Combining Answer Set Programming with Description Logics for the Semantic Web

Thomas Eiter[1] **Thomas Lukasiewicz**[2,1] **Roman Schindlauer**[1] **Hans Tompits**[1]

[1] Institut für Informationssysteme 184/3, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter, roman, tompits}@kr.tuwien.ac.at

[2] Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Rome, Italy
lukasiewicz@dis.uniroma1.it

Abstract

Towards the integration of rules and ontologies in the Semantic Web, we propose a combination of logic programming under the answer set semantics with the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which underly the Web ontology languages OWL Lite and OWL DL, respectively. This combination allows for building rules on top of ontologies but also, to a limited extent, building ontologies on top of rules. We introduce *description logic programs* (*dl-programs*), which consist of a description logic knowledge base L and a finite set of *description logic rules* (*dl-rules*) P. Such rules are similar to usual rules in logic programs with negation as failure, but may also contain *queries to* L, possibly default negated, in their bodies. We define Herbrand models for dl-programs, and show that satisfiable positive dl-programs have a unique least Herbrand model. More generally, consistent stratified dl-programs can be associated with a unique minimal Herbrand model that is characterized through iterative least Herbrand models. We then generalize the (unique) minimal Herbrand model semantics for positive and stratified dl-programs to a *strong answer set semantics* for all dl-programs, which is based on a reduction to the least model semantics of positive dl-programs. We also define a *weak answer set semantics* based on a reduction to the answer sets of ordinary logic programs. Strong answer sets are weak answer sets, and both properly generalize answer sets of ordinary normal logic programs. We then give fixpoint characterizations for the (unique) minimal Herbrand model semantics of positive and stratified dl-programs, and show how to compute these models by finite fixpoint iterations. Furthermore, we give a precise picture of the complexity of deciding strong and weak answer set existence for a dl-program.

Introduction

The *Semantic Web* initiative (Berners-Lee 1999; Berners-Lee, Hendler, & Lassila 2001; Fensel *et al.* 2002) is an extension of the current World Wide Web by standards and technologies that help machines to understand the information on the Web so that they can support richer discovery, data integration, navigation, and automation of tasks. The main ideas behind are to add a machine-readable meaning to Web pages, to use ontologies for a precise definition of shared terms in Web resources, to make use of KR technology for automated reasoning from Web resources, and to apply cooperative agent technology for processing the information of the Web. The Semantic Web is conceived in hierarchical layers, where the Ontology layer in the form of the *OWL Web Ontology Language* (W3C 2004; Horrocks, Patel-Schneider, & van Harmelen 2003) is currently the highest layer of sufficient maturity.

OWL has three increasingly expressive sublanguages, namely *OWL Lite*, *OWL DL*, and *OWL Full*, where OWL DL basically corresponds to DAML+OIL (Horrocks 2002a; 2002b), which merges DAML (Hendler & McGuiness 2000) and OIL (Fensel *et al.* 2001). OWL Lite and OWL DL are essentially very expressive description logics with an RDF syntax (Horrocks, Patel-Schneider, & van Harmelen 2003). As shown by Horrocks & Patel-Schneider (2003b), ontology entailment in OWL Lite and OWL DL reduces to knowledge base (un)satisfiability in the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, respectively, where the latter is closely related to $\mathcal{SHOQ}(\mathbf{D})$ (Horrocks & Sattler 2001).

On top of the Ontology layer, the Rules, Logic, and Proof layers of the Semantic Web will be developed next, which should offer sophisticated representation and reasoning capabilities. A first effort in this direction is *RuleML* (Rule Markup Language) (Boley, Tabet, & Wagner 2001), fostering an XML-based markup language for rules and rule-based systems, while the OWL Rules Language (Horrocks & Patel-Schneider 2003a) is a first proposal for extending OWL by Horn clause rules.

A key requirement of the layered architecture of the Semantic Web is to integrate the Rule and the Ontology layer. In particular, it is crucial to allow for building rules on top of ontologies, that is, for rule-based systems that use vocabulary specified in ontology knowledge bases. Another type of combination is to build ontologies on top of rules, which means that ontological definitions are supplemented by rules or imported from rules.

In this paper, we propose, towards the integration of rules and ontologies in the Semantic Web, a combination of logic programming under the answer set semantics with description logics, focusing here on $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$. This combination allows for building rules on top of ontologies but also, to some extent, building ontologies on top of rules. The main innovations and contributions of this paper can be summarized as follows:

(1) We introduce *description logic programs* (*dl-programs*), which consist of a knowledge base L in a description logic and a finite set of description logic rules (*dl-rules*) P. Such rules are similar to usual rules in logic programs with negation as failure, but may also contain *queries to* L, possibly default negated, in their bodies. As an important feature, such queries also allow for specifying an input from P, and thus for a *flow of information from* P *to* L, besides the flow of information from L to P, given by any query to L. For example, concepts and roles in L may be enhanced by facts generated from dl-rules, possibly involving heuristic knowledge and other concepts and roles from L.

(2) The queries to L are treated, fostering an encapsulation view, in a way such that logic programming and description logic inference are technically separated; mainly interfacing details need to be known. Compared to other similar work, this increases flexibility and is also amenable to privacy aspects for L and P. Moreover, the nondeterminism inherent in answer sets is retained, supporting brave reasoning and the answer set programming paradigm in which solutions of problems are encoded in answer sets of a logic program.

(3) We define Herbrand models for dl-programs, and show that satisfiable positive dl-programs, in which default negation does not occur and all queries to L are monotonic, have a unique least Herbrand model. Furthermore, we show that more general stratified dl-programs can be associated, if consistent, with a unique minimal Herbrand model that is characterized through iterative least Herbrand models.

(4) We define *strong answer sets* for all dl-programs, based on a reduction to the least model semantics of positive dl-programs. For positive and stratified dl-programs, the strong answer set semantics coincides with the (unique) minimal Herbrand model semantics associated. We also consider *weak answer sets* based on a reduction to the answer sets of ordinary logic programs. Strong answer sets are weak answer sets, and both properly generalize answer sets of ordinary normal logic programs.

(5) We give fixpoint characterizations for the least model of a positive dl-program and the canonical minimal model of a stratified dl-program, and show how to compute these models by finite fixpoint iterations.

(6) Finally, we give a precise picture of the complexity of deciding strong and weak answer set existence for a dl-program KB. From this, the complexity of brave and cautious reasoning is easily derived. We consider the general case, as well as the restrictions where KB is (a) positive, (b) stratified and has only monotonic queries, and (c) stratified. We consider $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, but most of our results can be easily transferred to other description logics having the same complexity (EXP resp. NEXP).

Previous work on combining logic programs and description logics can be roughly divided into (i) hybrid approaches, which use description logics to specify structural constraints in the bodies of logic program rules, and (ii) approaches that reduce description logic inference to logic programming. The basic idea behind (i) is to combine the semantic and computational strengths of the two systems, while the main rationale of (ii) is to use powerful logic programming technology for inference in description logics. However, both kinds of approaches significantly differ from our work, as we discuss in more detail in the section on related work later on.

Note that proofs of all results are in (Eiter *et al.* 2003).

Preliminaries

In this section, we recall normal programs (over classical literals) under the answer set semantics, and the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$.

Normal Programs under the Answer Set Semantics

Syntax. Let Φ be a first-order vocabulary with nonempty finite sets of constant and predicate symbols, but no function symbols. Let $\mathcal{X}$ be a set of variables. A *term* is any variable from $\mathcal{X}$ or constant symbol from Φ. An *atom* is of form $p(t_1, \ldots, t_n)$, where p is a predicate symbol of arity $n \geq 0$ from Φ, and $t_1, \ldots, t_n$ are terms. A *classical literal* (or *literal*) l is an atom p or a negated atom $\neg p$. A *negation as failure literal* (or *NAF-literal*) is a literal l or a default-negated literal $not\, l$. A *normal rule* (or *rule*) r is of form

$$a \leftarrow b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m\,,\ m \geq k \geq 0\,, \quad (1)$$

where $a, b_1, \ldots, b_m$ are classical literals. We refer to the literal a as the *head* of r, denoted by $H(r)$, while the conjunction $b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m$ is called the *body* of r; its *positive* (resp., *negative*) part is $b_1, \ldots, b_k$ (resp., $not\, b_{k+1}, \ldots, not\, b_m$). We denote by $B(r)$ the set of body literals $B^+(r) \cup B^-(r)$, where $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. A *normal program* (or *program*) P is a finite set of rules; P is *positive* iff it is "not"-free.

Semantics. The *Herbrand base* of a program P, denoted HB_P, is the set of all ground (classical) literals with predicate and constant symbols appearing in P (if no such constant symbol exists, with an arbitrary constant symbol c from Φ). The notions of *ground terms*, *atoms*, *literals*, etc., are defined as usual. We denote by $ground(P)$ the grounding of P (with respect to HB_P).

