Extending Logic Programs with Description Logic Expressions for the Semantic Web

Yi-Dong Shen[1] and Kewen Wang[2]

[1] State Key Laboratory of Computer Science, Chinese Academy of Sciences, China
[2] School of Computing and Information Technology, Griffith University, Australia

Abstract. Recently much attention has been directed to extending logic programming with description logic (DL) expressions, so that logic programs have access to DL knowledge bases and thus are able to reason with ontologies in the Semantic Web. In this paper, we propose a new framework for extending logic programs with DL expressions. In this framework, arbitrary DL expressions are allowed to appear in rule bodies and atomic DL expressions (i.e., atomic concepts and atomic roles) allowed in rule heads. We extend the key condition of well-supportedness for normal logic programs to logic programs with DL expressions and define an answer set semantics which satisfies the extended condition of well-supportedness. As a result, answer sets under the well-supported semantics are free of circular justifications. When an external DL knowledge base L is from the description logic $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$, it is decidable to compute all answer sets of a logic program Π with DL expressions relative to L.

1 Introduction

In the development of Semantic Web languages we are concerned with two major components: ontologies and rules. Ontologies describe terminological knowledge and rules model constraints and exceptions over the ontologies. Since the two components provide complementary descriptions of the same problem domain, they are supposed to be integrated in some ways (e.g., [1–6]; see [7] for a survey). Under the web ontology language (OWL), which is a W3C recommendation [8, 9], ontologies are expressed in description logics (DLs) [10], so they are also called DL knowledge bases.

Logic programming under the answer set semantics [11] is currently the most widely used declarative language paradigm for knowledge representation and reasoning. A (normal) logic program Π consists of rules of the form $H \leftarrow A_1, \cdots, A_m, \mathit{not}\ B_1, \cdots, \mathit{not}\ B_n$, where H and each A_i and B_i are atoms. Such a rule states that if the body $A_1, \cdots, A_m, \mathit{not}\ B_1, \cdots, \mathit{not}\ B_n$ holds, then the head H holds. The semantics of Π is defined by *answer sets*, which are Herbrand models of Π satisfying the *well-supportedness* condition [12]. Informally, a Herbrand model I is well-supported if there exists a partial order $\prec$ on I such that for any $H \in I$, there is a rule as above from Π such that I satisfies the rule body and for every A_i, $A_i \prec H$. It is this well-supportedness condition that lets

rules in a logic program differ from formulas (implications) in classical logic and guarantees that answer sets are free of circular justifications.

Recently, much attention has been directed to using logic programs to express rules in the Semantic Web by extending logic programming under the answer set semantics with DL expressions [2–4, 6]. By allowing DL expressions to appear in rules, logic programs have access to DL knowledge bases and thus are able to reason with ontologies in the Semantic Web. Major current proposals for extending logic programming with DL expressions include *description logic programs* (or DL-programs) [2, 3], $\mathcal{DL}$+log [6] and *disjunctive DL-program* [4].

Given an external DL knowledge base L, a DL-program Π extends a normal logic program by adding *dl-atoms* to rule bodies as an interface to access to L. A dl-atom is of the form $DL[S_1op_1P_1, \cdots, S_mop_mP_m; Q](\mathbf{t})$, where each $S_iop_iP_i$ semantically maps a predicate symbol P_i in Π to a concept or role S_i in L via a special interface operator $op_i \in \{ \uplus , \cup , \cap \}$, and $Q(\mathbf{t})$ is an arbitrary DL expression which will be evaluated against L after the predicate mapping. For instance, $p(a) \leftarrow DL[c \uplus p, b \cap q; c \sqcap \neg b](a)$ is a rule extended with a dl-atom $DL[c \uplus p, b \cap q; c \sqcap \neg b](a)$. This dl-atom queries L if a is in the concept c but not in the concept b, given the mapping that for any x, if $p(x)$ is true then x is in c and if $q(x)$ is false then x is not in b. As an interface-based extension, DL-programs require predicate symbols in Π be disjoint from predicate symbols (concepts or roles) in L. Therefore, to add a DL expression $Q(\mathbf{t})$ to a rule in Π, a mapping on predicate symbols between Π and L must be made by adding operations $S_iop_iP_i$ in each dl-atom. Moreover, in DL-programs DL expressions are not allowed to appear in the head of a rule. Thus, no conclusion about L can be inferred from Π. For example, in a DL-program we cannot write a rule with a head $S(X)$, where S is a concept in L.

It is necessary to allow DL expressions to occur in rule heads because DL knowledge bases (ontologies) define only general terminological knowledge, while additional constraints and exceptions over some DL concepts/roles must be defined by rules. To avoid predicate mapping between L and Π and allow DL expressions to appear in rule heads, another extension, called $\mathcal{DL}$+log, is introduced [6], which extends a normal logic program Π with DL expressions relative to L by letting Π and L share some predicate symbols so that DL expressions can appear either in bodies or heads of rules without using any interface operators. A $\mathcal{DL}$+log program consists of rules of the form $H \leftarrow A_1, \cdots, A_m$, $not\ B_1, \cdots, not\ B_n$, where H and each A_i are either atoms or atomic DL expressions (i.e. atomic concepts or atomic roles), and each B_i is an atom. Note that DL expressions are not allowed to appear behind the negation operator *not*.

To allow atomic DL expressions to appear anywhere in a rule, a third extension, called *disjunctive DL-programs*, is introduced [4]. A disjunctive DL-program consists of rules of the form $H_1 \vee \cdots, \vee H_k \leftarrow A_1, \cdots, A_m$, $not\ B_1, \cdots, not\ B_n$, where each H_i, A_i and B_i are either atoms or atomic DL expressions. As in logic programming, the semantics of a disjunctive DL-program Π relative to L is defined over Herbrand models of Π. Assume that Π is built over a first-order function-free vocabulary Φ with finite non-empty sets of constant and predicate

symbols, and that L is built on a vocabulary which includes all constants in Φ and shares some unary and binary predicate symbols with Φ. The Herbrand base of Π, denoted HB_Π, is the set of all ground atoms constructed with constant and predicate symbols from Φ. An interpretation I is any subset of HB_Π. Let $\neg I^- = \{\neg A \mid A \in HB_\Pi \setminus I\}$. I is a model of Π relative to L if (1) I is a model of Π, and (2) $L \cup I \cup \neg I^-$ is satisfiable. The *FLP-reduct* of Π w.r.t. I is a disjunctive DL-program Π^I, which consists of all ground instances of rules of Π whose bodies are satisfied by I. An interpretation I is an answer set of Π relative to L if I is a minimal model of Π^I relative to L. This answer set semantics is a proper extension of the answer set semantics for normal logic programs. However, the well-supportedness condition of the answer set semantics is not extended to disjunctive DL-programs, so that answer sets of disjunctive DL-programs may incur circular justifications by self-supporting loops. As an example, consider a disjunctive DL-program

$$\begin{array}{ll} \Pi: & A(g). \quad B(g) \leftarrow C(g). \quad C(g) \leftarrow D(g). \\ L: & D \equiv (A \sqcap \neg C) \sqcup B. \end{array}$$

The formula in L is a DL axiom, which can be translated to a first-order formula $\forall_X D(X) \equiv (A(X) \wedge \neg C(X)) \vee B(X)$. $I = \{A(g), B(g), C(g), D(g)\}$ is an answer set of Π relative to L. Observe that the evidence of the truth of $B(g), C(g), D(g)$ in the answer set can only be inferred via a self-supporting loop $B(g) \Leftarrow C(g) \Leftarrow D(g) \Leftarrow B(g)$.

