature morphisms.

Proposition 4.3 The above constructed institution representation is strongly persistently liberal.

Proof. For a $SubPFOL^=$-signature $\Sigma = (S, TF, PF, P, \leq)$, let $\Phi'(\Sigma)$ be the $SubPFOL^=$-presentation resulting from removing the axioms of form (1) from $\Phi(\Sigma)$, while adding the partial function symbols from PF and the axioms

$$\forall x_1 : s_1, \ldots x_n : s_n \bullet f_{s_1 \ldots s_n, s^?}(x_1, \ldots x_n) \in s \Leftrightarrow def\ f_{s_1 \ldots s_n, s}(x_1, \ldots x_n) \quad (8)$$

$$\forall x_1 : s_1, \ldots x_n : s_n \bullet def\ f_{s_1 \ldots s_n, s}(x_1, \ldots x_n) \Rightarrow \mathtt{inj}_{(s, s^?)}(f_{s_1 \ldots s_n, s}(x_1, \ldots x_n)) = f_{s_1 \ldots s_n, s^?}(x_1, \ldots x_n) \quad (9)$$

for each $f : s_1, \ldots, s_n \rightarrow ?\ s \in PF$. The axioms relate the partial functions with their total encoding, in such a way that a partial function is uniquely determined by its total encoding, and moreover, the way in which it is determined corresponds to the action of β_Σ.

In this way, $\Phi'(\Sigma)$ includes both Σ and $\Phi(\Sigma)$ (except from the axioms of form (1)). Let ι_Σ and κ_Σ be the inclusions of Σ and $\Phi(\Sigma)$ into $\Phi'(\Sigma)$. By Theorem 4.16 [12], the free functor $F_{\iota_\Sigma} : \mathbf{Mod}(\Sigma) \longrightarrow \mathbf{Mod}(\Phi'(\Sigma))$ exists. Moreover, since $\bot \in s$ does not follow from $\Phi'(\Sigma)$, models constructed by this free functor also satisfy the axioms of form (1).

We can now define γ_Σ to act as follows: Given a Σ-model M,

$$\gamma_\Sigma(M) := F_{\iota_\Sigma}(M)|_{\kappa_\Sigma}.$$

It is tedious but not difficult to check that ι_Σ is sufficiently complete, i.e.

$$M \cong F_{\iota_\Sigma}(M)|_{\iota_\Sigma}$$

via the unit of the adjunction. By Lemma 8.14 of [28], we can choose F_{ι_Σ} such that the identity is the unit of the adjunction, i.e. $M = F_{\iota_\Sigma}(M)|_{\iota_\Sigma}$. But then, since the unique determination of a partial function through (8) and (9) is the

[12] Though this is a forward reference, it is not circular: Theorem 4.16 depends on several subinstitution representations presented in Section 4.2, but not on the institution representation currently being considered.

same as the construction of the partial function through β_Σ, we get

$$\beta_\Sigma(\gamma_\Sigma(M)) = M.$$

To show the universal property of $M \xrightarrow{id} \beta_\Sigma(\gamma_\Sigma(M))$, let $h\colon M \longrightarrow \beta_\Sigma(N)$ be a Σ-homomorphism. Let $N' \in \mathbf{Mod}(\Phi'(\Sigma))$ be the expansion of N with new partial functions (uniquely determined by (8) and (9)). We thus have $N'|_{\kappa_\Sigma} = N$ and, since the expansion acts like β, $\beta_\Sigma(N) = N'|_{\iota_\Sigma}$. By the universal property of F_{ι_Σ}, we get $h^\#\colon M \longrightarrow N'|_{\iota_\Sigma}$ with $h^\#|_{\iota_\Sigma} = h$. Hence, $h^\#|_{\kappa_\Sigma}\colon \gamma_\Sigma(M) \longrightarrow N$ is the required extension of h with $\beta_\Sigma(h^\#|_{\kappa_\Sigma}) = h^\#|_{\iota_\Sigma} = h$. $\square$

Also, by modifying (5) (similar to the passage from (4) to (4a)), we get a representation (5a) that is model-bijective. The exact axioms are described in (5a') below.

Further, we can extend the representation (4b) to get a representation (5b) of $SubPFOL^=$ in $SubFOL^=$ that is even a subinstitution representation. For signatures and models, the extension of (4b) is quite straightforward: the subsorting structure is just kept. Sentence translation is extended as follows:

- $\alpha_\Sigma(\mathtt{pr}_{(s',s)}(t) \stackrel{\mathrm{e}}{=} t_0) \;=\; \exists x : s \bullet \alpha_\Sigma(\mathtt{inj}_{(s,s')}(x) \stackrel{\mathrm{e}}{=} t) \wedge \alpha_\Sigma(x \stackrel{\mathrm{e}}{=} t_0)$, where x is a new variable,
- $\alpha_\Sigma(t \in s) \;=\; \exists x : s \bullet \alpha_\Sigma(t \stackrel{\mathrm{e}}{=} x) \wedge x \in s$, where x is a new variable.

Due to the presence of subsorting, we also can replace the represention of partial functions as graphs by another one: we can, for each partial function, introduce a new (sub)sort holding its domain, and thus making the function total. However, note that sentence translation will become equally clumsy as for (5b) above, since it will involve existential quantifiers over the domains. Moreover, since the domain generally is a subsort of a product and $SubFOL^=$ lacks products, we have to introduce products explicitly.

4.1.8 (5′), (5a′) and (5b′): Mapping SubPCFOL= to SubCFOL=

Again, the translation (5a') is very similar to that of the previous section. As in (4a'), due to the need to translate sort generation constraints, we must be able to generate also the "undefined" elements in the $SubCFOL^=$-models. One could think of using the sorts without a question mark for the translation of sort generation constraints. However, this does not work for sort generation constraints involving partial function symbols: partial function symbols are always translated to a function symbol with result sort of form $s^?$. Also, the projection symbols do not help to get back to s: the also have result sort $s^?$. This forces us to consider generation of the $s^?$ sorts.

In order to be able to do this, we again restrict the models to those with exactly one "undefined" element. This leads to a loss of persistent liberality of the representation.

Signatures A signature is translated as in (5), with the addition that the sentences $\neg\ x = \bot \Rightarrow x \in s$ (for each $s \in S$) are added to the presentation. Thus, $\bot_s$ becomes the *unique* "undefined" element.
Models Models are translated exactly as in (5).
Sentences First-order sentences are translated as in (5). A Σ-sort generation constraint $(\overset{\bullet}{S}, \overset{\bullet}{F}, \theta{:}\,\bar{\Sigma} \longrightarrow \Sigma)$ is translated to $(\overset{\bullet}{S}, \overset{\bullet\,\prime}{F} \cup\{\bot_s : s | s \in S\}, \theta')$, where $\overset{\bullet\,\prime}{F}$ is $\overset{\bullet}{F}$ with all function symbols of form $\mathtt{pr}_{(s,s')}$ replaced by $\mathtt{proj}_{(s,s'^{?})}$ and $\theta'{:}\,\Phi(\bar{\Sigma}) \longrightarrow \Phi(\Sigma)$ is the extension of θ to $\Phi(\bar{\Sigma})$ that is defined by mapping the added symbols in $\Phi(\bar{\Sigma})$ to the corresponding added symbols in $\Phi(\Sigma)$.
Satisfaction The representation condition for formulas is proved as in $(4a')$. □

As in $(4')$, we can restore strongly persistent liberality by a restriction of sort generation constraints in $SubPCFOL^{=}$ to those not involving partial function symbols: in this case, the extra axioms for the $\bot$ functions are not needed. We thus get a representation $(5')$.

Moreover, with the same restriction of $SubPCFOL^{=}$, we also can extend $(5b)$ to $(5b')$ by just leaving sort generation constraints unchanged.

4.1.9 (6) and (6′): Mapping Sub(C)FOL= to (C)FOL=

Here we can use a restriction of the translation of $Sub(C)PFOL^{=}$ to $P(C)FOL^{=}$ described in Section 4.1.3 above, which leaves out the partial projection symbols and those axioms from $\hat{J}(\Sigma)$ involving these. There is one problem connected with this: the membership predicate was originally axiomatized using partial projections and definedness:

$$\forall x : s'\bullet \in^{s}_{s'} (x) \Leftrightarrow \mathit{def}\ \mathtt{pr}_{(s',s)}(x)$$

for $s \leq s'$. Now we cannot just leave out these axioms, since then the membership predicates could be interpreted in an unintended way. Therefore, these axioms have to be replaced by

$$\forall x : s'\bullet \in^{s}_{s'} (x) \Leftrightarrow \exists y \bullet \mathtt{inj}_{(s,s')}(y) = x$$

4.1.10 Comparison of (4)∘(3) with (6)∘(5)

Comparing both ways of encoding $SubPFOL^{=}$ into $FOL^{=}$, we can see that the way via $SubFOL^{=}$ has the disadvantage that many new sorts are introduced,

but the advantage than the original sort system and its interpretation are kept. Moreover, many-sorted total specifications are left unchanged (but the encoding via $PFOL^=$ may also be adapted to have this property).

4.1.11 (7)*: Mapping $CFOL^=$ to $SOL^=$*

$CFOL^=$ adds sort generation constraints to $FOL^=$. These cannot be expressed within $FOL^=$, but can be translated to induction schemes, which can be expressed in second-order logic with equality ($SOL^=$) by second-order quantification over predicates.

Signatures Signatures and signature morphisms are translated identically.

Models Models, model homomorphisms and reducts are translated identically.

Sentences First-order sentences are translated identically. For a sort generation constraint

$$(\dot{S}, \dot{F}, \theta\colon \bar{\Sigma} \longrightarrow \Sigma)$$

we assume without loss of generality that all the result sorts of function symbols in $\dot{F}$ occur in $\dot{S}$ (see the corresponding remark in Section 3.1 where sort generation constraints have been introduced). Let

$$\dot{S} = \{s_1; \ldots; s_n\}, \quad \dot{F} = \{f_1\colon s_1^1 \ldots s_{m_1}^1 \longrightarrow s^1; \ldots; f_1\colon s_1^k \ldots s_{m_k}^k \longrightarrow s^k\}$$

The sort generation constraint is now translated to the $SOL^=$-sentence

$$\forall P_{s_1} : pred(\theta(s_1)) \ldots \forall P_{s_n} : pred(\theta(s_n)) \bullet (\varphi_1 \wedge \cdots \wedge \varphi_k) \Rightarrow \textstyle\bigwedge_{j=1,\ldots,n} \forall x : \theta(s_j) \bullet P_{s_j}(x)$$

where

$$\varphi_j = \forall x_1 : \theta(s_1^j), \ldots, x_{m_j} : \theta(s_{m_j}^j) \bullet \left(\textstyle\bigwedge_{i=1,\ldots,m_j;\ s_i^j \in \dot{S}} P_{s_i^j}(x_i)\right) \Rightarrow P_{s^j}\left(\theta(f_j)(x_1, \ldots, x_{m_j})\right)$$

Satisfaction To prove the representation condition, let a Σ-model M and a Σ-sort generation constraint $(\dot{S}, \dot{F}, \theta\colon \bar{\Sigma} \longrightarrow \Sigma)$ be given. Call an $\bar{S}$-sorted set $\bar{\mathcal{P}} \subseteq |M|_\theta|$ $(\dot{S}, \dot{F}, \theta)$*-closed* iff it is closed under the application of functions $\theta(f)_M$ with $f \in \dot{F}$ and the filling in of arbitrary values of $M_{\theta(s)}$ for $s \notin \dot{S}$.

Lemma 4.4 Let X be the variable system $\{P_{s_1} : pred(\theta(s_1)), \ldots, P_{s_n} : pred(\theta(s_n))\}$. Given an assignment $\xi\colon X \longrightarrow M$, consider the $\bar{S}$-sorted set $\bar{\mathcal{P}}(\xi)$ consisting of $\xi(P_s)$ for $s \in \dot{S}$ and of $M_{\theta(s)}$ for $s \notin \dot{S}$.

Then we have

$$\xi \Vdash \varphi_1 \wedge \cdots \wedge \varphi_k \text{ iff } \bar{\mathcal{P}}(\xi) \text{ is } (\dot{S}, \dot{F}, \theta)\text{-closed.}$$

Proof. This directly follows from the form of the φ_j. Note that the filling in of arbitrary values of $M_{\theta(s)}$ for $s \notin \dot{S}$ is captured by the condition $s_i^j \in \dot{S}$ in the conjunction in the premise of φ_j: for $s_i^j \notin \dot{S}$, nothing is required. □

Now M satisfies the sort generation constraint $(\dot{S}, \dot{F}, \theta\colon \bar{\Sigma} \longrightarrow \Sigma)$
iff the smallest $(\dot{S}, \dot{F}, \theta)$-closed set is $|M|_\theta|$
iff for all $\xi\colon X \longrightarrow M$, $\bar{\mathcal{P}}(\xi)$ is $(\dot{S}, \dot{F}, \theta)$-closed implies $\bar{\mathcal{P}}(\xi) = |M|_\theta|$
iff for all $\xi\colon X \longrightarrow M$,
$\xi \Vdash \varphi_1 \wedge \cdots \wedge \varphi_k$ implies $\xi \Vdash \bigwedge_{j=1,\ldots,n} \forall x : \theta(s_j) \bullet P_{s_j}(x)$
iff M satisfies the translation of the sort generation constraint. □

4.2 The Positive Conditional Level

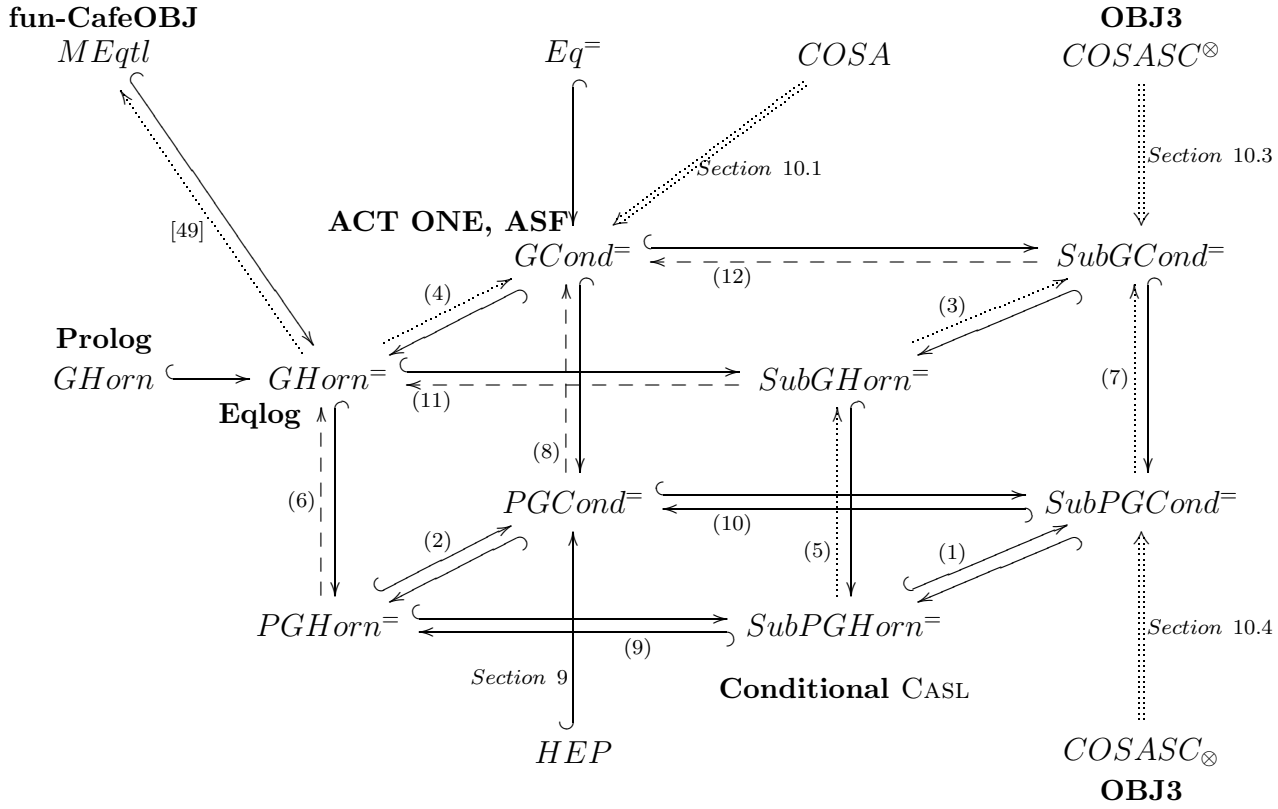
At the positive conditional level (see Fig. 5), we have used generalized conditional logic ($GCond^=$, $GHorn^=$ etc.). The reason for this is that we want to be able to use equivalences (and not just implications) in the translations of sentences, which would not be possible within ordinary conditional logic ($Cond^=$, $Horn^=$ etc.). However, note that there is a *conjunctive* subinstitution representation from any of the generalized conditional logics to the corresponding ordinary conditional logic. Note also that in most cases, the diagram remains the same if the G is deleted everywhere (only a few arrows become conjunctive representations).

$MEqtl$ denotes Meseguer's Membership Equational Logic, see [49], where also the arrows to and from $MEqtl$ are described. Also, Equational Type Logic [46] and Unified Algebras [63] and similar frameworks could be added at the same corner of the diagram (however, the arrow from $GHorn^=$ will not be persistently liberal in all these cases).

At the positive conditional level, we have the following levels of expressiveness:

(1) The institutions $SubPGHorn^=$, $PGHorn^=$, $SubPGCond^=$ and $PGCond^=$ (all involving partiality) are all subinstitutions of each other. They form the strongest level of expressiveness. By using more complicated representations, one also can count $SubGHorn^=$ and $SubGCond^=$ to belong to this level, see [53]. We have not included these representations here, since they are based on a weaker notion of embedding, which is too weak to be practically useful for borrowing.
(2) The next level of expressiveness is $GHorn^=$. Here, we lose the ability of specifying conditional generation of data using partial functions within **free** specifications, as shown in [52].
(3) Then comes $GCond^=$. As shown in [52], due to lack of predicates, we cannot specify conditional generation of relations using **free**, as in the transitive closure example (see Example 2.11).
(4) The lowest level is $Eq^=$, where we have no conditional axioms.

