TLP **10** (4–6): 531–545, 2010. © Cambridge University Press 2010
doi:10.1017/S1471068410000268

Loop formulas for description logic programs

YISONG WANG

Department of Computer Science, Guizhou University, Guiyang, China
Department of Computing Science, University of Alberta, Canada

JIA-HUAI YOU, LI YAN YUAN

Department of Computing Science, University of Alberta, Canada

and YI-DONG SHEN*

State Key Laboratory of Computer Science Institute of Software, Chinese Academy of Sciences, China

submitted 6 February 2010; revised 7 April 2010; accepted 14 May 2010

Abstract

Description Logic Programs (dl-programs) proposed by Eiter *et al.* constitute an elegant yet powerful formalism for the integration of answer set programming with description logics, for the Semantic Web. In this paper, we generalize the notions of completion and loop formulas of logic programs to description logic programs and show that the answer sets of a dl-program can be precisely captured by the models of its completion and loop formulas. Furthermore, we propose a new, alternative semantics for dl-programs, called the *canonical answer set semantics*, which is defined by the models of completion that satisfy what are called canonical loop formulas. A desirable property of canonical answer sets is that they are free of circular justifications. Some properties of canonical answer sets are also explored.

KEYWORDS: semantic web, description logic programs, answer sets, loop formulas

1 Introduction

Logic programming under the answer set semantics (ASP) is a nonmonotonic reasoning paradigm for declarative problem solving (Marek and Truszczynski 1999; Niemelä 1999). Recently, there have been extensive interests in combining ASP with other computational and reasoning paradigms. One of the main interests in this direction is the integration of ASP with ontology reasoning, for the Semantic Web.

The Semantic Web is an evolving development of the World Wide Web in which the meaning of information and services on the web are defined, so that the web content can be precisely understood and used by agents (Berners-Lee *et al.* 2001). For this purpose, a layered structure including the Rules Layer built on top of the Ontology Layer has been recognized as a fundamental framework. Description Logics (DLs) (Baader *et al.* 2007) provide a formal basis for the Web Ontology Language which is the standard of the Ontology Layer (W3C OWL Working Group 2009).

* Yisong Wang was supported in part by NSFC grants 90718009 and 60703095, the Fund of Guizhou Science and Technology 2008[2119], the Fund of Education Department of Guizhou Province 2008[011], Scientific Research Fund for talents recruiting of Guizhou University 2007[042]. Yi-Dong Shen is supported in part by NSFC grants 60970045 and 60721061.

Adding nonmonotonic rules to the Rules Layer would allow default reasoning with ontologies. For example, we know that most *natural kinds* do not have a clear cut definition. For instance, a precise definition of *scientist* seems to be difficult by enumerating what a scientist is, and does. Though we can say that a scientist possesses expert knowledge on the subject of his or her investigation, we still need a definition of expert knowledge, which cannot be defined quantitatively. Using nonmonotonic rules, we can perform default, typicality reasoning over categories, concepts, and roles. The integration of DLs and (nonmonotonic) rules has been extensively investigated as a crucial problem in the study of the Semantic Web, such as *Semantic Web Rule Language* (SWRL) (Horrocks and Patel-Schneider 2004), *MKNF knowledge base* (Motik and Rosati 2010), and *Description Logic Programs* (*dl-programs*) (Eiter *et al.* 2008).

There are different approaches to the integration of ASP with description logics. The focus of this paper is on the approach based on dl-programs. Informally, a dl-program is a pair (O,P), where O is a DL knowledge base and P is a logic program whose rule bodies may contain queries, embedded in *dl-atoms*, to the knowledge base O. The answer to such a query depends on inferences by rules over the DL knowledge base O. In this way, rules are built on top of ontologies. On the other hand, ontology reasoning is also enhanced, since it depends not only on O but also on inferences using (nonmonotonic) rules. Two semantics for dl-programs have been proposed, one of which is based on *strong answer sets* and the other based on *weak answer sets*.

In this paper, we generalize the notions of completion and loop formulas of logic programs (Lin and Zhao 2004) to dl-programs and show that weak and strong answer sets of a dl-program can be captured precisely by the models of its completion and the corresponding loop formulas. This provides not only a semantic characterization of answer sets for dl-programs but also an alternative mechanism for answer set computation, using a dl-reasoner and a SAT solver.

As commented by (Eiter *et al.* 2008), the reason to introduce strong answer sets is because some weak answer sets seem counterintuitive due to “self-supporting” loops. Recently however, one of the co-authors of this paper, Yi-Dong Shen, discovered that strong answer sets may also possess self-supporting loops, and a detailed analysis leads to the conclusion that the problem cannot be easily fixed by an alternative definition of *reduct*, since the reduct of dl-atoms may not be able to capture dynamically generated self-supports arising from the integrated context.

The solution proposed in this paper is to use loop formulas as a way to define answer sets for dl-programs that are free of self-supports. Thus, we define what are called *canonical loops* and *canonical loop formulas*. Given a dl-program, the models of its completion satisfying the canonical loop formulas constitute a new class of answer sets, called *canonical answer sets*, that are minimal and noncircular.

The paper is organized as follows. In the next section, we recall the basic definitions of description logics and dl-programs. In Section 3, we define completion, weak and strong loop formulas for dl-programs. The new semantics of dl-programs based on canonical loop formulas is given in Section 4. Section 5 discusses related work, and finally Section 6 gives concluding remarks. The proofs of the main theorems can be found at `http://webdocs.cs.ualberta.ca/~you/papers/iclp2010-full-paper.pdf`.

2 Preliminaries

In this section, we briefly review the basic notations for description logics and description logic programs (Eiter *et al.* 2008).

2.1 Description logics

In principle, the description logics employed in description logic programs can be arbitrary, with the restriction that the underlying entailment relation is decidable. Due to space limitation, we introduce the basic description logic $\mathscr{ALC}$ (Baader *et al.* 2007), instead of the description logics $\mathscr{SHIF}$ and $\mathscr{SHOIN}$ described in (Eiter *et al.* 2008). The notations introduced here will be used throughout the paper, particularly the entailment relation $O \models F$, given at the end of this subsection.

For the language $\mathscr{ALC}$, we assume a vocabulary $\Psi = (\mathbf{A} \cup \mathbf{R}, \mathbf{I})$, where $\mathbf{A}, \mathbf{R}$ and $\mathbf{I}$ are pairwise disjoint (denumerable) sets of *atomic concepts*, *roles* (including equality $\approx$ and inequality $\not\approx$), and *individuals* respectively. The *concepts* of $\mathscr{ALC}$ are defined as follows:

$$C, D \longrightarrow A|\top|\bot|\neg C|C \sqcap D|C \sqcup D|\forall R \cdot C|\exists R \cdot C$$

where A is an atomic concept and R is a role. The *assertions* of $\mathscr{ALC}$ are of the forms $C(a)$ or $R(b,c)$, where C is a concept, R is a role, and a, b, c are individuals. An *inclusion axiom* of $\mathscr{ALC}$ has the form $C \sqsubseteq D$ where C and D are concepts. A *description knowledge base* (or *ontology*) of $\mathscr{ALC}$ is a set of inclusion axioms and assertions of $\mathscr{ALC}$.

