Normal Description Logic Programs as Default Theories

Yisong Wang[1], Jia-Huai You[2], Li-Yan Yuan[2] and Yi-Dong Shen[3]

[1] Department of Computer Science, Guizhou University, Guiyang, China
[2] Department of Computing Science,University of Alberta, Canada
[3] Institute of Software, Chinese Academy of Sciences, China

Abstract. Eiter *et al*. formulated description logic programs (dl-programs) and defined the strong answer set semantics for these programs. There has been little understanding as how dl-programs are related to other nonmonotonic formalisms. In this paper, we show that normal dl-programs can be captured in default logic of Reiter, where a dl-program is normal if there is no monotonic dl-atom that mentions the constraint operator. This is achieved by two translations, one of which eliminates the constraint operator from nonmonotonic dl-atoms and the other translates a dl-program into a default theory. The results presented here have several implications. First, we improve a result of Motik and Rosati - the class of dl-programs that can be translated to MKNF knowledge bases are enlarged from canonical dl-programs (the dl-programs that do not contain the constraint operator) to normal dl-programs. Second, as normal dl-programs have an equivalent default logic representation, the reasoning techniques for the former are closely related to those investigated for the latter. Third, our work provides compelling evidence that default logic itself is a potential framework for integrating ontology and rules.

1 Introduction

Logic programming under the answer set semantics (ASP) has been recognized as a nonmonotonic reasoning framework for declarative problem solving [10,12]. Recently, there has been an extensive interest in combining ASP with other logics or reasoning mechanisms. One of the main interests in this direction is the integration of ASP with description logics (DLs) for the Semantic Web.

A number of proposals for integrating ontology and (nonmonotonic) rules have been put forward [3,5,6,11,14,15]. The existing approaches can be roughly classified into three categories. In the first, typically a nonmonotonic formalism is adopted that naturally embodies both first-order logic and rules, where ontology and rules are written in the same language resulting in a tight coupling [3,11]. The second is a loose approach: An ontology knowledge base and rules share the same constants but not the same predicates, and the communication is via a well-designed interface, called dl-atoms [6]. In the third, rules are treated as hybrid so that the predicates in the language of ontology are interpreted classically, whereas those in the language of rules are interpreted nonmonotonically [3,14,15].

The loose approach above stands out as quite unique and it possesses some advantages. In many practical 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 formulation of dl-programs seems particularly intuitive, as it does not rely on the use of modal operators nor on a 3-valued logic. It is worth noticing that dl-programs share many similarities with another recent interest, *multi-context systems*, in which knowledge bases under arbitrary logics communicate through *bridge rules* [2].

Informally, a dl-program is a pair (O, P), where O is an ontology knowledge base expressed in a description logic, and P a logic program, where rule bodies may contain queries to the knowledge base O, called *dl-atoms*. Such queries allow to specify inputs from a logic program to the ontology knowledge base. In more detail, a dl-atom is of the form

$$DL[S_1\ op_1\ p_1, \ldots, S_m\ op_m\ p_m; Q](\boldsymbol{t})$$

where $Q(\boldsymbol{t})$ is a query to O, and for each i $(1 \leq i \leq m)$, S_i is a concept or a role in O, p_i is a predicate symbol in P having the same arity as S_i, and the operator $op_i \in \{\oplus, \odot, \ominus\}$. Intuitively, $\oplus$ (resp., $\odot$) increases S_i (resp., $\neg S_i$) by the extension of p_i, while $\ominus$ (called the *constraint operator*) constrains S_i to p_i, i.e., for an expression $S \ominus p$, for any tuple of constants $\boldsymbol{t}$, in the absence of $p(\boldsymbol{t})$ we infer $\neg S(\boldsymbol{t})$. Eiter *et al*. proposed weak and strong answer sets for dl-programs [6].

Currently, the relationships between the loose approach and other approaches are not well understood. The only known relation is by Motik and Rosati [11], which proved that if a dl-program does not contain the constraint operator $\ominus$, then it can be translated to an MKNF knowledge base [9] while preserving its strong answer sets.[4]

In this paper, we are interested in whether dl-programs can be captured in default logic [13]. Our interest in default logic is due to the fact that default logic is one of the dominant nonmonotonic formalisms, yet despite the fact that default logic naturally accommodates first-order logic and rules (defaults), surprisingly it has not been considered explicitly as a framework of integrating ontology and rules.

The main technical result of this paper is that normal dl-programs can be translated to default theories while preserving their strong answer sets. This is achieved in two steps. In the first, we investigate the operators in dl-programs and reveal that, the constraint operator $\ominus$ is the only one causing a dl-atom to be nonmonotonic, and a dl-atom may still be monotonic even though it mentions the constraint operator $\ominus$. To eliminate $\ominus$ from nonmonotonic dl-atoms, we propose a translation π and show that, given a dl-program $\mathcal{K}$, the strong and weak answer sets of $\mathcal{K}$ correspond to the strong and weak answer sets of $\pi(\mathcal{K})$, respectively, i.e., while restricting to the language of $\mathcal{K}$, the strong and weak answer sets of $\pi(\mathcal{K})$ are precisely those of $\mathcal{K}$, and vice versa.

The class of dl-programs our translation is designed for are called *normal dl-programs*. A dl-program is normal if there are no monotonic dl-atoms that mention the constraint operator $\ominus$. Practically, since the constraint operator $\ominus$ is the only one that causes nonmonotonicity in dl-atoms, one does not have to express a monotonic dl-atom using $\ominus$. Thus, we'd like to argue that normal dl-programs represent a broad class. Our result improves a result of [11], in that we now know that a much larger class of dl-programs, the class of normal dl-programs, can be translated to MKNF knowledge bases.

[4] The theorem given in [11] (Theorem 7.6) only claims to preserve satisfiability. In a personal communication with Motik, it is confirmed that the proof of the theorem indeed establishes a one-to-one correspondence.