A set of literals $X \subseteq HB_P$ is *consistent* iff $\{p, \neg p\} \not\subseteq X$ for every atom $p \in HB_P$. An *interpretation* I relative to P is a consistent subset of HB_P. A *model* of a positive program P is an interpretation $I \subseteq HB_P$ such that $B(r) \subseteq I$ implies $H(r) \in I$, for every $r \in ground(P)$. An *answer set* of a positive program P is the least model of P w.r.t. set inclusion.

The *Gelfond-Lifschitz transform* of a program P relative to an interpretation $I \subseteq HB_P$, denoted P^I, is the positive program obtained from $ground(P)$ by (i) deleting every rule r with $B^-(r) \cap I \neq \emptyset$, and (ii) deleting the negative body from every remaining rule. An *answer set* of a program P is an interpretation $I \subseteq HB_P$ that is an answer set of P^I.

$\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$

Syntax. We first describe the syntax of $\mathcal{SHOIN}(\mathbf{D})$. We assume a set $\mathbf{D}$ of *elementary datatypes*. Every $d \in \mathbf{D}$ has a set of *data values*, called the *domain* of d, denoted $\mathrm{dom}(d)$.

We use $\mathrm{dom}(\mathbf{D})$ to denote $\bigcup_{d \in \mathbf{D}} \mathrm{dom}(d)$. A *datatype* is either an element of $\mathbf{D}$ or a subset of $\mathrm{dom}(\mathbf{D})$ (called *datatype oneOf*). Let $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ be nonempty finite and pairwise disjoint sets of *atomic concepts*, *abstract roles*, *datatype roles*, and *individuals*, respectively. We use $\mathbf{R}_A^-$ to denote the set of all inverses R^- of abstract roles $R \in \mathbf{R}_A$.

A *role* is an element of $\mathbf{R}_A \cup \mathbf{R}_A^- \cup \mathbf{R}_D$. *Concepts* are inductively defined as follows. Every $C \in \mathbf{A}$ is a concept, and if $o_1, o_2, \ldots \in \mathbf{I}$, then $\{o_1, o_2, \ldots\}$ is a concept (called *oneOf*). If C and D are concepts and if $R \in \mathbf{R}_A \cup \mathbf{R}_A^-$, then $(C \sqcap D)$, $(C \sqcup D)$, and $\neg C$ are concepts (called *conjunction*, *disjunction*, and *negation*, respectively), as well as $\exists R.C$, $\forall R.C$, $\geq nR$, and $\leq nR$ (called *exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geq 0$. If $d \in \mathbf{D}$ and $U \in \mathbf{R}_D$, then $\exists U.d$, $\forall U.d$, $\geq nU$, and $\leq nU$ are concepts (called *datatype exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geq 0$. We write $\top$ and $\bot$ to abbreviate $C \sqcup \neg C$ and $C \sqcap \neg C$, respectively, and we eliminate parentheses as usual.

An *axiom* is an expression of one of the following forms: (1) $C \sqsubseteq D$, where C and D are concepts (*concept inclusion*); (2) $R \sqsubseteq S$, where either $R, S \in \mathbf{R}_A$ or $R, S \in \mathbf{R}_D$ (*role inclusion*); (3) $\mathrm{Trans}(R)$, where $R \in \mathbf{R}_A$ (*transitivity*); (4) $C(a)$, where C is a concept and $a \in \mathbf{I}$ (*concept membership*); (5) $R(a, b)$ (resp., $U(a, v)$), where $R \in \mathbf{R}_A$ (resp., $U \in \mathbf{R}_D$) and $a, b \in \mathbf{I}$ (resp., $a \in \mathbf{I}$ and $v \in \mathrm{dom}(\mathbf{D})$) (*role membership axiom*); and (6) $a = b$ (resp., $a \neq b$), where $a, b \in \mathbf{I}$ (*equality* (resp., *inequality*)). A *knowledge base* L is a finite set of axioms. (For decidability, number restrictions in L are restricted to simple abstract roles (Horrocks *et al.* 1999)).

The syntax of $\mathcal{SHIF}(\mathbf{D})$ is as the above syntax of $\mathcal{SHOIN}(\mathbf{D})$, but without the oneOf constructor and with the atleast and atmost constructors limited to 0 and 1.

Semantics. An *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ with respect to $\mathbf{D}$ consists of a nonempty (*abstract*) *domain* Δ disjoint from $\mathrm{dom}(\mathbf{D})$, and a mapping $\cdot^{\mathcal{I}}$ that assigns to each $C \in \mathbf{A}$ a subset of Δ, to each $o \in \mathbf{I}$ an element of Δ, to each $r \in \mathbf{R}_A$ a subset of $\Delta \times \Delta$, and to each $U \in \mathbf{R}_D$ a subset of $\Delta \times \mathrm{dom}(\mathbf{D})$. The mapping $\cdot^{\mathcal{I}}$ is extended to all concepts and roles as usual (Eiter *et al.* 2003).

The *satisfaction* of a description logic axiom F in an interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, denoted $\mathcal{I} \models F$, is defined as follows: (1) $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; (2) $\mathcal{I} \models R \sqsubseteq S$ iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$; (3) $\mathcal{I} \models \mathrm{Trans}(R)$ iff $R^{\mathcal{I}}$ is transitive; (4) $\mathcal{I} \models C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$; (5) $\mathcal{I} \models R(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$; (6) $\mathcal{I} \models U(a, v)$ iff $(a^{\mathcal{I}}, v) \in U^{\mathcal{I}}$; (7) $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$; and (8) $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. The interpretation $\mathcal{I}$ *satisfies* the axiom F, or $\mathcal{I}$ is a *model* of F, iff $\mathcal{I} \models F$. $\mathcal{I}$ *satisfies* a knowledge base L, or $\mathcal{I}$ is a *model* of L, denoted $\mathcal{I} \models L$, iff $\mathcal{I} \models F$ for all $F \in L$. We say that L is *satisfiable* (resp., *unsatisfiable*) iff L has a (resp., no) model. An axiom F is a *logical consequence* of L, denoted $L \models F$, iff every model of L satisfies F. A negated axiom $\neg F$ is a *logical consequence* of L, denoted $L \models \neg F$, iff every model of L does not satisfy F.

Description Logic Programs

In this section, we introduce *description logic programs* (or simply *dl-programs*), which are a novel combination of normal programs and description logic knowledge bases.

Syntax

Informally, a dl-program consists of a description logic knowledge base L and a generalized normal program P, which may contain queries to L. Roughly, in such a query, it is asked whether a certain description logic axiom or its negation logically follows from L or not.

A *dl-query* $Q(\mathbf{t})$ is either

(a) a concept inclusion axiom F or its negation $\neg F$; or

(b) of the forms $C(t)$ or $\neg C(t)$, where C is a concept and t is a term; or

(c) of the forms $R(t_1, t_2)$ or $\neg R(t_1, t_2)$, where R is a role and t_1, t_2 are terms.

A *dl-atom* has the form

$$DL[S_1\, op_1 p_1, \ldots, S_m\, op_m\, p_m; Q](\mathbf{t})\,, \qquad m \geq 0, \quad (2)$$

where each S_i is either a concept or a role, $op_i \in \{\uplus, \cup\!\!\!\!-, \cap\!\!\!\!-\}$, p_i is a unary resp. binary predicate symbol, and $Q(\mathbf{t})$ is a dl-query. We call $p_1, \ldots, p_m$ its *input predicate symbols*. Intuitively, $op_i = \uplus$ (resp., $op_i = \cup\!\!\!\!-$) increases S_i (resp., $\neg S_i$) by the extension of p_i, while $op_i = \cap\!\!\!\!-$ constrains S_i to p_i. A *dl-rule* r has the form (1), where any literal $b_1, \ldots, b_m \in B(r)$ may be a dl-atom. We denote by $\widetilde{B}^+(r)$ (resp., $\widetilde{B}^-(r)$) the set of all dl-atoms in $B^+(r)$ (resp., $B^-(r)$). A *dl-program* $KB = (L, P)$ consists of a description logic knowledge base L and a finite set of dl-rules P.

We use the following example to illustrate our main ideas.

Example 1 (**Reviewer Selection**) Suppose we want to assign reviewers to papers, based on certain information about the papers and the available persons, using a description logic knowledge base L_S (partially given in the appendix), which contains knowledge about scientific publications.

We assume not to be aware of the entire structure and contents of L_S, but of the following aspects. L_S classifies papers into research areas, depending on keyword information. The research areas are stored in a concept *Area*. The roles *keyword* and *inArea* associate with each paper its relevant keywords and the areas it is classified into (obtained, e.g., by reification of the classes). Furthermore, a role *expert* relates persons to their areas of expertise, and a concept *Referee* contains all referees. Finally, a role *hasMember* associates with a cluster of similar keywords all its members.

