DLV-HEX: Dealing with Semantic Web under Answer-Set Programming*

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

Abstract

We present an implementation of HEX programs, which are nonmonotonic logic programs admitting *higher-order atoms* as well as *external atoms*. Higher-order features are widely acknowledged as useful for various tasks, including meta-reasoning. Furthermore, the possibility to exchange knowledge with external sources in a fully declarative framework such as *answer-set programming* (ASP) is nowadays 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-logic knowledge bases.

1 Introduction

A 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 [Gelfond and Lifschitz, 1991], which gives rise to answer-set programming (ASP), is a good candidate host for giving 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, furthermore, interoperability with other software is an important issue (not only in this context), Eiter *et al.* [2005] extended the answer-set semantics 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 in the rule

$$C(X) \leftarrow subClassOf(D, C), D(X).$$

*This work was partially supported by the Austrian Science Fund (FWF) under grant P17212 and by the European Commission through the IST REWERSE Network of Excellence (IST-506779).

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 predicates extracts RDF statements from the set of URI specified by means of 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-logic 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-Schema) is well-supported. Due to the higher-order features, the representation is succinct. An experimental prototype implementation of the language is available, based on a reduction to ordinary ASP.

Other logic-based formalisms, like TRIPLE [Sintek and Decker, 2002] or F-Logic [Kifer *et al.*, 1995], feature also higher-order predicates for meta-reasoning in Semantic-Web applications. Our formalism is fully declarative and offers the possibility of non-deterministic predicate definition with higher complexity, in a decidable setting. This proved already useful for a range of applications with inherent non-determinism, such as ontology merging or matchmaking, and thus provides a rich basis for integrating these areas with meta-reasoning.

2 HEX Programs

HEX programs are sets of rules of the form

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

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 *negation as failure* (or *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 *input* and *output list*, respectively), and $\&g$ is an *external* predicate name.

A *higher-order atom* (or *atom*) is a tuple $Y_0(Y_1, \ldots, Y_n)$, where $Y_0, \ldots, Y_n$ are terms. It is possible to specify *molecules* of atoms in frame-logic-like syntax. For instance, $gi[father \rightarrow X, Z \rightarrow iu]$ is a shortcut for the conjunction $father(gi, X), Z(gi, iu)$.

The semantics of HEX program is given by generalizing the answer-set semantics [Eiter *et al.*, 2005]. We note that the answer-set semantics may yield no, one, or multiple models (i.e., answer sets) in general. Therefore, for query answering, *brave* and *cautious reasoning* (truth in some resp. all models) are considered in practice, depending on the application.

3 Current Prototype

An experimental working prototype for evaluating HEX programs is available and accessible at `http://www.kr.tuwien.ac.at/staff/roman/dlvhex`. Its further development is work in progress. A wide class of HEX programs is already enabled for evaluation, namely, positive programs (i.e., programs without negation as failure) with both external atoms and higher-order atoms, and disjunctive non-stratified programs without external atoms. The current prototype relies on the systems DLT [Calimeri *et al.*, 2004] and DLV [Leone et al., 2005]. A framework for programming a variety of external atoms is under development. Currently, it features the $\&RDF$ atom, which interfaces the RAPTOR RDF library; the $\&DL$ atom, already available in the companion system NLP-DL [Eiter *et al.*, 2005] will enable DLV-HEX to access the RACER reasoner [Haarslev and Möller, 2001].

4 Applications

HEX programs are well-suited as a convenient tool for a variety of tasks related to ontology languages and for Semantic-Web applications in general, since, in contrast to other approaches, they keep decidability but do not lack the possibility of exploiting non-determinism, performing meta-reasoning, or encoding aggregates and sophisticated constructs through external atoms. An interesting application scenario where several features of HEX programs come into play is *ontology alignment*. Merging knowledge from different sources in the context of the Semantic Web is a very important task [Calvanese *et al.*, 2001].

In order to perform ontology alignment, HEX programs allow to express tasks such as the following ones:

• *Importing* external theories, such as in the following way:

$$\begin{aligned} tripleY(X, Z) &\leftarrow \&RDF[uri](X, Y, Z); \\ tripleY(X, Z) &\leftarrow \&RDF[uri2](X, Y, Z); \\ proposition(P) &\leftarrow triple(P, rdf{:}type, rdf{:}Statement). \end{aligned}$$

• *Searching* in the space of assertions, in order to choose non-deterministically which propositions have to be included in the merged theory and which not, with statements like

$$pick(P) \vee drop(P) \leftarrow proposition(P).$$

• *Translating and manipulating* reified assertions. For instance, it is possible to choose how to put RDF triples (possibly including OWL assertions) in an easier manipulatable and readable format, and to make selected propositions true such as in the following way:

$$\begin{aligned} (X, Y, Z) \leftarrow pick(P), &\; triple(P, rdf{:}subject, X), \\ &\; triple(P, rdf{:}predicate, Y), \\ &\; triple(P, rdf{:}object, Z); \\ C(X) \leftarrow (X, &\; rdf{:}type, C). \end{aligned}$$

• *Defining* ontology semantics. The semantics of the ontology language at hand can be defined in terms of entailment rules and constraints expressed in the language itself or in terms of external knowledge, like in:

$$\begin{aligned} D(X) &\leftarrow subClassof(D, C), C(X), \\ &\leftarrow \&inconsistent[pick], \end{aligned}$$

where the external predicate $\&inconsistent$ takes as input a set of assertions and establishes through an external reasoner whether the underlying theory is inconsistent.

