Model-Level vs Theory-Level Semantics

Till Mossakowski [1] Florian Rabe [2] Mihai Codescu [1]

[1]DFKI GmbH Bremen and University of Bremen

[2]Jacobs University Bremen

12.09.2009, Udine, IFIP WG 1.3 Meeting

Introduction

Two different semantics for structured specifications:

- model-level semantics: the semantics of a specification SP is a signature $Sig(SP)$ and a class of models $Mod(SP)$ over $Sig(SP)$
- theory-level semantics: the semantics of a specification is a signature $Sig(SP)$ and a set of sentences $Th(SP)$ over $Sig(SP)$

Both semantics are easily reconciled if there is no hiding (and also no freeness), because in this case:

$$Mod(SP) = Mod(Th(SP))$$

However, in presence of hiding, this equation does not hold! (examples: later)

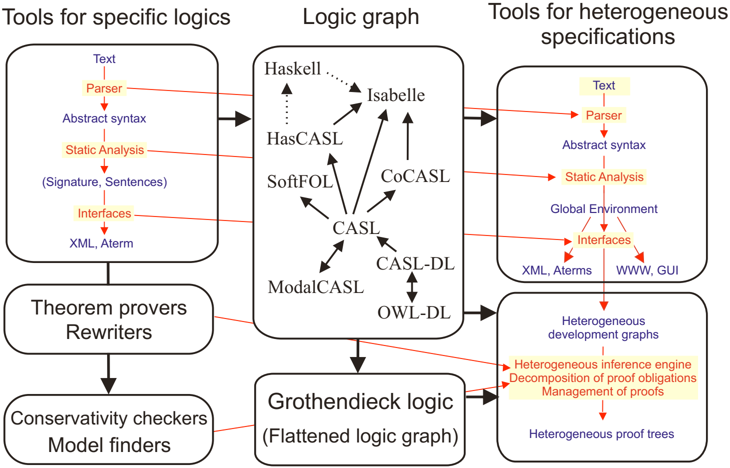
Some history

- model-level semantics
  - ASL (Sannella, Wirsing 1983)
  - specification in an arbitrary institution (Sannella, Tarlecki 1988)
  - algebraic specification languages (CASL, 1990's ff)
- theory-level semantics
  - Clear (Goguen, Burstall 1980)
  - structured theories, conservative extensions and interpolation (Maibaum, Dimitrakos, Veloso 1980's ff.)
  - hidden information modules over inclusive institutions (Goguen, Roşu 2004)
  - MathML, OpenMath, OMDoc (Kohlhase et al. 2000's)
  - ontologies and description logics (Wolter, Lutz 2000's)

Motivation of the talk

- Report on recent developments of heterogeneous tool set (which relies on model-level semantics)
- Report on recent developments of OMDoc/MMT (which relies on theory-level semantics)
- Discuss the pros and cons
- Can both semantics be reconciled?

