

# Foundations of Heterogeneous Specification

Till Mossakowski

BISS, Dept. of Computer Science, University of Bremen

**Abstract.** We provide a semantic basis for heterogeneous specifications that not only involve different logics, but also different kinds of translations between these. We show that Grothendieck institutions based on spans of (co)morphisms can serve as a unifying framework providing a simple but powerful semantics for heterogeneous specification.

## 1 Introduction

For the specification of large software systems, heterogeneous multi-logic specifications are needed, since complex problems have different aspects that are best specified in different logics. A combination of all the used logics would become too complex in many cases. Moreover, using heterogeneous specifications, different approaches being developed at different sites can be related, i.e. there is a formal interoperability among languages and tools. In many cases, specialized languages and tools have their strengths in particular aspects. Using heterogeneous specification, these strengths can be combined with comparably small effort.

The most prominent approach to heterogeneous specification is CafeOBJ with its cube of eight logics and twelve projections (formalized as *institution morphisms*) among them [13], and having a semantics based on Diaconescu's notion of Grothendieck institution [12]. However, this approach has a limitation: only one type of translation between institution is used, namely institution morphisms. Tarlecki [47] is more general, he introduces a whole bunch of heterogeneous constructs for different kinds of translations. However, only one kind of translation can be used at a time. The goal of the present work is to overcome these limitations while simultaneously staying as simple as possible.

## 2 Institutions and Their (Co)Morphisms

Following [21], we formalize logics as institutions.

**Definition 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  $\mathbf{Sen}^I: \mathbf{Sign}^I \rightarrow \mathbf{Set}$  giving, for each signature  $\Sigma$ , the set of *sentences*  $\mathbf{Sen}^I(\Sigma)$ , and for each signature morphism  $\sigma: \Sigma \rightarrow \Sigma'$ , the *sentence translation map*  $\mathbf{Sen}^I(\sigma): \mathbf{Sen}^I(\Sigma) \rightarrow \mathbf{Sen}^I(\Sigma')$ , where often  $\mathbf{Sen}^I(\sigma)(\varphi)$  is written as  $\sigma(\varphi)$ ,

- a functor  $\mathbf{Mod}^I: (\mathbf{Sign}^I)^{op} \rightarrow \mathcal{CAT}^1$  giving, for each signature  $\Sigma$ , the category of *models*  $\mathbf{Mod}^I(\Sigma)$ , and for each signature morphism  $\sigma: \Sigma \rightarrow \Sigma'$ , the *reduct functor*  $\mathbf{Mod}^I(\sigma): \mathbf{Mod}^I(\Sigma') \rightarrow \mathbf{Mod}^I(\Sigma)$ , where often  $\mathbf{Mod}^I(\sigma)(M')$  is written as  $M'|_\sigma$  (the  $\sigma$ -reduct of  $M'$ ),
- a satisfaction relation  $\models_\Sigma^I \subseteq |\mathbf{Mod}^I(\Sigma)| \times \mathbf{Sen}^I(\Sigma)$  for each  $\Sigma \in \mathbf{Sign}^I$ ,

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

$$M' \models_{\Sigma'}^I \sigma(\varphi) \Leftrightarrow M'|_\sigma \models_\Sigma^I \varphi$$

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

The notion of institutions gains much of its importance by the fact that several languages for modularizing specifications have been developed in a completely institution independent way [42, 16, 14, 22, 15, 34], one of which also has been extended to the heterogeneous case [47]. Most of their constructs can be translated into the formalism of *development graphs* introduced below, which hence can be seen as a core formalism for structured and heterogeneous theorem proving. For the language CASL, such a translation has been laid out explicitly in [4].

**Definition 2.** Given an arbitrary but fixed institution  $I$ , a *development graph* over  $I$  is an acyclic directed graph  $\mathcal{S} = \langle \mathcal{N}, \mathcal{L} \rangle$ .

$\mathcal{N}$  is a set of nodes. Each node  $N \in \mathcal{N}$  is a tuple  $(\Sigma^N, \Gamma^N)$  such that  $\Sigma^N \in \mathbf{Sign}^I$  is a signature and  $\Gamma^N \subseteq \mathbf{Sen}^I(\Sigma^N)$  is the set of *local axioms* of  $N$ .

$\mathcal{L}$  is a set of directed links, so-called *definition links*, between elements of  $\mathcal{N}$ . Each definition link from a node  $M$  to a node  $N$  is either

- *global* (denoted  $M \xrightarrow{\sigma} N$ ), annotated with a signature morphism  $\sigma: \Sigma^M \rightarrow \Sigma^N \in \mathbf{Sign}^I$ , or
- *hiding* (denoted  $M \xrightarrow[h]{\sigma} N$ ), annotated with a signature morphism  $\sigma: \Sigma^N \rightarrow \Sigma^M \in \mathbf{Sign}^I$  *going against the direction of the link*. Typically,  $\sigma$  will be an inclusion, and the symbols of  $\Sigma^M$  not in  $\Sigma^N$  will be hidden.

What is the meaning of such development graphs? Development graphs without hiding have a theory-level semantics, see [4]. For development graphs with hiding, a model-level semantics seems to be more appropriate:

**Definition 3.** Given a node  $N \in \mathcal{N}$ , its associated class  $\mathbf{Mod}_{\mathcal{S}}(N)^2$  of models (or  $N$ -models for short) consists of those  $\Sigma^N$ -models  $n$  for which

- $n$  satisfies the local axioms  $\Gamma^N$ ,
- for each  $K \xrightarrow{\sigma} N \in \mathcal{S}$ ,  $n|_\sigma$  is a  $K$ -model, and
- for each  $K \xrightarrow[h]{\sigma} N \in \mathcal{S}$ ,  $n$  has a  $\sigma$ -expansion  $k$  (i.e.  $k|_\sigma = n$ ) which is a  $K$ -model.

<sup>1</sup>  $\mathcal{CAT}$  be the (quasi-)category of categories and functors.

<sup>2</sup>  $\mathbf{Mod}_{\mathcal{S}}$  is not to be confused with the model functor  $\mathbf{Mod}$  of the institution.

Complementary to definition and hiding links, which *define* the theories of related nodes, we introduce the notion of a *theorem link* with the help of which we are able to *postulate* relations between different theories. Global theorem links<sup>3</sup> (denoted by  $N - \overset{\sigma}{\succ} M$ , where  $\sigma: \Sigma^N \longrightarrow \Sigma^M$ ) are the central data structure to represent proof obligations arising in formal developments.

