A Uniform Integration of Higher-Order Reasoning and External Evaluations in 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 introduce HEX programs, which are nonmonotonic logic programs admitting *higher-order atoms* as well as *external atoms*, and we extend the well-known answer-set semantics to this class of programs. Higher-order features are widely acknowledged as useful for performing meta-reasoning, among other tasks. 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 model some important extensions to ASP, and are a useful KR tool for expressing various applications. Finally, complexity and implementation issues for a preliminary prototype are discussed.

1 Introduction

Answer-Set Programming (ASP) [Gelfond and Lifschitz, 1991] has recently attracted increasing interest as a declarative problem solving paradigm. In this approach, a problem is encoded in terms of a nonmonotonic logic program such that the solutions of the former can be extracted from the *answer sets* of the latter. Due to the availability of efficient answer-set solvers, like Smodels [Simons *et al.*, 2002] or DLV [Leone *et al.*, 2005], and various extensions of the basic language with features such as classical negation, weak constraints, or aggregates, ASP has become an important KR formalism for declaratively solving AI problems in areas including planning, diagnosis, information integration, and reasoning about inheritance. For the challenging area of Semantic Web reasoning, extensions of ASP have been proposed, facilitating interoperability with Description Logic reasoners [Rosati, 1999; Eiter *et al.*, 2004] or aiming at handling infinite, tree-structured models [Heymans and Vermeir, 2003].

However, for important issues such as *meta-reasoning* in the context of the Semantic Web, no adequate support is available in ASP to date. Motivated by this fact and the observation that interoperability with other software is (not only in this context) an important issue, we extend in this paper the answer-set semantics to HEX *programs*, that is, *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

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

An *external atom* facilitates to determine the truth value of an atom through an external source of computation. For instance, the rule

$$reached(X) \leftarrow \#reach[edge, a](X)$$

computes the predicate $reached$ taking values from the predicate $\#reach$, which computes via $\#reach[edge, a]$ all the reachable nodes in the graph $edge$ from node a, delegating this task to an external computational source (e.g., an external deduction system, an execution library, etc.).

Our main contributions are summarized as follows.

(1) We define the syntax and answer-set semantics of HEX programs, extending ASP with higher-order features and powerful interfacing of external computation sources. While answer-set semantics for higher-order logic programs has been proposed earlier by Ross [1994], further extension of that proposal to accommodate external atoms is technically difficult since the approach of Ross is based on the notion of unfounded set, which cannot be easily generalized to this setting. Our approach, instead, is based on a recent notion of program reduct, due to Faber *et al.* [2004], which admits a natural definition of answer-set semantics.

(2) External atoms are a useful abstraction of several extensions to ASP including, among others, aggregates, description logic atoms, or agent programs. External atoms thus facilitate investigating common properties of such extensions, and can serve as a uniform framework for defining semantics of further similar extensions of ASP. Moreover, HEX programs are a basis for the efficient design of generic evaluation algorithms for such extensions in this framework.

*This work was partially supported by the Austrian Science Fund (FWF) under grant P17212-N04, and by the European Commission through the IST Networks of Excellence REWERSE (IST-2003-506779) and CologNeT (IST-2001-33123), and the IST Working Group in Answer Set Programming (IST-2001-37004 WASP).

(3) By means of HEX programs, powerful meta-reasoning becomes available in a decidable context, 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.

(4) A simple prototype implementation of the language is available, based on a reduction to ordinary ASP.

Note that 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. However, TRIPLE is low-level oriented and lack precise semantics, while F-Logic in its implementations (Flora, Florid, Ontoweb) restricts its expressiveness to well-founded semantics for negation, in order to gain efficiency. Our formalism, instead, is fully declarative and offers the possibility of nondeterministic predicate definition with higher complexity. This proved already useful and reasonably efficient for a range of applications with inherent nondeterminism, such as diagnosis, planning, or configuration, and thus provides a rich basis for integrating these areas with meta-reasoning.

2 HEX Programs

2.1 Syntax

Let $\mathcal{C}$, $\mathcal{X}$, and $\mathcal{G}$ be mutually disjoint sets whose elements are called *constant names*, *variable names*, and *external predicate names*, respectively. Unless explicitly specified, elements from $\mathcal{X}$ (resp., $\mathcal{C}$) are denoted with first letter in upper case (resp., lower case), while elements from $\mathcal{G}$ are prefixed with "#". We note that constant names serve both as individual and predicate names.