Architecture of the heterogeneous tool set Hets



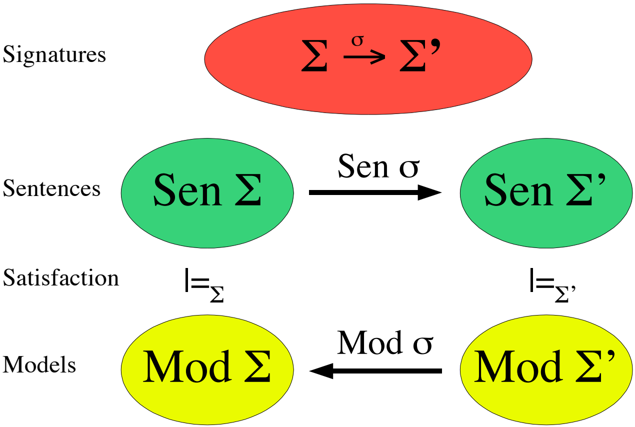
Logics currently supported by Hets

CASL	many-sorted first-order logic, partial functions, subsorting, datatypes (induction)
CoCASL	coalgebraic specification of reactive systems
ModalCASL	first-order modal logic
HasCASL	higher order logic, polymorphism, type classes
Haskell	pure functional programming language
CspCASL	combination of CASL with the process algebra CSP
OWL DL	description logic (DL) fragment of Web Ontology Language (OWL)
Maude	rewriting logic with preorder algebra semantics
VSE	a dynamic logic with Pascal-like programs
RelScheme	Relational schemes
Propositional	classical propositional logic
SoftFOL	softly typed first-order logic ($\Rightarrow$ TPTP)
Isabelle	Isabelle's higher-order logic

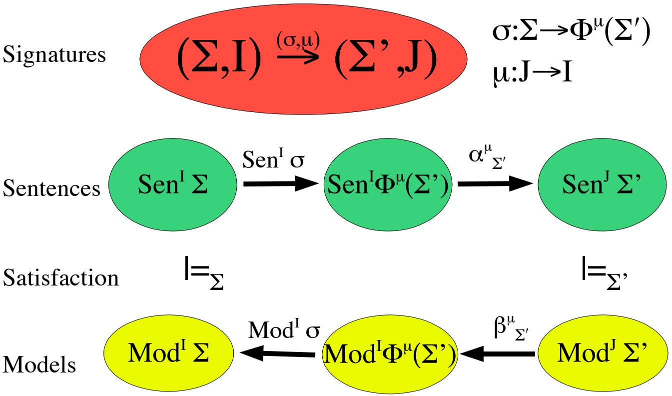
Sound Integration of Heterogeneity

- logics are formalized as institutions (Goguen, Burstall 1984)
- logic translations are formalized as institution (co)morphisms (Goguen, Rosu 2002)
- logic translations embed or encode logical structure in a way that truth is preserved
- Grothendieck logic = flat combination of the logics in a logic graph (Diaconescu 2002)
- Hets provides an object-oriented interface for plugging in institutions and (co)morphisms

Institutions



The Grothendieck Institution



Syntax of Structured Specifications

SP ::=	BASIC-SPEC	basic specification
	| SP then SP	extension
	| SP and SP	union
	| SP with SYMBOL-MAP	renaming
	| SP hide SYMBOLS	hiding
	| SPEC-NAME [PARAM*]	reference to named spec

LIBRARY-ITEM ::=

spec SPEC-NAME [PARAM*] = SP end — name a spec

| view VIEW-NAME : SP to SP = SYMBOL-MAP end

refinement between specifications

Syntax of Structured Specifications

SP ::= BASIC-SPEC
| SP then SP
| SP and SP
| SP with SYMBOL-MAP
| SP hide SYMBOLS
| SPEC-NAME [PARAM*]

LIBRARY-ITEM ::=
spec SPEC-NAME [PARAM*] = SP end
| view VIEW-NAME : SP to SP = SYMBOL-MAP end

Syntax of Heterogeneous Specifications

SP ::= BASIC-SPEC | logic LOGIC-NAME : {SP}
| SP then SP
| SP and SP
| SP with SYMBOL-MAP | SP with logic COMORPHISM
| SP hide SYMBOLS | SP hide logic MORPHISM
| SPEC-NAME [PARAM*]

LIBRARY-ITEM ::=
spec SPEC-NAME [PARAM*] = SP end
| view VIEW-NAME : SP to SP = SYMBOL-MAP end
| view VIEW-NAME : SP to SP = SYMBOL-MAP, COMORPHISM
| logic LOGIC-NAME

Structured specifications over an arbitrary institution

$$SP ::= \langle \Sigma, \Gamma \rangle \ \mid SP \cup SP \ \mid \sigma(SP) \ \mid \sigma^{-1}(SP)$$

