**I N F S Y S**
**R** E S E A R C H
**R** E P O R T


INSTITUT FÜR INFORMATIONSSYSTEME
ARBEITSBEREICH WISSENSBASIERTE SYSTEME


# COMBINING ANSWER SET PROGRAMMING WITH DESCRIPTION LOGICS FOR THE SEMANTIC WEB


THOMAS EITER THOMAS LUKASIEWICZ ROMAN SCHINDLAUER
HANS TOMPITS



INFSYS RESEARCH REPORT 1843-03-13
DECEMBER 2003



Institut für Informationssysteme
AB Wissensbasierte Systeme
Technische Universität Wien
Favoritenstrassße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at


TU WIEN

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-03-13, DECEMBER 2003

# COMBINING ANSWER SET PROGRAMMING WITH DESCRIPTION LOGICS FOR THE SEMANTIC WEB

(PRELIMINARY VERSION, 9TH MARCH 2004)

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

**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.

---

[1]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria; e-mail: {eiter, lukasiewicz, roman, tompits}@kr.tuwien.ac.at.

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

**Acknowledgements**: This work has been partially supported by the Austrian Science Fund project Z29-N04 and a Marie Curie Individual Fellowship of the European Community programme "Human Potential" 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.

# Contents

# 1 Introduction

The World Wide Web is impressively successful. Both the information that is stored in the Web and the number of its human users have been growing exponentially in the recent years. For many people, the Web has started to play a fundamental role as a means of providing and searching for information. However, searching the Web in its current form is not always a joyful experience, since today's search engines often return a huge number of answers, many of which are completely irrelevant, while some relevant answers are not returned. One of the main reasons for this problem is that the current Web is designed for human consumption, but not for automated processing through machines, since the HTML standard only allows for describing the layout of Web pages, but not their semantic content.

The *Semantic Web* [5, 6, 13] is an extension of the current 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 Semantic Web will not only allow for more exact answers when we search for information, but also provide knowledge necessary for integrating and comparing information from different sources, and allow for various forms of automated services. Roughly, the main idea behind the Semantic Web is 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 development of the Semantic Web proceeds in layers of Web technologies and standards, where every layer is lying on top of lower layers. The highest layer that has currently reached a sufficient maturity is the Ontology layer in the form of the *OWL Web Ontology Language* [33, 22].

The language OWL provides the three increasingly expressive sublanguages *OWL Lite*, *OWL DL*, and *OWL Full*, where OWL DL basically corresponds to the web ontology language DAML+OIL [18, 19], which was developed by merging DAML [15] and OIL [12]. The languages OWL Lite and OWL DL are essentially very expressive description logics with an RDF syntax [22]. One can therefore exploit a large body of existing previous work on description logic research, for example, to define the formal semantics of the languages, to understand their formal properties (in particular, the decidability and the complexity of key inference problems), and for an automated reasoning support. In fact, as shown in [21], 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 $\mathcal{SHIF}(\mathbf{D})$ is a restriction of $\mathcal{SHOIN}(\mathbf{D})$, and $\mathcal{SHOIN}(\mathbf{D})$ is closely related to the description logic $\mathcal{SHOQ}(\mathbf{D})$ [23].

The next step in the development of the Semantic Web is the realization of the Rules, Logic, and Proof layers, which will be developed on top of the Ontology layer, and which should offer sophisticated representation and reasoning capabilities. A first effort in this direction is *RuleML* (Rule Markup Language) [7], fostering an XML-based markup language for rules and rule-based systems, while the OWL Rules Language [20] 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 Rules 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 are the following:

- 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 $L$.

- 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 $P$. Furthermore, 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.

- 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.

- 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.

- 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.

- 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 and NEXP, respectively).

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 different 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 in this paper; cf. Section 6 for a detailed discussion on related work.

The rest of this paper is organized as follows. Sections 2 and 3 recall normal logic programs under the answer set semantics and the description logics $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$, respectively. In Section 4, we introduce the syntax of dl-programs. We also define Herbrand models of dl-programs, unique minimal Herbrand models of positive and stratified dl-programs, and finally the strong and weak answer set semantics for general dl-programs. Section 5 shows how the unique minimal Herbrand models of positive and stratified dl-programs can be computed through fixpoint iterations. It also provides a precise picture of the complexity

of deciding strong and weak answer set existence for a dl-program. Section 7 summarizes the main results and gives an outlook on future research. Detailed proofs of most results are given in Appendices A and B.

# 2 Normal Programs under the Answer Set Semantics

In this section, we recall normal programs (over classical literals) under the answer set semantics.

## 2.1 Syntax

Let $\Phi$ 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 either a variable from $\mathcal{X}$ or a constant symbol from $\Phi$. An *atom* is an expression of the form $p(t_1, \ldots, t_n)$, where $p$ is a predicate symbol of arity $n \geq 0$ from $\Phi$, and $t_1, \ldots, t_n$ are terms. A *classical literal* (or simply *literal*) $l$ is an atom $p$ or a negated atom $\neg p$. Its *complementary* literal is $\neg p$ (resp., $p$). A *negation as failure literal* (or *NAF-literal*) is a literal $l$ or a default-negated literal $not\, l$. A *normal rule* (or simply *rule*) $r$ is an expression of the form

$$a \leftarrow b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m\,, \qquad m \geq k \geq 0\,, \tag{1}$$

where $a, b_1, \ldots, b_m$ are classical literals. The literal $a$ is the *head* of $r$, while the conjunction $b_1, \ldots, b_k$, $not\, b_{k+1}, \ldots, not\, b_m$ is the *body* of $r$, where $b_1, \ldots, b_k$ (resp., $not\, b_{k+1}, \ldots, not\, b_m$) is the *positive* (resp., *negative*) *body* of $r$. We use $H(r)$ to denote its head literal $a$, and $B(r)$ to denote the set of all its body literals $B^+(r) \cup B^-(r)$, where $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. If the body of $r$ is empty (that is, $k = m = 0$), then $r$ is a *fact*, and we often omit "$\leftarrow$". A *normal program* (or simply *program*) $P$ is a finite set of rules. A *positive program* $P$ is a finite set of "$not$"-free rules.