As we mentioned earlier, well-supportedness is a key condition of logic programming under the answer set semantics and thus it is desirable yet necessary to extend this condition to logic programs with DL expressions. In fact, the well-supportedness condition has recently been extended to DL-programs and a new answer set semantics has been developed for DL-programs, which satisfies the extended well-supportedness condition [13]. Therefore, in this paper we advance one step further and improve disjunctive DL-programs by introducing a fourth extension of logic programs with DL expressions as follows:

1. Given an external DL knowledge base L, we extend a normal logic program Π with DL expressions relative to L by introducing rules of the form $H \leftarrow A_1, \cdots, A_m, not\ B_1, \cdots, not\ B_n$, where H is an atom or an atomic DL expression, and each A_i and B_i are either atoms or arbitrary DL expressions. Note that like DL-programs, we accommodate arbitrary DL expressions in rule bodies, and like $\mathcal{DL}$+log and disjunctive DL-programs, we use no interface operators and allow atomic DL expressions to appear in rule heads.
2. As one extreme case, in $\mathcal{DL}$+log, DL concepts and roles occurring in Π are all interpreted with first-order interpretations and take arbitrary truth values from all first-order models of L (as in first-order logic). As another extreme case, in disjunctive DL-programs, these concepts and roles are all included in the Herbrand base of Π, thus they are all interpreted with Herbrand interpretations and take truth values only from all minimal Herbrand models of Π (as in logic programming). In our framework, everyone of these DL concepts and roles occurring in Π can choose between first-order interpretations and Herbrand interpretations, depending on specific applications.

3. We extend the well-supportedness condition from normal logic programs to logic programs with the above rules, and define an answer set semantics that satisfies the extended well-supportedness condition and thus whose answer sets are free of circular justifications. When the DL knowledge base L is empty and Π contains no DL expressions, the semantics coincides with the standard answer set semantics [11].
4. Like DL-programs, we show that when L is from the description logic $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$ [9], it is decidable to compute all answer sets of Π relative to L.

2 Preliminaries

2.1 Logic Programs

Let $\Phi = (\mathbf{P}, \mathbf{C})$, where $\mathbf{P}$ is a finite set of predicate symbols and $\mathbf{C}$ a nonempty finite set of constants. A *term* is either a constant from $\mathbf{C}$ or a variable. Predicate symbols begin with a capital letter, and constants with a lower case letter. We use strings starting with X, Y or Z to denote variables. An *atom* is of the form $P(t_1, ..., t_m)$, where P is a predicate symbol from $\mathbf{P}$, and t_i is a term. A *rule* r is of the form

$$H \leftarrow A_1, \cdots, A_m, not\ B_1, \cdots, not\ B_n \tag{1}$$

where H, A_i and B_i are atoms. Each A_i is called a *positive literal*, and each $not\ B_i$ called a *negative literal*. We use $head(r)$ and $body(r)$ to denote the head H and the body $A_1, \cdots, A_m, not\ B_1, \cdots, not\ B_n$, respectively. We also use $pos(r)$ to denote the positive literals $A_1, \cdots, A_m$, and $neg(r)$ to denote the negative literals $not\ B_1, \cdots, not\ B_n$. Therefore, a rule r can simply be written as $head(r) \leftarrow body(r)$ or $head(r) \leftarrow pos(r), neg(r)$.

A *normal logic program* Π consists of a finite set of rules. A *ground instance* of a rule r is obtained by replacing every variable in r with a constant from $\mathbf{C}$. We use $ground(\Pi)$ to denote the set of all ground instances of rules in Π. The *Herbrand base* of Π, denoted HB_Π, is the set of all ground atoms $P(t_1, ..., t_m)$, where $P \in \mathbf{P}$ occurs in Π and t_i is in $\mathbf{C}$. Any subset of HB_Π is an *interpretation*. For an interpretation I, let $I^- = HB_\Pi \setminus I$ and $\neg I^- = \{\neg A \mid A \in I^-\}$.

For a normal logic program Π, an interpretation I *satisfies* a ground atom $A \in HB_\Pi$ if $A \in I$, and I satisfies $not\ A$ if $A \notin I$. For a rule r in $ground(\Pi)$, I satisfies $body(r)$ if for each (positive or negative) literal l in $body(r)$, I satisfies l; I satisfies r if I does not satisfy $body(r)$ or I satisfies $head(r)$. I is a *model* of Π if I satisfies all $r \in ground(\Pi)$. A *minimal model* is a model that is minimal in terms of set inclusion.

Let $\Pi^I = \{A \leftarrow pos(r) \mid A \leftarrow pos(r), neg(r) \in ground(\Pi)$ and I satisfies $neg(r)\}$. Since Π^I has no negative literals in rule bodies, it has the least model. The *standard answer set semantics* defines I to be an *answer set* of Π if I is the least model of Π^I [11].

2.2 DL Knowledge Bases

We assume familiarity with the basics of description logics (DLs) [10], and consider DLs which are built over 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, atomic roles* and *individuals*, respectively. This can easily be extended to DLs with *datatypes*, such as $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, following the formulation of [3]. $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ provide the logical account for the Web ontology languages OWL Lite and OWL DL, respectively, and the satisfiability of a DL knowledge base from them is decidable [9].

A *role* is either an atomic role R from $\mathbf{R}$ or its inverse, denoted R^-. General *concepts* C are formed from atomic concepts, roles and individuals, according to the following syntax:

$$C ::= \top \mid \bot \mid A \mid \{a_1, ..., a_n\} \mid C \sqcap C_1 \mid C \sqcup C_1 \mid \neg C \mid \exists R.C \mid \forall R.C \mid \geq_n R \mid \leq_n R$$

where A is an atomic concept from $\mathbf{A}$, R is a role, a_i is an individual from $\mathbf{I}$, C and C_1 are concepts, and n is a non-negative integer. An *axiom* is of the form $C \sqsubseteq D$ (*concept inclusion axiom*), $R \sqsubseteq R_1$ (*role inclusion axiom*), Trans(R) (*transitivity axiom*), $C(a)$ (*concept membership axiom*), $R(a, b)$ (*role membership axiom*), $=(a, b)$ (*equality axiom*), or $\neq(a, b)$ (*inequality axiom*), where C, D are concepts, R, R_1 are atomic roles in $\mathbf{R}$, and a, b are individuals in $\mathbf{I}$. We use $C \equiv D$ to denote $C \sqsubseteq D$ and $D \sqsubseteq C$.

A *DL knowledge base* L is a finite set of axioms. Since DLs are fragments of first-order logic with equality, where atomic concepts (resp. roles) are unary (resp. binary) predicate symbols, and individuals are constants, L has the first-order semantics. When we say predicate symbols in L, we refer to atomic concepts or atomic roles in L. L is *consistent* (or *satisfiable*) if L has a model. For an axiom F, the *entailment* relation $L \models F$ is defined as in classical logic, i.e., L entails F if all models of L are models of F. Note that if L is inconsistent, L entails everything.