Consider then the dl-program $KB_S = (L_S, P_S)$, where P_S contains in particular the following dl-rules:

(1) $\mathit{paper}(p_1)$; $\mathit{kw}(p_1, \mathit{Semantic\_Web})$;
(2) $\mathit{paper}(p_2)$; $\mathit{kw}(p_2, \mathit{Bioinformatics})$; $\mathit{kw}(p_2, \mathit{Answer\_Set\_Programming})$;
(3) $\mathit{kw}(P, K_2) \leftarrow \mathit{kw}(P, K_1),\ DL[\mathit{hasMember}](S, K_1),\ DL[\mathit{hasMember}](S, K_2)$;
(4) $\mathit{paperArea}(P, A) \leftarrow DL[\mathit{keyword} \uplus \mathit{kw}; \mathit{inArea}](P, A)$;
(5) $\mathit{cand}(X, P) \leftarrow \mathit{paperArea}(P, A),\ DL[\mathit{Referee}](X),\ DL[\mathit{expert}](X, A)$;
(6) $\mathit{assign}(X, P) \leftarrow \mathit{cand}(X, P),\ \mathit{not}\ \neg \mathit{assign}(X, P)$;
(7) $\neg \mathit{assign}(Y, P) \leftarrow \mathit{cand}(Y, P), \mathit{assign}(X, P),\ X \neq Y$;
(8) $a(P) \leftarrow \mathit{assign}(X, P)$;
(9) $\mathit{error}(P) \leftarrow \mathit{paper}(P), \mathit{not}\ a(P)$.

Intuitively, rules (1) and (2) specify the keyword information of two papers, p_1 and p_2, which should be assigned to reviewers. Rule (3) augments, by choice of the designer, the keyword information with similar ones. Rule (4) queries the augmented L_S to retrieve the areas that each paper is classified into, and rule (5) singles out review candidates based on this information from experts among the reviewers according to L_S. Rules (6) and (7) pick one of the candidate reviewers for a paper (multiple reviewers can be selected similarly). Finally, rules (8) and (9) check if each paper is assigned; if not, an error is flagged. Note that, in view of rules (3)–(5), information flows in both directions between the knowledge encoded in L_S and the one encoded in P_S.

To illustrate the use of $\cap\!\!\!-$, a predicate $poss\_Referees$ may be defined in the dl-program, and "$Referee \cap\!\!\!- poss\_Referees$" may be added in the first dl-atom of (5), which thus constrains the set of referees.

The dl-rule below shows in particular how dl-rules can be used to encode certain qualified number restrictions, which are not available in $\mathcal{SHOIN}(\mathbf{D})$. It defines an *expert* as an author of at least three papers of the same area:

$$\begin{aligned} expert(X, A) \leftarrow\ & DL[isAuthorOf](X, P_1), \\ & DL[isAuthorOf](X, P_2), \\ & DL[isAuthorOf](X, P_3), \\ & DL[inArea](P_1, A), \\ & DL[inArea](P_2, A), \\ & DL[inArea](P_3, A), \\ & P_1 \neq P_2,\ P_2 \neq P_3,\ P_3 \neq P_1. \end{aligned}$$

Semantics

We first define Herbrand interpretations and the truth of dl-programs in Herbrand interpretations. In the sequel, let $KB = (L, P)$ be a dl-program.

The *Herbrand base* of P, denoted HB_P, is the set of all ground literals with a standard predicate symbol that occurs in P and constant symbols in Φ. An *interpretation I relative to P* is a consistent subset of HB_P. We say I is a *model* of $l \in HB_P$ under L, denoted $I \models_L l$, iff $l \in I$, and of a ground dl-atom $a = DL[S_1\, op_1\ p_1, \ldots, S_m\, op_m p_m; Q](\mathbf{c})$ under L, denoted $I \models_L a$, iff $L \cup \bigcup_{i=1}^{m} A_i(I) \models Q(\mathbf{c})$, where

- $A_i(I) = \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$, for $op_i = \uplus$;
- $A_i(I) = \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$, for $op_i = \cup\!\!\!-$;
- $A_i(I) = \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I$ does not hold$\}$, for $op_i = \cap\!\!\!-$.

We say that I is a *model* of a ground dl-rule r iff $I \models_L H(r)$ whenever $I \models_L l$ for all $l \in B^+(r)$ and $I \not\models_L l$ for all $l \in B^-(r)$, and of a dl-program $KB = (L, P)$, denoted $I \models KB$, iff $I \models_L r$ for all $r \in ground(P)$. We say KB is *satisfiable* (resp., *unsatisfiable*) iff it has some (resp., no) model.

Least Model Semantics of Positive dl-Programs. We now define positive dl-programs, which are "not"-free dl-programs that involve only monotonic dl-atoms. Like ordinary positive programs, every positive dl-program that is satisfiable has a unique least model, which naturally characterizes its semantics.

A ground dl-atom a is *monotonic* relative to $KB = (L, P)$ iff $I \subseteq I' \subseteq HB_P$ implies that if $I \models_L a$ then $I' \models_L a$. A dl-program $KB = (L, P)$ is *positive* iff (i) P is "not"-free, and (ii) every ground dl-atom that occurs in $ground(P)$ is monotonic relative to KB.

Observe that a dl-atom containing $\cap\!\!\!-$ may fail to be monotonic, since an increasing set of $p_i(\mathbf{e})$ in P results in a reduction of $\neg S_i(\mathbf{e})$ in L, whereas dl-atoms containing $\uplus$ and $\cup\!\!\!-$ only are always monotonic.

For ordinary positive programs P, it is well-known that the intersection of two models of P is also a model of P. The following theorem shows that a similar result holds for positive dl-programs KB.

Theorem 1 *Let $KB = (L, P)$ be a positive dl-program. If the interpretations $I_1, I_2 \subseteq HB_P$ are models of KB, then $I_1 \cap I_2$ is also a model of KB.*

Proof. Suppose that $I_1, I_2 \subseteq HB_P$ are models of KB. We show that $I = I_1 \cap I_2$ is also a model of KB, i.e., $I \models_L r$ for all $r \in ground(P)$. Consider any $r \in ground(P)$, and assume that $I \models_L l$ for all $l \in B^+(r) = B(r)$. That is, $I \models_L l$ for all classical literals $l \in B(r)$ and $I \models_L a$ for all dl-atoms $a \in B(r)$. Hence, $I_i \models_L l$ for all classical literals $l \in B(r)$, for every $i \in \{1, 2\}$. Moreover, $I_i \models_L a$ for all dl-atoms $a \in B(r)$, for every $i \in \{1, 2\}$, since every dl-atom in $ground(P)$ is monotonic relative to KB. Since I_1 and I_2 are models of KB, it follows that $I_i \models_L H(r)$, for every $i \in \{1, 2\}$, and thus $I \models_L H(r)$. This shows that $I \models_L r$. Hence, I is a model of KB. $\Box$

As an immediate corollary of this result, every satisfiable positive dl-program KB has a unique least model, denoted M_{KB}, which is contained in every model of KB.

Corollary 2 *Let $KB = (L, P)$ be a positive dl-program. If KB is satisfiable, then there exists a unique model $I \subseteq HB_P$ of KB such that $I \subseteq J$ for all models $J \subseteq HB_P$ of KB.*

Example 2 Consider the dl-program comprising rules (1)–(5) from Example 1. Clearly, this program is "not"-free. Moreover, since the dl-atoms do not contain occurrences of $\cap\!\!\!-$, they are all monotonic. Hence, the dl-program is positive. As well, its unique least model contains all review candidates for the given papers p_1 and p_2.

Iterative Least Model Semantics of Stratified dl-Programs. We next define stratified dl-programs, which are intuitively composed of hierarchic layers of positive dl-programs linked via default negation. Like for ordinary stratified programs, a canonical minimal model can be singled out by a number of iterative least models, which naturally describes the semantics, provided some model exists. We can accommodate this with possibly non-monotonic dl-atoms by treating them similarly as NAF-literals. This is particularly useful, if we do not know a priori whether some dl-atoms are monotonic, and determining this might be costly; recall, however, as noted above, that the absence of $\cap\!\!\!-$ in (2) is a simple syntactic criterion which implies monotonicity of a dl-atom (cf. also Example 2).

For any dl-program $KB = (L, P)$, we denote by DL_P the set of all ground dl-atoms that occur in $ground(P)$. We assume that KB has an associated set $DL_P^+ \subseteq DL_P$ of ground dl-atoms which are known to be monotonic, and we denote by $DL_P^? = DL_P - DL_P^+$ the set of all other dl-atoms. An *input literal* of $a \in DL_P$ is a ground literal with an input predicate of a and constant symbols in Φ.