## 2.2 Semantics

The *Herbrand universe* of a program $P$, denoted $HU_P$, is the set of all constant symbols appearing in $P$. If there is no such constant symbol, then $HU_P = \{c\}$, where $c$ is an arbitrary constant symbol from $\Phi$. As usual, terms, atoms, literals, rules, programs, etc. are *ground* iff they do not contain any variables. The *Herbrand base* of a program $P$, denoted $HB_P$, is the set of all ground (classical) literals that can be constructed from the predicate symbols appearing in $P$ and the constant symbols in $HU_P$. A *ground instance* of a rule $r \in P$ is obtained from $r$ by replacing every variable that occurs in $r$ by a constant symbol from $HU_P$. We use $ground(P)$ to denote the set of all ground instances of rules in $P$.

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

The main reasoning tasks that are associated with normal programs under the answer set semantics are the following: (1) decide whether a given program $P$ has an answer set; (2) given a program $P$ and a ground formula $\phi$, decide whether $\phi$ holds in every (resp., some) answer set of $P$ (*cautious* (resp., *brave*) *reasoning*);

(3) given a program $P$ and an interpretation $I \subseteq HB_P$, decide whether $I$ is an answer set of $P$ (*answer set checking*); and (4) compute the set of all answer sets of a given program $P$.

# 3 $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$

In this section, we recall the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which correspond to the ontology languages OWL Lite and OWL DL, respectively [21, 22].

## 3.1 Syntax

We first describe the syntax of the description logic $\mathcal{SHOIN}(\mathbf{D})$. We assume a set $\mathbf{D}$ of *elementary datatypes*. Every $d \in \mathbf{D}$ is associated with 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 atomic concept from $\mathbf{A}$ is a concept. If $o_1, o_2, \ldots$ are individuals from $\mathbf{I}$, then $\{o_1, o_2, \ldots\}$ is a concept (called *oneOf*). If $C$ and $D$ are concepts, then also $(C \sqcap D)$, $(C \sqcup D)$, and $\neg C$ (called *conjunction*, *disjunction*, and *negation*, respectively). If $C$ is a concept, $R$ is a role from $\mathbf{R}_A \cup \mathbf{R}_A^-$, and $n$ is a nonnegative integer, then $\exists R.C$, $\forall R.C$, $\geq nR$, and $\leq nR$ are concepts (called *exists*, *value*, *atleast*, and *atmost restriction*, respectively). If $U$ is a datatype role from $\mathbf{R}_D$, $n$ is a nonnegative integer, and $d$ is a datatype from $\mathbf{D}$, then $\exists U.d$, $\forall U.d$, $\geq nU$, and $\leq nU$ are concepts (called *datatype exists*, *value*, *atleast*, and *atmost restriction*, respectively). We write $\top$ (resp., $\bot$) to abbreviate $C \sqcup \neg C$ (resp., $C \sqcap \neg C$), and we eliminate parentheses as usual.

*Axioms* are expressions 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 (7) $a = b$ (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 reasons, number restrictions in a knowledge base $L$ are restricted to simple abstract roles [24]: A role $R$ is called *simple* w.r.t. $L$ iff for each role $S$ such that $S \sqsubseteq^\star R$, it holds that $\mathrm{Trans}(S) \notin L$, where $\sqsubseteq^\star$ is the transitive and reflexive closure of $\sqsubseteq$ on $L$, that is, $S \sqsubseteq^\star R$ iff either (i) $S \sqsubseteq R$ is in $L$, or (ii) $S = R$, or (iii) $S \sqsubseteq^\star Q$ and $Q \sqsubseteq^\star R$, for some role $Q$.

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.

## 3.2 Semantics

An *interpretation* $\mathcal{I} = (\Delta, \cdot^\mathcal{I})$ w.r.t. $\mathbf{D}$ consists of a nonempty *(abstract) domain* $\Delta$, which is disjoint from the datatype domain $\mathrm{dom}(\mathbf{D})$, and a mapping $\cdot^\mathcal{I}$ that assigns to each atomic concept from $\mathbf{A}$ a subset of $\Delta$, to each individual $o \in \mathbf{I}$ an element of $\Delta$, to each abstract role from $\mathbf{R}_A$ a subset of $\Delta \times \Delta$, and to each datatype role from $\mathbf{R}_D$ a subset of $\Delta \times \mathrm{dom}(\mathbf{D})$. The mapping $\cdot^\mathcal{I}$ is extended by induction to all concepts and roles as follows (where $\#S$ denotes the cardinality of a set $S$):

- $(\{o_1, o_2, \ldots\})^\mathcal{I} = \{o_1^\mathcal{I}, o_2^\mathcal{I}, \ldots\}$, $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$,
  $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$, and $(\neg C)^\mathcal{I} = \Delta \setminus C^\mathcal{I}$,