The semantics of $\mathscr{ALC}$ is defined by translating to first-order logic and then using classical first-order interpretations as its semantics. Informally, let the transformation be τ: (1) $\tau(A) = A(x)$, $\tau(R) = R(x,y)$ where A is an atomic concept and R a role; (2) $\tau(\forall R \cdot C) = \forall x \cdot R(y,x) \supset \tau(C)(x)$, and $\tau(\exists R \cdot C) = \exists x \cdot R(y,x) \wedge \tau(C)(x)$; (3) $\tau(\neg C) = \neg\tau(C)(x)$, $\tau(C \sqcap D) = \tau(C)(x) \wedge \tau(D)(x)$, and $\tau(C \sqcup D) = \tau(C)(x) \vee \tau(D)(x)$; (4) $\tau(A(a)) = A(a)$, $\tau(R(b,c)) = R(b,c)$; (5) $\tau(C \sqsubseteq D) = \forall x \cdot \tau(C)(x) \supset \tau(D)(x)$. Then, the semantics of $\mathscr{ALC}$ follows from that of first-order logic, so is the entailment relation $O \models F$, for a description knowledge base O and an assertion or inclusive axiom F.

2.2 Description logic programs

Let $\Phi = (\mathscr{P}, \mathscr{C})$ be a first-order vocabulary with nonempty finite sets $\mathscr{C}$ and $\mathscr{P}$ of constant symbols and predicate symbols respectively such that $\mathscr{P}$ is disjoint from $\mathbf{A} \cup \mathbf{R}$ and $\mathscr{C} \subseteq \mathbf{I}$. *Atoms* are formed from the symbols in $\mathscr{P}$ and $\mathscr{C}$ as usual.

A *dl-atom* is an expression of the form

$$DL[S_1\ op_1\ p_1, \ldots, S_m\ op_m\ p_m; Q](\vec{t}), \quad (m \geqslant 0) \tag{1}$$

where

- each S_i is either a concept, a role or a special symbol in $\{\approx, \not\approx\}$;
- $op_i \in \{\oplus, \odot, \ominus\}$;
- p_i is a unary predicate symbol in $\mathscr{P}$ if S_i is a concept, and a binary predicate symbol in $\mathscr{P}$ otherwise. The p_is are called *input predicate symbols*;

- $Q(\vec{t})$ is a *dl-query*, i.e., either (1) $C(t)$ where $\vec{t} = t$; (2) $C \sqsubseteq D$ where $\vec{t}$ is an empty argument list; (3) $R(t_1, t_2)$ where $\vec{t} = (t_1, t_2)$; (4) $t_1 \approx t_2$ where $\vec{t} = (t_1, t_2)$; or their negations, where C and D are concepts, R is a role, and $\vec{t}$ is a tuple of constants.

The precise meanings of $\{\oplus, \odot, \ominus\}$ will be defined shortly. Intuitively, $S \oplus p$ (resp. $S \odot p$) extends S (resp. $\neg S$) by the extension of p, and $S \ominus p$ constrains S to p.

For example, suppose the interface is such that if any individual x is registered for a course (the information from outside an ontology) then x is a student (x may not be a student by the ontology before this communication), and we query if a is a student. We can then write the dl-atom $DL[Student \oplus registered; Student](a)$. Similarly, $DL[Student \ominus registered; \neg Student \sqcap \neg Employed](a)$ queries if a is not a student nor employed, with the ontology enhancement that if we cannot show x is registered, then x is not a student.

A *dl-rule* (or simply a *rule*) is an expression of the form

$$A \leftarrow B_1, \ldots, B_m, not\ B_{m+1}, \ldots, not\ B_n, (n \geqslant m \geqslant 0) \tag{2}$$

where A is an atom, each $B_i (1 \leqslant i \leqslant n)$ is an atom[1] or a dl-atom. We refer to A as its *head*, while the conjunction of $B_i (1 \leqslant i \leqslant m)$ and *not* $B_j (m+1 \leqslant j \leqslant n)$ is its *body*. For convenience, we may abbreviate a rule in the form (2) as

$$A \leftarrow Pos, not\ Neg \tag{3}$$

where $Pos = \{B_1, \ldots, B_m\}$ and $Neg = \{B_{m+1}, \ldots, B_n\}$. Let r be a rule of the form (3). If $Neg = \emptyset$ and $Pos = \emptyset$, r is a fact and we may write it as "A" instead of "$A \leftarrow$". A *description logic program* (*dl-program*) $\mathscr{K} = (O, P)$ consists of a DL knowledge base O and a finite set P of dl-rules. In what follows we assume the vocabulary of P is implicitly given by the constant symbols and predicates symbols occurring in P, unless stated otherwise.

Given a dl-program $\mathscr{K} = (O, P)$, the *Herbrand base* of P, denoted by HB_P, is the set of atoms formed from the predicate symbols in $\mathscr{P}$ occurring in P and the constant symbols in $\mathscr{C}$ occurring in P. An *interpretation* I (relative to P) is a subset of HB_P. Such an I is a *model* of an atom or dl-atom A under O, written $I \models_O A$, if the following holds:

- if $A \in HB_P$, then $I \models_O A$ iff $A \in I$;
- if A is a dl-atom $DL(\lambda; Q)(\vec{t})$ of the form (1), then $I \models_O A$ iff $O(I;\lambda) \models Q(\vec{t})$ where $O(I;\lambda) = O \cup \bigcup_{i=1}^{m} A_i(I)$ and, for $1 \leqslant i \leqslant m$,

$$A_i(I) = \begin{cases} \{S_i(\vec{e}) | p_i(\vec{e}) \in I\}, & \text{if } op_i = \oplus; \\ \{\neg S_i(\vec{e}) | p_i(\vec{e}) \in I\}, & \text{if } op_i = \odot; \\ \{\neg S_i(\vec{e}) | p_i(\vec{e}) \notin I\}, & \text{if } op_i = \ominus; \end{cases}$$

where $\vec{e}$ is a tuple of constants over $\mathscr{C}$. The interpretation I is a *model* of a dl-rule of the form (3) iff $I \models_O B$ for any $B \in Pos$ and $I \not\models_O B'$ for any $B' \in Neg$ implies $I \models_O A$. I is a *model* of a dl-program $\mathscr{K} = (O, P)$, written $I \models_O \mathscr{K}$, iff I is a model of each rule of P. I is a *supported model* of $\mathscr{K} = (O, P)$ iff, for any $h \in I$, there is a rule ($h \leftarrow Pos, not\ Neg$) in P such that $I \models_O A$ for any $A \in Pos$ and $I \not\models_O B$ for any $B \in Neg$.

