Heterogeneous Specification and the Heterogeneous Tool Set

Till Mossakowski

BISS, Dept. of Computer Science, University of Bremen

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. True logic combinations (in the sense of fibring [8], or colimits of logical systems [14, 17, 4, 3]) work well for certain classes of logics. However, the true combination approach reaches its limits when logics involving very different features (like modalities, higher-order polymorphism, and calculi for concurrent systems) shall be combined. In such cases, a true combination of all the used logics will quickly become too complex. Hence, heterogeneous specification provide a weaker form of logic combination (corresponding to weighted colimits), where basically the logics are put side by side, but can interact via logic translations.

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.

A general semantic framework for heterogeneous specification has been outlined in another paper in this volume [20]. The goal of the present work is to show how such a framework can be equipped with a proof calculus and tool support. Although some calculus for deriving theorems from a heterogeneous specification is already given in [20], some further work is required in order to obtain a calculus for proving refinements between heterogeneous specifications.

We show that Grothendieck institutions based on institution comorphisms can serve as a framework for developing such a proof calculus and building tools. In particular, we show how to extend the verification semantics given for structured specifications in [13] to the heterogeneous case. This semantics translates a heterogeneous specification into a kernel formalism called development graphs.

The heterogeneous tool set provides tool support for heterogeneous specification. Based on an object-oriented interface for institutions (using type classes in Haskell), it implements the Grothendieck institution and provides a heterogeneous parser, static analysis and proof support for heterogeneous specification. This is based on parsers, static analysers and proof support for the individual institutions, on the above mentioned heterogeneous verification semantics, and on a proof calculus for development graphs over the Grothendieck institution.

2 Institutions, Their (Co)Morphisms, and Heterogeneous Specifications

We only rather briefly recall the technical preliminaries, referring to [20] for more explanation.

Following [10], 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 $\mathbf{Sen}^I : \mathbf{Sign}^I \longrightarrow \mathbf{Set}$ 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 $\mathbf{Mod}^I : (\mathbf{Sign}^I)^{op} \longrightarrow \mathcal{CAT}$[1] 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 $M'|_\sigma$ (the σ-reduct of M'),
- 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 $\varphi \in \mathbf{Sen}^I(\Sigma)$. □

Given an institution, it is possible to define the semantics of structured specifications over it in an entirely institution independent way [16]:

***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:

$$\begin{aligned} Sig[\langle \Sigma, \Gamma \rangle] &:= \Sigma \\ \mathbf{Mod}[\langle \Sigma, \Gamma \rangle] &:= \{M \in \mathbf{Mod}(\Sigma) \mid M \models \Gamma\} \end{aligned}$$

***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:

$$\begin{aligned} Sig[SP_1 \cup SP_2] &:= \Sigma \\ \mathbf{Mod}[SP_1 \cup SP_2] &:= \mathbf{Mod}[SP_1] \cap \mathbf{Mod}[SP_2] \end{aligned}$$

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

$$\begin{aligned} Sig[\textbf{translate } SP \textbf{ by } \sigma] &:= \Sigma' \\ \mathbf{Mod}[\textbf{translate } SP \textbf{ by } \sigma] &:= \{M' \in \mathbf{Mod}(\Sigma') \mid M'|_\sigma \in \mathbf{Mod}[SP]\} \end{aligned}$$

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

$$\begin{aligned} Sig[\textbf{derive from } SP' \textbf{ by } \sigma] &:= \Sigma \\ \mathbf{Mod}[\textbf{derive from } SP' \textbf{ by } \sigma] &:= \{M'|_\sigma \mid M' \in \mathbf{Mod}[SP']\} \end{aligned}$$

[1] $\mathcal{CAT}$ be the (quasi-)category of categories and functors.

We now come to the task of relating different institutions. Institution *morphisms* [10] 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.

