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

WELL-FOUNDED SEMANTICS FOR DESCRIPTION LOGIC PROGRAMS IN THE SEMANTIC WEB

THOMAS EITER GIOVAMBATTISTA IANNI

THOMAS LUKASIEWICZ ROMAN SCHINDLAUER

INFSYS RESEARCH REPORT 1843-09-01

MARCH 2009

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 1843-09-01, MARCH 2009

WELL-FOUNDED SEMANTICS FOR DESCRIPTION LOGIC PROGRAMS IN THE SEMANTIC WEB

Thomas Eiter[1] Giovambattista Ianni[3]

Thomas Lukasiewicz[2,1] Roman Schindlauer[1]

Abstract. The realization of the Semantic Web vision, in which computational logic has a prominent role, has stimulated a lot of research on combining rules and ontologies, which are formulated in different formalisms, into a framework that is more useful for describing semantic content. In particular, combining logic programming with the Web Ontology Language (OWL), which is a standard based on description logics, emerged as an important issue for linking the Rules and Ontology Layers of the Semantic Web. Non-monotonic description logic programs (or *dl-programs*) were introduced for such a combination, in which a pair (L, P) of a description logic knowledge base L and a set of rules P with negation as failure is given a model-based semantics that generalizes the answer set semantics of logic programs. In this paper, we reconsider dl-programs and present a well-founded semantics for them as an analog for the other main semantics of logic programs. It generalizes the canonical definition of the well-founded semantics based on unfounded sets, and, as we show, lifts many of the well-known properties from ordinary logic programs to dl-programs. Among these properties: our semantics amounts to a partial model approximating the answer set semantics, which yields for positive and stratified dl-programs a total model coinciding with the answer set semantics; it has polynomial data complexity provided the access to the description logic knowledge base is polynomial; under suitable restrictions, it has lower complexity and even first-order rewritability is achievable. The results add to previous evidence that dl-programs are a versatile and robust combination approach, which moreover is implementable using legacy engines.

---

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

[2]Computing Laboratory, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK; e-mail: thomas.lukasiewicz@comlab.ox.ac.uk.

[3]Dipartimento di Matematica, Università della Calabria, P.te P. Bucci, Cubo 30B, I-87036 Rende, Italy

Acknowledgements: This work has been partially supported by the Austrian Science Fund (FWF) under projects P17212, P20840, and P20841, by the German Research Foundation (DFG) under the Heisenberg Programme, by the Italian Research Ministry (MIUR) under project INTERLINK II04CG8AGG, and by the European Commission under the IST REWERSE Network of Excellence IST-2003-506779, ONTORULE (ICT 231875), and the Marie Curie Individual Fellowship 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).

This paper is a significantly extended and revised version of a paper that has appeared in: *Proceedings RuleML-2004*, pp. 81–97, Hiroshima, Japan, November 2004. *LNCS* 3323, Springer, 2004.

Contents

1 Introduction

During the last years, the *Semantic Web* [Berners-Lee et al. 2001; Fensel et al. 2002] has been gaining momentum as a backbone for future information systems. A layered architecture has been conceived to materialize this vision, with the World Wide Web Consortium (W3C) being a steering force behind. This vision comprises low-level syntactic data levels to high-level semantic layers for which computational logic plays a prominent role. The W3C devotes particular efforts to develop recommended standards, which should ease interoperability of intrinsically distributed applications. Important such standards are, e.g, the Resource Description Framework (RDF) for the Data Layer of the architecture and the Web Ontology Language (OWL), which is based on Description Logics, for the Ontology Layer; the *Rule Interchange Format (RIF)* Working Group currently aims at a standard exchange format for rules at the Rules Layer rather than a common semantics, given the plethora of existing languages and types of rules.

It has been realized that rule bases and ontologies, formulated in different languages, need to be combined in order to have, on the one hand, the expressive capabilities that are needed to model certain scenarios, and on the other hand to make interoperability of knowledge bases in different languages possible. However, due to an impedance mismatch between rule and ontology formalisms, which adhere to different underlying principles, such a combination is non-trivial. Many proposals have been made, cf. [Drabent et al. 2009; Eiter et al. 2008; Motik et al. 2006; Rosati 2006; Lukasiewicz 2007] and references therein, which also give taxonomies to distinguish different types of combinations and discuss fundamental technical issues.

Roughly, there are *homogeneous combinations*, where the rule and the ontology predicates are not distinguished in the integrated framework, and *heterogeneous combinations*, where the rule and the ontology predicates are distinguished; among the latter, *loose couplings*, in which the rule bodies may contain queries to the ontology, and *tight integrations*, in which the integrated language has a semantics that defines models of hybrid knowledge bases by referring to the semantics of the original rule language and to the FOL models of the ontology [Drabent et al. 2009].

An advanced approach of loose coupling are *description logic programs* (or *dl-programs*) [Eiter et al. 2004;2008], which are of the form $KB = (L, P)$, where L is a knowledge base in a description logic, and P is a finite set of description logic rules (or *dl-rules*). Such dl-rules are similar to usual rules in logic programs with negation as failure, but may also contain *queries to* L in their bodies which are given by special atoms (on which possibly default negation may apply). For example, a rule

$$cand(X, P) \leftarrow paperArea(P, A),\ DL[Referee](X),\ DL[expert](X, A)$$

may express that X is a candidate reviewer for a paper P, if the paper is in area A, and X is known to be a referee and an expert for area A. Here, the latter two are queries to the description logic knowledge base L, which has a concept *Referee* and role *expert* in its signature. For the evaluation, the precise definition of *Referee* and *expert* within L is fully transparent, and only the logical contents at the level of inference counts. Thus, dl-programs fully support encapsulation and privacy of L: note indeed that, in many cases, parts of L should be (or are) not accessible. For example, if L contains an ontology about risk assessment in credit assignment, or if L is accessible only via web through a querying service, it must be assumed that only extensional and/or external reasoning services are available for accessing L.

Another important feature of dl-rules is that queries to L 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. Hence, dl-programs allow for building rules on top of ontologies, but also (to some extent) building ontologies on top of rules. This is achieved by dynamic update operators through which the extensional part

of L can be modified for subjunctive querying. For example, the rule

$$paperArea(P, A) \leftarrow DL[keyword \uplus kw; inArea](P, A)$$

intuitively says that paper P is in area A, if P is in A according to the description logic knowledge base L, where the extensional part of the *keyword* role in L (which is known to influence *inArea*) is augmented by the facts of a binary predicate *kw* from the program. In this way, additional knowledge (gained in the program) can be supplied to L before querying. Using this mechanism, also more involved relationships between concepts and/or roles in L can be defined and exploited.

Eiter et al. [2004; 2008] faithfully extended the answer set semantics [Gelfond and Lifschitz 1991] for ordinary normal programs, which is one of the most widely used semantics for nonmonotonic logic programs, to dl-programs. More precisely, they defined the notions of *weak* and *strong answer sets* of dl-programs, which coincide with usual answer sets in the case of ordinary normal programs. The description logic knowledge bases in dl-programs are specified in the well-known description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ which underly OWL Lite and OWL DL [Horrocks and Patel-Schneider 2004; Horrocks et al. 2003], respectively, but may be easily adapted to description logics in the upcoming OWL2 standard [Cuenca Grau et al. 2008]. The resulting formalism is very expressive and facilitates advanced applications like closed-world reasoning, default logic, non-deterministic model generation etc.

However, under a data-oriented perspective, similar as in deductive databases, also the *well-founded semantics* [van Gelder et al. 1991] is of great importance for the Web. It is, besides the answer set semantics, the most widely used semantics for nonmonotonic logic programs. Differently from the answer set semantics, the well-founded semantics remains agnostic in the presence of conflicting information and leaves truth values undefined, rather than to reason by cases in different worlds; on the other hand, it assigns the truth value *false* to a maximal set of atoms that cannot become true during the evaluation of a given program. The well-founded semantics has several attractive features, of which the most important are perhaps that: it extends the perfect model semantics of stratified programs and it has polynomial time complexity (measured by the data size), while the answer set semantics is intractable; indeed, efficient implementations are available, of which XSB[1] is widely known. The well-founded semantics assigns a coherent meaning to *all* logic programs, while some programs may have no answer sets: moreover, it is a skeptical approximation of the answer set semantics, in the sense that every well-founded consequence of a given ordinary normal program P is contained in every answer set of P. For the Web context, the significance of the well-founded semantics is evidenced by the fact that several reasoners in this area adopt it for handling nonmonotonic negation, including $\mathcal{F}$lora-2[2] (which builds on XSB) and OntoBroker[3] that are based on F-Logic, and IRIS and MINS,[4] which target the WSML-Rule language [de Bruijn et al. 2006].

Motivated by these observations, in this paper, we consider the issue of the well-founded semantics for dl-programs. Such a semantics should fulfill some desired properties. Naturally, we expect that it faithfully generalizes the well-founded semantics of ordinary logic programs; that it approximates the answer set semantics of dl-programs, in particular, in the case where negation is layered (where strong answer sets are unique); furthermore, for any underlying description logic with a polynomial data complexity, the data complexity of dl-programs under the well-founded semantics should be polynomial as well, or even lower, depending on the structure of the rules and the description logic knowledge base.

[1] `http://xsb.sourceforge.net/`
[2] `http://flora.sourceforge.net/`
[3] `http://www.ontoprise.de/en/home/products/ontobroker/`
[4] `http://iris-reasoner.org/`, `http://tools.sti-innsbruck.at/mins/`

The semantics proposed in this paper has the above and several other beneficial properties. Our main contributions can be summarized as follows:

- We define the well-founded semantics for normal dl-programs by generalizing Van Gelder *et al.*'s [1991] fixpoint characterization of the well-founded semantics for ordinary normal programs based on *greatest unfounded sets*. While such a characterization adheres to the intuitive definition of well-founded semantics, technical issues require careful thought for the proper extension to hybrid rule languages that incorporate description logics. Our proposal is the first definition of well-founded semantics for such a language that is directly based on the intuitive notion of unfounded set; other related hybrid languages with well-founded semantics [Drabent et al. 2007; Knorr et al. 2007] allow either only limited interaction between the rule and the ontology part, or are defined by alternating fixpoints giving the semantics a more technical flavor (see Section 9). It is important to point out that the dl-programs under the well-founded semantics considered here are modularly defined and not restricted to a specific underlying description logic; they are easily adapted to the description logics of the upcoming OWL 2 proposal.[5]

- We then prove some appealing semantic properties of the well-founded semantics for dl-programs. In particular, it generalizes the well-founded semantics for ordinary normal programs. Moreover, for general dl-programs, the well-founded semantics is a partial model, and for positive (resp., stratified) dl-programs, it is a total model and the canonical least (resp., iterative least) model of these dl-programs. Furthermore, we also show that the well-founded semantics tolerates abbreviations for dl-atoms.

- Generalizing a result by Baral and Subrahmanian [1993], we then show that the well-founded semantics for dl-programs can be characterized in terms of the least and the greatest fixpoint of an operator γ^2_{KB}, which is defined using a generalized Gelfond-Lifschitz transform of dl-programs relative to an interpretation.

- We also show that, similarly as for ordinary normal programs, the well-founded semantics for dl-programs approximates the strong answer set semantics for dl-programs. That is, every *well-founded* ground atom is true in every answer set, and every *unfounded* ground atom is false in every answer set. Hence, every well-founded ground atom and no unfounded ground atom is a cautious (resp., brave) consequence of a dl-program under the strong answer set semantics. Furthermore, we prove that when the well-founded semantics of a dl-program is total, then it is the only strong answer set.

- As for computation, we show how the well-founded semantics of dl-programs KB can be computed by finite sequences of finite fixpoint iterations, using the operator γ_{KB} and the immediate consequence operator T_{KB} of positive dl-programs KB. We also report on an implementation of the well-founded semantics, which is based on these ideas.

- We then give a characterization of the combined complexity of the well-founded semantics for dl-programs, over both $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$. Like for ordinary normal programs, it is lower or equal to the complexity under the answer set semantics for $\mathcal{SHIF}(\mathbf{D})$. More precisely, relative to program complexity [Dantsin et al. 2001], literal inference under the well-founded semantics for dl-programs over $\mathcal{SHIF}(\mathbf{D})$ is EXP-complete, while cautious literal inference under the strong answer

[5] `http://www.w3.org/TR/2008/WD-owl2-profiles-20081202/`

set semantics for dl-programs over $\mathcal{SHIF}(\mathbf{D})$ is complete for co-NEXP [Eiter et al. 2004]. However, the problem is $\mathrm{P}^{\mathrm{NEXP}}$-complete under both the well-founded and the answer set semantics for dl-programs over $\mathcal{SHOIN}(\mathbf{D})$ [Eiter et al. 2008]. Intuitively, the latter is explained by the fact that in case of very expressive description logics, the power of non-determinism in the rules of P can be emulated by the description logic part.

- We also characterize the data complexity of literal inference from dl-programs under the well-founded semantics, which does not increase much compared to the data complexity of query answering in the underlying description logics: For dl-programs over both $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, the problem is P^{NP}-complete under data complexity.

- We then delineate several data tractable cases. In detail, we show that when all dl-queries in a dl-program can be evaluated in polynomial time (e.g., for certain dl-queries over Horn-$\mathcal{SHIQ}$ [Hustadt et al. 2005] as underlying description logic), then reasoning from dl-programs under the well-founded semantics is complete for P under data complexity, and thus has the same data complexity as reasoning from ordinary normal programs under the well-founded semantics. Furthermore, when the evaluation of dl-queries in a dl-program is first-order rewritable (e.g., for certain dl-queries over *DL-Lite* [Calvanese et al. 2007] as underlying description logic), and the dl-program is additionally acyclic, then reasoning from dl-programs under the well-founded semantics is also first-order rewritable, and thus can be done in LOGSPACE under data complexity. Hence, in the latter case, dl-programs under the well-founded semantics can be efficiently evaluated by means of commercial, SQL-expressive relational database systems.

The rest of this paper is organized as follows. In Section 2, we revisit some basic concepts of nonmonotonic logic programs and description logics. Section 3 recalls dl-programs and their answer set semantics as defined in [Eiter et al. 2008]. In Section 4, we introduce the well-founded semantics for dl-programs, and in Section 5, we analyze its semantic properties. Sections 6 and 7 contain complexity characterizations and data tractable cases, respectively, while Section 8 briefly reports on a prototype implementation. After a discussion of related work in Section 9, we give in Section 10 a brief summary and an outlook on future research issues. Note that detailed proofs of all results in the body of the paper are given in Appendices A–D.

2 Preliminaries

In this section, we recall normal programs under the well-founded semantics, as well as the expressive description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$.

2.1 Normal Programs

We now recall the syntax of normal programs and their well-founded semantics.

2.1.1 Syntax

As for the syntax of normal programs, we assume a function-free first-order vocabulary $\Phi = (\mathcal{P}, \mathcal{C})$, consisting of two nonempty finite sets $\mathcal{C}$ and $\mathcal{P}$ of constant and predicate symbols, respectively, and a set $\mathcal{X}$ of variables. We adopt the convention that variables start with an uppercase letter, while constant and predicate

symbols start with a lowercase letter. A *term* is either a variable from $\mathcal{X}$ or a constant symbol from Φ. A *classical literal* (or *literal*) l is an atom a or a negated atom $\neg a$. A *negation-as-failure* (*NAF*) *literal* is an atom a or a default-negated atom $not\, a$. A *normal rule* (or *rule*) r is of the form

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

where $a, b_1, \ldots, b_m$ are atoms. We refer to a as the *head* of r, denoted $H(r)$, while the conjunction $b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m$ is 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 define $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. We say r is a *fact* iff $m = 0$. A *normal program* (or *program*) P is a finite set of rules. We say P is *positive* iff no rule in P contains default-negated atoms.

Example 2.1 All variables X in $\mathcal{X}$ and constant symbols c in Φ are terms; $supplied(cpu, S)$ and $vendor(V)$ are atoms. An example rule is $r = avoid(V) \leftarrow vendor(V), not\, rebate(V)$, which may encode that vendors without rebate are avoided. Then, $H(r) = avoid(V)$, $B^+(r) = \{vendor(V)\}$, and $B^-(r) = \{rebate(V)\}$.

2.1.2 Well-Founded Semantics

The well-founded semantics of normal programs P has many different equivalent definitions [van Gelder et al. 1991; Baral and Subrahmanian 1993]. We recall here the one based on unfounded sets, via the operators U_P, T_P, and W_P.

Let P be a program. *Ground terms*, *atoms*, *literals*, etc., are defined as usual. We denote by HB_P the *Herbrand base* of P, that is, the set of all ground atoms with predicate and constant symbols from P (if P contains no constant symbol, then choose an arbitrary one from Φ), and by $ground(P)$ the set of all ground instances of rules in P (relative to HB_P).

For literals $l = a$ (resp., $l = \neg a$), we use $\neg.l$ to denote $\neg a$ (resp., a), and for sets of literals S, we define $\neg.S = \{\neg.l \mid l \in S\}$ and $S^+ = \{a \in S \mid a \text{ is an atom}\}$. We use $Lit_P = HB_P \cup \neg.HB_P$ to denote the set of all ground literals with predicate and constant symbols from P. A set of ground literals $S \subseteq Lit_P$ is *consistent* iff $S \cap \neg.S = \emptyset$. A *(three-valued) interpretation* relative to P is any consistent set of ground literals $I \subseteq Lit_P$.

