dlvhex: A Prover for Semantic-Web Reasoning under the Answer-Set Semantics*

Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits
Technische Universität Wien, Institut für Informationssysteme 184/3,
Favoritenstrasse 9-11, A-1040 Vienna, Austria
{eiter, ianni, roman, tompits}@kr.tuwien.ac.at

Abstract

We present the system dlvhex*, a solver for* HEX*-programs, which are nonmonotonic logic programs admitting both* higher-order atoms *as well as* external atoms. *Higher-order features are widely acknowledged as being useful for various tasks, including meta-reasoning. Furthermore, the possibility to exchange knowledge with external sources in a fully declarative paradigm such as* answer-set programming (*ASP*) *becomes increasingly important, in particular in view of applications in the Semantic-Web area. Through external atoms,* HEX*-programs can deal with external knowledge and reasoners of various nature, such as RDF datasets or description-logics knowledge bases.*

1 Introduction

Nonmonotonic semantics is often requested by Semantic-Web designers in cases where the reasoning capabilities of the *ontology layer* of the Semantic Web turn out to be too limiting, since they are based on monotonic logics. The widely acknowledged *answer-set semantics* of nonmonotonic logic programs [5], which gives rise to the *answer-set programming* (ASP) paradigm, is a good candidate for supplying nonmonotonic semantics to the *rules*, *logic*, and *proof layers* in the Semantic Web.

However, for important issues such as *meta-reasoning* in the context of the Semantic Web, no adequate answer-set engines have been available so far. Motivated by this fact and the observation that interoperability with other software is another important issue (not only in this context), in previous work [4], the answer-set semantics has been extended to HEX-*programs*, which are *higher-order* logic programs (which accommodate meta-reasoning through *higher-order atoms*) with *external atoms* for software interoperability. Intuitively, a *higher-order atom* allows to quantify values over predicate names, and to freely exchange predicate symbols with constant symbols, like, for instance, in the rule $C(X) \leftarrow subClassOf(D, C), D(X)$. An *external atom* facilitates to determine the truth value of an atom through an external source of computation. For instance, the rule $t(Sub, Pred, Obj) \leftarrow \&rdf[in](Sub, Pred, Obj), uri(in)$ computes the predicate t taking values from the predicate $\&rdf$. This latter predicate extracts RDF statements from the URI specified by in; this task is delegated to an external computational source (e.g., an external deduction system, an execution library, etc.). External atoms allow a bidirectional flow of information to and from external sources of computation such as description-logics reasoners.

By means of HEX-programs, powerful meta-reasoning becomes available in a decidable setting, e.g., for Semantic-Web applications, for meta-interpretation in ASP itself, or for defining policy languages. For example, advanced closed-world reasoning or the definition of constructs for an extended ontology language (e.g., of RDF(S)) is well-supported. Due to the higher-order features, the representation is succinct. The experimental prototype dlvhex implements the language of HEX-programs, and is based on a reduction to ordinary ASP.

Other logic-based formalisms, like TRIPLE [8] or F-Logic [7], feature also higher-order predicates for meta-reasoning in Semantic Web applications. Our formalism is fully declarative and offers the possibility of nondeterministic predicate definition in a decidable setting.

2 HEX-Programs

HEX-programs are sets of rules of the form

$$\alpha_1 \vee \cdots \vee \alpha_k \leftarrow \beta_1, \ldots, \beta_n, not\, \beta_{n+1}, \ldots, not\, \beta_m,$$

where $m, k \geq 0$, $\alpha_1, \ldots, \alpha_k$ are *higher-order atoms*, and $\beta_1, \ldots, \beta_m$ are either *higher-order atoms* or *external atoms*; the operator "*not*" is *default negation*. An external atom is of the form $\&g[Y_1, \ldots, Y_n](X_1, \ldots, X_m)$, where $Y_1, \ldots, Y_n$ and $X_1, \ldots, X_m$ are two lists of terms (called

*This work was partially supported by the Austrian Science Fund (FWF) under grant P17212-N04, as well as by the European Commission through the IST Networks of Excellence REWERSE (IST-2003-506779).

input list and *output list*, respectively), and $\&g$ is an *external* predicate name. A higher-order atom (or simply *atom*) is a tuple $Y_0(Y_1, \dots, Y_n)$, where $Y_0, \dots, Y_n$ are terms.