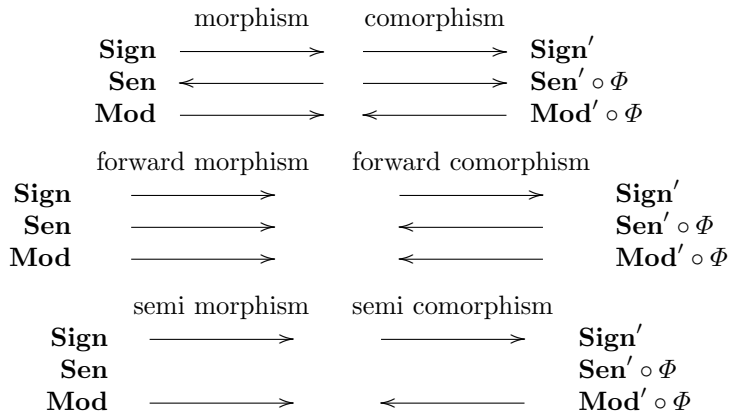
Given institutions I and J, an *institution morphism* [10] $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J$ consists of

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

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

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

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 [9, 18]:



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

These various notions of institution translations naturally lead to the following heterogeneous specification constructs [19, 20]:

***heterogeneous translation*:** For any institution comorphism, forward comorphism or semi-comorphism $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow I'$ and Σ-specification SP in I, **translate** SP **by** μ is a specification with:

$Sig[\textbf{translate } SP \textbf{ by } \mu] := \Phi(\Sigma)$

$\mathbf{Mod}[\textbf{translate } SP \textbf{ by } \mu] := \{M' \in \mathbf{Mod}(\Phi(\Sigma)) \mid \beta_\Sigma(M') \in \mathbf{Mod}[SP]\}$

***heterogeneous hiding*:** For any institution morphism, forward morphism or semi-morphism $\mu = (\Phi, \alpha, \beta)\colon I \longrightarrow I'$ and Σ-specification SP in I, **derive from** SP **by** μ is a specification with:

$Sig[\textbf{derive from } SP \textbf{ by } \mu] := \Phi(\Sigma)$

$\mathbf{Mod}[\textbf{derive from } SP \textbf{ by } \mu] := \{\beta_\Sigma(M') \mid M' \in \mathbf{Mod}[SP]\}$

3 Development Graphs

The notion of institutions gains much of its importance by the fact that one can design languages for structured specifications in a completely institution independent and even heterogeneous way, as explained in the previous section.

However, the standard proof calculi for proving entailment within and refinement between such structured specifications [2] rely on some form of the *Craig interpolation* property, which fails in some institutions (even for many-sorted logic, it holds only for sort-injective signature morphisms). Moreover, although Craig interpolation for heterogeneous specification has been studied [7], this of course relies on Craig interpolation for the individual logics, and some extra assumptions about Craig interpolation for the comorphisms that are not satisfied in many practical examples.

We hence propose to follow a different path, namely to use the formalism of *development graphs* [1]. The proof calculus for this does not rely on Craig interpolation; rather, is based on a different assumption, namely the existence of *weakly amalgamable cocones* [12]. Note that the latter property (although technically incomparable in strength with Craig interpolation) for practical purposes is a weaker assumption than Craig interpolation (cf. the results of [6]).

A development graph consists of a set of nodes (corresponding to whole structured specifications or parts thereof), and a set of arrows called definition links, indicating the dependency of each involved structured specification on its subparts.

Definition 2. Given an arbitrary but fixed institution I, a *development graph* $\mathcal{DG}$ 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 (Σ^N, Γ^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 $\sigma : \Sigma^M \to \Sigma^N \in \mathbf{Sign}^I$, or
- *hiding* (denoted $M \xrightarrow[h]{\sigma} N$), annotated with a signature morphism $\sigma : \Sigma^N \to \Sigma^M \in \mathbf{Sign}^I$ *going against the direction of the link*. Typically, σ will be an inclusion, and the symbols of Σ^M not in Σ^N will be hidden.

What is the meaning of such development graphs? Development graphs without hiding have a theory-level semantics, see [1]. For development graphs with hiding, a model-level semantics seems to be more appropriate:

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

[2] $\mathbf{Mod}_{\mathcal{DG}}$ is not to be confused with the model functor $\mathbf{Mod}$ of the institution.