- $(\exists R.C)^{\mathcal{I}} = \{x \in \Delta \mid \exists y\colon (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$,
- $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta \mid \forall y\colon (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$,
- $(\geq nR)^{\mathcal{I}} = \{x \in \Delta \mid \#(\{y \mid (x,y) \in R^{\mathcal{I}}\}) \geq n\}$,
- $(\leq nR)^{\mathcal{I}} = \{x \in \Delta \mid \#(\{y \mid (x,y) \in R^{\mathcal{I}}\}) \leq n\}$,
- $(\exists U.d)^{\mathcal{I}} = \{x \in \Delta \mid \exists y\colon (x,y) \in U^{\mathcal{I}} \wedge y \in \mathrm{dom}(d)\}$,
- $(\forall U.d)^{\mathcal{I}} = \{x \in \Delta \mid \forall y\colon (x,y) \in U^{\mathcal{I}} \rightarrow y \in \mathrm{dom}(d)\}$.
- $(\geq nU)^{\mathcal{I}} = \{x \in \Delta \mid \#(\{y \mid (x,y) \in U^{\mathcal{I}}\}) \geq n\}$,
- $(\leq nU)^{\mathcal{I}} = \{x \in \Delta \mid \#(\{y \mid (x,y) \in U^{\mathcal{I}}\}) \leq n\}$,
- $(R^-)^{\mathcal{I}} = \{(a,b) \mid (b,a) \in R^{\mathcal{I}}\}$.

The *satisfaction* of an 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 $L$ 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 $F$.

Some important reasoning tasks related to description logic knowledge bases $L$ are the following: (1) decide whether a given $L$ is satisfiable; (2) given $L$ and a concept $C$, decide whether $L \not\models C \sqsubseteq \bot$; (3) given $L$ and two concepts $C$ and $D$, decide whether $L \models C \sqsubseteq D$; (4) given $L$, $a \in \mathbf{I}$, and a concept $C$, decide whether $L \models C(a)$; and (5) given $L$, $a, b \in \mathbf{I}$ (resp., $a \in \mathbf{I}$ and $v \in \mathrm{dom}(\mathbf{D})$), and $R \in \mathbf{R}_A$ (resp., $U \in \mathbf{R}_D$), decide whether $L \models R(a,b)$ (resp., $L \models U(a,v)$). Here, (1) is a special case of (2), since $L$ is satisfiable iff $L \not\models \top \sqsubseteq \bot$. Moreover, (2) and (3) can be reduced to each other, since $L \models C \sqcap \neg D \sqsubseteq \bot$ iff $L \models C \sqsubseteq D$. Finally, in $\mathcal{SHOIN}(\mathbf{D})$, (4) and (5) are special cases of (3).

# 4 DL-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.

## 4.1 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.

We first define dl-queries and dl-atoms, which are used to express queries to the description logic knowledge base $L$. 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

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

where each $S_i$ is either a concept or a role, $op_i \in \{\uplus, \uplus, \cap\}$, $p_i$ 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., $op_i = \uplus$)

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. 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 4.1** (**Reviewer Selection**) Suppose we want to assign reviewers to papers, based on certain information about the papers and available persons, using a description logic knowledge base $L_S$ containing 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, stored in a concept *Area*, depending on keyword information. 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. Furthermore, a role *hasMember* associates with a cluster of similar keywords all its members. Consider then the following dl-program:

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

Intuitively, lines (1) and (2) specify the keyword information of two papers, $p_1$ and $p_2$, which should be assigned to reviewers. The rule (3) augments, by choice of the designer, the keyword information with similar ones (hoping for good). The rule (4) queries the augmented $L_S$ to retrieve the areas each paper is classified into, and the 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, (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 description logic knowledge base $L_S$ and the knowledge represented by the above dl-program.

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:

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

## 4.2 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 $\Phi$. 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 $L$, or $I$ *satisfies* $l$ under $L$, denoted $I \models_L l$, iff $l \in I$. It is a *model* of a ground dl-atom $a = DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{c})$ under $L$, or $I$ *satisfies* $a$ 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 $op_i = \cup$;
- $A_i(I) = \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I \text{ does not hold}\}$, for $op_i = \cap$.

We say $I$ is a *model* of a ground dl-rule $r$ iff $I \models_L l$ for all $l \in B^+(r)$ and $I \not\models_L l$ for all $l \in B^-(r)$ implies $I \models_L H(r)$. It is a *model* of a dl-program $KB = (L, P)$, or $I$ *satisfies* $KB$, 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 $P$, the intersection of two models of $P$ is also a model of $P$. The following lemma shows that a similar result holds for positive dl-programs $KB$.

**Lemma 4.2** *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 $KB$.*

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 4.3** *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 4.4** Consider the dl-program $KB$ comprising of the rules (1)–(6) from Example 4.1. Clearly, $KB$ is "$not$"-free. Moreover, as the dl-atoms do not contain $\cap$, they are all monotonic. Thus, 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 and certain dl-atoms. 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; notice, however, that absence of $\uplus$ in (2) is a simple syntactic criterion which implies monotonicity of a dl-atom (cf. also Example 4.4).

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 others. An *input literal* of $a \in DL_P$ is a ground literal with an input predicate of $a$ and constant symbols in $\Phi$.

A *stratification of* $KB = (L, P)$ (*w.r.t.* $DL_P^+$) is a mapping $\lambda : HB_P \cup DL_P \to \{0, 1, \ldots, k\}$ 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., $a \in DL_P^?$),

and $k \geq 0$ is its *length*. 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 $\lambda$ 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 $M_k$. Notice that $M_{KB}$ is well-defined, since it does not depend on a particular $\lambda$ (cf. Corollary 4.10). The following result shows that $M_{KB}$ is in fact a minimal model of $KB$.