. . . and their semantics

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

$$Sig(SP_1 \cup SP_2) = Sig(SP_1) = Sig(SP_2)$$
$$Mod(SP_1 \cup SP_2) = Mod(SP_1) \cap Mod(SP_2)$$

$$Sig(\sigma\colon \Sigma_1 \longrightarrow \Sigma_2(SP)) = \Sigma_2$$
$$Mod(\sigma(SP)) = \{M \in Mod(\Sigma_2) \mid M|_\sigma \in Mod(SP)\}$$

$$Sig((\sigma\colon \Sigma_1 \longrightarrow \Sigma_2)^{-1}(SP)) = \Sigma_1$$
$$Mod((\sigma\colon \Sigma_1 \longrightarrow \Sigma_2)^{-1}(SP)) = \{M|_\sigma \mid M \in Mod(SP)\}$$

Heterogeneous Development Graphs

Heterogeneous structured specifications are mapped into heterogeneous development graphs:

- nodes correspond to individual specification modules
- definition links correspond to imports of modules
- theorem links express proof obligations

Development graphs

- are a tool for management and reuse of proofs
- have already proved to scale to industrial applications (cf. verification support environment VSE)

Development graphs $\mathcal{S} = \langle \mathcal{N}, \mathcal{L} \rangle$

Nodes in $\mathcal{N}$: (Σ^N, Γ^N) with

- Σ^N signature,
- $\Gamma^N \subseteq \mathbf{Sen}(\Sigma^N)$ set of local axioms.

Links in $\mathcal{L}$:

- global $M \xrightarrow{\sigma} N$, where $\sigma : \Sigma^M \to \Sigma^N$,
- local $M \xrightarrow{\sigma} N$ where $\sigma : \Sigma^M \to \Sigma^N$, or
- hiding $M \xrightarrow[h]{\sigma} N$ where $\sigma : \Sigma^N \to \Sigma^M$ *going against the direction of the link.*

Semantics of development graphs

$Mod_S(N)$ consists of those Σ^N-models n for which

1. n satisfies the local axioms Γ^N,
2. for each $K \xrightarrow{\sigma} N \in \mathcal{S}$, $n|_\sigma$ is a K-model,
3. for each $K \xrightarrow{\sigma} N \in \mathcal{S}$, $n|_\sigma$ satisfies the local axioms Γ^K,
4. 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.

Theorem links

Theorem links come, like definition links, in different versions:

- global theorem links $M \overset{\sigma}{- - - - - \succ} N$, where $\sigma : \Sigma^M \longrightarrow \Sigma^N$,
- local theorem links $M \overset{\sigma}{- - - - - \succ} N$, where $\sigma : \Sigma^M \longrightarrow \Sigma^N$

Semantics of theorem links

- $\mathcal{S} \models M \overset{\sigma}{\dashrightarrow} N$ iff for all $n \in Mod_S(N)$, $n|_\sigma \in Mod_S(M)$.
- $\mathcal{S} \models M \overset{\sigma}{\dashrightarrow} N$ iff for all $n \in Mod_S(N)$, $n|_\sigma \models \Gamma^M$.

Proof Calculus

Theorem

The proof calculus for heterogeneous development graphs is sound and complete relative to an oracle checking conservative extensions.

- decompose global theorem links semi-automatically into local ones
- choose specific provers for local proof goals

Reachability of nodes

Global reachability

$M \xrightarrow{\sigma}\!\!\twoheadrightarrow N$ is defined inductively and holds iff

- either $M = N$ and $\sigma = id$, or
- $M \xrightarrow{\sigma'} K \in \mathcal{S}$, and $K \xrightarrow{\sigma''}\!\!\twoheadrightarrow N$, with $\sigma = \sigma'' \circ \sigma'$.

Local reachability

$M \rightarrowtail\!\!\xrightarrow{\sigma}\!\!\twoheadrightarrow N$ iff $M \xrightarrow{\sigma}\!\!\twoheadrightarrow N$ or there is a node K with $M \xrightarrow{\sigma'} K \in \mathcal{S}$ and $K \xrightarrow{\sigma''}\!\!\twoheadrightarrow N$, such that $\sigma = \sigma'' \circ \sigma'$.