A set $U \subseteq HB_P$ is an *unfounded set* of P relative to $I \subseteq Lit_P$, if for every $a \in U$ and every $r \in ground(P)$ with $H(r) = a$, either (i) $\neg b \in I \cup \neg.U$ for some atom $b \in B^+(r)$, or (ii) $b \in I$ for some atom $b \in B^-(r)$. There exists the greatest unfounded set of P relative to I, denoted $U_P(I)$. Intuitively, if I is compatible with P, then all atoms in $U_P(I)$ can be safely switched to false and the resulting interpretation is still compatible with P.

The two operators T_P and W_P on consistent $I \subseteq Lit_P$ are then defined by:

- $T_P(I) = \{H(r) \mid r \in ground(P),\ B^+(r) \cup \neg.B^-(r) \subseteq I\}$;
- $W_P(I) = T_P(I) \cup \neg.U_P(I)$.

The operator W_P is monotonic, and thus has a least fixpoint, denoted $lfp(W_P)$,[6] which is the *well-founded semantics* of P, denoted $WFS(P)$. A ground atom $a \in HB_P$ is *well-founded* (resp., *unfounded*) relative to

[6] As usual, for a generic operator T, we define $T^0(A) = A$ and $T^{i+1}(A) = T(T^i(A))$ for every integer $i > 0$. If T is monotonic, then T has a least fixpoint, denoted $lfp(T)$, and $lfp(T) = T^\infty(\emptyset) = \bigcup_{i \geqslant 0} T^i(\emptyset)$.

P, if a (resp., $\neg a$) is in $lfp(W_P)$. Intuitively, starting with $I = \emptyset$, rules are applied to obtain new positive and negated facts (via $T_P(I)$ and $\neg.U_P(I)$, respectively). This process is repeated until no longer possible.

The unfounded set of a partial interpretation I intuitively collects all those atoms that cannot become true when extending I with further information. An atom b is unfounded iff there is no rule with b in its head and with a body that can be made true. For example, an atom not appearing in any head is clearly unfounded. One crucial point in the definition of unfounded set is that falsity of rule bodies can be testified by unfounded atoms belonging to the same unfounded set, giving a notion of "self-supportedness".

Example 2.2 Consider the ground program $P = \{p \leftarrow not\, q;\ q \leftarrow p;\ p \leftarrow not\, r\}$. For $I = \emptyset$, we have that $T_P(I) = \emptyset$ and $U_P(I) = \{r\}$: p cannot be unfounded because of the first rule and condition (ii), and hence q cannot be unfounded because of the second rule and condition (i). Thus, $W_P(I) = \{\neg r\}$. Since $T_P(\{\neg r\}) = \{p\}$ and $U_P(\{\neg r\}) = \{r\}$, it then follows that $W_P(\{\neg r\}) = \{p, \neg r\}$. As $T_P(\{p, \neg r\}) = \{p, q\}$ and $U_P(\{p, \neg r\}) = \{r\}$, it then follows $W_P(\{p, \neg r\}) = \{p, q, \neg r\}$. Thus, $lfp(W_P) = \{p, q, \neg r\}$. That is, r is unfounded relative to P, and the other atoms are well-founded.

2.2 Description Logics

In this section, we recall the syntax and the semantics of the expressive Description Logics (DLs) $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which provide the logical underpinning of the Web ontology languages OWL Lite and OWL DL, respectively (see [Horrocks and Patel-Schneider 2004; Horrocks et al. 2003] for further details and background). While we focus here on these DLs, dl-programs can be based on many other DLs such as those of the upcoming OWL 2 proposal, with little adaptation (see also Footnote 7).

Intuitively, DLs model a domain of interest in terms of concepts and roles, which represent classes of individuals and binary relations on classes of individuals, respectively. A DL knowledge base encodes in particular subset relationships between classes of individuals, subset relationships between binary relations on classes of individuals, the membership of individuals to classes, and the membership of pairs of individuals to binary relations on classes. Other important ingredients of $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) are datatypes (resp., datatypes and individuals) in concept expressions.

2.2.1 Syntax

We first describe the syntax of $\mathcal{SHOIN}(\mathbf{D})$, which has the following datatypes and elementary ingredients. We assume a set $\mathbf{E}$ of *elementary datatypes* and a set $\mathbf{V}$ of *data values*. A *datatype theory* $\mathbf{D} = (\Delta^{\mathbf{D}}, \cdot^{\mathbf{D}})$ consists of a *datatype* (or *concrete*) *domain* $\Delta^{\mathbf{D}}$ and a mapping $\cdot^{\mathbf{D}}$ that assigns to every elementary datatype a subset of $\Delta^{\mathbf{D}}$ and to every data value an element of $\Delta^{\mathbf{D}}$. The mapping $\cdot^{\mathbf{D}}$ is extended to all datatypes by $\{v_1, \ldots\}^{\mathbf{D}} = \{v_1^{\mathbf{D}}, \ldots\}$. Let $\Psi = (\mathbf{A} \cup \mathbf{R}_A \cup \mathbf{R}_D, \mathbf{I} \cup \mathbf{V})$ be a vocabulary, where $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ are pairwise disjoint (denumerable) sets of *atomic concepts*, *abstract roles*, *datatype* (or *concrete*) *roles*, and *individuals*, respectively. We denote by $\mathbf{R}_A^-$ the set of inverses R^- of all $R \in \mathbf{R}_A$.

Roles and concepts are defined as follows. 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 $C \in \mathbf{A}$ 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 $\neg C$ are concepts (called *conjunction*, *disjunction*, and *negation*, respectively). If C is a concept, R is an abstract role from $\mathbf{R}_A \cup \mathbf{R}_A^-$, and n is a nonnegative integer, then $\exists R.C$, $\forall R.C$, $\geqslant nR$, and $\leqslant nR$ are concepts (called *exists*, *value*, *atleast*, and *atmost restriction*, respectively). If D is a datatype, U is a datatype role from $\mathbf{R}_D$, and n is a nonnegative integer, then $\exists U.D$, $\forall U.D$, $\geqslant nU$, and $\leqslant nU$ are concepts

(called *datatype exists*, *value*, *atleast*, and *atmost restriction*, respectively). We use $\top$ and $\bot$ to abbreviate the concepts $C \sqcup \neg C$ and $C \sqcap \neg C$, respectively, and we eliminate parentheses as usual.

We next define axioms and knowledge bases as follows. An *axiom* is an expression of one of the following forms:

1. $C \sqsubseteq D$, called *concept inclusion axiom*, where C and D are concepts;
2. $R \sqsubseteq S$, called *role inclusion axiom*, where either $R, S \in \mathbf{R}_A$ or $R, S \in \mathbf{R}_D$;
3. $\mathrm{Trans}(R)$, called *transitivity axiom*, where $R \in \mathbf{R}_A$;
4. $C(a)$, called *concept membership axiom*, where C is a concept and $a \in \mathbf{I}$;
5. $R(a, b)$ (resp., $U(a, v)$), called *role membership axiom*, 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 is a data value); and
6. $a = b$ (resp., $a \neq b$), or $=(a, b)$ (resp., $\neq(a, b)$), called *equality* (resp., *inequality*) *axiom*, where $a, b \in \mathbf{I}$.

We also use $F \equiv G$ to abbreviate the two concept or role inclusion axioms $F \sqsubseteq G$ and $G \sqsubseteq F$. A (*description logic*) *knowledge base* L is a finite set of axioms.

For an abstract role $R \in \mathbf{R}_A$, we define $\mathrm{Inv}(R) = R^-$ and $\mathrm{Inv}(R^-) = R$. Let the *transitive and reflexive closure of* $\sqsubseteq$ on abstract roles *relative to* L, denoted $\sqsubseteq^\star$, be defined as follows. For two abstract roles R and S in L, let $S \sqsubseteq^\star R$ relative to L iff either (a) $S = R$, (b) $S \sqsubseteq R \in L$, (c) $\mathrm{Inv}(S) \sqsubseteq \mathrm{Inv}(R) \in L$, or (d) some abstract role Q exists such that $S \sqsubseteq^\star Q$ and $Q \sqsubseteq^\star R$ relative to L. An abstract role R is *simple relative to* L iff, for each abstract role S such that $S \sqsubseteq^\star R$ relative to L, it holds that (i) $\mathrm{Trans}(S) \notin L$ and (ii) $\mathrm{Trans}(\mathrm{Inv}(S)) \notin L$. For decidability, number restrictions in L are restricted to simple abstract roles [Horrocks et al. 1999]. Observe that in $\mathcal{SHOIN}(\mathbf{D})$, concept and role membership axioms can also be expressed through concept inclusion axioms. The knowledge that the individual a is an instance of the concept C can be expressed by the concept inclusion axiom $\{a\} \sqsubseteq C$, while the knowledge that the pair (a, b) (resp., (a, v)) is an instance of the role R (resp., U) can be expressed by $\{a\} \sqsubseteq \exists R.\{b\}$ (resp., $\{a\} \sqsubseteq \exists U.\{v\}$).

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

The following example introduces a DL knowledge base for a product database, which is also used in some subsequent examples.

Example 2.3 (Product Database) A small computer store obtains its hardware from several vendors. It uses the following DL knowledge base L_1, which contains information about the product range that is provided by each vendor and about possible rebate conditions (we assume here that choosing two or more parts from the same seller causes a discount). For some parts, a shop may already be contracted as supplier.

$$\geqslant 1\ supplier \sqsubseteq Shop;\ \top \sqsubseteq \forall supplier.Part;\ \geqslant 2\ supplier \sqsubseteq Discount;$$
$$Shop(s_1);\ Shop(s_2);\ Shop(s_3);$$
$$Part(harddisk);\ Part(cpu);\ Part(case);$$
$$provides(s_1, cpu);\ provides(s_1, case);\ provides(s_2, harddisk);$$
$$provides(s_2, cpu);\ provides(s_3, harddisk);\ provides(s_3, case);$$
$$supplier(s_3, case);\ case \neq cpu;\ case \neq harddisk;\ cpu \neq harddisk.$$

Here, the first two axioms determine $Shop$ and $Part$ as domain and range of the property $supplier$, respectively, while the third axiom constitutes the concept $Discount$ by putting a cardinality constraint on $supplier$.

2.2.2 Semantics

We now define the semantics of $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ in terms of general first-order interpretations, as usual, and we also recall some important reasoning problems in DLs.

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

- $(R^-)^{\mathcal{I}} = \{(a,b) \mid (b,a) \in R^{\mathcal{I}}\}$;
- $\{o_1, \ldots, o_n\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\}$;
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, and $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;
- $(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$;
- $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y\colon (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$;
- $(\geqslant n R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \mid (x,y) \in R^{\mathcal{I}}\}) \geqslant n\}$;
- $(\leqslant n R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \mid (x,y) \in R^{\mathcal{I}}\}) \leqslant n\}$;
- $(\exists U.D)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \in U^{\mathcal{I}} \wedge y \in D^{\mathbf{D}}\}$;
- $(\forall U.D)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y\colon (x,y) \in U^{\mathcal{I}} \rightarrow y \in D^{\mathbf{D}}\}$;
- $(\geqslant n U)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \mid (x,y) \in U^{\mathcal{I}}\}) \geqslant n\}$;
- $(\leqslant n U)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \mid (x,y) \in U^{\mathcal{I}}\}) \leqslant n\}$.

The *satisfaction* of a DL axiom F in the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with respect to $\mathbf{D} = (\Delta^{\mathbf{D}}, \cdot^{\mathbf{D}})$, 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 $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ (resp., $\mathcal{I} \models U(a,v)$ iff $(a^{\mathcal{I}}, v^{\mathbf{D}}) \in U^{\mathcal{I}}$); and (6) $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$ (resp., $\mathcal{I} \models a \neq b$ 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 $\mathcal{I} \models F$. The interpretation $\mathcal{I}$ *satisfies* a DL 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.

Some important reasoning problems related to DL 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, an individual $a \in \mathbf{I}$, and a concept C, decide whether $L \models C(a)$; (5) given L, two individuals $a, b \in \mathbf{I}$ (resp., an individual $a \in \mathbf{I}$ and a data value v), and an abstract role $R \in \mathbf{R}_A$ (resp., a datatype role $U \in \mathbf{R}_D$), decide whether $L \models R(a,b)$ (resp., $L \models U(a,v)$), and (6) given L and two individuals $a, b \in \mathbf{I}$, decide whether $L \models a = b$ or whether $L \models a \neq b$.

Here, (1) is a special case of (2), since L is satisfiable iff $L \not\models \top \sqsubseteq \bot$. Furthermore, (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 $\mathcal{SHOIN}(\mathbf{D})$, since concept and role membership axioms can also be expressed through concept inclusion axioms (see above), (4) and (5) are special cases of (3).

Example 2.4 (Product Database cont'd) Consider again L_1 of Example 2.3. We observe that, for example, $Discount \sqsubseteq Shop$ is not a logical consequence of L_1. Furthermore, $\geqslant 2\ provides(s_3)$ is a logical consequence of L_1, while $Discount(s_3)$ is not.

3 Description Logic Programs

In this section, we recall *description logic programs* (or simply *dl-programs*) under the answer set semantics from [Eiter et al. 2004;2008], which combine DLs (under the general first-order semantics) and normal programs under the answer set semantics. They consist of a DL knowledge base L and a finite set of generalized rules (called *dl-rules*) P. Such rules are similar to usual rules in logic programs with negation as failure, but may also contain *queries to* L in their bodies, possibly default negated. In such a query, it is asked whether a certain DL axiom or its negation logically follows from L. In [Eiter et al. 2004;2008], we considered dl-programs that may also contain classical negation and not necessarily monotonic queries to L. Here, we consider only the case where classical negation is absent and all queries to L are monotonic. The former is in line with the traditional well-founded semantics in the ordinary case, while the latter makes the development of a well-founded semantics for dl-programs simpler, putting the focus on the premier fragment of dl-programs. Indeed, most atoms with queries to L are in fact monotonic (naturally, a dl-program may still contain NAF-literals). Furthermore, non-monotonic queries to L may be naturally emulated by atoms with monotonic queries under well-founded semantics (cf. Section 5).

3.1 Syntax

We now define the syntax of dl-programs. As in Section 2.1, we assume a function-free first-order vocabulary $\Phi = (\mathcal{P}, \mathcal{C})$, consisting of two nonempty finite sets $\mathcal{C}$ and $\mathcal{P}$ of constant and predicate symbols, respectively, and a set $\mathcal{X}$ of variables. A *term* is either a constant symbol from $\mathcal{C}$ or a variable from $\mathcal{X}$. As in Section 2.2, we assume a description logic vocabulary $\Psi = (\mathbf{A} \cup \mathbf{R}_A \cup \mathbf{R}_D, \mathbf{I} \cup \mathbf{V})$, where $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, $\mathbf{I}$, and $\mathbf{V}$ are pairwise disjoint (denumerable) sets of atomic concepts, abstract roles, datatype roles, individuals, and data values, respectively. We assume that $\mathbf{A} \cup \mathbf{R}_A \cup \mathbf{R}_D$ is disjoint from $\mathcal{P}$, while $I_P \subseteq \mathcal{C} \subseteq \mathbf{I} \cup \mathbf{V}$, where I_P is the set of all constant symbols appearing in P.

We define dl-queries and dl-atoms, which are used in rule bodies to express queries to the DL knowledge base L, as follows. 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 and t_2 are terms; or

(d) of the forms $=(t_1, t_2)$ or $\neq(t_1, t_2)$, where t_1 and t_2 are terms.

Note here that $\mathbf{t}$ is the empty argument list in (a), $\mathbf{t} = t$ in (b), and $\mathbf{t} = (t_1, t_2)$ in (c) and (d), and terms are defined as above. A *dl-atom* has the form

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

where each S_i is either a concept, a role, or a special symbol $\theta \in \{=, \neq\}$; $op_i \in \{\uplus, \sqcup\}$; p_i is a unary predicate symbol, if S_i is a concept, and a binary predicate symbol, otherwise; and $Q(\mathbf{t})$ is a dl-query. We

call $p_1, \ldots, p_m$ its *input predicate symbols*. Intuitively, $op_i = \uplus$ (resp., $op_i = \dot{\cup}$) increases S_i (resp., $\neg S_i$) by the extension of p_i. A *dl-rule* r is of the form (1), where any $b_1, \ldots, b_m \in B(r)$ may be a dl-atom. A *dl-program* $KB = (L, P)$ consists of a DL knowledge base L and a finite set of dl-rules P. We say $KB = (L, P)$ is *positive* iff P is positive.

Example 3.1 (Product Database cont'd) Consider the dl-program $KB_1 = (L_1, P_1)$, with L_1 as in Example 2.3 and P_1 given as follows, choosing vendors for needed parts relative to possible rebates:

(1) $vendor(s_2)$; $vendor(s_1)$; $vendor(s_3)$;
(2) $needed(cpu)$; $needed(harddisk)$; $needed(case)$;
(3) $avoid(V) \leftarrow vendor(V), not\ rebate(V)$;
(4) $rebate(V) \leftarrow vendor(V), DL[supplier \uplus buy\_cand; Discount](V)$;
(5) $buy\_cand(V, P) \leftarrow vendor(V), not\ avoid(V), DL[provides](V, P), needed(P),$
$not\ exclude(P)$;
(6) $exclude(P) \leftarrow buy\_cand(V_1, P), buy\_cand(V_2, P), V_1 \neq V_2$;
(7) $exclude(P) \leftarrow DL[supplier](V, P), needed(P)$;
(8) $supplied(V, P) \leftarrow DL[supplier \uplus buy\_cand; supplier](V, P), needed(P)$.