**Theorem 4.5** *Let $KB = (L, P)$ be a stratified dl-program. Then, $M_{KB}$ is a minimal model of $KB$.*

**Example 4.6** Consider the dl-program $KB = (L, P)$ given by the rules and facts from Example 4.1, but without the 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 least model of $P$ 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* of $KB$ (see 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 4.3, has a least model if it is satisfiable.

**Definition 4.1** 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 4.7** *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 $P$.*

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

**Theorem 4.8** *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 $DL_P = DL_P^+$.*

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 4.9** *Let $KB = (L, P)$ 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 unsatisfiable (resp., inconsistent), then $KB$ has no strong answer set.*

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

**Corollary 4.10** *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 4.11** Consider now the full dl-program from Example 4.1. This program is not stratified, in view of rules (6) and (7), which take care of the selection between the different candidates for being reviewers. Each strong answer set containing 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* of dl-programs, 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 $wP_L^I$, 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 $wP_L^I$ is an ordinary ground positive program, which does not contain any dl-atoms anymore, and which also does not contain any NAF-literals anymore. We thus define the weak answer set semantics by reduction to the least model semantics of ordinary ground positive programs as follows.

**Definition 4.2** 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 $wP_L^I$.

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 4.12** *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 $P$.*

The following 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 generally not minimal models of $KB$, even if $KB$ has only monotonic dl-atoms.

**Theorem 4.13** *Let $KB$ be a dl-program. Then, every weak answer set of $KB$ is also a model of $KB$.*

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

**Theorem 4.14** *Let $KB = (L, P)$ be a dl-program. Let $I \subseteq HB_P$ and let $P_L^I$ be obtained from $ground(P)$ by (i) deleting every dl-rule $r$ where either $I \not\models_L a$ for some dl-atom $a \in B^+(r)$, or $I \models_L a$ for some 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 $P_L^I$.*

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. Hence, the set of all weak answer sets of $KB$ can be seen as an approximation of the set of all strong answer sets of $KB$. Note that the converse of the following theorem generally does not hold. That is, there exist dl-programs $KB$, which have a weak answer set that is not a strong answer set.

**Theorem 4.15** *Every strong answer set of a dl-program $KB$ is also a weak answer set of $KB$.*

# 5 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, and of brave and cautious reasoning from the strong and weak answer sets of a dl-program, respectively. We start with a general guess-and-check algorithm for computing weak answer sets.

## 5.1 General Algorithm for Computing Weak Answer Sets

The construction of the weak answer sets of a given dl-program $KB = (L, P)$ can be realized by a straightforward guess-and-check algorithm as follows:

1. We first replace each dl-atom $a$ in $P$ of form

$$DL[S_1 op_1 p_1, \ldots, S_m op_m\ p_m; Q](\mathbf{t}) \tag{3}$$

by a globally new atom $d_a(\mathbf{t})$.

2. We then add to the result of Step 1 all ground facts of form

$$d_a(\mathbf{c}) \vee \neg d_a(\mathbf{c}) \leftarrow, \tag{4}$$

for each dl-atom $a$ occurring in $P$ and each ground term (resp., pair of ground terms) $\mathbf{c} \in HB_L$. Intuitively, the rules of form (4) "guess" the truth values of the dl-atoms of $P$. We denote the resulting program by $P_{guess}$.

3. We construct the answer sets of $P_{guess}$ and check whether the original "guess" of the truth values of the auxiliary atoms $d_a(\mathbf{c})$ is correct with respect to the given description logic knowledge base $L$. That is, for each answer set $I$ of $P_{guess}$ and each dl-atom $a$ of form (3), we check whether it holds that $d_a(\mathbf{c}) \in I$ 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 $op_i = \cup$, and $A_i(I) = \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I \text{ does not hold}\}$, for $op_i = \cap$. If this condition holds, then $I$ is a weak answer set of $P$.

Although this basic algorithm is in general not efficient and leaves room for improvements, it shows that the weak answer set semantics can be realized on top of extant answer set solvers like DLV [10] or GnT [25].

## 5.2 Fixpoint Semantics

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

### Positive dl-programs

For any (not necessarily satisfiable) dl-program $KB = (L, P)$, we define the operator $T_{KB}$ on the subsets of $HB_P$ as follows. For every $I \subseteq HB_P$, let

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

The following lemma shows that for positive $KB$, the operator $T_{KB}$ is monotonic, that is, $I \subseteq I' \subseteq HB_P$ implies $T_{KB}(I) \subseteq T_{KB}(I')$. This result is immediate from the fact that in $ground(P)$, for a positive dl-program $KB = (L, P)$, each dl-atom is monotonic relative to $KB$.

**Lemma 5.1** *Let $KB = (L, P)$ be a positive dl-program. Then, $T_{KB}$ is monotonic.*

The next result gives a characterization of the pre-fixpoints of $T_{KB}$. If $KB$ is satisfiable, then every pre-fixpoint of $T_{KB}$ is either a model of $KB$, or equal to $HB_P$. If $KB$ is unsatisfiable, then $HB_P$ is the only pre-fixpoint of $T_{KB}$. Recall that $I \subseteq HB_P$ is a *pre-fixpoint* of $T_{KB}$ iff $T_{KB}(I) \subseteq I$.