[1] Different from that of (Eiter *et al.* 2008), we consider ground atoms instead of literals for convenience.

A dl-atom A is *monotonic* relative to a dl-program $\mathscr{K} = (O, P)$ if $I \models_O A$ implies $I' \models_O A$, for all $I \subseteq I' \subseteq HB_P$, otherwise A is *nonmonotonic*. It is clear that if a dl-atom does not mention $\ominus$ then it is monotonic. However, a dl-atom may be monotonic even if it mentions $\ominus$. E.g., the dl-atom $DL[S \odot p, S \ominus p; \neg S](a)$ is monotonic (which is a tautology). Clearly, the $\ominus$ operator is the only one that may cause a dl-atom to be nonmonotonic. Thus one has no reason to use $\ominus$ in monotonic dl-atoms. It is a reasonable assumption that we can rewrite a monotonic dl-atom into an equivalent one without using $\ominus$ at all.

We use DL_P to denote the set of all dl-atoms that occur in P, $DL_P^+ \subseteq DL_P$ to denote the set of monotonic dl-atoms, and $DL_P^? = DL_P \setminus DL_P^+$. A dl-program $\mathscr{K} = (O, P)$ is *positive* if (i) P is "not"-free, and (ii) every dl-atom is monotonic relative to $\mathscr{K}$. It is evident that if a dl-program $\mathscr{K}$ is positive, then $\mathscr{K}$ has a (set inclusion) least model.

2.3 Strong and weak answer sets

Let $\mathscr{K} = (O, P)$ be a dl-program. The *strong dl-transform* of $\mathscr{K}$ relative to O and an interpretation $I \subseteq HB_P$, denoted by $\mathscr{K}^{s,I}$, is the positive dl-program (O, sP_O^I), where sP_O^I is obtained from P by deleting:

- the dl-rule r of the form (2) such that either $I \not\models_O B_i$ for some $1 \leqslant i \leqslant m$ and $B_i \in DL_P^?$, or $I \models_O B_j$ for some $m+1 \leqslant j \leqslant n$; and
- the nonmonotonic dl-atoms and *not* A from the remaining dl-rules where A is an atom or dl-atom.

The interpretation I is a *strong answer set* of $\mathscr{K}$ if it is the least model of $\mathscr{K}^{s,I}$.

The *weak dl-transform* of $\mathscr{K}$ relative to O and an interpretation $I \subseteq HB_P$, denoted by $\mathscr{K}^{w,I}$, is the positive dl-program (O, wP_O^I), where wP_O^I is obtained from P by deleting:

- the dl-rules of the form (2) such that either $I \not\models_O B_i$ for some $1 \leqslant i \leqslant m$ and $B_i \in DL_P$, or $I \models_O B_j$ for some $m+1 \leqslant j \leqslant n$; and
- the dl-atoms and *not* A from the remaining dl-rules where A is an atom or dl-atom.

The interpretation I is a *weak answer set* of $\mathscr{K}$ if I is the least model of $\mathscr{K}^{w,I}$.

Example 1
Consider the following dl-programs:

- $\mathscr{K}_0 = (O, P_0)$ where $O = \{c \sqsubseteq c'\}$ and $P_0 = \{w(a) \leftarrow DL[c \oplus p; c'](a); p(a) \leftarrow\}$. For this dl-program to make some sense, let's image this situation: c' and c are classes of good conference papers and ICLP papers respectively, $p(x)$ means that x is a paper in the TPLP special issue of ICLP 2010, $w(x)$ means that x is worth reading, and a stands for "this paper". Note that c and c' are concepts in O, and p and w are predicates outside of O. The communication is through the dl-rule, $w(a) \leftarrow DL[c \oplus p; c'](a)$, which says that if "this paper" is a good conference paper, given that any paper in the TPLP special issue of ICLP 2010 is an ICLP paper and ICLP papers are good conference papers (by the knowledge in O), then it is worth reading. $\mathscr{K}_0$ has exactly one strong answer set $\{p(a), w(a)\}$, which is also its unique weak answer set.

- Now, suppose someone writes $\mathcal{K}_1 = (O, P_1)$ where $O = \{c \sqsubseteq c'\}$ and $P_1 = \{p(a) \leftarrow DL[c \oplus p; c'](a)\}$. This program has a unique strong answer set $I_1 = \emptyset$ and two weak answer sets I_1 and $I_2 = \{p(a)\}$. It can be seen that there is a circular justification in the weak answer set I_2: that "this paper" is in the TPLP special issue of ICLP 2010 is justified by its being in it.
  The interested reader may verify the following. By the definition of $\oplus$, $O(I_2; c \oplus p) = O \cup \{c(a)\}$, and clearly $O \not\models c'(a)$ and $\{c(a), c \sqsubseteq c'\} \models c'(a)$. So the weak dl-transform relative to O and I_2 is $\mathcal{K}_1^{w,I_2} = (O, \{p(a) \leftarrow\})$. Since I_2 coincides with the least model of $\{p(a) \leftarrow\}$, it is a weak answer set of $\mathcal{K}_1$. Similarly, one can verify that the strong dl-transform relative to O and I_2 is $\mathcal{K}_1^{s,I_2} = (O, P_1)$. Its least model is the empty set, so I_2 is not a strong answer set of $\mathcal{K}_1$.
- $\mathcal{K}_2 = (O, P_2)$ where $O = \emptyset$ and $P_2 = \{p(a) \leftarrow DL[c \oplus p, b \ominus q; c \sqcap \neg b](a)\}$. Both $\emptyset$ and $\{p(a)\}$ are strong and weak answer sets of the dl-program.
- $\mathcal{K}_3 = (\emptyset, P_3)$ where $P_3 = \{p(a) \leftarrow DL[c \odot p, b \ominus q; \neg c \sqcap \neg b](a)\}$. $\emptyset$ and $\{p(a)\}$ are both strong and weak answer sets of the dl-program.
- $\mathcal{K}_4 = (\emptyset, P_4)$ where $P_4 = \{p(a) \leftarrow DL[c \ominus p; \neg c](a)\}$. $\mathcal{K}_4$ has no weak answer set, and thus it has no strong answer set either.

These dl-programs show that strong (and weak) answer sets may not be (set inclusion) minimal. It has been shown that if a dl-program contains no nonmonotonic dl-atoms then its strong answer sets are minimal (Eiter *et al.* 2008). However, this does not hold for weak answer sets as shown by the dl-program $\mathcal{K}_1$ above, even if it is positive. It is known that strong answer sets are always weak answer sets, but not vice versa (Eiter *et al.* 2008).

3 Completion and loop formulas

In this section, we define completion, characterize weak and strong answer sets by loop formulas, and outline an alternative method of computing weak and strong answer sets.