Elements from $\mathcal{C} \cup \mathcal{X}$ are called *terms*. A *higher-order atom* (or *atom*) is a tuple $(Y_0, Y_1, \ldots, Y_n)$, where $Y_0, \ldots, Y_n$ are terms; $n \geq 0$ is the *arity* of the atom. Intuitively, Y_0 is the predicate name, and we thus also use the more familiar notation $Y_0(Y_1, \ldots, Y_n)$. The atom is *ordinary*, if Y_0 is a constant.

For example, $(x, \mathit{rdf}{:}\mathit{type}, c)$, $\mathit{node}(X)$, and $D(a, b)$, are atoms; the first two are ordinary atoms.

An *external atom* is of the form

$$\#g[Y_1, \ldots, Y_n](X_1, \ldots, X_m), \tag{1}$$

where $Y_1, \ldots, Y_n$ and $X_1, \ldots, X_m$ are two lists of terms (called *input* and *output* lists, respectively), and $\#g \in \mathcal{G}$ is an external predicate name. We assume that $\#g$ has fixed lengths $\mathit{in}(\#g) = n$ and $\mathit{out}(\#g) = m$ for input and output lists, respectively. Intuitively, an external atom provides a way for deciding the truth value of an output tuple depending on the extension of a set of input predicates.

Example 1 The external atom $\#\mathit{reach}[\mathit{edge}, a](X)$ may be devised for computing the nodes which are reachable in the graph edge from the node a. Here, we have that $\mathit{in}(\#\mathit{reach}) = 2$ and $\mathit{out}(\#\mathit{reach}) = 1$. □

A *rule* r is of the form

$$\alpha_1 \vee \cdots \vee \alpha_k \leftarrow \beta_1, \ldots, \beta_n, \mathit{not}\beta_{n+1}, \ldots, \mathit{not}\beta_m, \tag{2}$$

where $m, k \geq 0$, $\alpha_1, \ldots, \alpha_k$ are atoms, and $\beta_1, \ldots, \beta_m$ are either atoms or external atoms. We define $H(r) = \{\alpha_1, \ldots, \alpha_k\}$ and $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{\beta_1, \ldots, \beta_n\}$ and $B^-(r) = \{\beta_{n+1}, \ldots, \beta_m\}$. If $H(r) = \emptyset$ and $B(r) \neq \emptyset$, then r is a *constraint*, and if $B(r) = \emptyset$ and $H(r) \neq \emptyset$, then r is a *fact*; r is *ordinary*, if it contains only ordinary atoms.

A HEX *program* is a finite set P of rules. It is *ordinary*, if all rules are ordinary.

2.2 Semantics

We define the semantics of HEX programs by generalizing the answer-set semantics [Gelfond and Lifschitz, 1991]. To this end, we use the recent notion of a reduct as defined by Faber *et al.* [2004] (referred to as *FLP-reduct* henceforth) instead of to the traditional reduct by Gelfond and Lifschitz [1991]. The FLP-reduct admits an elegant and natural definition of answer sets for programs with aggregate atoms, since it ensures answer-set minimality, while the definition based on the traditional reduct lacks this important feature.

In the sequel, let P be a HEX program. The *Herbrand base* of P, denoted HB_P, is the set of all possible ground versions of atoms and external atoms occurring in P obtained by replacing variables with constants from $\mathcal{C}$. The grounding of a rule r, $\mathit{grnd}(r)$, is defined accordingly, and the grounding of program P is given by $\mathit{grnd}(P) = \bigcup_{r \in P} \mathit{grnd}(r)$. Unless specified otherwise, $\mathcal{C}$, $\mathcal{X}$, and $\mathcal{G}$ are implicitly given by P.

Example 2 Given $\mathcal{C} = \{\mathit{edge}, \mathit{arc}, a, b\}$, ground instances of $E(X, b)$ are $\mathit{edge}(a, b)$, $\mathit{arc}(a, b)$, and $\mathit{arc}(\mathit{arc}, b)$; ground instances of $\#\mathit{reach}[\mathit{edge}, N](X)$ are $\#\mathit{reach}[\mathit{edge}, \mathit{edge}](a)$, $\#\mathit{reach}[\mathit{edge}, \mathit{arc}](b)$, and $\#\mathit{reach}[\mathit{edge}, \mathit{edge}](\mathit{edge})$, etc. □

An *interpretation relative to* P is any subset $I \subseteq HB_P$ containing only atoms. We say that I is a *model* of atom $a \in HB_P$, denoted $I \models a$, if $a \in I$.

With every external predicate name $\#g \in \mathcal{G}$, we associate an $(n{+}m{+}1)$-ary Boolean function $f_{\#g}$ assigning each tuple $(I, y_1 \ldots, y_n, x_1, \ldots, x_m)$ either 0 or 1, where $n = \mathit{in}(\#g)$, $m = \mathit{out}(\#g)$, $I \subseteq HB_P$, and $x_i, y_j \in \mathcal{C}$.

