A Proof Theoretic Interpretation of Model Theoretic Hiding

Mihai Codescu, Fulya Horozal, Michael Kohlhase, Till Mossakowski, Florian Rabe

Jacobs University Bremen, DFKI Bremen

Abstract. Logical frameworks like LF are used for formal representations of logics in order to make them amenable to formal machine-assisted meta-reasoning. While the focus has originally been on logics with a proof theoretic semantics, we have recently shown how to define model theoretic logics in LF as well. We have used this to define new institutions in the Heterogeneous Tool Set in a purely declarative way.
It is desirable to extend this model theoretic representation of logics to the level of structured specifications. Here a particular challenge among structured specification building operations is hiding, which restricts a specification to some export interface. Specification languages like ASL and CASL support hiding, using an institution-independent model theoretic semantics abstracting from the details of the underlying logical system.
Logical frameworks like LF have also been equipped with structuring languages. However, their proof theoretic nature leads them to a theory-level semantics without support for hiding. In the present work, we show how to resolve this difficulty.

1 Introduction

This work is about reconciling the model theoretic approach of algebraic specifications and institutions [AKKB99,ST10,GB92] with the proof theoretic approach of logical frameworks [HHP93,Pau94].

In [Rab10,CHK+10], we show how to represent institutions in logical frameworks, notably LF [HHP93], and extend the Heterogeneous Tool Set [MML07] with a mechanism to add new logics that are specified declaratively in a logical framework.

In the present work, we extend this to the level of structured specifications, including hiding. In particular, we will translate the ASL-style structured specifications with institutional semantics [SW83,Wir86,ST88] (also used in CASL [Mos04]) into the module system MMT [Rab08] that has been developed in the logical frameworks community.

Like ASL, MMT is a generic structuring language that is parametric in the underlying language. But where ASL assumes a model theoretic base language – given as an institution – MMT assumes a proof theoretic base language given in terms of typing judgments. If we instantiate MMT with LF (as done in [RS09]), we can represent both logics and theories as MMT-structured LF signatures. This is used in the LATIN project [KMR09] to obtain a large body of structured representations of logics and logic translations. An important practical benefit of MMT is that it is integrated with a scalable knowledge management infrastructure based on OMDoc [Koh06].

However, contrary to model theoretic structuring languages like ASL, structuring languages like MMT for logical frameworks have a proof theoretic semantics and do not support hiding, which makes them less expressive than ASL. Therefore, we proceed in two steps. Firstly, we extend LF+MMT with primitives that support hiding while preserving its proof theoretic flavor. Here we follow and extend the theory-level semantics for hiding given in [GR04]. Secondly, we assume an institution that has been represented in LF, and give a translation of ASL-structured specifications over it into the extended LF+MMT language.

The paper is organised as follows. In Sect. 2, we recall ASL- or CASL-style structured specifications with their institution-independent semantics; and in Sect. 3 we recall LF and MMT with its proof theoretic semantics. In Sect. 4, we extend MMT with hiding, and in Sect. 5, we define a translation of ASL style specifications into MMT and prove its correctness. Sect. 6 concludes the paper.

2 Structured specifications

The notion of institution [GB92] has been introduced as a formalisation of the notion of logical system. It abstracts away from the details of signatures, sentences and models. Moreover, it assumes that signatures can be related via signature morphisms (and this carries over to sentences and models). This will be of importance for structuring languages.

Definition 1. *An* **institution** *is a quadruple $I = (Sig, Sen, Mod, \models)$ where:*

- *Sig is a category of* signatures*;*
- *$Sen : Sig \to \mathcal{SET}$ is a functor to the category $\mathcal{SET}$ of small sets and functions, giving for each signature Σ its set of* sentences *$Sen(\Sigma)$ and for any signature morphism $\varphi : \Sigma \to \Sigma'$ the* sentence translation *function $Sen(\varphi) : Sen(\Sigma) \to Sen(\Sigma')$ (denoted also φ);*
- *$Mod : Sig^{op} \to Cat$ is a functor to the category of categories and functors Cat [1] giving for any signature Σ its category of models $Mod(\Sigma)$ and for any signature morphism $\varphi : \Sigma \to \Sigma'$ the* model reduct *functor $Mod(\varphi) : Mod(\Sigma') \to Mod(\Sigma)$ (denoted $\_|_\varphi$);*
- *a satisfaction relation $\models_\Sigma \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$ for each signature Σ*

such that the following satisfaction condition holds:

$$M'|_\varphi \models_{\Sigma'} e \Leftrightarrow M' \models_\Sigma \varphi(e)$$

for each $M' \in |Mod(\Sigma')|$ and $e \in Sen(\Sigma)$, expressing that truth is invariant under change of notation and context.

For an institution I, a *theory* is a pair (Σ, E) where Σ is a signature and E is a set of sentences. For a class E of Σ-sentences, let us denote $Mod_\Sigma(E)$ the class of all Σ-models satisfying E and $Cl_\Sigma(E)$ the logical consequences of E.

Working with monolithic specifications is only suitable for specifications of fairly small size. For practical situations, in the case of large systems, a flat specification would become impossible to understand and use efficiently. Moreover, a modular design allows for reuse of specifications. Therefore, algebraic specification languages provide support for structuring specifications.

[1] We disregard here the foundational issues, but notice however that Cat is actually a so-called quasi-category.

The semantics of (structured) specifications can be given as a signature and either (i) a class of models of that signature (*model-level semantics*) or (ii) a set of sentences over that signature (*theory-level semantics*). In the presence of structuring, the two semantics may be different in a sense that will be made precise below. The first algebraic specification language, Clear [BG80], used a theory-level semantics; the first algebraic specification language using model-level semantics for structured specifications was ASL [SW83,Wir86], whose structuring mechanisms were extended to an institution-independent level in [ST88].

In Fig. 1, we present a kernel of specification-building operations and their semantics over an arbitrary institution, similar to the one introduced in [ST88]. The third and fourth columns of the table contain the model-level and the theory-level semantics for the corresponding **structured specification** SP, denoted $Mod[SP]$ and $Thm[SP]$ respectively, while the signature of SP, denoted $Sig[SP]$ is in the second column of the table. Note that we restrict attention to hiding against inclusion morphisms. Moreover, we will only consider basic specifications that are finite.

SP	$Sig[SP]$	$Mod[SP]$	$Thm[SP]$
(Σ, E)	Σ	$Mod_\Sigma(E)$	$Cl_\Sigma(E)$
$SP_1 \cup SP_2$ $Sig[SP_1] = Sig[SP_2]$	$Sig[SP_1]$	$Mod(SP_1) \cap Mod(SP_2)$	$Cl_\Sigma(Thm[SP_1] \cup Thm[SP_2])$
$\sigma(SP)$ $\sigma : Sig[SP] \to \Sigma'$	Σ'	$\{M \in Mod(\Sigma') \mid M\|_\sigma \in Mod[SP]\}$	$Cl_\Sigma(\{\sigma(e) \mid e \in Thm[SP]\})$
$\sigma^{-1}(SP)$ $\sigma : \Sigma \hookrightarrow Sig[SP]$	Σ	$\{M\|_\sigma \mid M \in Mod[SP]\}$	$\{e \in Sen(\Sigma) \mid \sigma(e) \in Thm[SP]\}$

Fig. 1. Semantics of Structured Specifications

Without hiding, the two semantics can be regarded as dual because we have $Thm[SP] = Thm(Mod[SP])$, which is called *soundness* and *completeness* in [ST10]. But completeness does not hold in general in the presence of hiding [Bor02]. Moreover, in [ST10] it is proved that this choice for defining the theory level semantics is the strongest possible choice with good structural properties (e.g. compositionality). This shows that the mismatch between theory-level semantics and model-level semantics cannot be bridged in this way. We will argue below that this is not a failure of formalist methods in general; instead, we will pursue a different approach that takes model-level aspects into account while staying mechanizable.