A *DL expression*, also called a *DL query* in [3], which is allowed to appear in rules of a logic program, is either (i) a concept inclusion axiom F or its negation $\neg F$; or (ii) of the form $C(t)$ or $\neg C(t)$, where C is a concept, and t is a term (i.e., a variable or a constant); or (iii) of the form $R(t_1, t_2)$ or $\neg R(t_1, t_2)$, where R is a role, and t_1 and t_2 are terms; or (iv) of the form $=(t_1, t_2)$ or $\neq(t_1, t_2)$, where t_1 and t_2 are terms. An *atomic DL expression* is either $C(t)$ or $R(t_1, t_2)$, where C is an atomic concept and R an atomic role. For convenience, we denote a DL expression by $Q(\mathbf{t})$, where $\mathbf{t}$ denotes all terms occurring in the expression (e.g., t_1 and t_2 in (iii)), and Q denotes the remaining part of the expression (e.g., R or $\neg R$ in (iii)).

3 Logic Programs with DL Expressions

Let L be a DL knowledge base built over a vocabulary $\Psi = (\mathbf{A} \cup \mathbf{R}, \mathbf{I})$, and Π be a normal logic program built over $\Phi = (\mathbf{P}, \mathbf{C})$. To extend Π with DL expressions

relative to L, we first extend Φ such that: (i) all constants in $\mathbf{C}$ are individuals in $\mathbf{I}$ (i.e., $\mathbf{C} \subseteq \mathbf{I}$), so that constants occurring in DL expressions are individuals, and (ii) some concepts and roles in $\mathbf{A} \cup \mathbf{R}$ are included in $\mathbf{P}$ (as unary and binary predicate symbols, respectively), so that we can make conclusions about them in the same way as other predicate symbols in $\mathbf{P}$. To ensure decidability, we require that $\mathbf{P}$ and $\mathbf{C}$ be finite. Let $\Omega = \mathbf{P} \cap (\mathbf{A} \cup \mathbf{R})$ denote the predicate symbols shared by Π and L.

Definition 1. Let L be a DL knowledge base. A *logic program Π with DL expressions relative to L* consists of a finite set of rules of form (1), where H is an atom or an atomic DL expression, and each A_i and B_i is either an atom or a DL expression.

A *ground instance* of a rule (resp. a DL expression) in Π is obtained by replacing all variables with constants in $\mathbf{C}$. Let $ground(\Pi)$ denote the set of ground instances of all rules in Π. The *Herbrand base* HB_Π of Π relative to L is the set of all ground atoms $P(t_1, ..., t_m)$, where $P \in \mathbf{P}$ occurs either in Π or in L and t_i is in $\mathbf{C}$. Any subset of HB_Π is an *interpretation* of Π relative to L. When the context is clear, we omit the phrase "relative to L."

For an interpretation I, let $I|_\Omega = \{A \in I \mid$ the predicate symbol of A is in $\Omega\}$ and $I^-|_\Omega = \{A \in I^- \mid$ the predicate symbol of A is in $\Omega\}$. We say that I is *consistent* with L if $L \cup I|_\Omega \cup \neg I^-|_\Omega$ is consistent. Note that when I is consistent with L, L must be consistent.

Since DL expressions must be evaluated against L, the satisfaction relation for normal logic programs needs to be extended. In the sequel, by a *literal* we refer to A or *not* A, where A is an atom or a DL expression.

Definition 2. Let Π be a logic program with DL expressions relative to a DL knowledge base L, I an interpretation, and l a ground literal. We use $I \models_L l$ to denote that I *satisfies* l *under* L, which is defined as follows:

1. For a ground atom $A \in HB_\Pi$, which is not an atomic DL expression, $I \models_L A$ if $A \in I$.
2. For a ground DL expression A, $I \models_L A$ if $L \cup I|_\Omega \cup \neg I^-|_\Omega \models A$.
3. For a ground atom or a ground DL expression A, $I \models_L not\ A$ if $I \not\models_L A$.

For a rule r in $ground(\Pi)$, $I \models_L body(r)$ if for each (positive or negative) literal l in $body(r)$, $I \models_L l$; $I \models_L r$ if $I \not\models_L body(r)$ or $I \models_L head(r)$. I is a *model* of Π relative to L if I is consistent with L and $I \models_L r$ for all $r \in ground(\Pi)$. Note that when L is inconsistent, Π has no model relative to L.

Example 1. Let $L = \{\neg B(a)\}$ and $\Pi = \{A(X) \leftarrow not\ \neg(A \sqcup B)(X)\}$. Let $\mathbf{P} = \{A\}$, $\mathbf{C} = \{a\}$ and $\Omega = \{A\}$. Note that $\neg(A \sqcup B)(X)$ is a DL expression, and $A(X)$ is both an atom and an atomic DL expression. We have $HB_\Pi = \{A(a)\}$ and $ground(\Pi) = \{A(a) \leftarrow not\ \neg(A \sqcup B)(a)\}$. Π has two models relative to L: $I_1 = \emptyset$ and $I_2 = \{A(a)\}$. For the rule r in $ground(\Pi)$, $I_1 \not\models_L body(r)$, $I_2 \models_L body(r)$, and $I_2 \models_L head(r)$.

3.1 Well-Supported Models

The notion of well-supportedness in logic programming is defined by Fages in [12]: For a normal logic program Π, an interpretation I is *well-supported* if there exists a strict well-founded partial order $\prec$ on I such that for any $A \in I$, there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that I satisfies $body(r)$ and for every positive literal B in $body(r)$, $B \prec A$. A binary relation $\leq$ is *well-founded* if there is no infinite decreasing chain $A_0 \geq A_1 \geq \cdots$. A well-supported interpretation I guarantees that every $A \in I$ is free of circular justifications in I.

To extend Fages' well-supportedness condition to logic programs with DL expressions, we introduce a notion of *up to satisfaction*.

Definition 3. Let Π be a logic program with DL expressions relative to a DL knowledge base L, I an interpretation consistent with L, and l a ground literal. For any $E \subseteq I$, we use $(E, I) \models_L l$ to denote that E *up to* I *satisfies* l *under* L, which is defined as follows: For any ground atom or ground DL expression A, $(E, I) \models_L A$ if for every F with $E \subseteq F \subseteq I$, $F \models_L A$; $(E, I) \models_L not\ A$ if for no F with $E \subseteq F \subseteq I$, $F \models_L A$.

For a rule r in $ground(\Pi)$, $(E, I) \models_L body(r)$ if for every literal l in $body(r)$, $(E, I) \models_L l$. As the phrase "up to" suggests, for any ground (positive or negative) literal l, $(E, I) \models_L l$ means that for all interpretations F between E and I, $F \models_L l$. This implies that the truth of l depends on E and $\neg I^-$ and is independent of atoms in $I \setminus E$, since for any $A \in I \setminus E$ and any interpretation F with $E \subseteq F \subseteq I$, whether or not A is in F, $F \models_L l$.

The following result shows that this up to satisfaction has the property of monotonicity.