We say that $I \subseteq HB_P$ is a *model* of a ground external atom $a = \#g[y_1, \ldots, y_n](x_1, \ldots, x_m)$, denoted $I \models a$, if and only if $f_{\#g}(I, y_1 \ldots, y_n, x_1, \ldots, x_m) = 1$.

Example 3 Let us associate with $\#\mathit{reach}$ a function $f_{\#\mathit{reach}}$ such that $f_{\#\mathit{reach}}(I, E, A, B) = 1$ iff B is reachable in the graph E from A. Let $I = \{e(b, c), e(c, d)\}$. Then, I is a model of $\#\mathit{reach}[e, b](d)$ since $f_{\#\mathit{reach}}(I, e, b, d) = 1$. □

Let r be a ground rule. We define (i) $I \models H(r)$ iff there is some $a \in H(r)$ such that $I \models a$, (ii) $I \models B(r)$ iff $I \models a$ for all $a \in B^+(r)$ and $I \not\models a$ for all $a \in B^-(r)$, and (iii) $I \models r$ iff $I \models H(r)$ whenever $I \models B(r)$. We say that I is a *model* of a HEX program P, denoted $I \models P$, iff $I \models r$ for all $r \in \mathit{grnd}(P)$. We call P *satisfiable*, if it has some model.

Given a HEX program P, the *FLP-reduct* of P with respect to $I \subseteq HB_P$, denoted fP^I, is the set of all $r \in \mathit{grnd}(P)$ such that $I \models B(r)$. $I \subseteq HB_P$ is an *answer set of* P iff I is a minimal model of fP^I.

We next give an illustrative example.

Example 4 Consider the following HEX program P:

$$subRelation(brotherOf, relativeOf) \leftarrow ;$$
$$brotherOf(john, al) \leftarrow ;$$
$$relativeOf(john, joe) \leftarrow ;$$
$$brotherOf(al, mick) \leftarrow ;$$
$$invites(john, X) \vee skip(X) \leftarrow X <> john, \#reach[relativeOf, john](X);$$
$$R(X, Y) \leftarrow subRelation(P, R), P(X, Y);$$
$$\leftarrow \#degs[invites](Min, Max), Min < 1;$$
$$\leftarrow \#degs[invites](Min, Max), Max > 2.$$

Informally, this program randomly selects a certain number of John's relatives for invitation. The first line states that $brotherOf$ is a subrelation of $relativeOf$, and the next two lines give concrete facts. The disjunctive rule chooses relatives, employing the external predicate $\#reach$ from Example 3. The next rule declares a generic subrelation inclusion exploiting higher-order atoms.

The constraints ensure that the number of invitees is between 1 and 2, using (for illustration) an external predicate $\#degs$ from a graph library, where $f_{\#degs}(I, E, Min, Max)$ is 1 iff Min and Max is the minimum and maximum vertex degree of the graph induced by the edges E, respectively. As John's relatives are determined to be Al, Joe, and Mick, P has six answer sets, each of which contains one or two of the facts $invites(john, al)$, $invites(john, joe)$, and $invites(john, mick)$. $\Box$

We now state some basic properties of the semantics.

Theorem 1 *The answer-set semantics of* HEX *programs extends the answer-set semantics of ordinary programs as defined by Gelfond and Lifschitz [1991], as well as the answer-set semantics of HiLog programs as defined by Ross [1994].*

The next property, which is easily proved, expresses that answer sets adhere to the principle of minimality.

Theorem 2 *Every answer set of a* HEX *program* P *is a minimal model of* P.

A ground external atom a is called *monotonic relative to* P iff $I \subseteq I' \subseteq HB_P$ and $I \models a$ imply $I' \models a$. For instance, the ground versions of $\#reach[edge, a](X)$ are all monotonic.

Theorem 3 *Let* P *be a* HEX *program without "*not*" and constraints. If all external atoms in* $grnd(P)$ *are monotonic relative to* P*, then* P *has some answer set. Moreover, if* P *is disjunction-free, it has a single answer set.*

Notice that this property fails if external atoms can be nonmonotonic. Indeed, we can easily model default negation $not\, p(a)$ by an external atom $\#not[p](a)$; the HEX program $p(a) \leftarrow \#not[p](a)$ amounts then to the ordinary program $p(a) \leftarrow not\, p(a)$, which has no answer set.

3 Modeling ASP Extensions by External Atoms

By means of external atoms, different important extensions of ASP can be expressed in terms of HEX programs.