All these levels are separated from each other in Theorem 5.1.



Index for arrow types:

- 1. Admits model expansion: Borrowing of entailment and refinement for flat specifications
- 2. Strongly persistently liberal: Lifting of free constructions, implies 1.
- 3. Embedding: Borrowing of entailment for certain structured specifications including **free**
- 4. Model-bijective: Borrowing of entailment and refinement for structured specifications excluding **free**, implies 1.
- 5. Subinstitution: Borrowing of entailment and refinement for all structured specifications

Fig. 5. The positive conditional level

We now come the the description of the representations.

4.2.1 (1): Mapping $SubPGHorn^=$ to $SubPGCond^=$

Here, we cannot directly use the idea of representation (2) from the first-order level of replacing predicates by Boolean functions, since the set of Booleans

cannot be axiomatized monomorphically in positive conditional logic.[13] However, what we can do is to axiomatize a *single-valued* set $Bool1$ of truth-values and map predicates to *partial* functions into $Bool1$.

Signatures A signature is translated by adding disjointly to it the presentation

spec BOOL1 =
sort *Bool*
op *True1* : *Bool1*
axiom $\forall x, y : Bool1 \bullet x = y$
end

and replacing each predicate symbol $p : w$ by a partial function symbol $p : w \longrightarrow ?Bool1$.

Models A model M is translated by replacing each partial function $(p_{w,Bool1})_M$ by its domain, considered as a predicate. Note that model homomorphisms can be translated identically, since they behave in the same way for domains of partial functions and for predicates.

Sentences A sentence is translated by replacing each occurrence of a predicate symbol application $p_w(t_1, \ldots, t_n)$ by the equation $p_{w,Bool1}(t_1, \ldots, t_n) = True1$.

Satisfaction The representation condition is straightforward to show. □

It is easy to show this representation is a subinstitution representation.

4.2.2 (2): Mapping $PGHorn^=$ to $PGCond^=$

Here, we can use the same representation as in (1), just restricted to signatures with trivial subsort relation.

4.2.3 (3): Mapping $SubGHorn^=$ to $SubGCond^=$

Here, we can use a representation similar to that in (1). The only difference is that predicate symbols $p : w$ are translated to *total* function symbols $p : w \longrightarrow Bool1$. (Note that we cannot represent predicates by subsorts, since predicates may be empty, while subsorts may not.) This leads to a loss of the subinstitution property, but we still have a strongly persistently liberal representation, which can be seen as follows: Let $\Sigma = (S, TF, P, \leq)$ be a signature in $SubGHorn^=$. Given a Σ-model M, $\gamma_\Sigma(M)$ extends M by interpreting $Bool1$

[13] Actually, since positive conditional logic is closed under products by the well-known Mal'cev theorem, any theory having a model with a two-valued Boolean carrier set has also models where the Boolean carrier set exceeds any given cardinality.

as

$$\{true\} \uplus \{p_w(a_1, \ldots, a_n) \,|\, p : w \in P, a_1, \ldots, a_n \in M_w \setminus p_M\}$$

$True1$ is interpreted with $true$, and

$$(p_w)_{\gamma_\Sigma(M)}(a_1, \ldots, a_n) = \begin{cases} true, & \text{if } (a_1, \ldots, a_n) \in p_M \\ p_w(a_1, \ldots, a_n), & \text{otherwise} \end{cases}$$

Clearly, $\beta_\Sigma(\gamma_\Sigma(M)) = M$. Now let $h\colon M \longrightarrow \beta_\Sigma(N)$ be a Σ-homomorphism. Then h can be uniquely extended to a $\Phi(\Sigma)$-homomorphism $h^\#\colon \gamma(M) \longrightarrow N$ with $\beta_\Sigma(h^\#) = h$ by putting

$$(h^\#)_{Bool1}(true) = True1_N$$
$$(h^\#)_{Bool1}(p_w(a_1, \ldots, a_n)) = (p_w)_N(h(a_1), \ldots, h(a_n))$$

4.2.4 (4): Mapping $GHorn^=$ to $GCond^=$

Here, we can use the same representation as in (3), just restricted to signatures with trivial subsort relation. Again, the representation is strongly persistently liberal.

4.2.5 (5): Mapping $SubPGHorn^=$ to $SubGHorn^=$

This strongly persistently liberal representation works like the representation (5) from the first-order level, except that the constants $\bot$ and their axiomatizations are omitted: then, all axioms in $\Phi(\Sigma)$ are in Horn form. The omission of $\bot$ leads to a loss of the weak (injective)-amalgamation property.

4.2.6 (6): Mapping $PGHorn^=$ to $GHorn^=$

This representation works similar to the representation (4) from the first-order level. As in (5), we have to remove the constants $\bot$ and their axiomatizations which are not in Horn form (thereby losing the weak (injective)-amalgamation property). Moreover, the axioms $\exists x : s \bullet D_s(x)$ are not in Horn form either. These axioms have to be replaced by the introduction of new constants $c_s : s$ (for each $s \in S$). The new constant has essentially the same effect as the existential axiom, except that homomorphisms have to preserve it, which leads to a loss of persistent liberality (but the representation still admits model expansion).

4.2.7 (7): Mapping $SubPGCond^=$ to $SubGCond^=$

This works like the (strongly persistently liberal) representation (5) above.

4.2.8 (8): Mapping PGCond$^=$ to GCond$^=$

Here, we cannot restrict (4) from the first-order level, since we need to eliminate the definedness predicates. Therefore, we use the composition

$$PGCond^{=} \hookrightarrow PGHorn^{=} \xrightarrow{(6)} GHorn^{=} \overset{(4)}{\dashrightarrow} GCond^{=}$$

which is a representation admitting model expansion.

4.2.9 (9): Mapping SubPGHorn$^=$ to PGHorn$^=$

This subinstitution representation works exactly as the representation (3) from the first-order level: all axioms in $\Phi(\Sigma)$ are already in Horn form.

4.2.10 (10): Mapping SubPGCond$^=$ to PGCond$^=$

This representation works similar to (9). The difference it that the membership predicates have to be deleted, Now when looking at the axioms in $\hat{J}$, one can see that the membership predicates are just the domains of the partial projections. Thus, all occurrences of

$$t \in s$$

in sentences have to be translated to

$$def\ \mathtt{pr}_{(s',s)}(t).$$

4.2.11 (11): Mapping SubGHorn$^=$ to GHorn$^=$

When setting up this representation, we encounter the difficulty to axiomatize the membership predicates. In $\hat{J}$, they are axiomatized as the domains of the partial projections. Now we do not have partial functions at hand. In the representation (6) at the first-order level this problem is solved by the axiomatization

$$\forall x : s' \bullet \in^{s}_{s'} (x) \Leftrightarrow \exists y \bullet \mathtt{inj}_{(s,s')}(y) = x$$

But this axiom is not in Horn form. The best that we can do at the moment instead of this is just to use the composition

$$SubGHorn^{=} \hookrightarrow SubPGHorn^{=} \xhookrightarrow{(9)} PGHorn^{=} \xrightarrow{(6)} GHorn^{=}$$

which is a representation admitting model expansion.

4.2.12 (12): Mapping $SubGCond^{=}$ to $GCond^{=}$

For the same reasons as for (11) above, we use the composition

$$SubGCond^{=} \hookrightarrow SubPGCond^{=} \overset{(10)}{\hookrightarrow} PGCond^{=} \overset{(8)}{\twoheadrightarrow} GCond^{=}$$

which again is a representation admitting model expansion.

4.3 Theorem proving and liberality

We can now combine the results of the previous subsections with the general results about institution representations in Section 2 to obtain specific results about theorem provers and liberality.

Fact 4.5 $Cond^{=}$ allows conditional term rewriting and paramodulation as sound and complete proof systems.

Proof. See e.g. [67]. □

Theorem 4.6 Each of the institutions shown in Fig. 5 allows the re-use of conditional term rewriting and paramodulation as sound and complete proof systems for flat specifications.

Proof. By chasing the diagram in Fig. 5 and using the remark at the beginning of Section 4.2 about the representation of $GCond^{=}$ in $Cond^{=}$, one can see that each of the institutions in Fig. 5 can be represented in $Cond^{=}$ with a representation admitting model expansion. The result now follows from Fact 4.5 and Theorem 2.26. □

Theorem 4.7 Each of the institutions shown in Fig. 3 (except $SOL^{=}$) allows the re-use of a first-order theorem prover for flat specifications.

Proof. By chasing the diagram in Fig. 3, one can see that each of the institutions in Fig. 3 except $SOL^{=}$ can be represented in $FOL^{=}$ with a representation admitting model expansion. The result now follows from Theorem 2.26. □

Theorem 4.8 Each of the institutions shown in Fig. 4 (except $SOL^{=}$) allows the re-use of a first-order theorem prover with induction (or second-order, or higher-order prover) for flat specifications.

Proof. By chasing the diagram in Fig. 4, one can see that each of the institutions in Fig. 4 can be represented in $CFOL^{=}$ with a representation admitting model expansion. The result now follows from Theorem 2.26. □

Theorem 4.9 Each of the institutions shown in Fig. 3, except from $SOL^{=}$,

$COSASC^{\otimes}$-*Bool*, and $COSASC_{\otimes}$-*Bool* allows the re-use of first-order logic theorem provers for both entailments and refinements concerning structured specifications without **free**.

Proof. By chasing the diagram in Fig. 3, one can see that the above mentioned institutions have a model-bijective representation in $FOL^{=}$. The result now follows from Proposition 2.52 and Theorem 2.31. □

Theorem 4.10 Each of the institutions shown in Fig. 4 except from $SOL^{=}$ and *LSL-Bool* allows the re-use of first-order theorem provers with induction (or second-order, or higher-order provers) for both entailments and refinements concerning structured specifications without **free**.

Proof. By chasing the diagram in Fig. 4, one can see that each of the above mentioned institutions has a model-bijective representation in $CFOL^{=}$. The result now follows from Propositions 2.52 and Theorem 2.31. □

Theorem 4.11 For each of the institutions shown in Fig. 4 except *LSL-Bool* and $SOL^{=}$, take the subinstitution obtained by forbidding sort generation constraints with partial function symbols. Each of the thus obtained institutions allows

- the re-use of a checker for $CFOL^{=}$ that determines whether presentation morphisms are liberal, and
- the re-use of first-order theorem provers with induction (or second-order, or higher-order provers) for entailments concerning structured specifications SP (including **free**), such that for the strongly persistently liberal representation $(\Phi, \alpha, \beta), \gamma)$ into $CFOL^{=}$,
  - for each **free** SP' **along** $\sigma\colon \Sigma \longrightarrow \Sigma_1$ occurring in SP, either $\Phi(\sigma)\colon \Sigma \longrightarrow SP'$ is strongly persistently liberal or γ is σ-natural, and moreover, γ_Σ is surjective on objects,
  - for each **derive from** SP_1 **by** σ occurring in SP, γ is σ-natural,
  - for each **translate** SP_1 **by** σ occurring in SP, either γ is σ-natural, or SP_1 is flattenable, i.e. contains neither **free** nor **derive**.

Proof. By chasing the diagram in Fig. 4, one can see that each of the above mentioned institutions is a subinstitution of $SubPCFOL^{=}$ with sort generation constraints restricted to total functions. Representation $(4') \circ (3')$ of the first-order level describes a strongly persistently liberal institution representation from the thus restricted $SubPCFOL^{=}$ to $CFOL^{=}$. The result now follows from Theorems 2.43 and 2.45. □

Theorem 4.12 Consider the restriction of the CASL institution $SubPCFOL^{=}$ to sort generation constraints only with total function symbols. This institution can be represented in $CFOL^{=}$ in such a way that it is possible to

- re-use of first-order theorem provers with induction (or second-order, or higher-order provers) for both entailments and refinements concerning structured specifications without **free**, and with **derive** only along injective signature morphisms,
- re-use of a checker for $CFOL^=$ that determines whether presentation morphisms are liberal, and
- re-use first-order theorem provers with induction (or second-order, or higher-order provers) for entailments concerning structured specifications SP (including **free**), where SP obeys the restrictions listed in Theorem 4.11 above.

Proof. Compose representations (6′) and (5′) in Fig. 4, thus obtaining a strongly persistently liberal institution representation admitting weak (injective)-amalgamation going from (the restricted) $SubPCFOL^=$ to $CFOL^=$. The result now follows from Proposition 2.52 and Theorems 2.31, 2.43 and 2.45. □

Of course, the question arises whether the above semantic restrictions on the specifications that include **free** can be reformulated in more syntactic terms.

Concerning surjectivity of γ_Σ on objects, this can be ensured by simply letting the parameter signature Σ have no partial operations and no subsorts [14], since then both β_Σ and γ_Σ are isomorphisms.

Concerning strongly persistent liberality of $\Phi(\sigma)\colon \Sigma \longrightarrow SP'$, this can be ensured if SP' is a presentation and only introduces new operation symbols that are specified in a sufficiently complete way (i.e. such that each new term can be reduced to an old term). The crucial point is here that this must also hold (in the encoding) for terms containing partial operation symbols. That is, we have to specify their behaviour also in the undefined case (with $\neg def\ f(t_1, \ldots, t_n)$). However, in general this would destroy the existence of free models. But we can use a trick: namely, we can make sentences of form $\neg def\ t$ parts of signatures, and as parts of signatures, we can translate them to $t = \bot$ (while doing so at the sentence level would destroy the representation condition). β and γ can be easily adapted to work also with these new signatures.

Concerning σ-naturality of γ: this always holds if σ is an isomorphism. Hence, bijective renamings are covered. If we want to hide something, with each operation symbol, we also have to hide its result sort: this also implies σ-naturality of γ, but it is of course a rather strong restriction.

If we want to overcome these restrictions, we can use the following:

Theorem 4.13 Each of the institutions shown in Fig. 4 except from $SOL^=$

[14] Since subsorts lead to partial projection functions, we have to exclude them as well. An alternative is to allow subsorts without the possibility to use partial projections.

and *LSL-Bool* allows the re-use of first-order theorem provers with induction (or second-order, or higher-order provers) for both entailments and refinements concerning structured specifications including **free**.

Proof. By chasing the diagram in Fig. 4, one can see that each of the above mentioned institutions has a subinstitution representation in $CFOL^{=}$. The result now follows from Theorem 2.39. □

Note, however, that the sentence translation involved in this result is rather clumsy. Therefore, in practice, one will use the weaker results (Theorems 4.10, 4.11) above in order to get a better sentence translation, and fall back to the clumsier sentence translation only if dealing with specifications involving uses of **free** that are not covered by Theorem 4.11.

Definition 4.14 In any of the institutions introduced so far, a signature Σ is called *strict*, if for each sort, there is a total ground (i.e. variable-free) Σ-term of that sort. A theory is called strict if its signature is. A theory morphism is called strict if its target theory is. □

Fact 4.15 In $PCond^{=}$, every strict theory morphism is liberal, i.e. $PCond^{=}$ is (strict)-liberal.

Proof. For a modification of $PCond^{=}$ with empty carriers in the models allowed, this is quite standard and follows e.g. from the very general results in [73]. The restriction to strict theory morphisms ensures that the free model has non-empty carriers, thus it is always contained in the full subcategory of models without empty carriers. Since free objects in a full subcategory are again free, the result follows. □

Theorem 4.16 Each of the institutions shown in Fig. 5 is (strict)-liberal.

Proof. By chasing the diagram in Fig. 5 and using the remark at the beginning of Section 4.2 about the representation of $PGCond^{=}$ in $PCond^{=}$, one can see that each of the institutions in Fig. 5 can be represented strongly persistently liberally (even as an embedding) in $PCond^{=}$. By Fact 4.15, $PCond^{=}$ is (strict)-liberal, and by Theorem 2.43, this is lifted against strongly persistently liberal institution representations (noting that all representations in Fig. 5 map strict theory morphisms to strict theory morphisms). □

5 Hierarchy Theorems

In this section, at the positive conditional level, some negative results will be proved. This yields a strict hierarchy of institutions and shows that the

different levels introduced in the previous section really are different levels of expressiveness.

Hierarchy theorems are usually proved by finding some property which holds for all objects in some class, but which does not hold for some objects in a more expressive class, which then turns out to be strictly more expressive. Here, we use properties of the model categories of institutions, to show that some model category which is definable in a more expressive institution is not definable in a less expressive institution, showing that the more expressive institution is not a subinstitution of (and even cannot be embedded into) the less expressive institution. In [52], we also use properties of parameterized abstract data types specifiable with **free** and **derive** to separate the same levels of expressiveness.

Both ways lead to a hierarchy of five levels.

Theorem 5.1 The five institutions we consider form a proper hierarchy:

	Institution	Properties of model categories	Separating example
1	$Eq^=$	has effective equivalence relations	
2	$Atom^=$	subobjects commute with coequalizers	graphs = binary relations
3	$GCond^=$	forgetful functors preserve regular epis	some special theory
4	$GHorn^=$	(regular epi, mono)-factorizable	transitive binary relations
5	$SubPGHorn^=$	locally finitely presentable	transitive multigraphs or small categories

The proof is split into four parts, each of which is presented in one of the subsequent subsections. Before coming to that, we quote the following characterization of model categories at level 5:

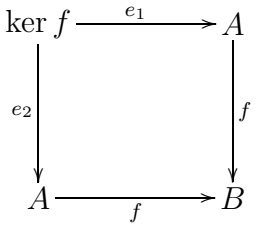
Theorem 5.2 [53]

(1) All model categories specifiable in $SubPGHorn^=$ (or any of its subinstitutions) are locally finitely presentable.
(2) If we extend $SubPGHorn^=$ to infinite signatures (consisting of infinite sets of sorts, infinite sets of (finitary) operation and predicate symbols), then exactly the locally finitely presentable categories can be specified as model categories of theories. □

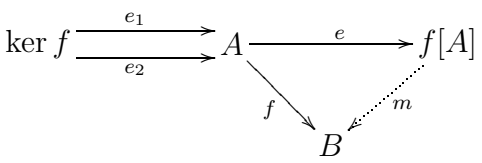
The essential difference between level 4 and level 5 is that in the former, the theorem of homomorphisms holds, while in the latter, it fails. Categorically, the theorem of homomorphisms means that there exist (regular epi, mono)-factorizations.