Theorem 1. *Let A be a ground atom or ground DL expression. For any $E_1 \subseteq E_2 \subseteq I$, if $(E_1, I) \models_L A$ then $(E_2, I) \models_L A$; if $(E_1, I) \models_L not\ A$ then $(E_2, I) \models_L not\ A$.*

Proof: When $(E_1, I) \models_L A$, $F \models_L A$ for every F with $E_1 \subseteq F \subseteq I$. Since $E_1 \subseteq E_2 \subseteq I$, $F \models_L A$ for every F with $E_2 \subseteq F \subseteq I$. Hence $(E_2, I) \models_L A$. When $(E_1, I) \models_L not\ A$, $F \not\models_L A$ for every F with $E_1 \subseteq F \subseteq I$. Since $E_1 \subseteq E_2 \subseteq I$, $F \not\models_L A$ for every F with $E_2 \subseteq F \subseteq I$. Thus $(E_2, I) \models_L not\ A$. □

In addition, the up to satisfaction has the following two useful properties.

Proposition 1. *For any ground DL expression A, $(E, I) \models_L A$ iff $L \cup E|_\Omega \cup \neg I^-|_\Omega \models A$.*

Proof: $(E, I) \models_L A$ means that $E \subseteq I$ and for every F with $E \subseteq F \subseteq I$, $F \models_L A$. Then, by Definition 2, $(E, I) \models_L A$ means that for every F with $E \subseteq F \subseteq I$, $L \cup F|_\Omega \cup \neg F^-|_\Omega \models A$. Note that $F = E \cup (F \setminus E)$ and $F^- = I^- \cup (I \setminus F)$. So $(E, I) \models_L A$ means that for every F with $E \subseteq F \subseteq I$,

$$L \cup E\,|_\Omega \cup \neg I^-\,|_\Omega \cup (F \setminus E)\,|_\Omega \cup \neg (I \setminus F)\,|_\Omega \models A \tag{2}$$

Then, to prove this proposition it suffices to prove that the entailment (2) holds for every F with $E \subseteq F \subseteq I$ iff the following entailment holds:

$$L \cup E\,|_{\Omega} \cup \neg I^{-}\,|_{\Omega} \models A \tag{3}$$

Note that for any model M of the left side of the entailment (2) or (3), we have $E \subseteq M \subseteq I$.

Assume that the entailment (2) holds for every F with $E \subseteq F \subseteq I$. Let M be a model of the left side of the entailment (3). Since $E \subseteq M \subseteq I$, M is a model of the left side of the entailment (2), where $F = M$. Then, M is a model of A (the right side of the entailment (2)). This means the entailment (3) holds.

Conversely, assume the entailment (3) holds. Let M be a model of the left side of the entailment (2). M is also a model of the left side of the entailment (3) and thus M is a model of A (the right side of the entailment (3)). This means the entailment (2) holds. □

Proposition 2. *For any ground atom $A \in HB_{\Pi}$, which is not an atomic DL expression, $(E, I) \models_L A$ iff $A \in E$; $(E, I) \models_L$ not A iff $A \notin I$.*

Proof: By Definitions 3 and 2, $(E, I) \models_L A$ iff for every F with $E \subseteq F \subseteq I$, $A \in F$ iff $A \in \bigcap_{E \subseteq F \subseteq I} F$ iff $A \in E$. $(E, I) \models_L$ *not* A iff for every F with $E \subseteq F \subseteq I$, $F \not\models_L A$ (i.e., $A \notin F$) iff $A \notin \bigcup_{E \subseteq F \subseteq I} F$ iff $A \notin I$. □

We extend the well-supportedness condition for normal logic programs to logic programs with DL expressions by means of the up to satisfaction.

Definition 4. Let Π be a logic program with DL expressions relative to a DL knowledge base L, and I an interpretation consistent with L. I is *well-supported* if there exists a strict well-founded partial order $\prec$ on I such that for any $A \in I$, there exists $E \subset I$, where for every $B \in E$, $B \prec A$, such that either (i) $L \cup E|_{\Omega} \cup \neg I^{-}|_{\Omega} \models A$, or (ii) there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(E, I) \models_L body(r)$.

By Proposition 1, the conditions (i) and (ii) imply that the truth of $A \in I$ is determined by E and $\neg I^{-}$. Since for every $B \in E$, $B \prec A$, the truth of A is not circularly dependent on itself. As a result, a well-supported interpretation I of Π guarantees that every $A \in I$ is free of circular justifications in I.

Observe in Definition 4 that due to the occurrence of DL expressions, some $A \in I$ may be supported by no rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $I \models_L body(r)$. Instead, A is supported by L such that $L \cup I|_{\Omega} \cup \neg I^{-}|_{\Omega} \models A$. This is a special property of well-supportedness for logic programs with DL expressions. The next example further illustrates this property.

Example 2. Let $L = \{B(a), B \sqsubseteq A\}$ and $\Pi = \{A(X) \leftarrow C(X)\}$. Let $\mathbf{P} = \{A, C\}$, $\mathbf{C} = \{a\}$ and $\Omega = \{A\}$. We have $HB_{\Pi} = \{A(a), C(a)\}$ and $ground(\Pi) = \{A(a) \leftarrow C(a)\}$. Π has two models relative to L: $I_1 = \{A(a)\}$ and $I_2 = \{A(a), C(a)\}$. Only I_1 is a well-supported model, where for $A(a) \in I_1$, we have $E = \emptyset$ and condition (i) of Definition 4 holds. Note that there is no rule of the form $A(a) \leftarrow body(r)$ in $ground(\Pi)$ such that $I_1 \models_L body(r)$.

The following result shows that Definition 4 is a proper extension to Fages' well-supportedness condition.

Theorem 2. *Let $L = \emptyset$ and Π be a normal logic program without DL expressions. An interpretation I is a well-supported model of Π relative to L iff I is a well-supported model of Π under Fages' definition.*

Proof: Let I be an interpretation of Π relative to L. I is also an interpretation of Π. Since $L = \emptyset$, for any $A \in I$ the condition (i) of Definition 4 does not hold. Since Π is a normal logic program without DL expressions, each A_i and B_i occurring in the body of each rule r of the form $A \leftarrow A_1, \cdots, A_m, not\ B_1, \cdots, not\ B_n$ in $ground(\Pi)$ are ground atoms. For such rules, $I \models_L body(r)$ iff every A_i is in I and no B_i is in I.

Assume that I is a well-supported model of Π relative to L. By Definition 4, there exists a strict well-founded partial order $\prec$ on I such that for any $A \in I$, there exists $E \subset I$, where for every $B \in E$, $B \prec A$, and there is a rule r as above in $ground(\Pi)$ such that $(E, I) \models_L body(r)$. Note that $(E, I) \models_L body(r)$ implies $E \models_L body(r)$ and $I \models_L body(r)$, which implies that both I and E satisfy $body(r)$. This means that for every positive literal A_i in $body(r)$, $A_i \in E$ and thus $A_i \prec A$. As a result, for any $A \in I$, there is a rule r as above in $ground(\Pi)$ such that I satisfies $body(r)$ and for every positive literal A_i in $body(r)$, $A_i \prec A$. This shows that I is a well-supported model of Π under Fages' definition.