A *stratification of* $KB = (L, P)$ (*with respect to* DL_P^+) is a mapping $\lambda\colon HB_P \cup DL_P \to \{0, 1, \ldots, k\}$ such that

(i) $\lambda(H(r)) \geq \lambda(l')$ (resp., $\lambda(H(r)) > \lambda(l')$) for each $r \in ground(P)$ and $l' \in B^+(r)$ (resp., $l' \in B^-(r)$), and

(ii) $\lambda(a) \geq \lambda(l)$ (resp., $\lambda(a) > \lambda(l)$) for each input literal l of each $a \in DL_P^+$ (resp., $a \in DL_P^?$),

where $k \geq 0$ is the *length* of λ. For $i \in \{0, \ldots, k\}$, let

$$KB_i = (L, P_i) = (L, \{r \in ground(P) \mid \lambda(H(r)) = i\}),$$

and let HB_{P_i} (resp., $HB^\star_{P_i}$) be the set of all $l \in HB_P$ such that $\lambda(l) = i$ (resp., $\lambda(l) \leq i$).

A dl-program $KB = (L, P)$ is *stratified* iff it has a stratification λ of some length $k \geq 0$. We define its iterative least models $M_i \subseteq HB_P$ with $i \in \{0, \ldots, k\}$ as follows:

(i) M_0 is the least model of KB_0;

(ii) if $i > 0$, then M_i is the least model of KB_i such that $M_i | HB^\star_{P_{i-1}} = M_{i-1} | HB^\star_{P_{i-1}}$.

We say KB is *consistent*, if every M_i with $i \in \{0, \ldots, k\}$ exists, and KB is *inconsistent*, otherwise. If KB is consistent, then M_{KB} denotes M_k. Observe that M_{KB} is well-defined, since it does not depend on a particular λ (cf. Corollary 7).

The following theorem shows that M_{KB} is in fact a minimal model of KB.

Theorem 3 *Let* $KB = (L, P)$ *be a stratified dl-program. Then,* M_{KB} *is a minimal model of* KB.

Proof (sketch). The statement can be proved by induction along a stratification of KB. $\Box$

Example 3 Consider the dl-program $KB = (L, P)$ given by the rules and facts from Example 1, but without rules (6) and (7). This program has a stratification of length 2, with the associated set DL_P^+ comprising all dl-atoms occurring in P. The minimal model M_{KB} contains all review candidates of the given papers, together with error flags for them, because no paper is assigned so far.

Strong Answer Set Semantics of dl-Programs. We now define the *strong answer set semantics* of general dl-programs KB, which is reduced to the least model semantics of positive dl-programs. We use a generalized transformation that removes all NAF-literals and all dl-atoms except for those known to be monotonic. If we ignore this knowledge and remove all dl-atoms, then we arrive at the *weak answer set semantics* for KB, which associates with KB a larger set of models (cf. next subsection).

In the sequel, let $KB = (L, P)$ be a dl-program and let DL_P, DL_P^+, and $DL_P^?$ be as above.

The *strong dl-transform* of P relative to L and an interpretation $I \subseteq HB_P$, denoted sP_L^I, is the set of all dl-rules obtained from $ground(P)$ by

(i) deleting every dl-rule r such that either $I \not\models_L a$ for some $a \in B^+(r) \cap DL_P^?$, or $I \models_L l$ for some $l \in B^-(r)$, and

(ii) deleting from each remaining dl-rule r all literals in $B^-(r) \cup (B^+(r) \cap DL_P^?)$.

Notice that (L, sP_L^I) has only monotonic dl-atoms and no NAF-literals anymore. Thus, (L, sP_L^I) is a positive dl-program, and by Corollary 2, has a least model if satisfiable.

Definition 1 Let $KB = (L, P)$ be a dl-program. A *strong answer set* of KB is an interpretation $I \subseteq HB_P$ such that I is the least model of (L, sP_L^I).

The following result shows that the strong answer set semantics of a dl-program $KB = (L, P)$ without dl-atoms coincides with the ordinary answer set semantics of P.

Theorem 4 *Let* $KB = (L, P)$ *be a dl-program without dl-atoms. Then,* $I \subseteq HB_P$ *is a strong answer set of* KB *iff it is an answer set of the ordinary program* P.

Proof. Let $I \subseteq HB_P$. If KB does not contain any dl-atoms, then $sP_L^I = P^I$. Thus, I is the least model of (L, sP_L^I) iff I is the least model of P^I. Thus, I is a strong answer set of KB iff I is an answer set of P. $\Box$

The next result shows that, as desired, strong answer sets of a dl-program KB are also models, and, moreover, minimal if all dl-atoms are monotonic (and known as such).

Theorem 5 *Let* $KB = (L, P)$ *be a dl-program, and let* M *be a strong answer set of* KB. *Then,* (*a*) M *is a model of* KB, *and* (*b*) M *is a minimal model of* KB *if* $DL_P = DL_P^+$.

Proof. (a) Let I be a strong answer set of KB. To show that I is also a model of KB, we have to show that $I \models_L r$ for all $r \in ground(P)$. Consider any $r \in ground(P)$. Suppose that $I \models_L l$ for all $l \in B^+(r)$ and $I \not\models_L l$ for all $l \in B^-(r)$. Then, the dl-rule r' that is obtained from r by removing all the literals in $B^-(r) \cup (B^+(r) \cap DL_P^?)$ is contained in sP_L^I. Since I is the least model of (L, sP_L^I) and thus in particular a model of (L, sP_L^I), it follows that I is a model of r'. Since $I \models_L l$ for all $l \in B^+(r')$ and $I \not\models_L l$ for all $l \in B^-(r') = \emptyset$, it follows that $I \models_L H(r)$. This shows that $I \models_L r$. Hence, I is a model of KB.

(b) By part (a), every strong answer set I of KB is a model of KB. Assume that every dl-atom in DL_P is monotonic relative to KB. We show that I is a minimal model of KB. Towards a contradiction, suppose the contrary. That is, there exists a model J of KB such that $J \subset I$. Since J is a model of KB, it follows that J is also a model of (L, sP_L^J). As every dl-atom in DL_P is monotonic relative to KB, it then follows that $sP_L^I \subseteq sP_L^J$. Hence, J is also a model of (L, sP_L^I). But this contradicts that I is the least model of (L, sP_L^I). Hence, I is a minimal model of KB. $\Box$

The following theorem shows that positive and stratified dl-programs have at most one strong answer set, which coincides with the canonical minimal model M_{KB}.

Theorem 6 *Let KB be a* (*a*) *positive* (*resp.,* (*b*) *stratified*) *dl-program. If KB is satisfiable* (*resp., consistent*)*, then M_{KB} is the only strong answer set of KB. If KB is not satisfiable* (*resp., consistent*)*, then KB has no strong answer set.*

Proof. (a) If $KB=(L,P)$ is satisfiable, then M_{KB} is defined. A strong answer set of KB is an interpretation $I \subseteq HB_P$ such that I is the least model of (L, sP^I_L). Since KB is a positive dl-program, it follows that sP^I_L coincides with $ground(P)$. Hence, $I \subseteq HB_P$ is a strong answer set of KB iff $I = M_{KB}$. If KB is unsatisfiable, then KB has no model. Thus, by Theorem 5, KB has no strong answer set.

(b) Let λ be a stratification of KB of length $k \geq 0$. Suppose that $I \subseteq HB_P$ is a strong answer set of KB. That is, I is the least model of (L, sP^I_L). Hence,

- $I|HB^\star_{P_0}$ is the least of all models $J \subseteq HB^\star_{P_0}$ of $(L, s{P_0}^I_L)$;
- if $i>0$, then $I|HB^\star_{P_i}$ is the least among all models $J \subseteq HB^\star_{P_i}$ of $(L, s{P_i}^I_L)$ with $J|HB^\star_{P_{i-1}} = I|HB^\star_{P_{i-1}}$.

It thus follows that

- $I|HB^\star_{P_0}$ is the least of all models $J \subseteq HB^\star_{P_0}$ of KB_0; and
- if $i>0$, then $I|HB^\star_{P_i}$ is the least among all models $J \subseteq HB^\star_{P_i}$ of KB_i with $J|HB^\star_{P_{i-1}} = I|HB^\star_{P_{i-1}}$.

Hence, KB is consistent, and $I = M_{KB}$. Since the above line of argumentation also holds in the converse direction, it follows that $I \subseteq HB_P$ is a strong answer set of KB iff KB is consistent and $I = M_{KB}$. □

Since the strong answer sets of a stratified dl-program KB are independent of the stratification λ of KB, we thus obtain that consistency of KB and M_{KB} are independent of λ.

Corollary 7 *Let KB be a stratified dl-program. Then, the notion of consistency of KB and the model M_{KB} do not depend on the stratification of KB.*

Example 4 Consider now the full dl-program from Example 1. This dl-program is not stratified, in view of the rules (6) and (7), which take care of the selection between the different candidates for being reviewers. Every strong answer set that contains no error flags corresponds to an acceptable review assignment scenario.

