The Heterogeneous Tool Set, HETS ⋆

Till Mossakowski[1], Christian Maeder[1], and Klaus Lüttich[2]

[1] DFKI Lab Bremen and Department of Computer Science, University of Bremen, Germany
[2] SFB/TR 8 and Department of Computer Science, University of Bremen, Germany

1 Introduction

Heterogeneous specification becomes more and more important because complex systems are often specified using multiple viewpoints, involving multiple formalisms (see Fig. 1). Moreover, a formal software development process may lead to a change of formalism during the development.

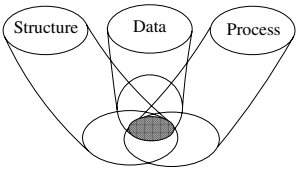

Fig. 1. Multiple viewpoints

Some of the current heterogeneous approaches deliberately stay informal, like UML. Current formal integration approaches have the drawback that they are uni-lateral in the sense that typically there is one logic (and one theorem prover) which serves as the central integration device, even if this central logic may not be needed or desired in particular applications.

By contrast, the heterogeneous tool set is a both flexible, multi-lateral *and* formal (i.e. based on a mathematical semantics) integration tool, providing parsing, static analysis and proof management for heterogeneous multi-logic specifications by combining various tools for individual specification languages. Unlike other tools, it treats logic translations (e.g. codings between logics) as first-class citizens. The architecture of the heterogeneous tool set is shown in Fig. 2. In the sequel, we will explain the details of this figure.

2 Logics in Hets

The notion of *institution* [2] captures in a very abstract and flexible way the essence of a logical system. *Institution morphisms* or *comorphisms* relate institutions.

In HETS, each logic (institution) is realized in the programming language Haskell [7] by a list of types (e.g. for signatures, signature morphisms, sentences) and functions (e.g. for parsing, static analysis and theorem proving, see the left column of Fig. 2). In Haskell jargon, the interface is called a multiparameter type class with functional dependencies.

The following logics have been integrated in HETS so far, with varying degree of support (see the middle column of Fig. 2 and [4,1] for more details and references).

CASL [1] extends many sorted first-order logic with partial functions and subsorting. It also provides induction sentences, expressing the (free) generation of datatypes.

⋆ This work has been supported by the *Deutsche Forschungsgemeinschaft* unders grants KR 1191/5-2 and KR 1191/7-2 and in the project I4-SPIN in the SFB/TR8 "Spatial Cognition". We thank Stefan Wölfl for providing the first heterogeneous verification example.