In the second step, we present two approaches to translating dl-programs to default theories. The difference between the two is on the handling of inconsistent ontology knowledge bases. In the first one, an inconsistent ontology knowledge base trivializes the resulting default theory, while following the spirit of dl-programs in the second approach nontrivial answer sets may still exist in the case of an inconsistent ontology knowledge base. We show that, for a canonical dl-program $\mathcal{K}$, there is an one-to-one correspondence between the strong answer sets of $\mathcal{K}$ and the extensions of its corresponding default theory (whenever the underlying knowledge base is consistent for the first approach). This, along with the result given in the first step, shows that normal dl-programs under the strong answer set semantics can be captured by default theories.

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 present a transformation to eliminate the constraint operator from nonmonotonic dl-atoms. In Section 4, we give transformations from dl-programs to default theories, followed by Sections 5 and 6 on related work and concluding remarks respectively. The proofs of the main theorems can be found at `http://webdocs.cs.ualberta.ca/~you/papers/NonMon30-full-paper.pdf`.

2 Preliminary

In this section, we briefly review the basic notations for description logics and description logic programs [6].

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. Here, we introduce the basic description logic $\mathcal{ALC}$ [1], instead of the description logics $\mathcal{SHIF}$ and $\mathcal{SHOIN}$ described in [6]. 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 $\mathcal{ALC}$, 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 $\mathcal{ALC}$ are defined as follows:

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

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

The semantics of $\mathcal{ALC}$ is defined by translating to first-order logic and then using classical first-order interpretations as its semantics. Briefly, let the transformation be ξ: (1) $\xi(A) = A(x)$, $\xi(R) = R(x, y)$ where A is an atomic concept and R a role; (2) $\xi(\forall R.C) = \forall x.R(y, x) \supset \xi(C)(x)$, and $\xi(\exists R.C) = \exists x.R(y, x) \wedge \xi(C)(x)$; (3) $\xi(\neg C) = \neg\xi(C)(x)$, $\xi(C \sqcap D) = \xi(C)(x) \wedge \xi(D)(x)$, and $\xi(C \sqcup D) = \xi(C)(x) \vee$

$\xi(D)(x)$; (4) $\xi(A(a)) = A(a)$, $\xi(R(b,c)) = R(b,c)$; (5) $\xi(C \sqsubseteq D) = \forall x.\xi(C)(x) \supset \xi(D)(x)$. Then, the semantics of $\mathcal{ALC}$ follows from that of first-order logic, so does the entailment relation $O \models F$, for a description knowledge base O and an assertion or inclusion axiom F.

2.2 Description logic programs

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

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

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

where

- each S_i is either a concept, a role or a special symbol in $\{\approx, \not\approx\}$;
- $op_i \in \{\oplus, \odot, \ominus\}$ (we call $\ominus$ the *constraint operator*);
- p_i is a unary predicate symbol in $\mathcal{P}$ if S_i is a concept, and a binary predicate symbol in $\mathcal{P}$ otherwise. The p_is are called *input predicate symbols*;
- $Q(\boldsymbol{t})$ is a *dl-query*, i.e., either (1) $C(t)$ where $\boldsymbol{t} = t$; (2) $C \sqsubseteq D$ where $\boldsymbol{t}$ is an empty argument list; (3) $R(t_1, t_2)$ where $\boldsymbol{t} = (t_1, t_2)$; (4) $t_1 \approx t_2$ where $\boldsymbol{t} = (t_1, t_2)$; or their negations, where C and D are concepts, R is a role, and $\boldsymbol{t}$ is a tuple of constants.

The precise meanings of $\{\oplus, \odot, \ominus\}$ will be defined shortly. Intuitively, $S \oplus p$ extends S by the extension of p. Similarly, $S \odot p$ extends $\neg S$ by the extension of p, and $S \ominus p$ constrains S to p. A *dl-rule* (or simply a *rule*) is an expression of the form

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

where A is an atom, each B_i $(1 \leq i \leq n)$ is an atom[5] or a dl-atom. We refer to A as its *head*, while the conjunction of $B_i (1 \leq i \leq m)$ and $not\ B_j$ $(m+1 \leq j \leq n)$ is its *body*. For convenience, we abbreviate a rule in the form (2) as

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

where $\mathit{Pos} = \{B_1, \ldots, B_m\}$ and $\mathit{Neg} = \{B_{m+1}, \ldots, B_n\}$. Let r be a rule of the form (3). If $\mathit{Neg} = \emptyset$ and $\mathit{Pos} = \emptyset$, r is a *fact* and we may write it as "A" instead of "$A \leftarrow$". A *description logic program* (*dl-program*) $\mathcal{K} = (O, P)$ 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 predicate symbols occurring in P, and $\mathcal{C}$ consists of the constants occurring in P, unless stated otherwise.

Given a dl-program $\mathcal{K} = (O, P)$, the *Herbrand base* of P, denoted by HB_P, is the set of atoms occurring in P and the ones formed from the predicate symbols of $\mathcal{P}$ occurring in some dl-atoms of P and the constant symbols in $\mathcal{C}$. It is clear that HB_P is in polynomial size of $\mathcal{K}$. 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:

[5] Different from that of [6], we consider ground atoms instead of literals for convenience.

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

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

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

A dl-atom A is *monotonic* (relative to a dl-program $\mathcal{K} = (O, P)$) if $I \models_O A$ implies $I' \models_O A$, for all I' such that $I \subseteq I' \subseteq \mathit{HB}_P$, otherwise A is *nonmonotonic*. It is clear that if a dl-atom does not mention the constraint operator then it is monotonic. However, a dl-atom may be monotonic even if it mentions constraint operators. E.g., the dl-atom $DL[S \odot p, S \ominus p; \neg S](a)$ is monotonic (which is a tautology). Evidently, the constraint operator is the only one that may cause a dl-atom to be nonmonotonic.