Proposition 5.3 There is no institution embedding from $PGHorn^{=}$ to $GHorn^{=}$. Thus, $PGHorn^{=}$ (and therefore also $SubPGHorn^{=}$) is strictly more expressive than $GHorn^{=}$.

Proof. For a theory $T = \langle \Sigma, \Gamma \rangle$ in $PGHorn^{=}$ and a model morphism $f\colon A \longrightarrow B$ in $\mathbf{Mod}(T)$, we can try to get a factorization of f through its image by the following procedure: first, take the kernel of f, that is, the pullback



(By Theorem 5.2 and Proposition B.3, $\mathbf{Mod}(T)$ is complete and cocomplete.) Then, take the coequalizer of e_1 and e_2 to get the image $f[A]$:



By the universal property of the coequalizer, there exists a unique $m\colon f[A] \longrightarrow B$ with $f = m \circ e$.

By remarks 3.4(2) and 5.13(2) of [2], if T belongs to $GHorn^{=}$, e is surjective, which implies that m is injective, hence a monomorphism, and (e, m) is a (regular epi,mono)-factorization of f. This is the well-known theorem of homomorphisms, generalized to $GHorn^{=}$.

In contrast to this, in $SubPGHorn^{=}$ and even in $PGHorn^{=}$, e may be not surjective and m no monomorphism, that is, there is no (regular epi,mono)-factorization of f. In [70, 3.4], a counterexample very similar to Example 5.11 is discussed in detail. For factorizations in $PGCond^{=}$ and $PGHorn^{=}$, we have to iterate the construction of kernels and coequalizers (taking e instead of f and so on) possibly infinitely often.

Now (regular epi,mono)-factorizability clearly is preserved by equivalences of categories. So there cannot be an institution representation from $PGHorn^{=}$ to $GHorn^{=}$ with model translation being a pointwise equivalence of categories. □

Corollary 5.4 *$SubPHorn^{=}$ and $PHorn^{=}$ are strictly more expressive than $Horn^{=}$.*

5.2 Level 4 versus Level 3: horn clause logic and conditional equational logic

The difference between level 3 and level 4 is that in the latter, there are quotient homomorphisms which are not closed, while in the former all homomorphisms are closed. Categorically this means that forgetful functors preserve regular epis.

Proposition 5.5 *$GHorn^{=}$ cannot be embedded into $GCond^{=}$. Therefore, $GHorn^{=}$ is strictly more expressive than $GCond^{=}$.*

Proof. A difference between $GHorn^{=}$ and $GCond^{=}$ is that in $GHorn^{=}$, there are quotients which are not closed. This cannot happen in $GCond^{=}$, where all homomorphisms are closed. Consider the theories BinaryRelation and TransitiveClosure from Example 2.11.

Now in TransitiveClosure, if we identify the elements b and c in

$$A = \{\, a \leq b;\ c \leq d \,\}$$

we get the quotient

$$Q = \{\, [a] \leq [b, c] \leq [d] \,\}$$

with $[a] \leq [d]$. The quotient homomorphism $q\colon A \longrightarrow Q$ with $q(x) = [x]$ is not closed: we have $q(a) \leq q(d)$, but not $a \leq d$. On the other hand, in BinaryRelation the quotient $q'\colon A \longrightarrow Q'$ with $Q' = \{\, [a] \leq [b, c] \leq [d] \,\}$ *is* closed, since we do not have $q(a) \leq q(d)$ in Q'.

To be able to use this for the theory of institution representations, we have to switch to a categorical formulation. We observe that q is a regular epimorphism in TransitiveClosure, but not in BinaryRelation (since it does not factor through q'). So the forgetful functor $\mathbf{Mod}\,\sigma$ associated to the inclusion $\sigma\colon$ BinaryRelation $\longrightarrow$ TransitiveClosure does not preserve regular epis.

On the other hand, in $GCond^{=}$, the regular epis are precisely the surjective homomorphisms (this follows from [1], 7.72(1), 23.39 and 24.9). Since all forgetful functors in $GCond^{=}$ preserve surjectivity, they also preserve regular

epis.

Now, preservation of regular epis by reduct functors is a categorical property inherited against embeddings (this follows from the model translations being categorical equivalences and being natural). So $GHorn^{=}$ cannot be embedded into $GCond^{=}$.

Other categorical properties used to separate $GCond^{=}$ from $GHorn^{=}$ use essentially the same intuition. In $GCond^{=}$, all model categories have a dense set of regular projectives, whereas $GHorn^{=}$ lacks this property (see [2, 3.19, 3.21]; see [6] for a similar property). □

5.3 Level 3 versus Level 2: conditional equational logic and partial equational logic

The difference of level 3 and level 2 lies in the presence of conditional axioms at level 3. These imply that the construction of quotients (categorically: coequalizers) is more complex: We cannot take just the algebraic quotient, but have to factor it by new equations which may be derived using the conditional axioms from the elements of the congruence relation. In contrast to this, at level 2, quotients are just algebraic quotients.

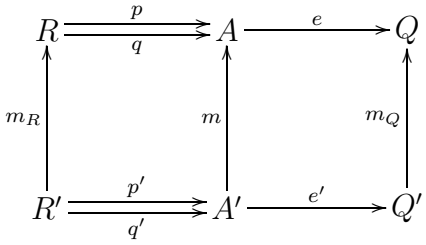
To be able to formulate this categorically, we need to compare two different quotient constructions, one of which is done on a subalgebra. Now at level 2, quotients commute with subalgebras in the following sense: Given a congruence relation R on an algebra A and a subalgebra A' of A, then R induces a congruence relation R' on A'. Then A'/R' is a subalgebra of A/R in a natural way, while at level 3, this is not the case.

Definition 5.6 For a category having coequalizers and pullbacks, we say that *subobjects commute with coequalizers* in this category, if the following holds: Given an equivalence relation $R \overset{p}{\underset{q}{\rightrightarrows}} A$ on an object A and a subobject $m\colon A' \longrightarrow A$ of A (i.e. m is a monomorphism), taking the pullback

$$\begin{array}{ccc} R & \xrightarrow{\langle p,q\rangle} & A \times A \\ {\scriptstyle m_R}\uparrow & & \uparrow{\scriptstyle m\times m} \\ R' & \xrightarrow{\langle p',q'\rangle} & A' \times A' \end{array}$$

and then taking the coequalizers $R \overset{p}{\underset{q}{\rightrightarrows}} A \xrightarrow{e} Q$ and $R' \overset{p'}{\underset{q'}{\rightrightarrows}} A' \xrightarrow{e'} Q'$ has the result that the unique factorization $m_Q\colon Q \longrightarrow Q'$ of $e \circ m$ through e' (which

exists by the universal property of the coequalizer) *is a monomorphism.*

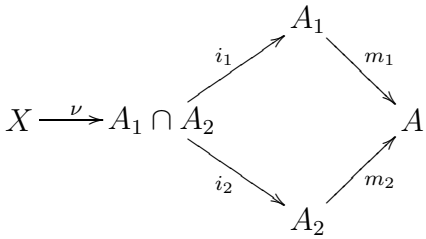


Lemma 5.7 For all theories $T = (\Sigma, \Gamma)$ in $PEq^{=}$, coequalizers in $\mathbf{Mod}(T)$ are just quotients.

Proof. Surjective homomorphisms preserve existence equations by Table 8.1 of [14]. Therefore, the quotient taken in $\mathbf{Mod}(\Sigma)$ satisfies Γ. The coequalizing property now follows from the diagram completion lemma (Lemma 2.7.1 of [14]). $\square$

Proposition 5.8 For all theories $T = (\Sigma, \Gamma)$ in $PEq^{=}$, categorical intersections in $\mathbf{Mod}(T)$ are constructed by taking the relative subalgebra on the intersection of the carriers.

Proof. By 4.2.4 of [14], pullbacks in $\mathbf{Mod}(\Sigma)$ are constructed upon the pullbacks of the underlying sets. In particular, intersections are constructed by taking the relative subalgebra on the intersection of the carriers. Now if $t_1 \overset{e}{=} t_2 \in \Gamma$ and $\nu\colon X \longrightarrow A_1 \cap A_2$ is a valuation into the intersection,



then we know that $(i_1 \circ \nu)^{\#}(t_1)$ and $(i_1 \circ \nu)^{\#}(t_2)$ are defined and equal, and similarly for i_2. Thus, assuming w. l. o. g. that i_1 and i_2 are inclusions, we have that $(i_1 \circ \nu)^{\#}(t_1) = (i_2 \circ \nu)^{\#}(t_1) = \nu^{\#}(t_1)$, and similarly for t_2. Thus $\nu^{\#}(t_1) = \nu^{\#}(t_2)$. $\square$

Proposition 5.9 For all theories $T = \langle \Sigma, \Gamma \rangle$ in $PEq^{=}$, in $\mathbf{Mod}(T)$, subobjects commute with coequalizers.

Proof. Let $R \overset{p}{\underset{q}{\rightrightarrows}} A$ be a relation on A and $m\colon A \longrightarrow A'$ be a subobject of A. By Lemma 5.7, coequalizers are just quotients. Thus m_Q is given by

$$m_Q([a]_{\langle p', q' \rangle}) = [m(a)]_{\langle p, q \rangle}$$

for $a \in A'$ (here, $[a]_\equiv$ is the $\equiv$-congruence class of a). By Proposition 5.8, $R' \overset{p'}{\underset{q'}{\rightrightarrows}} A'$ is $R \overset{p}{\underset{q}{\rightrightarrows}} A$ restricted to A'. Thus, m_Q is injective. $\square$

A counterexample showing that in $GCond^=$, not for all model categories subobjects commute with coequalizers, is the following. Let $T = \langle \Sigma, \Gamma \rangle$ be the presentation

spec T =
  sort s
  ops $f : s \to s$;
    $a, b : s$;
  forall $x : s$
  • $f(x) = x \Rightarrow a = b$
end

Let A be the T-algebra with $A_s = \{1, 2, 3, 4\}$, $a_A = 1$, $b_A = 2$, $f_A(1) = 2$, $f_A(2) = 1$, $f_A(3) = 4$, and $f_A(4) = 3$. Let A' be the subalgebra on the set $\{1, 2\}$. Now let $\equiv$ be the congruence generated by $3 \equiv 4$. Then $A/\equiv$ does not satisfy the T-axioms: in order to do enforce this, we have to identify 1 and 2. This then is the corresponding coequalizer, having carrier set $\{[1, 2], [3, 4]\}$. $\equiv$ restricted to A' is the diagonal relation $\Delta A'$, and $A'/\Delta A' \cong A'$ is the corresponding coequalizer. The unique homomorphism $m_Q : A'/\Delta A' \longrightarrow F_{(\Sigma,\emptyset)\longrightarrow(\Sigma,\Gamma)}(A/\equiv)$ is not injective, since it identifies 1 and 2. $\square$

5.4 Level 2 versus Level 1: partial equational logic and total equational logic

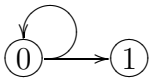
The difference of level 2 and level 1 concerns the role of congruence relations: while at level 1, the algebraic structure of a congruence relation (considered as an algebra) is determined already by the set-theoretical relation, at level 2, there is more flexibility, i.e. there are several algebras with the same congruence relation as carrier, but only one of them is "effectively" generated as a kernel of some homomorphism.

Proposition 5.10 In $Eq^=$, each model category has effective equivalence relations (for the category-theoretic definition of effective equivalence relation, see Definition C.1).

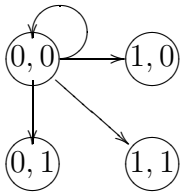
Proof. See Remarks 3.4(8) and 3.6(7) in [2]. $\square$

Example 5.11 Consider the signature `Graph` consisting of one sort symbol and one binary relation symbol. This is clearly a signature in $Atom^=$. In the sequel, we identify the models of `Graph` with graphs. Now **Mod**(`Graph`), the

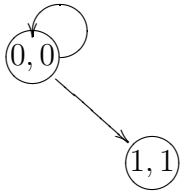
category of graphs, does not have effective equivalence relations: Consider the graph G



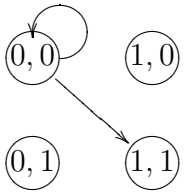
Then $G \times G$ is



and $\Delta G \subseteq G \times G$, the diagonal $\langle id_G, id_G \rangle : G \longrightarrow G \times G$ is



Now consider the relation R on G:

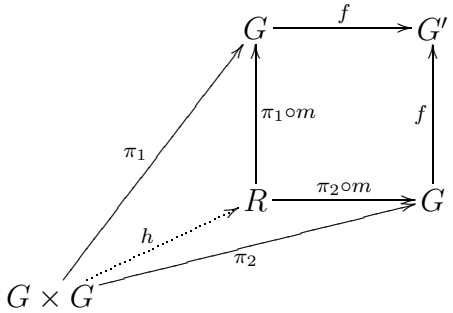


It is given by the inclusion $m : R \longrightarrow G \times G$, and is represented by the pair $\langle \pi_1 \circ m, \pi_2 \circ m \rangle$. Now ΔG is contained in R, so $R \xrightarrow{m} G$ is reflexive.

Define $flip : R \longrightarrow R$ by just flipping $(1, 0)$ and $(0, 1)$. Then $flip$ is an isomorphism from the subobject of G represented by $\langle \pi_1 \circ m, \pi_2 \circ m \rangle$ to the subobject represented by $\langle \pi_2 \circ m, \pi_1 \circ m \rangle$. Thus $R \xrightarrow{m} G$ is symmetric.

Composing $R \xrightarrow{m} G$ with itself by pulling back yields the diagonal relation (as a subobject of $G \times G \times G$, but this plays no role), which is contained in $R \xrightarrow{m} G$ by reflexivity above. Thus $R \xrightarrow{m} G$ is transitive.

Now suppose that for some graph homomorphism $f : G \longrightarrow G'$ the inner square in



is a pullback. Since m is the identity on the underlying sets, the outer diagram commutes as well. By the pullback property, there has to exists a graph homomorphism $h: G \times G \longrightarrow R$ with $\pi_1 \circ m \circ h = \pi_1$ and $\pi_2 \circ m \circ h = \pi_2$. By the uniqueness property of the product, $m \circ h = id$, so h has, like m, to be the identity on the underlying sets. But this violates the graph homomorphism property.

Therefore, $R \xrightarrow{m} G$ is an equivalence relation which cannot be given by a kernel pair. Thus $\mathbf{Mod}(\texttt{Graph})$ does not have effective equivalence relations. □

Corollary 5.12 *$Atom^=$ cannot be embedded into $Eq^=$.*

Proof. Clearly, effective equivalence relations are preserved by equivalences of categories. □

A very similar counterexample can be constructed in the category of unary relations, but we found the above example more instructive. In general, predicates introduce some kind of flexibility: information which is not represented by data elements. This kind of flexibility is present in $Atom^=$, but not in $Eq^=$.

After these more theoretical studies, we now come to the examination of different *specification languages*.

6 The Larch Shared Language

Larch [40] consists of a family of languages supporting a *two-tiered* approach of specification. A specification consists of programming language-independent parts, formulated in the *Larch Shard Language* (LSL), and of programming language-specific parts, formulated in one of the *Larch interface languages*. Since the relation of CASL to programming languages has not been worked out yet, we here can treat the Larch Shared Language only.

6.1 The LSL Institution and Its Translation to $CFOAlg^=$

There is no explicit logic given in the Larch report [39]. Therefore, we here introduce an institution LSL that fits to interpret the LSL constructs, mostly following [8]. Most of LSL will be inherited from $Eq^=$.

Signatures LSL-signatures and signature morphisms are those of $Eq^=$.
Models LSL-models and reducts are those of $Eq^=$.
Sentences Let $\Sigma = (S, TF)$ be an LSL-signature. A Σ-sentence is one of the following:

(1) a Σ-sentence in $Eq^{=}$,
(2) a sentence

$$s \texttt{ generated by } TF' \texttt{wrt } \theta$$

where $\theta\colon \bar{\Sigma} \longrightarrow \Sigma$ is a signature morphism, $\bar{\Sigma} = (\bar{S}, \bar{TF})$, $s \in \bar{S}$ and $TF' \subseteq \bar{TF}$

(3) a sentence

$$s \texttt{ partitioned by } TF'$$

where $s \in S$ and $TF' \subseteq TF$ such for that each function symbol in TF', s occurs exactly once as an argument sort.

Concerning sentence translation,

$$s \texttt{ generated by } TF' \texttt{ wrt } \theta$$

is translated along a signature morphism σ to

$$s \texttt{ generated by } TF' \texttt{ wrt } \sigma \circ \theta,$$

and

$$s \texttt{ partitioned by } TF'$$

is translated to

$$\sigma^S(s) \texttt{ partitioned by } \sigma(TF'),$$

where

$$\sigma(TF')_{w',s'} := \bigcup_{w \in S^*, s \in S, \sigma^{S^*}(w)=w', \sigma^S(s)=s'} \sigma^F_{w,s}({TF'}_{w,s}).$$

Satisfaction Satisfaction of an equation from $Eq^{=}$ is defined as in $Eq^{=}$. Satisfaction of

$$s \texttt{ generated by } TF' \texttt{ wrt } \theta$$

is the same as satisfaction of $(\{s\}, TF', \theta)$ in $CEq^{=}$. Finally, satisfaction of

$$s \texttt{ partitioned by } TF'$$

is the same as satisfaction of

$\forall x_1, x_2 : s \bullet$
$(\bigwedge_{i=1,\ldots,n} \forall y_1 : s^i_1; \ldots; y_{m_i-1} : s^i_{m_i-1}, s, y_{m_i+1} : s^i_{m_i+1}, \ldots, y_{n_i} : s^i_{n_i} \bullet$
$f_i(y_1, \ldots, y_{m_i-1}, x_1, y_{m_i+1}, \ldots, y_{n_i}) = f_i(y_1, \ldots, y_{m_i-1}, x_2, y_{m_i+1}, \ldots, y_{n_i}))$
$\Rightarrow x_1 = x_2$

where $TF' =$
$\{op_1\colon s^1_1, \ldots, s^1_{m_1-1}, s, s^1_{m_1+1}, \ldots, s^1_{n_1} \longrightarrow s^1,$
$\ldots,$
$op_k\colon s^k_1, \ldots, s^k_{m_k-1}, s, s^k_{m_k+1}, \ldots, s^k_{n_k} \longrightarrow s^k\}$ $\square$

LSL has no logical connectives in the standard sense, but rather a sort *Boolean* with the usual connectives is built-in, and one can use equations over *Boolean* to simulate the logical connectives. We do not introduce this built-in *Boolean* into the institution, since we think that this is better dealt with at the level of

structured specification: One can think of an LSL-specification *Boolean* that is automatically included into each user-defined specification.