Weak Answer Set Semantics of dl-Programs. We finally introduce the *weak answer set semantics*, which associates with a dl-program a larger set of models than the strong answer set semantics. It is based on a generalized transformation that removes all dl-atoms and NAF-literals, and reduces to the answer set semantics of ordinary programs.

In the sequel, let $KB=(L,P)$ be a dl-program. The *weak dl-transform* of P relative to L and to an interpretation $I \subseteq HB_P$, denoted wP^I_L, is the ordinary positive program obtained from $ground(P)$ by

(i) deleting all dl-rules r where either $I \not\models_L a$ for some dl-atom $a \in B^+(r)$, or $I \models_L l$ for some $l \in B^-(r)$; and

(ii) deleting from every remaining dl-rule r all the dl-atoms in $B^+(r)$ and all the literals in $B^-(r)$.

Observe that wP^I_L is an ordinary ground positive program, which does neither contain any dl-atoms, nor any NAF-literals. We thus define the weak answer set semantics by reduction to the least model semantics of ordinary ground positive programs as follows.

Definition 2 Let $KB=(L,P)$ be a dl-program. A *weak answer set* of KB is an interpretation $I \subseteq HB_P$ such that I is the least model of the ordinary positive program wP^I_L.

The following result shows that the weak answer set semantics of a dl-program $KB=(L,P)$ without dl-atoms coincides with the ordinary answer set semantics of P.

Theorem 8 *Let $KB=(L,P)$ be a dl-program without dl-atoms. Then, $I \subseteq HB_P$ is a weak answer set of KB iff it is an answer set of the ordinary normal program P.*

Proof. Let $I \subseteq HB_P$. If KB does not contain any dl-atoms, then $wP^I_L = P^I$. Thus, I is the least model of wP^I_L iff I is the least model of P^I. Hence, I is a weak answer set of KB iff I is an answer set of P. □

The next result shows that every weak answer set of a dl-program KB is also a model of KB. Note that differently from strong answer sets, the weak answer sets of KB are in general not minimal models of KB, even if KB has only monotonic dl-atoms.

Theorem 9 *Let KB be a dl-program. Then, every weak answer set of KB is also a model of KB.*

Proof. Let $I \subseteq HB_P$ be a weak answer set of $KB = (L,P)$. To show that I is also a model of KB, we have to show that $I \models_L r$ for all $r \in ground(P)$. Consider any $r \in ground(P)$. Suppose that $I \models_L l$ for all $l \in B^+(r)$ and $I \not\models_L l$ for all $l \in B^-(r)$. Then, the dl-rule r' that is obtained from r by removing all the dl-atoms in $B^+(r)$ and all literals in $B^-(r)$ is in wP^I_L. As I is the least model of wP^I_L, and thus in particular a model of wP^I_L, it follows that $I \models_L r'$. Since $I \models_L l$ for all $l \in B^+(r')$ and $I \not\models_L l$ for all $l \in B^-(r') = \emptyset$, it follows that $I \models_L H(r') = H(r)$. This shows that $I \models_L r$. Thus, I is a model of KB. □

The following result shows that the weak answer set semantics of dl-programs can be expressed in terms of the answer set semantics of ordinary normal programs.

Theorem 10 *Let $KB=(L,P)$ be a dl-program. Let $I \subseteq HB_P$ and let P^I_L be obtained from $ground(P)$ by*

(*i*) *deleting every dl-rule r where either $I \not\models_L a$ for a dl-atom $a \in B^+(r)$, or $I \models_L a$ for a dl-atom $a \in B^-(r)$, and*

(*ii*) *deleting from every remaining dl-rule r every dl-atom in $B^+(r) \cup B^-(r)$.*

Then, I is a weak answer set of KB iff I is an answer set of P_L^I.

Proof. Immediate by observing that $wP_L^I = (P_L^I)^I$. □

Finally, the next result shows that the set of all strong answer sets of a dl-program KB is contained in the set of all weak answer sets of KB. Intuitively, the additional information about the monotonicity of dl-atoms that we use for specifying strong answer sets allows for focusing on a smaller set of models. Thus, the set of all weak answer sets of KB approximates the set of all strong answer sets of KB.

Theorem 11 *Every strong answer set of a dl-program KB is also a weak answer set of KB.*

Proof. Let $I \subseteq HB_P$ be a strong answer set of $KB = (L, P)$. That is, I is the least model of (L, sP_L^I). Hence, I is also a model of wP_L^I. We show that I is in fact the least model of wP_L^I. Towards a contradiction, assume the contrary. That is, there exists a model $J \subset I$ of wP_L^I. Hence, J is also a model of (L, sP_L^I). But this is contradicts the fact that I is the least model of (L, sP_L^I). Hence, I the least model of wP_L^I, and so I is a weak answer set of KB. □

Note that the converse of the above theorem does not hold in general. That is, there exist dl-programs KB which have a weak answer set that is not a strong answer set.

Computation and Complexity

In this section, we give fixpoint characterizations for the strong answer set of satisfiable positive and consistent stratified dl-programs, and we show how to compute it by finite fixpoint iterations. We then draw a precise picture of the complexity of deciding strong and weak answer set existence for a dl-program, respectively.

Fixpoint Semantics

The answer set of an ordinary positive resp. stratified normal logic program P has a well-known fixpoint characterization in terms of an immediate consequence operator T_P, which easily generalizes to dl-programs. This can be exploited for a bottom-up computation of the strong answer set of a positive resp. stratified dl-program.

For a dl-program $KB = (L, P)$, define the operator T_{KB} on the subsets of HB_P as follows. For every $I \subseteq HB_P$, let

$$T_{KB}(I) \;=\; \begin{cases} HB_P, & \text{if } I \text{ is not consistent;} \\ \{H(r) \mid r \in \mathit{ground}(P),\, I \models_L \ell \\ \quad \text{for all } \ell \in B(r)\}\,, & \text{otherwise.} \end{cases}$$

The following lemma shows that, if KB is positive, then T_{KB} is monotonic, which is immediate from the fact that in $\mathit{ground}(P)$, each dl-atom is monotonic relative to KB.

Lemma 12 *For any positive dl-program $KB{=}(L,P)$, T_{KB} is monotonic (i.e., $I \subseteq I' \subseteq HB_P$ implies $T_{KB}(I) \subseteq T_{KB}(I')$).*

Proof. Let $I \subseteq I' \subseteq HB_P$. Consider any $r \in \mathit{ground}(P)$. Then, for every classical literal $l \in B(r)$, it holds that $I \models_L l$ implies $I' \models_L l$. Furthermore, for every dl-atom $a \in B(r)$, it holds that $I \models_L a$ implies $I' \models_L a$, since a is monotonic relative to KB. This shows that $T_{KB}(I) \subseteq T_{KB}(I')$. □

Since every monotonic operator has a least fixpoint, also T_{KB} has one, denoted $\mathit{lfp}(T_{KB})$. Moreover, $\mathit{lfp}(T_{KB})$ can be computed by finite fixpoint iteration (given finiteness of P and the number of constant symbols in Φ).

For every $I \subseteq HB_P$, we define $T_{KB}^i(I) = I$, if $i = 0$, and $T_{KB}^i(I) = T_{KB}(T_{KB}^{i-1}(I))$, if $i > 0$.

Theorem 13 *For every positive dl-program $KB = (L, P)$, it holds that (a) $\mathit{lfp}(T_{KB}) = M_{KB}$, if KB is satisfiable, and (b) $\mathit{lfp}(T_{KB}) = HB_P$, if KB is unsatisfiable. Furthermore,*

$$\mathit{lfp}(T_{KB}) = \textstyle\bigcup_{i=0}^{n} T_{KB}^i(\emptyset) = T_{KB}^n(\emptyset), \textit{ for some } n \geq 0.$$

Example 5 Suppose that P in $KB{=}(L, P)$ consists of the rules r_1: $b \leftarrow DL[S \uplus p; C](a)$ and r_2: $p(a) \leftarrow$, and L contains only the axiom $S \sqsubseteq C$. Then, KB is positive, and we have $\mathit{lfp}(T_{KB}) = \{p(a), b\}$, where $T_{KB}^0(\emptyset) = \emptyset$, $T_{KB}^1(\emptyset) = \{p(a)\}$, and $T_{KB}^2(\emptyset) = \{p(a), b\}$.

We finally describe a fixpoint iteration for stratified dl-programs. Using Theorem 13, we can characterize the strong answer set M_{KB} of a stratified dl-program KB as follows. Let $\widehat{T}_{KB}^i(I) = T_{KB}^i(I) \cup I$, for all $i \geq 0$.