3.1 Completion

Given a dl-program $\mathcal{K} = (O, P)$, we assume an underlying propositional language $\mathcal{L}_{\mathcal{K}}$, such that the propositional atoms of $\mathcal{L}_{\mathcal{K}}$ include the atoms and dl-atoms occurring in P. The *formulas* of $\mathcal{L}_{\mathcal{K}}$ are defined as usual using the connectives $\neg, \wedge, \vee, \supset$ and $\leftrightarrow$. The *dl-interpretations* (or simply *interpretations* if it is clear from context) of the language $\mathcal{L}_{\mathcal{K}}$ are the interpretations relative to P, i.e., the subsets of HB_P. For a formula ψ of $\mathcal{L}_{\mathcal{K}}$ and an interpretation I of $\mathcal{L}_{\mathcal{K}}$, we say I is a *model* of ψ relative to O, denoted $I \models_O \psi$, whenever (i) if ψ is an atom, then $\psi \in I$; (ii) if ψ is a dl-atom, then $I \models_O \psi$; and (iii) the above is extended in the usual way to arbitrary formulas of $\mathcal{L}_{\mathcal{K}}$.

Let $\mathcal{K} = (O, P)$ be a dl-program and h an atom in HB_P. The *completion* of h (relative to $\mathcal{K}$), written $COMP(h, \mathcal{K})$, is the following formula of $\mathcal{L}_{\mathcal{K}}$:

$$h \leftrightarrow \bigvee_{1 \leqslant i \leqslant n} \left(\bigwedge_{A \in Pos_i} A \wedge \bigwedge_{B \in Neg_i} \neg B \right),$$

where $(h \leftarrow Pos_1, not\ Neg_1), \ldots, (h \leftarrow Pos_n, not\ Neg_n)$ are all the rules in P whose heads are the atom h. The *completion* of $\mathcal{K}$, written $COMP(\mathcal{K})$, is the collection of completions of all atoms in HB_P.

Recall that a model $M \subseteq HB_P$ of a dl-program $\mathscr{K} = (O, P)$ is a *supported model* if for any atom $a \in M$, there is a rule in P whose head is a and whose body is satisfied by M.

Proposition 1
Let $\mathscr{K} = (O, P)$ be a dl-program and I an interpretation of P. Then I is a supported model of $\mathscr{K}$ if and only if $I \models_O COMP(\mathscr{K})$.

Proposition 2
Every weak (resp. strong) answer set of a dl-program $\mathscr{K}$ is a supported model of $\mathscr{K}$.

3.2 *Weak loop formulas*

In order to capture weak answer sets of dl-programs using completion and loop formulas, we define weak loops. Formally, let $\mathscr{K} = (O, P)$ be a dl-program. The *weak positive dependency graph* of $\mathscr{K}$, written $G^w_{\mathscr{K}}$, is the directed graph (V, E), where $V = HB_P$ (note that a dl-atom is not in V), and $(u, v) \in E$ if there is a dl-rule of the form (2) in P such that $A = u$ and $B_i = v$ for some $i\,(1 \leqslant i \leqslant m)$. A nonempty subset L of HB_P is a *weak loop* of $\mathscr{K}$ if there is a cycle in $G^w_{\mathscr{K}}$ which goes through only and all the nodes in L.

Given a weak loop L of a dl-program $\mathscr{K} = (O, P)$, the *weak loop formula* of L (relative to $\mathscr{K}$), written $wLF(L, \mathscr{K})$, is the following formula of $\mathscr{L}_{\mathscr{K}}$:

$$\bigvee L \supset \bigvee_{1 \leqslant i \leqslant n} \left(\bigwedge_{A \in Pos_i} A \wedge \bigwedge_{B \in Neg_i} \neg B \right)$$

where $(h_1 \leftarrow Pos_1, not\ Neg_1), \ldots, (h_n \leftarrow Pos_n, not\ Neg_n)$ are all the rules in P such that $h_i \in L$ and $Pos_i \cap L = \emptyset$ for any $i\,(1 \leqslant i \leqslant n)$.

Theorem 1
Let $\mathscr{K} = (O, P)$ be a dl-program and I an interpretation of P. Then I is a weak answer set of $\mathscr{K}$ if and only if $I \models_O COMP(\mathscr{K}) \cup wLF(\mathscr{K})$, where $wLF(\mathscr{K})$ is the set of weak loop formulas of all weak loops of $\mathscr{K}$.

3.3 *Strong loop formulas*

Let $\mathscr{K} = (O, P)$ be a dl-program. The *strong positive dependency graph* of $\mathscr{K}$, denoted by $G^s_{\mathscr{K}}$, is the directed graph (V, E), where $V = HB_P$ and $(p(\vec{c}), q(\vec{c}\,')) \in E$ if there is a rule of the form (2) in P such that, (1) $A = p(\vec{c})$ and, (2) for some $i\,(1 \leqslant i \leqslant m)$, either

- $B_i = q(\vec{c}\,')$, or
- B_i is a monotonic dl-atom mentioning the predicate q and $\vec{c}\,'$ is a tuple of constants matching the arity of q. (If this condition is ignored then it becomes the definition of weak positive dependency graph.)

A nonempty subset L of HB_P is a *strong loop* of $\mathscr{K}$ if there is a cycle in $G^s_{\mathscr{K}}$ which passes only and all the nodes in L.

To define strong loop formulas of a dl-program $\mathscr{K} = (O, P)$, we need to extend the vocabulary Φ, such that, for any predicate symbol p and a nonempty set of atoms L, Φ contains the predicate symbol p_L that has the same arity as that of p.

Let L be a nonempty set of atoms, $A = DL[\lambda; Q](\vec{t})$ be a dl-atom. The *irrelevant formula* of A relative to L, written by $IF(A, L)$, is the conjunction of (1) $DL[\lambda_L; Q](\vec{t})$, where λ_L is obtained from λ by replacing each predicate symbol p with p_L whenever p appears in both λ and L and, (2) for each predicate symbol p mentioned in both λ and L, the instantiation on $\mathscr{C}$ (Chen *et al.* 2006) of the formula:

$$\forall \vec{X} \cdot \left[p_L(\vec{X}) \leftrightarrow \left(p(\vec{X}) \wedge \bigwedge_{p(\vec{c}) \in L} \vec{X} \neq \vec{c} \right) \right] \tag{4}$$

where $\vec{X}$ is a tuple of distinct variables matching the arity of p, and $\vec{X} \neq \vec{c}$ stands for $\neg(\vec{X} = \vec{c})$, i.e., $\neg(x_1 = c_1 \wedge \ldots \wedge x_k = c_k)$ if $\vec{X} = (X_1, \ldots, X_k)$ and $\vec{c} = (c_1, \ldots, c_k)$. Please note that, the instantiation of a formula $\forall x \cdot \psi$ on a finite set D of constants is the formula $\bigwedge_{d \in D} \psi[x/d]$, in which $c = c$ (resp., $c = c'$) is replaced with $\top$ (*true*) (resp., $\bot$ (*false*)), where c and c' are two distinct constants. In what follows, we identify the formula (4) with its instantiation whenever it is clear from its context, unless otherwise stated.