Assume I is a well-supported model of Π under Fages' definition. There exists a strict well-founded partial order $\prec$ on I such that for any $A \in I$, there is a rule r as above in $ground(\Pi)$ such that I satisfies $body(r)$ and for every positive literal A_i in $body(r)$, $A_i \prec A$. Let $E \subset I$ and for every $A_i \in body(r)$, $A_i \in E$. Then, E contains no B_i in $body(r)$, since no B_i is in I. For any F with $E \subseteq F \subseteq I$, F satisfies $body(r)$ and thus $F \models_L body(r)$. That means $(E, I) \models_L body(r)$. As a result, for any $A \in I$, there exists $E \subset I$, where for every $B \in E$, $B \prec A$, and there is a rule r as above in $ground(\Pi)$ such that $(E, I) \models_L body(r)$. This shows that I is a well-supported model of Π relative to L. □

3.2 Well-Supported Answer Set Semantics

We define an answer set semantics for logic programs with DL expressions whose answer sets are well-supported models. We first introduce an immediate consequence operator $\mathcal{T}_\Pi$.

Definition 5. Let Π be a logic program with DL expressions relative to a DL knowledge base L, and I an interpretation consistent with L. For $E \subseteq I$, define

$$\mathcal{T}_\Pi(E, I) = \{A \mid A \leftarrow body(r) \in ground(\Pi) \text{ and } (E, I) \models_L body(r)\}.$$

By Theorem 1, when the second argument I is a model of Π, $\mathcal{T}_\Pi$ is monotone w.r.t. its first argument E.

Theorem 3. *Let Π be a logic program with DL expressions relative to a DL knowledge base L, and I a model of Π relative to L. For any $E_1 \subseteq E_2 \subseteq I$, $\mathcal{T}_\Pi(E_1, I) \subseteq \mathcal{T}_\Pi(E_2, I) \subseteq I$.*

Proof: For any $A \in \mathcal{T}_\Pi(E_1, I)$, there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(E_1, I) \models_L body(r)$. Since $E_1 \subseteq E_2$, by Theorem 1, $(E_2, I) \models_L body(r)$, and thus $A \in \mathcal{T}_\Pi(E_2, I)$. This shows $\mathcal{T}_\Pi(E_1, I) \subseteq \mathcal{T}_\Pi(E_2, I)$. Since $E_2 \subseteq I$, it follows $(I, I) \models_L body(r)$ and $A \in \mathcal{T}_\Pi(I, I)$. Therefore, $\mathcal{T}_\Pi(E_1, I) \subseteq \mathcal{T}_\Pi(E_2, I) \subseteq \mathcal{T}_\Pi(I, I)$. Note that $(I, I) \models_L body(r)$ means $I \models_L body(r)$. Since I is a model of Π relative to L, $I \models_L body(r)$ implies $A \in I$. This shows that when I is a model of Π relative to L, every $A \in \mathcal{T}_\Pi(I, I)$ is in I. Hence, $\mathcal{T}_\Pi(E_1, I) \subseteq \mathcal{T}_\Pi(E_2, I) \subseteq \mathcal{T}_\Pi(I, I) \subseteq I$. □

Therefore, for any model I of Π relative to L, the sequence $\langle \mathcal{T}_\Pi^i(\emptyset, I)\rangle_{i=0}^\infty$, where $\mathcal{T}_\Pi^0(\emptyset, I) = \emptyset$ and $\mathcal{T}_\Pi^{i+1}(\emptyset, I) = \mathcal{T}_\Pi(\mathcal{T}_\Pi^i(\emptyset, I), I)$, converges to a fixpoint, denoted $\mathcal{T}_\Pi^\alpha(\emptyset, I)$. This fixpoint has the following interesting properties.

Theorem 4. *Let I be a model of Π relative to L. (1) $\mathcal{T}_\Pi^\alpha(\emptyset, I) \subseteq I$. (2) $L \cup \mathcal{T}_\Pi^\alpha(\emptyset, I)|_\Omega \cup \neg I^-|_\Omega$ is consistent. (3) For any model J of Π relative to L with $J \subset I$, $\mathcal{T}_\Pi^\alpha(\emptyset, I) \subseteq \mathcal{T}_\Pi^\alpha(\emptyset, J) \subseteq J$.*

Proof: (1) It suffices to prove that for any $i \geq 0$, $\mathcal{T}_\Pi^i(\emptyset, I) \subseteq I$. It is obvious for $i = 0$. Assume $\mathcal{T}_\Pi^k(\emptyset, I) \subseteq I$ for $k \geq 0$. For $i = k + 1$, by Theorem 3, $\mathcal{T}_\Pi^{k+1}(\emptyset, I) = \mathcal{T}_\Pi(\mathcal{T}_\Pi^k(\emptyset, I), I) \subseteq I$. Therefore, $\mathcal{T}_\Pi^\alpha(\emptyset, I) \subseteq I$.

(2) Since I is a model of Π relative to L, $L \cup I|_\Omega \cup \neg I^-|_\Omega$ is consistent. Since $\mathcal{T}_\Pi^\alpha(\emptyset, I) \subseteq I$, $L \cup \mathcal{T}_\Pi^\alpha(\emptyset, I)|_\Omega \cup \neg I^-|_\Omega$ is also consistent.

(3) By Definition 3, for any rule body $body(r)$ in $ground(\Pi)$ and any E_1 and E_2 with $E_1 \subseteq E_2 \subseteq J$, if $(E_1, I) \models_L body(r)$ then $(E_2, J) \models_L body(r)$. Then, by Definition 5, $\mathcal{T}_\Pi(E_1, I) \subseteq \mathcal{T}_\Pi(E_2, J)$. Next we prove that for any $i \geq 0$, $\mathcal{T}_\Pi^i(\emptyset, I) \subseteq \mathcal{T}_\Pi^i(\emptyset, J) \subseteq J$. It is obvious for $i = 0$. Assume $\mathcal{T}_\Pi^i(\emptyset, I) \subseteq \mathcal{T}_\Pi^i(\emptyset, J) \subseteq J$ for any $i \leq k \geq 0$. For $i = k + 1$, by Theorem 3, $\mathcal{T}_\Pi^{k+1}(\emptyset, I) = \mathcal{T}_\Pi(\mathcal{T}_\Pi^k(\emptyset, I), I) \subseteq \mathcal{T}_\Pi^{k+1}(\emptyset, J) = \mathcal{T}_\Pi(\mathcal{T}_\Pi^k(\emptyset, J), J) \subset J$. Therefore, $\mathcal{T}_\Pi^\alpha(\emptyset, I) \subseteq \mathcal{T}_\Pi^\alpha(\emptyset, J) \subseteq J$. □

We define answer sets by means of the above fixpoint.

Definition 6. Let Π be a logic program with DL expressions relative to a DL knowledge base L, and I a model of Π relative to L. I is an *answer set* of Π relative to L if for every $A \in I$, either $A \in \mathcal{T}_\Pi^\alpha(\emptyset, I)$ or $L \cup \mathcal{T}_\Pi^\alpha(\emptyset, I)|_\Omega \cup \neg I^-|_\Omega \models A$.

The answer set semantics for Π relative to L is then defined by answer sets of Π relative to L. That is, a ground literal l is *credulously* (resp. *skeptically*) true in Π relative to L if $I \models_L l$ for some (resp. every) answer set I of Π relative to L.

It is immediate that when $L = \emptyset$, a model I is an answer set of Π relative to L iff $I = \mathcal{T}_\Pi^\alpha(\emptyset, I)$.