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

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[3] (denoted by $N - \stackrel{\sigma}{-} \succ M$, where $\sigma\colon \Sigma^N \longrightarrow \Sigma^M$) are the central data structure to represent proof obligations arising in formal developments.

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

4 Grothendieck Institutions and Spans of Comorphisms

Heterogeneous specification can be viewed as structured specification over a Grothendieck construction. Diaconescu's Grothendieck institution construction [5] basically flattens a diagram of institution and morphisms. We here recall the Grothendieck institution for the comorphism-based case [12]:

Definition 5. An *indexed coinstitution* is a functor $\mathcal{I}\colon Ind^{op} \longrightarrow \mathbf{CoIns}$ into the category **CoIns** of institutions and institution comorphisms[4].

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* $\mathcal{I}\colon Ind^{op} \longrightarrow \mathbf{CoIns}$, define the *Grothendieck institution* $\mathcal{I}^\#$ as follows:

– signatures in $\mathcal{I}^\#$ are pairs (Σ, i), where $i \in |Ind|$ and Σ a signature in the institution $\mathcal{I}(i)$,
– signature morphisms $(\sigma, e)\colon (\Sigma_1, i) \longrightarrow (\Sigma_2, j)$ consist of a morphism $e\colon j \longrightarrow i \in Ind$ and a signature morphism $\sigma\colon \Phi^{\mathcal{I}(e)}(\Sigma_1) \longrightarrow \Sigma_2$ (here, $\mathcal{I}(e)\colon \mathcal{I}(i) \longrightarrow \mathcal{I}(j)$ is the institution comorphism corresponding to the arrow $e\colon j \longrightarrow i$ in the indexed coinstitution, and $\Phi^{\mathcal{I}(e)}$ is its signature translation component),

[3] There are also local and hiding theorem links, which are omitted here for simplicity.
[4] Indeed, the name is justified by the fact that the category of institutions and institution comorphisms is isomorphic to the category of coinstitutions and coinstitution morphisms. A coinstitution is an institution with model translations covariant to signature morphisms, while sentence translations are contravariant.

– the (Σ, i)-sentences are the Σ-sentences in $\mathcal{I}(i)$, and sentence translation along (σ, e) is the composition of sentence translation along σ with sentence translation along $\mathcal{I}(e)$,
– the (Σ, i)-models are the Σ-models in $\mathcal{I}(i)$, and model reduction along (σ, e) is the composition of model translation along $\mathcal{I}(e)$ with model reduction along σ, and
– satisfaction w.r.t. (Σ, i) is satisfaction w.r.t. Σ in $\mathcal{I}(i)$. □

While comorphism-based Grothendieck institutions are a good semantic basis for heterogeneous specifications involving institution comorphisms, the question arises what to do with the other kinds of translations, like institution morphisms or semi-morphisms. Rather than complicate the Grothendieck construction with different kinds of translations, instead we represent the other kinds of translations as *spans* of institution comorphisms.

The span construction works as follows:

Each institution morphism $\mu = (\Phi, \alpha, \beta) \colon I \longrightarrow J = I \;\begin{array}{c}\xrightarrow{\Phi}\\ \xleftarrow{\alpha}\\ \xrightarrow{\beta}\end{array}\; J$ can be translated into a span $I \xleftarrow{\mu^-} J \circ \Phi \xrightarrow{\mu^+} J$ of institution comorphisms as follows:

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

Here, the "middle" institution $J \circ \Phi$ is the institution with signature category inherited from I, but sentences and models inherited from J via Φ.

Consider now a semi-comorphism $I \;\begin{array}{c}\xrightarrow{\Phi}\\ \xleftarrow{\beta}\end{array}\; J$. It can be translated into a span $I \xleftarrow{\mu^-} I^\emptyset \xrightarrow{\mu^+} J$ of comorphisms

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

while a semi-morphism $I \;\begin{array}{c}\xrightarrow{\Phi}\\ \xrightarrow{\beta}\end{array}\; J$ is translated into a span $I \xleftarrow{\mu^-} J \circ \Phi^\emptyset \xrightarrow{\mu^+} J$ of comorphisms

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

where in each case the "middle" institution has the indicated components.