3.1 Programs with aggregates

Extending ASP with special *aggregate atoms*, through which the sum, maximum, etc. of a set of numbers can be referenced, is an important issue which has been considered in several recent works (cf., e.g., [Faber *et al.*, 2004]). A nontrivial and challenging problem in this context is giving a natural semantics for aggregates involving recursion. The recent proposal of a semantics by Faber *et al.* [2004] is an elegant solution of this problem. We show here how it can be easily captured by HEX programs.

An aggregate atom $a(Y, T)$ has the form $f\{S\} \prec T$, where f is an aggregate function (sum, $count$, max, etc.), $\prec \in \{=, <, \leq, >, \geq\}$, T is a term, and S is an expression $X{:}\vec{E}(\vec{X}, \vec{Y}, \vec{Z})$, where $\vec{X}$ and $\vec{Y}$ are lists of *local variables*, $\vec{Z}$ is a list of *global variables*, and $\vec{E}$ is a list of atoms whose variables are among $\vec{X}, \vec{Y}, \vec{Z}$.

For example, $\#count\{X : r(X, Z), s(Z, Y)\} \geq T$ is an aggregate atom which is intuitively true if, for given Y and T, at least T different values for X are such that the conjunction $r(X, Z), s(Z, Y)$ holds.

Given $a(Y, T) = f\{S\} \prec T$ as above, an interpretation I, and values y for Y and t for T, f is applied to the set $S(I, y)$ of all values x for X such that $I \models E(x, y, z)$ for some value z for Z. We then have $I \models a(y, t)$ (i.e., $I \models f\{X{:}E(X, y, Z)\} \prec t$) iff $f(S(I, y)) \prec t$.

Using the above notion of truthhood for $a(y, t)$, Faber *et al.* [2004] define answer sets of an ordinary program plus aggregates using the reduct fP^I.

We can model an aggregate atom $a(Y, T)$ by an external atom $\#a[Y](T)$ such that for any interpretation I and ground version $\#a[y](t)$ of it, $f_{\#a}(I, y, t) = 1$ iff $I \models a(y, t)$. Note that writing code for evaluating $f_{\#a}(I, y, t)$ is easy.

For any ordinary program P with aggregates, let $\#agg(P)$ be the HEX program which results from P by replacing each aggregate atom $a(Y, T)$ with the respective external atom $\#a[Y](T)$. The following result can then be shown:

Theorem 4 *For any ordinary program* P *with aggregates, the answer sets of* P *and* $\#agg(P)$ *coincide.*

3.2 Description logic programs

The aim of *description logic programs* (or *dl-programs*), due to Eiter *et al.* [2004], is to combine a rule language under the answer-set semantics with description logics. Informally, a dl-program consists of a description logic (DL) knowledge base L and a generalized normal program P which may contain queries to L, realized by means of special atoms, called *dl-atoms*, appearing in the body of rules. A dl-atom allows for specifying an input from P to L, and thus for a bidirectional flow of information between P to L, and for querying whether a certain DL axiom or its negation logically follows from L. The DL knowledge bases in dl-programs are theories in the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which represent the logical underpinnings of the Web ontology languages OWL Lite and OWL DL, respectively [Bechhofer *et al.*, 2004].

Formally, a *dl-atom* is an expression $dl(X)$ of form

$$DL[S_1\ op_1\ p_1, \ldots, S_m\ op_m\ p_m; Q](X)\,, \qquad m \geq 0,$$

where each S_i is a DL concept or role name, op_i a change operator, p_i a unary resp. binary predicate symbol, Q a unary resp. binary predicate, and X a list of terms matching the arity of Q. For space reasons, we confine here to $op_i = \uplus$ and Q being a possibly negated unary predicate name, for which X is a single term. Intuitively, $S_i \uplus p_i$ increases S_i in L by the extension of p_i. For example, the dl-atom

$$DL[\mathit{hasColor} \uplus \mathit{color}; \mathit{whiteWine}](W)$$

queries a wine ontology if W is known to be a white wine, after augmenting the ontology about wine color ($\mathit{hasColor}$) with facts about color from a program P.

An interpretation I of P is a *model* of a ground instance $dl(c)$ of dl-atom $dl(X)$ with respect to DL knowledge base L, denoted $I \models_L dl(c)$, if $L \cup \bigcup_{i=1}^{m} \{S_i(b) \mid p_i(b) \in I\} \models Q(c)$, where $\models$ is the entailment operator of the given description logic. That is, $I \models_L dl(c)$ iff c belongs to concept Q after augmenting L.