The mismatch between model and theory-level semantics is particularly apparent when looking at **refinements**. For two Σ-specifications SP and SP', we write $SP \rightsquigarrow_\Sigma SP'$ if $Mod[SP'] \subseteq Mod[SP]$. Without hiding, this is equivalent to $Thm[SP] \subseteq Thm[SP']$, which can be seen as soundness and completeness properties for refinements. But in the presence of hiding, both soundness (if SP has hiding) and completeness (if SP' has hiding) for refinements may fail.

3 LF and MMT

The Edinburgh Logical Framework LF [HHP93] is a proof theoretic logical framework based on a dependent type theory related to Martin-Löf type theory [ML74]. Precisely, it is the corner of the λ-cube [Bar92] that extends simple type theory with dependent function types. In [RS09], LF was extended with a module system called MMT. MMT [Rab08] is a generic module system which structures signatures using named imports and signature morphisms. The expressivity of MMT is similar to that of ASL or development graphs [AHMS99] except for hiding. In [Rab10], LF is used as a logical framework to represent both proof and model theory of object logics.

We give a brief summary of basic LF signatures, MMT-structured LF signatures, and the representation of model theory in LF in Sect. 3.1, 3.2, and 3.3, respectively. Our approach is not restricted to LF and can be easily generalized to other frameworks such as Isabelle or Maude along the lines of [CHK+10].

3.1 LF

LF expressions E are grouped into kinds K, kinded type-families $A : K$, and typed terms $t : A$. The kinds are the base kind `type` and the dependent function kinds $\Pi x : A. K$. The type families are the constants a, applications $a\ t$, and the dependent function type $\Pi x : A. B$; type families of kind `type` are called types. The terms are constants c, applications $t\ t'$, and abstractions $\lambda x : A. t$. We write $A \to B$ instead of $\Pi x : A. B$ if x does not occur in B. An LF **signature** Σ is a list of kinded type family declarations $a : K$ and typed constant declarations $c : A$. Optionally, declarations may carry definitions. A grammar is given in Sect. 3 below.

Given two signatures Σ and Σ', an LF **signature morphism** $\sigma : \Sigma \to \Sigma'$ is a typing- and kinding-preserving map of Σ-symbols to Σ'-expressions. Thus, σ maps every constant $c : A$ of Σ to a term $\sigma(c) : \overline{\sigma}(A)$ and every type family symbol $a : K$ to a type family $\sigma(a) : \overline{\sigma}(K)$. Here, $\overline{\sigma}$ is the homomorphic extension of σ to Σ-expressions, and we will write σ instead of $\overline{\sigma}$ from now on. Signature morphisms preserve typing and kinding: if $\vdash_\Sigma E : E'$, then $\vdash_{\Sigma'} \sigma(E) : \sigma(E')$.

Composition and identity of signature morphisms are straightforward, and we obtain a category $\mathbb{LF}$ of LF signatures and morphisms. This category has **inclusion** morphisms by taking inclusions between sets of declarations. Moreover, it has **pushouts** along inclusions [HST94]. Finally, a **partial morphism** from Σ to Σ' is a signature morphism from a subsignature of Σ to Σ'.

LF uses the Curry-Howard correspondence to represent axioms as constants and theorem as defined constants (whose definiens is the proof). Then the typing-preservation of signature morphisms corresponds to the theorem preservation of theory morphisms.

3.2 LF+MMT

The motivation behind the MMT structuring operations is to give a flattenable, concrete syntax for a module system on top of a declarative language. Signature morphisms are used as the main concept to relate and form modular signatures, and signature morphisms can themselves be given in a structured way. Moreover, signature morphisms are always named and can be composed into morphism expressions.

The grammar for the LF+MMT language is given below where $[-]$ denotes optional parts. Object level expressions E unify LF terms, type families, and kinds, and morphism level expressions are composed morphisms:

Signature graph	$G ::= \cdot \mid G,\ \%\mathtt{sig}\,T = \{\Sigma\} \mid \%\mathtt{view}\,v : S \to T = \{\sigma\}$
Signatures	$\Sigma ::= \cdot \mid \Sigma,\ \%\mathtt{struct}\,s : S = \{\sigma\} \mid \Sigma,\ c : E[= E']$
Morphisms	$\sigma ::= \cdot \mid \sigma,\ \%\mathtt{struct}\,s := \mu \mid \sigma,\ c := E$
Object level expressions	$E ::= \mathtt{type} \mid c \mid x \mid E\ E \mid \lambda x : E.\,E \mid \Pi x : E.\,E \mid E \to E$
Morphism level expressions	$\mu ::= \cdot \mid T.s \mid v \mid \mu\ \mu'$

The $\mathbb{LF}$ signatures and signature morphisms are those without the keyword %`struct`. Those are called **flat**.

Syntax The module level declarations consist of named signatures R, S, T and two kinds of signature morphism declarations. Firstly, **views** $\%\mathtt{view}\,v : S \to T = \{\sigma\}$ occur on toplevel and declare an explicit morphism from S to T given by σ. Secondly, **structures** $\%\mathtt{struct}\,s : S = \{\sigma\}$ occur in the body of a signature T and declare an import from S into T. Structures carry a partial morphism σ from S to T, i.e., σ maps some symbols of S to expressions over T. Views and structures correspond to refinements and inclusion of subspecifications in unions in ASL and CASL.

MMT differs from ASL-like structuring languages in that it uses named imports. Consequently, the syntax of MMT can refer to all paths in the signature graph using composed morphisms; these morphism level expressions μ are formed from structure names $T.s$, view names v, and diagram-order composition $\mu\ \mu'$.

MMT considers morphisms μ from S to T as expressions on the module level. Such a morphism μ has type S and is valid over T. Most importantly, MMT permits structured morphisms: The morphisms σ occurring in views and structures from S to T may map a structure $\%\mathtt{struct}\,r : R = \{\sigma\}$ declared in S (i.e., a morphism level constant of type R over S) to a morphism $\mu : R \to T$ (i.e., a morphism level expression of type R over T). These are called structure maps $\%\mathtt{struct}\,r := \mu$.

Semantics The semantics of LF+MMT is given by **flattening**. Every well-formed LF+MMT signature graph G is flattened into a diagram $\overline{G}$ over $\mathbb{LF}$. Every signature S in G produces a node $\overline{S}$ in $\overline{G}$; every structure $\%\mathtt{struct}\,s : S = \{\sigma\}$ occurring in T produces an edge $\overline{T.s}$ from $\overline{S}$ to $\overline{T}$; and every view $\%\mathtt{view}\,v : S \to T = \{\sigma\}$ produces an edge $\overline{v}$ from $\overline{S}$ to $\overline{T}$. Accordingly, every morphism expression μ yields a morphism $\overline{\mu}$. These results can be found in [RS09], and we will only sketch the central aspects here.

The flattening is defined by recursively replacing all structure declarations and structure maps with lists of flat declarations. To flatten a structure declaration $\%\mathtt{struct}\,s : S = \{\sigma\}$ in a signature T, assume that S and σ have been flattened already. For every declaration $c : E[= E']$ in $\overline{S}$, we have in $\overline{T}$

- a declaration $s.c : \overline{T.s}(E) = E''$ in $\overline{S}$ if σ contains $c := E''$,
- a declaration $s.c : \overline{T.s}(E)\,[= \overline{T.s}(E')]$ in $\overline{S}$ otherwise.