Forward comorphisms $\mu = (\Phi, \alpha, \beta) \colon I \longrightarrow J = I \xrightarrow{\Phi} \xleftarrow{\alpha} \xleftarrow{\beta} J$ are translated into spans of form $I \xleftarrow{\mu^-} J \circ \Phi^{\mathbf{Sen}} \xrightarrow{\mu^+} J$ consisting of institution comorphisms as follows:

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

The "middle" institution $J \circ \Phi^{\mathbf{Sen}}$ inherits signatures and models from I, but sentences (via Φ) from J. The satisfaction relation $M \models^{J \circ \Phi^{\mathbf{Sen}}}_{\Sigma} \varphi$ holds iff $\beta_\Sigma(M) \models^I_\Sigma \varphi$ in I.

Dually, a forward morphism $\mu = (\Phi, \alpha, \beta) \colon I \longrightarrow J = I \xrightarrow{\Phi} \xrightarrow{\alpha} \xrightarrow{\beta} J$ can be translated into a span $I \xleftarrow{\mu^-} J \circ \Phi^{\mathbf{Mod}} \xrightarrow{\mu^+} J$ of institution comorphisms as follows:

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

The "middle" institution $J \circ \Phi^{\mathbf{Mod}}$ inherits signatures and sentences from I, but models (via Φ) from J. The satisfaction relation $M \models^{J \circ \Phi^{\mathbf{Mod}}}_{\Sigma} \varphi$ holds iff $M \models^J_{\Phi(\Sigma)} \alpha_\Sigma(\varphi)$ in J.

5 The Heterogeneous Verification Semantics

The purpose of the heterogeneous verification semantics, which follows a similar structured verification semantics in [13], is to provide a proof calculus for heterogeneous specifications. Some heterogeneous calculus rules have been given already in [20]; however, they cover only the question whether a heterogeneous specification entails a sentence, but not the question whether a heterogeneous specification entails (or refines to) another one.

General assumption We assume to work with an indexed coinstitution that contains all the "middle" institutions as well as the μ^- and μ^+ comorphisms given by the constructions of the previous section, applied to those morphisms, semi-morphisms etc. that we expect to occur in our heterogeneous specifications. Of course, we assume that all institutions and comorphisms that are used directly are included as well.

The heterogeneous verification semantics now takes a heterogeneous specification and translates it into a development graph over the Grothendieck institution induced by the indexed coinstitution given by the above assumption.

In the sequel, if we want to extend a given development graph $\mathcal{DG}$, we use a suggestive concise notation like $\mathcal{DG}' = \mathcal{DG} \uplus \{N' := (\Sigma', \Gamma);\ \ N \xrightarrow{\sigma} N'\ \}$ which should be largely self-explanatory (in particular, '$N' := (\Sigma', \Gamma)$' means that we introduce a new node N' with $\Sigma^{N'} = \Sigma'$ and $\Gamma^{N'} = \Gamma$).

Furthermore, by abuse of notation, we identify institutions and comorphisms with their respective indices in the index category of the indexed coinstitution (in general, it is expected that the indexed coinstitution is an embedding of categories; hence this abuse of notation will not lead to ambiguities).

The verification semantics uses judgements of form

$$\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG})$$

which read as: the specification SP is translated to the node N in development graph $\mathcal{DG}$.

$$\frac{\Sigma \text{ is a signature in } I}{\vdash \langle \Sigma, \Gamma \rangle \rhd\!\!\rhd\!\!\rhd (N, \{N := ((\Sigma, I), \Gamma)\})}$$

$$\frac{\vdash SP_1 \rhd\!\!\rhd\!\!\rhd (N_1, \mathcal{DG}_1) \quad \vdash SP_2 \rhd\!\!\rhd\!\!\rhd (N_2, \mathcal{DG}_2)}{\vdash SP_1 \cup SP_2 \rhd\!\!\rhd\!\!\rhd (K, \mathcal{DG}_1 \uplus \mathcal{DG}_2 \uplus \{K := (\Sigma^{N_1}, \emptyset)\} \uplus \{\ N_i \xrightarrow{id} K \mid i = 1, 2\})}$$