Eiter *et al.* [2004] define answer sets of an ordinary non-disjunctive program P relative to a DL knowledge base L through a reduct sP^I_L, which extends the traditional reduct of Gelfond and Lifschitz [1991]. Assuming that each ground dl-atom $dl(c)$ is monotonic (i.e., $I \models dl(c)$ implies $I' \models dl(c)$, for $I \subseteq I'$; this is the predominant setting), sP^I_L treats negated dl-atoms like negated ordinary atoms. The resulting ground program sP^I_L has a least model, $LM(sP^I_L)$. Then, I is a *strong answer set* of (L, P) iff $I = LM(sP^I_L)$ holds.

We can simulate dl-atoms by external atoms in several ways. A simple one is to use external atoms $\#dl[\,](X)$ where $f_{\#dl}(I, c) = 1$ iff $I \models_L dl(c)$. Let $\#dl_L(P)$ be the HEX program obtained from a dl-program (L, P) by replacing each dl-atom $dl(X)$ with $\#dl[\,](X)$. We can then show:

Theorem 5 *Let (L, P) be any dl-program for which all ground dl-atoms are monotonic. Then, the strong answer sets of (L, P) and $\#dl_L(P)$ coincide.*

Note that we can extend the strong answer-set semantics to disjunctive dl-programs by simply extending the embedding $\#dl_L(P)$ to disjunctive programs. This illustrates the use of HEX programs as a framework for defining semantics.

3.3 Programs with monotone cardinality atoms

Marek *et al.* [2004] present an extension of ASP by *monotone cardinality atoms* (*mc-atoms*) $k\,X$, where X is a finite set of ground atoms and $k \geq 0$. Such an atom is true in an interpretation I, if $k \geq |X \cap I|$ holds. Note that an ordinary atom A amounts to $1\{A\}$. An *mca-program* is a set of rules

$$H \leftarrow B_1, \ldots, B_m, \mathit{not}\, B_{m+1}, \ldots, \mathit{not}\, B_n \qquad (3)$$

where H and the B_i's are mc-atoms. Answer sets (stable models) for an mca-program P are interpretations I which are *derivable* models of an extended reduct P^I (in the sense of Gelfond and Lifschitz [1991]), which treats negated mc-atoms like negated ordinary atoms. Informally, a model of P^I is derivable, if it can be created from the empty set by iterative rule applications in which the heads of firing rules are nondeterministically satisfied.

We can embed any mca-program P into a HEX program $\#mc(P)$ as follows. Each mc-atom $k\,X$ is modeled by an external atom $e(k\,X) = \#k\_X[\,]()$, where $f_{\#k\_X}(I) = 1$ iff $k \geq |X \cap I|$. In each rule of form (3), we replace H with a new atom t_H and all B_i with $e(B_i)$, and add the following rules (for $H = k\,\{A_1, \ldots, A_m\}$):

$$\begin{aligned} A_i \vee n\_A_i &\leftarrow t_H, \qquad\qquad 1 \leq i \leq m, \\ &\leftarrow \mathit{not}\, e(H), t_H, \end{aligned}$$

where, globally, n_A is a new atom for each atom A. Informally, these rules simulate the occurrence of the mc-atom in the head. Then, the following correspondence holds.

Theorem 6 *For any finite mca-program P over atoms At, the answer sets of P and $\#mc(P)$ projected to At coincide.*

As shown by Marek *et al.* [2004], ASP extensions similar to mca-programs can be modeled as mca-programs. Hence, these extensions can be similarly embedded into HEX programs.

3.4 Agent programs

Eiter *et al.* [1999] describe logic-based *agent programs*, consisting of rules of the form

$$Op_0\alpha_0 \leftarrow \chi, [\neg]\, Op_1\alpha_1, \ldots, [\neg]\, Op_m\alpha_m,$$

governing an agent's behavior. The Op_i are *deontic modalities*, the α_i are *action atoms*, and χ is a *code-call condition*. The latter is a conjunction of (i) *code-call atoms* of the form $in(X, f(Y))$ resp. $notin(X, f(Y))$, which access the data structures of the internal agent state through API functions $f(Y)$ and test whether X is in the result, and (ii) *constraint atoms*. For example, the rule

$$Do\ dial(N) \leftarrow in(N, phone(P)), O\ call(P)$$

intuitively says that the agent should dial phone number N if she is obliged to call P.

A semantics of agent programs in terms of "reasonable status sets", which are certain sets of ground formulas $Op\,\alpha$, is defined by Eiter *et al.* [1999]. They show that the answer sets of a disjunction-free logic program P correspond naturally to the reasonable status sets of a straightforward agent program $AG(P)$. Conversely, code-call atoms as above can be modeled by external atoms $\#in_f[Y](X)$ resp. $\#notin_f[Y](X)$, and deontic modalities by different propositions and suitable rules. In this way, a class of agent programs can be embedded into HEX programs as a host for evaluation.