For convenience, 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 $\mathcal{K} = (O, P)$ is *positive* if (i) P is "not"-free, and (ii) every dl-atom is monotonic relative to $\mathcal{K}$. It is evident that if a dl-program $\mathcal{K}$ is positive, then $\mathcal{K}$ has a (set inclusion) least model. A dl-program $\mathcal{K} = (O, P)$ is *normal* if there exist no monotonic dl-atoms in P that mention the constraint operator; $\mathcal{K}$ is *canonical* if P mentions no constraint operator.

2.3 Strong and weak answer sets

Let $\mathcal{K} = (O, P)$ be a dl-program. The *strong dl-transform* of $\mathcal{K}$ relative to O and an interpretation $I \subseteq \mathit{HB}_P$, denoted by $\mathcal{K}^{s,I}$, 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 $1 \leq i \leq m$ and $B_i \in DL_P^?$, or $I \models_O B_j$ for some $m+1 \leq j \leq n$; and
– the nonmonotonic dl-atoms and $\mathit{not}\, A$ from the remaining dl-rules where A is an atom or dl-atom.

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

The *weak dl-transform* of $\mathcal{K}$ relative to O and an interpretation $I \subseteq HB_P$, denoted by $\mathcal{K}^{w,I}$, 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 $1 \leq i \leq m$ and $B_i \in DL_P$, or $I \models_O B_j$ for some $m+1 \leq j \leq n$; and
– the dl-atoms and $\mathit{not}\, A$ from the remaining dl-rules where A is an atom or dl-atom.

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

The following proposition shows that, given a dl-program $\mathcal{K} = (O, P)$, if O is inconsistent then strong and weak answer sets of $\mathcal{K}$ coincide, and are minimal.

Proposition 1. *Let $\mathcal{K} = (O, P)$ be a dl-program and $I \subseteq HB_P$. If O is inconsistent then we have that*

- *I is a strong answer set of $\mathcal{K}$ if and only if I is a weak answer set of $\mathcal{K}$.*
- *The strong and weak answer sets of $\mathcal{K}$ are minimal under set inclusion.*

Example 1. Consider the following dl-programs.

- $\mathcal{K}_1 = (O, P_1)$ 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 $I_2 = \{p(a)\}$. The interested reader may verify the following: $O(I_2; c \oplus p) = O \cup \{c(a)\}$, 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 $\mathcal{K}_1^{s,I_2} = \mathcal{K}_1$. 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 $\mathcal{K}_2$.

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 (cf. Theorem 4.13 of [6]). 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 has also been shown that strong answer sets are always weak answer sets, but not vice versa. Question arises: is it the case that, for any dl-program $\mathcal{K}$ and interpretation I, if I is a weak answer set of $\mathcal{K}$, then there is $I' \subseteq I$ such that I' is a strong answer of $\mathcal{K}$? We give a negative answer to this question by the following example.

Example 2. Let $\mathcal{K} = (\emptyset, P)$ where P consists of

$$p(a) \leftarrow DL[c \oplus p; c](a), \qquad p(a) \leftarrow not\, DL[c \oplus p; c](a).$$

Let $I = \{p(a)\}$. We have that $wP_O^I = \{p(a) \leftarrow\}$, thus I is a weak answer set of $\mathcal{K}$. However, please note that $sP_O^I = \{p(a) \leftarrow DL[c \oplus p; c](a)\}$. The least model of $\mathcal{K}^{s,I}$ is $\emptyset$ ($\neq I$). So that I is not a strong answer set of $\mathcal{K}$. Now consider $I' = \emptyset$. We have $sP_O^{I'} = \{p(a) \leftarrow DL[c \oplus p; c](a),\ p(a) \leftarrow\}$. The least model of $\mathcal{K}^{s,I'}$ is $\{p(a)\}$ ($\neq I'$). Thus I' is not a strong answer set of $\mathcal{K}$. In fact, $\mathcal{K}$ has no strong answer sets at all.

3 Eliminating the Constraint Operator

A dl-rule of the form (2) is called *dl-simple* if either (1) $n \leq 1$, or (2) each B_i is an atom for all $1 \leq i \leq n$. A dl-program $\mathcal{K} = (O, P)$ is *dl-simple* if every rule in P is dl-simple. It is obvious that, any dl-program $\mathcal{K}$ can be rewritten into another dl-program $\mathcal{K}'$ in an extended language of $\mathcal{K}$, such that (1) $\mathcal{K}'$ is dl-simple and, (2) the strong (resp.,

weak) answer sets of $\mathcal{K}$ are exactly the strong (resp., weak) answer sets of $\mathcal{K}'$ restricted to the language of $\mathcal{K}$. In what follows, for convenience and without loss of generality, we assume that dl-programs are dl-simple unless stated otherwise.

Let $\mathcal{K} = (O, P)$ be a dl-program and $r \in P$ a dl-rule. We define $\pi(r)$ to be a set of rules as follows:

(i) if r is of the form $(h \leftarrow not\, DL[\lambda; Q](\boldsymbol{t}))$, then $\pi(r)$ consists of the rule (4) and the (ground) instantiations of (5):

$$h \leftarrow not\, DL[\lambda'; Q](\boldsymbol{t}), \tag{4}$$

$$p'_i(\boldsymbol{X}_i) \leftarrow not\, p_i(\boldsymbol{X}_i),\ (1 \leq i \leq k) \tag{5}$$

(ii) if r is of the form $(h \leftarrow DL[\lambda; Q](\boldsymbol{t}))$, then $\pi(r)$ consists of

$$h \leftarrow not\, h', \tag{6}$$

$$h' \leftarrow not\, DL[\lambda'; Q](\boldsymbol{t}), \tag{7}$$

and the instantiated rules obtained from (5);

(iii) $\pi(r) = \{r\}$ otherwise;

where

- $DL[\lambda; Q](\boldsymbol{t})$ is a nonmonotonic dl-atom such that p_i $(1 \leq i \leq k)$ are all the predicate symbols occurring in λ in the form of $S_i \ominus p_i$, for some S_i,
- λ' is obtained from λ by replacing "$S_i \ominus p_i$" with "$S_i \odot p'_i$", where p'_i is a fresh predicate symbol having the same arity as p_i,
- h' is a fresh atom, and $\boldsymbol{X}_i$ is a tuple of distinct variables matching the arity of p_i for all i $(1 \leq i \leq k)$.

