Extending Development Graphs with Hiding

Till Mossakowski[1], Serge Autexier[2], and Dieter Hutter[3]

[1] BISS, Dept. of Computer Science, University of Bremen, P.O. Box 330 440, D 28334 Bremen, till@tzi.de, Fax: +49 421 218 3054
[2] FR 6.2 Informatik, Saarland University, P.O. Box 15 11 50, D 66041 Saarbrücken, Autexier@ags.uni-sb.de, Fax: +49 681 302 2235
[3] DKFI GmbH, Stuhlsatzenhausweg 3, D 66123 Saarbrücken, hutter@dfki.de, Fax: +49 681 302 2235

Abstract. Development graphs are a tool for dealing with structured specifications in a formal program development in order to ease the management of change and reusing proofs. In this work, we extend development graphs with hiding (e.g. hidden operations). Hiding is a particularly difficult to realize operation, since it does not admit such a good decomposition of the involved specifications as other structuring operations do. We develop both a semantics and proof rules for development graphs with hiding. The rules are proven to be sound, and also complete relative to an oracle for conservative extensions. We also show that an absolute complete set of rules cannot exist. The whole framework is developed in a way independent of the underlying logical system (and thus also does not prescribe the nature of the parts of a specification that may be hidden).

1 Introduction

It has long been recognized that *specifications in the large* are only manageable if they are built in a structured way. Specification languages, like CASL [CASL98], provide various mechanisms to combine basic specifications to structured specifications. Analogously, verification tools have to provide appropriate mechanisms to structure the corresponding logical axiomatizations. In practice, a formal program development is an evolutionary process [VSE96]. Specification and verification are mutually intertwined. Failed proofs give rise to changes of the specification which in turn will render previously found proofs invalid. For practical purposes it is indispensable to restrict the effects of such changes to a minimum in order to preserve as much proof effort as possible after a change of the specification.

Various structuring operations have been proposed (e.g. [DGS91,ST88,ST92]) in order to modularize specifications and proof systems have been described to deal with them (e.g. [CM97,HWB97]). Traditionally, the main motivations for modularization have been the sharing of sub-specifications within one specification, the reuse of specifications, and the structuring of proof obligations as well as applicable lemmas. However, the structure of specifications can also be exploited when the effects of changes are analyzed.

H. Hussmann (Ed.): FASE 2001, LNCS 2029, pp. 269–283, 2001.
Springer-Verlag Berlin Heidelberg 2001

In [AHMS00], development graphs have been introduced as a tool for dealing with structured specifications in a way easing management of change and reusing proofs. Also, a translation of structured specifications in CASL, an international standard for algebraic specification languages, to development graphs has been set up. However, this translation does not cover the case of hiding yet. In this work, we extend development graphs in a way that allows also to deal with hiding. Compared with other structuring operations like union, renaming and parameterization, hiding is a particularly difficult to realize operation. This is because hiding does not admit such a good decomposition of the involved specifications as other structuring operations do.

2 Motivation

As a running example consider the following example of specifying and refining a sorting function *sorter*.

Given some specification of total orders and lists, an abstract specification of this sorting function may be denoted in CASL syntax as follows:

spec SORTING
[TOTALORDER]
=
{
LIST [**sort** *Elem*]
then
preds $is\_ordered : List[Elem];$
$permutation : List[Elem] \times List[Elem];$
forall $x, y : Elem;$
$L, L1, L2 : List[Elem]$
- $is\_ordered([])$
- $is\_ordered([x])$
- $is\_ordered(x :: (y :: L)) \Leftrightarrow x \leq y \wedge is\_ordered(y :: L)$
- $permutation(L1, L2) \Leftrightarrow (\forall x : Elem \bullet x \in L1 \Leftrightarrow x \in L2)$

then
op $sorter : List[Elem] \to List[Elem];$
forall $L : List[Elem]$
- $is\_ordered(sorter(L))$
- $permutation(L, sorter(L))$

}
hide $is\_ordered, permutation$
end

is_ordered and *permutation* are auxiliary predicates to specify *sorter*, and are hidden to the outside. A model of this specification is just an interpretation of the *sorter* function (together with a model of the imported specifications of total orders and lists) that can be extended to a model of the whole specification (including *is_ordered* and *permutation*).

During a development, we may refine SORTING into a design specification describing a particular sorting algorithm. For simplicity, we choose a sorting algorithm which recursively inserts the head element in the sorted tail list. In CASL we obtain the following specification:

spec INSERTSORT
  [TOTALORDER]
  =
{
    LIST [**sort** *Elem*]
  then
    ops $insert : Elem \times List[Elem] \to List[Elem];$
      $sorter : List[Elem] \to List[Elem];$

    forall $x, y : Elem;$
      $L : List[Elem]$
    - $insert(x, []) = [x]$
    - $insert(x, y :: L) =$
      $x :: insert(y, L)$ *when* $x \leq y$ *else* $y :: insert(x, L)$
    - $sorter([]) = []$
    - $sorter(x :: L) = insert(x, sorter(L))$

}
hide *insert*
end

Now the interesting question arises whether INSERTSORT is actually a refinement of SORTING; i.e. whether each INSERTSORT-model is also a SORTING-model.

3 Preliminaries

When studying development graphs with hiding, we want to focus on the structuring and want to abstract from the details of the underlying logical system. Therefore, we recall the abstract notion of logic from Meseguer [Mes89]. Logics consist of model theory and proof theory. Model theory is captured by the notion of *institution*, providing an abstract framework for talking about signatures, models, sentences and satisfaction. Proof theory is captured by the notion of *entailment system*, providing an abstract framework for talking about signatures, sentences and entailment.