There is an obvious institution representation from LSL to $CFOAlg^{=}$:

Signatures Signatures are translated identically.
Models Models are translated identically as well.
Sentences Equations are translated identically. Sentences of form

$$s \texttt{ generated by } TF' \texttt{wrt } \theta$$

or

$$s \texttt{ partitioned by } TF'$$

are translated into the corresponding $CFOAlg^{=}$-sentences that were used above for the definition of their satisfaction.
Satisfaction The representation condition is obvious. □

6.2 Translating LSL Language Constructs Into CASL *Constructs*

Overloading, disambiguation of terms and operation symbols, and mixfix operation symbols are treated in the same way in both LSL and CASL. Thus, there should be no difficulty with the translation. LSL's built-in precedences can be specified using CASL's precedence annotations (note, however, that it is necessary for if__then__else__ to override the standard CASL precedence scheme between infix and prefix operations).

Enumeration types are expanded to LSL-theories in the Larch report. However, when translating LSL to CASL, it would be more advisable to translate them directly for CASL's free types. That is,

$$s \texttt{ enumeration of } c_1 \ldots c_n$$

is translated to

free type
$s ::= c_1 \mid \ldots \mid c_n;$
op $succ : s \to s;$
axioms $succ(c_1) = c_2;$
$\ldots$
$succ(c_{n-1}) = c_n;$

An LSL union type like

$$s \texttt{ union of } f_1 : s_1 \ldots f_n : s_n$$

is translated to

generated type
$s \ ::= \ f_1(\_\_.f_1 : s_1) \ | \ \ldots \ | \ f_n(\_\_.f_n : s_n);$

Both of the above translations directly correspond to the LSL expansion scheme. However, note that both the selectors $\_\_.f_i$ as well as the successor operation *succ* are total, which generally leads to unwanted error elements. If one does not want to have this rather strange behaviour of LSL, one also could use partial operations here. This would lead to the translations

free type
$s \ ::= \ c_1 \ | \ \ldots \ | \ c_n;$
op $succ \ : \ s \rightarrow ? \ s;$
axioms $succ(c_1) \ = \ c_2;$
$\ldots$
$succ(c_{n-1}) \ = \ c_n;$
$\neg def \, succ(c_n);$

and

generated type
$s \ ::= \ f_1(\_\_.f_1 :? s_1) \ | \ \ldots \ | \ f_n(\_\_.f_n :? s_n);$

This would, however, require to extend the target institution to $PCFOAlg^{=}$.

7 ACT ONE

ACT ONE originally has been based on equational logic [28] and later was extended to conditional equational logic [20]. Thus, the underlying institution is $Cond^{=}$, modulo the problem of empty carriers, which is treated in Section 11.

The ACT ONE languages constructs can be translated very easily to CASL. Overloading in ACT ONE is a special case of overloading in CASL. ACT ONE's `let` definitions can be translated to CASL's operation definitions. Note that strictly speaking, one has to hide the locally defined operation afterwards; this would require an in-the-large construct of CASL.

ACT ONE distinguishes *constructor* and *function* declarations. Both lead to declarations of operations in CASL. However, the constructors additionally lead to special axioms concerning the equality predicate *eq*. These axioms also have to be included in the CASL translation, of course. An alternative way of dealing with constructors is described in Section 11.1.

8 ASF

The language ASF [11] is based on conditional equational logic. Thus, again the underlying institution is $Cond^{=}$, modulo the problem of empty carriers, which is treated in Section 11. ASF has an initial semantics for the modules. Initial semantics can be chosen in CASL by writing a **free { ... }** around a specification.

Concerning the language constructs of ASF, the prefix and infix syntax is a special case of CASL's mixfix syntax. Now there is a syntax definition formalism (called SDF) built on top of ASF, which is much more flexible. SDF definitions of context free syntax can be translated to mixfix syntax declarations in CASL. SDF's priorities and associativity attributes are available as annotations in CASL as well (in fact, the CASL design followed the SDF design at this point). Concerning lexical syntax, CASL is not as flexible as SDF: it is not possible to redefine the lexical syntax in CASL: one has to use the fixed built-in lexical syntax. However, CASL provides a means to define special literal syntaxes for numbers, strings and list-like structures. This should already cover many applications of redefinition of lexical syntax in SDF.

ASF allows product types as result types of functions and provides a tupling notation for terms. In [11], a translation of this into extra product sorts and explicit tupling functions is described; this translation has to be applied when translating ASF to CASL.

Overloading in ASF is a special case of overloading in CASL. ASF's `if` function can be translated to CASL's __ when __ else __ construct (indeed, the expansion of these two constructs to formulas of the underlying institution is essentially the same in ASF and in CASL).

A further informal comparison of CASL and ASF can be found in [64].

9 HEP Theories

HEP theories (for Hierarchical Equationally Partial Theories) have been introduced by Reichel [70]. The idea was to introduce the domains of partial functions in a structured way.

Signatures HEP-signatures are of form $\Sigma = (S, TF, PF, \preceq, Def)$ such that
- (S, TF, PF) is a signature in $PCond^{=}$,
- $\preceq$ is a well-founded partial order on PF, and
- Def is a mapping that assigns to each $f\colon s_1, \ldots, s_n \longrightarrow ? s \in PF$ a finite set

of existence equations over the variables $x_1 : s_1, \ldots, x_n : s_n$, such that all partial function symbols occurring in $Def(f)$ are strictly less than f w.r.t. $\preceq$.

Signature morphisms are those from $PCond^{=}$ with the additional requirement that

$$Def(\sigma(f)) = \sigma(Def(f))$$

Models Σ-models are (S, TF, PF)-models A in $PCond^{=}$ such that dom f_A equals

$$\{(\nu(x_1), \ldots, \nu(x_n)) \in A_w \mid \nu: X \longrightarrow A \text{ with } \nu \Vdash t_1 \stackrel{e}{=} t_2 \text{ for all } t_1 \stackrel{e}{=} t_2 \in Def(f)\}$$

for each $f: w \longrightarrow ?s \in PF$, $w = s_1 \ldots s_n$.

Reducts are defined as in $PCond^{=}$.

Sentences Σ-sentences are (S, TF, PF)-sentences in $PCond^{=}$.

Satisfaction Satisfaction is defined as in $PCond^{=}$. □

Note that Reichel's theory morphisms defined in [70] are slightly more general than theory morphisms in our HEP. However, Reichel does not define an institution. Indeed, from the definitions in [70], one can only extract a specification frame (i.e., a category of theories **Th** and a contravariant model functor $\mathbf{Mod}: \mathbf{Th}^{op} \longrightarrow \mathcal{CAT}$ into the quasicategory of categories). If signatures and sentences have to be separated, our slight restriction has to be made.

Now there is an easy institution representation from HEP to $PCond^{=}$:

Signatures A HEP-signature $\Sigma = (S, TF, PF, \preceq, Def)$ is sent to the $PCond^{=}$-presentation consisting of the signature (S, TF, PF) and of an axiom

$$\forall x_1 : s_1, \ldots, x_n : s_n \bullet def\ f(x_1, \ldots, x_n) \Leftrightarrow \bigwedge_{t_1 \stackrel{e}{=} t_2 \in Def(f)} t_1 \stackrel{e}{=} t_2$$

for each partial function symbol $f: s_1, \ldots, s_n \longrightarrow ?s \in PF$.

Models The model translation is just the identity.

Sentences The sentence translation is the identity as well.

Satisfaction The representation condition is obvious. □

In [53], we also have set up a so-called *weak* institution representation in the other direction, i.e. from $PCond^{=}$ to HEP. It uses an encoding of partial function as sorts and of axioms as partial functions. This leads to rather complex translated theories. We therefore do not go into the details here.

Reichel also defines *canons* that can be used to restrict subtheories of a given theory to an initial interpretation. This can be easily translated to the **free** $\{\ldots\}$ construct in CASL. However, we do not show this translation here: Since the **free** $\{\ldots\}$ construct belongs to the level of specification in-the-large, this is beyond the scope of this paper.

10 OBJ3 and functional CafeOBJ

OBJ3 [31] is a specification and high-level programming language supporting subsorting. The logic is similar to $SubCond^{=}$, but has some differences. Apart from the empty carrier problem, OBJ has a different way of overload resolution. Furthermore, casts from a supersort to a subsort, which are partial functions in CASL, are total functions called retracts in OBJ. Moreover, membership formulas are not present in OBJ, though there are attempts to extend OBJ with sort constraints [77], which are nothing but CASL's membership formulas (do not confuse them with CASL's sort *generation* constraints).

CafeOBJ [26] is a successor of OBJ3 extending OBJ3 with hidden algebra (for modeling object orientation) and rewriting logic (for modeling concurrency). Now the comparison of these features with some corresponding (object-oriented resp. concurrent) extension of CASL is beyond the scope of this paper. We therefore concentrate on the functional part of CafeOBJ, called fun-CafeOBJ to be short. With respect to subsorting, the CafeOBJ report [26] leaves open the details, and states only some very general constraints on the logic. However, it is expected [25] that the subsorting in fun-CafeOBJ will be some kind of Meseguer's Membership Equational Logic $MEqtl$ [49] (it seems that also sorts in fun-CafeOBJ always have error supersorts; but this is more a question of language constructs). Since $MEqtl$ already has been treated in Section 4.2, we here concentrate on OBJ3. See [66] for further comparison of CASL and CafeOBJ (also at the in-the-large level) and more examples.

10.1 The Institution of Order-Sorted Algebra

The OBJ3 manual [31] claims that OBJ3 is based on order-sorted algebra as developed in [36]. We here describe the institution $COSASC$ of coherent order-sorted algebra enriched with sort constraints, as introduced by Goguen and Meseguer [36,50,30], in some detail, since it appears that there is no complete detailed description of it in the literature [15].

Definition 10.1 The institution $COSASC$.

Signatures Order signatures are triples $\Sigma = (S, TF, \leq)$, where $\leq$ is a partial order on S and (S, TF) is a many-sorted signature, such that the following conditions are satisfied:

- $f \in TF_{w,s} \cap TF_{w',s'}$ and $w \leq w'$ imply $s \leq s'$ (*signature monotonicity*)

[15] Han Yan [77] comes closest to describing such an institution, but [77] is not publicly available.

- for each $f \in TF_{w,s}$ and $w_0 \leq w \in S^n$, there is a least arity for f that is greater than or equal to w_0 (*regularity*). This condition implies that there also is a least *rank* (i.e. arity and coarity) for f with arity greater than or equal to w_0, see [36].
- each connected component of $(S, \leq)$ is filtered, i. e. any two elements (of the connected component) have a common upper bound (*local filtration*). (The conjunction of regularity and local filtration is called *coherence*.)

We further assume that each signature is *strongly locally filtered* [57], which means that any two elements of a connected component have a *least* upper bound. This assumption is needed for setting up the institution representations, but it might be omitted when translating the language constructs.

Given two signatures $\Sigma = (S, TF, \leq)$ and $\Sigma' = (S', TF', \leq')$, an OSA-signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$ consists of

- a map $\sigma^S\colon S \longrightarrow S'$,
- a family of maps $\sigma^F_{w,s}\colon TF_{w,s} \longrightarrow TF'_{\sigma^S(w),\sigma^S(s)}$ for each $w \in S^*$, $s \in S$

such that

- $\sigma^S(s) \leq' \sigma^S(s')$ whenever $s \leq s'$ (*morphism monotonicity*),
- for $w \leq w'$, $f \in TF_{w,s} \cap TF_{w',s'}$, we have $\sigma^F_{w,s}(f) = \sigma^F_{w',s'}(f)$ (*preservation of overloading*) [16],
- for each $f \in TF_{w,s}$ and $w_0 \leq w$, $(\sigma^S)^n$ applied to the least arity for f which is $\geq w_0$ yields the least arity for $\sigma^F_{w,s}(f)$ which is $\geq (\sigma^S)^n(w_0)$ (*preservation of regularity*). [17]

We further assume that least upper bounds of connected sorts are preserved by signature morphisms.

Models Given $\Sigma = (S, TF, \leq)$, an order-sorted Σ-algebra is a many sorted (S, TF)-algebra M such that

(1) $s \leq s'$ implies $M_s \subseteq M_{s'}$ (*subsort inclusion*),
(2) $f \in TF_{w,s} \cap TF_{w',s'}$ and $w \leq w'$ imply $f_M\colon M_w \longrightarrow M_s$ equals $f_M\colon M_{w'} \longrightarrow M_{s'}$ on M_w (*algebra monotonicity*).

Homomorphisms $h\colon M \longrightarrow N$ are many-sorted homomorphisms $h\colon M \longrightarrow N$ such that $s \leq s'$ and $a \in M_s$ imply $h_s(a) = h_{s'}(a)$. Reducts are defined as for many-sorted algebras. Since many-sorted reducts preserve inclusions and signature morphisms preserve overloading, the resulting many-sorted algebra is again an order-sorted algebra.

Sentences An *variable system* X *over* $\Sigma = (S, TF, \leq)$ is a family of sets $X = (X_s)_{s \in S}$ with the X_s pairwise disjoint. Given such a variable system, the set $T_\Sigma(X)_s$ of $\Sigma(X)$*-terms of sort* s is inductively defined:

(1) $x \in T_\Sigma(X)_s$ for $x \in X_s$,
(2) $f(t_1, \ldots, t_n) \in T_\Sigma(X)_s$ for $f\colon s_1, \ldots, s_n \longrightarrow s \in TF$ and $t_i \in T_\Sigma(X)_{s_i}$,

[16] This condition is needed to ensure that reduct functors preserve monotonicity of models (cf. Lemma 3.5, [41]). In [77], this condition is missing, while the condition of [32] is too weak.

[17] This condition is needed to ensure that the least sort parse algorithm described below is compatible with signature morphisms.

$i = 1, \ldots, n$,

(3) $T_\Sigma(X)_s \subseteq T_\Sigma(X)_{s'}$ for $s \leq s'$.

It is straightforward to make $T_\Sigma(X)$ into an order-sorted Σ-algebra. Moreover we have the following Lemma from [36], holding for regular signatures.

Lemma 10.2 Each valuation $\nu: X \longrightarrow M$ has a unique homomorphic extension $\nu^\#: T_\Sigma(X) \longrightarrow M$.

Proof. Given a variable valuation (i.e. an S-sorted map) $\nu: X \longrightarrow M$ into a Σ-algebra M, the homomorphism $\nu^\#: T_\Sigma(X) \longrightarrow M$ is defined by

- $\nu^\#_s(x) = \nu_s(x)$ for $x \in X_s$, $s \in S$,
- $\nu^\#_s(f(t_1, \ldots, t_n)) = f_M(\nu^\#_{s_1}(t_1), \ldots, \nu^\#_{s_n}(t_n))$ for $f \in TF_{s_1 \ldots s_n, s}, t_i \in T_\Sigma(X)_{s_i}$,
- $\nu^\#_s(t) = \nu^\#_{s'}(t)$ for $s' \leq s$, $t \in T_\Sigma(X)_{s'}$.

By regularity, this is well-defined, and moreover, $\nu^\#$ is the unique homomorphism from $T_\Sigma(X)$ to M extending ν. By the last equation, we can omit the sort index and write just $\nu^\#(t)$ instead of $\nu^\#_s(t)$ whenever $t \in T_\Sigma(X)$. $\square$

A Σ-sentence in OSA is a conditional formula

$$\forall X \bullet e_1 \wedge \ldots \wedge e_n \Rightarrow e$$

where X is a variable system over Σ and the atomic formulas e and e_i are of two kinds: Either equations

$$t_1 = t_2$$

where $t_1 \in T_\Sigma(X)_{s_1}$, $t_2 \in T_\Sigma(X)_{s_2}$ and s_1, s_2 are in the same connected component of $(S, \leq)$, or sort constraints

$$t : s$$

where $t \in T_\Sigma(X)_{s'}$ and s is a sort in the same connected component as s'.

Given a signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$ and a variable system X over Σ, we can get a variable system $\sigma(X)$ over Σ' by putting

$$\sigma(X)_{s'} = \bigcup_{\sigma^S(s) = s'} X_s$$

The inclusion $\zeta_{\sigma,X}: X \longrightarrow T_{\Sigma'}(\sigma(X))|_\sigma$ has a unique homomorphic extension $\zeta^\#_{\sigma,X}: T_\Sigma(X) \longrightarrow T_{\Sigma'}(\sigma(X))|_\sigma$. Now a Σ-atomic formula $t_1 = t_2$ is translated to $\zeta^\#_{\sigma,X}(t_1) = \zeta^\#_{\sigma,X}(t_2)$, and a Σ-atomic formula $t : s$ is translated to $\zeta^\#_{\sigma,X}(t_1) : \sigma^S(s)$. This easily extends to conditional formulas.

Satisfaction A Σ-algebra M *satisfies* a conditional axiom, if all valuations which satisfy the premises also satisfy the conclusion. A valuation $\nu: X \longrightarrow M$ satisfies an equation $t_1 = t_2$ if $\nu^\#(t_1) = \nu^\#(t_2)$, and it satisfies a sort constraint $t : s$ if $\nu^\#(t) \in M_s$. The satisfaction condition can be proved by noting that (similarly as in Lemma 3.2) valuations $\nu: \sigma(X) \longrightarrow M$ are in a one-one-correspondence to valuations $\nu|_\sigma: X \longrightarrow M|_\sigma$ with the property that $\zeta^\#_{\sigma,X} \circ \nu^\#|_\sigma = (\nu|_\sigma)^\#$. The satisfaction condition for sort constraints is proved in [77]. $\square$

We will also need the *reduction theorem* from [36], which translates order-sorted to many-sorted algebra.

In $COSASC$, terms are not disambiguated; they can have many different sorts. In contrast to this, terms in $SubPFOL^{=}$ are always fully qualified.