**Definition 4.** Let  $\mathcal{S}$  be a development graph.  $\mathcal{S}$  *implies* a global theorem link  $N - \overset{\sigma}{\succ} M$  (denoted  $\mathcal{S} \models N - \overset{\sigma}{\succ} M$ ), iff for all  $m \in \mathbf{Mod}_{\mathcal{S}}(M)$ ,  $m|_{\sigma} \in \mathbf{Mod}_{\mathcal{S}}(N)$ .

We now come to the task of relating different institutions. Institution *morphisms* [21] relate two given institutions. A typical situation is that an institution morphism expresses the fact that a “larger” institution *is built upon* a “smaller” institution by *projecting* the “larger” institution onto the “smaller” one.

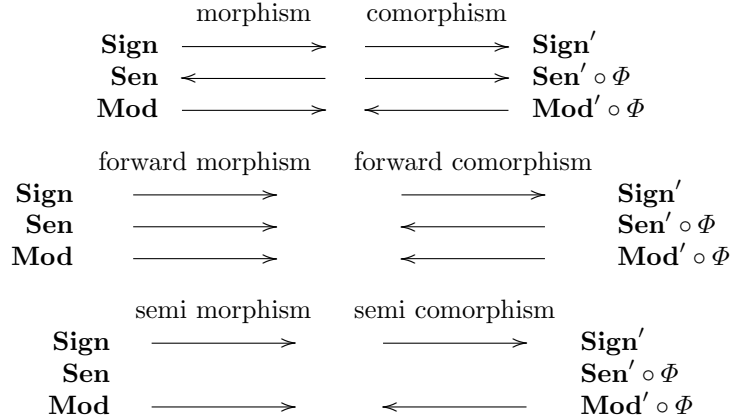
Given institutions  $I$  and  $J$ , an *institution morphism* [21]  $\mu = (\Phi, \alpha, \beta): I \longrightarrow J$  consists of

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

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

$$M \models_{\Sigma}^I \alpha_{\Sigma}(\varphi') \Leftrightarrow \beta_{\Sigma}(M) \models_{\Phi(\Sigma)}^J \varphi'$$

The notion of institution morphism can be varied in several ways by changing the directions of the arrows or even, in the case of semi-morphisms, omitting the arrows [20, 46]:



The respective satisfaction conditions are quite obvious (note that for semi-(co)morphisms, none is required).

<sup>3</sup> There are also local and hiding theorem links, which are omitted here for simplicity.

Finally, each type of morphism also comes in a *simple theoroidal* variant [20], meaning that signatures may be mapped to theories. Following [26], the category **Th** of theories has as objects theories, i.e. signatures plus sets of axioms. A theory morphism is a signature morphism mapping axioms to logical consequences. Let  $Sig: \mathbf{Th} \rightarrow \mathbf{Sign}$  be the functor forgetting axioms, and  $\iota: \mathbf{Sign} \rightarrow \mathbf{Th}$  denote the obvious inclusion, which is a right inverse to  $Sig$ .

Again following [26], a theoroidal comorphism  $\mu = (\Phi, \alpha, \beta): I \rightarrow J$  is said to be a *subinstitution comorphism* (and  $I$  is said to be a subinstitution of  $J$ ) if  $\Phi$  is an embedding of categories,  $\alpha$  is a pointwise injection, and  $\beta$  is a natural isomorphism.

In the literature, a whole bunch of different types of translations has been used. The following table partitions them by some informal classification scheme (a “th” stands for the simple theoroidal case, “semi” denotes semi-morphisms, and an “x” stand for folklore knowledge or trivialities):

	(semi) morphism	(theoroidal) comorphism	(theoroidal) forward morphism	forward comor- phism
Inclusion	[41]	[2, 26]	x	x
Coding	$([41, 1])^4$	th [3, 8, 9, 24] [25–28, 31, 33, 45]	th [5, 48], $([39, 40, 44])^5$	
Projection	[2, 41, 13]			
Feature interaction	[30]	th		
Implementation	semi [43, 46, 41]	x	[48]	

### 3 Heterogeneous Specification

One typical scenario (cf. e.g. [19, 18]) of heterogeneity arises in the specification of reactive systems: some equational or first-order logic is used to specify the data (here, lists over arbitrary elements), some process algebra (here, CSP) is used to describe the system (here, a buffer implemented as a list), and some temporal logic is used to state fairness or eventuality properties that go beyond the expressiveness of the process algebra (here, we express the fairness property that the buffer cannot read infinitely often without writing). A corresponding heterogeneous specification (using the structuring constructs of CASL) is given in Fig. 1, the corresponding development graph in Fig. 2. Here,  $(List, Ax)$  is a specification of lists,  $Buf$  is the buffer process,  $List'$  is the signature resulting from the translation to  $CFOL=LTL$ , and  $Fair$  is the fairness axiom.

Actually, one should add that the process  $Buf$  does not meet the fairness constraint, since it can read infinitely often without ever writing. However, a

<sup>4</sup> It is not entirely clear whether these should be really called encodings, since — unlike the other codings in this row — it is not clear that they are suitable for re-use of theorem provers.

<sup>5</sup> Salibra and Scollo introduce a relaxed kind of forward morphism mapping models to sets of models.

```

logic CSP-CFOL=
spec BUFFER =
  data LIST
  channels read, write : Elem
  process let Buf(l : List[Elem]) =
    read?x → Buf(cons(x, nil))
    □ if l = nil then STOP
    else write!last(l) → Buf(rest(l))
    in Buf(nil)
  with logic → CFOL=-LTL
  then %implies
  ∀x : ds . in_any_case(x, always eventually label_cond(y . fst(y) = write))
  %%      Roughly corresponds to AGF fst(label) = write in CTL*
end

```

**Fig. 1.** A sample heterogeneous specification.

simplicistic buffer such as

$$Copy = read?x \rightarrow write!x \rightarrow Copy$$

satisfies the fairness constraint, and so does a buffer using bounded lists.

$$\begin{array}{ccc}
 (List, Ax) \in CFOL^= & \xrightarrow{pr} & (List, \{Buf\}) \in \text{CSP-CFOL}^= \\
 & & \downarrow | \text{toLTL} \\
 & & (List', \{Fair\}) \in CFOL^=-LTL
 \end{array}$$

**Fig. 2.** An informal sample heterogeneous development graph

We now briefly introduce several institutions involved:

$FOL^=$  is *many-sorted first-order logic with equality*. Signatures consist of a set of sorts, a set of function symbols and a set of predicate symbols (each symbol coming with a string of argument sorts and, for function symbols, a result sort). Signature morphisms map the three components in a compatible way. Models are first order structures, and sentences are the usual first-order sentences built from equations, predicate applications and logical connectives and quantifiers  $\forall, \exists$ . Sentence translations and model reducts are quite straightforward. Satisfaction is defined inductively in the usual way. A detailed description of this institution can be found in [21].