Flattenable Nodes in a Development Graph

A node N in a development graph is called flattenable if for any node M with incoming hiding definition links, N is not reachable from M.

The theory of a flattenable node N is defined as

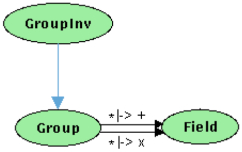
$$Th(N) = \Gamma^N \cup \bigcup_{K \xrightarrow{\sigma} N} \sigma(Th(K)) \bigcup_{K \overset{\sigma}{\rightarrowtail} N} \sigma(\Gamma^K)$$

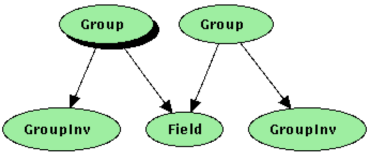
and captures the node N completely.

For non-flattenable nodes, we compute normal forms.

Normal Forms

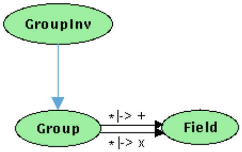
The normal form of a non-flattenable node (in our example Field) is computed by unfolding its subgraph to be able to distinguish between instances of the same node imported via different paths to another node

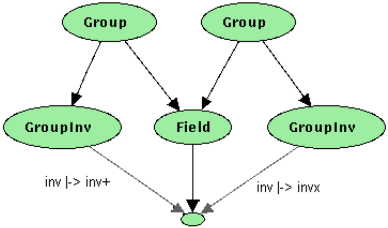




Normal Forms

The normal form of a non-flattenable node (in our example Field) is computed by unfolding its subgraph to be able to distinguish between instances of the same node imported via different paths to another node and then computing the colimit of the resulting diagram.





Normal forms

Theorem

Let $\sigma : N \to nf(N)$ the inclusion morphism from a node N to its normal form. Under weak amalgamability assumptions, $Mod(N) = Mod(nf(N))|_{\sigma}$.

Weakly amalgamable pushouts

For a heterogeneous diagram of the shape

$$(L_1, \Sigma_1) \xleftarrow{(c_1,\phi_1)} (L, \Sigma) \xrightarrow{(c_2,\phi_2)} (L_2, \Sigma_2)$$

we look for a logic where we can compute a weakly amalgamable cocone.

$$\begin{tikzcd}
 & L' & \\
L1 \arrow[ur, dotted, "e_1"] \arrow[r, phantom, "\overset{u1}{\Longrightarrow}"] & {} \arrow[r, phantom, "\overset{u2}{\Longleftarrow}"] & L2 \arrow[ul, dotted, "e_2"'] \\
 & L \arrow[ul, "c_1"] \arrow[ur, "c_2"'] \arrow[uu, dotted, "d"] &
\end{tikzcd}$$

Conditions:

- the logic L' has weakly amalgamable pushouts;
- the comorphisms are weakly amalgamable;
- the square is weakly amalgamable

Weakly amalgamable pushouts

For a heterogeneous diagram of the shape

$$(L_1, \Sigma_1) \xleftarrow{(c_1,\phi_1)} (L, \Sigma) \xrightarrow{(c_2,\phi_2)} (L_2, \Sigma_2)$$

we look for a logic where we can compute a weakly amalgamable cocone.

$$\begin{tikzcd}
& L' & \\
L1 \arrow[ur, dotted, "e_1"] \arrow[r, phantom, "\overset{u1}{\Longrightarrow}"] & & L2 \arrow[ul, dotted, "e_2"'] \arrow[l, phantom, "\overset{u2}{\Longleftarrow}"] \\
& L \arrow[ul, "c_1"] \arrow[ur, "c_2"'] \arrow[uu, dotted, "d"] &
\end{tikzcd}$$

Conditions:

- the logic L' has weakly amalgamable pushouts;
- the comorphisms are weakly amalgamable;
- the square is weakly amalgamable