For instance, let $A = DL[c \oplus p; c](a)$ and $L = \{p(a), p(b)\}$. Then $IF(A, L)$ is the formula:

$$DL[c \oplus p_L; c](a) \wedge (p_L(a) \leftrightarrow p(a) \wedge a \neq a) \wedge (p_L(b) \leftrightarrow p(b) \wedge a \neq b)$$

which is equivalent to

$$DL[c \oplus p_L; c](a) \wedge \neg p_L(a) \wedge (p_L(b) \leftrightarrow p(b))\cdot$$

Intuitively, the irrelevant formula of A relative to L says that the truth of A only depends on the truth of the atoms not in L.

We are now in a position to define strong loop formulas. Let L be a strong loop of $\mathscr{K} = (O, P)$. The *strong loop formula* of L (relative to $\mathscr{K}$), written $sLF(L, \mathscr{K})$, is the following formula of $\mathscr{L}_{\mathscr{K}}$:

$$\bigvee L \supset \bigvee_{1 \leqslant i \leqslant n} \left(\bigwedge_{A \in Pos_i} \gamma(A, L) \wedge \bigwedge_{B \in Neg_i} \neg B \right)$$

where

- $(h_1 \leftarrow Pos_1, not\ Neg_1), \ldots, (h_n \leftarrow Pos_n, not\ Neg_n)$ are all the rules in P such that $h_i \in L$ and $Pos_i \cap L = \emptyset$ for all $i\,(1 \leqslant i \leqslant n)$,
- $\gamma(A, L) = IF(A, L)$ if A is a monotonic dl-atom, and A otherwise.

In general, we have to recognize the monotonicity of dl-atoms in order to construct strong loops of dl-programs. In this sense, the strong loops and strong loop formulas are defined semantically. If a dl-atom does not mention the operator $\ominus$ then it is obviously monotonic. Thus for the class of dl-programs in which no monotonic dl-atoms mention $\ominus$, the strong loops and strong loop formulas are given syntactically, since it is sufficient to determine the monotonicity of a dl-atom by checking whether it contains the operator $\ominus$.

Example 2
Let $\mathscr{K} = (\emptyset, P)$ be a dl-program where P consists of

$$p(a) \leftarrow DL[c \oplus p; c](a); \quad p(a) \leftarrow not\ DL[c \oplus p; c](a)\cdot$$

The dl-program $\mathscr{K}$ has a unique strong loop $L = \{p(a)\}$, but doesn't have any weak loops. Its completion is the formula:

$$p(a) \leftrightarrow DL[c \oplus p; c](a) \vee \neg DL[c \oplus p; c](a)$$

which equals to the formula $p(a) \leftrightarrow \top$, i.e., $p(a)$. Note that, the strong loop formula $sLF(L, \mathscr{K})$ is the formula:

$$p(a) \supset \left[\begin{array}{l} DL[c \oplus p_L; c](a) \wedge (p_L(a) \leftrightarrow p(a) \wedge a \neq a) \\ \vee \neg DL[c \oplus p; c](a) \end{array} \right].$$

It is clear that the interpretation $I = \{p(a)\}$ is a model of $COMP(\mathscr{K})$ relative to the DL knowledge base $O = \emptyset$. However, $I \not\models_O sLF(L, \mathscr{K})$.

Theorem 2
Let $\mathscr{K} = (O, P)$ be a dl-program and I an interpretation of P. Then I is a strong answer set of $\mathscr{K}$ if and only if $I' \models_O COMP(\mathscr{K}) \cup sLF(\mathscr{K})$, where $sLF(\mathscr{K})$ is the set of strong loop formulas of all strong loops of $\mathscr{K}$ and I' is the extension of I satisfying (4).

Since a weak loop of a dl-program $\mathscr{K}$ is also a strong loop of $\mathscr{K}$, as a by-product, our loop formula characterizations yield an alternative proof that strong answer sets are also weak answer sets.

Proposition 3
Let $\mathscr{K} = (O, P)$ be a dl-program, I an interpretation of P and L a weak loop of $\mathscr{K}$. Then we have $I' \models_O sLF(L, \mathscr{K}) \supset wLF(L, \mathscr{K})$, where I' is the extension of I satisfying (4).

3.4 An alternative method of computing weak and strong answer sets

Theorems 1 and 2 serve as the basis for an alternative method of computing weak and strong answer sets using a SAT solver, along with a dl-reasoner $\mathscr{R}$ with the following property: $\mathscr{R}$ is sound, complete, and terminating for entailment checking. Let $\mathscr{K} = (O, P)$ be a dl-program and $T = COMP(\mathscr{K})$. We replace all dl-atoms in T with new propositional atoms to produce T'. Let ξ_A be the new atom in T', for the dl-atom A in T, and X be the set of all such new atoms in T'. Below, we outline an algorithm to compute the weak answer sets of $\mathscr{K}$ (here we only describe how to compute the first such an answer set). To compute a strong answer set, replace the word weak with strong.

(i) Generate a model I of T; if there is none, then there is no weak answer set.
(ii) Check if I is a weak answer set of $\mathscr{K}$,
   (a) if yes, return I as a weak answer set of $\mathscr{K}$.
   (b) if no, add a weak loop formula into T that is not satisfied by I relative to O, and goto (i).

To generate a model of T, we compute a model M of T' using a SAT solver, and then use $\mathscr{R}$ to check the entailment: For any dl-atom A in T, if $M \models \xi_A$ then $M \models_O A$ otherwise $M \not\models_O A$. Let $M' = M/X$. It is not difficult to verify that M' is a model of $\mathscr{K}$.

The strong and weak answer set semantics of dl-programs have been implemented in a prototype system called SWLP[2], using the ASP solver DLV and a dl-reasoner. The main difference in the method outlined here is that we use a SAT solver to generate candidate models, which allows to take the advantages of the state-of-the-art SAT technology.

For strong answer sets, the construction of a strong loop formula requires checking monotonicity of dl-atoms. However, for the class of dl-programs mentioning no $\ominus$, this checking is not needed and the construction of a strong loop formula is hence tractable.

4 Canonical answer sets

4.1 Motivation: the problem of self-support

As commented by Eiter *et al.* (Eiter *et al.* 2008), some weak answer sets may be considered counterintuitive because of "self-supporting" loops. For instance, consider the weak answer set $\{p(a)\}$ of the dl-program $\mathscr{K}_1$ in Example 1. The evidence of the truth of $p(a)$ is inferred by means of a self-supporting loop: "$p(a) \Leftarrow DL[c \oplus p\,; c'](a) \Leftarrow p(a)$", which involves not only the dl-atom $DL[c \oplus p\,; c'](a)$ but the DL knowledge base O. Thus the truth of $p(a)$ depends on the truth of itself. This self-support is excluded by the strong loop formula of the loop $L = \{p(a)\}$.