• *Performing the closed-world assumption* (*CWA*) *and default reasoning* in a controlled way. Assuming that a generic external atom $\&DL[C](X)$ is available for querying the concept C in a given description-logic 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.

Inconsistency of the CWA can be checked by pushing back inferred values to the external knowledge base:

$$\begin{aligned} set\_false(C, X) &\leftarrow cwa(C, C'), C'(X), \\ inconsistent &\leftarrow \&DL1[set\_false](b), \end{aligned}$$

where $\&DL1[N](X)$ effects a check whether a knowledge base, augmented with all negated facts $\neg c(a)$ such $N(c, a)$ holds, entails the empty concept $\bot$ (entailment of $\bot(b)$, for any constant b, is tantamount to inconsistency).

References

[Calimeri *et al.*, 2004] G. Ianni, G. Ielpa, A Pietramala, M. C. Santoro and F. Calimeri. Enhancing answer set programming with templates, Proc. of NMR 2004. 233-239.

[Calvanese *et al.*, 2001] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In *Proc. SWWS 2001*, pp. 303–316, 2001.

[Eiter *et al.*, 2005] 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*.

[Eiter *et al.*, 2005] Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Nonmonotonic Description Logic Programs: Implementation and Experiments. In: Proc. LPAR 2004.

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

[Haarslev and Möller, 2001] Haarslev, V., Möller, R.: RACER System Description. In: Proc. IJCAR 2001.

[Leone et al., 2005] Leone, N., *et al*: The DLV System for Knowledge Representation and Reasoning. ACM TOCL, to appear.

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

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

System Demo Description

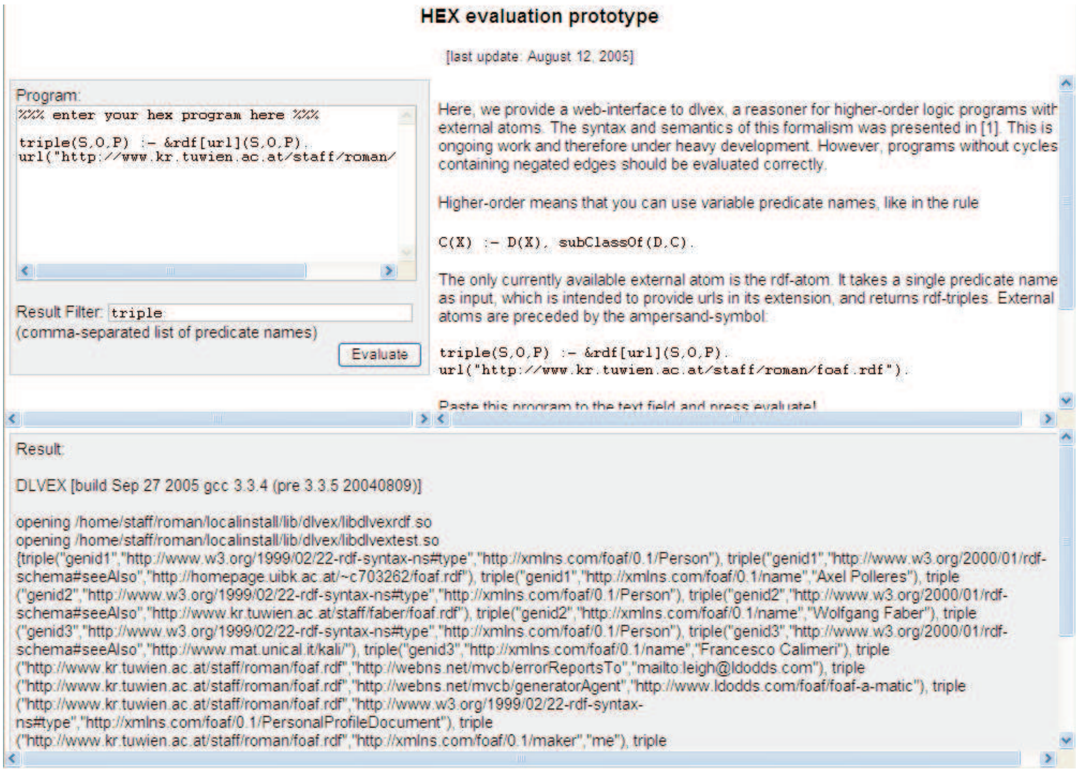

Figure 1: DLV-HEX Web evaluation prototype.

The current prototype is accessible through a publicly accessible Web page, which can compute the models for a given HEX program. Such a program is entered in an apposite text field, using traditional logic programming syntax (extended by the syntax for external atoms and keeping in mind that uppercase letters denote variables for predicate names as well). The "filter" text field can be used for listing the predicate to be shown in the output field. On pushing the "Evaluate"-button, the computation procedure is started. Figure 1 shows the above mentioned fields where a small program is entered. The output, presented in the bottom-most part of the page, is filtered and displays only the extension of the *triple* predicate. This Web prototype is under development and will soon include a variety of external predicate libraries and examples.