**Proposition 5.2** *Let $KB = (L, P)$ be a positive dl-program. Then, $I \subseteq HB_P$ is a pre-fixpoint of $T_{KB}$ iff $I$ is either (a) a model of $KB$ or (b) equal to $HB_P$.*

Since every monotonic operator has a least fixpoint, which coincides with its least pre-fixpoint, we immediately obtain the following corollary: The least fixpoint of $T_{KB}$, denoted $lfp(T_{KB})$, is given by the least model of $KB$, if $KB$ is satisfiable, and by $HB_P$, if $KB$ is unsatisfiable.

**Corollary 5.3** *Let $KB = (L, P)$ be a positive dl-program. Then, (a) $lfp(T_{KB}) = M_{KB}$, if $KB$ is satisfiable, and (b) $lfp(T_{KB}) = HB_P$, if $KB$ is unsatisfiable.*

The next result shows that the least fixpoint of $T_{KB}$ can be computed by a finite fixpoint iteration (which is based on the assumption that $P$ and the number of constant symbols in $\Phi$ are finite). Note that for every $I \subseteq HB_P$, we define $T^i_{KB}(I) = I$, if $i = 0$, and $T^i_{KB}(I) = T_{KB}(T^{i-1}_{KB}(I))$, if $i > 0$.

**Theorem 5.4** *Let $KB$ be a positive dl-program. Then, $lfp(T_{KB}) = \bigcup_{i=1}^{n} T^i_{KB}(\emptyset) = T^n_{KB}(\emptyset)$ for some $n \geq 0$.*

**Example 5.5** Suppose that $P$ in $KB{=}(L, P)$ consists of the rules $r_1$: $b \leftarrow DL[S \uplus p; C](a)$ and $r_1$: $p(a) \leftarrow$, and $L$ is the axiom $S \sqsubseteq C$. Then, $KB$ is positive, and $lfp(T_{KB}) = \{p(a), b\}$, where $T^0_{KB}(\emptyset) = \emptyset$, $T^1_{KB}(\emptyset) = \{p(a)\}$, and $T^2_{KB}(\emptyset) = \{p(a), b\}$. □

**Stratified dl-programs**

Using Theorem 5.4, we can characterize the answer set $M_{KB}$ of a stratified dl-program $KB$ by a sequence of fixpoint-iterations along a stratification as follows. Let $\widehat{T}^i_{KB}(I) = T^i_{KB}(I) \cup I$, for all $i \geq 0$.

**Theorem 5.6** *Suppose $KB = (L, P)$ has a stratification $\lambda$ of length $k \geq 0$. Define the literal sets $M_i \subseteq HB_P$, $i \in \{-1, 0, \ldots, k\}$, as follows: $M_{-1} = \emptyset$ and*

$$M_i = \widehat{T}^{n_i}_{KB_i}(M_{i-1}), \text{ where } n_i \geq 0 \text{ such that } \widehat{T}^{n_i}_{KB_i}(M_{i-1}) = \widehat{T}^{n_i+1}_{KB_i}(M_{i-1}), i \geq 1.$$

*Then, $KB$ is consistent iff $M_k \neq HB_P$, and in this case, $M_k = M_{KB}$.*

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

**Example 5.7** Assume that also rule $r_3$: $q(x) \leftarrow not\ \neg b, not\ DL[S](x)$ is in $P$ of Example 5.5. Then, the $\lambda$ 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 $M_0 = lfp(T_{KB_0}) = \{p(a), b\}$ and $M_1 = \{p(a), b, q(a)\} = M_{KB}$. □

## 5.3 Complexity

In this subsection, we address the complexity of dl-programs, where we consider the following canonical reasoning problems:

- Deciding whether a given dl-program $KB$ has a strong (resp., weak) answer set.
- Deciding whether a given literal $l \in HB_P$ is in every strong (resp., weak) answer set of a given dl-program $KB$ (Cautious Reasoning).
- Deciding whether a given literal $l \in HB_P$ is in some strong (resp., weak) answer set of a given dl-program $KB$ (Brave Reasoning).

We recall that deciding whether a given normal logic program has an answer set is complete for NEXP (nondeterministic exponential time) [8]. Furthermore, deciding satisfiability of a knowledge base $L$ in

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

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

$\mathcal{SHIF}(\mathbf{D})$ (resp. $\mathcal{SHOIN}(\mathbf{D})$) is complete for EXP (exponential time) [31, 21] (resp., NEXP, assuming unary number encoding; see [21] and the NEXP-hardness proof for $\mathcal{ALQI}$ in [31], which implies the NEXP-hardness of $\mathcal{SHOIN}(\mathbf{D})$).

An easy consequence is that deciding, evaluating a given ground dl-atom $a$ of form (1) in a given dl-program $KB = (L, P)$ and an interpretation $I_p$ of its input predicates $p = p_1, \ldots, p_m$ (i.e., deciding whether $I \models_L a$ for each $I$ which coincides on $p$ with $I_p$) is EXP-complete for $L$ from $\mathcal{SHIF}(\mathbf{D})$ resp. co-NEXP-complete for $L$ from $\mathcal{SHOIN}(\mathbf{D})$.

### 5.3.1 Deciding strong / answer set existence

We first consider the problem of deciding whether a given dl-program $KB$ has an answer set resp. a weak answer set. Table I compactly summarizes our results on this problem. There, mon-dl means that all dl-atoms in $P$ are monotone and treated accordingly in case of strong answer sets. The results are briefly explained as follows.

An important observation is that each dl-program $KB$ give rise to only a polynomial number (in the size of $KB$) of ground dl-atoms $a$. Moreover, there is an exponential number of different concrete inputs to $a$ (given by an interpretation $I_p$ of the respective input predicates), but each of them has polynomial size.