Intuitively, "$S \ominus p$" means that we should have $\neg S(\boldsymbol{c})$ if $p(\boldsymbol{c})$ is absent. Thus if $p'(\boldsymbol{c})$ stands for the absence of $p(\boldsymbol{c})$ for any $\boldsymbol{c}$ then "$S \ominus p$" should have the same meaning as that of "$S \odot p'$". Thus, a nonmonotonic dl-atom can be re-expressed by a monotonic dl-atom.

For a dl-program $\mathcal{K} = (O, P)$, we define $\pi(\mathcal{K}) = (O, \pi(P))$ where $\pi(P) = \bigcup_{r \in P} \pi(r)$. Please note that, due to the difficulty of checking the monotonicity of a dl-atom, the translation π of an arbitrary dl-program can be quite expensive as it depends on checking the entailment relation over the underlying description logic. However, for the class of normal dl-programs, π takes polynomial time since checking the monotonicity of dl-atoms amounts to checking the existence of the constraint operator, and the arity of predicates occurring in dl-atoms is at most 2.

Example 3 (Continued from Example 1). Consider the dl-programs in Example 1. It is clear that $\pi(\mathcal{K}_1) = \mathcal{K}_1$ and, $\pi(\mathcal{K}_2) = (\emptyset, \pi(P))$ where $\pi(P)$ is

$$\left\{ \begin{array}{ll} h \leftarrow not\, DL[c \oplus p, b \odot q'; c \sqcap \neg b](a), & \\ p(a) \leftarrow not\, h, & q'(a) \leftarrow not\, q(a) \end{array} \right\}.$$

One can verify that $\pi(\mathcal{K}_2)$ has only two strong answer sets, $\{q'(a), h\}$ and $\{q'(a), p(a)\}$. When restricted to the language of $\mathcal{K}_2$, they are $\emptyset$ and $\{p(a)\}$ respectively.

For any dl-program $\mathcal{K}$, $\pi(\mathcal{K})$ does not mention nonmonotonic dl-atoms any more. Thus, by Theorem 4.13 of [6], we have

Proposition 2. *Let $\mathcal{K}$ be a dl-program. If $I \subseteq HB_{\pi(P)}$ is a strong answer set of $\pi(\mathcal{K})$ then I is minimal, i.e, there is no $I' \subset I$ such that I' is a strong answer set of $\pi(\mathcal{K})$.*

The following example further demonstrates the translation π for dl-programs where nonmonotonic dl-atoms occur negatively.

Example 4. Consider the following dl-programs.

- Let $\mathcal{K}_1 = (\emptyset, P_1)$ where P_1 consists of

$$p(a) \leftarrow not\ DL[c \ominus p; \neg c](a).$$

  It is not difficult to verify that $\mathcal{K}_1$ has two weak answer sets $\emptyset$ and $\{p(a)\}$. They are strong answer sets of $\mathcal{K}_1$ as well. Note that, $\pi(\mathcal{K}_1) = (\emptyset, \pi(P_1))$, where $\pi(P_1)$ is

$$\{p(a) \leftarrow not\ DL[c \odot p'; \neg c](a), \quad p'(a) \leftarrow not\, p(a)\}.$$

  It is easy to see that $\pi(\mathcal{K}_1)$ has only two weak answer sets, $\{p(a)\}$ and $\{p'(a)\}$, which are also strong answer sets of $\pi(\mathcal{K}_1)$. When restricted to the language of $\mathcal{K}_1$, they are $\{p(a)\}$ and $\emptyset$ respectively.
- Let $\mathcal{K}_2 = (\emptyset, P_2)$ where P_2 consists of

$$p(a) \leftarrow not\ DL[c \ominus p, b \odot q, b \ominus q; \neg c \sqcap \neg b](a).$$

  Recall that the dl-atom $DL[b \odot q, b \ominus q; \neg b](a)$ is a tautology. The strong and weak answer sets of $\mathcal{K}_2$ are the same as those of $\mathcal{K}_1$. Please note that $\pi(P_2)$ equals

$$\left\{ \begin{array}{ll} p(a) \leftarrow not\ DL[c \odot p', b \odot q, b \odot q'; \neg c \sqcap \neg b](a), & \\ p'(a) \leftarrow not\, p(a), & q'(a) \leftarrow not\, q(a) \end{array} \right\}.$$

  Thus the strong answer sets of $\pi(\mathcal{K}_2)$ are $\{q'(a), p'(a)\}$ and $\{q'(a), p(a)\}$. They are $\emptyset$ and $\{p(a)\}$, respectively, when restricted to the language of $\mathcal{K}_2$.

The dl-programs in the above two examples show that the translation π preserves both strong and weak answer sets of a given program in the extended language, i.e., the strong and weak answer sets of $\pi(\mathcal{K})$ are those of $\mathcal{K}$ when restricted to the language of $\mathcal{K}$. As a matter of fact, we have the following key theorem.

Theorem 1. *Let $\mathcal{K} = (O, P)$ be a dl-program. An interpretation $I \subseteq HB_P$ is a strong (resp., weak) answer set of $\mathcal{K}$ if and only if $\pi(\mathcal{K})$ has a strong (resp., weak) answer set I^* such that $I = I^* \cap HB_P$.*

Unfortunately, when the transformation π is applied to eliminate the constraint operator in monotonic dl-atoms, neither strong nor weak answer sets may be preserved. For instance, consider the dl-program $\mathcal{K} = (\emptyset, P)$ where $P = \{p(a) \leftarrow DL[c \oplus p, b \ominus q; c](a)\}$. The dl-atom in it is monotonic. If we apply π to eliminate

constraint operators in monotonic dl-atoms as what we do for nonmonotonic dl-atoms, we would get the dl-program $(\emptyset, P')$ where P' equals

$$\left\{ \begin{array}{ll} h \leftarrow not\, DL[c \oplus p, b \odot q'; c](a), & \\ p(a) \leftarrow not\, h, & q'(a) \leftarrow not\, q(a) \end{array} \right\}.$$

One can verify that the dl-program has two strong answer sets, $\{p(a), q'(a)\}$ and $\{h, q'(a)\}$. They are $\{p(a)\}$ and $\emptyset$ when restricted to the language of $\mathcal{K}$. However, $\mathcal{K}$ has a unique strong answer set $\emptyset$.