Let $\mathcal{CAT}$ be the category of categories and functors.[1]

Definition 1. *An* institution *[GB92]* $\mathcal{I} = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$ *consists of*

- *a category* $\mathbf{Sign}$ *of* signatures,
- *a functor* $\mathbf{Sen}: \mathbf{Sign} \longrightarrow \mathbf{Set}$ *giving the set of* sentences $\mathbf{Sen}(\Sigma)$ *over each signature* Σ*, and for each signature morphism* $\sigma: \Sigma \longrightarrow \Sigma'$*, the sentence translation map* $\mathbf{Sen}(\sigma): \mathbf{Sen}(\Sigma) \longrightarrow \mathbf{Sen}(\Sigma')$*, where often* $\mathbf{Sen}(\sigma)(\varphi)$ *is written as* $\sigma(\varphi)$*,*

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

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

such that for each $\sigma\colon \Sigma \longrightarrow \Sigma'$ *in* $\mathbf{Sign}$ $M' \models_{\Sigma'} \sigma(\varphi) \Leftrightarrow M'|_\sigma \models_\Sigma \varphi$ *holds for each* $M' \in \mathbf{Mod}(\Sigma')$ *and* $\varphi \in \mathbf{Sen}(\Sigma)$ *(*satisfaction condition*).*

Definition 2. *An* entailment system $\mathcal{E} = (\mathbf{Sign}, \mathbf{Sen}, \vdash)$ *consists of a category* $\mathbf{Sign}$ *of* signatures*, a functor* $\mathbf{Sen}\colon \mathbf{Sign} \longrightarrow \mathbf{Set}$ *giving the set of* sentences *over a given signature, and for each* $\Sigma \in |\mathbf{Sign}|$, *an entailment relation* $\vdash_\Sigma \subseteq |\mathbf{Sen}(\Sigma)| \times \mathbf{Sen}(\Sigma)$ *such that the following properties are satisfied:*

1. reflexivity: *for any* $\varphi \in \mathbf{Sen}(\Sigma)$, $\{\varphi\} \vdash_\Sigma \varphi$,
2. monotonicity: *if* $\Gamma \vdash_\Sigma \varphi$ *and* $\Gamma' \supseteq \Gamma$ *then* $\Gamma' \vdash_\Sigma \varphi$,
3. transitivity: *if* $\Gamma \vdash_\Sigma \varphi_i$, *for* $i \in I$, *and* $\Gamma \cup \{\varphi_i \mid i \in I\} \vdash_\Sigma \psi$, *then* $\Gamma \vdash_\Sigma \psi$,
4. $\vdash$-translation: *if* $\Gamma \vdash_\Sigma \varphi$, *then for any* $\sigma\colon \Sigma \longrightarrow \Sigma'$ *in* $\mathbf{Sign}$, $\sigma[\Gamma] \vdash_{\Sigma'} \sigma(\varphi)$.

We write $\Gamma^{\vdash_\Sigma}$ for $\{\varphi \mid \Gamma \vdash_\Sigma \varphi\}$.

Definition 3. *A* logic *is a 5-tuple* $\mathcal{LOG} = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \vdash, \models)$ *such that:*

1. $(\mathbf{Sign}, \mathbf{Sen}, \vdash)$ *is an entailment system (denoted by* $ent(\mathcal{LOG})$*),*
2. $(\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$ *is an institution (denoted by* $inst(\mathcal{LOG})$*), and*
3. *the following* soundness condition *is satisfied: for any* $\Sigma \in |\mathbf{Sign}|$, $\Gamma \subseteq \mathbf{Sen}(\Sigma)$ *and* $\varphi \in \mathbf{Sen}(\Sigma)$, $\Gamma \vdash_\Sigma \varphi$ *implies* $\Gamma \models_\Sigma \varphi$

A logic is complete *if, in addition,* $\Gamma \models_\Sigma \varphi$ *implies* $\Gamma \vdash_\Sigma \varphi$.

Throughout the rest of the paper, we will work with an arbitrary but fixed logic $\mathcal{LOG} = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \vdash, \models)$ such that $\mathbf{Sign}$ has finite colimits, and $\mathcal{LOG}$ admits weak amalgamation, i.e. $\mathbf{Mod}$ maps finite colimits to weak limits.

Given a diagram $D\colon I \longrightarrow \mathbf{Sign}$, let us call a family $(m_i)_{i \in |I|}$ *consistent with* I, if for each $i \in |I|$, $m_i \in \mathbf{Mod}(D(i))$, and for each $l\colon i \longrightarrow j \in I$, $m_j|_{D(l)} = m_i$.

The weak amalgamation property can now be reformulated as follows: $\mathcal{LOG}$ admits weak amalgamation iff for each diagram $D\colon I \longrightarrow \mathbf{Sign}$ and each family $(m_i)_{i \in |I|}$ *consistent with* I, there exists a model $m \in Colim\ D$ with $m|_{\mu_i} = m_i$, where $\mu_i\colon D(i) \longrightarrow Colim\ D$ are the colimit injections.

There are plenty of logics satisfying the above requirements, e.g. many-sorted equational logic, many-sorted first-order logic, various temporal and object-oriented logics etc. The logic underlying CASL, subsorted partial first-order logic with sort generation constraints, does not admit weak amalgamation. However, the CASL logic can be embedded into a logic with a richer signature category and a model functor admitting (weak) amalgamation [SMHKT01]. Hence, the results of this paper also are applicable for CASL, if colimits are taken in the richer signature category.

4 Development Graphs with Hiding

A development graph, as introduced in [AHMS00], represents the actual state of a formal program development. It is used to encode the structured specifications in various phases of the development. Roughly speaking, each node of the graph represents a theory like for instance List, Sorting or InsertSort in Casl. The links of the graph define how theories can make use of other theories.