Now for $L$ in $\mathcal{SHIF}(\mathbf{D})$, the evaluation of any ground dl-atom is, as mentioned above, feasible in EXP. Thus, the least fixpoint $lfp(T_{KB})$ for a positive $KB$ is computable in exponential time; notice that any ground dl-atom $a$ needs to be evaluated only polynomially often, as its input can increase only that many times. From $lfp(T_{KB})$ it is immediate whether $KB$ has an answer set resp. a weak answer set, viz. iff $lfp(T_{KB}) \neq HB_P$. For other $KB$, we can, one by one, explore the exponentially many possible inputs of those dl-atoms which disappear in the reduction $sP_L^I$ resp. $wP_L^I$. For each input, evaluating these dl-atoms and building $sP_L^I$ resp. $wP_L^I$ is feasible in exponential time. If we are left with a positive resp. stratified dl-program $KB'$, we need just to compute $M_{KB}$, which we can do by (a sequence of) fixpoint iterations, and check compliance with the input of the dl-atoms. For unstratified $KB$, we need in addition an (exponential size) guess for the value of the default negated classical literals, which brings us to NEXP. The EXP- resp. NEXP-hardness lower bound for positive resp. general $KB$ is inherited from the complexity of datalog and of deciding the existence of an answer set for a normal logic program, respectively [8].

For $L$ in $\mathcal{SHOIN}(\mathbf{D})$, we make use of the following observation: A positive $KB$ has a strong resp. weak answer set, just if there exists an interpretation $I$ and a subset $S \subseteq \{a \in L_P \mid I \not\models_L a\}$, such that the positive logic program $P_{I,S}$ obtained from $ground(P)$ by deleting each rule which contains a dl-atom $a \in S$, and all remaining dl-atoms, has an answer set included in $I$. A suitable $I$ and $S$, along with "proofs" $L \not\models_I a$ for all $a \in S$ (where each proof is a certificate of size bounded by an exponential), can be guessed and verified in exponential time. Hence, deciding the existence of a a strong resp. weak answer set (tantamount to consistency of $KB$) is in NEXP. The matching NEXP-hardness follows from co-NEXP-hardness of

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

Table II: Complexity of cautious reasoning from the strong / weak answer sets of a dl-program (completeness results)

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

Table III: Complexity of brave reasoning from the strong / weak answer sets of a dl-program (completeness results)

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 $sP_L^I$, we can further guess a chain $\emptyset = I_p^0 \subset I_p^1 \subset \cdots \subset I_p^k = I_p$ along which their inputs are increased in a fixpoint computation for $sP_L^I$, 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 $sP_L^I$ resp. $wP_L^I$ compliant with the inputs and valuations of the dl-atoms and as well as their input 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 obtain matching lower bounds by a generic reduction from Turing machines, by exploiting an NEXP-hardness proof for $\mathcal{ACLQI}$ in [31]. The idea is to use a dl-atom to decide the result of the $i$-th oracle call made by a polynomial-time bounded Turing machine $M$ 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 further providing nondeterministically generated bits (either by unstratified rules or dl-atoms).

### 5.3.2 Brave and cautious reasoning

The canonical tasks of cautious and brave reasoning, whether a classical literal $l$ belongs to every (resp., some) strong answer set or weak answer set of $KB$, are as usually easily reduced to the complement of answer set existence and answer set existence itself, respectively, by adding two rules $p \leftarrow l$ and $\neg p \leftarrow l$ in $P$ (resp., $p \leftarrow not\, l$ and $\neg p \leftarrow not\, l$ in $P$), where $p$ is a fresh propositional symbol.

Thus except for brave reasoning from positive dl-programs, the complexity of cautious and brave reasoning is in all cases considered in Table I immediate from the respective complexity of deciding answer set existence for a dl-program. The results are summarized in Tables II and III.

Brave reasoning from the (unique) strong answer set of a positive dl-program is complete for $D^{exp} = \{L \times L' \mid L \in \text{NEXP}, L' \in \text{co-NEXP}\}$, which is the "conjunction" of NEXP and co-NEXP. Intuitively, we have to decide consistency of $KB$, which is in NEXP, and a literal $l$ might be included in the answer set on behalf of a dl-atom, which is in co-NEXP.

Brave reasoning from the weak answer sets of a positive dl-program is harder, however, and in fact $\text{NP}^{\text{NEXP}}$-complete. Intuitively, an exponential number of candidate weak answer sets containing the query literal $l$ might exist which are larger than the (unique) answer set of $KB$. This is a source of complexity which requires another guess.

# 6 Related Work

The works by Donini et al. [9], Levy and Rousset [26], and Rosati [29] are representatives of hybrid approaches using description logic as input. In detail, Donini et al. [9] introduce a combination of (disjunction-, negation-, and function-free) datalog with the description logic $\mathcal{ALC}$. An integrated knowledge base consists of a structural component in $\mathcal{ALC}$ and a relational component in datalog, where the integration of both components lies in using concepts from the structural component as constraints in rule bodies of the relational component. Donini et al. also present a technique for answering conjunctive queries (existentially quantified conjunctions of atoms) with such constraints, where SLD-resolution as an inference method for datalog is integrated with a method for inference in $\mathcal{ALC}$. The closely related work by Levy and Rousset [26] presents a combination of Horn rules with the description logic $\mathcal{ALCNR}$. In contrast to Donini et al. [9], Levy and Rousset also allow for roles as constraints in rule bodies, and do not require the safety condition that variables in constraints in the body of a rule $r$ must also appear in ordinary atoms in the body of $r$. Levy and Rousset [26] also present a technique for answering queries, which are of the very general form of disjunctions of conjunctive queries, conditioned on conjunctive queries. Finally, Rosati [29] presents a combination of disjunctive datalog (with classical and default negation, but without function symbols) with the description logic $\mathcal{ALC}$, which is based on a generalized answer set semantics. Similarly to Levy and Rousset [26], Rosati [29] also allows for roles as constraints in rule bodies, and does not require the above-mentioned safety condition. He presents a technique for answering queries of the form of ground atoms, which is based on a combination of ordinary answer set programming with inference in $\mathcal{ALC}$.