$$\frac{\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG}) \quad N = (\Sigma, I)}{\vdash \textbf{translate } SP \textbf{ by } \sigma\colon \Sigma \longrightarrow \Sigma' \rhd\!\!\rhd\!\!\rhd (K, \mathcal{DG} \uplus \{\ N \xrightarrow{(\sigma, id_I)} K\ := ((\Sigma', I), \emptyset)\})}$$

$$\frac{\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG}) N = (\Sigma', I)}{\vdash \textbf{derive from } SP' \textbf{ by } \sigma\colon \Sigma \longrightarrow \Sigma' \rhd\!\!\rhd\!\!\rhd (K, \mathcal{DG} \uplus \{\ N \xrightarrow[h]{(\sigma, id_I)} K\ := ((\Sigma, I), \emptyset)\})}$$

$$\frac{\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG}) \quad N = (\Sigma, I) \quad \mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J \text{ is a comorphism}}{\vdash \textbf{translate } SP \textbf{ by } \mu\colon I \longrightarrow J \rhd\!\!\rhd\!\!\rhd (K, \mathcal{DG} \uplus \{\ N \xrightarrow{(id, \mu)} K\ := ((\Phi(\Sigma), J), \emptyset)\})}$$

$$\frac{\begin{array}{c}\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG}) \\ N = (\Sigma, I) \\ \mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J \text{ is a morphism} \\ \mathcal{DG}' = \mathcal{DG} \uplus \{\ N \xrightarrow[h]{(id,\mu^-)} K\ := (\Sigma, J \circ \Phi), \emptyset);\quad K \xrightarrow{(id,\mu^+)} P\ := ((\Phi(\Sigma), J), \emptyset)\}\end{array}}{\vdash \textbf{derive from } SP \textbf{ by } \mu\colon I \longrightarrow J \rhd\!\!\rhd\!\!\rhd (P, \mathcal{DG}')}$$

$$\frac{\begin{array}{c}\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG}) \\ N = (\Sigma, I) \\ \mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J \text{ is a semi-comorphism} \\ \mathcal{DG}' = \mathcal{DG} \uplus \{\ N \xrightarrow[h]{(id,\mu^-)} K\ := (\Sigma, I^{\emptyset}), \emptyset);\quad K \xrightarrow{(id,\mu^+)} P\ := ((\Phi(\Sigma), J), \emptyset)\}\end{array}}{\vdash \textbf{translate } SP \textbf{ by } \mu\colon I \longrightarrow J \rhd\!\!\rhd\!\!\rhd (P, \mathcal{DG}')}$$

$$\frac{\begin{array}{c}\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG}) \\ N = (\Sigma, I) \\ \mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J \text{ is a semi-morphism} \\ \mathcal{DG}' = \mathcal{DG} \uplus \{\ N \xrightarrow[h]{(id,\mu^-)} K\ := (\Sigma, J \circ \Phi^{\emptyset}), \emptyset);\quad K \xrightarrow{(id,\mu^+)} P\ := ((\Phi(\Sigma), J), \emptyset)\}\end{array}}{\vdash \textbf{translate } SP \textbf{ by } \mu\colon I \longrightarrow J \rhd\!\!\rhd\!\!\rhd (P, \mathcal{DG}')}$$

$$\frac{\begin{array}{c}\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG}) \\ N = (\Sigma, I) \\ \mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J \text{ is a forward comorphism} \\ \mathcal{DG}' = \mathcal{DG} \uplus \{\ N \xrightarrow[h]{(id,\mu^-)} K\ := (\Sigma, J \circ \Phi^{\mathbf{Sen}}), \emptyset);\quad K \xrightarrow{(id,\mu^+)} P\ := ((\Phi(\Sigma), J), \emptyset)\}\end{array}}{\vdash \textbf{translate } SP \textbf{ by } \mu\colon I \longrightarrow J \rhd\!\!\rhd\!\!\rhd (P, \mathcal{DG}')}$$

$$\frac{\begin{array}{c}\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG}) \\ N = (\Sigma, I) \\ \mu = (\Phi, \alpha, \beta)\colon I \longrightarrow J \text{ is a forward morphism} \\ \mathcal{DG}' = \mathcal{DG} \uplus \{\ N \xrightarrow[h]{(id,\mu^-)} K\ := (\Sigma, J \circ \Phi^{\mathbf{Mod}}), \emptyset);\quad K \xrightarrow{(id,\mu^+)} P\ := ((\Phi(\Sigma), J), \emptyset)\}\end{array}}{\vdash \textbf{translate } SP \textbf{ by } \mu\colon I \longrightarrow J \rhd\!\!\rhd\!\!\rhd (P, \mathcal{DG}')}$$