The morphism $\overline{T.s}$ from $\overline{S}$ to $\overline{T}$ maps every $\overline{S}$-symbol c to the $\overline{T}$ symbol $s.c$.

For a view $\%\mathtt{view}\, v : S \to T = \{\sigma\}$, the morphism $\overline{v}$ from $\overline{S}$ to $\overline{T}$ is given by the flattening of σ. $\overline{\cdot}$ is the identity morphism in $\mathbb{LF}$, and $\overline{\mu\,\mu'}$ is the composition $\overline{\mu'} \circ \overline{\mu}$.

Finally, morphisms σ from S to T are flattened as follows. To flatten a structure map $\%\mathtt{struct}\, r := \mu$ where r is a structure from R to S, assume that R has been flattened already. Then the flattening of σ contains $s.c := \overline{\mu}(c)$ for every constant c in $\overline{R}$.

In particular, if $\%\mathtt{sig}\, T = \{\Sigma,\ \%\mathtt{struct}\, s : S = \{\sigma\}\}$ the semantics of signature graphs is such that the left diagram below is a pushout. Here S_0 is a subsignature of $\overline{S}$ such that $\overline{\sigma} : S_0 \to \overline{\Sigma}$. Moreover, if S declares a structure r of type R, then the semantics of a structure map $\%\mathtt{struct}\, r := \mu$ occurring in σ is that the diagram on the right commutes.

$$\begin{tikzcd}
S_0 \arrow[r, hook] \arrow[d, "\overline{\sigma}"] & \overline{S} \arrow[d, "\overline{T.s}"'] \\
\overline{\Sigma} \arrow[r, hook] & \overline{T}
\end{tikzcd}
\qquad
\begin{tikzcd}
\overline{R} \arrow[r, "\overline{S.r}"] \arrow[dr, "\overline{\mu}"] & \overline{S} \arrow[d, "\overline{T.s}"] \\
 & \overline{T}
\end{tikzcd}$$

3.3 Representing Logics in LF

The main idea behind the representation of models in LF is to represent models of Σ as LF morphisms $\ulcorner\Sigma\urcorner \to \mathcal{F}$ where $\ulcorner\Sigma\urcorner$ encodes Σ and $\mathcal{F}$ is some LF signature. Usually, $\mathcal{F}$ is a fixed signature representing the foundation of mathematics such as an encoding of set theory. The feasibility of this approach has been demonstrated in [HR10,IR10], which in particular give encodings of ZFC set theory, Mizar's set theory, and Isabelle's higher-order logic.

$$\begin{tikzcd}
\mathcal{F} \arrow[r, "id_{\mathcal{F}}"] \arrow[d, hook] & \mathcal{F} \\
L^{Mod} \arrow[r, hook] & \Sigma^{Mod} \arrow[u, "M"'] \\
L^{Syn} \arrow[r, hook] \arrow[u, "L^{mod}"'] & \Sigma^{Syn} \arrow[u, "\Sigma^{mod}"']
\end{tikzcd}$$

We use a simplified variant and represent the underlying logic L as tuples $(L^{Syn}, L^{Mod}, L^{mod}, \mathcal{F})$ as in the commuting LF diagram on the right. L^{Syn} represents the syntax of the logic, $\mathcal{F}$ represents the foundation of mathematics, and L^{Mod} represents individual models as an extension of $\mathcal{F}$. Finally, L^{mod} interprets the syntax in the model.

Individual signatures are represented as inclusion morphisms out of L, from which we obtain Σ^{Mod} and Σ^{mod} as a pushout. Now we can see Σ^{Mod} as a theory in the meta-language $\mathcal{F}$ axiomatizing Σ models. Thus models M can finally be represented as morphisms from Σ^{Mod} into the foundation that are the identity on $\mathcal{F}$.

Due to the Curry-Howard representation of proofs as terms, there is no conceptual difference between representing signatures and theories of the underlying logic. If Σ^{Syn} contains axioms, then so does Σ^{Mod}, and M must map these axioms to proofs in $\mathcal{F}$.

We assume that L^{Syn} contains two distinguished declarations $o : \mathtt{type}$ and $\mathtt{ded} : o \to \mathtt{type}$. Then Σ-sentences are represented as closed $\beta\eta$-normal terms of type o over Σ^{Syn}. The satisfaction of a sentence F by a model M is represented as the inhabitation of the type $M(\Sigma^{mod}(\mathtt{ded}\ F))$ over $\mathcal{F}$. Theorems are represented as sentences F for which the type $\mathtt{ded}\ F$ is inhabited over Σ^{Syn}. In [Rab10,CHK+10], the proof theory of the logic is represented parallel to the model theory as a morphism $L^{pf} : L^{Syn} \to L^{Pf}$ where L^{Pf} adds the proof rules that populate the types $\mathtt{ded}\, F$. Here, we will assume for simplicity that $L^{Syn} = L^{Pf}$, and our results easily extend to the general case.

4 Hiding in LF and MMT

In a proof theoretic setting, flattening is not a theorem but rather the way to assign meaning to a modular signature. Therefore, hiding is a particularly difficult operation to add to systems like LF+MMT because hiding precludes flattening. We follow the approach taken in [GR04] and represent signatures with hidden information as inclusions $\Sigma_v \hookrightarrow \Sigma_h$ where Σ_v represents the visible interface $\Sigma_h \setminus \Sigma_v$ the hidden information. We will abstractly introduce LF signatures with hidden declarations in and morphisms between such signatures in Sect. 4.1. Then we will give concrete syntax for them in and generalize the MMT structuring operations in Sect. 4.2.

4.1 LF with Hiding

We will not only introduce LF signatures with hidden declarations but also LF morphisms that hide constants. The latter is similar to partial morphisms but has to be distinguished from the partial morphisms that already occur in MMT-structures. Therefore, we need two kinds of partiality and use the following definition.

Given two $\mathbb{LF}$-signatures Σ and Σ', an **H-morphism** from Σ to Σ' consists of two subsignatures $\Sigma_0 \hookrightarrow \Sigma_1 \hookrightarrow \Sigma$ and an $\mathbb{LF}$ signature morphism $\sigma : \Sigma_0 \to \Sigma'$. The intuition is that σ maps all constants in Σ_0 to Σ'-expressions and hides all constants in $\Sigma \setminus \Sigma_1$; for the intermediate declarations in $\Sigma_1 \setminus \Sigma_0$, σ is left undefined, i.e., partial. We call Σ_0 the **revealed domain** and Σ_1 the **non-hidden domain** of σ. We call σ **total** if $\Sigma_1 = \Sigma_0$ and otherwise **partial**; and we call σ **revealing** if $\Sigma = \Sigma_1$ and otherwise **hiding**. Then the (partial) revealing morphisms are exactly the (partial) $\mathbb{LF}$-morphisms.

For a Σ-expression E, we say that σ **maps** E if E is a Σ_0-expression and that σ **hides** E if E is not a Σ_1-expression. Then we can define a **composition** of total H-morphisms as follows: The revealed domain of $\sigma' \circ \sigma$ is the largest subsignature of the revealed domain of σ comprising only constants c such that σ' maps $\sigma(c)$; then we can put $(\sigma' \circ \sigma)(c) = \sigma'(\sigma(c))$.

An **H-signature** is a pair $\Sigma = (\Sigma_v, \Sigma_h)$ such that Σ_v is a subsignature of Σ_h. We call Σ_h the **domain** and Σ_v the **visible domain** of Σ.