Let's consider the dl-program $\mathscr{K}_2$ in Example 1 again. Note that $\{p(a)\}$ is a strong answer set of $\mathscr{K}_2$. The truth of the atom $p(a)$ depends on the truth of $[c \sqcap \neg b](a)$ which depends on the truth of $p(a)$ and $\neg q(a)$. Thus the truth of $p(a)$ depends on the truth of itself. The self-supporting loop is: "$p(a) \Leftarrow DL[c \oplus p, b \ominus q\,; c \sqcap \neg b](a) \Leftarrow (p(a) \wedge \neg q(a))$". In this sense, some strong answer sets may be considered counterintuitive as well.

The notion of "circular justification" was formally defined by (Liu and You 2008) to characterize self-supports for lparse programs, which was motivated by the notion of *unfoundedness* for logic programs (Van Gelder *et al.* 1991) and logic programs with aggregates (Calimeri *et al.* 2005). With slight modifications, we extend the concept of circular justification to dl-programs. Formally, let $\mathscr{K} = (O, P)$ be a dl-program and $I \subseteq HB_P$ be a supported model of $\mathscr{K}$. I is said to be *circularly justified* (or simply *circular*) if there is a nonempty subset M of I such that

$$I \setminus M \not\models_O \bigwedge_{A \in Pos} A \wedge \bigwedge_{B \in Neg} \neg B \tag{5}$$

for any dl-rule ($h \leftarrow Pos, not\ Neg$) in P with $h \in M$ and $I \models_O \bigwedge_{A \in Pos} A \wedge \bigwedge_{B \in Neg} \neg B$. Otherwise, we say that I is *noncircular*. Intuitively speaking, Condition (5) means that the atoms in M have no support from outside of M, i.e., they have to depend on themselves.

Example 3
Let $\mathscr{K} = (\emptyset, P)$ where P consists of

$$p(a) \leftarrow not\ DL[b \ominus p\,; \neg b](a)\cdot$$

It is not difficult to verify that $\mathscr{K}$ has two weak answer sets $\emptyset$ and $\{p(a)\}$. They are strong answer sets of $\mathscr{K}$ as well. In terms of the above definition, $\{p(a)\}$ is circular.

2 https://www.mat.unical.it/ianni/swlp/; also see (Eiter *et al.* 2008) for the details of the implementation and interesting dl-programs

It is interesting to note that weak answer sets allow self-supporting loops involving any dl-atoms (either monotonic or nonmonotonic), while strong answer sets allow self-supporting loops only involving nonmonotonic dl-atoms and their default negations. These considerations motivate us to define a new semantics which is free of circular justifications.

4.2 Canonical answer sets by loop formulas

Let $\mathscr{K} = (O, P)$ be a dl-program. The *canonical dependency graph* of $\mathscr{K}$, written $G^c_{\mathscr{K}}$, is the directed graph (V, E), where $V = HB_P$ and $(u, v) \in E$ if there is a rule of the form (2) in P such that $A = u$ and there exists an interpretation $I \subseteq HB_P$ such that either of the following two conditions holds:

(1) $I \not\models_O B_i$ and $I \cup \{v\} \models_O B_i$, for some $i\,(1 \leqslant i \leqslant m)$. In this case, we say that v is a *positive monotonic* (resp., *nonmonotonic*) dependency of B_i if B_i is a monotonic (resp., nonmonotonic) dl-atom. Intuitively, the truth of B_i may depend on that of v while the truth of u may depend on that of B_i. Thus the truth of u may depend on that of v.
(2) $I \models_O B_j$ and $I \cup \{v\} \not\models_O B_j$, for some $j\,(1 + m \leqslant j \leqslant n)$. Clearly, B_j must be nonmonotonic. In this case, we say that v is a *negative nonmonotonic dependency* of B_j. Intuitively, the truth of u may depend on that of "*not* B_j", while its truth may depend on that of v. Thus the truth of u may depend on that of v.

A nonempty subset L of HB_P is a *canonical loop* of $\mathscr{K}$ if there is a cycle in $G^c_{\mathscr{K}}$ that goes through only and all the nodes in L. It is clear that if $B_i = v$ then the interpretation $I = \{v\}$ satisfies v while $I \setminus \{v\}$ does not. Thus the notion of canonical loops is a generalization of that of weak loops given in Subsection 3.2, and a generalization of the notion of loops for normal logic programs (Lin and Zhao 2004).

Note further that the canonical dependency graph is not a generalization of the strong positive dependency graph, since some strong loops are not canonical loops. E.g., with the dl-program $\mathscr{K} = (\emptyset, P)$, where $P = \{p(a) \leftarrow DL[c \odot p, c \ominus p, \neg c](a)\}$, the dl-atom $A = DL[c \odot p, c \ominus p, \neg c](a)$ is equivalent to $\top$. So it is monotonic. It follows that $L = \{p(a)\}$ is a strong loop of $\mathscr{K}$. However L is not a canonical loop of $\mathscr{K}$ because there is no interpretation I such that $I \not\models_O A$ and $I \cup \{p(a)\} \models_O A$.

Due to the two kinds of dependencies in a canonical dependency graph defined above, to define canonical loop formulas, we need two kinds of irrelevant formulas: Let L be a set of atoms and $A = DL[\lambda; Q](\vec{t})$ a nonmonotonic dl-atom. The *positive canonical irrelevant formula* of A with respect to L, written *pCF*(A, L), is the conjunction of (1) $DL[\lambda_L; Q](\vec{t})$, where λ_L is obtained from λ by replacing each predicate p with p_L if L contains an atom $p(\vec{c})$ which is a positive nonmonotonic dependency of A and, (2) for each predicate p occurring in λ, the instantiation on $\mathscr{C}$ of the formula (4) if L contains an atom $p(\vec{c})$ which is a positive nonmonotonic dependency of A. The *negative canonical irrelevant formula* of A with respect to L, written *nCF*(A, L), is the conjunction of (1) $DL[\lambda_L; Q](\vec{t})$, where λ_L is obtained from λ by replacing each predicate p with p_L if L contains an atom $p(\vec{c})$ which is a negative nonmonotonic dependency of A and, (2) for each predicate p occurring in λ, the instantiation on $\mathscr{C}$ of the formula (4) if L contains an atom $p(\vec{c})$ which is a negative nonmonotonic dependency of A.