Leaves in the graph correspond to basic specifications, which do not make use of other theories (e.g. Total_order). Inner nodes correspond to structured specifications which define theories using other theories (e.g. Sorting using Total_order). The corresponding links in the graph are called *definition links*. Arising proof obligations are attached as so-called *theorem links* to this graph. We here add a new type of definition links corresponding to *hiding*.

Definition 4. *A* development graph *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, Φ^N) such that Σ^N is a signature and $\Phi^N \subseteq \mathbf{Sen}(\Sigma^N)$ 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 $\sigma : \Sigma^M \to \Sigma^N$, or*
- **local** *(denoted $M \xrightarrow{\sigma} N$), again annotated with a signature morphism $\sigma : \Sigma^M \to \Sigma^N$, or*
- **hiding** *(denoted $M \xrightarrow[h]{\sigma} N$), annotated with a signature morphism $\sigma : \Sigma^N \to \Sigma^M$* going against the direction of the link.

To simplify matters, we write $M \xrightarrow{\sigma} N \in \mathcal{S}$ instead of $M \xrightarrow{\sigma} N \in \mathcal{L}$ when $\mathcal{L}$ are the links of $\mathcal{S}$.

In Fig. 1 we present the development graph for the running example: The left part of the graph represents the structured specification Sorting, and the the right part the structured specification InsertSort.

The next definition captures the existence of a path of local and global definition links between two nodes. Notice that such a path must not contain any hiding links.

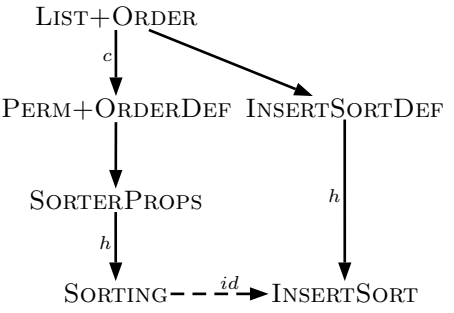


Fig. 1. Development graph for the sorting example

Definition 5. *Let $\mathcal{S}$ be a development graph. A node M is* **globally reachable** *from a node N via a mapping σ, $N \xrightarrow{\sigma} M \in \mathcal{S}$ for short, iff either $N = M$ and $\sigma = \lambda$, or $N \xrightarrow{\sigma'} K \in \mathcal{S}$, and $K \xrightarrow{\sigma''} M \in \mathcal{S}$, with $\sigma = \sigma'' \circ \sigma'$.*

A node M is **locally reachable** *from a node N via a mapping σ, $N \xrightarrow{\sigma} M \in \mathcal{S}$ for short, iff $N \xrightarrow{\sigma} M \in \mathcal{S}$ or there is a node K with $N \xrightarrow{\sigma'} K \in \mathcal{S}$, $K \xrightarrow{\sigma''} M \in \mathcal{S}$, such that $\sigma = \sigma'' \circ \sigma'$.*

Obviously global reachability implies local reachability since the theory of a node is defined with the help of local axioms.

Definition 6. *Let $\mathcal{S} = \langle \mathcal{N}, \mathcal{L} \rangle$ be a development graph. A node $N \in \mathcal{N}$ is* flatable *iff for all nodes $M \in \mathcal{N}$ with incoming hiding definition links, it holds that N is not reachable from M.*

The models of flatable nodes does not depend on existing hiding links. Thus we use the original approach of defining the theory of a node:

Definition 7. *Let $\mathcal{S} = \langle \mathcal{N}, \mathcal{L} \rangle$ be a development graph. For $N \in \mathcal{N}$, the* **theory** *$Th_{\mathcal{S}}(N)$ of N wrt. a development graph $\mathcal{S}$ is defined by*

$$Th_{\mathcal{S}}(N) \quad = \quad \left[\Phi^N \cup \bigcup_{K \xrightarrow{\sigma} N \in \mathcal{S}} \sigma(Th_{\mathcal{S}}(K)) \cup \bigcup_{K \xrightarrow{\sigma} N \in \mathcal{S}} \sigma(\Phi^K) \right]^{\vdash_{\Sigma^N}}$$

For flatable nodes N, $Th_{\mathcal{S}}(N)$ captures N completely. However, this is not the case for nodes that are not flatable. Therefore, we cannot define a proof-theoretic semantics of development graphs as in [AHMS00]. Rather, we have to use a model-theoretic semantics. In Sect. 5 we will show how this model-theoretic semantics relates to the proof theoretic semantics given in [AHMS00].

Definition 8. *Given a node $N \in \mathcal{N}$, its associated class $\mathbf{Mod}_S(N)$ of models (or N-models for short) consists of those Σ^N-models n for which*

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

Complementary to definition 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. Theorem links are the central data structure to represent proof obligations arising in formal developments. Again we distinguish between local and global theorem links (denoted by $N \overset{\sigma}{-\!\!\blacktriangleright} M$ and $N \overset{\sigma}{-\!\!\rightarrow} M$ respectively). We also need theorem links $N \overset{\sigma}{-\!\!\blacktriangleright} \theta M$ (where for some Σ, $\theta \colon \Sigma \longrightarrow \Sigma^N$ and $\sigma \colon \Sigma \longrightarrow \Sigma^M$) involving hiding. The semantics of theorem links is given by the next definition.