Rules (3)–(5) choose a possible vendor (buy_cand) for each needed part, taking into account that the selection might affect the rebate condition (by feeding the possible vendor back to L_1, where the discount is determined). Rules (6) and (7) assure that each hardware part is bought only once, considering that for some parts a supplier might already be chosen. Rule (8) eventually summarizes all purchasing results.

3.2 Answer Set Semantics

We now define the answer set semantics of dl-programs and summarize some of its semantic properties. We first define (Herbrand) interpretations and the satisfaction of dl-programs in interpretations. The latter hinges on defining the truth of ground dl-atoms in interpretations. In the sequel, let $KB = (L, P)$ be a dl-program over the vocabulary $\Phi = (\mathcal{P}, \mathcal{C})$.

The *Herbrand base* of P, denoted HB_P, is the set of all ground atoms with (a) predicate symbols in $\mathcal{P}$ that occur in P and (b) constant symbols in $\mathcal{C}$. An *interpretation I relative to P* is any subset of HB_P. Such an I is a *model* of a ground atom or dl-atom a (or I *satisfies* a) under L, denoted $I \models_L a$, if the following holds:

- if $a \in HB_P$, then $I \models_L a$ iff $a \in I$;
- if a is a ground dl-atom $DL[\lambda; Q](\mathbf{c})$, where $\lambda = S_1 op_1 p_1, \ldots, S_m op_m p_m$, then $I \models_L a$ iff $L(I; \lambda) \models Q(\mathbf{c})$, where $L(I; \lambda) = L \cup \bigcup_{i=1}^{m} A_i(I)$ and, for $1 \leqslant i \leqslant m$,

$$A_i(I) = \begin{cases} \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, & \text{if } op_i = \uplus; \\ \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, & \text{if } op_i = \dot{\cup}. \end{cases}$$

We say I is a *model* of a ground dl-rule r iff $I \models_L H(r)$ whenever $I \models_L B(r)$, that is, $I \models_L a$ for all $a \in B^+(r)$ and $I \not\models_L a$ for all $a \in B^-(r)$. We say I is a *model* 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.

Observe that the above satisfaction of dl-atoms a in Herbrand interpretations I also involves negated concept inclusion axioms $\neg(C \sqsubseteq D)$, negated concept membership axioms $\neg C(a)$, and negated role membership axioms $\neg R(a,b)$ and $\neg U(a,v)$. For this reason, we slightly extend the standard syntax and semantics of $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ by also allowing such negated axioms.[7] The notions of satisfaction, satisfiability, and entailment are naturally extended to handle such negated axioms. In particular, a first-order interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies $\neg(C \sqsubseteq D)$ (resp., $\neg C(a)$, $\neg R(a,b)$, $\neg U(a,v)$) iff $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$ (resp., $a^{\mathcal{I}} \notin C^{\mathcal{I}}$, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin R^{\mathcal{I}}$, $(a^{\mathcal{I}}, v^{\mathbf{D}}) \notin U^{\mathcal{I}}$). Entailment (for dl-atoms) in the slight extensions of $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ can then be reduced to entailment in $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ [Eiter et al. 2008], respectively.

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 $I' \models_L a$. In this paper, we focus on monotonic ground dl-atoms relative to a dl-program (which seem to be most natural), but one can also define non-monotonic ones (see [Eiter et al. 2004;2008] and Section 9 for further discussion).

Like ordinary positive programs, every positive dl-program KB is satisfiable and has a unique least model, denoted M_{KB}, which naturally characterizes its semantics.

The *strong answer set semantics* of general dl-programs is then defined by a reduction to the least model semantics of positive ones as follows, using a generalized transformation that removes all default-negated atoms in dl-rules. For dl-programs $KB = (L, P)$, 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 $I \models_L a$ for some $a \in B^-(r)$, and (ii) deleting from each remaining dl-rule r the negative body. Notice that sP_L^I generalizes the Gelfond-Lifschitz reduct P^I [Gelfond and Lifschitz 1991]. Let KB^I denote the dl-program (L, sP_L^I). Since KB^I is positive, it has a unique least model. A *strong answer set* (or simply *answer set*) of KB is an interpretation $I \subseteq HB_P$ that coincides with the unique least model of KB^I.

Example 3.2 (Product Database cont'd) The dl-program $KB_1 = (L_1, P_1)$ of Example 3.1 has the following three strong answer sets (only relevant atoms are shown):

$\{supplied(s_3, case);\ supplied(s_2, cpu);\ supplied(s_2, harddisk);\ rebate(s_2); \ldots\};$
$\{supplied(s_3, case);\ supplied(s_3, harddisk);\ rebate(s_3); \ldots\};$
$\{supplied(s_3, case); \ldots\}.$

Since the supplier s_3 was already fixed for the part $case$, two possibilities for a discount remain ($rebate(s_2)$ or $rebate(s_3)$; s_1 is not offering the needed part $harddisk$, and the shop will not give a discount only for the part cpu).

We finally summarize some semantic properties. The strong answer set semantics of dl-programs $KB = (L, P)$ without dl-atoms coincides with the ordinary answer set semantics of P [Gelfond and Lifschitz 1991]. Moreover, strong answer sets of a general dl-program KB are also minimal models of KB. Finally, positive and stratified dl-programs have exactly one strong answer set, which coincides with their canonical minimal model. Here, *stratified dl-programs* are composed of hierarchic layers of positive dl-programs that are linked via default negation [Eiter et al. 2004;2008].

[7] Actually, OWL 2 follows a similar pattern, allowing for negative property membership assertions, cf. `http://www.w3.org/TR/2008/WD-owl2-quick-reference-20081202/`. Negative role membership axioms can also be easily emulated using qualified role expressions, cf. [Eiter et al. 2008]; for DLs with limited expressiveness, $\uplus$ can be simply restricted to concepts.

4 Well-Founded Semantics

In this section, we define the well-founded semantics for dl-programs. We do this by generalizing the well-founded semantics for ordinary normal programs. More specifically, we generalize the definition based on unfounded sets as given in Section 2.

We first define the notion of an unfounded set for dl-programs $KB = (L, P)$. This is not that easy technically: first, truth and falsity of dl-atoms depend on L, besides P. Second, establishing definite falsity of a positive classical atom b in a rule body is as easy as checking that $\neg b$ appears in the current interpretation. Instead, for proving that a positive dl-atom is definitely false, it is necessary to consider a more general sufficient condition, which accounts for any possible further expansion of the current interpretation. These considerations lead to the following notion of unfounded set for dl-programs.

Definition 4.1 (Unfounded Set) Let $I \subseteq Lit_P$ be consistent. A set $U \subseteq HB_P$ is an *unfounded set* of KB relative to I iff the following holds:

$(*)$ for every $a \in U$ and every $r \in ground(P)$ with $H(r) = a$, either (i) $\neg b \in I \cup \neg.U$ for some ordinary atom $b \in B^+(r)$, or (ii) $b \in I$ for some ordinary atom $b \in B^-(r)$, or (iii) for some dl-atom $b \in B^+(r)$, it holds that $S^+ \not\models_L b$ for every consistent $S \subseteq Lit_P$ with $I \cup \neg.U \subseteq S$, or (iv) for some dl-atom $b \in B^-(r)$, $I^+ \models_L b$.

What is new here are conditions (iii) and (iv). Intuitively, (iv) says that $not\, b$ is for sure false, regardless of how I is further expanded, while (iii) says that b will never become true, if we expand I in a way such that all unfounded atoms are kept false. The following examples illustrate the concept of an unfounded set for dl-programs.

Example 4.2 Consider $KB_2 = (L_2, P_2)$, where $L_2 = \{S \sqsubseteq C\}$ and P_2 is as follows:

$$p(a) \leftarrow DL[S \uplus q; C](a); \qquad q(a) \leftarrow p(a); \qquad r(a) \leftarrow not\, q(a),\ not\, s(a).$$

Here, $S_1 = \{p(a), q(a)\}$ is an unfounded set of KB_2 relative to $I = \emptyset$, since $p(a)$ is unfounded due to (iii), while $q(a)$ is unfounded due to (i). The set $S_2 = \{s(a)\}$ is trivially an unfounded set of KB_2 relative to I, since no rule defining $s(a)$ exists.

Relative to $I = \{q(a)\}$, S_1 is not an unfounded set of KB_2 (for $p(a)$, the condition fails) but S_2 is. The set $S_3 = \{r(a)\}$ is another unfounded set of KB_2 relative to I.

Example 4.3 Consider a variant $KB_3 = (L_2, P_3)$ of the dl-program $KB_2 = (L_2, P_2)$ of Example 4.2, where P_3 is obtained from P_2 by negating the dl-literal in P_2. Then, $S_1 = \{p(a), q(a)\}$ is not an unfounded set of KB_3 relative to $I = \emptyset$ (condition (iv) fails for $p(a)$), but $S_2 = \{s(a)\}$ is. Relative to $I = \{q(a)\}$, however, both S_1 and S_2 as well as $S_3 = \{r(a)\}$ are unfounded sets of KB_3.

Example 4.4 Among the unfounded sets of $KB_1 = (L_1, P_1)$ in Example 3.1 relative to $I_0 = \emptyset$, there is $\{buy\_cand(s_1, harddisk),\ buy\_cand(s_2, case),\ buy\_cand(s_3, cpu)\}$ due to (iii), since the dl-atom in rule (5) of P_1 will never evaluate to true for these pairs. It reflects the intuition that the concept $provides$ narrows the choice for buying candidates.

The following lemma shows that the set of unfounded sets of KB relative to I is closed under union, which implies that KB has a greatest unfounded set relative to I.

Lemma 4.5 *Let $KB = (L, P)$ be a dl-program, and let $I \subseteq Lit_P$ be consistent. Then, the set of unfounded sets of KB relative to I is closed under union.*

Based on the above result that KB has a greatest unfounded set relative to I, we now generalize the operators T_P, U_P, and W_P to dl-programs as follows.

Definition 4.6 (T_{KB}, U_{KB}, W_{KB}) The operators T_{KB}, U_{KB}, and W_{KB} on all consistent $I \subseteq Lit_P$ are as follows:

- $a \in T_{KB}(I)$ iff $a \in HB_P$ and some $r \in ground(P)$ exists such that (a) $H(r) = a$, (b) $I^+ \models_L b$ for all $b \in B^+(r)$, (c) $\neg b \in I$ for all ordinary atoms $b \in B^-(r)$, and (d) $S^+ \not\models_L b$ for each consistent $S \subseteq Lit_P$ with $I \subseteq S$, for all dl-atoms $b \in B^-(r)$;
- $U_{KB}(I)$ is the greatest unfounded set of KB relative to I; and
- $W_{KB}(I) = T_{KB}(I) \cup \neg.U_{KB}(I)$.

Note that $T_{KB}(I) \cap U_{KB}(I) = \emptyset$, and thus $W_{KB}(I)$ is indeed well-defined. The following result shows that the three operators are all monotonic.

Lemma 4.7 *Let KB be a dl-program. Then, T_{KB}, U_{KB}, and W_{KB} are monotonic.*

Thus, in particular, W_{KB} has a least fixpoint, denoted $lfp(W_{KB})$. The well-founded semantics of dl-programs can thus be defined as follows.

Definition 4.8 (Well-founded Semantics) Let $KB = (L, P)$ be a dl-program. The *well-founded semantics* of KB, denoted $WFS(KB)$, is defined as $lfp(W_{KB})$. An atom $a \in HB_P$ is *well-founded* (resp., *unfounded*) relative to KB iff a (resp., $\neg a$) belongs to $WFS(KB)$.

The following examples illustrate the well-founded semantics of dl-programs.

Example 4.9 Consider KB_2 of Example 4.2. For $I_0 = \emptyset$, we have $T_{KB_2}(I_0) = \emptyset$ and $U_{KB_2}(I_0) = \{p(a), q(a), s(a)\}$. Hence, $W_{KB_2}(I_0) = \{\neg p(a), \neg q(a), \neg s(a)\}$ (=I_1). In the next iteration, $T_{KB_2}(I_1) = \{r(a)\}$ and $U_{KB_2} = \{p(a), q(a), s(a)\}$. Thus, $W_{KB_2}(I_1) = \{\neg p(a), \neg q(a), r(a), \neg s(a)\}$ (=I_2). Since I_2 is total and W_{KB_2} is monotonic, it follows $W_{KB_2}(I_2) = I_2$ and hence $WFS(KB_2) = \{\neg p(a), \neg q(a), r(a), \neg s(a)\}$. Accordingly, $r(a)$ is well-founded and all other atoms are unfounded relative to KB_2. Note that KB_2 has the unique answer set $I = \{r(a)\}$.

Example 4.10 Now consider KB_3 of Example 4.3. For $I_0 = \emptyset$, we have $T_{KB_3}(I_0) = \emptyset$ and $U_{KB_3}(I_0) = \{s(a)\}$. Hence, $W_{KB_3}(I_0) = \{\neg s(a)\}$ (=I_1). In the next iteration, we have $T_{KB_3}(I_1) = \emptyset$ and $U_{KB_3}(I_1) = \{s(a)\}$. Then, $W_{KB_3}(I_1) = I_1$ and $WFS(KB_3) = \{\neg s(a)\}$; atom $s(a)$ is unfounded relative to KB_3. Note that KB_3 has no answer set.

Example 4.11 Consider again $U_{KB_1}(I_0 = \emptyset)$ of Example 4.4. Then, $W_{KB_1}(I_0)$ consists of $\neg U_{KB_1}(I_0)$ and all facts of P_1. This input to the first iteration along with (iii) applied to rule (8) adds those $supplied$ atoms to $U_{KB_1}(I_1)$ that correspond to the (negated) buy_cand atoms of $U_{KB_1}(I_0)$. Then, $T_{KB_1}(I_1)$ contains $exclude(case)$ which forces additional buy_cand atoms into $U_{KB_1}(I_2)$, regarding (i) and rule (5). The same unfounded set thereby includes $rebate(s_1)$, stemming from rule (4). As a consequence, $avoid(s_1)$ is in $T_{KB_1}(I_3)$. Eventually, the final $WFS(KB_1)$ is not able to make any positive assumption about choosing a new vendor (buy_cand), but it is clear about s_1 being definitely not able to contribute to a discount situation, since a supplier for $case$ is already chosen in L_1, and s_1 offers only a single further part.

5 Semantic Properties

In this section, we explore the semantic properties of the well-founded semantics for dl-programs, and their relationship to the strong answer set semantics. An immediate result is that it conservatively extends the well-founded semantics for ordinary normal programs.

Theorem 5.1 *Let $KB = (L, P)$ be a dl-program without dl-atoms. Then, the well-founded semantics of KB coincides with the well-founded semantics of P.*

The next result shows that the well-founded semantics of a dl-program $KB = (L, P)$ is a partial model of KB. Here, a consistent $I \subseteq Lit_P$ is a *partial model* of KB iff some consistent $J \subseteq Lit_P$ exists such that (i) $I \subseteq J$, (ii) J^+ is a model of KB, and (iii) J is *total*, that is, $J^+ \cup (\neg.J)^+ = HB_P$. Intuitively, the three-valued I can be extended to a (two-valued) model $I' \subseteq HB_P$ of KB.

Theorem 5.2 *Let KB be a dl-program. Then, $WFS(KB)$ is a partial model of KB.*

Importantly, the well-founded semantics for dl-programs can be characterized in terms of the least and the greatest fixpoint of a monotonic operator γ^2_{KB} similar as the well-founded semantics for ordinary normal programs [Baral and Subrahmanian 1993]. We then use this characterization to derive further properties of the well-founded semantics for dl-programs.

Definition 5.3 For a dl-program $KB = (L, P)$, let the operator γ_{KB} on $I \subseteq HB_P$ be

$$\gamma_{KB}(I) = M_{KB^I},$$

which is the least model of the positive dl-program $KB^I = (L, sP^I_L)$.

The next result shows that γ_{KB} is anti-monotonic, like its counterpart for ordinary normal programs [Baral and Subrahmanian 1993]. Note that this result holds only if all dl-atoms in P are monotonic.

Proposition 5.4 *Let $KB = (L, P)$ be a dl-program. Then, γ_{KB} is anti-monotonic.*

Hence, the operator $\gamma^2_{KB}(I) = \gamma_{KB}(\gamma_{KB}(I))$, for all $I \subseteq HB_P$, is monotonic and thus has a least and a greatest fixpoint, denoted $lfp(\gamma^2_{KB})$ and $gfp(\gamma^2_{KB})$, respectively. We can use these fixpoints to characterize the well-founded semantics of KB.

Theorem 5.5 *Let $KB = (L, P)$ be a dl-program. Then, an atom $a \in HB_P$ is well-founded* (*resp., unfounded*) *relative to KB iff $a \in lfp(\gamma^2_{KB})$* (*resp., $a \notin gfp(\gamma^2_{KB})$*).

Example 5.6 Consider again the dl-program KB_1 of Example 3.1. The set $lfp(\gamma^2_{KB_1})$ contains the atoms $avoid(s_1)$ and $supplied(s_3, case)$, while $gfp(\gamma^2_{KB_1})$ does not contain $rebate(s_1)$. Thus, $WFS(KB_1)$ contains the literals $avoid(s_1)$, $supplied(s_3, case)$, and $\neg rebate(s_1)$, corresponding to the result of Example 4.11 (and, moreover, to the intersection of all answer sets of KB_1).

The next theorem shows that the well-founded semantics for dl-programs approximates their strong answer set semantics. That is, every well-founded ground atom is true in every answer set, and every unfounded ground atom is false in every answer set.