Some representatives of approaches reducing description logic to logic programming are the works by Van Belleghem et al. [32], Alsaç and Baral [3], Swift [30], Grosof et al. [14], and Heymans and Vermeir [16, 17]. In detail, Van Belleghem et al. [32] analyze the close relationship between description logics and open logic programs, and present a mapping of description logic knowledge bases in $\mathcal{ALCN}$ to open logic programs. They also show how other description logics correspond to sublanguages of open logic programs, and they explore the computational correspondences between a typical algorithm for description logic inference and the resolution procedure for open logic programs. The works by Alsaç and Baral [3] and Swift [30] reduce inference in the description logic $\mathcal{ALCQI}$ to query answering from normal logic programs (with default negation, but without disjunctions and classical negations) under the answer set semantics. Grosof et al. [14] show 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 conversely how the latter inference can be reduced to the former. Finally, Heymans and Vermeir [16, 17] present an extension of disjunctive logic programming under the answer set semantics by inverses and an infinite universe. In particular, they prove that this extension is still decidable under the assumption that the rules form a tree structure, and they show how inference in the description logic $\mathcal{SHIF}$ extended by transitive closures of roles can be simulated in it.

Closest in spirit to our work is perhaps the work by Rosati [29], which also combines description logics and answer set programming. There are, however, several crucial differences. (1) Rather than the description logic $\mathcal{ALC}$, we use the more expressive description logics $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$, which underly OWL DL and OWL Lite, respectively. On the other hand, [29] 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 [29]. 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 [29] technically permits only cautious reasoning. Indeed, in [29], 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 [1], which deals with a combination of defeasible reasoning with description logics. Like in other work mentioned above, the considered description logic serves in [1] 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 logic is the approach due to Baader and Hollunder [2], where Reiter's default logic is adapted to terminological knowledge bases, differing significantly from our approach. Less closely related work includes also the investigations in [4] and [28].

## 7 Summary and Outlook

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.

An 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.

On the computational side, it remains to find efficient means for implementation of the approach, and to identify suitable algorithms and strategies for realizing (possibly interleaved) execution of the rules in the dl-program and calls to the description logic knowledge base. On the other hand, also mappings of the semantics to answer set programming itself, for which work on mapping description logics to (disjunctive) logic programs may be utilized [14, 27, 30], may be investigated. We note that the problems in Section 5.3 which have 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 transformed into disjunctive logic programs, since $\mathrm{NP}^{\mathrm{NEXP}} \subseteq \mathrm{NEXP}^{\mathrm{NP}}$, and for disjunctivel logic programs deciding answer set existence, as well as brave reasoning $\mathrm{NEXP}^{\mathrm{NP}}$-complete [8]. However, intuitively $\mathrm{NP}^{\mathrm{NEXP}}$ has much less computational power than $\mathrm{NEXP}^{\mathrm{NP}}$, and thus not the full power of logic programming may be needed. It thus remains to find efficient and useful transformations tailored to the complexity of the problems.

# A Appendix: Proofs for Section 4

**Proof of Lemma 4.2.** Suppose that $I_1, I_2 \subseteq HB_P$ are models of $KB$. We now show that $I = I_1 \cap I_2$ is also a model of $KB$, that is, $I \models_L r$ 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 $KB$. □

**Proof of Theorem 4.5 (sketch).** The statement can be proved by induction along a stratification of $KB$. □

**Proof of Theorem 4.7.** Let $I \subseteq HB_P$. If $KB$ is free of dl-atoms, then $sP_L^I = P^I$. Hence, $I$ 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 $P$. □

**Proof of Theorem 4.8.** (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 a 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 Theorem 4.8, 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 now show that then $I$ is also 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 $(L, sP_L^J)$. Since 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 $I$ being the least model of $(L, sP_L^I)$. This shows that $I$ is a minimal model of $KB$. □

**Proof of Theorem 4.9.** (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 $(L, sP_L^I)$. Since $KB$ is a positive dl-program, it follows that $sP_L^I$ 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 4.8, $KB$ has no strong answer set. □

(b) Let $\lambda$ 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 $(L, sP_L^I)$. Hence,

- $I|HB^\star_{P_0}$ is the least among all models $J \subseteq HB^\star_{P_0}$ of $(L, sP_0{}_L^I)$; and
- if $i > 0$, then $I|HB^\star_{P_i}$ is the least among all models $J \subseteq HB^\star_{P_i}$ of $(L, sP_i{}_L^I)$ 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 among 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 $I = M_{KB}$. □

**Proof of Theorem 4.12.** Let $I \subseteq HB_P$. If $KB$ is free of dl-atoms, then $wP_L^I = P^I$. Thus, $I$ is the least model of $wP_L^I$ iff $I$ is the least model of $P^I$. So, $I$ is a weak answer set of $KB$ iff $I$ is an answer set of $P$. □

**Proof of Theorem 4.13.** 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 $wP_L^I$. As $I$ is the least model of $wP_L^I$ and thus in particular a model of $wP_L^I$, 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 $I \models_L H(r') = H(r)$. This shows that $I \models_L r$. Thus, $I$ is a model of $KB$. □