An Ontology Example

logic OWL
spec FoodOntology =
Class: *Human*
SubClassOf: *eats some Food*
Class: *Plant*
SubClassOf: *grows_in some Area*
Class: *Vegetarian*
SubClassOf: *Healthy*
Class: *FoodAndPlant*
EquivalentTo: *Food and Plant*
Class: *FoodAndPlant*
SubClassOf: *Vegetarian*
Class: *PlantEater*
EquivalentTo: *Human and eats only Plant*

An Ontology Example

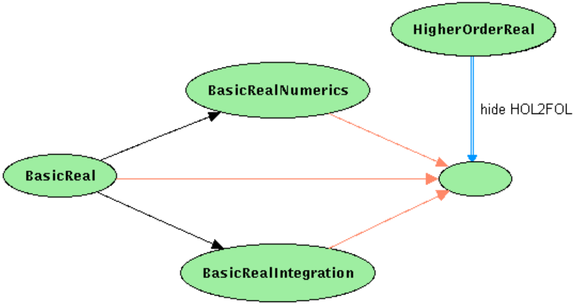
logic OWL
spec FoodOntology =
Class: *Human*
SubClassOf: *eats some Food*
Class: *Plant*
SubClassOf: *grows_in some Area*
Class: *Vegetarian*
SubClassOf: *Healthy* **%%(Wolter and Lutz)**
Class: *FoodAndPlant*
EquivalentTo: *Food and Plant*
Class: *FoodAndPlant*
SubClassOf: *Vegetarian*
Class: *PlantEater*
EquivalentTo: *Human and eats only Plant*

An Ontology Example

spec FOODONTOLOGYHIDE =
    FOODONTOLOGY **hide** *Food*
spec FOODONTOLOGYGOAL =
    FOODONTOLOGYHIDE
then %implies
    Class: *PlantEater*
    SubClassOf: *eats some Vegetarian*

First-order Specification of Real Numbers

(Roggenbach, Schröder, Mossakowski - WADT 1999)
axiomatization of the weak theory of real numbers.



A Heterogeneous Refinement

Consider a specification *SortSpec* of sorting written in CASL and a sorting program *SortProg* given in the institution of programming languages PLNG. To express in HETS that *SortProg* is an implementation of *SortSpec*

$$\mathit{logicCASL} : \mathit{SortSpec} \rightsquigarrow \mathit{logicPLNG} : \mathit{SortProg}$$

we use a heterogeneous view

$$\textbf{view}\ \textsc{correctness} : \textsc{SortSpec}\ \textbf{to}\ \{\textsc{SortProg}\ \textbf{hide}\ \mathit{toCASL}\}$$

which translates to

$$\beta(\mathit{Mod}^{\mathrm{PLNG}}(\mathit{SortProg})) \subseteq \mathit{Mod}^{\mathrm{CASL}}(\mathit{SortSpec})$$

A Heterogeneous Refinement

We can encode the semi-morphism $\mathit{toCASL} = (\Phi, \beta)$ as a span of comorphisms $\mathit{PLNG} \xleftarrow{\mathit{toCASL}^-} \textsc{Casl} \circ \Phi \xrightarrow{\mathit{toCASL}^+} \mathit{CASL}$:

$$
\begin{array}{ccccc}
\mathbf{Sign}^{\mathit{PLNG}} & \overset{id}{\leftarrow} & \mathit{Sign}^{\mathit{PLNG}} & \overset{\Phi}{\rightarrow} & \mathit{Sign}^{\textsc{Casl}} \\
\mathit{Sen}^{\mathit{PLNG}} & \overset{incl}{\leftarrow} & \emptyset & \overset{incl}{\rightarrow} & \mathit{Sen}^{\textsc{Casl}} \circ \Phi \\
\mathit{Mod}^{\mathit{PLNG}} & \overset{\beta}{\rightarrow} & \mathit{Mod}^{\textsc{Casl}} \circ \Phi^{op} & \overset{id}{\leftarrow} & \mathit{Mod}^{\textsc{Casl}} \circ \Phi^{op}
\end{array}
$$