Recall from Section 3.2 that the many-sorted signature $\hat{\Sigma}$ is obtained from a subsorted signature Σ by adding injection, projection and membership symbols. Following Goguen and Meseguer in [36], let $\Sigma^{\#}$ be the subsignature of $\hat{\Sigma}$ without projection and membership symbols, and let $J(\Sigma)$ be the subset of $\hat{J}(\Sigma)$ consisting of the identity, embedding-injectivity and transitivity axioms and of those function monotonicity axioms for which $w = w' \leq w''$ in the condition. Further, let $\mu_\Sigma: \Sigma^{\#} \longrightarrow \hat{\Sigma}$ be the inclusion. We will write J for $J(\Sigma)$ and $\hat{J}$ for $\hat{J}(\Sigma)$ if Σ is clear from the context.

Theorem 10.3 A reduction from order-sorted to many-sorted algebra, can be formalized as a simple institution representation from $COSA$ (coherent order-sorted algebra as introduced above, but *without* sort constraints) to $Cond^{=}$ (many-sorted algebra), as follows.

Signatures A $COSA$-signature Σ is mapped to the $Cond^{=}$-presentation $\langle \Sigma^{\#}, J \rangle$. A signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$ in $COSA$ is mapped to its obvious extension $\sigma^{\#}: \Sigma^{\#} \longrightarrow \Sigma'^{\#}$.

Models By Theorem 4.2 of [36], there is an equivalence of categories $(\_)^{\bullet}$ between $\mathbf{Mod}^{Cond^{=}}\langle \Sigma^{\#}, J \rangle$ and $\mathbf{Mod}^{COSA}(\Sigma)$. Basically, given a $\langle \Sigma^{\#}, J \rangle$-model N, the subsort injections are replaced by set inclusions in $N^{\bullet}$ using colimits of filtered diagrams built up by the injections. Vice versa, any Σ-algebra M can be mapped to a $\langle \Sigma^{\#}, J \rangle$-algebra $M^{\#}$ by just taking the inclusions as injections. Since viewing the inclusions as injections and then going back to the injections is the identity, the representation is strongly persistently liberal. Since we have an equivalence of categories, it is also an embedding.

Sentences Each $COSA$-term $t \in T_\Sigma(X)$ has a *least sort* $LS_\Sigma(t)$ [36, 2.10] and a *least sort parse* $LP_\Sigma(t) \in T_{\Sigma^{\#}}(X)$ [36, p. 252]. They are defined inductively as follows: If $t = x$ is a variable, it has a unique sort $LS_\Sigma(x)$ due to the disjointness condition for variable systems, and $LP_\Sigma(x) = x$. If $t = f(t_1, \ldots, t_s) \in T_\Sigma(X)_s$, then by induction hypothesis, t_i has least sort s_i. Take $w_0 = s_1 \ldots s_n$. Then by the term formation rules, $f \in TF_{w',s'}$ for some w', s' with $s' \leq s$ and $w_0 \leq w'$. By regularity, there is a least rank w', s' for f such that $w' \geq w_0$. This least s' is the desired least sort of t, while $\mathtt{inj}_{(s',s)}(f_{w',s'}(\mathtt{inj}_{(s_1,s'_1)}(LP_\Sigma(t_1)), \ldots, \mathtt{inj}_{(s_n,s'_n)}(LP_\Sigma(t_n))))$ is the least sort parse of t.

Given an equation $t_1 = t_2$, let $s_1 = LS(t_1)$, $s_2 = LS(t_2)$ and $s = lub(s_1, s_2)$, the least upper bound of s_1 and s_2 which exists by strong lo-

cal filtration. Then $LP_\Sigma(t_1 = t_2)$ is

$$\mathtt{inj}_{(s_1,s)}(LP_\Sigma(t_1)) = \mathtt{inj}_{(s_2,s)}(LP_\Sigma(t_2))$$

This can easily be extended to conditional equations, giving translations $LP_\Sigma\colon \mathbf{Sen}^{COSA}(\Sigma) \longrightarrow \mathbf{Sen}^{Cond^=}(\Sigma^\#)$. Since signature morphisms preserve regularity and least upper bounds, the construction is natural in Σ.

Satisfaction The representation condition follows from Theorem 4.4 of [36]. □

We also have a representation in the converse direction:

Theorem 10.4 Let *COS-SubCond*$^=$ be the restriction of *SubCond*$^=$ to coherent signatures. There is an embedding institution representation from *COS-SubCond*$^=$ to *COSASC*.

Proof. The representation works follows:

Signatures A coherent CASL signature (morphism) obviously is a *COSASC*-signature (morphism).

Models Since the membership predicates in $\hat{\Sigma}$ are fully determined by the axioms in $\hat{J}(\Sigma)$, we can think of CASL Σ-models as $\langle\Sigma^\#, J\rangle$-models. The model translation now works as follows: Just take inclusions in *COSASC*-models to be the injections in *COS-SubCond*$^=$. Obviously, this is not bijective. However, it is a (pointwise) equivalence of categories, as shown in Theorem 10.3.

Sentences Just delete the injections. Applications of membership are translated into sort constraints.

Satisfaction As in Theorem 10.3.

□

The institution *COSASC* is not yet the right institution to give a precise meaning to all the constructs of OBJ3. Most importantly, retracts are missing. The OBJ3 system automatically extends an *OSA*-presentation $\langle\Sigma, \Gamma\rangle$ with $\Sigma = (S, TF, \leq)$ to an *OSA*-presentation $\langle\Sigma^\otimes, \Gamma^\otimes\rangle$ which is $\langle\Sigma, \Gamma\rangle$ extended by functions

$$(r : s' > s)\colon s' \longrightarrow s$$

and axioms

$$\forall x : s \bullet (r : s' > s)(x) = x$$

for s, s' in the same connected component of $(S, \leq)$. Let $\kappa_{\Sigma,\Gamma}\colon \langle\Sigma, \Gamma\rangle \longrightarrow \langle\Sigma^\otimes, \Gamma^\otimes\rangle$ be the inclusion, and $\kappa_\Sigma = \kappa_{\Sigma,\emptyset}$. Signature morphisms $\sigma\colon \Sigma \longrightarrow \Sigma$ can also be extended to signature morphisms $\sigma^\otimes\colon \Sigma^\otimes \longrightarrow \Sigma'^\otimes$. The following lemma is important for the semantics of retracts:

Lemma 10.5 Let $F_{\kappa_{\Sigma,\Gamma}}\colon \mathbf{Mod}(\langle\Sigma, \Gamma\rangle) \longrightarrow \mathbf{Mod}(\langle\Sigma^\otimes, \Gamma^\otimes\rangle)$ be the free construction (left adjoint) w.r.t. $\mathbf{Mod}(\kappa_{\Sigma,\Gamma})$. Then the unit $\iota_M\colon M \longrightarrow F_{\kappa_{\Sigma,\Gamma}}(M)|_{\kappa_{\Sigma,\Gamma}}$

is injective.

Proof. The lemma follows from Theorem 3.5 of [36] [18] by using the diagram method for free extensions [73]. □

10.2 The Institutions Underlying OBJ3

Joseph Goguen claims in [32] that the semantics of a theory $\langle \Sigma, \Gamma \rangle$ is neither $\mathbf{Mod}(\langle \Sigma, \Gamma \rangle)$ nor $\mathbf{Mod}(\langle \Sigma^{\otimes}, \Gamma^{\otimes} \rangle)$, but rather $\iota \colon Id_{\mathbf{Mod}(\langle \Sigma, \Gamma \rangle)} \longrightarrow \mathbf{Mod}(\kappa_{\Sigma,\Gamma}) \circ F_{\kappa_{\Sigma,\Gamma}}$ (actually, he considers initial semantics only, and the above is the straightforward generalization to loose semantics). But there is no explicit description of an institution that uses ι as a semantics in the literature. Perhaps the OBJ community does not feel a need of such an institution. But when translating OBJ3 to other languages, we need a precise semantics of all the underlying concepts. Therefore, in this section, we introduce two institutions that can serve as underlying institutions of OBJ3.

The following institution captures OBJ3's retracts and also allows to deal with error recovery by treating error values as first-class citizens.

Definition 10.6 The institution $COSASC^{\otimes}$.

Signatures A $COSASC^{\otimes}$-signature (resp. signature morphism) is just a signature (resp. signature morphism) in $COSASC$.

Models A Σ-algebra in $COSASC^{\otimes}$ is a $\langle \Sigma^{\otimes}, \emptyset^{\otimes} \rangle$-algebra in $COSASC$. Much in the same way, reducts are inherited.

Sentences A Σ-sentence in $COSASC^{\otimes}$ is a $\Sigma^{\otimes}$-sentence in $COSASC$. Sentence translation is inherited in the same way.

Satisfaction A Σ-algebra M satisfies a Σ-sentence φ in $COSASC^{\otimes}$ if we have in $COSASC$ that the $\Sigma^{\otimes}$-algebra M satisfies the $\Sigma^{\otimes}$-sentence φ. Thus the satisfaction condition for $COSASC^{\otimes}$ follows from that for $COSASC$. □

Treating error values as first-class citizens in $COSASC^{\otimes}$ has one main disadvantage: equations involving retracts may lead to confusion among ordinary values, which generally is undesirable. So one has to prove that retract equations do not interfere with ordinary values. More importantly, in $COSASC^{\otimes}$ we entirely lose the distinction between ordinary and error values, since ordinary functions and retracts can both map ordinary values to error values and vice versa. Also CafeOBJ's error supersorts do not help for separating retract-generated error values from ordinary values. This is because retract functions

[18] Notice that here we need the assumption that all carriers are non-empty, which implies faithfulness of $\langle \Sigma, \Gamma \rangle$ in the sense of [36]. This will be further discussed in Section 11.2.

have to go to ordinary sorts, since they serve for the purpose of coercing a value to an (ordinary) subsort of a sort.

This justifies the introduction of a more complex institution that keeps error values and ordinary values distinct. It also comes closer to Joseph Goguen's claim that the injection ι *is* the semantics, meaning that "the 'new error elements' reside in a shadowy penumbra, distinctly set off from the bright central region of pure elements by the injection, as well as from the dark outer region of error terms by a subsort relation" [32]. The best approximation of this "shadowy penumbra" seems to be the institution $COSASC_{\otimes}$ introduced below. In the presence of CafeOBJ's error supersorts, satisfaction in this institution is closely related to existential super satisfaction in [32].

Definition 10.7 The institution $COSASC_{\otimes}$.

Signatures A $COSASC_{\otimes}$-signature (resp. signature morphism) is just a signature (resp. signature morphism) in $COSASC$.
Models A Σ-algebra in $COSASC_{\otimes}$ is a Σ-algebra in $COSASC$.
Sentences A Σ-sentence in $COSASC_{\otimes}$ is a $\Sigma^{\otimes}$-sentence in $COSASC$.
Satisfaction A valuation $\nu: X \longrightarrow M$ satisfies a Σ-equation $t_1 = t_2$ with $s = lub(LS(t_1), LS(t_2))$ if

$$(\iota_M \circ \nu)^{\#}(t_1) = (\iota_M \circ \nu)^{\#}(t_2) \in \iota_M(M_s)$$

where $\iota_M \circ \nu$ is considered to be a valuation from X to $F_{\kappa_\Sigma}(M)$. A valuation $\nu: X \longrightarrow M$ satisfies a Σ-sort constraint $t : s$ if ,

$$(\iota_M \circ \nu)^{\#}(t) \in \iota_M(M_s)$$

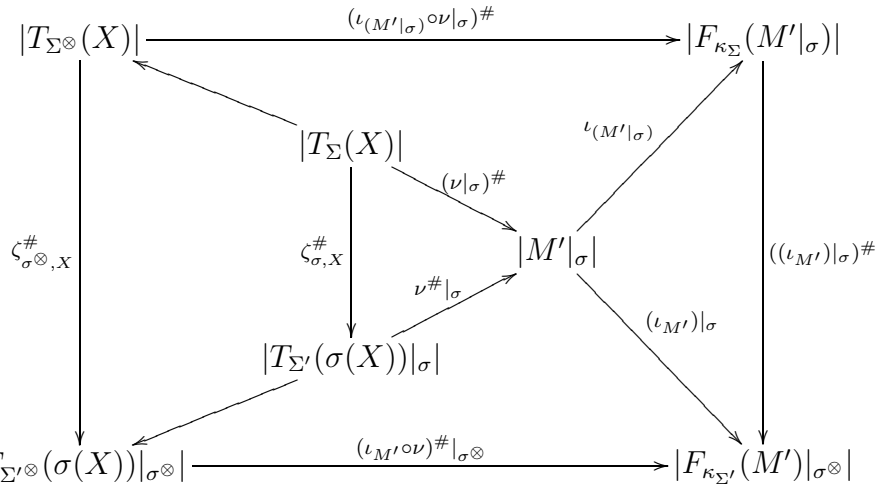
This is extended to sentences in the same way as for $COSASC$.

In order to prove the satisfaction condition, consider a signature morphism $\sigma: \Sigma \longrightarrow \Sigma'$ and a Σ'-algebra M'. By the universal property of the free extension $F_{\kappa_\Sigma}(M'|_\sigma)$, the homomorphism $(\iota_{M'})|_\sigma$ has a unique extension

$$(\iota_{M'})|_\sigma{}^{\#}: F_{\kappa_\Sigma}(M'|_\sigma) \longrightarrow F_{\kappa_{\Sigma'}}(M')|_{\sigma^{\otimes}}$$

making the right triangle in the diagram below commutative. Now we use the fact stated in the satisfaction part of Definition 10.1 that valuations $\nu: \sigma(X) \longrightarrow M'$ are in a one-one-correspondence to valuations $\nu|_\sigma: X \longrightarrow M'|_\sigma$ with the property that $\zeta^{\#}_{\sigma,X} \circ \nu^{\#}|_\sigma = (\nu|_\sigma)^{\#}$. The remaining subdiagrams in the following diagram commute by freeness of the respective term

algebras; thus, the outer square commutes as well.



Since $((\iota_{M'})|_\sigma)^\#|_{\kappa_\sigma} \circ \iota_{(M'|_\sigma)} = (\iota_{M'})|_\sigma$, $((\iota_{M'})|_\sigma)^\#$ takes the image of $\iota_{(M'|_\sigma)}$ exactly to the image of $(\iota_{M'})|_\sigma$. Moreover, on the former image, $((\iota_{M'})|_\sigma)^\#$ is injective, since $((\iota_{M'})|_\sigma)^\#|_{\kappa_\sigma} \circ \iota_{(M'|_\sigma)} = (\iota_{M'})|_\sigma$ is injective by Lemma 10.5. Therefore, there exists some $a \in (M'|_\sigma)_s$ with

$$(\iota_{(M'|_\sigma)} \circ \nu|_\sigma)^\#(t_1) = \iota_{(M'|_\sigma)}(a) = (\iota_{(M'|_\sigma)} \circ \nu|_\sigma)^\#(t_2)$$

if and only if there exists some $a \in {M'}_{\sigma^S(s)}$ with

$$(\iota_{M'} \circ \sigma(\nu))^\#(\zeta^\#_{\sigma^\otimes,X}(t_1)) = \iota_{M'}(a) = (\iota_{M'} \circ \sigma(\nu))^\#(\zeta^\#_{\sigma^\otimes,X}(t_2))$$

Since $\zeta^\#_{\sigma^\otimes,X}$ is used for sentence translation, the satisfaction condition follows easily. □

In $COSASC_\otimes$, error values generated by retracts only have a virtual existence, since retracts are always interpreted freely. This means that equations involving retracts are false in all models if they do not follow already from the built-in retract equations in $\Gamma^\otimes$. In the case that all user-defined retract equations involve only one retract, and this retract occurs at the outmost position of a term, this works fine, and $COSASC_\otimes$ corresponds to the operational semantics of OBJ3.

However, $COSASC_\otimes$ does not work for error recovery, where error values should be treated as first-class values (such that functions can compute with them). In this case, we have to use $COSASC^\otimes$, though we then lose the distinction between ordinary and error values.

Let us illustrate this with an example. Consider the following OBJ3 specification of paths over an arbitrary graph, which later on is instantiated to a particular graph with four nodes and three edges (the specification is also a

CafeOBJ specification, the only difference being that CafeOBJ generates the supersort Path_Err automatically):

```
th GRAPH is
  sorts Nodes Edges .
  ops source target : Edges -> Nodes .
endo

obj PATH[G :: GRAPH] is
  sorts Edges < Path .
  sorts Path < Path_Err .
  ops source target : Path -> Nodes .
  op  _ :: _ : Path Path -> Path_Err assoc .
  vars p,q : Path .
  ceq  source((p :: q) as Path) = source(p) if p :: q : Path .
  ceq  target((p :: q) as Path) = target(q) if p :: q : Path .
  ceq  target(p)=source(q) if p :: q : Path .
  ceq  p :: q in Path if target(p)=source(q) .
endo

obj MY-GRAPH is
  using GRAPH .
  ops a b c d : -> Nodes .
  ops 1 2 3 : -> Edges .
  eq source(1) = a .
  eq target(1) = b .
  eq source(2) = b .
  eq target(2) = c .
  eq source(3) = c .
  eq target(3) = d .
endo

obj MY-PATH is
  protecting PATH[MY-GRAPH]
endo
```

In $COSASC^{\otimes}$, the inital MY-PATH-model contains as valus of the sort Path not only values like 1, 2, 3, 1::2, 2::3, but also values like r:Path_Err>Path(2::1). Thus, not only the sort Path_Err contains "error elements" like

```
  r:Path_Err>Path(2::1),
```

also the sort Path contains them!

Now in $COSASC_{\otimes}$, the specification behaves as expected: the sort Path of

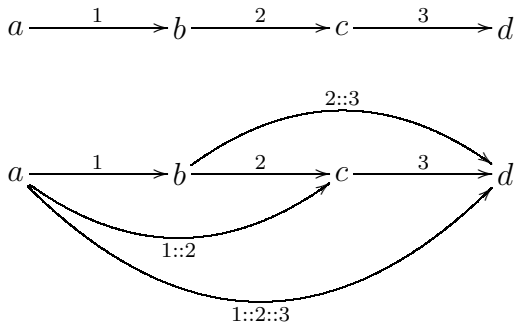


Fig. 6. The inital `MY-GRAPH`-model and the inital `MY-PATH`-model.

the inital `MY-PATH`-model indeed is just interpreted (up to change of representation) by the set

$$\{\texttt{1, 2, 3, 1::2, 2::3, 1::2::3 }\},$$

while the sort `Path_Err` additionally contains "error elements", for example `r:Path_Err>Path(2::1)`.