Let $\mathscr{K} = (O, P)$ be a dl-program, $M \subseteq HB_P$ and L a loop of $\mathscr{K}$. The *canonical loop formula* of L relative to $\mathscr{K}$ under M, written $cLF(L, M, \mathscr{K})$, is the following formula:

$$\bigvee L \supset \bigvee_{1 \leqslant i \leqslant n} \left(\bigwedge_{A \in Pos_i} \delta_1(A, L) \wedge \bigwedge_{B \in Neg_i} \neg \delta_2(B, L) \right)$$

where

- $(h_1 \leftarrow Pos_1, not\ Neg_1), \ldots, (h_n \leftarrow Pos_n, not\ Neg_n)$ are all the rules in P such that $h_i \in L$, $Pos_i \cap L = \emptyset$ and $M \models_O \bigwedge_{A \in Pos_i} A \wedge \bigwedge_{B \in Neg_i} \neg B$ for each $i\, (1 \leqslant i \leqslant n)$,
- $\delta_1(A, L) = pCF(A, L)$ if A is a nonmonotonic dl-atom, $\gamma(A, L)$ otherwise,
- $\delta_2(B, L) = nCF(B, L)$ if B is a nonmonotonic dl-atom, and B otherwise.

Given a dl-program $\mathscr{K} = (O, P)$ and an interpretation $I \subseteq HB_P$. We call I a *canonical answer set* of $\mathscr{K}$ if I' is a model of $COMP(\mathscr{K}) \cup cLF(I, \mathscr{K})$ relative to O, where I' is the extension of I satisfying (4) and $cLF(I, \mathscr{K}) = \{cLF(L, I, \mathscr{K}) | L$ is a canonical loop of $\mathscr{K}\}$. It is not difficult to prove that every canonical answer set of a dl-program $\mathscr{K}$ is a supported model of $\mathscr{K}$.

Example 4
Consider the dl-program $\mathscr{K}_2$ in Example 1, i.e., $\mathscr{K}_2 = (\emptyset, P_2)$ where $P_2 = \{p(a) \leftarrow DL[c \oplus p, b \ominus q\,; c \sqcap \neg b](a)\}$. It is easy to see that the dl-atom $DL[c \oplus p, b \ominus q\,; c \sqcap \neg b](a)$ is nonmonotonic, $\emptyset \not\models_O DL[c \oplus p, b \ominus q\,; c \sqcap \neg b](a)$, and $\{p(a)\} \models_O DL[c \oplus p, b \ominus q\,; c \sqcap \neg b](a)$. Thus $L = \{p(a)\}$ is a canonical loop of $\mathscr{K}_2$. Let $I = \{p(a)\}$. The canonical loop formula $cLF(L, I, \mathscr{K})$ is equivalent to

$$p(a) \supset DL[c \oplus p_L, b \ominus q\,; c \sqcap \neg b](a) \wedge (p_L(a) \leftrightarrow p(a) \wedge (a \neq a))$$

where the last conjunct is equivalent to $\neg p_L(a)$. Thus, the loop formula is not satisfied by the extension of I satisfying (4) relative to the knowledge base $\emptyset$. So I is not a canonical answer set of $\mathscr{K}_2$, even if I is a model of $COMP(\mathscr{K}_2)$ relative to the knowledge base $\emptyset$.

The next example demonstrates the difference among the positive dependency graphs of dl-programs.

Example 5
Let $\mathscr{K} = (O, P)$ be a dl-program where $O = \emptyset$ and P consists of the following rules:
$p(a_1) \leftarrow DL[c \oplus p, c](a_1)$, $\quad p(a_3) \leftarrow not\ DL[c \ominus p, \neg c](a_3)$,
$p(a_2) \leftarrow DL[c \oplus p, b \ominus q\,; c \sqcap \neg b](a_2)$,$p(a_4) \leftarrow p(a_4)$.
The only weak positive dependency on HB_P is $(p(a_4), p(a_4))$, the strong positive dependency includes $(p(a_1), p(a_1))$ besides the weak one, while the canonical positive dependency contains $(p(a_2), p(a_2))$ and $(p(a_3), p(a_3))$ in addition to the strong ones. Figure 1 depicts the various dependency relations on HB_P. The weak positive dependency graph is $G^w_{\mathscr{K}} = (V, E)$ where $V = \{p(a_i), q(a_i) | 1 \leqslant i \leqslant 4\}$ and $E = \{(p(a_4), p(a_4))\}$, while the strong one is $G^s_{\mathscr{K}} = (V, E')$ where $E' = E \cup \{(p(a_1), p(a_1))\}$. The canonical dependency graph is $G^c_{\mathscr{K}} = (V, E'')$ where $E'' = E' \cup \{(p(a_2), p(a_2)), (p(a_3), p(a_3))\}$.

Comparing with the previous definitions of loop formulas, in addition to the irrelevant formulas of nonmonotonic dl-atoms, the definition of canonical loop formulas has a notable distinction: it is given under a set M of atoms whose purpose is to restrict that the support

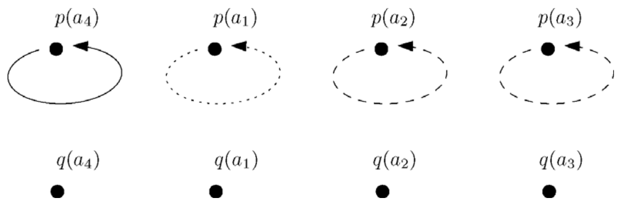

Fig. 1. The positive dependency relations on HB_P

of any atom in L come from the rules whose bodies are satisfied by M (relative to a knowledge base). The next proposition shows that the canonical loops and canonical loop formulas for dl-programs are indeed a generalization of loops and loop formulas for normal logic programs (Lin and Zhao 2004) respectively.

Proposition 4
Let P be a normal logic program, $L \subseteq HB_P$ and M a model of the completion of P.

(1) L is a loop of P if and only if L is a canonical loop of $\mathscr{K} = (\emptyset, P)$.
(2) $M \models LF(L,P)$ if and only if $M \models_O cLF(L,M,P)$, where $LF(L,P)$ is the loop formula associated with L under P (Lin and Zhao 2004) and $O = \emptyset$.

Proposition 5
Let $\mathscr{K} = (O,P)$ be a dl-program and I a canonical answer set of $\mathscr{K}$. Then I is minimal in the sense that $\mathscr{K}$ has no canonical answer set I' such that $I' \subset I$.

The following two propositions show that the canonical answer sets of dl-programs are noncircular strong answer sets. Thus canonical answer sets are weak answer sets as well.

Proposition 6
Let $\mathscr{K} = (O,P)$ be a dl-program and $I \subseteq HB_P$ a canonical answer set of $\mathscr{K}$. Then I is noncircular.

Proposition 7
Let $\mathscr{K} = (O,P)$ be a dl-program and $I \subseteq HB_P$ a canonical answer set of $\mathscr{K}$. Then I is a strong answer set of $\mathscr{K}$.

The following proposition, together with Proposition 6, implies that the operator $\ominus$ is the only cause that a strong answer set of a dl-program is circular.

Proposition 8
Let $\mathscr{K} = (O,P)$ be a dl-program in which P does not mention the operator $\ominus$. Then $I \subseteq HB_P$ is a canonical answer set of $\mathscr{K}$ if and only if I is a strong answer set of $\mathscr{K}$.