Definition 9. *Let $\mathcal{S}$ be a development graph and N, M nodes in $\mathcal{S}$.*

$\mathcal{S}$ **implies** *a global theorem link $N \overset{\sigma}{-\!\!\blacktriangleright} M$ (denoted $\mathcal{S} \models N \overset{\sigma}{-\!\!\blacktriangleright} M$) iff for all $m \in \mathbf{Mod}_S(M)$, $m|_\sigma \in \mathbf{Mod}_S(N)$.*

$\mathcal{S}$ **implies** *a local theorem link $N \overset{\sigma}{-\!\!\rightarrow} M$ (denoted $\mathcal{S} \vdash N \overset{\sigma}{-\!\!\rightarrow} M$) iff for all $m \in \mathbf{Mod}_S(M)$, $m|_\sigma \models \phi$ for all $\phi \in \Phi^N$.*

$\mathcal{S}$ **implies** *a hiding theorem link $N \overset{\sigma}{-\!\!\blacktriangleright} \theta M$ (denoted $\mathcal{S} \models N \overset{\sigma}{-\!\!\blacktriangleright} \theta M$) iff for all $m \in \mathbf{Mod}_S(M)$, $m|_\sigma$ has a θ-expansion to some N-model.*

E.g. consider the development graph of the running example (cf. Fig. 1): The theorem link from SORTING to INSERTSORT expresses the postulation that the latter is a refinement of the former. Furthermore, common proof obligations in a formal development can be encoded into properties that specific global theorem links are implied by the actual development graph.

A global definition link $M \xrightarrow{\sigma} N$ in a development graph is a *conservative extension*, if every M-model can be expanded along σ to an N-model. We will allow to annotate a global definition link as $M \xrightarrow[c]{\sigma} N$, which shall express that it is a conservative extension. These annotations can be seen as another kind of proof obligations.

5 Rules for Development Graphs with Hiding

The rules for theorem proving in development graphs given in [AHMS00] allow to decompose a global theorem link into local theorem links. Unfortunately, it is not possible to decompose global theorem links starting from nodes with hiding definition links going (directly or indirectly) into them. This is because hiding is some kind of existential quantification, and in general it is not possible to decompose an existential quantification of a conjunction into existential quantifications of the conjuncts.

We therefore have to extend the set of rules from [AHMS00] to deal with hiding. We have two kinds of rules:

1. *Rules for hiding and conservative extension.* These rules are suited to push theorem links along the hidings inside the development graph, such that they eventually can be decomposed into local theorem links.
2. *Decomposition rules* from [AHMS00]. They allow to split global theorem links into local and hiding theorem links.

5.1 Rules for Hiding and Conservative Extension

We now come to the rules for hiding and conservative extension. We introduce two rules to shift theorem links over hiding, one dealing with hiding links on the left hand side of a theorem link, and the other one with hiding links on the right hand side of a theorem link.

Since the first rule is quite powerful, we need some preliminary notions. Given a node N in a development graph $\mathcal{S} = \langle \mathcal{N}, \mathcal{L} \rangle$, the idea is that we unfold the subgraph below N into a tree and form a diagram with this tree. More formally, define the *diagram* $D\colon I \longrightarrow \mathbf{Sign}$ *associated with* N together with a map $G\colon |I| \longrightarrow \mathcal{N}$ inductively as follows:

- $\langle N \rangle$ is an object in I, with $D(\langle N \rangle) = N^{\Sigma}$. Let $G(\langle N \rangle)$ be just N.
- if $i = \langle\, M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \,\rangle$ is an object in I with $l_1, \ldots, l_n$ non-local links in $\mathcal{L}$, and $l = K \xrightarrow{\sigma} M$ is a (global or local) definition link in $\mathcal{L}$, then

$$j = \langle\, K \xrightarrow{l} M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \,\rangle$$

is an object in I with $D(j) = \Sigma^K$, and l is a morphism from j to i in I with $D(l) = \sigma$. We set $G(j) = K$.

– if $i = \langle\, M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \,\rangle$ is an object in I with $l_1, \ldots, l_n$ non-local links in $\mathcal{L}$, and $l = K \xrightarrow[h]{\sigma} M$ is a hiding link in $\mathcal{L}$, then

$$j = \langle\, K \xrightarrow{l} M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \,\rangle$$

is an object in I with $D(j) = \Sigma^K$, and l is a morphism from i to j in I with $D(l) = \sigma$. We set $G(j) = K$.

This means that the graph is just unfolded to the diagram. The unfolding is necessary to achieve that in the diagram there is a distinction between instances of the same node that are imported via different paths into another node.

Theorem-Hide-Shift. This rule (cf. Fig. 2) is used if a hiding link occurs on the right-hand side of a theorem link. For this rule $D\colon I \longrightarrow \mathbf{Sign}$ is the diagram associated with N, $\mu_i\colon D(i) \longrightarrow Colim\ D$ are the colimit injections ($i \in |I|$), C is a new isolated node with signature $Colim\ D$, and with ingoing global definition links $G(i) \xrightarrow{\mu_i} C$ for $i \in |I|$. Here, an isolated node is one with no local axioms and no ingoing definition links other than those shown in the rule

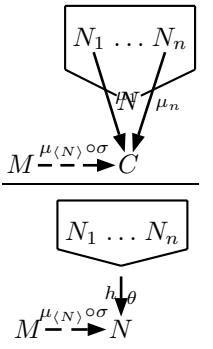
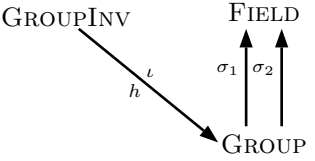
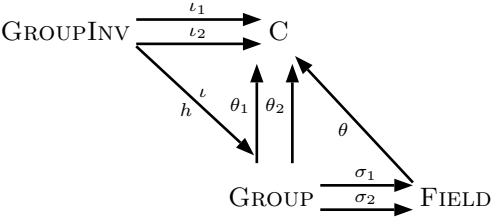


Fig. 2. Rule *Theorem-Hide-Shift*

We now illustrate why the unfolding of the subgraph under N in the rule *Theorem-Hide-Shift* is needed. Consider the development graph



defining groups with the help of groups with inverse (by hiding the inverse) and then defining fields using groups twice: both for the additive and the multiplicative group. If we would take the colimit of this graph, we would identify the additive with the multiplicative group in *Field*. This is not what we want. The unfolding of the rule *Theorem-Hide-Shift* now doubles the signature of groups and groups with inverse, leading to a signature $Colim\ D$ containing the additive and the multiplicative group, and an additive and a multiplicative inverse. The graph for the premise of the rule is then



In the node C, one can reason about both inverses in parallel, while the theory of groups with inverse is not doubled (as it would be the case with approaches that flatten specifications).