10.3 Translating $COSASC^{\otimes}$ to $SubCond^{=}$

In this and the follwing subsection, we describe two translations of institutions underlying OBJ3 to (fragments of) the CASL institution, serving different methodological purposes. The present section describes the translation of $COSASC^{\otimes}$ to subsorted conditional equational logic ($SubCond^{=}$). It translates retracts to *total* functions, and thus retracts usually generate new values in the carrier sets (which are called error values).

Theorem 10.8 The following describes a strongly persistently liberal simple institution representation from $COSASC^{\otimes}$ to $SubCond^{=}$ that is also an embedding.

Signatures A $COSASC^{\otimes}$-signature Σ is mapped to $\langle \Sigma^{\otimes}, \emptyset^{\otimes} \rangle$ considered as a $SubCond^{=}$-presentation (which means that the retracts in the retract equations have to be explicitly qualified with their profiles). A signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$ is mapped to the signature morphism $\sigma^{\otimes}\colon \Sigma^{\otimes} \longrightarrow \Sigma'^{\otimes}$. Now $\sigma^{\otimes}$ is monotone because σ is. It remains to show that $\sigma^{\otimes}$ preserves the overloading relation $\sim_F$. Let $f : w_1 \longrightarrow s_1 \sim_F f : w_2 \longrightarrow s_2$ in Σ, i.e. there exist $w_0 \in S^*$ with $w_0 \leq w_1$ and $w_0 \leq w_2$ and a common supersort s_0 of s_1 and s_2. By regularity of Σ, there is some $w \in S^*$, $s \in S$ with $f : w \longrightarrow s$ and $w_0 \leq w \leq w_1, w_2$. Since σ preserves overloading, $\sigma^F_{w_1,s_1}(f) = \sigma^F_{w,s}(f) = \sigma^F_{w_2,s_2}(f)$.

Models Given a $COSASC^{\otimes}$-signature Σ, the model translation is the fol-

lowing composite ($(\_)^\bullet$ has been introduced in Theorem 10.3):[19]

$$\begin{array}{ccccc}
\mathbf{Mod}^{COSA^\otimes}(\Sigma) & & & & \mathbf{Mod}^{SubCond^=}(\langle \Sigma^\otimes, \emptyset^\otimes \rangle) \\
\| & & & & \| \\
\mathbf{Mod}^{COSA}(\langle \Sigma^\otimes, \emptyset^\otimes \rangle) & \xleftarrow{(_)^\bullet} & \mathbf{Mod}^{Cond^=}(\langle \Sigma^{\otimes\#}, J^\otimes \rangle) & \xleftarrow{\mathbf{Mod}(\mu_{\Sigma^\otimes})} & \mathbf{Mod}^{Horn^=}(\langle \widehat{\Sigma^\otimes}, \hat{J}^\otimes \rangle)
\end{array}$$

Concerning strongly persistent liberality and the embedding property, $(\_)^\bullet$ has a left adjoint right inverse and is an equivalence of categories due to Theorem 10.3. $\mathbf{Mod}(\mu_{\Sigma^\otimes})$ just forgets membership predicates, which are fully determined by the axioms in $\hat{J}^\otimes$. Therefore, it is even an isomorphism. (The additional function-overloading axioms in $\hat{J}^\otimes$ already follow from those in $J^\otimes$ with an argument similar to the above argument showing preservation of overloading by signature morphisms.)

Sentences Given a $COSASC^\otimes$-signature Σ, for sentences without sort constraints, the sentence translation is the following composite (LP has been introduced in Theorem 10.3):

$$\begin{array}{ccccc}
\mathbf{Sen}^{COSA^\otimes}(\Sigma) & & & & \mathbf{Sen}^{SubCond^=}(\Sigma^\otimes) \\
\| & & & & \| \\
\mathbf{Sen}^{COSA}(\Sigma^\otimes) & \xrightarrow{LP_{\Sigma^\otimes}} & \mathbf{Sen}^{Cond^=}(\Sigma^{\otimes\#}) & \xrightarrow{\mathbf{Sen}(\mu_{\Sigma^\otimes})} & \mathbf{Sen}^{Horn^=}(\widehat{\Sigma^\otimes})
\end{array}$$

Sort constraints within sentences are translated as follows: Given $t : s$ as an atomic constituent of a sentence in $\mathbf{Sen}^{COSASC^\otimes}(\Sigma)$, let $s' = lub(s, LS(t))$ and translate it to

$$\mathtt{inj}_{(LS(t),s')}(\mathbf{Sen}(\mu_{\Sigma^\otimes})(LP_{\Sigma^\otimes}(t))) \in s$$

as an atomic constituent in $\mathbf{Sen}^{SubCond^=}(\Sigma^\otimes)$.

Satisfaction The representation condition follows from that of the representation in Theorem 10.3. For sort constraints, we also need the axioms for membership in $\hat{J}$. $\square$

When we translate the OBJ3 specification `MY-PATH` given above with this tranlsation, we get the CASL specification

spec GRAPH =
  sorts *Nodes*, *Edges*
  ops *source*, *target* : *Edges* $\rightarrow$ *Nodes*;
    a, *b*, *c*, *d* : *Nodes*;
    1, *2*, *3* : *Edges*

[19] Since $SubCond^=$ is a total framework, we have to leave out the partial projections from $\hat{\Sigma}$ as said in the paragraph about partiality in Section 3.4.1. Otherwise, the target of the translation would be $SubPCond^=$.

axioms $source(1) = a$;
$target(1) = b$;
$source(2) = b$;
$target(2) = c$;
$source(3) = c$;
$target(3) = d$
end

spec PATH[GRAPH] =
sorts $Edges < Path$;
$Path < Path\_Err$
ops $r\_Path\_Err\_Path : Path\_Err \to Path$;
$source, target : Path \to Nodes$;
$\_\_ :: \_\_ : Path \times Path \to Path\_Err$, **assoc**
forall $p, q : Path$
- $p :: q \in Path \Rightarrow source(r\_Path\_Err\_Path(p :: q)) = source(p)$
- $p :: q \in Path \Rightarrow target(r\_Path\_Err\_Path(p :: q)) = target(q)$
- $p :: q \in Path \Leftrightarrow target(p) = source(q)$

end

spec MY_GRAPH =
GRAPH
then
ops $a, b, c, d : Nodes$;
$1, 2, 3 : Edges$
axioms $source(1) = a$;
$target(1) = b$;
$source(2) = b$;
$target(2) = c$;
$source(3) = c$;
$target(3) = d$
end

spec MY_PATH =
PATH[MY_GRAPH]
end

10.4 Translating $COSASC_{\otimes}$ to $SubPCond^{=}$

The translation of $COSASC_{\otimes}$ to $SubPCond^{=}$ maps retracts to *partial* projections. This means that retracts do not generate new values in the carrier sets.

Theorem 10.9 The following describes a strongly persistently liberal simple institution representation from $COSASC_{\otimes}$ to $SubPCond^{=}$ that is also an

embedding.

Signatures Any $COSASC_{\otimes}$-signature is a $SubPCond^{=}$-signature by letting the set of partial function symbols be empty.

Models Given a $COSASC_{\otimes}$-signature Σ, the model translation is the following composite:

$$\begin{array}{ccccc} \mathbf{Mod}^{COSASC_{\otimes}}(\Sigma) & & & & \mathbf{Mod}^{SubPCond^{=}}(\Sigma) \\ \| & & & & \| \\ \mathbf{Mod}^{COSASC}(\Sigma) & \xleftarrow{(\_)^{\bullet}} & \mathbf{Mod}^{COSASC}(\langle \Sigma^{\#}, J \rangle) & \xleftarrow{\mathbf{Mod}(\mu_{\Sigma})} & \mathbf{Mod}^{PHorn^{=}}(\langle \hat{\Sigma}, \hat{J} \rangle) \end{array}$$

Concerning strongly persistent liberality and the embedding property, $(\_)^{\bullet}$ has a left adjoint right inverse and is a natural equivalence due to Theorem 10.3. $\mathbf{Mod}(\mu_{\Sigma})$ just forgets membership predicates and partial projections, which are fully determined by the axioms in $\hat{J}^{\otimes}$. Therefore, using a similar argument as in the proof of Theorem 10.8, it is even an isomorphism.

Sentences Given a $COSASC_{\otimes}$-signature Σ, sentences without sort constraints are translated with the following composite:

$$\begin{array}{ccccc} \mathbf{Sen}^{COSA_{\otimes}}(\Sigma) & & & & \mathbf{Sen}^{SubPCond^{=}}(\Sigma) \\ \| & & & & \| \\ \mathbf{Sen}^{COSA}(\Sigma^{\otimes}) & \xrightarrow{LP_{\Sigma^{\otimes}}} & \mathbf{Sen}^{Cond^{=}}(\Sigma^{\otimes\#}) & \xrightarrow{[\![\_]\!]} & \mathbf{Sen}^{PHorn^{=}}(\hat{\Sigma}) \end{array}$$

Here, $[\![\_]\!]$ is defined inductively as follows:

- $[\![x]\!] = x$
- $[\![\mathtt{inj}_{(s,s')}(t)]\!] = \mathtt{inj}_{(s,s')}([\![t]\!])$
- $[\![r : s' > s(t)]\!] = \mathtt{pr}_{(s',s)}([\![t]\!])$
- $[\![f_{w,s}(t_1, \ldots, t_n)]\!] = f_{w,s}([\![t_1]\!], \ldots, [\![t_n]\!])$
- $[\![t_1 = t_2]\!] = [\![t_1]\!] \stackrel{e}{=} [\![t_2]\!]$
- $[\![\forall X \bullet e_1 \wedge \ldots \wedge e_n \Rightarrow e]\!] = \forall X \bullet [\![e_1]\!] \wedge \ldots \wedge [\![e_n]\!] \Rightarrow [\![e]\!]$

A sort constraint $t : s$ occurring in a sentence is translated to

$$\mathtt{inj}_{(LS(t),s')}([\![LP_{\Sigma^{\otimes}}(t)]\!]) \in s,$$

where $s' = lub(s, LS(t))$.

Satisfaction Given $M \in \mathbf{Mod}^{PHorn^{=}}(\hat{\Sigma})$, by Theorem 10.3 without loss of generality we can assume that M consists of inclusions as injections already, and $(M|_{\mu_{\Sigma}})^{\bullet} = M|_{\mu_{\Sigma}}$ (note that also $(M|_{\mu_{\Sigma}})_s = M_s$). The key to prove the representation condition is the following lemma:

Lemma 10.10 Given $M \in \mathbf{Mod}^{PHorn^{=}}(\hat{\Sigma})$ with $(M|_{\mu_{\Sigma}})^{\bullet} = M|_{\mu_{\Sigma}}$, valuations $\nu\colon X \longrightarrow M$ are the same as valuations $\tilde{\nu}\colon X \longrightarrow (M|_{\mu_{\Sigma}})$. For $t \in T_{\Sigma^{\otimes}}(X)$, the following properties hold:

(1) $\nu^{\#}([\![LP_{\Sigma^{\otimes}}(t)]\!])$ is defined iff $(\iota_{(M|_{\mu_{\Sigma}})} \circ \tilde{\nu})^{\#}(t) \in \iota_{(M|_{\mu_{\Sigma}})}(M|_{\mu_{\Sigma}})$

(2) In case of (1), $\iota_{(M|_{\mu_\Sigma})}(\nu^{\#}([\![LP_{\Sigma^\otimes}(t)]\!])) = (\iota_{(M|_{\mu_\Sigma})} \circ \tilde{\nu})^{\#}(t)$.

Proof. The proof proceeds by induction over the structure of t. The only critical case is that of retracts. We have $\nu^{\#}([\![LP_{\Sigma^\otimes}(r : s' > s(t))]\!])$ is defined iff $\nu^{\#}(\mathtt{pr}_{(s',s)}([\![LP_{\Sigma^\otimes}(t)]\!]))$ is defined iff $\nu^{\#}([\![LP_{\Sigma^\otimes}(t)]\!])$ is defined and $\in M_s$ iff $(\iota_{M|_{\mu_\Sigma}} \circ \tilde{\nu})^{\#}(t) \in \iota_{M|_{\mu_\Sigma}}((M|_{\mu_\Sigma})_s)$ iff (since retracts are interpreted freely outside M_s) $(\iota_{M|_{\mu_\Sigma}} \circ \tilde{\nu})^{\#}(r : s' > s(t)) \in \iota_{M|_{\mu_\Sigma}}((M|_{\mu_\Sigma})_s)$. □

Proposition 10.11 The model translation component of the representation from $COSASC_\otimes$ to $SubPCond^{=}$ is a natural equivalence.

Proof. $(\_)^\bullet$ is shown to be a natural equivalence in [36], and $\mathbf{Mod}(\mu_\Sigma)$ is a natural equivalence since $\langle \hat{\Sigma}, \hat{J} \rangle$ is a definitional extension of $\langle \Sigma^{\#}, J \rangle$. □

With this translation, the specification `Path` is translated to CASL as follows:

spec GRAPH =
  sorts *Nodes*, *Edges*
  ops *source*, *target* : *Edges* → *Nodes*;
    a, *b*, *c*, *d* : *Nodes*;
    1, *2*, *3* : *Edges*;
  axioms *source*(*1*) = *a*;
    target(*1*) = *b*;
    source(*2*) = *b*;
    target(*2*) = *c*;
    source(*3*) = *c*;
    target(*3*) = *d*
end

spec PATH [GRAPH] =
  sorts *Edges* < *Path*;
    Path < *Path_Err*
  ops *source*, *target* : *Path* → *Nodes*;
    __ :: __ : *Path* × *Path* → *Path_Err*
  forall *p*, *q* : *Path*
  • *p* :: *q* ∈ *Path* ⇒ *source*((*p* :: *q*) *as Path*) = *source*(*p*)
  • *p* :: *q* ∈ *Path* ⇒ *target*((*p* :: *q*) *as Path*) = *target*(*q*)
  • *p* :: *q* ∈ *Path* ⇔ *target*(*p*) = *source*(*q*)
end

10.5 Translating OBJ3's Constructs into CASL *Constructs*

We now briefly describe how any of the institution representations defined in the previous section can be lifted to a translation of language constructs.

One difference between OBJ3 and CASL concerns disambiguation of terms. Sort disambiguation in OBJ3 inserts subsort inclusions and retracts, while in CASL, only subsort injections are inserted. Note that CASL's disambiguation [61] corresponds to the least sort parse for regular signatures. Thus, at the construct level, the translation from OBJ3 to CASL has just to insert retracts. (It does not matter here that at the concepts level, CASL terms are disambiguated while OBJ3 terms are not.)

Mixfix declaration facilities are essentially the same for OBJ and CASL. Precedences between mixfix operations are expressed with numeric priorities in OBJ3, while CASL allows arbitrary pre-orders as in ASF+SDF. Thus, OBJ3 numeric priorities have to be translated to the pre-order induced by them. Concerning the associativity of mixfix operations, OBJ3 allows to declare rather general *gathering patterns*. These are not available in CASL. CASL only allows to declare binary infix operations to be left or right associative, this corresponds to special gathering patterns in OBJ3. Other gathering patterns cannot be translated to CASL. Instead, one has to parse the formulas and terms with an OBJ3 parser (using the gathering patterns) and using explicit bracketing for disambiguation wherever necessary in the translation to CASL.

Explicit sort disambiguation (written $t.s$ in OBJ3) is also available in CASL (written $t : s$). Sort constraints, written as $t : s$ in (extensions of) OBJ3, are written as membership formulas $t \in s$ in CASL.

Associativity, commutativity, identity and idempotence attributes are slightly more general in OBJ3 than in CASL, since in CASL, argument and result sorts of the operations with attributes have to be the same. In the case that this condition does not hold, the attribute has to be translated to an axiom.

OBJ3's subsort declarations allow chains of subsorts to be declared. This has to be split up into several declarations in CASL. Finally, OBJ3's `let` definitions have to be translated to CASL's operation definitions.

Concerning functional CafeOBJ, a distinguishing feature of fun-CafeOBJ as used in the CafeOBJ report is that each sort comes equipped with an error supersort. This feature need not be treated at the level of the logic though: the right place to introduce the error supersorts is the level of language constructs, namely the function that extracts a signature in the institution from a given specification should introduce the error supersorts.

11 Boolean Functions and Empty Carriers

In this section, two problems that are common to all specification language translations are discussed, namely the treatment of boolean functions and empty carriers.

11.1 Translating Boolean Functions to Predicates

Larch, ACT ONE and OBJ3 all have a built-in sort `Bool` or `Boolean` of boolean values, together with some Boolean operations.[20] The translations of the previous sections treat `Bool` like any other sort. This means that the implicit inclusion of `Bool` into each specification must be handled at the level of structured specifications: When translating a Larch, ACT ONE or OBJ3 specification to CASL, one has to make the implicit import of `Bool` (together with possibly equality and if-then-else functions, depending of the language) explicit in the result of the translation. Further, when translating ACT ONE or OBJ3, the CASL specification of `Bool` has to be surrounded by a **free { ... }**, since the Booleans are always interpreted initially in these languages. For the declared constructor functions, ACT ONE generates special equations of sort `Bool`, which have to be included in the translation as well.

However, it may be desirable to treat `Bool` as a special sort and translate Larch's, ACT ONE's or OBJ3's `Bool`-valued functions into CASL's *predicates*. This works only for theories that do not use `Bool` as argument sort of a function.

We thus can modify the translations of Sections 6.1, 10.3 and 10.4 by mapping `Bool`-valued functions into predicates at the signature level. At the sentence level, applications of `Bool`-valued functions are translated to applications of predicates, and the standard `Bool`-functions are translated into the logical connectives.[21] At the model level, predicates are translated to `Bool`-valued functions.

For example, concerning OBJ3, let us restrict $COSASC^{\otimes}$ (or $COSASC_{\otimes}$) to signatures including the standard signature `BOOL` (with a fixed meaning), but without any functions having `Bool` as argument sort except those in `BOOL`, thus getting the institution $COSASC^{\otimes}$-$Bool$ (or the institution $COSASC_{\otimes}$-$Bool$). We then have a strongly persistently liberal institution representation from

[20] For LSL, the institution underlying Larch, Baumeister [8] has developed a formal treatment of signatures with fixed Boolean subsignature.