Note that, to remove the constraint operator from normal dl-programs, in general we must extend the underlying language. This is because there are normal dl-programs whose strong answer sets are not minimal, but the translated dl-program contains no nonmonotonic dl-atoms hence its strong answer sets are minimal (cf. Theorem 4.13 of [6]). Therefore, there is no transformation that eliminates the constraint operator from normal dl-programs while preserving strong answer sets, yet not using extra symbols.

Recall that Motik and Rosati [11] introduced a polynomial time transformation to translate a dl-atom mentioning no constraint operators into a first-order sentence and proved that, given a canonical dl-program $\mathcal{K}$, there is a one-to-one mapping between the strong answer sets of $\mathcal{K}$ and the MKNF models of the corresponding MKNF knowledge base (Theorem 7.6 of [11]). Theorem 1 above extends their result from canonical dl-programs to normal dl-programs.

4 To Default Theories

Let's briefly recall the basic notations of default logic [13]. We assume a first-order language $\mathcal{L}$ with a signature consisting of predicate, variable and constant symbols, including equality. A *default theory* Δ is a pair (D, W) where W is a set of closed formulas (sentences) of $\mathcal{L}$, and D is a set of *defaults* of the form:

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma} \tag{8}$$

where $\alpha, \beta_i (1 \leq i \leq n), \gamma$ are formulas of $\mathcal{L}$. A default δ of the form (8) is *closed* if $\alpha, \beta_i (1 \leq i \leq n), \gamma$ are sentences, and a default theory is *closed* if all of its defaults are closed. In what follows we assume that every default theory is closed, unless stated otherwise. Let $\Delta = (D, W)$ be a default theory, and S a set of sentences. We define $\Gamma_\Delta(S)$ to be the smallest set satisfying

- $W \subseteq \Gamma_\Delta(S)$,
- $Th(\Gamma_\Delta(S)) = \Gamma_\Delta(S)$, and
- If δ is a default of the form (8) in D, and $\alpha \in \Gamma_\Delta(S)$, and $\neg\beta_i \notin S$ for each $i (1 \leq i \leq n)$ then $\gamma \in \Gamma_\Delta(S)$,

where *Th* is the classical closure operator, i.e., $Th(\Sigma) = \{\psi \mid \Sigma \vdash \psi\}$ for a set of formulas Σ. A set of sentences E is an *extension* of Δ whenever $E = \Gamma_\Delta(E)$. Alternatively, a set of sentences E is an extension of Δ if and only if $E = \bigcup_{i \geq 0} E_i$ where

- $E_0 = W$, and for $i \geq 0$

– $E_{i+1} = \mathit{Th}(E_i) \cup \{\gamma | \frac{\alpha:\beta_1,\ldots,\beta_n}{\gamma} \in D \; s.t. \; \alpha \in E_i \; \textit{and} \; \neg\beta_1, \ldots, \neg\beta_n \notin E\}$.

We present two approaches to translating a dl-program to a default theory. In the first, if the given ontology is inconsistent, the resulting default theory is trivialized and possesses a unique extension that consists of all formulas of $\mathcal{L}$, while in the second, following the spirit of dl-programs, an inconsistent ontology does not trivialize the resulting default theory.

4.1 Translation trivializing inconsistent ontology knowledge bases

We present a transformation from dl-programs to default theories. For canonical dl-programs, the transformation preserves strong answer sets. The transformation also has the property that when applied to arbitrary dl-programs directly, the default extensions of the resulting default theories are guaranteed to be strong answer sets.

Let $\mathcal{K} = (O, P)$ be a dl-program. We define $\tau(\mathcal{K})$ to be the default theory (D, W) as follows:

– W is a first-order theory corresponding to O,
– D consists of (i) for each atom $p(\boldsymbol{c}) \in \mathit{HB}_P$, the default $\frac{:\neg p(\boldsymbol{c})}{\neg p(\boldsymbol{c})}$, and (ii) for each dl-rule of the form (2) in P, the default

$$\frac{\bigwedge_{1\leq i\leq m} \tau(B_i) : \neg\tau(B_{m+1}), \ldots, \neg\tau(B_n)}{A}$$

where $\tau(A)$ is defined as
- if A is an atom then $\tau(A) = A$,
- if A is a dl-atom of the form (1) then $\tau(A)$ is the first-order sentence:

$$\left[\bigwedge_{1\leq i\leq n} \tau(S_i \; op_i \; p_i)\right] \supset Q(\boldsymbol{t}) \;\; \text{where}$$

$$\tau(S \; op \; p) = \begin{cases} \bigwedge_{\boldsymbol{c}\in\boldsymbol{\mathcal{C}}}[p(\boldsymbol{c}) \supset S(\boldsymbol{c})] & \text{if } op = \oplus \\ \bigwedge_{\boldsymbol{c}\in\boldsymbol{\mathcal{C}}}[p(\boldsymbol{c}) \supset \neg S(\boldsymbol{c})] & \text{if } op = \odot \\ \bigwedge_{\boldsymbol{c}\in\boldsymbol{\mathcal{C}}}[\neg p(\boldsymbol{c}) \supset \neg S(\boldsymbol{c})] & \text{if } op = \ominus \end{cases}$$

where $\boldsymbol{c}$ is a tuple of constants over $\mathcal{C}$ matching the arity of p and we identify $S(\boldsymbol{c})$ with its corresponding first-order sentence. Please recall that we require $\mathcal{C}$ to be finite here. Since $\boldsymbol{t}$ and $\boldsymbol{c}$ mention no variables, $\tau(A)$ has no free variables.