$CFOL^=$  adds new sentences, namely *sort generation generation constraints*, to  $FOL^=$ . These express that a particular set of sorts is generated by terms built from a particular set of operations (and possibly variables valued with

values from other sorts). This allows specifying inductive datatypes like lists. Details can be found in the semantics of CASL [11, 31]. Actually,  $CFOL^=$  is a substitution of CASL. CASL additionally admits the use of subsorts and partiality, but we omit these here for simplicity.

CSP- $CFOL^=$  (actually a substitution of CSP-CASL [36, 37]) combines  $CFOL^=$  with the process algebra  $CSP$ . Signatures and signature morphisms are those from  $CFOL^=$ , but restricted to signature morphisms that are injective on sorts.

There are several notions of model for CSP- $CFOL^=$ . We here choose the most informative one, based on labeled transition systems (LTS) [38]. Thus, models are  $CFOL^=$ -models, possibly equipped with an LTS being labeled in the disjoint union of all carriers. On the  $CFOL^=$ -part, reducts are as in  $CFOL^=$ . If a model is not equipped with an LTS, neither is its reduct. If a model is equipped with an LTS, and the LTS is labeled only with labels from carriers of the  $CFOL^=$ -reduct, it is quite straightforward to construe the LTS as an LTS for the  $CFOL^=$ -reduct (injectivity of signature morphisms on sorts ensures that carrier sets are not doubled). Otherwise, the LTS is deleted.

Sentences are either  $CFOL^=$  sentences, or  $CSP$  process terms [23, 38] involving  $CFOL^=$ -terms in place of alphabet letters. Sentence translation is straightforward.

Satisfaction for  $CFOL^=$  sentences is as in  $CFOL^=$ . A  $CSP$  process term  $P$  is evaluated using the  $CFOL^=$ -part a model  $M$ , leading to an LTS  $L$  with labels in the disjoint union of all carriers. Now  $M$  satisfies  $P$  iff  $M$  is equipped with  $L$ . Details can be found in [36, 37].

$CFOL^=$ -LTL (actually a substitution of CASL-LTL [35]) combines  $CFOL^=$  with the computation tree logic CTL\* [17]<sup>6</sup>.

Signatures are  $CFOL^=$ -signatures with

- a distinguished set  $DS$  of *dynamic sorts*,
- an injective assignment of *label sorts*  $Label\_ds$  (outside  $DS$ ) to dynamic sorts  $ds$ , such that
- there exists a transition predicate  $-- \xrightarrow{\quad} -- : ds \times Label\_ds \times ds$  for each dynamic sort  $ds$ .

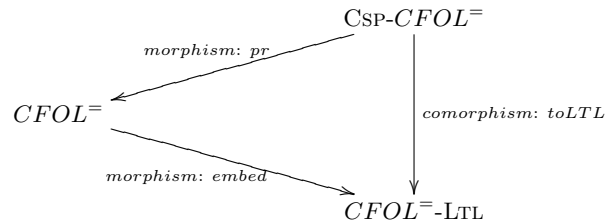
Signature morphisms are  $CFOL^=$ -signature morphisms preserving the extra structure on the nose (the latter is also called the *dynamic part* of the signature).

Models and model reducts are inherited from  $CFOL^=$ . The presence of the transition predicates means that each dynamic sort is interpreted as an LTS. Sentences are either first-order sentences, or CTL\* formulae anchored by the elements of dynamic sorts. Satisfaction is that of CTL\*. Details can be found in [35].

We will use the notation  $\mathbf{Sign}^{CFOL^=}$  etc. to denote the individual components of these institutions.

<sup>6</sup> Actually, the “LTL” in CASL-LTL is a bit misleading. It does not stand for linear temporal logic, as one might expect, but for labeled transition logic.

Among these institutions, we now introduce some morphisms and comorphisms (cf. Fig. 3):



**Fig. 3.** A (non-commutative) diagram of institutions and (co)morphisms.

- A morphism  $pr: \text{CSP-CFOL}^= \rightarrow \text{CFOL}^=$ . At the signature and sentence level, it is the obvious inclusion. For models, just the LTS (if present) is forgotten.
- A theoroidal semi-comorphism  $toLTL: \text{CSP-CFOL}^= \rightarrow \text{CFOL}^=-LTL$ . A signature is extended by a dynamic sort  $ds$ , a label sort  $Label_{ds}$  and a transition predicate, and injection operations  $inj: s \rightarrow Label_{ds}$  for each sort  $s$ . These are axiomatized to be injective and to jointly generate  $Label_{ds}$ . Signature morphisms are just extended to map the extra structure on the nose in the obvious way. (Actually, the translation of signature morphisms is only defined for identity signature morphisms; hence we have a *partial* semi-comorphism only. We will not formalize this here, since Section 6 offers a simple solution.)  
A  $\text{CFOL}^=-LTL$ -model is translated to a  $\text{CSP-CFOL}^=-$ -model by forgetting the interpretation of the dynamic part to get a  $\text{CFOL}^=-$ -model, and equipping it with the LTS determined by the interpretation of the transition relation.
- A comorphism  $embed: \text{CFOL}^= \rightarrow \text{CFOL}^=-LTL$ . This is just the obvious substitution comorphism.

This shows that practical examples may involve several types of translation between institutions. One might argue that one could try to modify the above introduced institutions in such a way that only one type of translation would be needed. However, the institutions are taken from the literature<sup>7</sup>, and it would require research effort (and in some cases tool development) in order to change them.

<sup>7</sup> In the case of  $\text{CSP-CFOL}^=$ , the present author has worked on the formalization as an institution [37], but not on the design [36]. There seems to be no alternative formulation of  $\text{CSP-CFOL}^=$  as an institution simplifying the above picture. Moreover, the problems to be solved when formalizing the quite popular language LOTOS [7] as an institution should be similar to those of  $\text{CSP-CFOL}^=$ .

## 4 The Bi-Grothendieck Institution

How can we give a precise meaning to the development graph involving several kinds of translations between institutions in Fig. 2?