[21] If-then-else is available only as a construct, not as a concept in CASL. The translation from CASL constructs to concepts can be used as translation here.

$COSASC^{\otimes}$-*Bool* to $SubFOL^{=}$, and another one from $COSASC_{\otimes}$-*Bool* to $SubPFOL^{=}$. A similar remark holds for LSL: we get a representation $LSL\text{-}Bool \longrightarrow CFOL^{=}$.

Concerning ACT ONE, we can use $GCond^{=}\text{-}Bool \longrightarrow FOL^{=}$ constructed in an analogous way. (This representation can also be used for ASF, although ASF does not have an implicitly imported `Bool`). Now in this setting (of translating `Bool`-valued functions to CASL predicates), the special axioms of sort `Bool` generated for constructors in ACT ONE (stating injectivity of constructors and disjointness of their images) can be generated with a **free type ...** construct in CASL, which generates exactly the same axioms as first-order formulas.

11.2 The Empty Carrier Problem

All institutions of order-sorted and partial algebras which we have defined in this paper do not admit empty carriers. The reader may be surprised by this, since many-sorted and subsorted algebras usually are defined in a way allowing empty carriers [28,36].

The main reason to disallow empty carriers in CASL was that when allowing empty carriers, the satisfaction of a sentence like

$$\forall x : s \bullet a = b$$

would depend not only on the equality of the interpretations of a and b, but also on the question whether the carrier for sort s is empty or not. This may lead to confusion, especially when the variable $x : s$ is not introduced locally, but through a global variable declaration.

Surprisingly, another reason to forbid empty carriers lies in the framework of order-sorted algebra that has been developed by advocates of empty carriers: Lemma 10.5, which is crucial for proving the satisfaction condition of $COSASC_{\otimes}$, only holds if we forbid models that mix empty and non-empty carriers. [22] We here additionally forbid models consisting only of empty carriers, but this additional restriction is not essential in any respect.

Forbidding empty carriers is not a great loss, since it is unlikely that a datatype contains no data. Moreover, it is even an advantage: since the standard definitions of first-order and higher-order logic forbid empty carriers as well, there is a straightforward bridge from CASL to $FOL^{=}$- or $HOL^{=}$-based theo-

[22] It also holds if we restrict ourselves to confluent sets of axioms [36], but this does not help for defining an *institution*.

rem proving tools (which usually assume non-empty carriers). Empty carriers would overly complicate the use of such tools for CASL.

It has been argued [35] that allowing empty carriers is necessary to get initial models and free extensions. However, Theorem 4.16 shows that initial models and free extensions exist if we restrict ourselves to strict signatures. Now in [35] it is argued that there are useful theories that are not strict (e.g. the theory of pre-ordered sets). However, such theories can still be used as *parameter* theory for free extensions, since Theorem 4.16 only requires the *target* theory to be strict. It will hardly be the case that one wants to have initial or free semantics for non-strict theories.

For the rare cases where an example specification should really use empty carriers, and for the re-use of tools supporting empty carriers, we now examine the relation between institutions with and without empty carriers and set up representations between these institutions. Then, by composing with one of these representations, we can shift the representations of Fig. 3, 4 and 5 to representations between institutions that differ in their behaviour w.r.t. empty carriers.

For any institution I introduced in the previous sections, let I^{mt} denote I with a slightly modified model functor that allows the possibility of empty carriers in the model.

Let us now consider the representation of I in I^{mt} and vice versa at the first-order level.[23] There are easy representations between $FOL^{=}$ and $(FOL^{=})^{mt}$:

- For the representation of $(FOL^{=})^{mt}$ in $FOL^{=}$, we can simply use representation (4) from the first-order level (Section 4.1.4) and delete the axiom $\exists x : s \bullet D_s(x)$ stating non-emptiness of the set of "defined" elements. This gives a representation of $(PFOL^{=})^{mt}$ in $FOL^{=}$, which then can be composed with the inclusion of $(FOL^{=})^{mt}$ into $(PFOL^{=})^{mt}$. As in (4), we thus get a strongly persistently liberal representation with the weak (injective)-amalgamation property.
- $FOL^{=}$ can be represented in $(FOL^{=})^{mt}$ by adding to a signature an axiom

$$\exists x : s \bullet T$$

  for each sort s. Axioms and models are translated identically. This gives a subinstitution representation. □

Instead of $FOL^{=}$ we could have used any other institution at the first-order level and get the same results.

[23] Note that most formulations of first-order logic exclude empty carriers anyway, so these representations will be needed very rarely only.

At the positive conditional level, we do not have such strong relations.[24] We have the following representations between $(SubP)Horn^{=}$ and $(SubP)Horn^{=mt}$:

- $(Horn^{=})^{mt}$ can be represented in $Horn^{=}$ in much the same way as $(FOL^{=})^{mt}$ can be represented in $FOL^{=}$ above. We have to delete not only the axioms of form $\exists x : s \bullet D_s(x)$, but also the operations $\bot$ and the axioms involving them in order to get theories in Horn form. This gives a strongly persistently liberal representation (however, due to the deletion of $\bot$, weak (injective)-amalgamation is lost).

  The same thing also works for the superinstitutions of $Horn^{=}$ defined at the positive conditional level.

  Concerning representing $(Cond^{=})^{mt}$ in $Cond^{=}$, we can use representation (8) from the positive conditional level and delete the newly introduced constants that guarantee the existence of "defined" elements.
- $Horn^{=}$ can be represented in $Horn^{=mt}$ by adding to a signature an overloaded constant $c : s$ for each sort s. Axioms and models are translated identically. This gives a model-bijective institution representation. The same thing also works for the sub- and superinstitutions of $Horn^{=}$ defined at the positive conditional level. $\square$

12 Conclusion

We have considered the relation of CASL to other specification languages at the level of specification in-the-small. To this end, we have defined a number of subinstitutions of the institution $SubPCFOL^{=}$ underlying the specification language CASL.

Among these subinstitutions of $SubPCFOL^{=}$, we have defined a number of institution representations. Altogether, three graphs of institutions and representations arise: two at the first-order level (one with, one without sort generation constraints), and one at the positive conditional level. We also have classified the institution representations according to different properties, and have shown how these properties lead to a good interaction with theorem proving. More specifically, the properties allow (1) the lifting of theorem provers for flat specifications, (2) for structured specifications without **free**, (3) for structured specifications including **free** and/or (4) the lifting of loose semantics. As a consequence, we get

(1) All the institutions of the positive conditional level are liberal w.r.t. so-called strict theory morphisms (this is proved by lifting free constructions

[24] Note that at the positive conditional level, there are much more frameworks allowing empty carriers than in the first-order case.

from well-known institutions to other subinstitutions of CASL).

(2) First-order theorem provers with induction (or, at the level of positive conditional specifications, conditional term rewriters and paramodulation) can be "lifted" for use for theorem proving within basic specifications in CASL. At the first-order level, this result also holds for structured specifications. Actually, we have used the composition of translations $(7) \circ (6') \circ (5a')$ (cf. Section 4.1) from the first-order level, going from the CASL logic $SubPCFOL^=$ to second-order logic $SOL^=$, as the basis for the logical encoding provided within the CASL tool set (CATS) [51]. CATS also provides the intermediate steps of this composite encoding, having targets $CFOL^=$ and $SubCFOL^=$ (the former roughly being first-order logic with induction). Thus it is possible to use CATS to connect CASL to standard first-order or higher-order theorem provers.

    As a first practical example, we have used this to encode CASL in Isabelle/HOL in the HOL-CASL system, see [58] for details. The experience was that the institution representation can serve as a semantical basis for the re-use of theorem provers, but to make it work in practice requires more work. One important point is the question of a partial inverse of the representation, which makes it possible to display intermediate formulas in a proof in the syntax of the represented institution.

(3) Moreover, we have described the translation of a number of well-known specification languages (Larch, OBJ3, functional CafeOBJ, ACT, ASF, and HEP-theories) into CASL. We have set up formally precise translations at the level of the underlying logics (formalized as institutions), and informally discussed how this lifts to the level of language constructs. Thus, specification libraries and case studies developed in these languages can be translated to CASL.

Most of the specification languages that we have examined are based on a rather straightforwardly defined subinstitution of the CASL institution. The exception is OBJ3. When attempting to translate OBJ3 to CASL, we first had to clarify what the institution underlying OBJ3 is. Actually, we have found two institutions for OBJ3, serving different methodological purposes. One institution ($COSASC^{\otimes}$) treats error values generated by retracts as first-class citizens, allowing error recovery, while the other one ($COSASC_{\otimes}$) treats error values as virtual values, allowing a clean separation of ordinary and error values.

We also have addressed the level of language constructs and have (informally) shown how the constructs for writing specifications in-the-small from other languages can be translated into CASL. A formal treatment would need a formal definition of the other languages, which is not available in all cases and which also would be too detailed.

Finally, we have shown how the problem that most of the existing algebraic

specification languages differ from CASL in that they have a semantics allowing models with empty carriers can be solved.

An important future extension of the translations will be to consider also the translation among different concepts for specifications in-the-large. Here, the main difficulty will be the different philosophies behind the module systems: While OBJ3, CafeOBJ and ACT TWO use a colimit-based approach where the same name can have different meanings (depending on its origin module), CASL follows a same-name-same-thing philosophy.

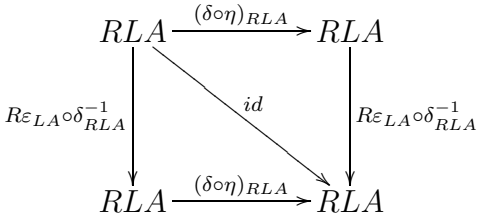
Acknowledgments I would like to thank Maura Cerioli, Răzvan Diaconescu, Joseph A. Goguen, Anne Haxthausen, Kolyang, Hans-Jörg Kreowski, Bernd Krieg-Brückner, José Meseguer, Peter D. Mosses, Markus Roggenbach, Grigore Rosu, Don Sannella, Lutz Schröder and Andrzej Tarlecki for intensive and sometimes controversial, but always productive discussions, and further, all the (other) participants of CoFI. Special thanks to Lutz Schröder for help with the appendix.

A Preservation of Freeness

The following propositions are important for proving properties about persistently liberal institution representations. We will use the following terminology: Given a functor F left adjoint to G, η^G and ε^F (or just η and ε, if no confusion can arise) will denote the unit and counit of some corresponding adjoint situation.

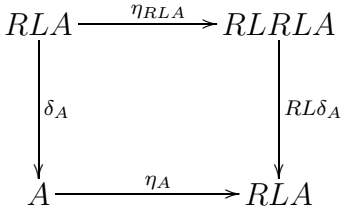
Proposition A.1 Given $R\colon \mathbf{B} \longrightarrow \mathbf{A}$ with left adjoint L such that $R \circ L \cong id$, then η, ε_L, and $R\varepsilon$ are isomorphisms. Moreover, L is full.

Proof. Let $\delta\colon R \circ L \longrightarrow id$ be a natural isomorphism, and let A be an object in $\mathbf{A}$.



Since $R\varepsilon \circ \eta_R = id$ for any adjunction, $R\varepsilon_{LA} \circ \delta^{-1}_{RLA} \circ \delta_{RLA} \circ \eta_{RLA} = id$. Hence, the upper triangle commutes. By naturality of $\delta \circ \eta$, also the square commutes. Thus, also the lower triangle commutes. But this shows $(\delta \circ \eta)_{RLA}$ to be an isomorphism. Since δ is an isomorphism, η_{RLA} is an isomorphism as well. By

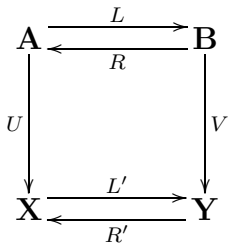
naturality of η,



commutes. Hence, η_A is an isomorphism. Since $\varepsilon_L \circ L\eta = id$ and $R\varepsilon \circ \eta_R = id$ for any adjunction, ε_L and $R\varepsilon$ are isomorphisms as well.

Fullness of L follows with the dual of [1], 19.14. $\square$

Proposition A.2 Let the following diagram of categories and functors be given.



Assume further that

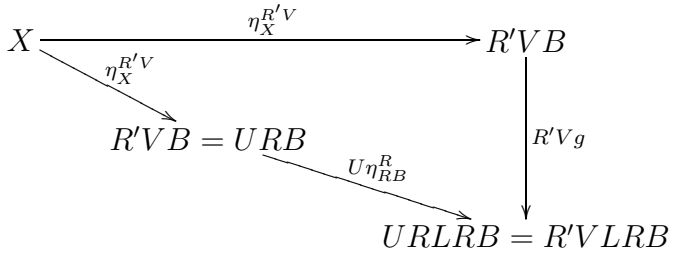
- $U \circ R = R' \circ V$,
- $R \circ L \cong id$ and $R' \circ L' \cong id$,
- L is left adjoint to R, and
- L' is left adjoint to R'.

Then

(1) If $B \in \mathbf{B}$ is V-free over $L'X$ for some $X \in \mathbf{X}$, then RB is U-free over X.
(2) $A \in \mathbf{A}$ is U-free over $X \in \mathbf{X}$ iff LA is V-free over $L'X$.
(3) If $B \in \mathbf{B}$ is strongly persistently V-free and VB is in the image of L', then RB is strongly persistently U-free.
(4) Further assume that $L' \circ U = V \circ L$. Then $A \in \mathbf{A}$ is strongly persistently U-free iff LA is strongly persistently V-free.

Proof. (1)

By Proposition A.1, η^R and ε^L_L are isomorphisms, and L is full. Assume that $B \in \mathbf{B}$ is V-free over $L'X$, i.e. there is a V-universal arrow $\eta^V_{L'X}: L'X \longrightarrow VB$. By composition with the R'-universal arrow $\eta^{R'}_X: X \longrightarrow R'L'X$ we get an $R'V$-universal arrow $\eta^{R'V}_X = R'\eta^V_{L'X} \circ \eta^{R'}_X: X \longrightarrow R'VB$. By its universality, there is a morphism $g: B \longrightarrow LRB$ with $R'Vg \circ \eta^{R'V}_X = U\eta^R_{RB} \circ \eta^{R'V}_X$.



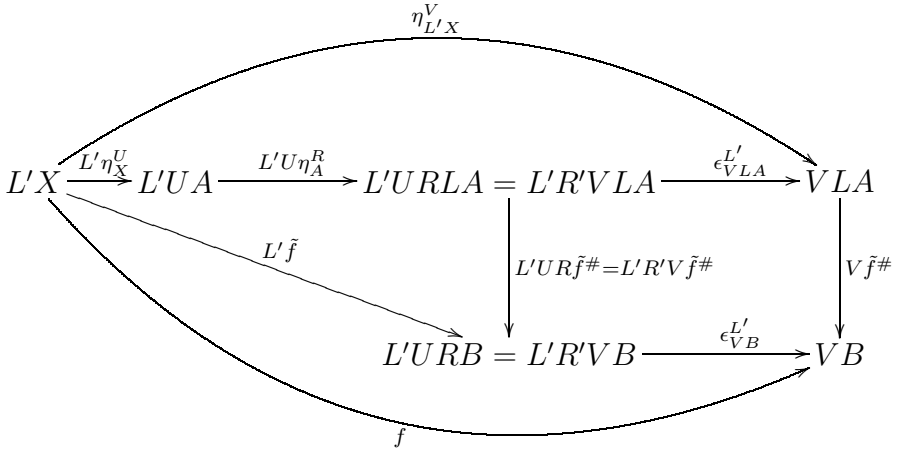
Now $R'V\varepsilon_B^L \circ R'Vg \circ \eta_X^{R'V} = UR\varepsilon_B^L \circ U\eta_{RB}^R \circ \eta_X^{R'V} = \eta_X^{R'V}$. By universality of $\eta_X^{R'V}$, we get $\varepsilon_B^L \circ g = id$, and thus also $\varepsilon_B^L \circ g \circ \varepsilon_B^L = \varepsilon_B^L$. Since L is full, $g \circ \varepsilon_B^L$ is the L-image of an $\mathbf{A}$-morphism. By co-universality of ε_B^L, we get $g \circ \varepsilon_B^L = id$. Thus, $g\colon B \longrightarrow LRB$ is an isomorphism. But then, $\eta_X^U := \eta_X^{R'V}\colon X \longrightarrow URB$ can be shown to be a U-universal arrow as follows: If $f\colon X \longrightarrow UA$ is an $\mathbf{X}$-morphism, by universality of $\eta_X^{R'V}$, there is a unique morphism $f^\#\colon B \longrightarrow LA$ satisfying $R'Vf^\# \circ \eta_X^{R'V} = U\eta_A^R \circ f$. Now $(\eta_A^R)^{-1} \circ R(f^\# \circ g^{-1}) \circ \eta_{RB}$ is a morphism from RB to A with $U((\eta_A^R)^{-1} \circ R(f^\# \circ g^{-1}) \circ \eta_{RB}) \circ \eta_X^{R'V} = U(\eta_A^R)^{-1} \circ UR(f^\#) \circ UR(g^{-1}) \circ U\eta_{RB}^R \circ UR\varepsilon_B^L \circ URg \circ \eta_X^{R'V} = U(\eta_A^R)^{-1} \circ UR(f^\#) \circ UR(g^{-1}) \circ URg \circ \eta_X^{R'V} = U(\eta_A^R)^{-1} \circ UR(f^\#) \circ \eta_X^{R'V} = U(\eta_A^R)^{-1} \circ U\eta_A^R \circ f = f$. Since g is an isomorphism, uniqueness follows from that of $f^\#$. $\square$

Proof of (2), "$\Longrightarrow$":

Let $\eta_X^U\colon X \longrightarrow UA$ be U-universal, and $\eta_A^R\colon A \longrightarrow RLA$ be R-universal. Then

$$\eta_{L'X}^V := L'X \xrightarrow{L'\eta_X^U} L'UA \xrightarrow{L'U\eta_A^R} L'URLA = L'R'VLA \xrightarrow{\epsilon_{VLA}^{L'}} VLA$$

is a V-universal arrow: Let $f\colon L'X \longrightarrow VB$ be an $\mathbf{X}$-morphism. By co-universality of $\epsilon_{VB}^{L'}$, there is some unique $\tilde{f}\colon X \longrightarrow R'VB = URB$ with $\epsilon_{VB}^{L'} \circ L'\tilde{f} = f$. By UR-universality of $U\eta_A^R \circ \eta_X^U$, there is some unique $\tilde{f}^\#\colon LA \longrightarrow B$ with $UR\tilde{f}^\# \circ U\eta_A^R \circ \eta_X^U = \tilde{f}$. By also considering the commutativity of the square (due to naturality of $\epsilon^{L'}$), $\tilde{f}^\#$ is the unique morphism from LA to B with $V\tilde{f}^\# \circ \eta_{L'X}^V = f$.