Finally, we define the category $\mathbb{LFH}$ whose objects are H-signatures and whose morphisms $(\Sigma_v, \Sigma_h) \to (\Sigma'_v, \Sigma'_h)$ are total H-morphisms from Σ_v to Σ'_v. Note that these morphisms are exactly the total morphisms from Σ_h to Σ'_v whose revealed domain is at most Σ_v.

Lemma 2. *$\mathbb{LFH}$ is indeed a category.*

Proof. The $\mathbb{LFH}$ identity of (Σ_v, Σ_h) is the $\mathbb{LF}$ identity of Σ_v. Neutrality is clear. Associativity follows after observing that $\sigma'' \circ (\sigma' \circ \sigma)$ hides c iff $(\sigma'' \circ \sigma') \circ \sigma$ hides c.

$\mathbb{LFH}$-morphisms only translate between the visible domains and may even use hiding in doing so. We are often interested in whether the hidden information could also be translated. Therefore, we define:

Definition 3. *For an $\mathbb{LFH}$-morphism $\sigma_0 : \Sigma \to \Sigma'$ with revealed domain Σ_0, we write $\sigma_0 : \Sigma \xrightarrow{!} \Sigma'$ if σ_0 can be extended to a total revealing morphism $\sigma : \Sigma_h \to \Sigma'_h$, i.e., if there is an $\mathbb{LF}$ morphism $\sigma : \Sigma_h \to \Sigma'_h$ that agrees with σ_0 on Σ_0.*

4.2 LF+MMT with Hiding

We can now extend the MMT structuring to $\mathbb{LFH}$, i.e., to a base language with hiding. The flattening of signature graphs with hiding will produce $\mathbb{LFH}$-diagrams.

We avoid using pairs (Σ_v, Σ_h) in the concrete syntax for H-signatures and instead extend the grammar of LF+MMT as follows:

$$\begin{array}{ll} \text{Signatures } \Sigma & ::= \cdot \mid \Sigma,\ [\%\texttt{hide}]\ \%\texttt{struct}\, s\ :S\ = \{\sigma\} \mid \Sigma,\ [\%\texttt{hide}]\ c : E[= E] \\ \text{Morphisms } \sigma & ::= \cdot \mid \sigma,\ \%\texttt{struct}\, s := \mu \mid \sigma,\ c := E \mid \%\texttt{hide}\, c \mid \%\texttt{hide}\, \%\texttt{struct}\, s \end{array}$$

If a declaration in Σ has the %`hide` modifier, we call it **hidden**, otherwise **visible**. Hidden declarations are necessary to keep track of the hidden information. From a proof theoretical perspective, it may appear more natural to delete them, but this would not be adequate to represent ASL specifications with hiding.

If σ contains %`struct` $c := E$ (or %`hide` c), we say that σ **maps** (or **hides**) c, and accordingly for structures. As before, we call signatures or morphisms **flat** if they do not contain the %`struct` keyword.

The semantics of a well-formed signature graph G is given in two steps: first G is flattened into a flat signature graph $\widetilde{G}$, second the semantics of a flat signature graph G is given by an $\mathbb{LFH}$-diagram $\overline{G}$. In particular, every composite μ from S to T occurring in G induces a total H-morphism $\overline{\mu} : \overline{S}_v \to \overline{T}_v$.

Well-formedness and semantics are defined in a joint induction on the structure of G, and only minor adjustments to the definition of $\overline{G}$ for LF+MMT are needed. We begin with the flat syntax.

Firstly, a flat signature %`sig` $T = \{\Sigma\}$ induces a hiding signature $\overline{T} = (\overline{T}_v, \overline{T}_h)$ as follows: $\overline{T}_h$ contains all declarations in Σ, and $\overline{T}_v$ is the largest subsignature of $\overline{T}_h$ that contains only visible declarations. Σ is well-formed if this is indeed a well-formed $\mathbb{LFH}$-object.

Secondly, consider a flat morphism σ and two flat signatures S and T in G. σ induces an H-morphism from $\overline{S}_v$ to $\overline{T}_v$ as follows: Its revealed domain is the smallest subsignature of $\overline{S}_v$ that contains all constants mapped by σ; its non-hidden domain is the largest subsignature of $\overline{S}_v$ that contains no constants hidden by σ. σ is well-formed if this is indeed a well-formed H-morphism from $\overline{S}_v$ to $\overline{T}_v$.

Next we define the semantics of the full syntax by flattening an arbitrary signature graph G to $\widetilde{G}$. We use the same definition as in [RS09] except for additionally keeping track of hidden declarations.

Firstly, consider a signature T with a structure %`struct` $s\ :\ S\ =\ \{\sigma\}$, and consider a declaration of c in $\widetilde{S}$. Then $\widetilde{T}$ contains a constant $s.c$ defined in the same way as for LF+MMT. Moreover, $s.c$ is hidden in $\widetilde{T}$ if s is hidden in T, c is hidden in S, or $\widetilde{\sigma}$ hides c.

Secondly, consider an occurrence of %`struct` $s := \mu$ in σ in a structure or view declaration with domain S. Since the semantics $\overline{\mu}$ of μ is a total H-morphism, we must consider two cases for every visible constant c in $\widetilde{S}$: if c is in the revealed domain of $\overline{\mu}$, then $\widetilde{\sigma}$ contains $s.c := \overline{\mu}(c)$ as for LF+MMT; otherwise, $\widetilde{\sigma}$ contains $\%\mathtt{hide}\, s.c$.

Thirdly, consider an occurrence of $\%\mathtt{hide}\,\%\mathtt{struct}\, s$ in σ in a structure or view declaration with domain S. Then $\widetilde{\sigma}$ contains $\%\mathtt{hide}\, s.c$ for every visible constant c of $\widetilde{S}$.

Finally, to define well-formedness of signature graphs, we use the same inference system as in [RS09] with the following straightforward restriction for morphisms: In a structure declaration $\%\mathtt{struct}\, s : S = \{\sigma\}$ within T or in a view declaration $\%\mathtt{view}\, v : S \to T = \{\sigma\}$, $\overline{\widetilde{\sigma}}$ must be an H-morphism from $\overline{\widetilde{S}}_v$ to $\overline{\widetilde{T}}_v$. $\overline{\widetilde{\sigma}}$ must be total for views and may be partial for structures. Such structures and views induce edges $\overline{T.s}$ and $\overline{v}$ in $\overline{G}$ in the obvious way.

It is easy to show that well-formedness of the flat syntax is decidable. Moreover, we have

Theorem 4. *$\overline{\widetilde{G}}$ is a diagram over $\mathbb{LFH}$ for every well-formed signature graph G.*

Proof. This is proved by a straightforward induction on the structure of G.

The morphisms σ in structures and views may only map symbols of the visible domain. Moreover, they may hide some of these symbols. However, if we inspect the definition of the flattening of a structure $\%\mathtt{struct}\, s : S = \{\sigma\}$, we see that it imports all constants of $\overline{S}$ including the hidden ones and including those hidden by σ. Therefore, we have:

Lemma 5. *Assume a well-formed signature graph with hiding G containing a structure $\%\mathtt{struct}\, s : S = \{\sigma\}$ in T. Then $\overline{T.s} : \overline{S} \xrightarrow{!} \overline{T}$.*

Proof. The extension of $\overline{T.s}$ to $\overline{S}_h$ maps every constant c to $s.c$.

5 Interpreting ASL in LF+MMT

We now introduce the translation from ASL-style structured specifications into LF+MMT. We assume that there is a representation of an institution I in LF (see Sect. 5.1), such that when translating an ASL-style specification over I (see Sect. 5.2), the resulting MMT specification is based on this representation. The subsequent subsections deal with proving adequacy of the translation.