Example 3. Consider Example 1. For $I_1 = \emptyset$, $\mathcal{T}_\Pi^\alpha(\emptyset, I_1) = \emptyset$, so I_1 is an answer set of Π relative to L. For $I_2 = \{A(a)\}$, $\mathcal{T}_\Pi^0(\emptyset, I_2) = \emptyset$ and $\mathcal{T}_\Pi^1(\emptyset, I_2) = \mathcal{T}_\Pi(\emptyset, I_2) = \emptyset$, so $\mathcal{T}_\Pi^\alpha(\emptyset, I_2) = \emptyset$. For $A(a) \in I_2$, $A(a) \notin \mathcal{T}_\Pi^\alpha(\emptyset, I_2)$ and $L \cup \mathcal{T}_\Pi^\alpha(\emptyset, I_2)|_\Omega \cup \neg I_2^-|_\Omega \not\models A(a)$. Thus I_2 is not an answer set of Π relative to L.

Consider Example 2. For $I_1 = \{A(a)\}$, $\mathcal{T}_{\Pi}^0(\emptyset, I_1) = \emptyset$ and $\mathcal{T}_{\Pi}^1(\emptyset, I_1) = \mathcal{T}_{\Pi}(\emptyset, I_1) = \emptyset$, so $\mathcal{T}_{\Pi}^\alpha(\emptyset, I_1) = \emptyset$. For $A(a) \in I_1$, $A(a) \notin \mathcal{T}_{\Pi}^\alpha(\emptyset, I_1)$, but $L \cup \mathcal{T}_{\Pi}^\alpha(\emptyset, I_1)|_\Omega \cup \neg I_1^-|_\Omega \models A(a)$, so I_1 is an answer set of Π relative to L. It is easy to verify that $I_2 = \{A(a), C(a)\}$ is not an answer set of Π relative to L. □

The following result shows that answer sets must be minimal models.

Theorem 5. *Let Π be a logic program with DL expressions relative to a DL knowledge base L. If I is an answer set of Π relative to L, then I is a minimal model of Π relative to L.*

Proof: Assume, on the contrary, that I is not a minimal model relative to L. Let $J \subset I$ be a minimal model relative to L. By Theorem 4, $\mathcal{T}_{\Pi}^\alpha(\emptyset, I) \subseteq \mathcal{T}_{\Pi}^\alpha(\emptyset, J) \subseteq J$. Let $S = I \setminus J$. Note that S is not empty and for any $A \in S$, $A \notin \mathcal{T}_{\Pi}^\alpha(\emptyset, I)$. Since I is an answer set of Π, for any $A \in S$, $L \cup \mathcal{T}_{\Pi}^\alpha(\emptyset, I)|_\Omega \cup \neg I^-|_\Omega \models A$. Since $J^- \supset I^-$, $L \cup J|_\Omega \cup \neg J^-|_\Omega \models A$. Since every $A \in S$ is in J^-, $L \cup J|_\Omega \cup \neg J^-|_\Omega \models \neg A$. This means that $L \cup J|_\Omega \cup \neg J^-|_\Omega$ is not consistent, and thus J is not a model of Π relative to L. We then have a contradiction. Therefore, I is a minimal model of Π relative to L. □

The next result shows that answer sets are exactly well-supported models.

Theorem 6. *Let Π be a logic program with DL expressions relative to a DL knowledge base L, and I a model of Π relative to L. I is an answer set of Π relative to L iff I is a well-supported model of Π relative to L.*

Proof: Assume that I is an answer set relative to L. We can construct a level mapping $f : I \to N$, where N is an integer, as follows: For each $A \in I$, we assign $f(A) = i$, where $i \geq 0$ is the smallest number such that either $L \cup \mathcal{T}_{\Pi}^i(\emptyset, I)|_\Omega \cup \neg I^-|_\Omega \models A$ or there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(\mathcal{T}_{\Pi}^i(\emptyset, I), I) \models_L body(r)$.

We then define a strict well-founded partial order $\prec$ on I such that for any $A, B \in I$, $B \prec A$ iff $f(B) < f(A)$. For each $A \in I$ with $f(A) = i$, we always have $E = \mathcal{T}_{\Pi}^i(\emptyset, I) \subset I$, where for every $B \in E$, $B \prec A$, such that either $L \cup E|_\Omega \cup \neg I^-|_\Omega \models A$ or there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(E, I) \models_L body(r)$. By Definition 4, I is a well-supported model relative to L.

Conversely, assume that I is a well-supported model relative to L. Then, there exists a strict well-founded partial order $\prec$ on I such that for any $A \in I$, there exists $E \subset I$, where for every $B \in E$, $B \prec A$, such that either $L \cup E|_\Omega \cup \neg I^-|_\Omega \models A$ or there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(E, I) \models_L body(r)$. Such a partial order establishes a level mapping $f : I \to N$ so that for any $A \in I$, A can be derived from some $E \subset I$ at lower levels in the way as above. Next, we show that for every $A \in I$ at level $i \geq 0$ we have $E = \mathcal{T}_{\Pi}^i(\emptyset, I)$ satisfying the above conditions.

First, each $A \in I$ at the lowest level ($i = 0$) does not depend on any other atom $B \in I$, i.e., there is no $B \in I$ with $B \prec A$. By the assumption that there exists $E \subset I$, where for every $B \in E$, $B \prec A$, such that either $L \cup E|_\Omega \cup \neg I^-|_\Omega$

$\models A$ or there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(E, I) \models_L body(r)$, we have $E = \emptyset$. Therefore, for each $A \in I$ at level 0, we have $E = \mathcal{T}_\Pi^0(\emptyset, I)$ which satisfies the above conditions.

As the induction hypothesis, assume that for any $i \leq n$ and any $A \in I$ at level i, we have $E = \mathcal{T}_\Pi^i(\emptyset, I)$ such that either $L \cup E|_\Omega \cup \neg I^-|_\Omega \models A$ or there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(E, I) \models_L body(r)$. Then, by Theorem 3, for each $A \in I$ at level $i \leq n$, we have $E = \mathcal{T}_\Pi^n(\emptyset, I)$ which satisfies the above conditions.

Consider $A \in I$ at level $n+1$. Then, there exists $E \subset I$, where for every $B \in E$, $B \prec A$, such that either (1) $L \cup E|_\Omega \cup \neg I^-|_\Omega \models A$, or (2) there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(E, I) \models_L body(r)$. Next, we show that when using $\mathcal{T}_\Pi^{n+1}(\emptyset, I)$ to replace E, the conditions (1) and (2) still hold for every $A \in I$ at level $n+1$.