Theorem 5.7 *Let $KB = (L, P)$ be a dl-program. Then, every strong answer set of KB includes all atoms $a \in HB_P$ that are well-founded relative to KB and no atom $a \in HB_P$ that is unfounded relative to KB.*

A ground atom a is a *cautious* (resp., *brave*) *consequence under the strong answer set semantics* of a dl-program KB iff a is true in every (resp., some) strong answer set of KB. Hence, under the strong answer set semantics, every well-founded and no unfounded ground atom is a cautious (resp., brave) consequence of KB.

Corollary 5.8 *Let $KB = (L, P)$ be a dl-program. Then, under the strong answer set semantics, every well-founded atom $a \in HB_P$ relative to KB is a cautious* (*resp., brave*) *consequence of KB, and no unfounded atom $a \in HB_P$ relative to KB is a cautious* (*resp., brave*) *consequence of a satisfiable KB.*

If the well-founded semantics of a dl-program $KB{=}(L, P)$ is total, that is, contains either a or $\neg a$ for every $a \in HB_P$, then it specifies the only strong answer set of KB.

Theorem 5.9 *Let $KB = (L, P)$ be a dl-program. If every atom $a \in HB_P$ is either well-founded or unfounded relative to KB, then the set of all well-founded atoms $a \in HB_P$ relative to KB is the only strong answer set of KB.*

Like in the case of ordinary normal programs, the well-founded semantics for positive and stratified dl-programs is total and coincides with their least model semantics and iterative least model semantics, respectively. This result can be elegantly proved using the characterization of the well-founded semantics given in terms of the γ^2_{KB} operator.

Theorem 5.10 *Let $KB = (L, P)$ be a dl-program. If KB is positive* (*resp., stratified*)*, then* (*a*) *$WFS(KB)$ is a total model, that is, $WFS(KB)^+ \cup (\neg.\, WFS(KB))^+ = HB_P$, and* (*b*) *$WFS(KB) \cap HB_P$ is the least model* (*resp., the iterative least model*) *of KB, which coincides with the unique strong answer set of KB.*

Example 5.11 The dl-program KB_2 in Example 4.2 is stratified (intuitively, the recursion through negation is acyclic) while KB_3 in Example 4.3 is not. The result computed in Example 4.9 verifies the conditions of Theorem 5.10.

We finally show that we can limit ourselves to dl-programs in *dl-query form*, where dl-atoms equate designated predicates. Formally, a dl-program $KB = (L, P)$ is in *dl-query form*, if each $r \in P$ involving a dl-atom is of the form $a \leftarrow b$, where b is a dl-atom. Any dl-program $KB = (L, P)$ can be transformed into a dl-program $KB^{dl} = (L, P^{dl})$ in dl-query form. Here, P^{dl} is obtained from P by replacing every dl-atom $a(\mathbf{t}) = DL[\lambda; Q](\mathbf{t})$ by $p_a(\mathbf{t})$, and by adding the dl-rule $p_a(\mathbf{X}) \leftarrow a(\mathbf{X})$ to P, where p_a is a new predicate symbol, and $\mathbf{X}$ is a list of variables corresponding to $\mathbf{t}$. Informally, p_a is an abbreviation for a.

The following result now shows that KB^{dl} and KB are equivalent under the well-founded semantics. Intuitively, this means that the well-founded semantics tolerates abbreviations in the sense that they do not change the semantics of a dl-program. This normal form is particularly useful for the computation of the well-founded semantics, as it allows to eliminate dl-atoms from arbitrary rules and to move them to special rules. Another good property is that the transformation to normal form preserves stratification.

Theorem 5.12 *Let $KB = (L, P)$ be a dl-program. Then, $WFS(KB) = WFS(KB^{dl}) \cap Lit_P$.*

Table 1: Complexity of literal entailment from dl-programs $KB = (L, P)$ under the well-founded semantics.

	L in $\mathcal{SHIF}(\mathbf{D})$	L in $\mathcal{SHOIN}(\mathbf{D})$
General Complexity	EXP-complete	$\mathrm{P}^{\mathrm{NEXP}}$-complete
Data Complexity	P^{NP}-complete	P^{NP}-complete

We close this section with a brief comment on dl-programs with nonmonotonic dl-atoms [Eiter et al. 2008]. The latter also have the form (2), but op_i may be $\not\cap$, where $S_i \not\cap p_i$ increases $\neg S_i$ by the *complement of* p_i. This is equivalent to $S_i \uplus \overline{p}_i$, given that $\overline{p}_i$ is the complement of p_i, which is expressible with a rule $\overline{p}_i(\mathbf{X}) \leftarrow \mathit{not}\ DL[S_i' \uplus p_i; S_i'](\mathbf{X})$, where S_i' is a fresh concept resp. role name, provided that the DL knowledge base is satisfiable. In this way, any dl-program $KB = (L, P)$ with satisfiable L can be rewritten to the premier fragment that we consider here; for unsatisfiable L, the rewriting is also usable (though $\overline{p}_i$ may not be the complement of p_i).

6 Computation and Complexity

In this section, we show how the well-founded semantics of dl-programs KB can be computed by finite sequences of finite fixpoint iterations, using the operator γ_{KB} and the immediate consequence operator T_{KB} of positive dl-programs KB. We also analyze the general and the data complexity of reasoning from dl-programs under the well-founded semantics. Our complexity results are compactly summarized in Table 1. In detail, deciding literal entailment from a dl-program $KB = (L, P)$ with L in $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) under the well-founded semantics is complete for EXP (resp., $\mathrm{P}^{\mathrm{NEXP}}$) in general, and complete for P^{NP} (for both DLs) under data complexity. P^{NP}-complete for $\mathcal{SHIF}(\mathbf{D})$ under data complexity (for $\mathcal{SHOIN}(\mathbf{D})$, the same is believed). In fact, the P^{NP} upper bound for data complexity extends to all description logics $\mathcal{L}$ for which literal inference $I \models_L a$ is decidable in polynomial time with an NP oracle under data complexity.

6.1 Fixpoint Iteration

The well-founded semantics of dl-programs KB can be computed by two finite fixpoint iterations, via the operator γ_{KB}, using in turn finite fixpoint iterations for computing the least models of positive dl-programs, via their immediate consequence operator.

More concretely, for any positive dl-program $KB = (L, P)$, the least model of KB, denoted M_{KB}, coincides with the least fixpoint of the immediate consequence operator T_{KB} [Eiter et al. 2004], which is defined as follows for every $I \subseteq HB_P$:

$$T_{KB}(I) = \{H(r) \mid r \in \mathit{ground}(P),\ I \models_L \ell \text{ for all } \ell \in B(r)\}\,.$$

To compute the well-founded semantics of a normal dl-program $KB = (L, P)$, that is, $WFS(KB) = \mathit{lfp}(\gamma_{KB}^2) \cup \neg(HB_P - \mathit{gfp}(\gamma_{KB}^2))$, we compute the least and the greatest fixpoint of γ_{KB}^2 as the limits of the

two fixpoint iterations

$$\mathit{lfp}(\gamma^2_{KB}) = U_\infty = \bigcup_{i \geqslant 0} U_i, \text{ where } U_0 = \emptyset, \text{ and } U_{i+1} = \gamma^2_{KB}(U_i), \text{ for } i \geqslant 0, \text{ and}$$
$$\mathit{gfp}(\gamma^2_{KB}) = O_\infty = \bigcap_{i \geqslant 0} O_i, \text{where } O_0 = HB_P, \text{ and } O_{i+1} = \gamma^2_{KB}(O_i), \text{ for } i \geqslant 0,$$

respectively, which are both reached within $|HB_P|$ many steps. Recall that the operator γ_{KB} is defined by $\gamma_{KB}(I) = M_{KB^I}$ (with $KB^I = (L, sP^I_L)$), for all $I \subseteq HB_P$. As argued above, M_{KB^I} coincides with $\mathit{lfp}(T_{KB^I})$, for all $I \subseteq HB_P$. To compute $\gamma_{KB}(I)$, for all $I \subseteq HB_P$, we thus compute the least fixpoint of T_{KB^I} as the limit of the fixpoint iteration

$$\mathit{lfp}(T_{KB^I}) = S_\infty = \bigcup_{i \geqslant 0} S_i, \text{ where } S_0 = \emptyset, \text{ and } S_{i+1} = T_{KB^I}(S_i), \text{ for } i \geqslant 0,$$

which is also reached within $|HB_P|$ many steps.

6.2 General Complexity

We recall that for a given ordinary normal program, computing the well-founded model needs exponential time in general (measured in the program size [Dantsin et al. 2001]), and also reasoning from the well-founded model has exponential time complexity. Furthermore, evaluating a ground dl-atom a of the form (2) for $KB = (L, P)$ given an interpretation I_p of its input predicates $p = p_1, \ldots, p_m$ (that is, deciding whether $I \models_L a$ holds for each I that coincides on p with I_p) is complete for EXP (resp., co-NEXP) for L in $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) [Eiter et al. 2004], where EXP (resp., NEXP) denotes exponential (resp., nondeterministic exponential) time; this is inherited from the complexity of deciding whether a knowledge base in $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) is satisfiable [Tobies 2001; Horrocks and Patel-Schneider 2004].

The following result shows that computing the well-founded semantics of a dl-program $KB = (L, P)$ over $\mathcal{SHIF}(\mathbf{D})$ can be done in exponential time, and that reasoning from such programs under the well-founded semantics is EXP-complete; hardness holds even when L is empty or P contains only one rule. That is, the complexity of the well-founded semantics for such programs does not increase over the one of ordinary normal programs. The membership part follows from the above fixpoint characterization of the well-founded semantics of dl-programs and the EXP-membership of deciding $I \models_L a$ for L in $\mathcal{SHIF}(\mathbf{D})$, while the hardness part follows from the EXP-hardness of reasoning from the well-founded semantics of ordinary normal programs as well as the EXP-hardness of deciding knowledge base satisfiability in $\mathcal{SHIF}(\mathbf{D})$.

Theorem 6.1 *Given a vocabulary Φ and a dl-program $KB = (L, P)$ with L in $\mathcal{SHIF}(\mathbf{D})$, computing $WFS(KB)$ is feasible in exponential time. Furthermore, given additionally a literal $l \in Lit_P$, deciding whether $l \in WFS(KB)$ holds is* EXP*-complete. Hardness holds even in the cases where (a) L is empty or (b) P contains only one rule.*

For dl-programs over $\mathcal{SHOIN}(\mathbf{D})$, the computation of the well-founded semantics and reasoning from it is expected to be more complex than for dl-programs over $\mathcal{SHIF}(\mathbf{D})$, since already evaluating a single dl-atom is co-NEXP-hard. Computing the well-founded semantics can be done, in a similar manner as in the case of $\mathcal{SHIF}(\mathbf{D})$, in exponential time using an oracle for evaluating dl-atoms; to this end, an NP oracle is sufficient. As for the reasoning problem, this means that deciding whether $l \in WFS(KB)$ holds is in $\mathrm{EXP}^{\mathrm{NP}}$. A more precise account reveals the following strict characterization of the complexity, showing that reasoning from dl-programs $KB = (L, P)$ over $\mathcal{SHOIN}(\mathbf{D})$ under the well-founded semantics

is complete for $\mathrm{P}^{\mathrm{NEXP}}$, which is intuitively strictly contained in $\mathrm{EXP}^{\mathrm{NP}}$,[8] and hardness holds even when P is stratified. The membership part follows from the above fixpoint characterization of the well-founded semantics of dl-programs and the co-NEXP-membership of deciding $I \models_L a$ for L in $\mathcal{SHOIN}(\mathbf{D})$, using a census technique, which essentially allows to evaluate all dl-atoms in advance in polynomial time with an oracle for NEXP, while the hardness part follows from the $\mathrm{P}^{\mathrm{NEXP}}$-hardness of strong answer set existence for stratified dl-programs [Eiter et al. 2004].

Theorem 6.2 *Given a vocabulary Φ, a dl-program $KB = (L, P)$ with L in $\mathcal{SHOIN}(\mathbf{D})$, and a literal $l \in Lit_P$, deciding whether $l \in WFS(KB)$ holds is $\mathrm{P}^{\mathrm{NEXP}}$-complete. Hardness holds even in the case where P is stratified.*

The results in Theorems 6.1 and 6.2 also show that, like for ordinary normal programs, inference under the well-founded semantics is computationally less complex than under the answer set semantics for dl-programs (L, P) with L from $\mathcal{SHIF}(\mathbf{D})$, as cautious reasoning from the strong answer sets such a dl-programs is complete for co-NEXP; with L from $\mathcal{SHOIN}(\mathbf{D})$, the complexity is the same. [Eiter et al. 2004;2008].

Analog complexity results for literal inference under the well-founded semantics can be derived for L from other DLs; for the upcoming OWL2 proposal, an adjusted proof of Theorem 6.2 shows that the problem is in $\mathrm{P}^{2\mathrm{NEXP}}$ (and presumably also complete for this class), and for the OWL2 profiles EL, QL, and RL, an adjusted proof of Theorem 6.1 that it is EXP-complete. This is because deciding $I \models_L a$ for L in the DL $\mathcal{SROIQ}$ underlying OWL2 is co-2NEXP-complete, as follows from [Kazakov 2008], and for L in EL, QL, and RL is polynomial. [9]

6.3 Data Complexity

We now explore the data complexity of reasoning from dl-programs $KB = (L, P)$ under the well-founded semantics. Here, only the constant symbols in the vocabulary Φ, the concept and role membership axioms in L, and the facts in P may vary, while the rest of Φ, L, and P is fixed. The following result, which follows from the above fixpoint characterization of the well-founded semantics of dl-programs, shows that the data complexity of dl-programs does not increase much compared to the one of query answering in the description logic where L is from.[10]

Proposition 6.3 *Given a vocabulary Φ, a dl-program $KB = (L, P)$ with L from a description logic $\mathcal{L}$ for which deciding $I \models_L a$ has data complexity in class $\mathcal{C}$, and a literal $l \in Lit_P$, deciding whether $l \in WFS(KB)$ holds is in $\mathrm{P}^{\mathcal{C}}$ under data complexity.*

Exploiting this, we derive that for both $\mathcal{L} = \mathcal{SHIF}(\mathbf{D})$ and $\mathcal{L} = \mathcal{SHOIN}(\mathbf{D})$ the problem is P^{NP}-complete under data complexity; hardness holds even when L is in $\mathcal{ALE}$ and P is stratified. Indeed, unsatisfiability and instance checking in $\mathcal{SHOIN}(\mathbf{D})$ (and $\mathcal{SROIQ}(\mathbf{D})$) are in co-NP under data complexity (which follows from results in [Pratt-Hartmann 2008]); the hardness part is shown by a generic reduction

[8] In $\mathrm{EXP}^{\mathrm{NP}}$, a NEXP oracle can be emulated, and computation trees with branching on the (emulated) oracle answers can have double exponentially many paths and exponential depth; intuitively, finding the correct computation path in such a tree needs exponentially many NEXP oracle calls. Still $\mathrm{P}^{\mathrm{NEXP}} = \mathrm{EXP}^{\mathrm{NP}}$ is possible, e.g., if NEXP = EXP and NP = P.

[9] As follows from `http://www.w3.org/TR/2008/WD-owl2-profiles-20081202/`.

[10] Note that a slightly modified construction can be used to derive the data complexity of deciding consistency and of cautious/brave reasoning under strong/weak answer sets.

from Turing machines, exploiting the co-NP-hardness proof for instance checking in $\mathcal{ALE}$ by Donini et al. [1994].

Theorem 6.4 *Given a vocabulary Φ, a dl-program $KB = (L, P)$ with L in $\mathcal{SHIF}(\mathbf{D})$ and a literal $l \in Lit_P$, deciding whether $l \in WFS(KB)$ holds is P^{NP}-complete under data complexity. Hardness holds even in the case where (i) L is in $\mathcal{ALE}$ and (ii) P is stratified.*

7 Data Tractability

We now delineate special cases where reasoning from dl-programs under the well-founded semantics can be done in polynomial time and in LOGSPACE in the data complexity.

7.1 Polynomial Case

We first focus on the case where the evaluation of all dl-atoms in a dl-program can be done in polynomial time. In this case, reasoning from dl-programs under the well-founded semantics is complete for P under data complexity, and thus has the same data complexity as reasoning from ordinary normal programs under the well-founded semantics. This result is formally expressed by the following theorem, whose membership part follows immediately from Proposition 6.3 while the hardness part follows from the P-completeness of reasoning from the well-founded semantics of ordinary normal programs.

Theorem 7.1 *Given a vocabulary Φ, a dl-program $KB = (L, P)$, and a literal $l \in Lit_P$, where every dl-atom in P can be evaluated in polynomial time, deciding whether $l \in WFS(KB)$ is complete for* P *under data complexity.*

Since there is a current trend towards highly scalable query answering and reasoning over ontologies, there are many recent DLs that allow for evaluating dl-atoms in polynomial time. Among the most expressive ones is Horn-$\mathcal{SHIQ}$ [Hustadt et al. 2005], which is a fragment of the description logic behind OWL Lite, and which allows for reasoning and conjunctive query answering in polynomial time under data complexity [Eiter et al. 2008]. The following theorem shows that reasoning from dl-programs $KB = (L, P)$ under the well-founded semantics, where L is defined in Horn-$\mathcal{SHIQ}$, has the same data complexity as in the ordinary case, when all concepts in dl-queries in P are atomic.

Theorem 7.2 *Given a vocabulary Φ, a dl-program $KB = (L, P)$, and a literal $l \in Lit_P$, where (i) L is defined in Horn-$\mathcal{SHIQ}$, and (ii) all concepts C and D in dl-queries of one of the forms among $C \sqsubseteq D$, $\neg(C \sqsubseteq D)$, $C(t)$, and $\neg C(t)$ in P are atomic (including $\bot$ and $\top$), deciding whether $l \in WFS(KB)$ is complete for* P *under data complexity.*

Similarly, under data complexity, literal inference under the well-founded semantics is P-complete for dl-programs over knowledge bases in the OWL2 profiles EL, QL, and RL.