It is evident that, given a dl-program $\mathcal{K} = (O, P)$, each extension of $\tau(\mathcal{K})$ is equal to $\mathit{Th}(I \cup \neg\overline{I} \cup O)$, for some $I \subseteq \mathit{HB}_P$, where $\overline{I} = \{a \in \mathit{HB}_P \mid a \notin I\}$ and $\neg M = \{\neg a \mid a \in M\}$ for a set M of atoms. In particular, if O is inconsistent then $\tau(\mathcal{K})$ has a unique extension which is inconsistent.[6]

[6] However, a dl-program may have nontrivial strong answer sets even if its ontology knowledge base is inconsistent. For instance, let $\mathcal{K} = (O, P)$ where $O = \{c(a), \neg c(a)\}$ and $P = \{p \leftarrow not\, q, \;\; q \leftarrow not\, p\}$. Obviously $\mathcal{K}$ has two strong answer sets, $\{p\}$ and $\{q\}$, while $\tau(\mathcal{K})$ has a unique extension which is inconsistent. We will deal with inconsistent ontology knowledge bases in the next subsection.

For dl-atoms, the translation τ is similar to the one proposed by Motik and Rosati in [11], where they didn't consider the constraint operator. But for dl-programs, here τ also negates the atoms in HB_P by default, that is the intention of the default $\frac{:\neg p(\boldsymbol{c})}{\neg p(\boldsymbol{c})}$. It is clear that $\tau(\mathcal{K})$ is of polynomial size of the dl-program $\mathcal{K}$.

Example 5 (Continued from Example 1).

- Note that $\tau(\mathcal{K}_1) = (\{d_1, d_2\}, W)$ where $W = \{\forall x.c(x) \supset c'(X)\}$ and

$$d_1 = \frac{:\neg p(a)}{\neg p(a)}, \qquad d_2 = \frac{(p(a) \supset c(a)) \supset c'(a) :}{p(a)}.$$

  It is easy to see that $\tau(\mathcal{K}_1)$ has a unique extension $Th(\{\neg p(a)\})$.
- Note that $\tau(\mathcal{K}_2) = (\{d_1, d_2, d_3\}, \emptyset)$ where

$$d_1 = \frac{:\neg p(a)}{\neg p(a)}, \qquad d_2 = \frac{:\neg q(a)}{\neg q(a)},$$
$$d_3 = \frac{(p(a) \supset c(a)) \wedge (\neg q(a) \supset \neg b(a)) \supset c(a) \wedge \neg b(a) :}{p(a)}.$$

  One can verify that $Th(\{\neg p(a), \neg q(a)\})$ is the unique extension of $\tau(\mathcal{K}_2)$ though we know that $\mathcal{K}_2$ has two strong answer set $\emptyset$ and $\{p(a)\}$.

The default theory $\tau(\mathcal{K}_2)$ in the above example shows that, if a dl-program $\mathcal{K}$ contains the constraint operator then there may not exist a one-to-one mapping between the strong answer sets of $\mathcal{K}$ and the extensions of $\tau(\mathcal{K})$. However, the one-to-one mapping does exist for canonical dl-programs whose knowledge bases are consistent.

Theorem 2. *Let $\mathcal{K} = (O, P)$ be a canonical dl-program and $I \subseteq HB_P$. If O is consistent then I is a strong answer set of $\mathcal{K}$ if and only if $E = Th(O \cup I \cup \neg\bar{I})$ is an extension of $\tau(\mathcal{K})$.*

Since normal dl-programs can be translated into canonical dl-programs according to Theorem 1, we immediately have the following:

Corollary 1. *Let $\mathcal{K} = (O, P)$ be a normal dl-program and $I \subseteq HB_P$. If O is consistent then we have that I is a strong answer set of $\mathcal{K}$ if and only if $\tau(\pi(\mathcal{K}))$ has an extension E such that $E \cap HB_P = I$.*

For instance, for the dl-program $\mathcal{K}_2$ of Example 1, $\pi(\mathcal{K}_2)$ is given in Example 3. Thus the default theory $\tau(\pi(\mathcal{K}_2))$ equals $(D, \emptyset)$ where D consists of

$$\frac{:\neg h}{p(a)}, \qquad \frac{:\neg q(a)}{q'(a)}, \qquad \frac{:\neg((p(a) \supset c(a)) \wedge (q'(a) \supset \neg b(a)) \supset c(a) \wedge \neg b(a))}{h},$$
$$\frac{:\neg h}{\neg h}, \qquad \frac{:\neg p(a)}{\neg p(a)}, \qquad \frac{:\neg q(a)}{\neg q(a)}, \qquad \frac{:\neg q'(a)}{\neg q'(a)}.$$

It is tedious but not difficult to verify that $\tau(\pi(\mathcal{K}_2))$ has only two extensions, $E = Th(\{h, q'(a), \neg p(a), \neg q(a)\})$ and $E' = Th(\{p(a), q'(a), \neg h, \neg q(a)\})$, which correspond to the strong answer sets $\{h, q'(a)\}$ and $\{q'(a), p(a)\}$ of $\pi(\mathcal{K}_2)$ respectively, and further correspond to the strong answer sets $\emptyset$ and $\{p(a)\}$ of $\mathcal{K}_2$ respectively.

The next proposition shows that, for any dl-program $\mathcal{K}$, each extension of $\tau(\mathcal{K})$ corresponds to a strong answer set of $\mathcal{K}$ if the ontology of $\mathcal{K}$ is consistent.

Proposition 3. *Let $\mathcal{K} = (O, P)$ be a dl-program where O is consistent. If a theory E is an extension of $\tau(\mathcal{K})$ then $E \cap HB_P$ is a strong answer set of $\mathcal{K}$.*

However, as demonstrated by Example 5, the default theory $\tau(\mathcal{K}_2)$ has a unique extension $Th(\{\neg p(a), \neg q(a)\})$, but $\mathcal{K}_2$ has two strong answer sets $\emptyset$ and $\{p(a)\}$, the latter corresponds to no extension of $\tau(\mathcal{K}_2)$. Thus the inverse of the above proposition does not generally hold.