Tarlecki's [47] approach to heterogeneous specification is to introduce a new heterogeneous language construct for each of the various kinds of translations between institutions. This would correspond to adding new types of definition, hiding and theorem links for each of the translation kinds. However, it is unclear how several kinds of translations will interact, e.g. with respect to amalgamation and interpolation properties, which are important for structured proof systems [6, 32].

Even if one does not use heterogeneous specifications *simultaneously* involving different kinds of translations, a unifying framework is highly desirable in order not to have to switch to a different tool when considering a heterogeneous specification with a different type of translation.

A good way to deal with these problems is to flatten the graph of institutions and translations, as it is done with Diaconescu's Grothendieck institution [12]. We here recall the Grothendieck institution for the comorphism-based case [29]:

**Definition 5.** An *indexed coinstitution* is a functor  $\mathcal{I}: \mathit{Ind}^{op} \rightarrow \mathbf{CoIns}$  into the category  $\mathbf{CoIns}$  of institutions and institution comorphisms<sup>8</sup>. A *discrete* indexed coinstitution is one with  $\mathit{Ind}$  discrete.  $\square$

The basic idea of the Grothendieck institution is that all signatures of all institutions are put side by side, and a signature morphism in this large realm of signatures consists of an intra-institution signature morphism plus an inter-institution translation (along some institution comorphism). The other components are then defined in a straightforward way.

**Definition 6.** Given an *indexed coinstitution*  $\mathcal{I}: \mathit{Ind}^{op} \rightarrow \mathbf{CoIns}$ , define the *Grothendieck institution*  $\mathcal{I}^\#$  as follows:

- signatures in  $\mathcal{I}^\#$  are pairs  $(\Sigma, i)$ , where  $i \in |\mathit{Ind}|$  and  $\Sigma$  a signature in the institution  $\mathcal{I}(i)$ ,
- signature morphisms  $(\sigma, e): (\Sigma_1, i) \rightarrow (\Sigma_2, j)$  consist of a morphism  $e: j \rightarrow i \in \mathit{Ind}$  and a signature morphism  $\sigma: \Phi^{\mathcal{I}(e)}(\Sigma_1) \rightarrow \Sigma_2$  (here,  $\mathcal{I}(e): \mathcal{I}(i) \rightarrow \mathcal{I}(j)$  is the institution comorphism corresponding to the arrow  $e: j \rightarrow i$  in the indexed coinstitution, and  $\Phi^{\mathcal{I}(e)}$  is its signature translation component),
- the  $(\Sigma, i)$ -sentences are the  $\Sigma$ -sentences in  $\mathcal{I}(i)$ , and sentence translation along  $(\sigma, e)$  is the composition of sentence translation along  $\sigma$  with sentence translation along  $\mathcal{I}(e)$ ,
- the  $(\Sigma, i)$ -models are the  $\Sigma$ -models in  $\mathcal{I}(i)$ , and model reduction along  $(\sigma, e)$  is the composition of model translation along  $\mathcal{I}(e)$  with model reduction along  $\sigma$ , and

<sup>8</sup> Indeed, the name is justified by the fact that the category of institutions and institution comorphisms is isomorphic to the category of coinstitution and coinstitution morphisms. A coinstitution is an institution with model translations covariant to signature morphisms, while sentence translations are contravariant.

– satisfaction w.r.t.  $(\Sigma, i)$  is satisfaction w.r.t.  $\Sigma$  in  $\mathcal{I}(i)$ . □

A Grothendieck institution for an indexed institution  $\mathcal{I}: \text{Ind}^{op} \rightarrow \mathbf{Ins}$  can be defined similarly [12], it will also be denoted by  $\mathcal{I}^\#$ . By contrast, the Grothendieck construction does not obviously generalize to diagrams consisting of forward or semi (co)morphisms, because of the lacking (contravariance between model and) sentence translation. Let us therefore for a moment concentrate on morphisms and comorphisms only.

We consider heterogeneous specification over a set of institutions, a set of morphisms and a set of comorphisms. This is formalized as an indexed institution  $\mathcal{I}_m$  (collecting the morphisms) together with an indexed coinstitution  $\mathcal{I}_c$  (collecting the comorphisms), both with the same underlying set of institutions, regarded as a discrete indexed institution  $\mathcal{I}_0$ .

**Definition 7.** Let  $(\mathcal{I}_m, \mathcal{I}_c, \mathcal{I}_0)$ , with  $\mathcal{I}_m$  an indexed institution,  $\mathcal{I}_c$  an indexed coinstitution and  $\mathcal{I}_0$  a discrete indexed institution be given, such that  $|\text{Ind}_m| = |\text{Ind}_c| = |\text{Ind}_0|$ , and  $\mathcal{I}_m, \mathcal{I}_c$  and  $\mathcal{I}_0$  agree on these.

Then we form the Grothendieck institutions  $\mathcal{I}_0^\#, \mathcal{I}_m^\#$  and  $\mathcal{I}_c^\#$ . Since  $\mathcal{I}_0^\#$  obviously is included in  $\mathcal{I}_m^\#$  and  $\mathcal{I}_c^\#$  via a (co)morphism, we can take the pushout

$$\begin{array}{ccc} \mathcal{I}_0^\# & \longrightarrow & \mathcal{I}_m^\# \\ \downarrow & & \downarrow \\ \mathcal{I}_c^\# & \dashrightarrow & \mathcal{J} \end{array}$$

in the category of institutions and institution morphisms (or comorphisms, this would make no difference here). The pushout in either category exists by results of [20]. By abuse of notation, we will denote the pushout  $\mathcal{J}$  by  $(\mathcal{I}_m, \mathcal{I}_c)^\#$ . It will be called the *Bi-Grothendieck institution*.

The heterogeneous development graph in Fig. 2 can now formally be understood as a development graph over the Bi-Grothendieck institution.