Hide-Theorem-Shift. This rule (cf. Fig. 3) replaces hiding theorem links by normal theorem links. This is only possible if on the right-hand side of the hiding theorem link, a conservative definition link

occurs, and furthermore $\sigma' \circ \theta = \theta' \circ \sigma$.

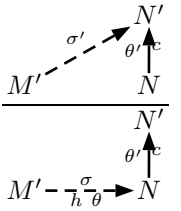


Fig. 3. Rule *Hide-Theorem-Shift*

Cons-Shift. The previous rule to replace hiding theorem links requires conservative definition links. In order to be able to derive new conservative definition links from existing ones, we introduce a rule which allows their derivation. For this rule (cf. Fig. 4) we must require that

$$\begin{array}{ccc} \Sigma^M & \xrightarrow{\sigma} & \Sigma^N \\ \downarrow{\scriptstyle \theta} & & \downarrow{\scriptstyle \theta'} \\ \Sigma^{M'} & \xrightarrow{\sigma'} & \Sigma^{N'} \end{array}$$

is a pushout, and moreover, that N' is isolated.

In addition to the above rules, one would use logic-specific rules for syntactically determining conservative extensions (e.g. extensions by definitions).

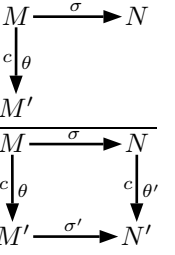


Fig. 4. Rule *Cons-shift*

Proposition 1. *The above rules are sound.*

Proof. Theorem-Hide-Shift: Assume that $\mathcal{S} \models M \xrightarrow{\mu_{\langle N \rangle} \circ \sigma} C$. Let n be an N-model. We have to show $n|_\sigma$ to be an M-model in order to establish the holding of $M \xrightarrow{\sigma} N$. We inductively define a family $(m_i)_{i \in |I|}$ of models $m_i \in \mathbf{Mod}(G(i))$ by putting

- $m_{\langle N \rangle} := n$,
- $m_{\langle K \xrightarrow{l} M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \rangle} := m|_\sigma$, where $l = K \xrightarrow{\sigma} M$ and $m = m_{\langle M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \rangle}$, and
- $m_{\langle K \xrightarrow{l} M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \rangle}$ is a σ-expansion of m to a K-model, where $l = K \xrightarrow[h]{\sigma} M$ and $m = m_{\langle M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \rangle}$.

It is easy to show that this family is consistent with D. By weak amalgamation, there is a $\Sigma^C = Colim\ D$-model c with $c|_{\mu_i} = m_i$. The latter implies that c is a C-model. By the assumption, $c|_{\mu_{\langle N \rangle} \circ \sigma} = m_{\langle N \rangle}|_\sigma = n|_\sigma$ is an M-model.

Hide-Theorem-shift: Assume that $\mathcal{S} \models M' \xrightarrow{\sigma'} N'$ and $N \xrightarrow[c]{\theta'} N'$ is conservative. We have to show that $\mathcal{S} \models M' \xrightarrow{\sigma} \theta N$. Let n be an N-model. Since $N \xrightarrow[c]{\theta'} N'$ is conservative, n can be expanded to an N'-model n' with $n'|_{\theta'} = n$. By

the assumption, $n'|_{\sigma'}$ is an M'-model. Thus, $n'|_{\sigma' \circ \theta} = n'|_{\theta' \circ \sigma} = n|_\sigma$ has a θ-expansion to an M'-model.

Cons-shift: Assume that $M \xrightarrow[c]{\theta} M'$ is conservative. We have to prove that $N \xrightarrow[c]{\theta'} N'$ is conservative as well. Let n be an N-model. Since $M \xrightarrow{\theta} M'$ is conservative, $n|_\sigma$ has a θ-expansion m' being an M'-model. By weak amalgamation, there is some $\Sigma^{N'}$-model n' with $n'|_{\sigma'} = m'$ and $n'|_{\theta'} = n$. Since N' is isolated, n' is an N'-model.

5.2 Rules for Decomposition

The rules for decomposition are taken from [AHMS00]. The rule

Glob-Decomposition has to be changed. It now

decomposes a global theorem link from N to M into local theorem links into M from those nodes from which N is reachable and into hiding theorem links into M from those nodes which are the source of a hiding link going into some node from which N is reachable.

Moreover, we have added a subsumption rule, stating that global reachability suffices to establish a global theorem link.

Glob-Decomposition:

$$\frac{\bigcup_{K \xrightarrow{\sigma'} N} \{K \xrightarrow{\sigma \circ \sigma'} M\} \cup \bigcup_{L \xrightarrow{\theta} K \text{ and } K \xrightarrow{\sigma'} N} \{L \xrightarrow{\sigma \circ \sigma'} \theta M\}}{N \xrightarrow{\sigma} M}$$

Subsumption:

$$\frac{N \xrightarrow{\sigma} M}{N \xrightarrow{\sigma} M}$$

Since the other rules have not changed, we just recall them here for sake of completeness.