5 Related work

Integrating ASP with description logics has attracted a great deal of attention recently. The existing approaches can be roughly classified into three categories. The first is to adopt a nonmonotonic formalism that covers both ASP and first-order logic (if not for the latter, then extend it to the first-order case) (Bruijn *et al.* 2007; Motik and Rosati 2010), where ontologies and rules are written in the same language, resulting in a tight coupling. The second is a loose approach: An ontology knowledge base and the rules share the same constants but not the same predicates, and the communication is via a well-defined interface, such as dl-atoms (Eiter *et al*. 2008). The third is to combine ontologies with hybrid rules (Rosati 2005; Rosati 2006; de Bruijn *et al*. 2007), where predicates in the language of ontologies are interpreted classically, whereas those in the language of rules are interpreted nonmonotonically.

Although each approach above has its own merits, the loose approach possesses some unique advantages. In many situations, we would like to combine existing knowledge bases, possibly under different logics. In this case, a notion of interface is natural and necessary. The loose approach seems particularly intuitive, as it does not rely on the use of modal operators nor on a multi-valued logic. One notices that dl-programs share similar characteristics with another recent interest, *multi-context systems*, in which knowledge bases of arbitrary logics communicate through *bridge rules* (Brewka and Eiter 2007).

However, the relationships among these different approaches are currently not well understood. For example, although we know how to translate a dl-program without the nonmonotonic operator $\ominus$ to an MKNF theory while preserving the strong answer set semantics (Motik and Rosati 2010), when $\ominus$ is involved, no such a translation is known. Similarly, although a variant of Quantified Equilibrium Logic (QEL) captures the existing hybrid approaches, as shown by (de Bruijn *et al*. 2007), it is not clear how one would apply the loop formulas for logic programs with arbitrary sentences (Lee and Meng 2008) to dl-programs, since, to the best of our knowledge, there is no syntactic, semantics-preserving translation from dl-programs to logic programs with arbitrary sentences or to QEL.

In fact, the loop formulas for dl-programs are more involved than any previously known loop formulas, due to mixing ASP with classical first-order logic. This is evidenced by the fact that weak loop formulas permit self-supports, strong loop formulas eliminate certain kind of self-supports, and finally canonical loop formulas remove all self-supports. This seems to be a unique phenomenon that arises to dl-programs, not to any other known extensions of ASP, including logic programs with arbitrary sentences.

6 Concluding remarks

In this paper, we characterized the weak and strong answer sets of dl-programs by program completion and loop formulas. Although these loop formulas also provide an alternative mechanism for computing answer sets, building such a system presents itself as an interesting future work. We also proposed the canonical answer sets for dl-programs, which are minimal and noncircular in a formal sense. From the perspective of loop formulas, we see a notable distinction among the weak, strong and canonical answer sets: the canonical answer sets permit no circular justifications, the strong answer sets permit circular justifications involving nonmonotonic dl-atoms but not monotonic ones, whereas the weak answer sets permit circular justifications that involve any dl-atoms but not atoms.

We remark that, for a given dl-program $\mathscr{K} = (O, P)$, to decide if a set $M \subseteq HB_P$ is a strong or canonical loop and to construct the strong or canonical loop formula of M are generally quite difficult, since we have to decide the monotonicity of the dl-atoms occurring in P. The exact complexity of deciding if a set of atoms is a strong or canonical loop is one of our ongoing studies, in addition to the complexity of deciding if a given dl-program has a canonical answer set.

Acknowledgement

We thank the anonymous reviewers for their detailed comments, which helped improve the presentation of the paper.

References

BAADER, F., CALVANESE, D., MCGUINNESS, D. L., NARDI, D., AND PATEL-SCHNEIDER, P. F. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2nd ed. Cambridge University Press, New York.

BERNERS-LEE, T., HENDLER, J., AND LASSILA, O. 2001. The semantic web. *Scientific American Magazine 284,* 5, 34–43.

BREWKA, G. AND EITER, T. 2007. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proceedings of AAAI 2007.* AAAI Press, Vancouver, BC, Canada, 385–390.

BRUIJN, J., EITER, T., POLLERES, A., AND TOMPITS, H. 2007. Embedding non-ground logic programs into autoepistemic logic for knowledge-base combination. In *Proceedings of 20th International Conference on Artificial Intelligence*. Morgan Kaufman, 304–309.

CALIMERI, F., FABER, W., LEONE, N., AND PERRI, S. 2005. Declarative and computational properties of logic programs with aggregates. In *Proceedings of IJCAI 2005.* Edinburgh, Scotland, UK, 406–411.

CHEN, Y., LIN, F., WANG, Y., AND ZHANG, M. 2006. First-order loop formulas for normal logic programs. In *Proceedings of KR 2006.* AAAI Press, Lake District, UK, 298–307.

DE BRUIJN, J., PEARCE, D., POLLERES, A., AND VALVERDE, A. 2007. Quantified equilibrium logic and hybrid rules. In *Proceedings of Web Reasoning and Rule Systems, First International Conference*. Lecture Notes in Computer Science, vol. 4524. Springer, Innsbruck, Austria.

EITER, T., IANNI, G., LUKASIEWICZ, T., SCHINDLAUER, R., AND TOMPITS, H. 2008. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence 172,* 12–13, 1495–1539.

HORROCKS, I. AND PATEL-SCHNEIDER, P. F. 2004. A proposal for an OWL rules language. In *Proceedings of WWW 2004.* ACM, New York, 723–731.

LEE, J. AND MENG, Y. 2008. On loop formulas with variables. In *Proceedings of KR 2008.* AAAI Press, Sydney, 444–453.

LIN, F. AND ZHAO, Y. 2004. ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence 157,* 1–2, 115–137.

LIU, G. AND YOU, J.-H. 2008. Lparse programs revisited: Semantics and representation of aggregates. In *Proceedings of ICLP 2008.* Lecture Notes in Computer Science, vol. 5366. Springer, Udine, Italy, 347–361.

MAREK, V. W. AND TRUSZCZYNSKI, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*, K. Apt, V. Marek, M. Truszczynski, and D. Warren, Eds. Springer, Berlin, 375–398.

MOTIK, B. AND ROSATI, R. 2010. Reconciling description logics and rules. *Journal of the ACM 36,* 165–228.

NIEMELÄ, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3-4, 241–273.

ROSATI, R. 2005. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics 3,* 1, 61–73.

ROSATI, R. 2006. DL+log: Tight integration of description logics and disjunctive datalog. In *Proceedings of KR 2006.* AAAI Press, Lake District, UK, 68–78.

VAN GELDER, A., ROSS, K. A., AND SCHLIPF, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM 38,* 3, 620–650.

W3C OWL WORKING GROUP. 2009. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, Available at http://www.w3.org/TR/owl2-overview/.