Hets/Ontohub Tutorial

Mihai Codescu Till Mossakowski Christian Maeder
Oliver Kutz Fabian Neuhaus

Otto-von-Guericke Universität Magdeburg

16.05.2014, Osnabrück, Coinvent Meeting

Overview

1. Presentation of HETS
2. HETS demo
3. Ontohub demo

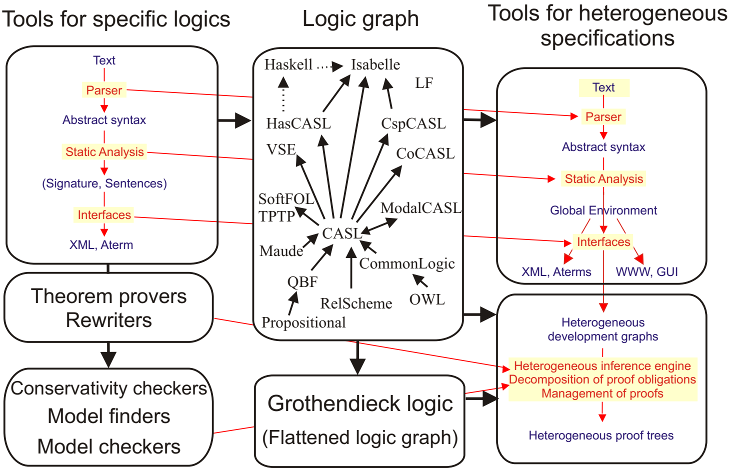
The *Heterogenous Tool Set* (Hets)

- **a tool integration platform for heterogeneous specification**
- connects state-of-the art provers
- parsers and static analysis tools for logical formalisms
- Hets understands
  - logic-specific module languages
  - DOL: logic-independent, heterogeneous module language
- logic-independent proof management

The *Heterogenous Tool Set* (Hets)

- uniform but heterogeneous semantics via the Grothendieck institution
- easy way to plug-in new formalisms and translations
- systematic connection of new formalisms to tools using translations
- currently 25 logics, 50 translations, 20 tools

Architecture of the heterogeneous tool set Hets



Logics supported by Hets

- General-purpose logics

Classical propositional logic	Propositional, QBF
First-order logic	SoftFOL / TPTP
Datatypes	CASL: multi-sorted first-order logic with subsorts, partial functions and datatypes
Higher-order logic	THF0, HasCASL (polymorphism, type classes)

- logical frameworks

Higher-order logic	Isabelle, HOL-light
Dependent types	DFOL, LF

- Ontologies

OWL	Web Ontology Language (OWL)
Common Logic	ISO-Standard for specification of ontologies
Relational Database Schemas	RelScheme

Logics supported by Hets (ctd.)

- Reactive and parallel systems

  Process algebra CspCASL
  Coalgebra CoCASL (Coalgebraic specification of reactive systems)
  First-order modal and temporal logic ModalCASL
  Rewriting Maude

- Programming languages

  First-order dynamic logic VSE (Pascal-like programs)
  Functional programming languages Haskell, FPL, Adl

- Tool-specific logics

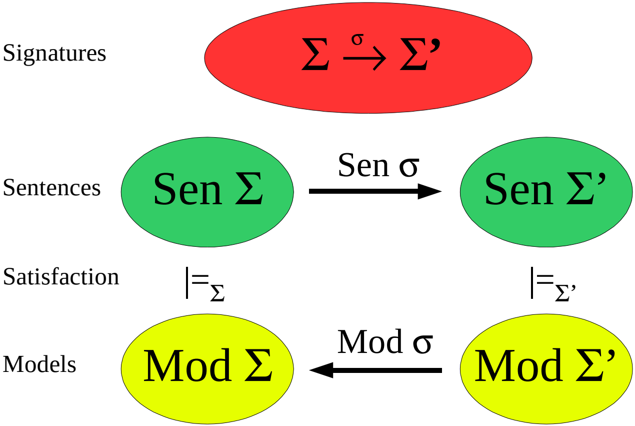
  Computer algebra systems EnCL (for *reduce*)
  CAD-Systems DMU, FreeCAD (ABoxes for the CAD-Systems CATIA and FreeCAD)

- planned: **UML, Java, JML**

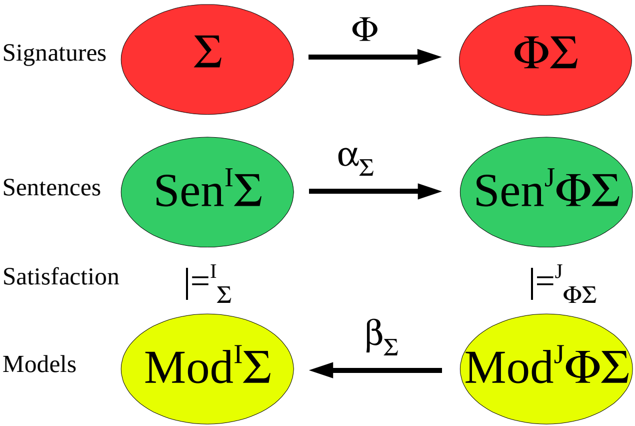
Sound Integration of Heterogeneity

- 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

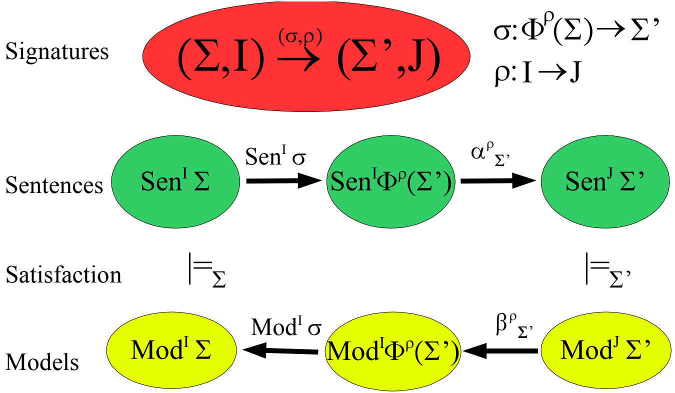
Institutions



Institution comorphisms



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
	| combine DIAGRAM	colimit

LIBRARY-ITEM ::=

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

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

refinement between specifications

| diagram DNAME = SPEC-NAME*, VIEW-NAME*, DNAME*
excluding SPEC-NAME*, VIEW-NAME*, DNAME*

diagrams

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

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.

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 $M \overset{\sigma}{\dashrightarrow} N$ where $\sigma : \Sigma^M \to \Sigma^N$, or
- hiding $M \xrightarrow[hide]{\sigma} N$ 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 $K \overset{\sigma}{\dashrightarrow} N \in \mathcal{S}$, $n|_\sigma$ satisfies the local axioms Γ^K,
4. for each $K \xrightarrow[hide]{\sigma} N \in \mathcal{S}$, 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 $M \xrightarrow{\sigma} N$, where $\sigma : \Sigma^M \longrightarrow \Sigma^N$,
- local theorem links $M \overset{\sigma}{\dashrightarrow} N$, where $\sigma : \Sigma^M \longrightarrow \Sigma^N$

Semantics of theorem links

- $\mathcal{S} \models M \xrightarrow{\sigma} N$ 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 under mild technical assumptions.

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