Loc-Decomposition I:

$$\frac{K \xrightarrow{\sigma} L}{K \xrightarrow{\sigma''} M} \text{ if } L \xrightarrow{\sigma'} M \text{ and } \sigma''(\Phi(K)) = \sigma'(\sigma(\Phi(K)))$$

Loc-Decomposition II:

$$\frac{}{N \xrightarrow{\sigma} M} \text{ if } N \xrightarrow{\sigma'} M \text{ and } \sigma(\Phi(N)) = \sigma'(\Phi(N))$$

Basic Inference:

$$\frac{Th_{\mathcal{S}}(M) \vdash \sigma(\phi) \text{ for each } \phi \in \Phi^N}{N \xrightarrow{\sigma} M}$$

Given a development graph containing some theorem links that have to be proved, the intention is that the above rules will be applied in a backwards manner as long as possible. The rules for hiding allow to shift the (global) theorem links to nodes without hiding involved, while the rules for decomposition allow to decompose the global theorem links into local ones. With *Basic Inference*, local theorem links can be proved using the entailment relation of the base logic $\mathcal{LOG}$.

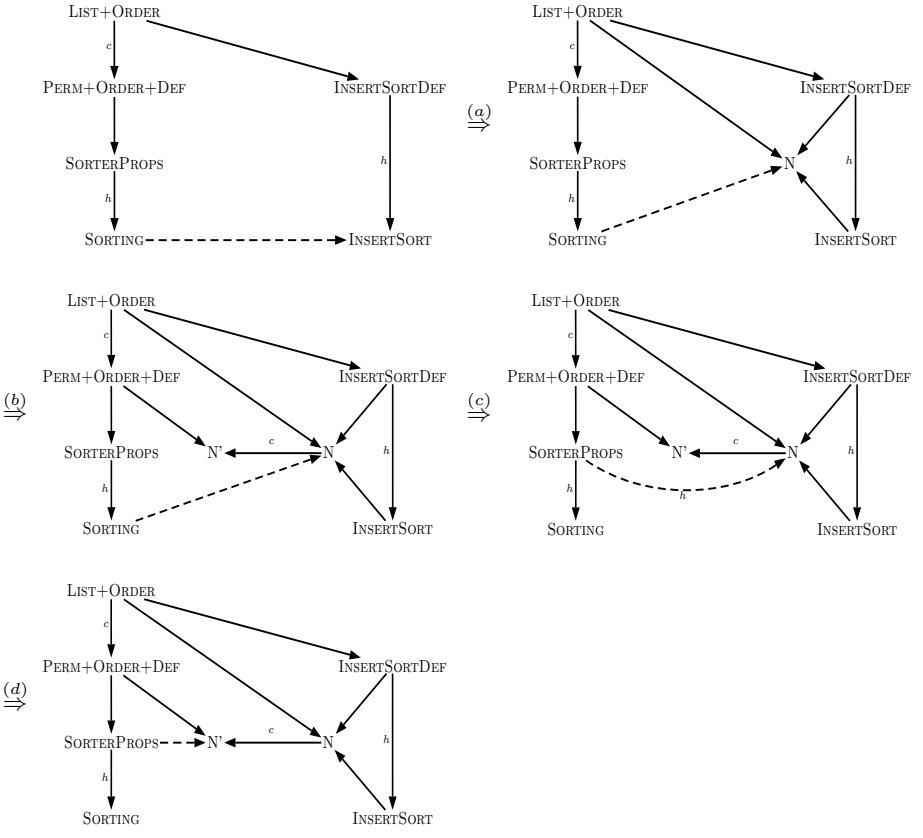


Fig. 5. Reduction of theorem links to *flatable* nodes in the running example.

6 Example

We now demonstrate the (backward manner) use of the rules with the example development graph from Sect. 4. The goal is to reduce the theorem link between Sorting and InsertSort to theorem links between flatable nodes. The derivation is shown in Fig. 5. In the first step (a) the *Theorem-Hide-Shift* rule is applied, which introduces the new node N and the new global definition links. In the second step (b), we infer conservative relationships by applying the rule *Cons-Shift*. This introduces the new node N' and the respective global definition links. Now the theorem link can be reduced to a hiding theorem link from SorterProps to N by *Glob-Decomposition* (step (c)). Finally, this hiding theorem link can be reduced to the theorem link between SorterProps and N' using the rule *Hide-Theorem-Shift*. Since both SorterProps and N' are flatable, one now can use reasoning in the logic to prove the remaining theorem link (via *Basic Inference*).

7 Results about Completeness

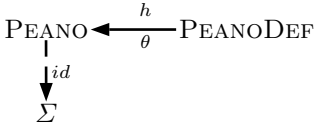
The soundness of our rules is established by proposition 1 for the hiding rules, while showing soundness of the other rules is easy. Another question is the completeness of our rules. We have the following counterexample:

Proposition 2. *Let FOL be the usual first-order logic with a recursively axiomatized complete entailment system. Solving the question whether a global theorem link holds in a development graph with hiding over FOL is not recursively enumerable. Thus, any recursively axiomatized calculus for development graphs with hiding is incomplete.*

Proof. This can be seen as follows. Let Σ be the FOL-signature with a sort *nat* and operations for zero and successor, addition and multiplication and take the usual second-order Peano axioms characterizing the natural numbers uniquely up to isomorphism, plus the defining axioms for addition and multiplication. Without loss of generality, we can assume that these axioms are combined into a single axiom of the form

$$\forall P\colon pred(nat)\ .\ \varphi$$

where φ is a first-order formula. Let ψ be any sentence over Σ. Let $\theta\colon \Sigma \longrightarrow \Sigma'$ add a predicate $P : pred(nat)$ to Σ. Consider the development graph



where Σ and PEANO are nodes with signature Σ and no local axioms, whereas PEANODEF is a node with signature Σ' and local axiom $\varphi \Rightarrow \psi$.