**Proof of Theorem 4.14.** Immediate by the observation that $wP_L^I$ coincides with $(P_L^I)^I$. □

**Proof of Theorem 4.15.** 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 now 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 contradicts $I$ being the least model of $(L, sP_L^I)$. This shows that $I$ the least model of $wP_L^I$. That is, $I$ is a weak answer set of $KB$. □

# B Appendix: Proofs for Section 5

**Proof of Lemma 5.1.** Let $I \subseteq I' \subseteq HB_P$. Consider any $r \in ground(P)$. 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 $T_{KB}(I) \subseteq T_{KB}(I')$. □

**Proof of Proposition 5.2.** ($\Rightarrow$) Assume that $T_{KB}(I) \subseteq I \subseteq HB_P$. Suppose first that $I$ is consistent. Then, for every $r \in ground(P)$, $I \models_L l$ for all $l \in B(r)$ implies that $I \models_L H(r)$, and thus $I \models_L r$. Hence, $I$ is a model of $KB$. Suppose next that $I$ is not consistent. Then, $T_{KB}(I) = HB_P$, and thus $I = HB_P$.

($\Leftarrow$) Suppose first that $I$ is a model of $KB$. That is, $I \models_L r$ for all $r \in ground(P)$. Equivalently, $I \models_L l$ for all $l \in B(r)$ implies that $I \models_L H(r)$, for all $r \in ground(P)$. It thus follows that $T_{KB}(I) \subseteq I$. Suppose next that $I = HB_P$. Then, $T_{KB}(I) = HB_P = I$. □

**Proof of Theorem 5.4.** Since $T_{KB}$ is monotonic and $HB_P$ is finite, it follows that $T^i_{KB}(\emptyset)$ for $i \geq 0$ is an increasing sequence of sets contained in $lfp(T_{KB})$, and $T^n_{KB}(\emptyset) = T^{n+1}_{KB}(\emptyset)$ for some $n \geq 0$. Since $T^n_{KB}(\emptyset)$ is a fixpoint of $T_{KB}$ that is contained in $lfp(T_{KB})$, it follows that $T^n_{KB}(\emptyset) = lfp(T_{KB})$. □

**Proof of Theorem 5.6 (sketch).** The statement follows from Theorem 5.4. □

# References


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

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

[3] C. Baral. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press, Cambridge, UK, 2002.

[4] P. Baumgartner, U. Furbach, and B. Thomas. Model-based deduction for knowledge representation. In *Proceedings of the 17th Workshop on Logic Programming (WLP-2002)*, pages 156–166, 2002.

[5] T. Berners-Lee. *Weaving the Web*. Harper, San Francisco, CA, 1999.

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

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

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

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

[10] T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Declarative problem-solving using the DLV system. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 79–103. Kluwer Academic Publishers, 2000.

[11] T. Eiter, G. Gottlob, and H. Veith. Modular logic programming and generalized quantifiers. In *Proceedings LPNMR-1997*, volume 1265 of *LNCS/LNAI*, pages 290–309. Springer, 1997.

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

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

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

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

[16] S. Heymans and D. Vermeir. Integrating ontology languages and answer set programming. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, pages 584–588. IEEE Computer Society, 2003.

[17] S. Heymans and D. Vermeir. Integrating semantic web reasoning and answer set programming. In *Proceedings ASP-2003*, pages 194–208, 2003.

[18] I. Horrocks. DAML+OIL: A description logic for the semantic web. *IEEE Bulletin of the Technical Committee on Data Engineering*, 25(1):4–9, 2002.

[19] I. Horrocks. DAML+OIL: A reason-able web ontology language. In *Proceedings EDBT-2002*, volume 2287 of *LNCS*, pages 2–13. Springer, 2002.

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

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

[22] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 2003. To appear.

[23] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ description logic. In *Proceedings IJCAI-01*, pages 199–204, 2001.

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

[25] T. Janhunen, I. Niemelä, P. Simons, and J.-H. You. Partiality and disjunctions in stable model semantics. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning* (*KR 2000*), pages 411–419. Morgan Kaufmann, 2000.

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

[27] B. Motik, R. Volz, and A. Maedche. Optimizing query answering in description logics using disjunctive deductive databases. In F. Bry, C. Lutz, U. Sattler, and M. Schoop, editors, *Proceedings of the 10th International Workshop on Knowledge Representation meets Databases (KRDB 2003), Hamburg, Germany, September 15-16, 2003*, volume 79 of *CEUR Workshop Proceedings*. Technical University of Aachen (RWTH), 2003. Online `http://CEUR-WS.org/Vol79/`.

[28] A. Provetti, E. Bertino, and F. Salvetti. Local Closed-World Assumptions for reasoning about Semantic Web data. In *Proceedings APPIA-GULP-PRODE*, LNCS/LNAI. Springer, 2003.

[29] R. Rosati. Towards expressive KR systems integrating datalog and description logics: Preliminary report. In *Proceedings of the 1999 International Workshop on Description Logics (DL-1999)*, pages 160–164, 1999.

[30] T. Swift. Deduction in ontologies via ASP. In *Proceedings LPNMR-7*. Springer, 2004. To appear.

[31] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001. `http://citeseer.nj.nec.com/tobies01complexity.html`.

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

[33] W3C. OWL web ontology language overview, 2003. W3C Candidate Recommendation (18 August 2003): `http://www.w3.org/TR/2003/CR-owl-features-20030818/`.