5.1 Logics

Consider an encoding as in Sect. 3 for an institution I. We make the following assumptions about the adequacy of the encoding.

Definition 6. *We say that a foundation $\mathcal{F}$ is* **adequate** *if there is (i) a $\mathcal{F}$-type $prop$: $\mathtt{type}$ such that the formal statements of mathematics can be encoded as $\mathcal{F}$-terms $F : prop$ and (ii) a type $\mathtt{ded}\, F$ of proofs of F for every $F : prop$ such that $\mathtt{ded}\, F$ is inhabited iff F is a provable statement.*

This definition is necessarily vague. To make it definite, we can assume that $\mathcal{F}$ is the encoding of Zermelo-Fraenkel set theory given in [HR10], in which case the terms of type *prop* are first-order formulas over the binary predicate symbol $\in$.

Definition 7. *Assume an adequate $\mathcal{F}$. We say that an institution $I = (Sig, Sen, Mod, \models)$ is* **adequately** *represented as $(L^{Syn}, \mathcal{F}, L^{Mod}, L^{mod})$ if there is a functor $\Phi : Sig \to \mathbb{LF}/L^{Syn}$ such that for every signature Σ (i) $\Phi(\Sigma) = \Sigma^{Syn}$ is an extension of L^{Syn}, (ii) there is a bijection $\ulcorner - \urcorner$ mapping Σ-sentences to $\beta\eta$-normal Σ^{Syn}-terms of type o, and $\ulcorner - \urcorner$ is natural with respect to sentence translation $Sen(\sigma)$ and morphism application $\Phi(\sigma)$ (iii) there is a bijection $\ulcorner - \urcorner$ mapping Σ-models to $\mathbb{LF}$-morphisms $\Sigma^{Mod} \to \mathcal{F}$, and $\ulcorner - \urcorner$ is natural with respect to model reduction $Mod(\sigma)$ and precomposition with $\Phi(\Sigma)^{mod}$, (iv) satisfaction $M \models_\Sigma F$ holds iff $\ulcorner M \urcorner$ maps $\Sigma^{mod}(\ulcorner F \urcorner)$ to an inhabited $\mathcal{F}$-type.*

Using the definitions of [Rab10], this can be stated as an institution comorphism from I to an appropriate institution based on LF.

Our assumption of a bijection between I-models and $\mathbb{LF}$-morphisms is quite strong. In most cases, not all models will be representable as morphisms. However, using canonical models constructed in completeness proofs, in many cases it will be possibly to represent all models up to elementary equivalence.

5.2 Specifications

We define a translation from ASL specifications to signature graphs of LF+MMT with hiding. Since the ASL structuring is built over an arbitrary institution, we assume that the underlying institution has already been represented in LF and the representation is adequate.

The translation proceeds by induction on the structure of the specification SP. However, MMT does not use signature expressions in the way ASL uses specification-building operations; in particular, MMT structures may import only named signatures. Therefore, the translation introduces one MMT signature declaration for every specification-building operation used in SP. Note that this leads to an increase in size but not to the exponential blow-up incurred when flattening.

The cases of the translation are given in Fig. 2. Every specification-building operation yielding a specification SP over a signature Σ is translated to two MMT signatures of the form

$$\%\mathtt{sig}\, N_\Sigma = \{\%\mathtt{struct}\, l : L^{Syn}, \ulcorner \Sigma \urcorner\}, \qquad \%\mathtt{sig}\, N_{SP} = \{\%\mathtt{struct}\, s : N_\Sigma, \ulcorner SP \urcorner\}$$

$\ulcorner \Sigma \urcorner$ is a list of declarations representing the visible signature symbols, and similarly $\ulcorner SP \urcorner$ represents the hidden signature symbols and all axioms. These must refer to the logical symbols of the underlying logic, which is why N_Σ starts with an import from L^{Syn}. N_Σ and N_{SP} are fresh names generated during the translation.

We will describe the translation case by case visualizing the involved objects using diagrams in $\mathbb{LF}$. First we introduce one simplification of the notation. Recall that technically, the semantics $\overline{N_{SP}}$ of N_{SP} is an $\mathbb{LFH}$ object $(\overline{N_{SP}}_v, \overline{N_{SP}}_h)$ and similarly for $\overline{N_\Sigma} = (\overline{N_\Sigma}_v, \overline{N_\Sigma}_h)$. A simple induction will show that N_Σ never contains hiding and that $\overline{N_{SP}.s} : \overline{N_\Sigma}_v = \overline{N_\Sigma}_h \to \overline{N_{SP}}_v$ is an isomorphism in $\mathbb{LF}$. Therefore, we will always write N_Σ instead of $\overline{N_\Sigma}_v$, N_{SP} instead of $\overline{N_{SP}}_h$, and $N_{SP}.s$ instead of $\overline{N_{SP}.s}$.

$$\dfrac{SP := (\Sigma, \{F_1, \dots, F_n\})}{\begin{array}{l}\%\mathtt{sig}\, N_\Sigma \;=\; \{\%\mathtt{struct}\, l \;:\; L^{Syn},\, \ulcorner\Sigma\urcorner\} \\ \%\mathtt{sig}\, N_{SP} \;=\; \{\%\mathtt{struct}\, s \;:\; N_\Sigma,\, \%\mathtt{hide}\, a_1 : \mathtt{ded}\, \ulcorner F_1\urcorner,\, \dots,\, \%\mathtt{hide}\, a_n : \mathtt{ded}\, \ulcorner F_n\urcorner\}\end{array}}\ \mathit{Basic}$$

$$\dfrac{\begin{array}{ll}\Sigma = \mathit{Sig}[SP_1] = \mathit{Sig}[SP_2] & \%\mathtt{sig}\, N_\Sigma \;=\; \{\ulcorner\Sigma\urcorner\} \\ SP' := SP_1 \cup SP_2 & \%\mathtt{sig}\, N_{SP_1} \;=\; \{\%\mathtt{struct}\, s_1 \;:\; N_\Sigma,\, \ulcorner SP_1\urcorner\} \\ & \%\mathtt{sig}\, N_{SP_2} \;=\; \{\%\mathtt{struct}\, s_2 \;:\; N_\Sigma,\, \ulcorner SP_2\urcorner\}\end{array}}{\begin{array}{l}N_{SP'} \;=\; \{ \\ \quad \%\mathtt{struct}\, s \;:\; N_\Sigma, \\ \quad \%\mathtt{struct}\, t_1 \;:\; N_{SP_1} \;=\; \{\%\mathtt{struct}\, s_1 := s\},\, \%\mathtt{struct}\, t_2 \;:\; N_{SP_2} \;=\; \{\%\mathtt{struct}\, s_2 := s\} \\ \}\end{array}}\ \mathit{Union}$$

$$\dfrac{\begin{array}{ll} & \%\mathtt{sig}\, N_\Sigma \;=\; \{\%\mathtt{struct}\, l \;:\; L^{Syn},\, \ulcorner\Sigma\urcorner\} \\ \sigma : \Sigma \to \Sigma' & \%\mathtt{sig}\, N_{\Sigma'} \;=\; \{\%\mathtt{struct}\, l' \;:\; L^{Syn},\, \ulcorner\Sigma'\urcorner\} \\ SP' := \sigma(SP) & \%\mathtt{sig}\, N_{SP} \;=\; \{\%\mathtt{struct}\, s \;:\; N_\Sigma,\, \ulcorner SP\urcorner\} \\ & \%\mathtt{view}\, N_\sigma \;:\; N_\Sigma \;\to N_{\Sigma'} \;=\; \{\%\mathtt{struct}\, l := l',\, \ulcorner\sigma\urcorner\}\end{array}}{\%\mathtt{sig}\, N_{SP'} \;=\; \{\%\mathtt{struct}\, s' \;:\; N_{\Sigma'},\, \%\mathtt{struct}\, t \;:\; N_{SP} \;=\; \{\%\mathtt{struct}\, s := N_\sigma s'\}\}}\ \mathit{Transl}$$