□

Proof of (2), "$\Longleftarrow$":

Follows from (1) with $B = LA$, since $RLA \cong A$. □

Proof of (3): Follows from (1) with X such that $L'X = VB$, where Proposition A.1 ensures that the unit constructed in the proof is an isomorphism, which leads to strongly persistent freeness by Proposition 2.10. □

Proof of (4):

Follows from (2) with $X = UA$, since by assumption, $L'UA = VLA$, where $\epsilon^{L'}_{VL} = \epsilon^{L'}_{L'U}$ and proposition A.1 ensure that the unit constructed in the proof is an isomorphism, which leads to strongly persistent freeness by Proposition 2.10. □

B Locally finitely presentable categories

Definition B.1 [2] An object K of a category $\mathcal{K}$ is called *finitely presentable* provided that its hom-functor

$$hom(K, \_): \mathcal{K} \longrightarrow \mathbf{Set}$$

preserves directed colimits.

For example, a set is finitely presentable in **Set** iff it is finite. A many-sorted algebra is finitely presentable in $\mathbf{Mod}(S, F)$ iff it can be presented by finitely many generators and finitely many equations in the usual algebraic sense.

Definition B.2 [2] A category $\mathcal{K}$ is called *locally finitely presentable* provided that is cocomplete and has a set $\mathcal{A}$ of finitely presentable objects such that every object is a directed colimit of objects from $\mathcal{A}$.

Proposition B.3 [2] Each locally finitely presentable category is complete. □

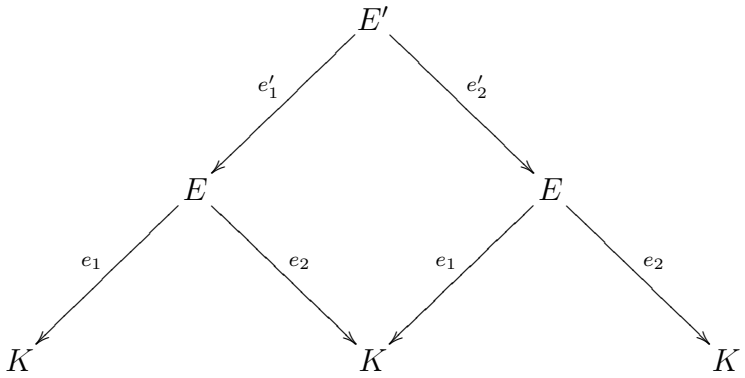
Locally finitely presentable categories are categories that satisfy some completeness properties (completeness and cocompleteness) and some smallness properties (roughly speaking, they are categories of structures with operations of finite arities).

C Effective equivalence relations

Definition C.1 (cf. [2, 3.4(8)])

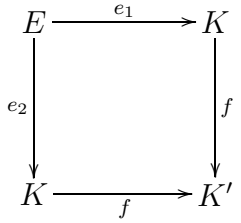
A *relation* on an object K is a subobject of $K \times K$ (usually represented by a pair $e_1, e_2 \colon E \longrightarrow K$ of morphisms such that the morphism $\langle e_1, e_2 \rangle \colon E \longrightarrow K \times K$ is a monomorphism). We call $e_1, e_2 \colon E \longrightarrow K$ an *equivalence relation* provided that it is

(1) *reflexive*, i.e., the diagonal of $K \times K$ is contained in the subobject represented by $\langle e_1, e_2 \rangle$
(2) *symmetric*, i.e., the monomorphisms $\langle e_1, e_2 \rangle$ and $\langle e_2, e_1 \rangle$ represent the same subobject
(3) *transitive*, i.e., when we form the pullback of e_2 and e_1:



then the subobject represented by $\langle e_1 \circ e'_1, e_2 \circ e'_2 \rangle$ is contained in that represented by $\langle e_1, e_2 \rangle$.

Definition C.2 A *kernel pair* of a morphism $f \colon K \longrightarrow K'$ in a category is a pair $e_1, e_2 \colon E \longrightarrow K$ such that



is a pullback.

A category has *effective equivalence relations*, if every equivalence relation is the kernel pair of some morphism.

References

[1] J. Adámek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories*. Wiley, New York, 1990.

[2] J. Adámek and J. Rosický. *Locally Presentable and Accessible Categories*. Cambridge University Press, 1994.

[3] D. Ancona, M. Cerioli, and E. Zucca. Extending CASL by late binding. In C. Choppy, D. Bert, and P. Mosses, editors, *Recent Trends in Algebraic Development Techniques, 14th International Workshop, WADT'99, Bonas, France*, volume 1827 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[4] E. Astesiano, M. Bidoit, H. Kirchner, B. Krieg-Brückner, P. D. Mosses, D. Sannella, and A. Tarlecki. CASL: The common algebraic specification language. *Theoretical Computer Science*, this volume.

[5] E. Astesiano and M. Cerioli. Free objects and equational deduction for partial conditional specifications. *Theoretical Computer Science*, 152:91–138, 1995.

[6] M. Barr. Models of Horn theories. *Contemporary Mathematics*, 92:1–7, 1989.

[7] J. Barwise. Axioms for abstract model theory. *Annals of Mathematical Logic*, 7:221–265, 1974.

[8] H. Baumeister. *Relations between Abstract Datatypes modeled as Abstract Datatypes*. PhD thesis, Universität des Saarlandes, 1998.

[9] H. Baumeister and A. Zamulin. State-based extension of CASL. Submitted for publication, 2000.

[10] J. Bergstra, J. Heering, and P. Klint. Module algebra. *J. ACM*, 37(2):335–372, 1990.

[11] J. A. Bergstra, J. Heering, and P. Klint, editors. *Algebraic Specification*. Addison-Wesley, 1989.

[12] T. Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, this volume.

[13] P. Burmeister. Partial algebras — survey of a unifying approach towards a two-valued model theory for partial algebras. *Algebra Universalis*, 15:306–358, 1982.

[14] P. Burmeister. *A model theoretic approach to partial algebras*. Akademie Verlag, Berlin, 1986.

[15] P. Burmeister, M. Llabrés, and F. Rosselló. Pushout complements for partly total algebras. Mathematical Structures in Computer Science, to appear.

[16] M. Cerioli. *Relationships between Logical Formalisms.* PhD thesis, TD-4/93, Università di Pisa-Genova-Udine, 1993.

[17] M. Cerioli, A. Haxthausen, B. Krieg-Brückner, and T. Mossakowski. Permissive subsorted partial logic in CASL. In M. Johnson, editor, *Algebraic methodology and software technology: 6th international conference, AMAST 97*, volume 1349 of *Lecture Notes in Computer Science*, pages 91–107. Springer-Verlag, 1997.

[18] M. Cerioli and J. Meseguer. May I borrow your logic? (transporting logical structures along maps). *Theoretical Computer Science*, 173:311–347, 1997.

[19] M. Cerioli, T. Mossakowski, and H. Reichel. From total equational to partial first order logic. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specifications*, pages 31–104. Springer Verlag, 1999.

[20] I. Classen, H. Ehrig, and D. Wolz. *Algebraic Specification Techniques and Tools for Software Development: The ACT Approach.* World Scientific, Singapore, 1993.

[21] CoFI. The Common Framework Initiative for algebraic specification and development, electronic archives. Notes and Documents accessible from `http://www.brics.dk/Projects/CoFI/`.

[22] CoFI Language Design Task Group. CASL – The CoFI Algebraic Specification Language – Summary. Documents/CASL/Summary, in [21], Mar. 2001.

[23] CoFI Semantics Task Group. CASL – The CoFI Algebraic Specification Language – Semantics. Note S-9 (Documents/CASL/Semantics, version 0.96), in [21], July 1999.

[24] R. Diaconescu. Extra theory morphisms for institutions: logical semantics for multi-paradigm languages. *J. Applied Categorical Structures*, 6:427–453, 1998.

[25] R. Diaconescu. Personal communication, 2001.

[26] R. Diaconescu and K. Futatsugi. *CafeOBJ Report.* World Scientific, Singapore, 1998.

[27] R. Diaconescu, J. Goguen, and P. Stefaneas. Logical support for modularisation. In G. Huet and G. Plotkin, editors, *Proceedings of a Workshop on Logical Frameworks*, 1991.

[28] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1.* Springer Verlag, Heidelberg, 1985.

[29] H. Ehrig, E. G. Wagner, and J. W. Thatcher. Algebraic specifications with generating constraints. In *Proc. 10th Intl. Colloq. on Automata, Languages and Programming*, pages 188–202. Springer LNCS 154, 1983.

[30] J. Goguen, J.-P. Jouannaud, and J. Meseguer. Operational semantics of order-sorted algebra. In W. Brauer, editor, *Proceedings, 1985 International Conference on Automata, Languages and Programming*, volume 194 of *Lecture Notes in Computer Science*, pages 221–231. Springer, 1985.

[31] J. Goguen, J. Meseguer, T. Winkler, K. Futatsugi, P. Lincoln, and J.-P. Jouannaud. Introducing OBJ. Technical Report SRI-CSL-88-8, Computer Science Lab, SRI International, August 1988; revised version from 24th October 1993.

[32] J. A. Goguen. Stretching first order equational logic: Proofs about partiality using subsorts and retracts. Unpublished Draft. Available from `http://www-cse.ucsd.edu/users/goguen/pubs/`. Earlier version appeared in Proceedings, International Workshop on First Order Theorem proving, edited by Maria Paola Bonacina and Ulrich Furbach, RISC-Linz Report 97-70, pages 78–85, 1997, 1998.

[33] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39:95–146, 1992. Predecessor in: LNCS 164, 221–256, 1984.

[34] J. A. Goguen and J. Meseguer. Eqlog: Equality, types, and generic modules for logic programming. In D. DeGroot and G. Lindstrom, editors, *Logic Programming. Functions, Relations and Equations*, pages 295–363. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[35] J. A. Goguen and J. Meseguer. Remarks on remarks on many-sorted equational logic. *EATCS Bulletin*, 30:66–73, 1986.

[36] J. A. Goguen and J. Meseguer. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.

[37] J. A. Goguen, J. W. Thatcher, and E. G. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In R. Yeh, editor, *Current Trends in Programming Methodology*, volume 4, pages 80–144. Prentice Hall, 1978.

[38] J. A. Goguen and T. Winkler. Introducing OBJ3. Research report SRI-CSL-88-9, SRI International, 1988.

[39] J. V. Guttag and J. J. Horning. Report on the Larch shared language. *Science of Computer Programming*, 6(2):103–134, 1986.

[40] J. V. Guttag, J. J. Horning, S. J. Garland, K. D. Jones, A. Modet, and J. M. Wing. *Larch: Languages and Tools for Formal Specification*. Springer-Verlag, New York, N.Y., 1993.

[41] A. Haxthausen and F. Nickl. Pushouts of order-sorted algebraic specifications. In *Proceedings of AMAST'96*, volume 1101 of *Lecture Notes in Computer Science*, pages 132–147. Springer-Verlag, 1996.

[42] H. Herrlich and G. Strecker. *Category Theory.* Allyn and Bacon, Boston, 1973.

[43] H. Hussmann, M. Cerioli, and H. Baumeister. From UML to CASL (static part), 2000. Technical Report of DISI - Universit di Genova, DISI-TR-00-06, Italy.

[44] H.-J. Kreowski and T. Mossakowski. Equivalence and difference of institutions: Simulating horn clause logic with based algebras. *Mathematical Structures in Computer Science*, 5:189–215, 1995.

[45] J. Lloyd. *Foundations of Logic Programming.* Springer Verlag, 1987.

[46] V. Manca, A. Salibra, and G. Scollo. Equational type logic. *Theoretical Computer Science*, 77:131–159, 1990.

[47] J. Meseguer. General logics. In *Logic Colloquium 87*, pages 275–329. North Holland, 1989.

[48] J. Meseguer. Conditional rewriting as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–156, 1992.

[49] J. Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi Presicce, editor, *Recent trends in algebraic development techniques. Proc. 12th International Workshop*, volume 1376 of *Lecture Notes in Computer Science*, pages 18–61. Springer, 1998.

[50] J. Meseguer and J. Goguen. Order-sorted algebra solves the constructor, selector, multiple representation and coercion problems. *Information and Computation*, 103(1):114–158, March 1993.

[51] T. Mossakowski. The CASL tool set. Available at `http://www.tzi.de/cofi/CATS`.

[52] T. Mossakowski. A hierarchy of institutions separated by properties of parameterized abstract data types. In E. Astesiano, G. Reggio, and A. Tarlecki, editors, *Recent Trends in Data Type Specification. Proceedings*, volume 906 of *Lecture Notes in Computer Science*, pages 389–405. Springer Verlag, London, 1995.

[53] T. Mossakowski. Equivalences among various logical frameworks of partial algebras. In H. K. Büning, editor, *Computer Science Logic. 9th Workshop, CSL'95. Paderborn, Germany, September 1995, Selected Papers*, volume 1092 of *Lecture Notes in Computer Science*, pages 403–433. Springer Verlag, 1996.

[54] T. Mossakowski. *Representations, hierarchies and graphs of institutions.* PhD thesis, Bremen University, 1996.

[55] T. Mossakowski. Using limits of parchments to systematically construct institutions of partial algebras. In M. Haveraaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specifications. 11th Workshop on Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 379–393. Springer Verlag, 1996.

[56] T. Mossakowski. Sublanguages of CASL. Note L-7, in [21], Dec. 1997.

[57] T. Mossakowski. Colimits of order-sorted specifications. In F. Parisi Presicce, editor, *Recent trends in algebraic development techniques. Proc. 12th International Workshop*, volume 1376 of *Lecture Notes in Computer Science*, pages 316–332. Springer, 1998.

[58] T. Mossakowski. CASL: From semantics to tools. In S. Graf and M. Schwartzbach, editors, *TACAS 2000*, volume 1785 of *Lecture Notes in Computer Science*, pages 93–108. Springer-Verlag, 2000.

[59] T. Mossakowski. Specification in an arbitrary institution with symbols. In C. Choppy, D. Bert, and P. Mosses, editors, *Recent Trends in Algebraic Development Techniques, 14th International Workshop, WADT'99, Bonas, France*, volume 1827 of *Lecture Notes in Computer Science*, pages 252–270. Springer-Verlag, 2000.

[60] T. Mossakowski, A. Haxthausen, and B. Krieg-Brückner. Subsorted partial higher-order logic as an extension of CASL. In C. Choppy, D. Bert, and P. Mosses, editors, *Recent Trends in Algebraic Development Techniques, 14th International Workshop, WADT'99, Bonas, France*, volume 1827 of *Lecture Notes in Computer Science*, pages 126–145. Springer-Verlag, 2000.

[61] T. Mossakowski, Kolyang, and B. Krieg-Brückner. Static semantic analysis and theorem proving for CASL. In F. Parisi Presicce, editor, *Recent trends in algebraic development techniques. Proc. 12th International Workshop*, volume 1376 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 1998.

[62] T. Mossakowski, A. Tarlecki, and W. Pawłowski. Combining and representing logical systems using model-theoretic parchments. In F. Parisi Presicce, editor, *Recent trends in algebraic development techniques. Proc. 12th International Workshop*, volume 1376 of *Lecture Notes in Computer Science*, pages 349–364. Springer, 1998.

[63] P. D. Mosses. Unified algebras and institutions. Proceedings of the 4th Annual IEEE Symposium on Logic in Computer Science, pages 304–312. 1989.

[64] P. D. Mosses. CASL for ASF+SDF users. In *ASF+SDF '97, Proc. 2nd Intl. Workshop on the Theory and Practice of Algebraic Specifications*, volume ASFSDF-97 of *Electronic Workshops in Computing*. British Computer Society, `http://www.ewic.org.uk/ewic/workshop/list.cfm`, 1997.

[65] P. D. Mosses. CoFI: The Common Framework Initiative for Algebraic Specification and Development. In *TAPSOFT '97, Proc. Intl. Symp. on Theory and Practice of Software Development*, volume 1214 of *LNCS*, pages 115–137. Springer-Verlag, 1997.

[66] P. D. Mosses. CASL for CafeOBJ users. In K. Futatsugi, A. T. Nakagawa, and T. Tamai, editors, *CAFE: An Industrial-Strength Algebraic Formal Method*, chapter 6, pages 121–144. Elsevier, 2000.

[67] P. Padawitz. *Computing in Horn Clause Theories*. Springer Verlag, Heidelberg, 1988.

[68] G. Reggio, E. Astesiano, and C. Choppy. CASL-LTL - a CASL extension for dynamic reactive systems - summary. Technical Report of DISI - Università di Genova, DISI-TR-99-34, Italy, 2000.

[69] G. Reggio and L. Repetto. CASL-CHART: a combination of statecharts and of the algebraic specification language CASL. In *Proc. AMAST 2000*, volume 1816 of *Lecture Notes in Computer Science*. Springer Verlag, 2000.

[70] H. Reichel. *Initial Computability, Algebraic Specifications and Partial Algebras*. Oxford Science Publications, 1987.

[71] D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.

[72] L. Schröder and T. Mossakowski. HasCASL: algebraic specification of Haskell programs. Submitted.

[73] A. Tarlecki. On the existence of free models in abstract algebraic institutions. *Theoretical Computer Science*, 37:269–304, 1985.

[74] A. Tarlecki. Moving between logical systems. In M. Haveraaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specifications. 11th Workshop on Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 478–502. Springer Verlag, 1996.

[75] A. Tarlecki. Towards heterogeneous specifications. In D. Gabbay and M. d. Rijke, editors, *Frontiers of Combining Systems 2, 1998*, Studies in Logic and Computation, pages 337–360. Research Studies Press, 2000.

[76] J. W. Thatcher, E. G. Wagner, and J. B. Wright. Specification of abstract data types using conditional axioms. Technical Report RC 6214, IBM Yorktown Heights, 1981.

[77] H. Yan. *Theory and Implementation of Sort Constraints for Order Sorted Algebra*. PhD thesis, Oxford University, 1993.