Theorem 14 *Suppose $KB = (L, P)$ has a stratification λ of length $k \geq 0$. Define $M_i \subseteq HB_P$, $i \in \{-1, 0, \ldots, k\}$, as follows: $M_{-1} = \emptyset$, and $M_i = \widehat{T}_{KB_i}^{n_i}(M_{i-1})$ for $i \geq 0$, where $n_i \geq 0$ such that $\widehat{T}_{KB_i}^{n_i}(M_{i-1}) = \widehat{T}_{KB_i}^{n_i+1}(M_{i-1})$. Then, KB is consistent iff $M_k \neq HB_P$, and in this case, $M_k = M_{KB}$.*

Notice that $M_0 = \mathit{lfp}(T_{KB_0})$ and $M_{i-1} = \widehat{T}_{KB_i}^{j}(M_{i-1}) \cap HB_{P_{i-1}}^{\star}$, for any $j \geq 0$, if $\widehat{T}_{KB_i}^{j}(M_{i-1})$ is consistent, which means that $n_i \geq 0$ always exists.

Example 6 Assume that also the dl-rule r_3: $q(x) \leftarrow \mathit{not}\ \neg b$, $\mathit{not}\ DL[S](x)$ is in P of Example 5. Then, the λ assigning 1 to $q(a)$, 0 to $DL[S](a)$, and 0 to all other atoms in $HB_P \cup DL_P$ stratifies KB, and $M_0 = \mathit{lfp}(T_{KB_0}) = \{p(a), b\}$ and $M_1 = \{p(a), b, q(a)\} = M_{KB}$.

Complexity

The complexity of deciding whether a given dl-program has a strong (resp., weak) answer set is summarized in Table 1. There, "mon-dl" means that all dl-atoms in DL_P are monotonic and treated as such in case of strong answer sets. Results on cautious and brave reasoning are easily derived from them by simple reductions (except for positive KB with L in $\mathcal{SHOIN}(\mathbf{D})$); cf. (Eiter *et al.* 2003) for more details.

The complexity results are based on the previous results that deciding answer set existence for a (non-ground) normal program P is complete for NEXP (nondeterministic exponential time) (Dantsin *et al.* 2001), and that deciding satisfiability of a knowledge base L in $\mathcal{SHIF}(\mathbf{D})$

Table 1: Complexity of deciding strong / weak answer set existence for dl-programs KB (completeness results)

$KB=(L,P)$	L in $\mathcal{SHIF}(\mathbf{D})$	L in $\mathcal{SHOIN}(\mathbf{D})$
positive	EXP	NEXP
stratified, mon-dl	EXP	$\mathrm{P}^{\mathrm{NEXP}}$ / $\mathrm{NP}^{\mathrm{NEXP}}$
stratified	EXP	$\mathrm{NP}^{\mathrm{NEXP}}$
general	NEXP	$\mathrm{NP}^{\mathrm{NEXP}}$

(resp., $\mathcal{SHOIN}(\mathbf{D})$) is complete for EXP (exponential time) (Tobies 2001; Horrocks & Patel-Schneider 2003b) (resp., NEXP, assuming unary number encoding; cf. (Horrocks & Patel-Schneider 2003b) and the NEXP-hardness proof for $\mathcal{ACLQIO}$ by Tobies (2001), which implies the NEXP-hardness). Thus, evaluating a ground dl-atom a of form (1) given an interpretation I_p of its input predicates $p = p_1, \ldots, p_m$ (i.e., deciding $I \models_L a$ for each I that coincides on p with I_p) is complete for EXP (resp., co-NEXP) for L from $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$).

The following theorem shows that deciding the existence of strong (resp., weak) answer sets of dl-programs with L in $\mathcal{SHIF}(\mathbf{D})$ is NEXP-complete in the general case, and EXP-complete in the positive and the stratified case.

Theorem 15 *Given Φ and a dl-program $KB=(L,P)$ with L in $\mathcal{SHIF}(\mathbf{D})$, deciding whether KB has a strong (resp., weak) answer set is complete for* NEXP *in the general case, and complete for* EXP *when KB is positive or stratified.*

Proof (sketch). Observe first that for each dl-program KB, the number of ground dl-atoms a is polynomial, and a has exponentially many different concrete inputs I_p in general, but each of them has polynomial size.

For positive KB, we can compute $\mathit{lfp}(T_{KB})$ in exponential time. Note that any ground dl-atom a needs to be evaluated only polynomially often, as its input can increase only that many times. From $\mathit{lfp}(T_{KB})$, it is then immediate whether KB has a strong (resp., weak) answer set, namely iff $\mathit{lfp}(T_{KB}) \neq HB_P$. For other KB, we can, one by one, explore the exponentially many possible inputs of those dl-atoms which disappear in the transform sP^I_L (resp., wP^I_L). For each input, evaluating these dl-atoms and building sP^I_L (resp., wP^I_L) is feasible in exponential time. If we are left with a positive or stratified KB', we aim to compute M_{KB} by (a sequence of) fixpoint iterations as above, and check compliance with the inputs of the dl-atoms. For unstratified KB, we need in addition an (exponential size) guess for the default-negated classical literals, which brings us to NEXP.

The EXP- and NEXP-hardness for positive and general KB, respectively, is inherited from the complexity of plain datalog and normal programs (Dantsin *et al.* 2001). □

The next theorem shows that deciding the existence of strong (resp., weak) answer sets of dl-programs with L in $\mathcal{SHOIN}(\mathbf{D})$ ranges from NEXP-completeness in the positive case to $\mathrm{NP}^{\mathrm{NEXP}}$-completeness in the general case.

Theorem 16 *Given Φ and a dl-program $KB=(L,P)$ with L in $\mathcal{SHOIN}(\mathbf{D})$, deciding whether KB has a strong (resp., weak) answer set is complete for* $\mathrm{NP}^{\mathrm{NEXP}}$ *in the general and in the stratified case, complete for* $\mathrm{P}^{\mathrm{NEXP}}$ *(resp.,* $\mathrm{NP}^{\mathrm{NEXP}}$*) when KB is stratified and has only monotonic dl-atoms, and complete for* NEXP *when KB is positive.*

Proof (sketch). We use the following observation: A positive KB has a strong (resp., weak) answer set, just if there exists an interpretation I and a subset $S \subseteq \{a \in DL_P \mid I \not\models_L a\}$ such that the positive logic program $P_{I,S}$, obtained from $\mathit{ground}(P)$ by deleting each rule that contains a dl-atom $a \in S$ and all remaining dl-atoms, has a strong answer set included in I. A suitable I and S, along with proofs $L \not\models_I a$ for all $a \in S$, can be guessed and verified in exponential time. The matching NEXP-hardness follows from co-NEXP-hardness of dl-atom evaluation.

For non-positive KB, we can guess inputs I_p for all dl-atoms, and evaluate them with a NEXP oracle in polynomial time. For the (monotonic) ones remaining in sP^I_L, we can further guess a chain $\emptyset = I^0_p \subset I^1_p \subset \cdots \subset I^k_p = I_p$, along which their inputs are increased in a fixpoint computation for sP^I_L, and evaluate the dl-atoms on it in polynomial time with a NEXP oracle. We then ask a NEXP oracle if an interpretation I exists which is the answer set of sP^I_L (resp., wP^I_L) compliant with the inputs and valuations of the dl-atoms and such that their inputs increase in fixpoint computation. This yields the $\mathrm{NP}^{\mathrm{NEXP}}$ upper bounds. For a strong answer set of a stratified, mon-dl KB, guesses can be avoided by increasing the monotonic dl-atoms along a stratification, and the problem is in $\mathrm{P}^{\mathrm{NEXP}}$.

We can show matching lower bounds by a generic reduction from Turing machines M, exploiting the NEXP-hardness proof for $\mathcal{ACLQIO}$ by Tobies (2001). The idea is to use a dl-atom to decide the result of the i-th oracle call made by a polynomial-time bounded M with access to a NEXP oracle, where the results of the previous calls are known and input to the dl-atom. By a proper sequence of dl-atom evaluations, the result of M's computation on input w can be obtained; a nondeterministic M is modeled by providing random bits generated by dl-atoms or unstratified rules. □

Related Work

The works by Donini *et al.* (1998), Levy & Rousset (1998), and Rosati (1999) are representatives of hybrid approaches using description logic as input. More specifically, Donini *et al.* (1998) combine plain datalog (no disjunction and negation) with the description logic $\mathcal{ALC}$. An integrated knowledge base has a structural component in $\mathcal{ALC}$ and a relational component in datalog; their integration lies in using concepts from the former as constraints in rule bodies of the latter. Donini *et al.* (1998) also present a technique for answering conjunctive queries (existentially quantified conjunctions of atoms) with such constraints, where SLD-resolution is integrated with an inference method for $\mathcal{ALC}$. Closely related is the approach by Levy & Rousset (1998), combining Horn rules with the description logic $\mathcal{ALCNR}$. In contrast to Donini *et al.*'s approach (1998), it allows for roles as constraints in rule bodies and does not require safety for variables in constraints. Also Levy & Rousset (1998) present a technique for answering disjunctive queries, i.e., disjunctions of conjunctive queries, conditioned on conjunctive queries. Finally, Rosati (1999) combines disjunctive datalog (with classical and default negation) with $\mathcal{ALC}$ based on a generalized answer set semantics. Like Levy & Rousset (1998), he allows for roles as constraints in rule bodies, and, similar to Donini *et al.* (1998), safety is not requested. Moreover, answering queries given by ground atoms is discussed, based on a combination of ordinary answer set programming with inference in $\mathcal{ALC}$.