Now we have that PEANO $\xrightarrow{id}$ Σ holds iff each $\Sigma - model$ has a PEANODEF-expansion. It is easy to see that this holds iff the second-order formula $\exists P : pred(nat).\varphi \Rightarrow \psi$ is valid. By the quantifier rules for prenex normal form, this is equivalent to $\forall P : pred(nat).\varphi \models \psi$, i.e. equivalent to the fact that ψ holds in the second-order axiomatization of Peano arithmetic. By Gödels incompleteness theorem, this question is not recursively enumerable. □

In spite of this negative result, there is still the question of a relative completeness w.r.t. a given oracle deciding conservative extensions. Such a completeness result has been proved by Borzyszkowski [Bor01] in a similar setting. We are going to prove a similar result here. We first need a preparatory lemma:

Lemma 1. *If C is a flatable node or if C is constructed as in the rule Theorem-Hide-Shift, then any Σ^C-model satisfying $Th_{\mathcal{S}}(C)$ is already a C-model.*

Proof. If C is flatable, the result follows by an easy induction over the definition links indirectly or directly going into C. Now let C be constructed as

in the rule *Theorem-Hide-Shift*. We use the notation introduced in connection with the construction of C. For $i \in |I|$, let $len(i)$ be the length of the path i, and let p be the maximum of all $len(i)$, $i \in |I|$. We prove by induction over $p - len(i)$ that for all $i \in |I|$, $c|_{\mu_i}$ is a $G(i)$-model. Since C contains no local axioms, the result then follows. Let $i = \langle\, M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \,\rangle \in |I|$. By induction hypothesis, for each ingoing link $K \xrightarrow{l} M$, $c|_{\mu_j}$ is a K-model for $j = \langle\, K \xrightarrow{l} M \xrightarrow{l_1} \cdots \xrightarrow{l_n} N \,\rangle$. Now if $l = K \xrightarrow{\sigma} M$, $c|_{\mu_i \circ \sigma} = c|_{\mu_j}$, while if $l = K \xrightarrow[h]{\sigma} M$, $c|_{\mu_j \circ \sigma} = c|_{\mu_i}$. In the former case, $c|_{\mu_i}$ reduces to a K-model, while in the latter case, $c|_{\mu_i}$ expands to a K-model. Hence in both cases, the link l is satisfied by $c|_{\mu_i}$. Since c satisfies $Th_{\mathcal{S}}(C)$ and $N = G(i) \xrightarrow{\mu_i} C$, $c|_{\mu_i}$ also satisfies the local axioms in M. Hence, $c|_{\mu_i}$ is a model of $M = G(i)$. □

Theorem 1. *Assume that the underlying logic $\mathcal{LOG}$ is complete. Then the rule system for development graphs with hiding is complete relative to an oracle for conservative extensions.*

Proof. Assume $\mathcal{S} \models M \overset{\sigma}{\dashrightarrow} N$. We show that there is some faithful extension $\mathcal{S}_1$ of $\mathcal{S}$ (i.e. new nodes and new definition links are added, but the latter go only into new nodes) such that $\mathcal{S}_1 \vdash M \overset{\sigma}{\dashrightarrow} N$.

Let $D\colon I \longrightarrow \mathbf{Sign}$ and C be as in the rule *Theorem-Hide-Shift*, and let c be a Σ^C-model satisfying $Th_{\mathcal{S}}(C)$. By Lemma 1, c is a C-model. Hence, $c|_{\mu_{\langle N \rangle}}$ is an N-model, and by the assumption $\mathcal{S} \models M \overset{\sigma}{\dashrightarrow} N$, $c|_{\mu_{\langle N \rangle} \circ \sigma}$ is an M-model. We now have for any $K \xrightarrow{\theta} M$:

1. $c|_{\mu_{\langle N \rangle} \circ \sigma \circ \theta} \models \Phi^K$. By the satisfaction condition for institutions, we get $c \models \mu_{\langle N \rangle}(\sigma(\theta(\Phi^K)))$. By completeness of the underlying logic, we get $Th_{\mathcal{S}}(C) \vdash \mu_{\langle N \rangle}(\sigma(\theta(\Phi^K)))$. By *Basic Inference*, $\mathcal{S} \vdash K \overset{\mu_{\langle N \rangle} \circ \sigma \circ \theta}{\dashrightarrow} C$.
2. For $L \xrightarrow[h]{\tau} K$, form the pushout

$$\begin{array}{ccc} \Sigma^K & \xrightarrow{\mu_{\langle N \rangle} \circ \sigma \circ \theta} & \Sigma^C \\ \downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \tau'} \\ \Sigma^L & \xrightarrow{\mu'} & \Sigma' \end{array}$$

   and obtain a new development graph $\mathcal{S}'$ from $\mathcal{S}$ by introducing a new node L' with signature Σ' and two ingoing definition links $L \xrightarrow{\mu'} L'$ and $C \xrightarrow{\tau'} L'$. The latter link is conservative: for any C-model c_1, $c_1|_{\mu_{\langle N \rangle} \circ \sigma \circ \theta}$ has a τ-expansion to an L-model c_2, and by weak amalgamation, there is some Σ'-model c_3 with $c_3|_{\tau'} = c_1$ and $c_3|_{\mu'} = c_2$, which is hence an L'-model. By the oracle for conservativity, we get $C \xrightarrow[c]{\tau'} L'$. Now $\mathcal{S}' \vdash L \overset{\mu'}{\dashrightarrow} L'$ by *Subsumption*. By *Hide-Theorem-Shift*, we get $\mathcal{S}' \vdash L \overset{\mu_{\langle N \rangle} \circ \sigma \circ \theta}{\dashrightarrow} \tau C$.