For every $B \in E$, since B is at a level below $n+1$, by the induction hypothesis, either (a) $L \cup \mathcal{T}_\Pi^n(\emptyset, I)|_\Omega \cup \neg I^-|_\Omega \models B$, or (b) there is a rule $B \leftarrow body(r)$ in $ground(\Pi)$ such that $(\mathcal{T}_\Pi^n(\emptyset, I), I) \models_L body(r)$. For case (a), we distinguish between two cases: (i) $B \in \mathcal{T}_\Pi^n(\emptyset, I)$. In this case, if we replace B in E by $\mathcal{T}_\Pi^{n+1}(\emptyset, I)$, the conditions (1) and (2) above still hold for each $A \in I$ at level $n+1$. (ii) $B \notin \mathcal{T}_\Pi^n(\emptyset, I)$. Then, for no $i \leq n$, $B \in \mathcal{T}_\Pi^i(\emptyset, I)$; thus B is an atomic DL expression. In this case, if we replace B in E by $\mathcal{T}_\Pi^{n+1}(\emptyset, I)$, the condition (1) above still holds, since $L \cup \mathcal{T}_\Pi^n(\emptyset, I)|_\Omega \cup \neg I^-|_\Omega \models B$. Consider the condition (2). $(E, I) \models_L body(r)$ means that for every F with $E \subseteq F \subseteq I$, $F \models_L body(r)$. Let us replace B in E by $\mathcal{T}_\Pi^{n+1}(\emptyset, I)$. Since B is a ground atomic DL expression, by Proposition 1, $(E \setminus \{B\} \cup \mathcal{T}_\Pi^{n+1}(\emptyset, I), I) \models_L B$, because $L \cup (E \setminus \{B\} \cup \mathcal{T}_\Pi^{n+1}(\emptyset, I))|_\Omega \cup \neg I^-|_\Omega \models B$. This shows that for any $body(r)$, $(E \setminus \{B\} \cup \mathcal{T}_\Pi^{n+1}(\emptyset, I), I) \models_L body(r)$ iff $(E \cup \mathcal{T}_\Pi^{n+1}(\emptyset, I), I) \models_L body(r)$. Then, when $(E, I) \models_L body(r)$, $(E \cup \mathcal{T}_\Pi^{n+1}(\emptyset, I), I) \models_L body(r)$ and thus $(E \setminus \{B\} \cup \mathcal{T}_\Pi^{n+1}(\emptyset, I), I) \models_L body(r)$. This shows that after replacing B in E by $\mathcal{T}_\Pi^{n+1}(\emptyset, I)$, the condition (2) above still holds, Therefore, if we replace B in E by $\mathcal{T}_\Pi^{n+1}(\emptyset, I)$, the conditions (1) and (2) above still hold. For case (b), $B \in \mathcal{T}_\Pi^{n+1}(\emptyset, I)$. So if we replace B in E by $\mathcal{T}_\Pi^{n+1}(\emptyset, I)$, the conditions (1) and (2) above still hold.

As a result, if we replace all $B \in E$ by $\mathcal{T}_\Pi^{n+1}(\emptyset, I)$, the conditions (1) and (2) above still hold. Therefore, for every $A \in I$ at level $n+1$. we have $E = \mathcal{T}_\Pi^{n+1}(\emptyset, I)$ such that either $L \cup E|_\Omega \cup \neg I^-|_\Omega \models A$ or there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $(E, I) \models_L body(r)$.

Consequently, for every $A \in I$, either $L \cup \mathcal{T}_\Pi^\alpha(\emptyset, I)|_\Omega \cup \neg I^-|_\Omega \models A$ or $A \in \mathcal{T}_\Pi^\alpha(\emptyset, I)$. This shows that I is an answer set of Π relative to L. □

Example 4. Let $L = \emptyset$ and

$$\Pi: \quad A(g). \quad B(g) \leftarrow C(g). \quad C(g) \leftarrow ((A \sqcap \neg C) \sqcup B)(g).$$

Let $\mathbf{P} = \{A, B, C\}$, $\mathbf{C} = \{g\}$ and $\Omega = \{A, B, C\}$. $HB_\Pi = \{A(g), B(g), C(g)\}$ and $ground(\Pi) = \Pi$. Π has only one model relative to L, $I = \{A(g), B(g), C(g)\}$. This model is not an answer set, since it is not a well-supported model of Π relative to L.

Note that we can use a fresh DL concept D to replace the DL expression $(A \sqcap \neg C) \sqcup B$ and add to L an axiom $D \equiv (A \sqcap \neg C) \sqcup B$. This yields

Π' : $A(g)$. $B(g) \leftarrow C(g)$. $C(g) \leftarrow D(g)$.
L' : $D \equiv (A \sqcap \neg C) \sqcup B$.

Using the same $\mathbf{P}$, $\mathbf{C}$ and Ω as above, Π' has the same answer sets relative to L' as Π relative to L.

The following result shows that this answer set semantics is a proper extension to the standard answer set semantics.

Theorem 7. *Let $L = \emptyset$ and Π be a normal logic program without DL expressions. An interpretation I is an answer set of Π relative to L iff I is an answer set of Π under the standard answer set semantics.*

Proof: By Theorem 6, I is an answer set of Π relative to L iff I is a well-supported model of Π relative to L. By Theorem 2, I is a well-supported model of Π relative to L iff I is a well-supported model of Π under Fages' definition. Then as shown in [12], the well-supported models of Π under Fages' definition are exactly the answer sets of Π under the standard answer set semantics. □

3.3 Decidability Property

For a logic program Π with DL expressions relative to a DL knowledge base L, the decidability of computing answer sets of Π relative to L depends on the decidability of satisfiability of L. Since DLs are fragments of first-order logic, the satisfiability of L is undecidable in general cases. However, if L is built from the description logic $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$, its satisfiability is decidable [3, 9].

Let L be a DL knowledge base from $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$. Since HB_Π and $ground(\Pi)$ are finite, it is decidable to determine if an interpretation I is a model of Π relative to L. For any $E \subseteq I$ and any ground atom or DL expression A in $ground(\Pi)$, it is decidable to determine if $(E, I) \models_L A$ (resp. $(E, I) \models_L not\ A$) holds, and thus it is decidable to determine if $(E, I) \models_L body(r)$ holds for each rule r in $ground(\Pi)$. Since $ground(\Pi)$ consists of a finite set of rules, it takes finite time to compute the fixpoint $\mathcal{T}_\Pi^\alpha(\emptyset, I)$. As a result, it is decidable to determine if an interpretation I is an answer set of Π relative to L. Since Π has only a finite set of interpretations, it is decidable to compute all answer sets of Π relative to L.

4 Related Work

Although many approaches to integrating rules and DLs have been proposed in the literature [1–6, 14, 15], to the best of our knowledge DL-programs [2, 3] are the first framework which extends normal logic programs under the answer set semantics to logic programs with arbitrary DL expressions relative to an external DL knowledge base. Four different answer set semantics have been defined for DL-programs. The first one, called *weak* answer set semantics [2, 3], easily incurs circular justifications by self-supporting loops, so a second one, called *strong* answer set semantics, was introduced [2, 3]. Answer sets under the strong answer set

semantics are not minimal models of a DL-program, then a third one, called *FLP-reduct based* answer set semantics, was proposed [16]. This semantics is based on the concept of FLP-reduct from [17]. It turns out, however, that none of the three answer set semantics extends the key well-supportedness condition from normal logic programs to DL-programs, so that their answer sets may incur circular justifications by self-supporting loops. To resolve this problem, a fourth semantics, called *well-supported* answer set semantics, was recently introduced [13], which extends the well-supportedness condition to DL-programs. In this paper, we extend the well-supportedness condition to our framework of logic programs with DL expressions in a way similar to that in [13]. However, DL-programs and our framework differ in fundamental ways. First, in a DL-program, Π and L share no predicate symbols, so DL expressions $Q(\mathbf{t})$ must occur together with predicate mapping operations $S_i op_i P_i$. Note that in DL-programs one cannot use only dl-atoms of the form $DL[Q](\mathbf{t})$ to express a DL-expression $Q(\mathbf{t})$ because that would cut the knowledge flow from Π to L. Second, in a DL-program, DL expressions (dl-atoms) are not allowed to occur in a rule head, so no conclusions about L can be inferred from Π. As we said in Introduction, it is necessary to allow DL expressions to occur in rule heads because DL knowledge bases (ontologies) define only general terminological knowledge, while additional constraints and exceptions over some DL concepts/roles must be defined by rules.