Some representatives of approaches reducing description logic inference to logic programming are the works by Van Belleghem *et al.* (1997), Baral (2003) (cf. also (Alsaç & Baral 2001)), Swift (2004), Grosof *et al.* (2003), and Heymans and Vermeir (2003a; 2003b). In detail, Van Belleghem *et al.* (1997) presents a mapping of description logic knowledge bases in $\mathcal{ALCN}$ to open logic programs, and shows how other description logics correspond to sublanguages of open logic programs. It also explores the computational correspondences between a typical algorithm for description logic inference and the resolution procedure for open logic programs. The works by Baral (2003) and Swift (2004) reduce inference in the description logic $\mathcal{ALCQI}$ to query answering from the answer sets of logic programs (with default negation, but no disjunction and classical negation). Grosof *et al.* (2003) shows especially how inference in a subset of the description logic $\mathcal{SHOIQ}$ can be reduced to inference in a subset of Horn programs (in which no function symbols, negations, and disjunctions are permitted), and vice versa. Finally, Heymans & Vermeir (2003a; 2003b) extend disjunctive logic programming under the answer set semantics by inverses and an infinite universe. As shown, this extension is decidable for rules forming a tree structure, and inference in $\mathcal{SHIF}$ extended by transitive role closures can be simulated in it.

Closest in spirit to our work is perhaps the approach by Rosati (1999), which also combines description logics and answer set programming. There are, however, several crucial differences. (1) Rather than $\mathcal{ALC}$, we use the more expressive description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which underly OWL Lite and OWL DL, respectively. On the other hand, Rosati (1999) considers disjunctive rule heads; we refrain from this here, but our approach can be easily extended in this direction (keeping the main conceptual ideas). (2) Instead of using roles and concepts from L *as constraints* in rule bodies of a logic program P, we allow for *queries to* L in rule bodies of P, where every query also allows for specifying an *input from* P, and thus for a *flow of knowledge from* P *to* L besides the flow of knowledge from L to P. Thus, in our approach, inference from L also depends on what is encoded in P, which is not the case in Rosati's approach. Furthermore, in our approach, queries to L are not subject to any safety condition and can be orthogonally combined with classical and default negation. (3) We allow for a technical separation and thus a more flexible combination of description logic inference and logic programming. Namely, our approach permits cautious as well as brave reasoning under the answer set semantics, while Rosati (1999) technically permits only cautious reasoning. Indeed, in Rosati's method, an integrated knowledge base $KB = (L, P)$ represents all pairs (I, S) of models I of L and answer sets S of P, while in our work, KB represents all answer sets S of P, where queries are evaluated *relative to each single answer set* S and all models I of L compatible with S. Furthermore, the technical separation complies with the impedance mismatch of the usual interpretations of answer set programs (finite Herbrand interpretations) and of description logics (general first-order interpretations over possibly infinite domains). This mismatch cannot be easily eliminated when combining existing implemented systems.

Finally, we mention recent work by Antoniou (2002), which deals with a combination of defeasible reasoning with description logics. Like in other work mentioned above, the considered description logic serves in that approach only as an input for the default reasoning mechanism running on top of it. Also, early work on dealing with default information in the context of description logics is the method due to Baader & Hollunder (1995), where Reiter's default logic is adapted to terminological knowledge bases, differing significantly from our approach. Less closely related work includes also the investigations by Baumgartner, Furbach, & Thomas (2002) and Provetti, Bertino, & Salvetti (2003).

Summary and Conclusion

Towards the integration of rules and ontologies in the Semantic Web, we have proposed a combination of logic programming under the answer set semantics with the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which stand behind OWL Lite and OWL DL, respectively. We have introduced dl-programs, which consist of a description logic knowledge base L and a finite set P of dl-rules, which may also contain queries to L, possibly default negated, in their bodies. We have defined Herbrand models for dl-programs, and shown that satisfiable positive dl-programs have a unique least Herbrand model. More generally, consistent stratified dl-programs can be associated with a unique minimal Herbrand model that is characterized through iterative least Herbrand models. We have then generalized the unique minimal Herbrand model semantics for positive and stratified dl-programs to a strong answer set semantics for all dl-programs, which is based on a reduction to the least model semantics of positive dl-programs. We have also defined a weak answer set semantics based on a reduction to the answer sets of ordinary logic programs. We have then given fixpoint characterizations for the unique minimal Herbrand model semantics of positive and stratified dl-programs, and shown how to compute these models by finite fixpoint iterations. Furthermore, we have given a precise picture of the complexity of deciding strong and weak answer set existence for a dl-program.

On the computational side, we have realized a prototype implementation for weak answer sets, employing the description logic engine RACER (Haarslev & Möller 2001) and the answer set engine `DLV` (Leone *et al.* 2002), which is based on interleaved calls until a fixpoint is reached. An interesting subject for further research is to find efficient means for implementing the approach as a whole. To this end, one may investigate mappings to answer set programming itself, which may utilize work on mapping description logics to (disjunctive) logic programs (Grosof *et al.* 2003; Motik, Volz, & Maedche 2003; Swift 2004). Note that the addressed problems of complexity within EXP (resp., NEXP) can be polynomially transformed into deciding consequence from an ordinary (negation-free) datalog program (resp., deciding answer set existence of an ordinary normal logic program). The problems with higher complexity can be polynomially reduced to disjunctive logic programming, since $\mathrm{NP}^{\mathrm{NEXP}} \subseteq \mathrm{NEXP}^{\mathrm{NP}}$, and for disjunctive logic programs, deciding answer set existence, as well as brave reasoning, is $\mathrm{NEXP}^{\mathrm{NP}}$-complete (Dantsin *et al.* 2001). However, intuitively, $\mathrm{NP}^{\mathrm{NEXP}}$ has much less computational power than $\mathrm{NEXP}^{\mathrm{NP}}$, and thus the full power of disjunctive logic programming may not be needed. It thus remains to find efficient and useful transformations that are tailored to the complexity of the problems at hand.

Another interesting topic of future research is to extend our approach to dl-programs with disjunctions, NAF-literals, and dl-atoms in the heads of dl-rules.

Acknowledgments. This work was partially supported by the Austrian Science Fund under grants Z29-N04 and P17212-N04, and the European Commission under grant FET-2001-37004 WASP, the IST-2001-33123 CologNeT Network of Excellence, the IST FP6 REWERSE Network of Excellence, and a Marie Curie Individual Fellowship under contract number HPMF-CT-2001-001286 (disclaimer: The authors are solely responsible for information communicated and the European Commission is not responsible for any views or results expressed). We would like to thank Ian Horrocks and Ulrike Sattler for providing valuable information on complexity-related issues during the preparation of this paper. We are also grateful to the reviewers for their constructive comments, which helped to improve our work.

Appendix

We now give some further details on the dl-program $KB_S = (L_S, P_S)$ of Example 1. In addition to the dl-rules (1)–(9), the set P_S also contains the following dl-rules:

$author(per_1)$; $author(per_2)$; $author(per_3)$; …
$area(A)$; $area(B)$; $area(C)$; $area(D)$;
$cluster(T1)$; $cluster(T2)$;
$key(Belief\_Revision)$;
$key(Nonmonotonic\_Reasoning)$;
$key(Answer\_Set\_Programming)$; …

The description logic knowledge base L_S is partially given below (note that in our current prototype implementation based on RACER, the ontology as well as the logic program have to be extended by workarounds since RACER does not support individuals as part of concept expressions). Here, D_{string} and $\mathrm{D}_{\mathbb{N}}$ denote the domains of the datatypes of strings and natural numbers, respectively.