4 Applications

In this section, we show the usage of HEX programs for different purposes, in which the joint availability of higher-order and external atoms is beneficial. For space reasons, the exposition is necessarily superficial and details will be omitted.

4.1 Semantic Web 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 nondeterminism, 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]. To avoid inconsistencies which arise in merging, it is important to diagnose the source of such inconsistencies and to propose a "repaired" version of the merged ontology. In general, given an entailment operator $\models$ and two theories T_1 and T_2, we want to find some theory $rep(T_1 \cup T_2)$ which, if possible, is consistent (with respect to $\models$). Usually, rep is defined according to some customized criterion, so that to save as much knowledge as possible from T_1 and T_2. Also, rep can be nondeterministic and admit more than one possible solution.

HEX programs allow to define $\models$ according to a range of possibilities; in the same way, HEX programs are a useful tool for modeling and customizing the rep operator. In order to perform ontology alignment, HEX programs must be able to express tasks such as the following ones:

Importing external theories. This can be achieved, e.g., in the following way:

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

We assume here to deal with RDF theories.[1] We take advantage of an external predicate $\#RDF$ intended to extract knowledge from a given URI (Uniform Resource Identifier), in form of a set of "reified" ternary assertions.

Searching in the space of assertions. This task is required in order to choose nondeterministically 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. E.g., for choosing how to put RDF triples (possibly including OWL assertions) in an easier manipulatable and readable format, and for making selected propositions true, the following rules can be employed:

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

Filtering propositions. This way, it is possible to customize criteria for selecting which propositions can be dropped and which cannot. For instance, a proposition cannot be dropped if it is an RDFS axiomatic triple:[2]

$$pick(P) \leftarrow axiomatic(P).$$

Defining ontology semantics. The operator $\models$ can be defined in terms of entailment rules and constraints expressed in the language itself, like in:

$$\begin{aligned} D(X) &\leftarrow (C, rdf{:}subClassOf, D), C(X); \\ &\leftarrow owl{:}maxCardinality(C,R,N), C(X), \\ &\qquad \#countr[R,X](M), M > N, \end{aligned}$$

where the external atom $\#countr[R,X](M)$ expresses the aggregate atom $\#count\{Y : R(X,Y)\} = M$. Also, semantics can be defined by means of external reasoners, using constraints like

$$\leftarrow \#inconsistent[pick],$$

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

4.2 Closed world and default reasoning

Reiter's well-known closed-world assumption (CWA)[3] is acknowledged as an important reasoning principle for inferring negative information from a logical knowledge base KB: For a ground atom $p(c)$, conclude $\neg p(c)$ if $KB \not\models p(c)$. Description logic knowledge bases lack this possibility.

Using HEX programs, the CWA may be easily expressed on top of an external KB which can be queried through suitable external atoms. We show this here for a description logic knowledge base L. Assuming that a generic external atom $\#dl_0[C](X)$ for modeling a dl-atom $DL[C](X)$ is available, the CWA principle can be stated as follows:

$$\begin{aligned} C'(X) \leftarrow not\, \#dl_0[C](X), &\, concept(C), \\ &\, cwa(C,C'), o(X), \end{aligned}$$

where $concept(C)$ is a predicate which holds for all concepts, $cwa(C,C')$ states that C' is the complement of C under the CWA, and $o(X)$ is a predicate that holds for all individuals occurring in L. For example, given that

$$L = \{\, man \sqsubseteq person,\ person(lee)\,\}$$

for concepts man and $person$, the CWA infers $\neg man(lee)$.

As well known, the CWA can become inconsistent. If in the above example, L contains a further axiom

$$person = man \sqcup woman,$$

with the concept $woman$, then the CWA infers $\neg man(lee)$ and $\neg woman(lee)$; this is inconsistent with L.

[1] See http://www.w3.org/tr/rdf-mt/ for information about RDF.

[2] In a language enriched with *weak constraints*, we could maximize the set of selected propositions using a constraint of form $:\sim drop(P)$.

[3] Throughout this section, we refer to Łukaszewicz [1990] for references to closed-world reasoning and circumscription.

We can check inconsistency of the CWA with further rules, though:

$$set\_false(C, X) \leftarrow cwa(C, C'), C'(X),$$
$$inconsistent \leftarrow \#dl_1[set\_false, \bot](b),$$

where $\#dl_1[N, C](X)$ effects a check whether L, augmented with all negated facts $\neg c(a)$ such that $N(c, a)$ holds, entails $C(X)$, and $\bot$ is the empty concept (entailment of $\bot(b)$, for any constant b, is tantamount to inconsistency).