7.2 First-Order Rewritable Case

We next consider the case where the evaluation of every dl-query in a dl-program $KB = (L, P)$ is first-order rewritable. In this case, if we make additional acyclicity assumptions about P, then reasoning from dl-programs under the well-founded semantics is also first-order rewritable, which implies that reasoning from dl-programs under the well-founded semantics can be done in LOGSPACE under data complexity.

Here, a dl-query $Q(\mathbf{t})$ over L is *first-order rewritable* iff it can be expressed in terms of a first-order formula $\phi(\mathbf{t})$ over the set L_{CR} of all concept and role membership axioms in L, that is, for every $\mathbf{c}$, it holds that $L \models Q(\mathbf{c})$ iff $I_{L_{CR}} \models \phi(\mathbf{c})$, where for any set of atoms F, we denote by I_F the total Herbrand interpretation that satisfies exactly the atoms in F (i.e., under the closed world assumption on F).[11] The dl-program KB is *first-order rewritable* iff the extension of every predicate $p(\mathbf{x})$ in $WFS(KB)$ can be expressed in terms of a first-order formula $\phi(\mathbf{x})$ over the set F of all concept and role membership axioms in L and all database facts in P, that is, for every $\mathbf{c}$, it holds that $p(\mathbf{c}) \in WFS(KB)$ iff $I_F \models \phi(\mathbf{c})$. Informally, such dl-atoms and predicates can be expressed in terms of SQL queries over a relational database. The notion of acyclicity for dl-programs assures that they are first-order rewritable when all dl-atoms are so. It is defined as follows. Let $\mathcal{P}_P$ denote the set of all predicate symbols in P. We say $KB = (L, P)$ is *acyclic* iff a mapping $\kappa\colon \mathcal{P}_P \to \{0, 1, \ldots, n\}$ exists such that for every $r \in P$, the predicate symbol p of $H(r)$, and every predicate symbol q of some ordinary $b \in B(r)$ or of an input argument of some dl-atom $b \in B(r)$, it holds $\kappa(p) > \kappa(q)$.

The following result shows that reasoning from acyclic dl-programs $KB = (L, P)$ under the well-founded semantics is first-order rewritable (and thus can be done in LOGSPACE under data complexity), when (i) all dl-queries in P are first-order rewritable, and (ii) if the operator $\uplus$ occurs in P, then L is defined over a description logic that (ii.a) is *CWA-satisfiable* (that is, for every description logic knowledge base L', the union of L' and all negations of concept and role membership axioms that are not entailed by L' is satisfiable) and (ii.b) allows for first-order rewritable concept and role memberships.

Theorem 7.3 *Given a vocabulary Φ, an acyclic dl-program $KB = (L, P)$, and a literal $l \in Lit_P$, where (i) every dl-query in P is first-order rewritable, and (ii) if the operator $\uplus$ occurs in P, then L is defined over a description logic that (ii.a) is CWA-satisfiable, and (ii.b) allows for first-order rewritable concept and role memberships, deciding whether $l \in WFS(KB)$ is first-order rewritable.*

In particular, reasoning from acyclic dl-programs $KB = (L, P)$ under the well-founded semantics is first-order rewritable (and thus can be done in LOGSPACE under data complexity), when (i) L is defined in a description logic of the *DL-Lite* family [Calvanese et al. 2007] (in which knowledge base satisfiability and conjunctive queries are both first-order rewritable) and (ii) we assume suitable restrictions on dl-queries in P.

Theorem 7.4 *Given a vocabulary Φ, an acyclic dl-program $KB = (L, P)$, and a literal $l \in Lit_P$, where (i) L is defined in a description logic of the DL-Lite family, and (ii) all dl-queries in P are of one of the forms $C \sqsubseteq D$, $\neg(C \sqsubseteq D)$, $C(t)$, and $R(t, s)$, where C is an atomic concept, and D is an atomic or a negated atomic concept, deciding whether $l \in WFS(KB)$ is first-order rewritable.*

Finally, we remark that the LOGSPACE feasibility generalizes from first-order rewritable dl-atoms to one that can be evaluated in LOGSPACE, but omit further details.

8 Implementation

Based on the ideas of Section 6, we developed an experimental system for computing the well-founded semantics of a given dl-program $KB = (L, P)$. It consists of three separate modules: the answer set solver

[11] Note that the notion of first-order rewritability here does not mean that every knowledge base L in a description logic $\mathcal{L}$ can be expressed as an equivalent first-order theory (which holds for most description logics). Note also that the first-order rewritability here corresponds to the first-order reducibility in [Calvanese et al. 2007].

DLV [Leone et al. 2006], the description logic reasoner RACER [Haarslev and Möller 2001], and a module W that computes $WFS(KB)$ by accessing DLV and RACER.

In a first step, a program P_d is computed from P by replacing every dl-atom $DL[\lambda; Q](\mathbf{t})$ by an atom $p_{DL[\lambda;Q]}(\mathbf{t})$, where $p_{DL[\lambda;Q]}$ is a fresh predicate. The program P_d is then grounded using the grounding module of the DLV system. For that, optimizations performed by that module are properly disabled (otherwise, the result may not be sound for our purposes). After appropriately reintroducing the dl-atoms in the obtained program $grd(P_d)$, the resulting program $P'' = grd(P_d)'$ is returned to the module W, which then computes $\mathit{lfp}(\gamma^2_{(L,P'')})$ and $\mathit{gfp}(\gamma^2_{(L,P'')})$ as defined in Section 6.1. Whenever the truth value of a given dl-atom has to be determined, W invokes the RACER system; the latter performs reasoning on L and variants thereof.

It is worth mentioning that the RACER module has been embedded within a caching module that shortcuts multiple (time consuming) similar queries; e.g., the truth value of $DL[\lambda; C](a)$ can be quickly established if $DL[C](a)$ is true and this information is cached; dually, if $DL[\lambda; C](a)$ is cached as false, subsequent queries $DL[C](a)$ can be answered by a quick cache lookup.

The module W is also exploited for computing the answer set semantics of KB. In virtue of Theorem 5.7, one can indeed, provided KB is consistent, compute $WFS(KB)$ and exploit this information for constraining atoms in $\mathit{lfp}(\gamma^2_{(L,P_d)})$ as true in any answer set, while atoms $\mathit{gfp}(\gamma^2_{(L,P_d)})$ can be constrained to not appear in any answer set. One can exploit *constraints* (i.e., rules with empty head) in DLV programs for this, which allow to prune models. An intermediate ordinary program P' obtained from P can be enriched with the constraint $\leftarrow not\ a$ for any atom a such that $a \in WFS(KB)$, and with a constraint $\leftarrow a$ for any atom a such that $\neg a \in WFS(KB)$. Notice that such constraints may also be added only for a subset of $WFS(KB)$ (e.g., the subset obtained after some steps in the least/greatest fixpoint iteration of γ^2_{KB}). This technique proves to be useful for helping the answer-set programming solver to converge to solutions faster.

The prototype system[12] in fact supports both the answer set semantics and the well-founded semantics of dl-programs. More details about the architecture and the algorithms, as well as optimization techniques, can be found in [Eiter et al. 2005; Schindlauer 2006; Eiter et al. 2008].

9 Related Work

In this section, we discuss related work on combining rules and ontologies, and also consider related work on logic programs with aggregates.

9.1 Combinations of Rules and Ontologies

A number of proposals to integrate rules and ontologies have been made in the last years; we refer to [Eiter et al. 2008; Drabent et al. 2009; Rosati 2006; Motik and Rosati 2007a] which also give (slightly different) taxonomies to distinguish different types of combinations, for recent surveys.

As for this paper, we confine our discussion to combinations of rules and ontologies, where ontologies are expressed in description logics, and rules and ontologies are combined into a native formalism rather than a common fragment (like DLPs [Grosof et al. 2003]), or one where rules have an inherent classical (implicational) semantics, like in SWRL [Horrocks et al. 2004] and its fragments (e.g., [Motik et al. 2005; Krötzsch et al. 2008]). Moreover, we focus on important approaches from the perspective of well-founded semantics.

[12] `https://www.mat.unical.it/ianni/swlp/`

Donini et al. [1998] combined Datalog with the DL $\mathcal{ALC}$ into $\mathcal{AL}$-log. A rule may have atoms $C(X)$ where C is a concept in the body, which act as "constraints"; the variable X must however also occur in an ordinary body atom (*DL-safety*). Levy and Rousset [1998] similarly combined Horn rules with the DL $\mathcal{ALCNR}$ into CARIN, allowing also role atoms $R(X, Y)$ in rule bodies; this leads to undecidability already in very simple settings.

Rosati's $\mathcal{DL}$+log [2006] distinguishes *DL* and *Datalog predicates*; DL and Datalog atoms may occur in both the head and the body of a rule, but *not*is restricted to Datalog atoms; in addition, each occurring variable X must occur in some unnegated atom in the body; the latter must be a Datalog atom, if X occurs in a dl-atom in the head (*weak DL-safety*). Rosati defined an answer set (stable model) semantics for a KB $(\mathcal{T}, P)$ in a two step process by handling first the classical part $\mathcal{T}$ and then the rules P (see Section 9.1.1 and [Rosati 2006] for details), which faithfully generalizes the semantics of $\mathcal{T}$ and P.

Rosati and Motik's [2007b] hybrid MKNF KBs $(\mathcal{T}, P)$ treat DL and Datalog predicates homogeneously, thus allowing that *not* is applied to dl-atoms. They resort to the logic of Minimal Knowledge and Negation as Failure [Lifschitz 1991], lifting in a sense $\mathcal{DL}$+log KBs to a more general and elegant framework. Two modal operators can change the interpretation of an atom (or a formula in general): $\mathbf{K}\phi$, which intuitively means that ϕ is necessarily known to be true, and $\mathbf{not}\ \phi$, which intuitively means that ϕ is not true, i.e., false in some scenario. Rosati and Motik's semantics of $(\mathcal{T}, P)$, which is based, in S5-style, on (pairs of) sets of possible worlds, captures naturally the answer set semantics of P.

The approaches above give semantics to a hybrid KB in terms of two-valued (classical) models resp. sets of such models in case of MKNF, where *not* is handled similarly as in answer set semantics. They informally give a canonical semantics to programs without recursion through negation, where for ordinary logic programs answer set and well-founded semantics do coincide. Well-founded semantics beyond such programs is a natural and important issue. In the following, we consider two such approaches.

9.1.1 Hybrid Programs

Drabent and Maluszynski [2007] considered *hybrid programs* $(\mathcal{T}, P)$ where $\mathcal{T}$ is an ontology specified as a set of DL axioms (in first-order logic) and P is a normal logic program where the rules may have constraint expressions $C_1, \ldots, C_m$ in the bodies, where each C_i is a disjunctive normal form over literal constraints $p(X)$ and $\neg p(X)$, where p is an ontology predicate. In some sense, hybrid programs can be viewed as a variant of $\mathcal{DL}$+log where the stable model semantics for P is replaced with the well-founded semantics; in fact, as ontology predicates can only occur in rule bodies, hybrid programs are closer in spirit to $\mathcal{AL}$-log [Donini et al. 1998].

The well-founded semantics for hybrid programs $(\mathcal{T}, P)$ is defined by a reduction to ordinary logic programming similar as in $\mathcal{DL}$+log. Given a model M for $\mathcal{T}$, the program P/M consists of all groundings of rules in P that satisfy all constraints in M, from which all constraints are removed. The well-founded model of P w.r.t. M is then the (unique) well-founded model of the ordinary logic program P/M. Semantically, this model approximates the answer sets of P/M according to $\mathcal{DL}$+log. A ground literal a (resp., $\neg a$) is true in the well-founded semantics of $(\mathcal{T}, P)$, if a is true (resp., false) in the well-founded model of P w.r.t. every model of $\mathcal{T}$. Thus, it can be viewed as an approximation of the skeptical answer set semantics of (T, P) in $\mathcal{DL}$+log, similar as the well-founded semantics of dl-programs is an approximation of its answer set semantics.

The declarative semantics has been complemented with an operational semantics for query answering, which is based on an extension of SLD-resolution handling negation and constraints; an implementation of a prototype for Datalog programs with negation, which uses XSB and via a standardized interface a

DL-reasoner (e.g., RacerPro) has been described in [Drabent et al. 2007].

Compared to dl-programs, hybrid programs seem more query-oriented than model-oriented. The restriction that ontology predicates cannot occur in rule heads means that hybrid programs (T, P) allow only a unidirectional flow of information from the ontology $\mathcal{T}$ to the rules P, while dl-programs enable a bidirectional flow of information between the rules and the ontology. Query answering from positive hybrid programs is thus, like for positive ordinary programs, reducible to (un)provability in classical logic; the same holds only for a fragment of the corresponding class of positive dl-programs. On the other hand, hybrid programs share with $\mathcal{DL}{+}log$ the possibility to express reasoning by cases from the ontology via simple rules. For dl-programs, this is not possible, but such reasoning may be shifted to dl-atoms or supported by more expressive dl-atoms like cq-atoms (cf. the example below and the discussion for $\mathcal{DL}{+}log$ in [Eiter et al. 2008]).

Every model M of $\mathcal{T}$ gives rise to a well-founded model of P/M and influences the well-founded consequences. On the other hand, if $\mathcal{T}$ has no model, the inconsistency spreads to the rules and all ground queries are true. For example, if P consists of the rules $q \leftarrow p(a)$, $q \leftarrow \neg p(a)$, and $r \leftarrow not\ q$, where p is an atomic concept, and $\mathcal{T}$ is unsatisfiable, then both r and q are concluded under hybrid programs semantics; however, intuitively one may expect that r is false, as it can never be true regardless of the contents of $\mathcal{T}$. The corresponding dl-program, with reasoning by cases of $p(a)$ expressed by $q \leftarrow DL[p \sqcup \neg p](a)$, would conclude this under the well-founded semantics.

Due to the quantification over the models of $\mathcal{T}$, also a nice model-based interpretation of the well-founded semantics for hybrid programs is non-obvious. Indeed, while our well-founded semantics yields a partial model of the dl-program with a simple totalization (cf. Theorem 5.2), this is not the case for hybrid programs. In connection with this, well-founded semantics of our dl-programs is directly defined on the intuitive and constructive notion of unfounded set (which results in a fixpoint), while an appealing notion of unfounded set for hybrid programs is unclear.

9.1.2 Hybrid MKNF Knowledge Bases under the Well-Founded Semantics

Knorr et al. [2008] gave a well-founded semantics for hybrid MKNF KBs $\mathcal{K} = (\mathcal{T}, P)$ where P amounts to a normal logic program, and more precisely consists of rules of the form

$$\mathbf{K}h \leftarrow \mathbf{K}b_1, \ldots \mathbf{K}b_m, \mathbf{not}\, b_{m+1}, \ldots, \mathbf{not}\, b_n\,,$$

where h and all b_i are atoms. The KB is transformed into the MKNF formula $\pi(\mathcal{K}) = \mathbf{K}\pi(\mathcal{T}) \wedge \pi(P)$, where $\pi(\mathcal{T}) = \bigwedge_{\phi\in\mathcal{T}} \phi$, $\pi(P) = \bigwedge_{r\in P} \pi(r)$ and $\pi(r)$ is the universal closure of r read as material implication (assuming that $\mathcal{T}$ and $\mathcal{P}$ are finite). As we aim here to give the flavor of the approach, we omit for simplicity further technical assumptions (safety of rules, treatment of equality, etc) and give just a superficial description.

Using an S5-style approach, $\mathbf{K}$ and $\mathbf{not}$ are evaluated over sets M of 2-valued (Herbrand) interpretations (or possible worlds) I; in fact, *pairs* $\mathcal{M} = (M_1, M_2)$ of such sets are used for the 3-valued logic, where M_1 serves for truth value $true$ and M_2 for $false$. Three-valued MKNF structures are of the form $(I, \mathcal{M}, \mathcal{M}')$, where $\bigcap M_1 \subseteq \bigcap M_2$ and where $\bigcap M_1' \subseteq \bigcap M_2'$, over which formulas are inductively evaluated, where $\mathcal{M}$ is used for $\mathbf{K}\phi$ and $\mathcal{M}'$ for $\mathbf{not}\,\phi$. A pair $\mathcal{M} = (M_1, M_2)$, where $M_1 \supseteq M_2$, satisfies a closed MKNF formula ϕ, if ϕ evaluates to $true$ on $(I, \mathcal{M}, \mathcal{M})$ for each $I \in M_1$; $\mathcal{M}$ is a *(partial) MKNF model* of ϕ, if in addition for every $\mathcal{M}' = (M_1', M_2')$ such that $M_i' \supseteq M_i$ for $i = 1, 2$ and one of the inclusions is proper, ϕ does not evaluate to $true$ on $(I, \mathcal{M}', \mathcal{M})$ for some $I' \in M_1'$.