A Heterogeneous Refinement

We find a weakly amalgamable square for the span $\mathit{PLNG} \xleftarrow{\mathit{toCASL}^-} \mathrm{CASL} \circ \Phi \xrightarrow{\mathit{toCASL}^+} \mathrm{CASL}$ by coding both PLNG and CASL to higher order logic:

$$
\begin{tikzcd}
 & \mathit{HOL} & \\
\mathit{PLNG} \arrow[ur, dotted, "\mathit{PLNG2HOL}"] \arrow[r, phantom, "\overset{id}{\Longrightarrow}"] & {} \arrow[r, phantom, "\overset{\theta}{\Longleftarrow}"] & \mathrm{CASL} \arrow[ul, dotted, "\mathit{CASL2HOL}"'] \\
 & \mathrm{CASL} \circ \Phi \arrow[ul, "\mathit{toCASL}^-"] \arrow[ur, "\mathit{toCASL}^+"'] \arrow[uu, dotted] &
\end{tikzcd}
$$

Notice that we are not commited to higher order logic but we could use instead any other logic with the same properties (e.g. rewriting logic).

A Heterogeneous Refinement

We obtain thus a weakly amalgamable square

$$
\begin{tikzcd}
 & (\mathrm{C}\textsc{asl} \circ \Phi, \Sigma_p) \arrow[dl, "(toCASL^{-}{,}id)"'] \arrow[dr, "(toCASL^{+}{,}id)"] & \\
(\mathrm{PLNG}, \Sigma_P) \arrow[dr, "(PLNG2HOL{,}id)"'] & & (\mathrm{C}\textsc{asl}, \Sigma_S) \arrow[dl, "(CASL2HOL{,}\theta_{\Sigma_S})"] \\
 & (HOL, PLNG2HOL(\Sigma_P)) &
\end{tikzcd}
$$

A Heterogeneous Refinement

The problem gets reformulated in *HOL* as

$$Mod^{HOL}(PLNG2HOL(SortProg)) \subseteq Mod^{HOL}(\theta(CASL2HOL(SortSpec)))$$

which comes to proving in *HOL* that

$$PLNG2HOL(SortProg) \vdash \theta(CASL2HOL(SortSpec))$$

Background MMT

- Developed with Michael Kohlhase in knowledge management group at Jacobs University Bremen
- Arose from efforts to consolidate
  - proof theoretical and model theoretical approaches to logic
  - logical and knowledge management approaches to mathematics
- Designed as web-scalable representation language of logical knowledge
- Forms kernel of currently developed OMDoc 2 language (Open Mathematical DOCuments)

Hets vs. MMT: Differences in spirit

- Focus
  - Hets: specifying and proving
  - MMT: focus on representation and web-scalability
- Ontological assumptions
  - Hets: institutions
  - MMT: foundation-independent
- Semantics
  - Hets: model level
  - MMT: theory level

MMT focus

- Interface between formal systems and web services
  - services *should* be implemented generically
    by people who do not know formal systems!
  - services *can* be implemented generically
  - standard compliance: XML, URI, OpenMath/MathML, OMDoc
- Modularity-aware interface between different formal systems
  - systems differ strongly in their ontological foundations
  - yet structuring mechanisms very similar: specification languages, type theories, programming languages

Foundation-independence

- Variety of foundations: set theories, type theories
- Variety of logical frameworks: model theory, proof theory
- MMT approach: foundations and logical frameworks represented as theories
- Possible by weak definition of theory
  - MMT theories are lists of symbol declarations
  - symbols may have types or definitions
  - no type system
- Still strong enough to define structured theory development

Model- vs. Theory-based Semantics

- Theory-based: semantics of structured theory is flat theory
  - constructive elimination of structuring concepts
    only way to be foundation-independent
  - works well except for hiding
- Model-based: semantics of structured theory is model class
  - requires model theoretic semantics of base language
    difficult for type theories
  - elegant framework using institutions
    but some people do not (want to) use them
  - elegant treatment of hiding