## 5 Inducibility

The Bi-Grothendieck institution is quite complex, and it is not immediately clear how to obtain e.g. proof support for it. It is therefore tempting to try to reduce the complexity of this construction by mapping morphisms to comorphisms or vice versa. This can be done by weakening the adjunction between morphisms and comorphisms introduced in [2]:

**Definition 8.** Given an institution comorphism  $\rho = (\Phi, \alpha, \beta): I \rightarrow J$ , a functor  $\Psi: \mathbf{Sign}^J \rightarrow \mathbf{Sign}^I$  and a natural transformation  $\varepsilon: \Phi \circ \Psi \rightarrow Id$ , we say that  $\rho$   $\varepsilon$ -induces the institution morphism  $\mu = (\Psi, \bar{\alpha}, \bar{\beta}): J \rightarrow I$  given by

$$\begin{aligned} \bar{\alpha} &= (\mathbf{Sen}^J \cdot \varepsilon) \circ (\alpha \cdot \Psi) \\ \bar{\beta} &= (\beta \cdot \Psi^{op}) \circ (\mathbf{Mod}^J \cdot \varepsilon^{op}) \end{aligned}$$

A morphism that is  $\varepsilon$ -induced by some comorphism is called *inducible*.

Dually, given an institution morphism  $\mu = (\Psi, \bar{\alpha}, \bar{\beta}): J \longrightarrow I$ , a functor  $\Phi: \mathbf{Sign}^I \longrightarrow \mathbf{Sign}^J$  and a natural transformation  $\eta: Id \longrightarrow \Psi \circ \Phi$ , we say that  $\mu$   $\eta$ -induces the institution comorphism  $\rho = (\Phi, \alpha, \beta): I \longrightarrow J$  given by

$$\begin{aligned}\alpha &= (\bar{\alpha} \cdot \Phi) \circ (\mathbf{Sen}^I \cdot \eta) \\ \beta &= (\mathbf{Mod}^I \cdot \eta^{op}) \circ (\bar{\alpha} \cdot \Phi)\end{aligned}$$

A comorphism that is  $\eta$ -induced by some morphism is called *inducible*. Moreover, it is straightforward to extend inducibility to the simple theoroidal case. Here,

$\mathbf{Sign}^I \xrightarrow{\Phi} \mathbf{Sign}^J$  has to be replaced with  $\mathbf{Sign} \xrightarrow{\Phi} \mathbf{Th}^J \xrightarrow{Sig} \mathbf{Sign}^J$ , leading to the equations

$$\begin{aligned}\alpha &= (\bar{\alpha} \cdot Sig \cdot \Phi) \circ (\mathbf{Sen}^I \cdot \eta) \\ \beta &= (\mathbf{Mod}^I \cdot \eta^{op}) \circ (\bar{\beta} \cdot Sig \cdot \Phi)\end{aligned}$$

Furthermore, inducibility also extend to semi-(co)morphisms. □

With this, we can easily obtain the desired reduction:

**Theorem 9.** Let  $(\mathcal{I}_m, \mathcal{I}_c, \mathcal{I}_0)$  as in Definition 7 be given.

If each morphism in  $\mathcal{I}_m$  is  $\varepsilon$ -induced by some comorphism in  $\mathcal{I}_c$ , then there is a retraction of  $(\mathcal{I}_m, \mathcal{I}_c)^\#$  onto  $\mathcal{I}_c^\#$ .

Dually, if each comorphism in  $\mathcal{I}_c$  is  $\eta$ -induced by some morphism in  $\mathcal{I}_m$ , then there is a retraction of  $(\mathcal{I}_m, \mathcal{I}_c)^\#$  onto  $\mathcal{I}_m^\#$ .

*Proof.* Consider the pushout construction in Definition 7. Clearly,  $\mathcal{I}_m^\#, \mathcal{I}_c^\#$  and  $\mathcal{I}_0^\#$  all have the same object class. Moreover, since  $\mathcal{I}_0$  is discrete, the signature morphisms in  $\mathcal{I}_0^\#$  are basically those of the individual institutions. With this, it is easy to see that the signature morphisms in  $(\mathcal{I}_m, \mathcal{I}_c)^\#$  are paths of morphisms coming from  $\mathcal{I}_m^\#$  and  $\mathcal{I}_c^\#$  in an alternating way.

The retraction of  $(\mathcal{I}_m, \mathcal{I}_c)^\#$  onto  $\mathcal{I}_c^\#$  (having the obvious inclusion as right inverse) is therefore given by the identity for the objects, while for a path of alternating morphisms, each morphism

$$(\Sigma_1 \xrightarrow{\sigma} \Psi^d(\Sigma_2), i \xrightarrow{d} j): (\Sigma_1, i) \longrightarrow (\Sigma_2, j)$$

from  $\mathcal{I}_m^\#$  is replaced with

$$(\Phi^e(\Sigma_1) \xrightarrow{\Phi^e(\sigma)} \Phi^e(\Psi^d(\Sigma_2)) \xrightarrow{\varepsilon_{\Sigma_2}} \Sigma_2, j \xrightarrow{e} i),$$

where  $e$  is the index of the comorphism inducing  $\mathcal{I}_m(d)$ , and  $\varepsilon$  the corresponding natural transformation. Since all the resulting morphisms live in  $\mathcal{I}_c^\#$ , they can be composed to a single morphism.

The other statement follows by a dual argument. □

However, unfortunately there are practically relevant situations where this is not applicable.

**Proposition 10.** Neither the morphism  $pr: \text{CSP-}CFOL^= \longrightarrow CFOL^=$  nor the theoroidal semi-comorphism  $toLTL: \text{CSP-}CFOL^= \longrightarrow CFOL^=-LTL$  is inducible.

Proof. Assume that  $pr = (\Psi, \bar{\alpha}, \bar{\beta}): \text{CSP-}CFOL^= \longrightarrow CFOL^=$  is  $\varepsilon$ -induced by a comorphism  $\rho = (\Phi, \alpha, \beta): CFOL^= \longrightarrow \text{CSP-}CFOL^=$ . Let  $\Sigma_1$  consist of a sort  $s$  and  $\Sigma_2$  of sorts  $t$  and  $u$  (both seen as  $\text{CSP-}CFOL^=$ -signatures). Let  $\sigma: \Psi(\Sigma_2) \longrightarrow \Psi(\Sigma_1)$  map both  $t$  and  $u$  to  $s$  (recall that  $\Psi$  is just an inclusion). Now  $\bar{\beta}_{\Sigma_2}$  just forgets the optional LTS component, and hence is surjective. Since  $\bar{\beta}_{\Sigma_2} = \beta_{\Psi(\Sigma_2)} \circ \varepsilon_{\Sigma_2}$ ,  $\beta_{\Psi(\Sigma_2)}$  is surjective as well. Since all signature morphisms in  $\text{CSP-}CFOL^=$  are injective (and carriers are assumed to be non-empty), the corresponding reduct functors are easily seen to be surjective. Hence, the lower right path in the naturality diagram for  $\beta$

$$\begin{array}{ccc}
\mathbf{Mod}^{\text{CSP-}CFOL^=}(\Phi(\Psi(\Sigma_1))) & \xrightarrow{\beta_{\Psi(\Sigma_1)}} & \mathbf{Mod}^{CFOL^=}(\Psi(\Sigma_1)) \\
\downarrow \text{-}|_{\Phi(\sigma)} & & \downarrow \text{-}|_{\sigma} \\
\mathbf{Mod}^{\text{CSP-}CFOL^=}(\Phi(\Psi(\Sigma_2))) & \xrightarrow{\beta_{\Psi(\Sigma_2)}} & \mathbf{Mod}^{CFOL^=}(\Psi(\Sigma_2))
\end{array}$$

is surjective as well. Hence, also the upper left path must be surjective, and hence its second component  $\text{-}|_{\sigma}$ . But  $\text{-}|_{\sigma}$  just doubles the carrier set, and this is clearly not surjective.