The semantics of HEX-program is given by generalizing the answer-set semantics [4], which admits no, one, or multiple models (i.e., answer sets) in general.

An interesting application scenario where several features of HEX-programs come into play is *ontology alignment* [2]. In order to perform the latter, HEX-programs allow, for instance, to import external theories and translate reified assertions, such as in the following way:

$$\begin{aligned} tripleY(X, Z) &\leftarrow \&rdf[uri](X, Y, Z);\\ proposition(P) &\leftarrow triple(P, rdf{:}type, rdf{:}Statement);\\ C(X) &\leftarrow (X, rdf{:}type, C). \end{aligned}$$

Another interesting application is to apply the closed-world assumption (CWA) and default reasoning in a controlled fashion. Assuming that a generic external atom $\&DL[C](X)$ is available for querying the concept C in a given description-logic knowledge base, the CWA principle can be stated as follows:

$$C'(X) \leftarrow not\,\&DL[C](X), concept(C), cwa(C, C'),$$

where $concept(C)$ is a predicate which holds for all concepts, and $cwa(C, C')$ states that C' is the CWA of C.

3 Implementation

The evaluation principle of dlvhex is to split the program according to its dependency graph into components and alternately call the answer-set solver DLV [6] and the external atom functions for the respective subprograms. The framework takes care of traversing the tree of components in the right order and combining their resulting models.

External atoms are embedded as *plug-ins*, i.e., libraries that define and provide one or more external atoms. Such plug-ins are implemented as shared libraries, which link dynamically to the main application at runtime. A lean, object-oriented interface reduces the effort of developing custom plug-ins to a minimum.

Currently, dlvhex provides the following extension to pure HEX-reasoning: (i) parsing both templates as well as frame syntax by using DLT [3] as a preparser; (ii) in addition to strict constraints, accepting *weak constraints* for optimization problems; and (iii) returning the result in XML syntax according to the RuleML specification [1].

The *RDF plug-in* provides a single external atom, which enables the user to import RDF triples from any RDF knowledge base. It takes a single constant as input, which denotes the RDF source (a file path or Web address).

To query knowledge bases in the well-known description logic $\mathcal{SHOIN}(\mathbf{D})$ (i.e., OWL-DL ontologies), we developed the *description-logic plug-in*, which includes a number of external atoms to query the extensions of concepts and roles as well as to check the knowledge base for consistency. Each such external atom can extend the knowledge base prior to submitting the query by means of the atoms' input parameters.

No adequate string manipulation routines for constants has been provided by ASP engines so far. The *string plug-in* provides several atoms for such tasks, such as $\&strstr$, which tests two strings for substring inclusion, or the $\&concat$ atom, which lets the user specify two strings in the input list and returns their concatenation as output value.

Another notable plug-in has been developed for interfacing the WordNet-database and thus provides means for integrating lexical reasoning in an ASP framework.

A Web-interface for dlvhex can be found at

`http://www.kr.tuwien.ac.at/research/dlvhex/`,

along with a more detailed documentation of all external atoms. Moreover, the system, its plug-ins, and a toolkit for developing custom plug-ins are publicly available as source packages.

References

[1] H. Boley, S. Tabet, and G. Wagner. Design Rationale for RuleML: A Markup Language for Semantic Web Rules. In *Proc. SWWS 2001*, pages 381–401, 2001.

[2] D. Calvanese, G. De Giacomo, and M. Lenzerini. A Framework for Ontology Integration. In *Proc. SWWS-2001*, pages 303–316, 2001.

[3] G. Ianni, G. Ielpa, A Pietramala, M. C. Santoro and F. Calimeri. Enhancing Answer Set Programming with Templates, In *Proc. NMR 2004*, pages 233–239, 2004.

[4] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. A Uniform Integration of Higher Order Reasoning and External Evaluations in Answer Set Programming. In *Proc. IJCAI 2005*, pages 90–96, 2005.

[5] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[6] Leone, N., *et al*: The DLV System for Knowledge Representation and Reasoning. *ACM TOCL*, to appear.

[7] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.

[8] M. Sintek and S. Decker. TRIPLE - A Query, Inference, and Transformation Language for the Semantic Web. In *Proc. ISWC-2002*, pages 364–378.