$$\dfrac{\begin{array}{ll}\sigma : \Sigma \hookrightarrow \Sigma' & \%\mathtt{sig}\, N_\Sigma \;=\; \{\%\mathtt{struct}\, l \;:\; L^{Syn},\, \ulcorner\Sigma\urcorner\} \\ \mathit{dom}(\Sigma) = \{c_1, \dots, c_m\} & \%\mathtt{sig}\, N_{\Sigma'} \;=\; \{\%\mathtt{struct}\, l' \;:\; L^{Syn},\, \ulcorner\Sigma'\urcorner\} \\ \mathit{dom}(\Sigma') \setminus \mathit{dom}(\Sigma) = \{h_1, \dots, h_n\} & \%\mathtt{sig}\, N_{SP'} \;=\; \{\%\mathtt{struct}\, s' \;:\; N_{\Sigma'},\, \ulcorner SP'\urcorner\} \\ SP := \sigma^{-1}(SP') & \end{array}}{\begin{array}{l}N_{SP} \;=\; \{\%\mathtt{struct}\, s \;:\; N_\Sigma, \\ \qquad \%\mathtt{struct}\, t \;:\; N_{SP'} \;=\; \{\%\mathtt{struct}\, s'.l' := s.l,\, s'.c_1 := s.c_1,\, \dots,\, s'.c_m := s.c_m, \\ \qquad\qquad \%\mathtt{hide}\, s'.h_1,\, \dots,\, \%\mathtt{hide}\, s'.h_n\} \\ \}\end{array}}\ \mathit{Hide}$$

Fig. 2. Translation of ASL specifications to LF+MMT with Hiding

$$\begin{tikzcd} N_\Sigma \arrow[d, "N_{SP}.s"'] \\ N_{SP} \end{tikzcd}$$

The rule *Basic* translates basic specification $SP = (\Sigma, E)$ using the LF representation of the underlying institution. $\ulcorner\Sigma\urcorner$ contains one declaration for every non-logical symbol declared in Σ. For example, if L^{Syn} encodes first-order logic and has a declaration $i : \mathtt{type}$ for the universe, a binary predicate symbol p in Σ leads to a declaration $p : l.i \to l.i \to l.o$ in $\ulcorner\Sigma\urcorner$. All axioms $F \in E$, lead to a declaration

$a : \mathtt{ded}\ulcorner F \urcorner$ where a is a fresh name. This has the effect that axioms are always hidden, which simplifies the notation significantly; it is not harmful because the semantics of ASL does not depend on whether an axiom is hidden or visible.

$$\begin{tikzcd}
N_\Sigma \arrow[r, "N_{SP_1}.s_1"] \arrow[d, "N_{SP_2}.s_2"'] \arrow[dr, "N_{SP'}.s"] & N_{SP_1} \arrow[d, "N_{SP'}.t_1"] \\
N_{SP_2} \arrow[r, "N_{SP'}.t_2"'] & N_{SP'}
\end{tikzcd}$$

The rule *Union* assumes translations of Σ, SP_1, and SP_2 and creates the translation of $SP' = SP_1 \cup SP_2$ by instantiating N_{SP_1} and N_{SP_2} in such a way that they share N_Σ. The semantics of LF+MMT guarantees that the resulting diagram on the left is a pushout in $\mathbb{LF}$.

The rule *Transl* translates $SP' = \sigma(SP)$ assuming that σ and SP have been translated already. The signature morphism σ is translated to a view in a straightforward way. Recall that N_Σ and $N_{\Sigma'}$ contain no hidden declarations or axioms so that N_σ is a (total) morphism in $\mathbb{LF}$. The resulting diagram is the left diagram below; it is again a pushout in $\mathbb{LF}$.

Similarly, the rule *Hide* translates $SP = \sigma^{-1}(SP')$ assuming that SP has been translated already. As $\sigma : \Sigma \hookrightarrow \Sigma'$ is an inclusion, we only need to know the names c_i of the symbols in Σ and the names h_j of the symbols in $\Sigma' \setminus \Sigma$, which are to be hidden. Then we can form N_{SP} by importing from $N_{SP'}$ and mapping all symbols that remain visible to their counterparts in N_{SP} and hiding the remaining symbols. The resulting diagram is the right diagram below. Note that by Lem. 5, $N_{SP}.t$ extends to a total $\mathbb{LF}$ morphism $N_{SP}.t^*$; moreover, it is easy to verify that $N_{SP}.t^*$ is an isomorphism.

$$\begin{tikzcd}
N_\Sigma \arrow[r, "N_\sigma"] \arrow[d, "N_{SP}.s"'] & N_{\Sigma'} \arrow[d, "N_{SP'}.s'"] \\
N_{SP} \arrow[r, "N_{SP'}.t"'] & N_{SP'}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
N_\Sigma \arrow[r, hook] \arrow[d, "N_{SP}.s"'] & N_{\Sigma'} \arrow[d, "N_{SP'}.s'"] \\
N_{SP} & N_{SP'} \arrow[l, "N_{SP}.t^*"]
\end{tikzcd}$$

5.3 Adequacy for Specifications

The general idea of the encoding of models is given in Fig. 3. The diagram corresponds to the one from Sect. 3.3 except that both N_Σ and N_{SP} are drawn. $(N_{SP}.s)^{mod}$ arises as the unique factorization through the pushout N_Σ^{Mod}.

Our result is that models $M \in Mod^I[SP] \subseteq Mod^I(\Sigma)$ are encoded as $\mathbb{LF}$ morphisms $m : N_\Sigma^{Mod} \to \mathcal{F}$ that factor through N_{SP}^{Mod}.

The translation of ASL to MMT yields pushouts between $\mathbb{LF}$ signatures extending L^{Syn}, but models are stated in terms of signatures extending L^{Mod}. Therefore, we use the following simple lemma:

Lemma 8. *If the left diagram below is a pushout in $\mathbb{LF}$, then so is the right one.*

$$
\begin{tikzcd}
\mathcal{F} \arrow[r, "id_{\mathcal{F}}"] \arrow[d, hook] & \mathcal{F} & \\
L^{Mod} \arrow[r, hook] & N_{\Sigma}^{Mod} \arrow[u, "m"] \arrow[r, "(N_{SP}.s)^{mod}"'] & N_{SP}^{Mod} \arrow[ul, "m^*"'] \\
L^{Syn} \arrow[u, "L^{mod}"] \arrow[r, hook] & N_{\Sigma} \arrow[u, "N_{\Sigma}^{mod}"] \arrow[r, "N_{SP}.s"] & N_{SP} \arrow[u, "N_{SP}^{mod}"]
\end{tikzcd}
$$

Fig. 3. Representation of Models in the Presence of Hiding

$$
\begin{tikzcd}
\Sigma \arrow[r, "\sigma_1"] \arrow[d, "\sigma_2"'] & \Sigma_1 \arrow[d, "\tau_2"] \\
\Sigma_2 \arrow[r, "\tau_1"'] & \Sigma'
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\Sigma^{Mod} \arrow[r, "\sigma_1^{mod}"] \arrow[d, "\sigma_2^{mod}"'] & \Sigma_1^{Mod} \arrow[d, "\tau_2^{mod}"] \\
\Sigma_2^{Mod} \arrow[r, "\tau_1^{mod}"'] & \Sigma'^{Mod}
\end{tikzcd}
$$

Proof. This is shown with a straightforward diagram chase.