The semi-comorphism  $toLTL$  is more precisely defined on the substitution  $\text{CSP-}CFOL^=-d$  consisting of identity signature morphisms only, i.e.  $toLTL = (\Phi, \beta): \text{CSP-}CFOL^=-d \longrightarrow CFOL^=-LTL$ . Assume that it is  $\eta$ -induced by a semi-morphism  $\mu = (\Psi, \bar{\beta}): CFOL^=-LTL \longrightarrow \text{CSP-}CFOL^=-d$ . Since all signature morphisms in  $\text{CSP-}CFOL^=-d$  are identities,  $\eta$  is the identity as well. Hence,  $\bar{\beta} \cdot \Phi = \beta$ , and one easily obtains a contradiction to the  $\bar{\beta}$ -naturality diagram for a signature morphism  $\sigma: \Phi(\Sigma_1) \longrightarrow \Phi(\Sigma_2)$  in  $CFOL^=-LTL$ . This proof relies on the severe restriction of the  $\text{CSP-}CFOL^=-d$  signature morphisms; however, also a proof not exploiting this is possible.  $\square$

## 6 Spans of Comorphisms

The method of the previous section to use inducibility to reduce the complexity of heterogeneous specifications involving different kinds of translations between institutions works for some cases, but the counterexamples of Proposition 10 have shown that the method is not general enough.

A more general idea is to express all the different kinds of translations as *spans* of morphisms or of comorphisms. The question is now whether to work

with morphisms or with comorphisms. Indeed, comorphisms interact with amalgamation properties in a much simpler way than morphisms do, see Propositions 3.5 and 3.6 of [29]. Amalgamation properties are important in many respects, e.g. for heterogeneous theorem proving [29]. Therefore, we work with spans of comorphisms. Nevertheless, the results presented below easily dualize to spans of morphisms.

$$\text{Each institution morphism } \mu: I \longrightarrow J = I \begin{array}{c} \xrightarrow{\Psi} \\ \xleftarrow{\alpha} \\ \xrightarrow{\beta} \end{array} J \text{ can be translated}$$

into a span  $I \xleftarrow{\mu^+} J \circ \Psi \xrightarrow{\mu^-} J$  of institution comorphisms as follows:

$$\begin{array}{ccccc} \mathbf{Sign}^I & \xleftarrow{id} & \mathbf{Sign}^I & \xrightarrow{\Psi} & \mathbf{Sign}^J \\ \mathbf{Sen}^I & \xleftarrow{\alpha} & \mathbf{Sen}^J \circ \Psi & \xrightarrow{id} & \mathbf{Sen}^J \circ \Psi \\ \mathbf{Mod}^I & \xrightarrow{\beta} & \mathbf{Mod}^J \circ \Psi & \xleftarrow{id} & \mathbf{Mod}^J \circ \Psi \end{array}$$

Here, the “middle” institution  $J \circ \Psi$  is the institution with signature category inherited from  $I$ , but sentences and models inherited from  $J$  via  $\Psi$ .

This span construction can also be lifted to the indexed level: Given an indexed institution  $\mathcal{I}_m$  and an indexed coinstitution  $\mathcal{I}_c$  (both over the same set of institutions in the sense of Definition 7), form an indexed coinstitution  $Span(\mathcal{I}_m, \mathcal{I}_c)$  as follows: the index category is obtained by freely adding pairs of formal morphisms to the index category of  $\mathcal{I}_c$ ; more precisely, one pair (obtained in a straightforward way) for each span of comorphisms corresponding to a morphism in  $\mathcal{I}_m$ .

Unfortunately, we cannot expect that Theorem 9 carries over to the present situation. But we have some weaker property that still is sufficiently strong for practical needs:

**Theorem 11.** Given an indexed institution  $\mathcal{I}_m$  and an indexed coinstitution  $\mathcal{I}_c$  (both over the same set of institutions in the sense of Definition 7), then each development graph over the Bi-Grothendieck institution  $(\mathcal{I}_m, \mathcal{I}_c)^\#$  can be translated into a development graph over the Grothendieck institution over the span-based indexed coinstitution  $Span(\mathcal{I}_m, \mathcal{I}_c)^\#$ , such that model categories are preserved.

*Proof.* As in the proof of Theorem 9, we rely on the fact that signature morphisms in the Bi-Grothendieck institution  $(\mathcal{I}_m, \mathcal{I}_c)^\#$  are paths of morphisms coming from  $\mathcal{I}_m^\#$  and  $\mathcal{I}_c^\#$  in an alternating way. A global definition link therefore has the form

$$M \xrightarrow{\langle (\sigma_1, e_1), (\sigma_2, d_2), \dots, (\sigma_n, e_n) \rangle} N ,$$

where the  $d_i$  are from  $\mathcal{I}_m$  and the  $e_i$  are from  $\mathcal{I}_c$  (with  $(\sigma_1, e_1)$ ,  $(\sigma_n, e_n)$  possibly not present). The definition link now is replaced by a sequence of definition and

hiding links:

$$M \xrightarrow{(\sigma_1, e_1)} M_1 \xrightarrow{(\sigma_2, id)} M_2 \xrightarrow[h]{(id, d_2^-)} M_3 \xrightarrow{(id, d_2^+)} M_4 \longrightarrow \dots \xrightarrow{(\sigma_n, e_n)} N$$

Here,  $d_2^-$  and  $d_2^+$  are the indices for the span of comorphisms associated to  $\mathcal{I}_m(d_2)$ , and  $M_1, \dots, M_4$  are new nodes with appropriate signatures and no local axioms. Of course, the path could also start and/or end with a  $d_i$  instead of an  $e_i$ , but this won't affect the general construction: each path element containing a  $d_i$  leads to a sequence of a definition link, a hiding link, and again a definition link, while path elements containing an  $e_i$  are just kept. The construction for theorem links is entirely analogous, except that only the last arrow in the sequence has to be a theorem link — the other ones must be definition links.