Please note that, the translation τ may not preserve weak answer sets of a normal dl-program, as shown by $\tau(\mathcal{K}_2)$ in Example 5, not even for canonical dl-programs, as shown by $\tau(\mathcal{K}_1)$ in Example 5. Before end of the subsection, let's demonstrate the translation τ by one more example.

Example 6 (Continued from Example 4).

- The default theory $\tau(\mathcal{K}_1) = (\{d_1, d_2\}, \emptyset)$, where

$$d_1 = \frac{: \neg p(a)}{\neg p(a)}, \qquad d_2 = \frac{: \neg((\neg p(a) \supset \neg c(a)) \supset \neg c(a))}{p(a)}.$$

  The interested readers can verify that both $E = Th(\{\neg p(a)\})$ and $E' = Th(\{p(a)\})$ are extensions of $\tau(\mathcal{K}_1)$, which correspond to the strong answer sets $\emptyset$ and $\{p(a)\}$ of $\mathcal{K}_1$ respectively. And $\tau(\mathcal{K}_1)$ has no other extensions.
- The default theory $\tau(\mathcal{K}_2) = (\{d_1, d_2, d_3\}, \emptyset)$ where

$$d_1 = \frac{: \neg p(a)}{\neg p(a)}, \qquad d_2 = \frac{: \neg q(a)}{\neg q(a)},$$
$$d_2 = \frac{: \neg((\neg p(a) \supset \neg c(a)) \wedge (\neg q(a) \supset \neg b(a)) \supset \neg c(a) \wedge \neg b(a))}{p(a)}.$$

  It is not difficult to see that $E = Th(\{\neg p(a), \neg q(a)\}$ and $E' = Th(\{p(a), \neg q(a)\}$ are the only two extensions of $\tau(\mathcal{K}_2)$ which correspond to the strong answer sets $\emptyset$ and $\{p(a)\}$ of $\mathcal{K}_2$, respectively.

4.2 Handling inconsistent ontology knowledge bases

Please note that in Theorem 2 and Corollary 1, we require O to be consistent. To relax the condition, we propose the following translation τ' which is slightly different from τ. Formally, given a dl-program $\mathcal{K} = (O, P)$, $\tau'(\mathcal{K})$ is the default theory $(D, \emptyset)$, where D is the same as the one in the definition of τ except for dl-atoms. Suppose A is a dl-atom of the form (1). We define $\tau'(A)$ to be the first-order sentence:

$$\left[O \wedge \left(\bigwedge_{1 \leq i \leq n} \tau(S_i \; op_i \; p_i) \right) \right] \supset Q(\boldsymbol{t})$$

where O is identified with its corresponding first-order theory. Evidently, given a dl-program $\mathcal{K}$, every extension of $\tau'(\mathcal{K})$ is consistent and has the form $Th(I \cup \neg\overline{I})$ for some $I \subseteq HB_P$.

Example 7. Let $\mathcal{K} = (O, P)$ be a dl-program where $O = \{c(a), \neg c'(a), c \sqsubseteq c'\}$ and P consists of $p(a) \leftarrow DL[c \oplus p; \neg c](a)$. It is evident that O is inconsistent and $\mathcal{K}$ has a unique strong answer set $\{p(a)\}$. Now we have that the corresponding first-order theory of O is $c(a) \wedge \neg c'(a) \wedge (\forall x.c(x) \supset c'(x))$, and $\tau'(\mathcal{K}) = (\{d_1, d_2\}, \emptyset)$ where

$$d_1 = \frac{: \neg p(a)}{\neg p(a)}, \qquad\qquad d_2 = \frac{(O \wedge (p(a) \supset c(a))) \supset \neg c(a) :}{p(a)}.$$

It is not difficult to verify that $E = Th(\{p(a)\})$ is a unique extension of $\tau'(\mathcal{K})$ which is consistent, while the unique extension of $\tau(\mathcal{K})$ is inconsistent.

The next proposition shows that the two translations τ and τ' coincide for consistent knowledge bases in the sense that there is an one-to-one mapping between the extensions of $\tau(\mathcal{K})$ and the extensions of $\tau'(\mathcal{K})$.

Proposition 4. *Let $\mathcal{K} = (O, P)$ be a dl-program where O is consistent. Then a set of sentences E is an extension of $\tau'(\mathcal{K})$ if and only if $Th(E \cup O)$ is an extension of $\tau(\mathcal{K})$.*

We have the following results, which are similar to Theorem 2, Corollary 1 and Propositions 3, respectively.

Theorem 3. *Let $\mathcal{K} = (O, P)$ be a canonical dl-program and $I \subseteq HB_P$. Then we have that I is a strong answer set of $\mathcal{K}$ iff $E = Th(I \cup \neg\overline{I})$ is an extension of $\tau'(\mathcal{K})$.*

Corollary 2. *Let $\mathcal{K} = (O, P)$ be a normal dl-program and $I \subseteq HB_P$. Then we have that I is a strong answer set of $\mathcal{K}$ if and only if $\tau'(\pi(\mathcal{K}))$ has an extension E such that $E \cap HB_P = I$.*

Proposition 5. *Let $\mathcal{K} = (O, P)$ be a dl-program. If a theory E is an extension of $\tau'(\mathcal{K})$ then $E \cap HB_P$ is a strong answer set of $\mathcal{K}$.*

5 Related Work

To the best of our knowledge, there is only one proposal to remove constraint operators in dl-programs that was proposed by Eiter *et al.* to define a well-founded semantics for dl-programs [7]. However their translation does not preserve strong answer sets of dl-programs. We denote their transformation by γ'. Given a dl-program $\mathcal{K} = (O, P)$ and a dl-rule $r \in P$, $\gamma'(r)$ consists of

(1) if $S \ominus p$ occurs in a dl-atom of r, then $\gamma'(r)$ includes the instantiated rules obtained from

$$\overline{p}(\boldsymbol{X}) \leftarrow not\ DL[S' \odot p; S'](\boldsymbol{X}),$$
$$\leftarrow p(\boldsymbol{X}), \overline{p}(\boldsymbol{X}).$$

where S' is a fresh concept (resp., role) name if S is a concept (resp., role), $\boldsymbol{X}$ is a tuple of distinct variables matching the arity of p,

(2) $\gamma'(r)$ includes the rule obtained from r by replacing each "$S \ominus p$" with "$S \odot \overline{p}$".