Then we are ready to state our main result:

Theorem 9. *Let I be an institution that is adequately represented in LF. Then for any signature Σ, any ASL-structured specification SP with $Sig[SP] = \Sigma$, and any Σ-model M*

$$M \in Mod^I[SP] \quad \text{iff} \quad \text{exists } m^* : N_{SP}^{Mod} \to \mathcal{F} \text{ such that } (N_{SP}.s)^{mod}\,;\, m^* = \ulcorner M \urcorner$$

Proof. The proof is done by induction on the structure of SP. All cases will refer to the corresponding diagrams in Sect. 5.2.

Case $SP = (\Sigma, E)$:

For the base case, the conclusion follows directly from the assumption that the representation of I in LF is adequate.

Case $SP = SP_1 \cup SP_2$:

Let $M \in Mod[SP]$ and $m := \ulcorner M \urcorner : N_{\Sigma}^{Mod} \to \mathcal{F}$. We want to factor m through N_{SP}^{Mod}. By definition, we have that $M \in Mod[SP_1]$ and $M \in Mod[SP_2]$. By the induction hypothesis for SP_1 and SP_2, we get that there are morphisms $m_i : N_{SP_i}^{Mod} \to \mathcal{F}$ such that $m = (N_{SP_i}.s)^{mod}; m_i$. Using the pushout property we get a unique morphism $m^* : N_{SP}^{Mod} \to \mathcal{F}$ such that $(N_{SP_i}.s)^{mod}; (N_{SP'}.t_i)^{mod}; m^* = m$ which gives us the needed factorization.

For the reverse inclusion, let $m := \ulcorner M \urcorner : N_{\Sigma}^{Mod} \to \mathcal{F}$ represent a Σ-model M and factor as $(N_{SP}.s)^{mod}; m^*$. Notice that by composing $(N_{SP}.t_i)^{mod}$ with m^* we get morphisms $m_i : N_{SP_i}^{Mod} \to \mathcal{F}$. By using the induction hypothesis, M is then a model of both SP_1 and SP_2 and by definition M is a model of SP.

Case $SP' = \sigma(SP)$**:**

Let $M' \in Mod[SP']$ and $m' := \ulcorner M' \urcorner : N_{\Sigma'}^{Mod} \to \mathcal{F}$. We want to prove that there is $m'^* : N_{SP'}^{Mod} \to \mathcal{F}$ such that $m' = (N_{SP'}.s')^{mod}; m'^*$. By definition $M'|_{\sigma} \in Mod[SP']$. By induction hypothesis for SP' we get a morphism $m := \ulcorner M'|_{\sigma} \urcorner : N_{\Sigma}^{Mod} \to \mathcal{F}$ and a morphism $m^* : N_{SP}^{Mod} \to \mathcal{F}$ such that $(N_{SP}.s)^{mod}; m^* = m = (N_{\sigma})^{mod}; m'$, where the latter equality holds due to the definition of model reduct. Using the pushout property we get the desired m'^*.

For the reverse inclusion, assume $m' := \ulcorner M' \urcorner : N_{\Sigma'}^{Mod} \to \mathcal{F}$ that factors as $(N_{SP'}.s')^{mod}; m'^*$. Then $(N_{SP}.s)^{mod}; (N_{SP'}.t)^{mod}; m'^*$ factors through N_{SP}^{Mod} and thus by induction hypothesis the reduct of M' is an SP-model, which by definition means that M' is an SP'-model.

Case $SP = \sigma^{-1}(SP')$**:**

Let M be an SP-model and let $m := \ulcorner M \urcorner : N_{\Sigma} \to \mathcal{F}$. We want to prove that m factors through N_{SP}^{Mod}. By definition M has an expansion M' to an SP'-model. By induction hypothesis, there are morphisms $m' := \ulcorner M' \urcorner : N_{\Sigma'}^{Mod} \to \mathcal{F}$ and $m'^* : N_{SP'}^{Mod} \to \mathcal{F}$ such that $(N_{SP'}.s')^{mod}; m'^* = m'$. Then $m = (N_{SP}.s)^{mod}; (N_{SP}.t^{*-1})^{mod}; m'^*$.

For the reverse inclusion, let $m := \ulcorner M \urcorner$ be a morphism that factors as $(N_{SP}.s)^{mod}; m^*$. We need to prove that M has an expansion to a SP'-model. We obtain it by applying the induction hypothesis to $m' := (N_{SP'}.s')^{mod}; (N_{SP}.t^*)^{mod}; m^*$.

Corresponding to the adequacy for models, we prove the adequacy for theorems:

Theorem 10. *Let I be an institution and assume that I has been represented in LF in an adequate way. Then for any signature Σ, any ASL-structured specification SP with $Sig[SP] = \Sigma$, and any Σ-sentence F*

$$F \in Thm^I[SP] \qquad \text{iff} \qquad N_{SP}.s(l.\mathtt{ded}\ulcorner F \urcorner) \text{ inhabited over } N_{SP}$$

Proof. This is proved by induction on SP. For the base case, this follows from the adequacy assumption. For the remaining cases, it follows easily after observing that due to Lem. 5 structures always translate (possibly hidden) theorems to (possibly hidden) theorems. We omit the details.

Finally we remark that $N_{nf(SP)}$ is a flat H-signature that is isomorphic to the flattening of N_{SP} where $nf(SP)$ is the normal form of SP as defined in [Bor02]. This can also be proved by induction and then used to prove the above results.

5.4 Adequacy for Refinements

We want to give a syntactical criterion for refinement $SP \leadsto_{\Sigma} SP'$. Consider the diagram on the right. $SP \leadsto_{\Sigma} SP'$ states that for all m, if m'^* exists, then some m^* exists such that the diagram commutes. Clearly, this holds if there is an $\mathbb{LF}$ morphism $\rho : N_{SP}^{Mod} \to N_{SP'}^{Mod}$.

$$\begin{tikzcd}
 & N_{SP} \arrow[rr, "N_{SP}^{mod}"] & & N_{SP}^{Mod} \arrow[d, dashed, "m^*"] \arrow[dd, bend left, Rightarrow, "\rho"] \\
N_{\Sigma} \arrow[ur, "N_{SP}.s"] \arrow[rr, "N_{\Sigma}^{mod}"] \arrow[dr, "N_{SP'}.s"'] & & N_{\Sigma}^{Mod} \arrow[ur, "(N_{SP}.s)^{mod}"] \arrow[r, "m"] \arrow[dr, "(N_{SP'}.s)^{mod}"'] & \mathcal{F} \\
 & N_{SP'} \arrow[rr, "N_{SP'}^{mod}"'] & & N_{SP'}^{Mod} \arrow[u, dashed, "m'^*"']
\end{tikzcd}$$

If $\mathcal{F}$ has some additional technical properties, we can also prove the opposite implication:

Theorem 11. *Let I be an institution that is adequately encoded in LF. Moreover, assume that (i) $\mathcal{F}$ can express I-models as tuples of their components, and (ii) whenever $\mathcal{F}$ can prove the existence of a model of a finite I-theory, $\mathcal{F}$ can also express some model of that theory.*

Then for ASL-specifications SP and SP' over the signature Σ, we have that $SP \rightsquigarrow_\Sigma SP'$ iff there is an $\mathbb{LF}$ morphism $\rho : N^{Mod}_{SP} \to N^{Mod}_{SP'}$ such that $(N_{SP}.s)^{mod}; \rho = (N_{SP'}.s)^{mod}$.