Let us now come to hiding links. The construction is very similar, so we restrict ourselves to the replacement of individual path elements of form  $(\sigma_i, d_i)$ . Such an element leads to

$$\dots \longrightarrow M_n \xrightarrow[h]{(id, d_i^+)} M_{n+1} \xrightarrow{(id, d_i^-)} M_{n+2} \xrightarrow[h]{(\sigma_i, id)} M_{n+3} \longrightarrow \dots$$

In comparison to the construction above, here the arrows are reversed, and definition and hiding links interchanged.

It is straightforward to see that the model class is left unchanged by these translations.

□

$$\begin{array}{ccc} & \xrightarrow{\Psi} & \\ \text{Consider now a semi morphism } I & & J. \text{ It can be translated into} \\ & \xrightarrow{\beta} & \end{array}$$

a span of comorphisms

$$\begin{array}{ccccc} \mathbf{Sign} & \xleftarrow{id} & \mathbf{Sign}^I & \xrightarrow{\Psi} & \mathbf{Sign}^J \\ \mathbf{Sen}^I & \xleftarrow{incl} & \emptyset & \xrightarrow{incl} & \mathbf{Sen}^J \circ \Psi \\ \mathbf{Mod}^I & \xrightarrow{\beta} & \mathbf{Mod}^J \circ \Psi & \xleftarrow{id} & \mathbf{Mod}^J \circ \Psi \end{array}$$

$$\begin{array}{ccc} & \xrightarrow{\Phi} & \\ \text{while a semi-comorphism } I & & J \text{ is translated into a span of co-} \\ & \xleftarrow{\beta} & \end{array}$$

morphisms

$$\begin{array}{ccccc} \mathbf{Sign} & \xleftarrow{id} & \mathbf{Sign}^I & \xrightarrow{\Phi} & \mathbf{Sign}^J \\ \mathbf{Sen}^I & \xleftarrow{incl} & \emptyset & \xrightarrow{incl} & \mathbf{Sen}^J \circ \Phi \\ \mathbf{Mod}^I & \xrightarrow{id} & \mathbf{Mod}^I & \xleftarrow{\beta} & \mathbf{Mod}^J \circ \Phi \end{array}$$

With this, we also can give a semantics to definition, hiding and theorem links involving semi-(co)morphisms. Partial (semi-)comorphisms and a restricted class of forward (co)morphisms can be covered as well.

Example 12. Extend the institutions and (co)morphisms introduced in Section 3 by the following ones:

- $CFOL^-inj$  is the restriction of  $CFOL^-$  to signature that are injective on sorts.
- $CSP-CFOL^-d-nosen$  is the restriction of  $CSP-CFOL^-$  to the empty set of sentences, for each signature, and to identity signature morphisms.
- The comorphism  $pr^-: CFOL^-inj \rightarrow CFOL^-$  is just the obvious substitution inclusion.
- The comorphism  $pr^+: CFOL^-inj \rightarrow CSP-CFOL^-$  behaves very similar to  $pr$ : At the signature level, it is the identity, at the sentence level, it is the obvious inclusion. For models, just the LTS (if present) is forgotten.

When applying the construction of Theorem 11 to (a formalized variant of) the development graph given in Fig. 2, we arrive at the development graph shown in Fig. 4 (for simplicity, we index the involved institutions and comorphisms by themselves here).

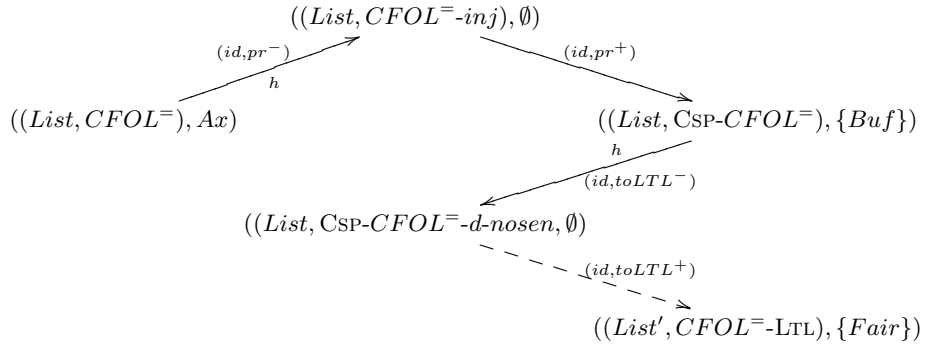


Fig. 4. The sample heterogeneous development graph with spans of comorphisms.

## 7 Conclusion

We have presented an example heterogeneous specification involving different kinds of translations between the involved institution. We have discussed several ways of giving a semantics to such specifications. The most promising way has turned out to use indexed coinstitutions and their Grothendieck construction

as a semantical foundation for heterogeneous specification, and express other types of translation by spans of comorphisms. Of course, the dual view (followed by CafeOBJ), which takes morphism-based Grothendieck institutions as foundation [12], also can be combined with the span approach. However, as shown in [29], the comorphism-based Grothendieck construction interacts more nicely with amalgamation. Hence, we stick to comorphisms here.

Our approach also implies that we can exploit techniques such as heterogeneous borrowing [30] and the heterogeneous proof calculus [29], which are based on comorphisms only, in a much wider context. Tool support for heterogeneous specifications and development graphs is under development in form of the heterogeneous tool set (hets). The latter provides an abstract programming interface for the implementable part of institutions and comorphisms. This will serve as a basis for implementing heterogeneous analysis and proof tools that are based on corresponding tools for the individual logics.

**Acknowledgments** Thanks to Andrzej Tarlecki, Joseph Goguen, Grigore Rosu, Serge Autexier and Dieter Hutter for useful cooperation and discussions, to Răzvan Diaconescu for inventing Grothendieck institutions, and to the anonymous referees for valuable criticism and suggestions. I would also like to thank the members of the Common Framework Initiative (CoFI).

This work has been supported by the *Deutsche Forschungsgemeinschaft* under Grant KR 1191/5-1.