In order to select a particular MKNF model $wf(\mathcal{K})$ of $\pi(\mathcal{K})$ as the semantics of $\mathcal{K}$, Knorr et al. single out subsets $\mathbf{P}_\mathcal{K}$ and $\mathbf{N}_\mathcal{K}$ of the set $KA(\mathcal{K})$ of all ground atoms $\mathbf{K}\xi$ such that either $\mathbf{K}\xi$ or $\mathbf{not}\,\xi$ occurs in the grounding of P; intuitively, $\mathbf{P}_\mathcal{K}$ and $\mathbf{N}_\mathcal{K}$ state ground atoms that evaluate to $true$ respectively $false$. They are obtained by an alternating fixpoint construction, similar to the one for ordinary logic programs (cf. Section 5), using a monotonic consequence operator $T_{\mathcal{K}'}$ for $\mathbf{not}$-free ("positive") KBs $\mathcal{K}'$ on subsets of $KA(\mathcal{K}')$ (in fact, $\mathcal{K}$ is slightly rewritten in order to correlate falsity of atoms ξ in the first-order part $\mathcal{T}$ to $\mathbf{not}\xi$ in P); general KBs are reduced to positive KBs using a Gelfond-Lifschitz style reduction.

Most of the properties of the traditional well-founded model are preserved; given that the first-order part $\mathcal{T}$ is consistent, $wf(\mathcal{K})$ is unique, and moreover, it is the least MKNF model of $\mathcal{K}$ according to a natural knowledge ordering. If $wf(\mathcal{K})$ is total, i.e., of form (M, M), then it coincides with the unique MKNF model of $\mathcal{K}$ as in [Motik and Rosati 2007b]; if $\mathcal{T} = \emptyset$, then $wf(\mathcal{K})$ corresponds to the well-founded model of P. Furthermore, computing $wf(\mathcal{K})$ is polynomial in data complexity if entailment in the DL underlying $\mathcal{T}$ has such complexity.

The approach of Knorr et al. bears some similarity to ours as it builds on a monotonic consequence operator. However, the alternating fixpoint construction has a strong technical flavor and may be less persuasive than a construction that works from first principles with unfounded sets. Similar as with hybrid programs above, $wf(\mathcal{K})$ may not exist if $\mathcal{T}$ itself or its interaction with the rules part P is not consistent. The latter can be detected in the iterated fixpoint construction, while inconsistency of $\mathcal{T}$ is not expressible at the object level of the semantics; in dl-programs, this is trivial (use e.g. a rule $incons \leftarrow DL[\top \sqsubseteq \bot]()$) and can be exploited for expressing paraconsistent behavior. Finally, the interfacing approach of dl-programs makes them more amenable for incorporating variants of entailment from the ontology and (possibly heterogenous) other knowledge bases, which seems more difficult for the tight integration realized by the hybrid MKNF approach.

We remark that several other interesting formalisms have potential for combining logic programs and DLs under the well-founded semantics. Among these are FO(ID) logic [Denecker and Vennekens 2007], which extends first-order logic with inductive definitions, quantified equilibrium logic [de Bruijn et al. 2007], and first-order autoepistemic logic [de Bruijn et al. 2007]; for the latter two, a well-founded semantics remains to be developed.

9.2 Logic Programming with Aggregates

Our dl-programs are related to extensions of logic programs with aggregates, for which also a well-founded semantics has been developed independently, e.g., [Calimeri et al. 2005; Pelov et al. 2007]. Such programs allow aggregate atoms in rule bodies, which in [Calimeri et al. 2005] are roughly of the form $f(S)\,\theta\,k$, where $f(S)$ is an aggregate function f such as $\min$, $\max$, sum, or count, applied to a set of elements S that is specified using a conjunction of ordinary atoms, θ is a comparison operator, and k a value. An example is $\#\mathrm{count}\{X : h(X), p(X, a)\} < 2$, which evaluates to true if less than two ground values for X satisfy the given conjunction. Pelov et al. [2007] considered a notion of aggregate where f and θ are abstracted to aggregate functions and aggregate relations.

Intuitively, aggregate atoms work similarly as dl-atoms over some given input from the program, even though the underlying evaluation domain is completely different. Noticeably, Calimeri et al. [2005] defined a well-founded semantics of non-monotonic logic programs P with aggregates (assuming each is either monotone or anti-monotone) based on a notion of unfounded set, in the usual way [van Gelder et al. 1991]. According to their definition, a set of (ordinary) ground atoms X is unfounded w.r.t. a given (partial) interpretation I, if for each rule r from the grounding of P that has some atom from X in the head, either

(a) some anti-monotone literal in the body of r is false w.r.t. I, or (b) some monotone body literal of r is false w.r.t. $(I - X) \cup \neg.X$; here, falsity of an aggregate atom in a partial interpretation amounts to falsity in all its totalizations. The condition (a) corresponds to our conditions (ii) and (iv) in Definition 4.1, while (b) corresponds to (i) and (iii). Note that the two notions of unfoundedness coincide if $I \cap X = \emptyset$. This is the relevant case for $WFS(KB)$, as in the least fixpoint-construction of W_{KB}, $U_{KB}(I)$ and I (which is contained in $T_{KB}(I)$) will be always disjoint. Thus, Calimeri et al.'s notion of unfounded set results in the same well-founded semantics as our notion.

The notion of unfounded set was extended later by Faber [2005] to arbitrary aggregates, by changing (a) and (b) to falsity of some literal in the body of r w.r.t. I and w.r.t. $(I - X) \cup \neg.X$, respectively. To accommodate non-monotonic dl-atoms like those in [Eiter et al. 2004;2008], we can to the same effect change (iv) in Definition 4.1 to (iv′) for some dl-atom $b \in B^-(r)$, $S^+ \models_L b$ for every consistent $S \subseteq Lit_P$ with $I \cup \neg.U \subseteq S$, and generalize (b) of $T_{KB}(I)$ to (b′) $S^+ \models_L b$, for all consistent $S \subseteq Lit_P$ with $I \subseteq S$ and all $b \in B^+(r)$. The properties in Section 5 then naturally carry over to the extended setting (where strong answer sets do not allow non-monotonic dl-atoms in positive rule bodies).

On the other hand, Pelov et al. defined well-founded semantics for logic programs with aggregates on a purely algebraic basis without unfounded sets, using operators on bilattices in the theory of approximating operators [Denecker et al. 2004]. Studying dl-programs and their properties in an analog framework would be an interesting issue for further research.

10 Conclusion

In this paper, we presented a well-founded semantics for non-monotonic dl-programs [Eiter et al. 2004;2008], which combine logic programs and description logic knowledge bases in a loose coupling by an interfacing approach. The semantics faithfully generalizes the canonical well-founded semantics for ordinary normal logic programs [van Gelder et al. 1991], and is, like the latter, defined via greatest unfounded sets for dl-programs. The proposal is distinct from other proposals of well-founded semantics for combinations of rules and description logics, such as [Drabent and Maluszynski 2007] and [Knorr et al. 2008], which provide a heterogenous but tight integration and a homogenous integration, respectively, and which are not based on unfounded sets. By its nature, it is amenable to realize non-monotonic rules over ontologies by combining existing reasoning engines which may be modularly replaced.

As we have shown, the proposed semantics retains a number of properties of the well-founded semantics for ordinary logic programs in the generalized context, including an equivalent characterization in terms of a generalized Gelfond-Lifschitz transform, and that the well-founded semantics is a partial model that approximates the (strong) answer set semantics, while in the positive and stratified case, it is a total model that coincides with the answer set semantics for dl-programs. Furthermore, we provided a complexity analysis, which shows that our proposal also retains the good computational properties of the well-founded semantics. In particular, it is polynomial under data complexity provided that the access to the description logic part is polynomial (as e.g. with the profiles EL, QL, and RL in the upcoming OWL2 standard[13]); depending on the structure of the program and the description logic class, one has even lower complexity and, in case of acyclic programs and *DL-Lite* ontologies, one even achieves first-order rewritability.

There are several directions for further work. One direction is optimization and efficient implementation of the well-founded semantics, but also of restricted fragments like those we considered, in particular the ones where ontology reasoning is first-order expressible. To this end, tightly integrated non-monotonic

[13] `http://www.w3.org/TR/2008/WD-owl2-profiles-20081202/`

logic programming and relational databases engines, like the DLV^{DB} system [Terracina et al. 2008], may be fruitfully exploited for evaluating programs with recursion. On the other hand, top-down evaluation methods for efficient query answering, as well as developing magic sets are intriguing issues.

Another direction are language extensions. The language we considered can be readily extended to use cq-atoms [Eiter et al. 2009], which allow to query the ontology also with conjunctive queries and unions thereof. In contrast, an extension to rules with disjunctive heads seems less straightforward; many proposals for well-founded semantics of disjunctive logic programs exist (see, e.g., [Wang and Zhou 2005] and [Knorr and Hitzler 2007] for discussion), but none is ultimately acknowledged and they have limited significance in practice. An extension to rules with explicit negation [Pereira and Alferes 1992] may be targeted, which then also may use three-valued dl-atoms, in line with the underlying logic.

Finally, an interesting direction would be to establish a similar formalism over multiple ontologies, possible even in heterogeneous formats (e.g., RDF and OWL).

Acknowledgments

We are grateful to Diego Calvanese, Magdalena Ortiz and Ulrike Sattler for providing valuable information on complexity-related issues about OWL-DL related description logics, and to Włodzimierz Drabent for interesting discussions. We further thank the reviewers of the RuleML-2004 preliminary version of this paper, whose useful and constructive comments have helped to improve this work.

A Appendix: Proofs for Section 4

Proof of Lemma 4.5. Suppose $U_1, U_2 \subseteq HB_P$ are both unfounded sets of KB w.r.t. I. We now show that $(*)$ holds for $U = U_1 \cup U_2$. Consider any $a \in U_1$ and $r \in ground(P)$ with $H(r) = a$. Then, one of (i)-(iv) holds for $U = U_1$, and thus one of (i)-(iv) holds for $U = U_1 \cup U_2$. Similarly, for any $a \in U_2$ and any $r \in ground(P)$ with $H(r) = a$, one of (i)-(iv) holds for $U = U_1 \cup U_2$. In summary, for any $a \in U_1 \cup U_2$ and any $r \in ground(P)$ with $H(r) = a$, one of (i)-(iv) holds for $U = U_1 \cup U_2$. That is, $(*)$ holds for $U = U_1 \cup U_2$. □

Proof of Lemma 4.7. It is sufficient to show that T_{KB} and U_{KB} are monotonic. Let $J_1 \subseteq J_2 \subseteq Lit_P$ be consistent. We first show that T_{KB} is monotonic. If some $r \in ground(P)$ exists such that conditions (a)–(d) in the definition of T_{KB} hold for $I = J_1$, then some $r \in ground(P)$ exists such that (a)–(d) hold for $I = J_2$. That is, $T_{KB}(J_1) \subseteq T_{KB}(J_2)$. We next prove that U_{KB} is monotonic. If $(*)$ holds for $I = J_1$, then $(*)$ holds for $I = J_2$. Hence, every unfounded set of KB w.r.t. J_1 is also an unfounded set of KB w.r.t. J_2. Thus, $U_{KB}(J_1) \subseteq U_{KB}(J_2)$. □

B Appendix: Proofs for Section 5

Proof of Theorem 5.2. Let $KB = (L, P)$. We have to show that there exists some total interpretation $M \supseteq WFS(KB)$ such that M^+ is a model of KB, that is, satisfies all instantiated rules of P.

Let $M = WFS(KB) \cup (HB_P - (WFS(KB) \cup \neg.WFS(KB)))$. That is, M is obtained from $WFS(KB)$ by assigning true to all ground atoms whose value is unknown in $WFS(KB)$. We now show that M^+ is a model of KB.

Each rule in $ground(P)$ such that $H(r) \in M$ is clearly satisfied in M^+. Consider thus any ground rule $r \in ground(P)$ such that $H(r) \notin M$. Then, $\neg.H(r) \in WFS(KB)$ and thus $H(r) \in U_{KB}(WFS(KB))$, and

one of (i)–(iv) in $(*)$ holds for $I = WFS(KB)$ and $U = U_{KB}(WFS(KB))$ there. Note that $I \cup \neg.U = I$. Thus, if (i) or (ii) holds, clearly some literal in $B(r)$ is false in M^+, and hence r is satisfied by M^+. If (iii) holds, then $S^+ \not\models_L b$ for every consistent $S \subseteq Lit_P$ such that $M \subseteq S$. Hence, in particular $M^+ \not\models_L b$, and thus b is false in M^+. Since $b \in B^+(r)$, this means that r is satisfied by M^+. Finally, if (iv) holds, then $WFS(KB)^+ \models_L b$ for some $b \in B^-(r)$. By monotonicity, $M^+ \models_L b$, and thus b is true in M^+. Again, r is satisfied by M^+. Since r was arbitrary, it follows that M^+ is a model of KB. □

Proof of Proposition 5.4. Let $I \subseteq J \subseteq HB_P$. Since every dl-atom in P is monotonic, it holds $sP_L^J \subseteq sP_L^I$. Hence, every model of (L, sP_L^I) is also a model of (L, sP_L^J). Thus, the least model of (L, sP_L^J) is a subset of every model of (L, sP_L^I), and thus in particular also of the least model of (L, sP_L^I). That is, γ_{KB} is anti-monotonic. □

Proof of Theorem 5.5 (sketch). The proof can be carried out by generalizing the proof in [Van Gelder 1989] that the alternating fixpoint partial model coincides with the well-founded partial model. One new aspect is to show that $\gamma_{KB}(I)$ is the set of all atoms $a \in HB_P$ that logically follow from KB and the negated atoms in $\neg.(HB_P - I)$. The operator $\mathbf{S}_P(J)$ on all $J \subseteq \neg.HB_P$ in [Van Gelder 1989] then coincides with $\gamma_{KB}(I)$, where $I = HB_P - \neg.J$. Another new aspect is to show that our notion of unfounded set is complete in the sense that no other atom outside the greatest unfounded set can be assumed false. This corresponds to showing that

$$W^? \subseteq \gamma_{KB}(W^+), \tag{3}$$

where $W = \mathit{lfp}(W_{KB})$ and $W^? = W - (W^+ \cup (\neg.W)^+)$. Roughly, (3) can be proved as follows. It can be shown that $W^+ \subseteq \gamma_{KB}(W^+) \subseteq W^+ \cup W^?$. Towards a contradiction, suppose that $U = W^? - \gamma_{KB}(W^+) \neq \emptyset$. Hence, for every $a \in U$ and every $r \in ground(P)$ with $H(r) = a$, it holds that either (i) $\neg b \in W \cup \neg.U$ for some ordinary atom $b \in B^+(r)$, or (ii) $b \in W$ for some ordinary atom $b \in B^-(r)$, or (iii) for some dl-atom $b \in B^+(r)$, we have that $\gamma_{KB}(W^+) \not\models_L b$, and thus $S^+ \not\models_L b$ for every consistent $S \subseteq Lit_P$ with $W \cup \neg.U \subseteq S$, or (iv) $W^+ \models_L b$ for some dl-atom $b \in B^-(r)$. Hence, U is an unfounded set of KB relative to W. But this contradicts $W = \mathit{lfp}(W_{KB})$. This shows that (3) holds. □

Proof of Theorem 5.7. For any $I \subseteq HB_P$, it holds that I is a strong answer set of KB iff I is a fixpoint of γ_{KB}. Since $\mathit{lfp}(\gamma_{KB}^2) \subseteq I \subseteq \mathit{gfp}(\gamma_{KB}^2)$ for every fixpoint of γ_{KB}, it thus follows that $\mathit{lfp}(\gamma_{KB}^2) \subseteq I \subseteq \mathit{gfp}(\gamma_{KB}^2)$ for every strong answer set I of KB. Thus, every such I includes every well-founded and no unfounded atom $a \in HB_P$ relative to KB. □

Proof of Theorem 5.9. If every $a \in HB_P$ is either well-founded or unfounded relative to KB, then $\mathit{lfp}(\gamma_{KB}^2)$ = $\mathit{gfp}(\gamma_{KB}^2)$. Hence, $\mathit{lfp}(\gamma_{KB}^2) = I = \mathit{gfp}(\gamma_{KB}^2)$, for every fixpoint $I \subseteq HB_P$ of γ_{KB}. That is, $\mathit{lfp}(\gamma_{KB}^2) = I = \mathit{gfp}(\gamma_{KB}^2)$ for every answer set I of KB. That is, the set of all well-founded $a \in HB_P$ relative to KB is the only answer set of KB. □

Proof of Theorem 5.10 (sketch). We use the characterization of $WFS(KB)$ given in Theorem 5.5. Assume first KB is positive. Then, for every $I \subseteq HB_P$, it holds that $s\,P_L^I = P$ and thus $\gamma_{KB}(I)$ is the least model of KB. Thus, the only fixpoint of γ_{KB} (and thus also the least and the greatest fixpoint of γ_{KB}) is the least model of KB, which in turn is the unique answer set of KB. Suppose next KB is stratified. Since $\mathit{lfp}(\gamma_{KB}^2) \subseteq I \subseteq \mathit{gfp}(\gamma_{KB}^2)$ holds for the unique answer set I of KB, it is sufficient to show that neither (a) $\mathit{lfp}(\gamma_{KB}^2) \subset I$ nor (b) $I \subset \mathit{gfp}(\gamma_{KB}^2)$ holds for the unique answer set I of KB. This can be proved by contradiction along a stratification λ of KB. □

Proof of Theorem 5.12. Given $KB = (L, P)$, the corresponding $KB^{dl} = (L, P^{dl})$, and an interpretation I over Lit_P, we will denote I^{dl} as $I \cup \{p_a(\mathbf{c}) \mid I \models_L a(\mathbf{c})$ for each ground dl-atom appearing in $ground(P)\}$. Also, define $G(I) = \gamma_{KB}(I)$ and $G^{dl}(I) = \gamma_{KB^{dl}}(I)$. The proof relies on the following intermediate results.