Let $\mathcal{S}_1$ be the union of all the $\mathcal{S}'$ constructed in step 2 above (assuming that all the added nodes are distinct). By *Glob-Decomposition*, we get $\mathcal{S}_1 \vdash M \overset{\mu_{\langle N \rangle} \circ \sigma}{\dashrightarrow} C$. By *Theorem-Hide-Shift*, we get $\mathcal{S}_1 \vdash M \overset{\sigma}{\dashrightarrow} N$. □

Corollary 1. *If $\mathcal{LOG}$ is complete, the decomposition rules are complete for proving theorem links between flatable nodes.*

Proof. By inspecting the proof of Theorem 1, one can see that for theorem links between flatable nodes, case 2 is never entered, and thus neither the rules *Cons-Shift* and *Hide-Theorem-Shift* nor the oracle for conservativeness are needed. Since Lemma 1 also holds for flatable nodes, one can replace the node C in the proof of Theorem 1 with the node N, thus also avoiding the use of *Theorem-Hide-Shift*. □

8 Conclusion and Related Work

We have extended the notion of development graph [AHMS00] to deal also with hiding. We have developed a semantics and a proof system for development graphs with hiding. The proof system can easily shown to be sound.

Concerning completeness, with a counterexample, we show that there can be no complete recursively axiomatized proof system for development graphs with hiding. However, we have shown our proof rules to be complete relative to a given oracle for detecting conservative extensions. We thus have achieved the same degree of completeness as Borzyszkowski [Bor01] rule system for structured specifications. In one sense our system is more complete than Borzyszkowski's: since our *Theorem-Hide-Shift* rule simulates something like Borzyszkowski's normal forms, we do not have to rely on the Craig interpolation property. For example, it is possible to solve a counterexample showing incompleteness in case of failure of interpolation in [Bor01] with our rules. Borzyszkowski refrains from doing normal form inference because with his way of computing normal forms, the structure of the specification is lost. Note that this is not the case with our rules, since they just extend the structure of the development graph, while the axioms are kept locally. One can even further optimize the rule *Theorem-Hide-Shift* by reducing the constructed diagram to those nodes that are really necessary, which can be achieved by taking the so-called final subcategory [AR94].

Compared with the rules in [Bor01], we have fewer but more complex rules. Our rules involve colimit computations that may be tedious for humans using the rules directly, but that are no problem for computer assisted proofs. Indeed, by exploiting the graph structure, development graphs with hiding can lead to much more efficient proofs than possible when using the usual proof rules for structured specifications as in [Bor01].

Moreover, we expect no difficulty when extending the management of change developed in [AHMS00] to the case of development graphs with hiding in order to integrate the presented approach into the development graph system of INKA 5.0 [AHMS99]. Then it will be possible to support the maintenance of changes in developments by machine also when hiding are present.

In [AHMS00], we have described a translation from CASL structured specifications to development graphs. It should be straightforward to extend this translation to structured specifications with hiding.

References

[AHMS00] S. Autexier, D. Hutter, H. Mantel, and A. Schairer. Towards an evolutionary formal software-development using CASL. In C. Choppy, D. Bert, and P. Mosses, (Eds.), *Recent Developments in Algebraic Development Techniques, 14th International Workshop, WADT'99*, Bonas, France, LNCS 1827, page 73–88, Springer-Verlag, 2000.

[AHMS99] S. Autexier, D. Hutter, H. Mantel, A. Schairer. System Description: Inka 5.0 - A Logic Voyager. In H. Ganzinger, (Ed.), *Proceedings of CADE-16*, Trento, Italy, LNAI 1632, Springer-Verlag, 1999.

[AR94] J. Adámek, J. Rosický. *Locally Presentable and Accessible Categories*, Cambridge University Press, 1994.

[Bor01] T. Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*. To appear.

[CASL98] CoFI Language Design Task Group. *The Common Algebraic Specification Language (*Casl*) – Summary*, Version 1.0 and additional Note S-9 on Semantics, available from `http://www.brics.dk/Projects/CoFI`, 1998.

[CM97] M. Cerioli, J. Meseguer. May I borrow your logic?, *Theoretical Computer Science*, 173:311–347, 1997.

[DGS91] R. Diaconescu, J. Goguen, P. Stefaneas. Logical support for modularization, In G. Huet, G. Plotkin, (Eds), *Workshop on Logical Frameworks*, 1991.

[GB92] 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.

[HWB97] R. Hennicker, M. Wirsing, M. Bidoit. Proof systems for structured specifications with observability operators, *Theoretical Computer Science*, 173(2):393–443, 1997.

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

[Mes89] J. Meseguer. General logics, In *Logic Colloquium 87*, pages 275–329, North Holland, 1989.

[MKK97] T. Mossakowski, Kolyang, B. Krieg-Brückner. Static semantic analysis and theorem proving for CASL, In *12th Workshop on Algebraic Development Techniques*, Tarquinia, LNCS 1376, pages 333–348, Springer-Verlag, 1998.

[ST88] D. Sannella, A. Tarlecki. Specifications in an arbitrary institution, *Information and Computation*, 76(2-3):165-210, 1988.

[ST92] D. Sannella, A. Tarlecki. Towards Formal Development of Programs from Algebraic Specifications: Model-Theoretic Foundations, *19th ICALP*, Springer, LNCS 623, pages 656–671, Springer-Verlag, 1992.

[SMHKT01] L. Schröder, T. Mossakowski, P. Hoffman, B. Klin, and A. Tarlecki. Semantics for architectural specifications in CASL, In H. Hußmann, (Ed.), *Proceedings of Fundamental Approaches to Software Engineering (FASE 2001)*, Genova, Italy, 2001.

[VSE96] D. Hutter et. al. Verification Support Environment (VSE), *Journal of High Integrity Systems*, Vol. 1, pages 523–530, 1996.