$\mathcal{DL}$+log [6] is closely related to but differs significantly from our framework. Syntactically, it divides predicate symbols into *Datalog* predicates and *DL* predicates. The former type can only occur in Π, while the latter is not allowed to occur (as DL expressions) behind the negation operator *not*. Semantically, it considers first-order interpretations, instead of Herbrand interpretations, and defines a semantics with a class of first-order models, called *NM-models*. In an NM-model, DL predicates can take arbitrary truth values (as in first-order logic), but Datalog predicates take truth values that must be minimal (as in logic programming) when the truth values of all DL predicates are fixed. For instance, consider a $\mathcal{DL}$+log program with $\Pi = \{B(g) \leftarrow A(g)\}$ and $L = \{A \sqcup C\}$. B is a Datalog predicate and A, C are DL predicates. Since A, C can take arbitrary truth values, this program has at least three NM-models: $I_1 = \{A(g), B(g)\}$, $I_2 = \{A(g), B(g), C(g)\}$ and $I_3 = \{C(g)\}$. Clearly, such NM-models are not minimal, well-supported models.

Disjunctive DL-programs [4] are closely related to our framework, but differs significantly from ours at least in two aspects: (1) Let Π be a disjunctive DL-program with DL expressions relative to an external DL knowledge base L, where Π is built over a vocabulary $\Phi = (\mathbf{P}, \mathbf{C})$, $\mathbf{P}$ is a finite set of predicate symbols, and $\mathbf{C}$ is a nonempty finite set of constants. Then, all concepts and roles occurring in Π are required to be included in $\mathbf{P}$, so that all such concepts and roles are interpreted with Herbrand interpretations w.r.t. Φ and are minimized under the semantics of disjunctive DL-programs. This requirement seems to be a bit too strong. For instance, in Example 4, since D is a concept of L' introduced to represent $(A \sqcap \neg C) \sqcup B$, D should be interpreted with first-order interpretations of L' and should not be minimized as in disjunctive DL-programs. In contrast,

in our framework concepts and roles can occur in any DL expressions in Π, but they are not necessarily included in $\mathbf{P}$. For instance, in Example 4, D is a concept of L', which occurs in Π' but is not included in $\mathbf{P}$. Therefore, D is not interpreted with Herbrand interpretations w.r.t. Φ. (2) The semantics of disjunctive DL-programs is based on FLP-reduct. Like the FLP-reduct based semantics for DL-programs [16], this FLP-reduct based answer set semantics for disjunctive DL-programs yields answer sets that are minimal models, but are not necessarily well-supported models. See the example in Introduction.

[1] and [5] present approaches to combining a first-order theory L and logic program rules Π by embedding L and Π into first-order autoepistemic logic [18] and the logic of Minimal Knowledge and Negation as Failure (MKNF) [19], respectively. Such embeddings to modal logics satisfy two nice properties: *faithfulness* and *tightness* [5]. Faithfulness means that when one component (L or Π) is empty, the combined knowledge base has the semantics which coincides with the semantics of the other component. Tightness means that L and Π share the same vocabulary. In our framework, we aim to directly extend a normal logic program to accommodate arbitrary DL expressions. This differs significantly from the above embeddings. Like DL-programs [3], we do not seek the property of faithfulness, but we require that when $L = \emptyset$ and Π is a normal logic program with no DL expressions, the semantics of Π should coincide with the standard answer set semantics (Theorem 7). We do not require L and Π share the same vocabulary; that would easily lead to undecidability of the semantics. Instead, like normal logic programs, DL-programs and disjunctive DL-programs, our answer set semantics is defined over a Herbrand base with a finite set of predicate symbols $\mathbf{P}$ and constants $\mathbf{C}$. As discussed earlier, in our framework, when L is from the description logic $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$, it is decidable to compute all answer sets of Π relative to L.

5 Summary

We have introduced a new framework for extending logic programs with DL expressions relative to an external DL knowledge base. In this framework, arbitrary DL expressions are allowed to appear in rule bodies and atomic DL expressions allowed in rule heads. We extend the key condition of well-supportedness for normal logic programs to logic programs with DL expressions and define an answer set semantics which satisfies the extended condition of well-supportedness. As a result, answer sets under the well-supported semantics are free of circular justifications. When a DL knowledge base L is from the description logic $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$, it is decidable to compute all answer sets of a logic program Π with DL expressions relative to L.

As ongoing work, we are considering how to extend the framework to disjunctive logic programs, where the head of a rule is a disjunction of atoms or atomic DL expressions.

References

1. de Bruijn, J., Eiter, T., Tompits, H.: Embedding approaches to combining rules and ontologies into autoepistemic logic. In: KR. (2008) 485–495
2. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. In: KR. (2004) 141–151
3. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. Artificial Intelligence **172**(12-13) (2008) 1495–1539
4. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the semantic web. IEEE Transactions on Knowledge and Data Engineering **22**(11) (2010) 1577–1592
5. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM **57**(5) (2010)
6. Rosati, R.: DL+log: Tight integration of description logics and disjunctive datalog. In: KR-06. (2006) 68–78
7. Eiter, T., Ianni, G., Krennwallner, T., Polleres, A.: Rules and ontologies for the semantic web. In: Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems (RR-08). (2008) 1–53
8. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: Owl 2: The next step for owl. Journal of Web Semantics **6**(4) (2008) 309–322
9. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: the making of a web ontology language. Journal of Web Semantics **1**(1) (2003) 7–26
10. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
11. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the 5th International Conference on Logic Programming (ICLP-88). (1988) 1070–1080
12. Fages, F.: Consistency of clark's completion and existence of stable models. Journal of Methods of Logic in Computer Science **1** (1994) 51–60
13. Shen, Y.D.: Well-supported semantics for description logic programs. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11). (2011)
14. Grosof, B., Horrocks, I., R. Volz, S.D.: Description logic programs: Combining logic programs with description logics. In: WWW-2003. (2003) 48–57
15. Horrocks, I., Patel-Schneider, P., Boley, H., S. Tabet, B.G., Dean, M.: Swrl: A semantic web rule language combining OWL and RuleML. In: W3C Member Submission, Available at http://www.w3.org/Submission/SWRL (2004)
16. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: IJCAI. (2005) 90–96
17. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: Semantics and complexity. In: Logics in Artificial Intelligence: European Workshop (JELIA-04). (2004) 200–212
18. Moore, R.C.: Semantical considerations on nonmonotonic logic. Artificial Intelligence **25**(1) (1985) 75–94
19. Lifschitz, V.: Nonmonotonic databases and epistemic queries. In: IJCAI. (1991) 381–386