## References

1. S. Alagi. Institutions: integrating objects, XML and databases. *Information and Software Technology*, 44:207–216, 2002.
2. M. Arrais and J. L. Fiadeiro. Unifying theories in different institutions. In M. Haverdaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specifications. 11th WADT*, LNCS 1130, pages 81–101. Springer Verlag, 1996.
3. E. Astesiano and M. Cerioli. Relationships between logical frameworks. In M. Bidoit and C. Choppy, editors, *Proc. 8th ADT workshop*, LNCS 655, pages 126–143. Springer Verlag, 1992.
4. S. Autexier, D. Hutter, H. Mantel, and A. Schairer. Towards an evolutionary formal software-development using CASL. In C. Choppy and D. Bert, editors, *Recent Trends in Algebraic Development Techniques, 14th International Workshop, WADT'99, Bonas, France*, LNCS 1827, pages 73–88. Springer-Verlag, 2000.
5. M. Bidoit and R. Hennicker. Using an institution encoding for proving consequences of structured COL-specifications. Talk at the WADT 2002, Frauenchiemsee.
6. T. Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286:197–245, 2002.
7. E. Brinksma, editor. *Information processing systems — open systems interconnection. LOTOS: a formal description technique based on the temporal ordering of observational behaviour*. 1988. International Standard ISO 8807.
8. M. Cerioli. *Relationships between Logical Formalisms*. PhD thesis, TD-4/93, Università di Pisa-Genova-Udine, 1993.

9. M. Cerioli and J. Meseguer. May I borrow your logic? (transporting logical structures along maps). *Theoretical Computer Science*, 173:311–347, 1997.
10. CoFI. The Common Framework Initiative for algebraic specification and development, electronic archives. Notes and Documents accessible from <http://www.cofi.info>.
11. CoFI Semantics Task Group. CASL – The CoFI Algebraic Specification Language – Semantics. Note S-9 (Documents/CASL/Semantics, version 0.96), in [10], July 1999.
12. R. Diaconescu. Grothendieck institutions. *Applied categorical structures*, 10:383–402, 2002.
13. R. Diaconescu and K. Futatsugi. Logical foundations of CafeOBJ. *Theoretical computer science*, 285:289–318, 2002.
14. 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.
15. F. Durán and J. Meseguer. Structured theories and institutions. In M. Hofmann, G. Rosolini, and D. Pavlovic, editors, *CTCS '99 Conference on Category Theory and Computer Science*, ENTCS 29. 1999.
16. H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2*. Springer Verlag, Heidelberg, 1990.
17. E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B*. Elsevier / MIT Press, 1990.
18. B. Ghribi and L. Logrippo. A validation environment for LOTOS. In *Protocol Specification, Testing and Verification*, pages 93–108, 1993.
19. P. Gibson, B. Mermet, and D. Mery. Feature interactions: A mixed semantic model approach. In *IWFM*, 1997.
20. J. Goguen and G. Rosu. Institution morphisms. *Formal aspects of computing*, 13:274–307, 2002.
21. 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.
22. J. A. Goguen and W. Tracz. An implementation-oriented semantics for module composition. In G. T. Leavens and M. Sitaraman, editors, *Foundations of Component-Based Systems*, chapter 11, pages 231–263. Cambridge University Press, New York, NY, 2000.
23. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
24. 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.
25. N. Martí-Oliet and J. Meseguer. From abstract data types to logical frameworks. LNCS 906, pages 48–80. Springer, 1995.
26. J. Meseguer. General logics. In *Logic Colloquium 87*, pages 275–329. North Holland, 1989.
27. J. Meseguer. Conditional rewriting as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–156, 1992.
28. T. Mossakowski. Equivalences among various logical frameworks of partial algebras. In H. K. Büning, editor, *Computer Science Logic. 9th Workshop, CSL'95.*, LNCS 1092, pages 403–433. Springer Verlag, 1996.
29. T. Mossakowski. Comorphism-based Grothendieck logics. In K. Diks and W. Rytter, editors, *Mathematical foundations of computer science*, LNCS 2420, pages 593–604. Springer, 2002.

30. T. Mossakowski. Heterogeneous development graphs and heterogeneous borrowing. In M. Nielsen and U. Engberg, editors, *Foundations of Software Science and Computation Structures*, LNCS 2303, pages 326–341. Springer-Verlag, 2002.
31. T. Mossakowski. Relating CASL with other specification languages: the institution level. *Theoretical Computer Science*, 286:367–475, 2002.
32. T. Mossakowski, S. Autexier, and D. Hutter. Extending development graphs with hiding. In H. Hußmann, editor, *Fundamental Approaches to Software Engineering*, LNCS 2029, pages 269–283. Springer-Verlag, 2001.
33. 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*, LNCS 1376, pages 333–348. Springer, 1998.
34. 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*, LNCS 1214, pages 115–137. Springer-Verlag, 1997.
35. 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.
36. M. Roggenbach. CSP-CASL – a new integration of process algebra and algebraic specification. Manuscript, Bremen, submitted for publication.
37. M. Roggenbach and T. Mossakowski. The CSP-CASL institution and its relation to temporal logic. Manuscript, University of Bremen.
38. A. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.
39. A. Salibra and G. Scollo. A soft stairway to institutions. In M. Bidoit and C. Choppy, editors, *Proc. 8th ADT workshop*, LNCS 655, pages 310–329. Springer Verlag, 1992.
40. A. Salibra and G. Scollo. Interpolation and compactness in categories of pre-institutions. *Mathematical Structures in Computer Science*, 6(3):261–286, June 1996.
41. D. Sannella and A. Tarlecki. *Working with multiple logical systems, In: Foundations of Algebraic Specifications and Formal Program Development*, chapter 10. Cambridge University Press, to appear. See <http://zls.mimuw.edu.pl/~tarlecki/book/index.html>.
42. D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.
43. D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: implementations revisited. *Acta Inf.*, 25:233–281, 1988.
44. G. Scollo. *On the engineering of logics*. PhD thesis, 1993. University of Twente, Enschede.
45. A. Tarlecki. Institution representation. draft note, 1987.
46. 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*, LNCS 1130, pages 478–502. Springer Verlag, 1996.
47. 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.
48. U. Wolter, K. Didrich, F. Cornelius, M. Klar, R. Wessäly, and H. Ehrig. How to cope with the spectrum of SPECTRUM. In M. Broy and S. Jähnichen, editors, *KORSO: Methods, Languages and Tools for the Construction of Correct Software*, LNCS 1009, pages 173–189. Springer-Verlag, New York, NY, 1995.