Openproof - A Flexible Framework for Heterogeneous Reasoning

Dave Barker-Plummer[1], John Etchemendy[1], Albert Liu[1], Michael Murray[1], and Nik Swoboda[2]

[1] Stanford University
Stanford, CA, 94305-4101, USA
[2] Universidad Politécnica de Madrid
Boadilla del Monte, Madrid, 28660, Spain

Abstract. In this paper we describe the Openproof heterogeneous reasoning framework. The Openproof framework provides support for the implementation of heterogeneous reasoning environments, i.e., environments for writing arguments or proofs involving a number of different kinds of representation. The resulting environments are in a similar spirit to our *Hyperproof* program, though the Openproof framework goes beyond *Hyperproof* by providing facilities for the inclusion of a variety of representation systems in the same environment. The framework serves as the core of a number of widely used educational programs including *Fitch*.

1 Introduction

In [1,2,3] our group pioneered the notion of *formal heterogeneous deduction*: formally specified inference systems in which different representations are used in concert to reach conclusions. This work resulted in the implementation of *Hyperproof* [4], the first proof checker for a heterogeneous logic.

Hyperproof employs two representations: the sentential representation of first-order logic (FOL), and a diagrammatic representation consisting of blocks on a checkerboard. Our theory, however, is general and does not depend on these two specific representations. The Openproof framework abstracts over the specific representations that are employed in the deduction. The framework can be used to create heterogeneous (and homogeneous) proof environments for any combination of representations by "plugging in" implementations of the specific representations. Aspects of this general architecture have been described in [5,6,7,8] though this is the first description of the software implementation.

The Openproof framework is in use as the core of the *Fitch* application for homogeneous deduction in FOL [9]. Additional Openproof modules are currently being built for Block's World, Euler/Venn [10], Position, and Coincidence Grid diagrams [11], and for sentences in plain text. Here we will briefly describe the different components of the Openproof framework.

G. Stapleton, J. Howse, and J. Lee (Eds.): Diagrams 2008, LNAI 5223, pp. 347–349, 2008.

2 Framework

The Openproof framework consists of three tiers:

- **Kernel** – the Openproof kernel provides services including the loading of components, saving and restoring files, and interfacing with the operating system and Java environment.
- **Tools** – the second tier consists of tool kits which can be used to build modules and provide intra-module and inter-module communication. Examples of tools provided by the framework include a tool kit to build basic diagrammatic editors, and facilities for integrating changes in representations into a proof.
- **Modules** – the third tier provides component-level interfaces for specifying the particular kinds of modules that can be combined into a heterogeneous proof environment. There are two kinds of modules, representation modules and proof modules. Representation modules have a common structure consisting of:
  - Representations in one of two flavors: sentential and diagrammatic.
  - Editors to enable users to construct and manipulate those representations.
  - Inference engines to support two kinds of reasoning: homogeneous and heterogeneous. (Homogeneous engines involve only one representation, while heterogeneous engines involve more than one.)

  Proof modules also have a common structure consisting of:
  - Editors to allow users to build and change proofs.
  - Proof engines to support the logic behind the proof system.

Thus, the Openproof framework provides core services to allow the automatic generation of heterogeneous reasoning systems. At the moment the framework does not provide support for the construction of single proof environments involving more than one proof system at the same time, but this is a possibility which opens a number of interesting theoretical and practical questions.

2.1 The Role of *interlingua*

By design, the framework does not have a single common language into which all representations in the system must be translated in order to perform or check inferences, i.e., an *interlingua*. As discussed in [2], it is our belief that heterogeneous rules of inference need not be defined upon the basis of a translation into an interlingua. Following this philosophy gives greater latitude to module designers when thinking about the design of new components. In general, not requiring an interlingua simplifies the design of homogeneous rules of inference (as they can be defined directly from one instance of a representation system to another without having to pass through a second kind of representation), and allows the possibility for a range of heterogeneous rules to co-exist simultaneously.

While we do not require the use of interlingua, the framework supports the partial or total use of *implicit* and *explicit* interlingua in particular heterogeneous

systems. By implicit, we mean the use of an interlingua to define heterogeneous rules of inference (a use which is transparent to the reasoner). For example, when defining rules of inference in an Euler diagram module, internally the rules could translate the Euler diagram into a Venn diagram and then rely upon already defined Venn rules. By explicit, we mean the use of a single representation as the "go-between" for a number of others. An example of this would be a system in which exchanging information between any two different representations requires translating the first into FOL and then translating from FOL into the second.

3 Conclusion

The Openproof framework will serve as a tool to free researchers interested in heterogeneous reasoning systems from many of the mundane tasks involved in developing such systems and thereby allow them to focus on the design of individual representations and heterogeneous relations between those modules.

References

1. Barwise, J., Etchemendy, J.: Information, infons and inference. In: Situation Theory and Its Applications, pp. 33–78. CSLI Publications, Stanford (1990)
2. Barwise, J., Etchemendy, J.: Heterogeneous logic. In: Logical reasoning with diagrams, pp. 179–200. Oxford University Press, New York (1996)
3. Barwise, J., Etchemendy, J.: Visual information and valid reasoning. In: Logical reasoning with diagrams, pp. 3–25. Oxford University Press, New York (1996)
4. Barwise, J., Etchemendy, J.: Hyperproof. CSLI Publications, Stanford (1994)
5. Barker-Plummer, D., Etchemendy, J.: Visual decision making: A computational architecture for heterogeneous reasoning. In: Kovalerchuk, B., Schwing, J. (eds.) Visual and Spatial Analysis: Advances in Data Mining, Reasoning and Problem Solving, pp. 79–109. Springer, Berlin (2004)
6. Barker-Plummer, D., Etchemendy, J.: A computational architecture for heterogeneous reasoning. Journal of Theoretical and Experimental Artificial Intelligence 19(3), 195–225 (2007)
7. Barker-Plummer, D., Etchemendy, J.: Applications of heterogeneous reasoning in design. Machine Graphics and Vision 12(1), 39–54 (2003)
8. Swoboda, N., Allwein, G.: Modeling heterogeneous systems. In: Hegarty, M., Meyer, B., Narayanan, N.H. (eds.) Diagrams 2002. LNCS (LNAI), vol. 2317, pp. 131–145. Springer, Heidelberg (2002)
9. Barwise, J., Etchemendy, J., Allwein, G., Barker-Plummer, D., Liu, A.: Language Proof and Logic. CSLI Publications, University of Chicago Press, Stanford (1999)
10. Swoboda, N., Allwein, G.: Heterogeneous reasoning with euler/venn diagrams containing named constants and fol. Electronic Notes in Theoretical Computer Science 134, 153–187 (2005)
11. Barker-Plummer, D., Swoboda, N.: A sequent based logic for coincidence grid. In: CEUR Workshop Proceedings, vol. 274, pp. 1–12 (2007)