Lemma B.1 *Let I be any interpretation, and let $J = G^{dl}(I)$. Then, $J = J^{dl}$.*

The above follows from the fact that $p_a(\mathbf{c}) \leftarrow a(\mathbf{c})$ appears in ${sP^{dl}}^{I}_{L}$, for each ground dl-atom appearing in $ground(P)$; so if $J \models_L a(\mathbf{c})$, then we will have $p_a(\mathbf{c}) \in J$.

Lemma B.2 *For every interpretation I over Lit_P, $G(I)^{dl} = G^{dl}(I^{dl})$.*

The above holds since one can observe that sP^I_L and ${sP^{dl}}^{I^{dl}}_{L}$ have the same rules, with the only difference that each (positive) dl-atom $a(\mathbf{c})$ in sP^I_L is replaced with $p_a(\mathbf{c})$ in ${sP^{dl}}^{I^{dl}}_{L}$, and a rule of the form $p_a(\mathbf{c}) \leftarrow a(\mathbf{c})$ is added; one can then easily observe that $G(I)^{dl}$ and $G^{dl}(I^{dl})$ coincide.

Proposition B.3 *$lfp(G^2)^{dl} = lfp((G^{dl})^2)$ and $gfp(G^2)^{dl} = gfp((G^{dl})^2)$.*

Let $I_0 = \emptyset$. One shows first by induction on $k \geqslant 0$ that for the k-th powers of $G(I_0)$ and $G^{dl}(I_0^{dl})$, denoted by $G^k(I_0)$ and $(G^{dl})^k(I_0^{dl})$, we have

$$G^k(I_0)^{dl} = (G^{dl})^k(I_0^{dl}). \tag{4}$$

The equality obviously holds for $k = 0$. Given (4) holds for k, then for $k + 1$, we have

$$G^{k+1}(I_0)^{dl} = (G(G^k(I_0)))^{dl}.$$

Now, let $I = G^k(I_0)$. Then, by Lemma B.2, we have

$$G(I)^{dl} = G^{dl}(I^{dl}),$$

since by the induction hypothesis, $G^k(I_0)^{dl} = (G^{dl})^k(I_0^{dl})$, we get

$$G(G^k(I_0))^{dl} = G^{dl}((G^{dl})^k(I_0^{dl})) = (G^{dl})^{k+1}(I_0^{dl}),$$

which proves (4) for each $k \geqslant 0$. Furthermore, we have that

$$I_0^{dl} \subseteq (G^{dl})^{2k}(I_0), \text{ for each } k \geqslant 0\,. \tag{5}$$

Observe indeed that $G^{dl}(I_0)$ contains I_0^{dl}, as well as $(G^{dl})^2(I_0)$, and that $(G^{dl})^2$ is monotonic. From (5) we conclude that $((G^{dl})^{2k})(I_0^{dl})$ and $((G^{dl})^{2k})(I_0)$ converge to the same limit, which is $lfp((G^{dl})^2)$. On the other hand, $G^{2k}(I_0)^{dl}$ converges to $lfp(G^2)^{dl}$. Thus, we get $lfp(G^2)^{dl} = lfp((G^{dl})^2)$.

In a similar way, one can show that the greatest fixpoints of G^2 and $(G^{dl})^2$ are related: indeed, by letting $I_0 = HB_P$, we have $G^{2k}(HB_P)^{dl} = (G^{dl})^{2k}(HB_P^{dl})$, where $HB_P^{dl} \supseteq (G^{dl})^{2k}(HB_P)$, thus $(G^{dl})^{2k}(HB_P^{dl})$ converges to $gfp((G^{dl})^2)$. □

C Appendix: Proofs for Section 6

Proof of Theorem 6.1. We first show that, given $KB = (L, P)$ and $I \subseteq HB_P$, computing $\gamma_{KB}(I)$ is feasible in exponential time, which then implies that computing $lfp(\gamma^2_{KB})$ and $gfp(\gamma^2_{KB})$ (and thus also $WFS(KB)$) is feasible in exponential time.

The reduct $KB^I = (L, sP^I_L)$ is constructible in exponential time, since (i) $ground(P)$ is computable in exponential time and (ii) $I \models_L a$ for each dl-atom a in $ground(P)$ can be decided in exponential time, by the complexity of deciding knowledge base satisfiability in $\mathcal{SHIF}(\mathbf{D})$. Furthermore, computing the least model of KB^I is feasible in exponential time by computing $lfp(T_{KB^I}) = \bigcup_{i=0}^{n} T^i_{KB^I}(\emptyset)$ with $n = |HB_P|$, which requires at most exponentially many applications of T_{KB^I}, each of which is computable in exponential time (deciding $I \models_L a$ for any dl-atom a in $ground(P)$ is feasible in exponential time, by the complexity of deciding knowledge base satisfiability in $\mathcal{SHIF}(\mathbf{D})$).

Therefore, we can compute $lfp(\gamma^2_{KB}) = U_\infty$, by computing U_0, U_1, ... until $U_i = \gamma^{2i}_{KB}(\emptyset) = \gamma^{2i+2}_{KB}(\emptyset) = U_{i+1}$ holds for some i. Since i is bounded by $|HB_P|$ and the latter is exponential in the size of Φ and KB, the positive part of $WFS(KB)$, that is, $lfp(\gamma^2_{KB})$, is computable in exponential time. The negative part of $WFS(KB)$ is easily obtained from $gfp(\gamma^2_{KB}) = O_\infty$, which can be similarly computed in exponential time. Therefore, computing $WFS(KB)$ is feasible in exponential time.

Hence, deciding whether $l \in WFS(KB)$ holds is in EXP. The EXP-hardness of the problem is immediate from the EXP-hardness of deciding whether a given positive Datalog program logically implies a given ground atom [Dantsin et al. 2001] as well as from the EXP-hardness of deciding whether a knowledge base in $\mathcal{SHIF}(\mathbf{D})$ is satisfiable. □

Proof of Theorem 6.2. For membership in $\mathrm{P}^{\mathrm{NEXP}}$, an algorithm is not allowed to use exponential work space (only polynomial one). Thus, differently from the situation in the proof of Theorem 6.1, we cannot simply compute the powers $\gamma^j_{KB}(\emptyset)$ and $\gamma^j_{KB}(HB_P)$, because $ground(P)$ is exponential. The idea is to move this problem inside an oracle call.

It is easy to see that we can compute $WFS(KB)$ and decide $l \in WFS(KB)$ in exponential time, if the answers for all dl-atom evaluations $I_p \models_L a$ that we encounter during the computation of the powers $\gamma^j_{KB}(\emptyset)$ and $\gamma^j_{KB}(HB_P)$ would be known. However, deciding $I_p \models_L a$ is co-NEXP-complete for a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base L, and thus these answers cannot be computed by a recursive call inside a NEXP oracle call itself. To surmount this problem, we apply a census technique that provides enough information to the oracle for verifying a correct guess for all the answers.

If $I_p \not\models_L a$, then there is an exponential size "proof" witnessing this fact which can be checked in time polynomial in its size. Therefore, given a ground dl-atom a and an integer $k \geqslant 0$, deciding whether there are at least k different inputs $I^1_p, \ldots, I^k_p$ such that $I^j_p \not\models_L a$ is in NEXP. As easily seen, the maximum k for which this holds is given by a number n_a which is exponential in the size of KB and Φ.

In order to decide whether $l \in WFS(KB)$ holds, we can thus proceed as follows:

1. For each ground dl-atom a in $ground(P)$, compute with binary search on $[0, \ldots, n_a]$, using the NEXP oracle, the exact number of inputs I_p such that $I_p \not\models a$, denoted f_a.

2. Ask the oracle whether (a) there are f_a different inputs $I^1_p, \ldots, I^{f_a}_p$ for each dl-atom a such that $I_p \not\models_L a$, and (b) such that for the computation of the powers $\gamma^i_{KB}(\emptyset)$ resp. $\gamma^i(HB_P)$ where for each $I_p \models_L a$ the value compliant with $I^1_p, \ldots, I^{f_a}_p$ is taken, it holds that l is contained in the limit U_∞ of the sequence $\gamma^{2k}(\emptyset)$ if l is a positive literal resp. that b is not contained in the limit O_∞ of the sequence $\gamma^{2k}(HB_P)$ if $l = not\ b$.

3. If the oracle answers yes, return yes, otherwise no.

Note that for the answer "yes", (b) is only relevant if the guess in (a) is correct. Hence, Step 3 correctly decides whether $l \in WFS(KB)$ holds.

Step 1 is feasible in polynomial time modulo the NEXP oracle calls, since the number of ground dl-atoms a in $ground(P)$ is polynomial and the binary search takes $O(\log n_a)$ many steps, which is polynomial in the size of KB and Φ. The oracle query in Step 2 is in NEXP, since for (a) the proper (unique) inputs $I_p^1, \ldots, I_p^{f_a}$ together with their witnesses can be guessed and verified in exponential time, and (b) is feasible in exponential time; In summary, this algorithm correctly decides whether $l \in WFS(KB)$ holds in polynomial time with a NEXP oracle. This proves the membership part.

The $\mathrm{P}^{\mathrm{NEXP}}$-hardness is easily derived from Theorem 5.10 and the result that deciding whether a stratified KB in which classical negation $\neg$ may occur has some strong answer set is $\mathrm{P}^{\mathrm{NEXP}}$-complete [Eiter et al. 2004]. Replace in a stratified KB classical negative literals $\neg p(\mathbf{t})$ by positive literals $\overline{p}(\mathbf{t})$, where $\overline{p}$ is a fresh predicate, and add rules $f \leftarrow p(\mathbf{t}), \overline{p}(\mathbf{t})$, where f is a fresh propositional atom. Then, for the resulting dl-program KB', we have $\neg f \in WFS(KB)$ iff KB has some strong answer set. □

Proof of Proposition 6.3. We show that, for $KB = (L, P)$ where L is in a DL such that evaluating $I \models_L a$ for given $I \subseteq HB_P$ and ground dl-atom a has a data complexity in class $\mathcal{C}$, computing $\gamma_{KB}(I)$ is feasible in polynomial time with a $\mathcal{C}$-oracle in the data complexity. This then implies that computing $lfp(\gamma_{KB}^2)$ and $gfp(\gamma_{KB}^2)$ (and thus also $WFS(KB)$) is feasible in polynomial time with a $\mathcal{C}$-oracle in the data complexity.

The reduct $KB^I = (L, sP_L^I)$ is constructible in polynomial time with a $\mathcal{C}$-oracle, since (i) $ground(P)$ is computable in polynomial time and (ii) $I \models_L a$ for each dl-atom a in $ground(P)$ is decidable using the $\mathcal{C}$-oracle. Furthermore, computing the least model of KB^I is feasible in polynomial time with a $\mathcal{C}$-oracle by computing $lfp(T_{KB^I}) = \bigcup_{i=0}^{n} T_{KB^I}^i(\emptyset)$ with $n = |HB_P|$, which requires at most polynomially many applications of T_{KB^I}, each of which is computable in polynomial time with a $\mathcal{C}$-oracle.

Hence, we can compute $lfp(\gamma_{KB}^2) = U_\infty$, by computing the sets U_0, U_1, ... until $U_i = \gamma_{KB}^{2i}(\emptyset) = \gamma_{KB}^{2i+2}(\emptyset) = U_{i+1}$ holds for some i. As i is polynomially bounded by $|HB_P|$, the positive part of $WFS(KB)$, that is, $lfp(\gamma_{KB}^2)$, is computable in polynomial time with a $\mathcal{C}$-oracle. The negative part of $WFS(KB)$ is easily obtained from $gfp(\gamma_{KB}^2) = O_\infty$, which can be similarly computed in polynomial time with a $\mathcal{C}$-oracle. Therefore, computing $WFS(KB)$ is feasible in polynomial time with a $\mathcal{C}$-oracle in the data complexity, and thus deciding whether $l \in WFS(KB)$ is in $\mathrm{P}^{\mathcal{C}}$ in the data complexity. □

Proof of Theorem 6.4. As for membership in P^{NP}, observe first that instance checking in $\mathcal{SHIF}(\mathbf{D})$ is in co-NP under data complexity. This follows from the results in [Glimm et al. 2008], which showed that the data complexity of answering conjunctive queries in $\mathcal{SHIQ}$ is co-NP-complete, where the knowledge bases are also allowed to contain negated role assertions. Thus, for suitable datatypes, the same data complexity holds for $\mathcal{SHIQ}(\mathbf{D})$. Hence, deciding whether $I \models_L a$ for interpretations I, knowledge bases L in $\mathcal{SHIF}(\mathbf{D})$, and dl-atoms a is clearly in co-NP in the data complexity for a with queries of the form $C(b)$, $\neg C(b)$, $R(b, c)$, $\neg R(b, c)$, $U(b, v)$, and $\neg U(b, v)$. Furthermore, it is also in co-NP in the data complexity for all other types of dl-atoms, since (i) $L' \models C \sqsubseteq D$ iff $L' \cup \{(C \sqcap \neg D)(e),\ A(d)\} \models \neg A(d)$; (ii) $L' \models \neg(C \sqsubseteq D)$ iff $L' \cup \{C \sqsubseteq D,\ A(d)\} \models \neg A(d)$; (iii) $L' \models\ =(b, c)$ iff $L' \cup \{\neq(b, c),\ A(d)\} \models \neg A(d)$; and (iv) $L' \models \neq(b, c)$ iff $L' \cup \{=(b, c),\ A(d)\} \models \neg A(d)$, where d and e are fresh individuals, and A is a fresh atomic concept. The P^{NP}-membership follows then by Proposition 6.3.

Hardness for P^{NP} of literal entailment from a stratified dl-program $KB = (L, P)$ with L in $\mathcal{ALE}$ is proved by a generic reduction from Turing machines M, exploiting the co-NP-hardness proof for instance checking in $\mathcal{ALE}$ by Donini et al. [1994]. Informally, the main idea behind the proof is to use a dl-atom to

decide the result of the j-th oracle call made by a polynomial-time bounded M with access to a NP oracle, where the results of the previous oracle 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 v can then be obtained.

More concretely, let M be a polynomial-time bounded deterministic Turing machine with access to a NP oracle, and let v be an input for M. Since every oracle call can simulate M's computation on v before that call, once the results of all the previous oracle calls are known, we can assume that the input of every oracle call is given by v and the results of all the previous oracle calls. Since M's computation after all oracle calls can be simulated within an additional oracle call, we can assume that the result of the last oracle call is the result of M's computation on v. Finally, since any input to an oracle call can be enlarged by "dummy" bits, we can assume that the inputs to all oracle calls have the same length $n = 2 \cdot (k + l)$, where k is the size of v, and $l = p(k)$ is the number of all oracle calls: We assume that the input to the m+1-th oracle call (with $m \in \{0, \ldots, l-1\}$) has the form

$$v_k\, 1\, v_{k-1}\, 1\, \ldots\, v_1\, 1\, c_0\, 1\, c_1\, 1\, \ldots\, c_{m-1}\, 1\, c_m\, 0\, \ldots\, c_{l-1}\, 0\,,$$

where $v_k, v_{k-1}, \ldots, v_1$ are the symbols of v in reverse order, which are all marked as valid by a subsequent "1", $c_0, c_1, \ldots, c_{m-1}$ are the results of the previous m oracle calls, which are all marked as valid by a subsequent "1", and $c_m, \ldots, c_{l-1}$ are "dummy" bits, which are all marked as invalid by a subsequent "0".

By the co-NP-hardness proof for instance checking in $\mathcal{ALE}$ in [Donini et al. 1994], for the NP oracle M' and any input $b \in \Sigma^*$, there exists a knowledge base $L' \cup L_b$ in $\mathcal{ALE}$, a concept D in $\mathcal{ALE}$, and an individual f such that M' accepts b iff $L' \cup L_b \not\models D(f)$, and L', L_b, D, and f can be constructed in polynomial time from b. More concretely, L', L_b, and D are given as follows:

$$\begin{aligned}
L' &= \{A(\mathit{true}),\, \neg A(\mathit{false})\}\,,\\
L_b &= \{\mathit{Cl}(f, c_1),\, \mathit{Cl}(f, c_2), \ldots, \mathit{Cl}(f, c_n),\\
&\quad\; P_1(c_1, l^1_{1+}), P_2(c_1, l^1_{2+}), N_1(c_1, l^1_{1-}), N_2(c_1, l^1_{2-}), \ldots,\\
&\quad\; P_1(c_n, l^n_{1+}), P_2(c_n, l^n_{2+}), N_1(c_n, l^n_{1-}), N_2(c_n, l^n_{2-})\}\,,\\
D &= \exists \mathit{Cl}.((\exists P_1.\neg A) \sqcap (\exists P_2.\neg A) \sqcap (\exists N_1.A) \sqcap (\exists N_2.A))\,.
\end{aligned}$$

Note that the entailment problem $L' \cup L_b \not\models D(f)$ in $\mathcal{ALE}$ encodes the satisfiability problem for a 2+2-CNF formula $F = C_1 \wedge C_2 \wedge \cdots \wedge C_n$, where $C_i = A^i_{1+} \vee A^i_{2+} \vee \neg A^i_{1-} \vee \neg A^i_{2-}$ and the A^i_j's are propositional symbols including true and false, which has been shown to be NP-hard by a reduction from 3-SAT in [Donini et al. 1994].