We now state the important property of the heterogeneous verification semantics:

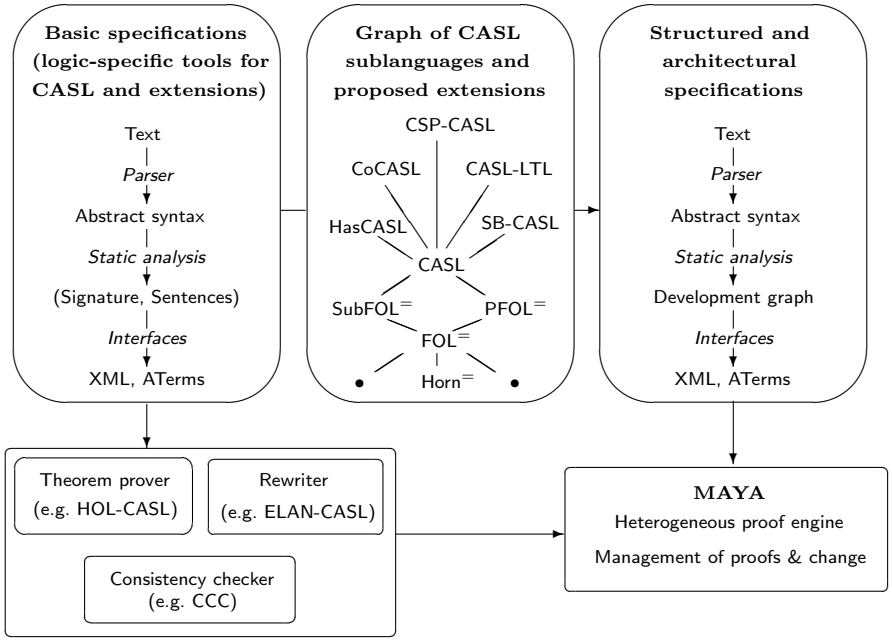
Theorem 7. The heterogeneous verification semantics preserves model classes of heterogeneous specifications. More precisely, given a heterogenous specification SP with $\vdash SP \rhd\!\!\rhd\!\!\rhd (N, \mathcal{DG})$, we have

$$\mathbf{Mod}[SP] = \mathbf{Mod}_{\mathcal{DG}}(N)$$

By inserting theorem links into a development graph generated by the heterogeneous verification semantics, it is now possible to tackle the problem of refinement between heterogeneous specifications. A proof calculus for such theorem links in the context of Grothendieck institutions has been given in [12]. This proof calculus relies on proof calculi (formalized as so-called entailment systems [11]) for the individual institutions, as well as some technical conditions on both the institutions (namely, the existence of weakly amalgamable cocones) as well as the comorphisms.

A word concerning the difference between morphisms and semi-morphisms is in order: although they are treated quite similarly in the heterogeneous verification semantics, their difference shows up when using the proof calculus: the "middle" institution is much poorer in the case of semi-morphisms (it has no sentences). The latter makes it much harder to conduct heterogeneous proofs; indeed, for proofs along semi-morphisms, typically both the source and target institution have to be translatable into some common target institution (or some special proof rules for the particular semi-morphism has to be added). Of course, a similar remark applies to semi-comorphisms as well.

6 The Heterogeneous Tool set (HETS)



The Heterogeneous Tool Set HETS is a tool implementing the theory developed so far. Its architecture is depicted above. HETS has an abstract interface corresponding to concept of institution (or more precisely, entailment system — since model theory is not directly implementable) in Haskell.