MMT Syntax Overview

- Theory level: theory graphs
- Symbol level
  - declarations of symbols in theories
  - maps of symbols in theory morphisms
- Object level: generic application and binding
  - MMT defines *structural well-formed*ness
  - MMT parametric in external definition of *well-typed*ness

Meta-Theories

- Meta-relation between theories
- Theory `lf` represents Edinburgh logical framework
- Theory `zfc` represents set theory
- Theory `fol` represents first-order logic, view v its semantics

$$
\begin{tikzcd}
 & & \texttt{lf} \arrow[dl, hook, "\text{meta}"'] \arrow[dr, hook, "\text{meta}"] & & \\
 & \texttt{fol} \arrow[rr, dashed, "v"] \arrow[dl, hook, "\text{meta}"'] \arrow[dr, hook, "\text{meta}"] & & \texttt{zfc} \arrow[dr, hook, "\text{meta}"] & \\
\texttt{ring} & & \texttt{monoid} \arrow[ll] \arrow[rr, dashed, "v_1"] & & \texttt{integers}
\end{tikzcd}
$$

Semantics of Flat Theories

Semantics induced by commitment to semantics of a meta-theory

Examples:

- Typing/entailment for LF induces typing/entailment for other nodes done: implemented MMT plugin for LF
- Model theory for LF induces model theory for other nodes done: institution for LF
- Ignore LF, give semantics for FOL and HOL separately done by Hets

$$
\begin{tikzcd}
 & & \mathtt{lf} \arrow[dl, hook, "\mathrm{meta}"'] \arrow[dr, hook, "\mathrm{meta}"] & \\
 & \mathtt{fol} \arrow[dl, hook, "\mathrm{meta}"'] \arrow[dr, hook, "\mathrm{meta}"] & & \mathtt{hol} \\
\mathtt{ring} & & \mathtt{monoid} \arrow[ll] &
\end{tikzcd}
$$

Semantics of Flat Theories

Semantics induced by commitment to semantics of a meta-theory

Examples:

- Typing/entailment for LF induces typing/entailment for other nodes done: implemented MMT plugin for LF
- Model theory for LF induces model theory for other nodes done: institution for LF
- Ignore LF, give semantics for FOL and HOL separately done by Hets

Goal

- give meta-theory LFIns (and semantics for it)
- use it to formalize institutions within MMT
- integrate that into Hets

$$
\begin{tikzcd}
 & & \mathtt{lf} \arrow[dl, hook, "\text{meta}"'] \arrow[dr, hook, "\text{meta}"] & \\
 & \mathtt{fol} \arrow[dl, hook, "\text{meta}"'] \arrow[dr, hook, "\text{meta}"] & & \mathtt{hol} \\
\mathtt{ring} & & \mathtt{monoid} \arrow[ll] &
\end{tikzcd}
$$

Structured Theories

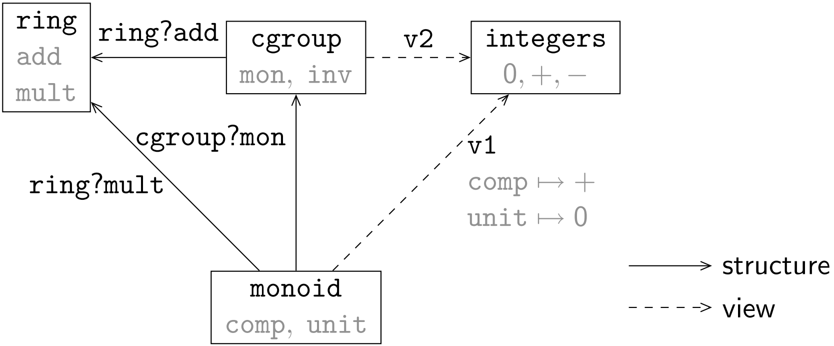
- Theory graph of theories and theory morphisms
- Three kinds of theory morphisms
  - inclusions (special case: meta-theories)
  - structures instantiate theories (also called: definitional link, import)
    import of theory S into theory T induces theory morphism $S \to T$
  - views translate between existing theories (also called: postulated link, theorem link)