Proof. The right-to-left implication follows immediately using Thm. 9.

For the left-to-right implication, we assume $SP \rightsquigarrow_\Sigma SP'$ and work within $N^{Mod}_{SP'}$. Due to the adequacy of $\mathcal{F}$, $\ulcorner SP \rightsquigarrow_\Sigma SP' \urcorner$ is a provable statement of $\mathcal{F}$ and thus of $N^{Mod}_{SP'}$ (iii). Using (i), we can tuple the declarations in $N^{Mod}_{SP'} \setminus \mathcal{F}$ (excluding the axioms) and obtain a Σ' model m as a term over $\mathcal{F}$; using the axioms in $N^{Mod}_{SP'} \setminus \mathcal{F}$, we can prove that m' is an SP' model. Using (iii), we show that m is also an SP model. Then using (ii), we obtain an expression m^* over $N^{Mod}_{SP'}$ that expresses a model of $N_{SP'}$. Finally, using (i), we can project out the components of m^*. The morphism ρ maps every symbol of $N^{Mod}_{SP} \setminus \mathcal{F}$ (excluding the axioms) to the corresponding components; it maps all axioms to proofs about m^*.

The assumption (i) of this theorem is mild. In fact, if (i) did not hold, it would be dubious how the foundation can express institutions and models at all. The assumption (ii) is more restricting. It holds for example in foundations that have a choice operator such as ZF with global choice function or higher-order logic. It is also possible to establish (ii) for individual institutions by giving a constructive model existence proof.

6 Conclusion

With the translation presented in this paper, it is possible to encode ASL- and CASL-style structured specifications with hiding in proof theoretic logical frameworks. This provides a new perspective on structured specifications that emphasizes constructive and mechanizable notions. Our translation is given for MMT-structured LF, but it easily generalizes to other MMT-structured logical frameworks.

Our encoding can be generalized to specifications given as development graphs. In this context, our representation theorem for refinements can be strengthened to represent the hiding theorem links of [MAH06]. Even heterogeneous specifications [MT09,MML07] can be covered. As LF+MMT uses the same structuring operations for logics as for theories, this requires only the representation of the involved logics and logic translations in LF.

A theorem very similar to our representation theorem for refinements can be obtained for conservative extensions. This permits the interpretation of the proof calculus for refinement given in [Bor02]. In particular, the rules using an oracle for conservative extensions can be represented elegantly as the composition of LF signature morphisms.

The translation to MMT also has the benefit that we can re-use the infrastructure provided by languages like OMDoc [Koh06] (an XML-based markup format for mathematical documents) and tools like TNTBase [ZK09] (a versioned XML database for OMDoc documents that supports complex searches and queries, e.g., via XQuery). Further tools developed along these lines are the JOBAD framework (a JavaScript library for interactive mathematical documents), which will provide a web-based frontend for the Heterogeneous Tool Set, GMoc (a change management system), DocTip (a document and tool integration platform) and integration with the Eclipse framework (an integrated development environment).

References

AHMS99. S. Autexier, D. Hutter, H. Mantel, and A. Schairer. Towards an Evolutionary Formal Software-Development Using CASL. In D. Bert, C. Choppy, and P. Mosses, editors, *WADT*, volume 1827 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 1999.

AKKB99. E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner. *Algebraic Foundations of Systems Specification*. Springer, 1999.

Bar92. H. Barendregt. Lambda calculi with types. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1992.

BG80. R. Burstall and J. Goguen. The semantics of Clear, a specification language. In Dines Bjøorner, editor, *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer Berlin / Heidelberg, 1980.

Bor02. Tomasz Borzyszkowski. Logical systems for structured specifications. *Theor. Comput. Sci.*, 286(2):197–245, 2002.

CHK+10. M. Codescu, F. Horozal, M. Kohlhase, T. Mossakowski, F. Rabe, and K. Sojakova. Logical Frameworks in Hets, 2010. Workshop on Abstract Development Techniques.

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

GR04. J. Goguen and G. Rosu. Composing Hidden Information Modules over Inclusive Institutions. In O. Owe, S. Krogdahl, and T. Lyche, editors, *From Object-Orientation to Formal Methods, Essays in Memory of Ole-Johan Dahl*, pages 96–123. Springer, 2004.

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

HR10. F. Horozal and F. Rabe. Representing Model Theory in a Type-Theoretical Logical Framework. Under review, see http://kwarc.info/frabe/Research/HR_folsound_10.pdf, 2010.

HST94. R. Harper, D. Sannella, and A. Tarlecki. Structured presentations and logic representations. *Annals of Pure and Applied Logic*, 67:113–160, 1994.

IR10. M. Iancu and F. Rabe. Formalizing Foundations of Mathematics. Under review, see http://kwarc.info/frabe/Research/IR_foundations_10.pdf, 2010.

KMR09. M. Kohlhase, T. Mossakowski, and F. Rabe. The LATIN Project, 2009. See https://trac.omdoc.org/LATIN/.

Koh06. M. Kohlhase. *OMDoc: An Open Markup Format for Mathematical Documents (Version 1.2)*. Number 4180 in Lecture Notes in Artificial Intelligence. Springer, 2006.

MAH06. T. Mossakowski, S. Autexier, and D. Hutter. Development graphs - Proof management for structured specifications. *J. Log. Algebr. Program*, 67(1–2):114–145, 2006.

ML74. P. Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. In *Proceedings of the '73 Logic Colloquium*, pages 73–118. North-Holland, 1974.

MML07. T. Mossakowski, C. Maeder, and K. Lüttich. The Heterogeneous Tool Set. In O. Grumberg and M. Huth, editor, *TACAS 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522, 2007.

Mos04. P. D. Mosses, editor. CASL *Reference Manual*. Number 2960 (IFIP Series) in LNCS. Springer Verlag, 2004.

MT09. Till Mossakowski and Andrzej Tarlecki. Heterogeneous logical environments for distributed specifications. In Andrea Corradini and Ugo Montanari, editors, *WADT 2008*, volume 5486 of *Lecture Notes in Computer Science*, pages 266–289. Springer, 2009.

Pau94. L. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer, 1994.

Rab08. F. Rabe. *Representing Logics and Logic Translations*. PhD thesis, Jacobs University Bremen, 2008. Available at `http://kwarc.info/frabe/Research/phdthesis.pdf`.

Rab10. F. Rabe. A Logical Framework Combining Model and Proof Theory. To be submitted, see `http://kwarc.info/frabe/Research/rabe_combining_09.pdf`, 2010.

RS09. F. Rabe and C. Schürmann. A Practical Module System for LF. In J. Cheney and A. Felty, editors, *Proceedings of the Workshop on Logical Frameworks: Meta-Theory and Practice (LFMTP)*, pages 40–48. ACM Press, 2009.

ST88. D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.

ST10. Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Program Development*. To appear in Springer-Verlag, 2010.

SW83. Donald Sannella and Martin Wirsing. A kernel language for algebraic specification and implementation. In *ADT*, 1983.

Wir86. Martin Wirsing. Structured algebraic specifications: A kernel language. *Theor. Comput. Sci.*, 42:123–249, 1986.

ZK09. V. Zholudev and M. Kohlhase. TNTBase: a Versioned Storage for XML. In *Proceedings of Balisage: The Markup Conference 2009*, volume 3. Mulberry Technologies, Inc., 2009.