Similarly, we define $\gamma'(\mathcal{K}) = (O, \gamma'(P))$ where $\gamma'(P) = \bigcup_{r \in P} \gamma'(r)$. Let's consider the dl-program $\mathcal{K}_2$ in Example 1, $\gamma'(P_2)$ consists of

$$
\begin{aligned}
p(a) &\leftarrow \mathit{DL}[c \oplus p, b \odot \overline{q}; c \sqcap \neg b](a), \\
\overline{q}(a) &\leftarrow \mathit{not}\ \mathit{DL}[b' \odot q; b'](a), \\
&\leftarrow q(a), \overline{q}(a).
\end{aligned}
$$

It is not difficult to verify that $\gamma'(\mathcal{K}_2)$ has a unique strong answer set $\{\overline{q}(a)\}$. Thus this approach of eliminating the constraint operator does not generally preserve strong answer sets of dl-programs, since $\{p(a)\}$ is a strong answer set of $\mathcal{K}_2$ but there is no corresponding strong answer set for $\gamma'(\mathcal{K}_2)$.

To connect default theories with dl-programs, Eiter *et al.* [6] and Dao-Tran *et al.* [4] proposed transformations from default theories, in which only conjunctions of literals are permitted in defaults, to canonical dl-programs (with variables) and to cq-programs respectively. Informally, a cq-program can be taken as a disjunctive canonical dl-program – the heads of dl-rules can be disjunctive and no constraint operators are used. Our transformation from normal dl-programs to default theories provides a connection from the other side. Normal logic programs are special classes of normal dl-programs. And they have been related to default logic [8]. This has been now generalized by our result.

6 Conclusion

In this paper, we have shown that normal dl-programs, i.e. dl-programs in which no monotonic dl-atoms mention the constraint operator $\ominus$, can be captured in default logic. This is achieved by a translation from these dl-programs to default theories. However, it should also be clear that the problem to determine whether a dl-program is normal cannot in general be checked in polynomial time, since it involves decision of an entailment relation in the underlying description logic. We believe it is possible to extend this main result to (general) dl-programs. A potential approach is to translate monotonic dl-atoms mentioning the constraint operator into ones without the constraint operator.

The results presented in this paper have several implications. First, we have improved a result of [11]: the class of dl-programs that can be translated to MKNF knowledge bases are enlarged from canonical dl-programs (the ones without the constraint operator) to normal dl-programs. Second, as normal dl-programs have an equivalent default logic representation, the reasoning techniques for the latter can be applied to compute strong answer sets of these dl-programs. Third, our work shows that default logic is a potential framework for integrating ontology and rules.

Acknowledgment: We thank the reviewers for their detailed comments, which helped improve the presentation of this paper. Yisong Wang was partially supported by the Natural Science Foundation of China under grants 90718009 and 60963009, and the Fund of Education Department of Guizhou Province: 2008[011]. Jia-Huai You and Li Yan Yuan were supported in part by NSERC discovery grants 90718009 and 60603095, and by the 863 Project of China under grant 2009AA01Z150.

References

1. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, 2nd edition, 2007.
2. Gerhard Brewka and Thomas Eiter. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 385–390, Vancouver, British Columbia, Canada, 2007. AAAI Press.
3. J. Bruijn, T. Eiter, A. Polleres, and H. Tompits. Embedding non-ground logic programs into autoepistemic logic for knowledge-base combination. In *Proceedings of International Joint Conference On Artificial Intelligence (IJCAI-07)*, pages 304–309, 2007.
4. Minh Dao-Tran, Thomas Eiter, and Thomas Krennwallner. Realizing default logic over description logic knowledge bases. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 10th European Conference, ECSQARU 2009, Verona, Italy*, volume 5590 of *Lecture Notes in Computer Science*, pages 602–613. Springer, 2009.
5. Jos de Bruijn, David Pearce, Axel Polleres, and Agustín Valverde. Quantified equilibrium logic and hybrid rules. In *Web Reasoning and Rule Systems, First International Conference, RR 2007*, volume 4524 of *Lecture Notes in Computer Science*, Innsbruck, Austria, 2007. Springer.
6. Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artifical Intelligence*, 172(12-13):1495–1539, 2008.
7. Thomas Eiter, Thomas Lukasiewicz, Giovambattista Ianni, and Roman Schindlauer. Well-founded semantics for description logic programs in the semantic web. *ACM Transactions on Computational Logic (TOCL)*, 12(2), 2011(in press). Article 3.
8. Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
9. Vladimir Lifschitz. Nonmonotonic databases and epistemic queries. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI1991)*, pages 381–386, Sydney, Australia, 1991. Morgan Kaufmann.
10. V. Wiktor Marek and Miroslaw Truszczynski. Stable models and an alternative logic programming paradigm. In K.R. Apt, V.W. Marek, M. Truszczynski, and D.S. Warren, editors, *The Logic Programming Paradigm: A 25-Year Perspective*, pages 375–398. Springer-Verlag, Berlin, 1999.
11. Boris Motik and Riccardo Rosati. Reconciling description logics and rules. *Journal of the ACM*, 57(5):1–62, 2010.
12. Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.
13. Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.
14. Riccardo Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):61–73, 2005.
15. Riccardo Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 68–78, Lake District of the United Kingdom, 2006. AAAI Press.