Minimal-model reasoning, as under circumscription and the extended closed-world assumption (ECWA), for instance, avoids the problem of CWA inconsistency. We can foster the minimal Herbrand models of L with respect to all concepts and individuals in L elegantly with the following HEX rules:

$$set\_false(C, X) \leftarrow concept(C), o(X), not\, C(X);$$
$$C(X) \leftarrow \#dl_1[set\_false, C](X).$$

Here, the first rule intuitively expresses that if $C(X)$ is not included in an answer set M of P, then it should be set to false. The second rule states that $C(X)$ is in M, if $C(X)$ can be proved in L after setting all atoms in L to false according to M. By the minimality of answer sets, $C(X)$ can only then be in M. Thus, in L no $C(X)$ can be switched to $\neg C(X)$ without raising inconsistency. Hence, M corresponds to a minimal model of L. Applied to our example, we obtain two answer sets (showing here only the interesting atoms):

$$M_1 = \{person(lee), woman(lee), set\_false(man, lee), \ldots\},$$
$$M_2 = \{person(lee), man(lee), set\_false(woman, lee), \ldots\},$$

corresponding to the minimal models of L.

Roles in L may be handled similarly. Furthermore, one can easily restrict minimization to a subset of concepts and roles, and accommodate the general setting of ECWA and circumscription, dividing the predicates into minimized, fixed, and varying predicates P, Q, and Z, respectively. On top of minimal models, e.g., reasoning tasks may then be performed.

By *maximizing* rather than minimizing extensions, default reasoning, as in the approach by Poole [1988], on top of a DL knowledge base L may be supported. For example, the rules

$$white(W) \leftarrow \#dl_1[null, sparklingWine](W), not\, n\_white(W),$$
$$n\_white(W) \leftarrow \#dl_2[sparklingWine, white, whiteWine](W)$$

on top of a wine ontology L, express that sparkling wines are white by default, where $\#dl_2[C, U, Q](X)$ checks whether L, together with all facts $C(a)$ such that $a \in U$, entails $\neg Q(X)$. Given

$$L = \{sparklingWine(veuveCliquot), lambrusco \sqsubseteq (sparklingWine \sqcap \neg whiteWine)\},$$

we then can conclude $white(veuveCliquot)$.

5 Computational Aspects

5.1 Complexity

It appears that higher-order atoms do not add complexity compared to ordinary atoms. Indeed, for finite $\mathcal{C}$, the grounding of an arbitrary HEX program P is, like for an ordinary program, at most exponential in the size of P and $\mathcal{C}$. Since HEX programs with higher-order atoms subsume ordinary programs, we obtain by well-known complexity results for ordinary programs [Dantsin *et al.*, 2001] the following result. Recall that *NEXP* denotes nondeterministic exponential time, and that for complexity classes C and D, C^D denotes complexity in C with an oracle for a problem in D.

Theorem 7 *Deciding whether a given* HEX *program P without external atoms has some answer set is NEXP^{NP}-complete in general, and NEXP-complete if P is disjunction-free.*

Classes of programs with lower complexity can be identified under syntactic restrictions, e.g., on predicate arities. Furthermore, if from the customary ASP perspective, P is fixed except for ground facts representing ad-hoc input, the complexity exponentially drops to NP^{NP} resp. NP.

On the other hand, external atoms clearly may be a source of complexity, and without further assumptions even incur undecidability. Viewing the function $f_{\#g}$ associated with an external predicate $\#g \in \mathcal{G}$ as an oracle with complexity in C, we have the following result:

Theorem 8 *Let P be a* HEX *program, and suppose that for every $\#g \in \mathcal{G}$ the function $f_{\#g}$ has complexity in C. Then, deciding whether P has some answer set is in NEXP$^{NP^C}$, and is in NEXP^C if P is disjunction-free.*

However, there is no complexity increase by external atoms under the following condition on the cardinality of $\mathcal{C}$:

Theorem 9 *Let P be a* HEX *program. Suppose that for every $\#g \in \mathcal{G}$, the function $f_{\#g}$ is decidable in exponential time in $|\mathcal{C}|$. Then, deciding whether P has some answer set is $NEXP^{NP}$-complete, and NEXP-complete if P is disjunction-free.*

Informally, the reason is that a possibly exponential-size grounding compensates the exponentiality of external atoms, whose evaluation then becomes polynomial in the size of $grnd(P)$. The hypothesis of Theorem 9 applies to external atoms modeling aggregate atoms and, under small adjustments, to dl-atoms, if $\models$ is decidable in exponential time. Some complexity results by Faber *et al.* [2004] on ASP with aggregates and by Eiter *et al.* [2004] on interfacing logic programs with the description logic $\mathcal{SHIF}(\mathbf{D})$ therefore follow easily from Theorems 4, 5, and 9.