Theory Graph Example

v2

$$\left\{ \begin{array}{l} \texttt{mon/comp} \mapsto + \\ \texttt{mon/unit} \mapsto 0 \end{array} \right\} \quad \text{or} \quad \texttt{mon} \mapsto \texttt{v1}$$

$\texttt{inv} \mapsto -$



Hets vs. MMT: Differences in Formal Details

- Imports unnamed in Hets, named in MMT
- Curry-Howard-based representation of axioms and theorems in MMT
- Only primitive hiding and partial views in MMT

Named vs. Unnamed Imports

- Named imports common in type theories and programming languages e.g., SML functors
- Less common in algebraic specification not present in OBJ, CASL, development graphs
- Advantages of named imports
  - multiple import of the same theory without need for renaming
  - concrete syntax for reference to theory morphism induced by structure
  - morphisms instantiate symbols with terms and structures with morphisms yields concrete syntax for decomposition of theorem links

Curry-Howard-Representation

- Axioms and theorems are named
- Axiom a asserting F represented as $a : \mathit{true}\ F$
- Theorem t with proof p asserting F represented as $a = p : \mathit{true}\ F$
- No loss of generality
- Operations and axioms/theorems treated uniformly

  blurs difference between signatures and theories

Hiding

- Syntax: morphism maps symbol to special term $\top$
- Strictness: symbols depending on hidden symbols are hidden
- Semantics of structures: hidden symbols are dropped
- Semantics of views: views induce partial mappings
  yields concrete syntax for local links
- MMT hiding more like deleting/forgetting

Use cases 1, 2

1) Hiding auxiliary symbols
   - avoids cluttering namespace in systems with unnamed imports
   - possible in Hets
   - not crucial in systems with named imports like MMT
2) Hiding defined symbols
   - replace with definition
   - possible in MMT and Hets

Use case 3

Hiding inexpressible parts

- permits to represent partial translations
- consequences of hidden axioms should *not* be hidden (if possible)
- possible in Hets

Example:

- import from HOL-Reals hiding higher-order axioms to obtain FOL-Reals
- Idea for MMT: FOL-expressible theorems (with non-FOL-expressible proofs) translated to axioms

Use case 4

Hiding axiomatized symbols

- possible in Hets
- possible in MMT if symbol is defined using choice/description operator
- general case awkward in MMT

Example:

- Hets: axiomatize concatenation c of lists by giving FOL-axioms a_{nil} and a_{cons}
- Idea for MMT
  1. define c using implicit definition
  2. hide c and replace with description operator $\mathit{the}\ c.(a_{nil} \wedge a_{cons})$
  3. replace $\mathit{the}\ x.P(x)$ with skolem constant c and axiom $c = \mathit{the}\ x.P(x)$
  4. apply axiom scheme $P(\mathit{the}\ x.P(x))$
  5. drop choice operators
  6. resulting FOL-theory is equivalent to Hets normal form

Use case 5

Hiding implementation details

- permits changes to hidden symbols without affecting visible interface
- theorems using hidden axioms should not be available
- current behavior of MMT
- not possible in Hets: hidden information not visible but still there
- generalization: proofs using hidden axioms become proofs with gaps

Example:

1. define real numbers in set theory
2. $\mathbb{R}$, 0, etc. are defined constants, properties are theorems
3. hide underlying set theory to obtain theory in which $\mathbb{R}$, 0, etc. have no definitions, properties are axioms
4. for division implemented as total function, hide axiom $1/0 = 0$

Idea for Reconciliation

- Call MMT hiding a different name, e.g., deleting or forgetting
- Add syntax for model-theoretical hiding to MMT
- MMT semantics of structured theory: pair of flat theory with visible interface given by subtheory
- Akin to Hidden Information Modules by Goguen, Rosu
- Given model theory for meta-theory (e.g., for `fol`), model-theoretical semantics of hiding can be recovered from MMT semantics