HETS implements this by providing a type class `Logic`. `Logic` is a multiparameter type classes with functional dependencies [15]. Such a type class can be thought of as a formal parameter signature. `Logic` contains types for signatures, signature morphisms, sentences, abstract syntax of basic specifications etc., and functions for parsing, printing, static analysis, and proving. Based on this abstract interface, we have implemented *heterogeneous* tools for parsing and static analysis of heterogeneous CASL (using a semantics similar to the verification semantics in Section 5 above). The static semantic analysis yields a development graph over the Grothendieck institution, and we are currently implementing the corresponding proof calculus.

Technically, heterogeneity is realized as follows. On top of the type class `Logic`, an existential datatype is constructed. Usually, existential types are used to realize e.g. heterogeneous lists, where each element may have a different type. We use lists of (components of) institutions and comorphisms instead. This leads to an implementation of the Grothendieck institution over an indexed coinstitution.

We have instantiated this general framework with institution-specific analysis tools for CASL, HASCASL, Haskell, CSP-CASL and MODALCASL. Future work will interface existing theorem proving tools with specific institutions in HETS. We already have implemented an experimental interface to the theorem prover Isabelle.

The Heterogeneous Tool Set is available at `www.tzi.de/cofi/hets`.

Acknowledgements

Thanks to Andrzej Tarlecki, Joseph Goguen, Grigore Rosu, Serge Autexier and Dieter Hutter for useful cooperation and discussions, and to Răzvan Diaconescu for inventing Grothendieck institutions.

This work has been supported by the *Deutsche Forschungsgemeinschaft* under Grant KR 1191/5-2.

References

1. 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*, volume 1827 of *Lecture Notes in Computer Science*, pages 73–88. Springer-Verlag, 2000.
2. T. Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286:197–245, 2002.
3. C. Caleiro, W. A. Carnielli, M. E. Coniglio, A. Sernadas, and C. Sernadas. Fibring non-truth-functional logics: Completeness preservation. *Journal of Logic, Language and Information*, 12(2):183–211, 2003.
4. C. Caleiro, P. Mateus, J. Ramos, and A. Sernadas. Combining logics: Parchments revisited. *Lecture Notes in Computer Science*, 2267:48–??, 2001.
5. R. Diaconescu. Grothendieck institutions. *Applied categorical structures*, 10:383–402, 2002.

6. R. Diaconescu. An institution-independent proof of Craig Interpolation Property. *Studia Logica*, 76(3), 2004.
7. R. Diaconescu. Interpolation in Grothendieck Institutions. *Theoretical Computer Science*, 311(1–3):439–461, Jan. 2004.
8. D. M. Gabbay. *Fibring Logics*. Oxford University Press, Oxford, 1999.
9. J. Goguen and G. Rosu. Institution morphisms. *Formal aspects of computing*, 13:274–307, 2002.
10. 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.
11. J. Meseguer. General logics. In *Logic Colloquium 87*, pages 275–329. North Holland, 1989.
12. T. Mossakowski. Comorphism-based Grothendieck logics. In K. Diks and W. Rytter, editors, *Mathematical foundations of computer science*, volume 2420 of *LNCS*, pages 593–604. Springer, 2002.
13. T. Mossakowski, P. Hoffman, S. Autexier, and D. Hutter. CASL logic. In P. D. Mosses, editor, *CASL Reference Manual*, volume 2960 of *Lecture Notes in Computer Science*, part ÏV. Springer Verlag, London, 2004. Edited by T. Mossakowski.
14. 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.
15. S. Peyton Jones, M. Jones, and E. Meijer. Type classes: exploring the design space. In *Haskell Workshop*. 1997.
16. D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.
17. A. Sernadas, C. Sernadas, C. Caleiro, and T. Mossakowski. Categorical fibring of logics with terms and binding operators. In D. Gabbay and M. d. Rijke, editors, *Frontiers of Combining Systems 2*, Studies in Logic and Computation, pages 295–316. Research Studies Press, 2000.
18. 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.
19. 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.
20. A. Tarlecki. Software specification and development in heterogeneous environments. This volume, 2004.