5.2 Implementation

An experimental working prototype for evaluating HEX programs is available. Several technical issues in an implementation arise, and we can only briefly address them here. In particular, higher-order and external atoms must be handled.

As for higher-order atoms, a polynomial reduction Λ from HEX programs P to ordinary programs $\Lambda(P)$ is possible if P has no external atoms. Indeed, each higher-order atom $Y_0(Y_1, \ldots, Y_n)$ in P can be substituted with an ordinary atom $a_n(Y_0, Y_1, \ldots, Y_n)$. Since HEX programs conservatively extend ordinary programs (cf. Theorem 1), the answer sets of any HEX program P without external atoms then correspond one-to-one with the answer sets of $\Lambda(P)$. Thus, HEX programs without external atoms can be efficiently evaluated by using an existing ASP solver.

The presence of external atoms makes matters more complex. Λ can still be applied to eliminate higher-order atoms from a HEX program P, and a similar correspondence holds. We may further replace external atoms $\#g[\vec{X}](\vec{Y})$ in $\Lambda(P)$ by ordinary atoms $p_{\#g}(\vec{X}, \vec{Y})$. In the absence of negation as failure and for monotone external atoms, the answer sets of $\Lambda(P)$ can be computed by a bottom-up fixpoint computation (which in case of disjunction is nondeterministic), in which ground atoms $p_{\#g}(\vec{a}, \vec{b})$ are evaluated with the external function $f_{\#g}$.

In the presence of negation as failure, a notion of *e-stratification*, which generalizes the usual notion of stratification and exploits further dependency information supplied for external atoms, can be used to identify a substantial fragment of HEX programs evaluable on the basis of a suitable operational semantics. In the unstratified case, guessing clauses

$$p_{\#g}(\vec{X}, \vec{Y}) \vee not\_p_{\#g}(\vec{X}, \vec{Y}) \leftarrow$$

may be added for generating candidate answer sets of P. For monotone external atoms, the candidates can be verified by a fixpoint computation. For the general case, however, efficient checking methods are needed.

6 Conclusion and Further Work

HEX programs are a natural and powerful evolution of Answer-Set Programming (ASP), which fulfills interoperability needs with other software and supports at the same time abstract problem modeling by higher-order features. These features are needed for a wide range of applications but missing in ASP systems today. In particular, user-defined libraries can be integrated, and thus customization to specific applications is enabled. Our further and ongoing work includes implementation beyond the working prototype, for which suitable algorithms and techniques are currently under development. This and the prototype will be discussed in detail elsewhere. Furthermore, an application in the context of an ongoing project for a personalized Web information system is targeted.

References

[Bechhofer *et al.*, 2004] S. Bechhofer, F. van Harmelen, J. Hendler, et al. OWL web ontology language reference. `http://www.w3.org/tr/owl-ref/`.

[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.

[Dantsin *et al.*, 2001] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Comp. Surveys*, 33:374-425, 2001.

[Eiter *et al.*, 1999] T. Eiter, V.S. Subrahmanian, and G. Pick. Heterogeneous active agents, I: Semantics. *Artificial Intelligence*, 108(1-2):179–255, 1999.

[Eiter *et al.*, 2004] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with description logics for the Semantic Web. In *Proc. KR-2004*, pp. 141–151, 2004.

[Faber *et al.*, 2004] W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proc. JELIA-2004*, pp. 200–212, 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.

[Heymans and Vermeir, 2003] S. Heymans and D. Vermeir. Integrating Semantic Web reasoning and Answer Set Programming. In *Proc. ASP-2003*, pp. 194–208, 2003.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.

[Leone *et al.*, 2005] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 2005. To appear.

[Łukaszewicz, 1990] W. Łukaszewicz. *Non-monotonic Reasoning: Formalizations of Commonsense Reasoning*. Ellis Horwood, 1990.

[Marek *et al.*, 2004] V. Marek, I. Niemelä, and M. Truszczyński. Logic programs with monotone cardinality atoms. In *Proc. LPNMR-2004*, pp. 154–166, 2004.

[Poole, 1988] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[Rosati, 1999] R. Rosati. Towards expressive KR systems integrating datalog and description logics: Preliminary report. In *Proc. DL-1999*, pp. 160–164, 1999.

[Ross, 1994] K. A. Ross. On negation in HiLog. *Journal of Logic Programming*, 18(1):27–53, 1994.

[Simons *et al.*, 2002] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, 2002.

[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, 2002.