O. Grumberg and M. Huth (Eds.): TACAS 2007, LNCS 4424, pp. 519–522, 2007.
© Springer-Verlag Berlin Heidelberg 2007

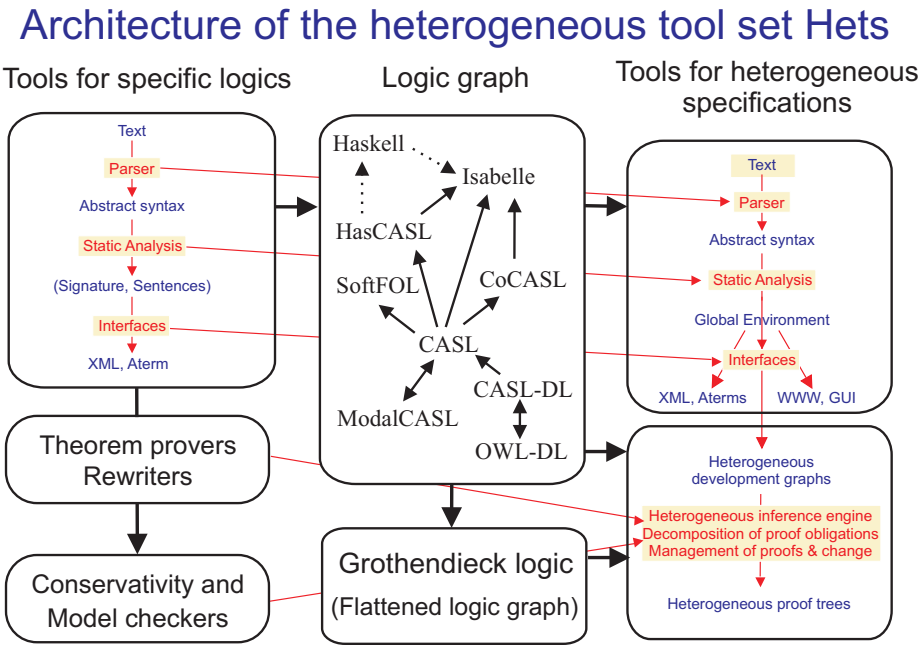

Fig. 2. Architecture of the heterogeneous tool set

CoCASL is a coalgebraic extension of CASL, suited for the specification of process types and reactive systems. The central proof method is coinduction.

ModalCASL is an extension of CASL with multi-modalities and term modalities. It allows the specification of modal systems with Kripke's possible worlds semantics.

HasCASL is a higher order extension of CASL allowing polymorphic datatypes and functions, closely related to the programming language Haskell.

Haskell [7] is a modern, pure and strongly typed functional programming language.

OWL DL is the Web Ontology Language (OWL) recommended by the World Wide Web Consortium (W3C, `http://www.w3c.org`). It is used for knowledge representation and the Semantic Web.

CASL-DL is an extension of a restriction of CASL, realizing a strongly typed variant of OWL DL in CASL syntax.

SoftFOL [3] offers three automated theorem proving systems (ATP) for first-order logic with equality: (1) SPASS [9]; (2) Vampire [8]; and (3) MathServe Broker [10]. These together comprise some of the most advanced theorem provers for first-order logic.

Isabelle [6] is an interactive theorem prover for higher-order logic, and (jointly with others) marks the frontier of current research in interactive higher-order provers.

3 Heterogeneous Specification

Heterogeneous specification is parameterized over some arbitrary graph of logics (institutions) and logic translations (comorphisms). The graph of currently supported logics

is shown in Fig. 2. However, this graph is just a parameter: indeed, the HETS modules implementing the logic graph can be compiled independently of the HETS modules implementing heterogeneous specification, and this separation of concerns is essential to keep the tool manageable from a software engineering point of view.

Heterogeneous CASL (HETCASL; see [4]) includes the structuring constructs of CASL, such as union and translation. A key feature of CASL is that syntax and semantics of these constructs are formulated over an arbitrary institution (i.e. also for institutions that are possibly *completely different* from first-order logic resp. the CASL institution). HETCASL extends this with constructs for choosing the current logic and translating specifications along logic translations (i.e. comorphisms).

4 Proof Management

The central device for structured theorem proving and proof management in HETS is the formalism of *heterogeneous development graphs* [5,4]. Development graphs have been used for large industrial-scale applications with hundreds of specifications. They also support management of change. The graph structure provides a direct visualization of the structure of specifications and the open proof obligations.

The *proof calculus* for development graphs [5,4] is given by rules that allow for decomposing proof obligations into simpler ones, until they can be proved by turning them into *local proof goals*. The latter can be discharged using a logic-specific theorem prover. This can be done using a graphical user interface (GUI), which allows for selecting the prover and the subset of axioms that is sent to the prover. Also, provers