$\geq 1\ title \sqsubseteq Publication$; $\top \sqsubseteq \forall title.\mathrm{D}_{string}$;
$\geq 1\ year \sqsubseteq Publication$; $\top \sqsubseteq \forall year.\mathrm{D}_{\mathbb{N}}$;
$\geq 1\ firstname \sqsubseteq Person$; $\top \sqsubseteq \forall firstname.\mathrm{D}_{string}$;
$\geq 1\ lastname \sqsubseteq Person$; $\top \sqsubseteq \forall lastname.\mathrm{D}_{string}$;
$\geq 1\ keyword \sqsubseteq Paper$; $\top \sqsubseteq \forall keyword.Kw$;
$\geq 1\ cites \sqsubseteq Paper$; $\top \sqsubseteq \forall cites.Paper$;
$\geq 1\ contains \sqsubseteq Area$; $\top \sqsubseteq \forall contains.Kw$;
$\geq 1\ hasAuthor \sqsubseteq Paper$; $\top \sqsubseteq \forall hasAuthor.Person$;
$\geq 1\ expert \sqsubseteq Person$; $\top \sqsubseteq \forall expert.Area$;
$\geq 1\ inArea \sqsubseteq Paper$; $\top \sqsubseteq \forall inArea.Area$;
$\geq 1\ hasMember \sqsubseteq TopicCluster$; $\top \sqsubseteq \forall hasMember.Kw$;
$isContainedIn = contains^-$;
$isAuthorOf = hasAuthor^-$;
$isMemberOf = hasMember^-$;
$Paper \sqsubseteq Publication$;
$Referee \sqsubseteq Person$;
$\exists inArea.\{A\} = \exists keyword.(\exists isContainedIn.\{A\})$;
$\exists expert.\{A\} = \exists isAuthorOf.(\exists inArea.\{A\})$;
$\exists inArea.\{B\} = \exists keyword.(\exists isContainedIn.\{B\})$;
$\exists expert.\{B\} = \exists isAuthorOf.(\exists inArea.\{B\})$;
$\exists inArea.\{C\} = \exists keyword.(\exists isContainedIn.\{C\})$;
$\exists expert.\{C\} = \exists isAuthorOf.(\exists inArea.\{C\})$;
$\exists inArea.\{D\} = \exists keyword.(\exists isContainedIn.\{D\})$;
$\exists expert.\{D\} = \exists isAuthorOf.(\exists inArea.\{D\})$;

$Kw(Belief\_Revision)$; $Kw(Frame\_Systems)$;
$Kw(Intelligent\_Agents)$; $Kw(Bioinformatics)$;
…
$Area(A)$; $Area(B)$; $Area(C)$; $Area(D)$;
$contains(A, Belief\_Revision)$;
$contains(A, Default\_Reasoning)$;
$contains(B, Frame\_Systems)$;
$contains(B, Ontologies)$;
$contains(C, Semantic\_Web)$;
…
$TopicCluster(T_1)$;
$hasMember(T_1, Semantic\_Web)$;
$hasMember(T_1, OWL)$;
$hasMember(T_1, Ontologies)$;
$TopicCluster(T_2)$;
$hasMember(T_2, Coherence\_and\_Coordination)$;
…
$Person(per_1)$;
$firstname(per_1, \text{“}Vladimir\text{”})$;
$lastname(per_1, \text{“}Lifschitz\text{”})$;
$Person(per_2)$;
$firstname(per_2, \text{“}Michael\text{”})$;
$lastname(per_2, \text{“}Gelfond\text{”})$;
…
$Referee(per_1)$;
$Referee(per_2)$;
…
$Paper(pub_1)$;
$title(pub_1, \text{“}Classical\ Negation\ in\ Logic\ Programs\ and\ Disjunctive\ Databases\text{”})$;
$year(pub_1, \text{“}1991\text{”})$;
$hasAuthor(pub_1, per_1)$;
$hasAuthor(pub_1, per_2)$;
$keyword(pub_1, Answer\_Set\_Programming)$;
$keyword(pub_1, Disjunctive\_Logic\_Programming)$.
…

References

Alsaç, G., and Baral, C. 2001. Reasoning in description logics using declarative logic programming. Technical report, Department of Computer Science and Engineering, Arizona State University.

Antoniou, G. 2002. Nonmonotonic rule systems on top of ontology layers. In *Proc. ISWC-2002*, volume 2342 of *LNCS*, 394–398. Springer.

Baader, F., and Hollunder, B. 1995. Embedding defaults into terminological representation systems. *J. Automated Reasoning* 14:149–180.

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge, UK: Cambridge University Press.

Baumgartner, P.; Furbach, U.; and Thomas, B. 2002. Model-based deduction for knowledge representation. In *Proc. WLP-2002*, 156–166.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web. *Scientific American* 284(5):34–43.

Berners-Lee, T. 1999. *Weaving the Web*. San Francisco, CA: Harper.

Boley, H.; Tabet, S.; and Wagner, G. 2001. Design rationale of RuleML: A markup language for semantic web rules. In *Proc. SWWS-2001*.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys* 33(3):374–425.

Donini, F. M.; Lenzerini, M.; Nardi, D.; and Schaerf, A. 1998. $\mathcal{AL}$-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems* 10(3):227–252.

Eiter, T.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2003. Combining answer set programming with description logics for the Semantic Web. Technical Report INFSYS RR-1843-03-13, Institut für Informationssysteme, Technische Universität Wien.

Fensel, D.; van Harmelen, F.; Horrocks, I.; McGuiness, D. L.; and Patel-Schneider, P. F. 2001. OIL: An ontology infrastructure for the Semantic Web. *IEEE Intelligent Systems* 16(2):38–45.

Fensel, D.; Wahlster, W.; Lieberman, H.; and Hendler, J., eds. 2002. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press.

Grosof, B. N.; Horrocks, I.; Volz, R.; and Decker, S. 2003. Description logic programs: Combining logic programs with description logics. In *Proc. WWW-2003*.

Haarslev, V., and Möller, R. 2001. RACER System Description. In *Proc. IJCAR-2001*, volume 2083 of *LNAI*, 701–705. Springer.

Hendler, J., and McGuiness, D. L. 2000. The DARPA agent markup language. *IEEE Intelligent Systems* 15(6):67–73.

Heymans, S., and Vermeir, D. 2003a. Integrating ontology languages and answer set programming. In *Proc. 14th Int. Workshop on Database and Expert Systems Applications*, 584–588. IEEE Computer Society.

Heymans, S., and Vermeir, D. 2003b. Integrating Semantic Web reasoning and answer set programming. In *Proc. ASP-2003*, 194–208.

Horrocks, I., and Patel-Schneider, P. F. 2003a. A proposal for an OWL Rules Language. Draft Version (16 October 2003). Available at `www.cs.man.ac.uk/~horrocks/DAML/Rules/WD-OWL-rules-20031016/`.

Horrocks, I., and Patel-Schneider, P. F. 2003b. Reducing OWL entailment to description logic satisfiability. In *Proc. ISWC-2003*, volume 2870 of *LNCS*, 17–29. Springer.

Horrocks, I., and Sattler, U. 2001. Ontology reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ description logic. In *Proc. IJCAI-2001*, 199–204.

Horrocks, I.; Patel-Schneider, P. F.; and van Harmelen, F. 2003. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics* 1(1):7–26.

Horrocks, I.; Sattler, U.; and Tobies, S. 1999. Practical reasoning for expressive description logics. In *Proc. LPAR-1999*, volume 1705 of *LNCS/LNAI*, 161–180. Springer.

Horrocks, I. 2002a. DAML+OIL: A description logic for the Semantic Web. *IEEE Bulletin of the Technical Committee on Data Engineering* 25(1):4–9.

Horrocks, I. 2002b. DAML+OIL: A reason-able web ontology language. In *Proc. EDBT-2002*, volume 2287 of *LNCS*, 2–13. Springer.

Leone, N.; Pfeifer, G.; Faber, W.; Eiter, T.; Gottlob, G.; Koch, C.; Mateis, C.; Perri, S.; and Scarcello, F. 2002. The DLV System for knowledge representation and reasoning. Technical Report INFSYS RR-1843-02-14, Institut für Informationssysteme, Technische Universität Wien.

Levy, A. Y., and Rousset, M.-C. 1998. Combining Horn rules and description logics in CARIN. *Artif. Intell.* 104(1-2):165–209.

Motik, B.; Volz, R.; and Maedche, A. 2003. Optimizing query answering in description logics using disjunctive deductive databases. In *Proc. KRDB-2003*, volume 79 of *CEUR Workshop Proceedings* (`CEUR-WS.org/Vol79/`).

Provetti, A.; Bertino, E.; and Salvetti, F. 2003. Local Closed-World Assumptions for reasoning about Semantic Web data. In *Proc. APPIA-GULP-PRODE 2003*.

Rosati, R. 1999. Towards expressive KR systems integrating datalog and description logics: Preliminary report. In *Proc. 1999 Int. Workshop on Description Logics* (*DL-1999*), 160–164.

Swift, T. 2004. Deduction in ontologies via ASP. In *Proc. LPNMR-2004*, volume 2923 of *LNCS*, 275–288. Springer.

Tobies, S. 2001. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. Doctoral Dissertation, RWTH Aachen, Germany.

Van Belleghem, K.; Denecker, M.; and De Schreye, D. 1997. A strong correspondence between description logics and open logic programming. In *Proc. ICLP-1997*, 346–360. MIT Press.

W3C. 2004. OWL web ontology language overview. W3C Recommendation (10 February 2004). Available at `www.w3.org/TR/2004/REC-owl-features-20040210/`.