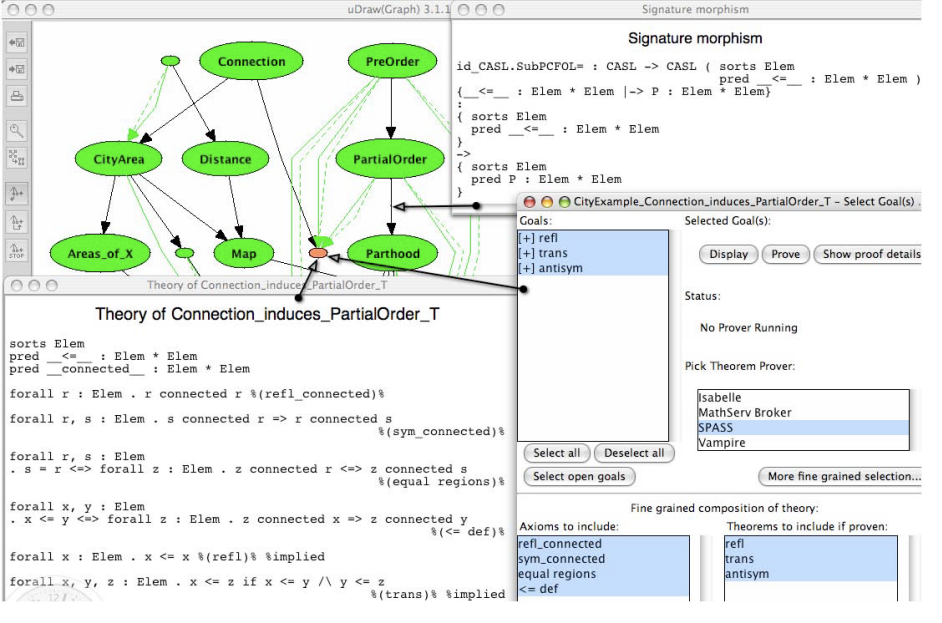

Fig. 3. A sample HETS session

for other logics than that of the current theory may be used, if there is a comorphism linking the two logics. In this way, theorem provers can be *borrowed* for other logics (e.g. a first-order prover can be also used for modal first-order logic). A typical session with HETS is shown in Fig. 3.

5 Conclusion

The Heterogeneous Tool Set (HETS) is available at `http://www.cofi.info/Tools`. A sample heterogeneous proof concerns the correctness of the composition table of a qualitative spatial calculus. This involves two different provers and logics: an automated first-order prover solving the vast majority of the goals, and an interactive higher-order prover used to prove a few bridge lemmas. The corresponding heterogeneous specification is found under `Calculi/Space/RCCVerification.het` in the repository available at `http://www.cofi.info/Libraries`.

It may appear that HETS just provides a combination of some first-order provers and Isabelle. But already now, HETS provides proof support for modal logic (via the translation to CASL, and then further to either SPASS or Isabelle), as well as for COCASL. Hence, it is quite easy to provide proof support for new logics by just implementing logic translations, which is at least an order of magnitude simpler than integrating a theorem prover. Future work will integrate more logics (such as CSP-CASL and other process calculi) and interface more existing theorem proving tools (such as CSP-Prover) with specific institutions in HETS, and provide more sample applications.

References

1. CoFI (The Common Framework Initiative). CASL *Reference Manual*. LNCS 2960 (IFIP Series). Springer, 2004.
2. 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.
3. K. Lüttich and T. Mossakowski. Reasoning Support for CASL with Automated Theorem Proving Systems. WADT 2006, Springer LNCS, to appear.
4. T. Mossakowski. Heterogeneous specification and the heterogeneous tool set. Habilitation thesis, University of Bremen, 2005.
5. T. Mossakowski, S. Autexier, and D. Hutter. Development graphs – proof management for structured specifications. *Journal of Logic and Algebraic Programming*, 67(1-2):114–145, 2006.
6. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Springer Verlag, 2002.
7. S. Peyton-Jones, editor. *Haskell 98 Language and Libraries — The Revised Report*. Cambridge, 2003. also: J. Funct. Programming **13** (2003).
8. A. Riazanov and A. Voronkov. The design and implementation of VAMPIRE. *AI Communications*, 15(2-3):91–110, 2002.
9. C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobalt, and D. Topic. SPASS version 2.0. In Andrei Voronkov, editor, *Automated Deduction – CADE-18*, LNCS 2392, pages 275–279. Springer-Verlag, 2002.
10. J. Zimmer and S. Autexier. The MathServe System for Semantic Web Reasoning Services. In U. Furbach and N. Shankar, editors, *3rd IJCAR*, LNCS 4130. Springer, 2006.