Let the stratified dl-program $KB = (L, P)$ now be defined as follows:

$$\begin{aligned}
L &= L'\,,\\
P &= \textstyle\bigcup_{j=0}^{l} P^j\,,
\end{aligned}$$

where $P^j = P^j_v \cup P^j_q \cup P^j_b$ for every $j \in \{0, \ldots, l\}$. Informally, every set of dl-rules P^j generates the input of the j+1-th oracle call, which includes the results of the first j oracle calls. Here, P^l prepares, for simplicity, the input of a "dummy" (non-happening) l+1-th oracle call which contains the result of the l-th (that is, the last) oracle call. More concretely, the bitstring $a_{-2k} \cdots a_{2l-1}$ is the input of the j+1-th oracle call iff $b^j_{-2k}(a_{-2k}), \ldots, b^j_{2l-1}(a_{2l-1})$ are in the canonical model of KB. The components P^j_v, P^j_q, and P^j_b of P^j, with $j \in \{0, \ldots, l\}$, are defined as follows:

1. P_v^0 writes v into the input of the first oracle call, and every P_v^j copies v into the input of the $j{+}1$-th oracle call, for $j \in \{1, \ldots, l\}$:

$$\begin{aligned} P_v^0 &= \{b^0_{-2i}(v_i) \leftarrow \mid i \in \{1, \ldots, k\}\} \cup \{b^0_{-2i+1}(1) \leftarrow \mid i \in \{1, \ldots, k\}\}\,, \\ P_v^j &= \{b^j_{-i}(x) \leftarrow b^{j-1}_{-i}(x) \mid i \in \{1, \ldots, 2k\}\}\,. \end{aligned}$$

2. P_q^0 initializes the rest of the input of the first oracle call with "dummy" bits, and every P_q^j with $j \in \{1, \ldots, l\}$ writes the result of the j-th oracle call into the input of the $j{+}1$-th oracle call and carries over all the other result and dummy bits from the input of the j-th oracle call (where we have $D = \exists Cl.((\exists P_1.\neg A) \sqcap (\exists P_2.\neg A) \sqcap (\exists N_1.A) \sqcap (\exists N_2.A))$):

$$\begin{aligned} P_q^0 &= \{b_i^0(0) \leftarrow \mid i \in \{0, \ldots, 2l{-}1\}\,, \\ P_q^j &= \{b_i^j(x) \leftarrow b_i^{j-1}(x) \mid i \in \{0, \ldots, 2l{-}1\},\ i \notin \{2j{-}2, 2j{-}1\}\} \cup \\ &\quad \{b^j_{2j-2}(0) \leftarrow DL[Cl \uplus cl^{j-1}, P_1 \uplus p_1^{j-1}, P_2 \uplus p_2^{j-1}, N_1 \uplus n_1^{j-1}, N_2 \uplus n_2^{j-1}\,; D](f); \\ &\quad\ b^j_{2j-2}(1) \leftarrow \mathit{not}\, b^j_{2j-2}(0); \\ &\quad\ b^j_{2j-1}(1) \leftarrow \}\,. \end{aligned}$$

3. Every P_b^j with $j \in \{0, \ldots, l\}$ realizes the polynomial-time reduction, which transforms any input b^j of the Turing machine M' into the knowledge base L_{b^j} in $\mathcal{ALE}$, represented as facts over the predicate symbols cl^j, p_1^j, p_2^j, n_1^j, and n_2^j.

Observe then that M accepts v iff the last oracle call returns "yes". The latter is equivalent to the condition that $b^l_{2l-2}(1) \in \mathit{WFS}(KB)$. In summary, M accepts v iff $b^l_{2l-2}(1) \in \mathit{WFS}(KB)$. □

D Appendix: Proofs for Section 7

Proof of Theorem 7.1. Membership in P follows from Proposition 6.3 and the assumption that all dl-atoms can be evaluated in polynomial time, as $\mathrm{P}^{\mathrm{P}} = \mathrm{P}$. Hardness for P follows from the P-completeness of literal inference from ordinary normal programs under the well-founded semantics (cf. [Dantsin et al. 2001]). □

Proof of Theorem 7.2. The statement of the theorem follows from Theorem 7.1 and the result that conjunctive query answering from a knowledge base in Horn-$\mathcal{SHIQ}$ can be done in polynomial time in the data complexity [Eiter et al. 2008], since all evaluations of dl-atoms can be reduced to this problem. Observe first that, for L in Horn-$\mathcal{SHIQ}$, any negated concept (resp., role) membership axiom $\neg C(b)$ (resp., $\neg R(b, c)$) in the input argument of a dl-atom can be ignored in the actual evaluation of the dl-query, and handled by evaluating an additional dl-query $C(b)$ (resp., $R(b, c)$): if any of these (polynomially many) additional dl-queries evaluates to true, then the original dl-query evaluates to true (since the description logic knowledge base along with the input of the dl-atom is unsatisfiable), otherwise the original dl-query is simply evaluated ignoring $\neg C(b)$ (resp., $\neg R(b, c)$). This is due to the fact that knowledge bases in Horn-$\mathcal{SHIQ}$ have canonical universal models [Eiter et al. 2008]. Observe then that dl-queries $C(b)$ and $R(b, c)$ are clearly conjunctive queries. Moreover, axioms $=(b, c)$ and $\neq(b, c)$ are disallowed in Horn-$\mathcal{SHIQ}$ and thus also cannot occur as dl-queries. Furthermore, all other dl-queries can be reduced to knowledge base unsatisfiability: (i) $L' \models \neg C(b)$ iff $L' \cup \{C(b)\}$ is unsatisfiable; (ii) $L' \models \neg R(b, c)$ iff $L' \cup \{R(b, c)\}$ is unsatisfiable; (iii) $L' \models C \sqsubseteq D$ iff $L' \cup \{C(e), D'(e), D \sqcap D' \sqsubseteq \bot\}$ is unsatisfiable; and (iv) $L' \models \neg(C \sqsubseteq D)$

iff $L' \cup \{C \sqsubseteq D\}$ is unsatisfiable, where e is a fresh individual, and D' is a fresh atomic concept. This can in turn be reduced to conjunctive queries: L' is unsatisfiable iff $L' \cup \{A'(d),\, A \sqcap A' \sqsubseteq \bot\} \models A(d)$, where d is a fresh individual, and A and A' are fresh atomic concepts. □

Proof of Theorem 7.3. Since KB is acyclic, there is a mapping $\kappa\colon \mathcal{P}_P \to \{0, 1, \ldots, n\}$ such that for every dl-rule $r \in P$, the predicate symbol p of $H(r)$, and every predicate symbol q of some ordinary $b \in B(r)$ or of an input argument of some dl-atom $b \in B(r)$, it holds that $\kappa(p) > \kappa(q)$. We call $\kappa(p)$ the *rank* of p. By assumption, every dl-query in P can be expressed in terms of a first-order formula over the set A of all concept and role membership axioms in L. We now show by induction on $\kappa(p) \in \{0, 1, \ldots, n\}$ that each predicate symbol $p \in \mathcal{P}_P$ can be expressed in terms of a first-order formula over the set F of all concept and role membership axioms in L and the *database facts* in P, constructed from predicate symbols of rank 0.

Basis: Each predicate $p \in \mathcal{P}_P$ of rank 0 can trivially be expressed in terms of a first-order formula over F.

Induction: We have to consider the evaluation of a dl-atom $DL[\lambda; Q](\mathbf{c})$ and the definition of a predicate $p \in \mathcal{P}_P$ via the set of all rules in P with p in their head:

(i) Consider the dl-atom $DL[\lambda; Q](\mathbf{c})$ with $\lambda = \lambda^+, \lambda^-$, where $\lambda^+ = S_1 \uplus p_1, \ldots, S_l \uplus p_l$, $\lambda^- = S_{l+1} \uplus p_{l+1}, \ldots, S_m \uplus p_m$, and $m \geqslant l \geqslant 0$. The dl-query $Q(\mathbf{c})$ can be expressed in terms of a first-order formula $\alpha(\mathbf{x})$ over A, that is, $L \models Q(\mathbf{c})$ iff $I_A \models \alpha(\mathbf{c})$. Since the underlying DL allows for first-order rewritable concept and role memberships, every S_i in λ^-, $l < i \leqslant m$, can be expressed in terms of a first-order formula $\psi_{S_i}(\mathbf{y})$ over A, that is, $L \models S_i(\mathbf{c})$ iff $I_A \models \psi_{S_i}(\mathbf{c})$ for every $\mathbf{c}$. By the induction hypothesis, every input predicate p_j in λ can be expressed in terms of a first-order formula $\psi_j(\mathbf{x})$ over F, that is, $p_j(\mathbf{c}) \in WFS(KB)$ iff $I_F \models \psi_j(\mathbf{c})$. We define the first-order formula $\delta(\mathbf{x})$ for $DL[\lambda; Q](\mathbf{x})$ over F as follows:

$$\delta(\mathbf{x}) = \alpha^{\lambda^+}(\mathbf{x}) \vee \bigvee_{j=l+1}^{m} \exists \mathbf{y}\, (\psi_{S_j}^{\lambda^+}(\mathbf{y}) \wedge \psi_j(\mathbf{y}))\,, \tag{6}$$

where β^{λ^+} is obtained from β by replacing every $S_i(\mathbf{s})$ such that S_i occurs in λ^+ by $S_i(\mathbf{s}) \vee \psi_{i_1}(\mathbf{s}) \vee \cdots \vee \psi_{i_{k_i}}(\mathbf{s})$, where $S_{i_1}, \ldots, S_{i_{k_i}}$ are all occurrences of S_j in λ^+.

For example, suppose $L = \{C(a)\}$ and

$$P = \{\ p(c);\ q(b);\ r \leftarrow p(x);\ r \leftarrow DL[C \uplus p; C](x);\ s \leftarrow not\ DL[C \uplus p, C \uplus q; C](x)\ \}.$$

Then, both dl-atoms in P have the same query $Q(x)$ $(= C(x))$ over L which can be expressed by the formula $\alpha(x) = C(x)$ over $A = \{C(a)\}$, and the predicates p and q can be expressed by the formulas $\psi_p(x) = p(x)$ and $\psi_q(x) = q(x)$, respectively, over $F = \{C(a), p(c), q(b)\}$. The dl-atom $DL[C \uplus p; C](x)$ is thus translated into $\delta_1(x) = \alpha^{\lambda^+}(x) = C(x) \vee p(x)$ over F (note that $m = l$), while the dl-atom $DL[C \uplus p,\, C \uplus q;\, C](x)$ is translated into $\delta_2(x) = C(x) \vee p(x) \vee \exists y\, ((C(y) \vee p(y)) \wedge q(y))$ over F.

Note that $I_F \models S_i(\mathbf{c})$ iff $S_i(\mathbf{c}) \in L$, for all $1 \leqslant i \leqslant l$. Hence,

$I_F \models S_i(\mathbf{c}) \vee \psi_{i_1}(\mathbf{c}) \vee \cdots \vee \psi_{i_{k_i}}(\mathbf{c})$

iff $S_i(\mathbf{c}) \in L$ or $p_{i_j}(\mathbf{c}) \in WFS(KB)$, for some $1 \leqslant j \leqslant k_i$

iff $S_i(\mathbf{c}) \in L \cup \bigcup_{i=1}^{l} A_i(WFS(KB))$ (recall $A_i(I)$ from Section 3.2)

iff $I_{A'} \models S_i(\mathbf{c})$, where $A' = A \cup \bigcup_{i=1}^{l} A_i(WFS(KB))$.

It follows from this that

$I_F \models \alpha^{\lambda^+}(\mathbf{c})$ iff $I_{A'} \models \alpha(\mathbf{c})$ and $I_F \models \psi_{S_j}^{\lambda^+}(\mathbf{c})$ iff $I_{A'} \models \psi_{S_j}(\mathbf{c})$, for all $l < j \leqslant m$.

This in turn implies that

$I_F \models \delta(\mathbf{c})$ iff (i) $L \cup A' \models Q(\mathbf{c})$, or
(ii) $L \cup A' \models S_j(\mathbf{d})$ and $p_j(\mathbf{d}) \in WFS(KB)$ for some $l < j \leqslant m$ and $\mathbf{d}$.

Let $A'' = A' \cup \bigcup_{j=l+1}^{m} A_j(WFS(KB))$. If $L \cup A'' \not\models Q(\mathbf{c})$, then clearly both (i) and (ii) are false; conversely, if $L \cup A' \not\models Q(\mathbf{c})$ and $L \cup A' \not\models S_j(\mathbf{d})$ for every $p_j(\mathbf{d}) \in WFS(KB)$ where $l < j \leqslant m$, then $L \cup A'' \not\models Q(\mathbf{c})$ holds since the underlying DL is CWA-satisfiable.

In summary, this shows that $I_F \models \delta(\mathbf{c})$ iff $L \cup A'' \models Q(\mathbf{c})$ iff $WFS(KB)$ satisfies $DL[\lambda; Q](\mathbf{c})$. That is, $\delta(\mathbf{x})$ is a first-order formula for $DL[\lambda; Q](\mathbf{x})$ over F.

(ii) Consider next the set of all rules in P with p in their head. W.l.o.g., the heads $p(\mathbf{x})$ of all these rules coincide. Let $\alpha(\mathbf{x})$ denote the disjunction of the existentially quantified bodies of these rules, where the default negations in the rule bodies are interpreted as classical negations. By the induction hypothesis, every body predicate in $\alpha(\mathbf{x})$ can be expressed in terms of a first-order formula over F, and the same holds for every dl-atom in $\alpha(\mathbf{x})$. Let $\alpha'(\mathbf{x})$ be obtained from $\alpha(\mathbf{x})$ by replacing all but the predicates of rank 0 by these first-order formulas. Then, $\alpha'(\mathbf{x})$ is a first-order formula over F for p.

Continuing our example, the rules for r in P are translated into the first-order formula

$$\exists x\, p(x) \vee \exists x \delta_1(x) = \exists x\, p(x) \vee \exists x\, (C(x) \vee p(x)) \;\equiv\; \exists x\, (C(x) \vee p(x))$$

and the rule for s into

$$\exists x\, \neg\delta_2(x) = \exists x\, \neg(C(x) \vee p(x) \vee \exists y\, ((C(y) \vee p(y)) \wedge q(y)))$$

over $\{C(a), p(c), q(b)\}$.

Proof of Theorem 7.4. We apply Theorem 7.3. Observe first that L is defined in a description logic of the *DL-Lite* family in which knowledge base satisfiability and conjunctive queries are both first-order rewritable. Observe also that L is defined in a CWA-satisfiable description logic [Calvanese et al. 2007] (and thus Theorem 7.3 also allows the operator $\uplus$ to occur in P). Hence, all dl-atoms with dl-queries of the form $C(t)$ and $R(t, s)$ are immediately first-order rewritable. Furthermore, all other dl-atoms are also first-order rewritable, since their dl-queries can be reduced to conjunctive queries as follows: (i) $L' \models C \sqsubseteq D$ iff $L' \cup \{C(e), D'(e), D' \sqsubseteq \neg D, A'(d), A' \sqsubseteq \neg A\} \models A(d)$, and (ii) $L' \models \neg(C \sqsubseteq D)$ iff $L' \cup \{C \sqsubseteq D, A'(d), A' \sqsubseteq \neg A\} \models A(d)$, where d and e are fresh individuals, and A, A', and D' are fresh atomic concepts. By Theorem 7.3, it thus follows that deciding whether $l \in WFS(KB)$ is first-order rewritable. □

References

BARAL, C. AND SUBRAHMANIAN, V. S. 1993. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *J. Autom. Reasoning 10,* 3, 399–420.

BERNERS-LEE, T., HENDLER, J., AND LASSILA, O. 2001. The Semantic Web. *Sci. Am. 284,* 5, 34–43.

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

CALVANESE, D., DE GIACOMO, G., LEMBO, D., LENZERINI, M., AND ROSATI, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning 39*, 3, 385–429.

CUENCA GRAU, B., HORROCKS, I., MOTIK, B., PARSIA, B., , AND SATTLER, P. P.-S. U. 2008. OWL 2: The next step for OWL. *J. Web Sem. 6*, 4, 309–322.

DANTSIN, E., EITER, T., GOTTLOB, G., AND VORONKOV, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv. 33*, 3, 374–425.

DE BRUIJN, J., EITER, T., POLLERES, A., AND TOMPITS, H. 2007. Embedding non-ground logic programs into autoepistemic logic for knowledge base combination. In *Proceedings IJCAI-2007*. AAAI Press/IJCAI, 304–309.

DE BRUIJN, J., LAUSEN, H., POLLERES, A., AND FENSEL, D. 2006. The Web service modeling language WSML: An overview. In *Proceedings ESWC-2006*. LNCS, vol. 4011. Springer, 590–604.

DE BRUIJN, J., PEARCE, D., POLLERES, A., AND VALVERDE, A. 2007. Quantified equilibrium logic and hybrid rules. In *Proceedings RR-2007*. LNCS, vol. 4524. Springer, 58–72.

DENECKER, M., MAREK, V. W., AND TRUSZCZYNSKI, M. 2004. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Inf. Comput. 192*, 1, 84–121.

DENECKER, M. AND VENNEKENS, J. 2007. Well-founded semantics and the algebraic theory of nonmonotone inductive definitions. In *Proceedings LPNMR-2007*. LNCS, vol. 4483. Springer, 84–96.

DONINI, F. M., LENZERINI, M., NARDI, D., AND SCHAERF, A. 1994. Deduction in concept languages: From subsumption to instance checking. *J. Log. Comput. 4*, 4, 423–452.

DONINI, F. M., LENZERINI, M., NARDI, D., AND SCHAERF, A. 1998. $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. Intell. Inf. Syst. 10*, 3, 227–252.

DRABENT, W., EITER, T., IANNI, G., KRENNWALLNER, T., LUKASIEWICZ, T., AND MAŁUSZYŃSKI, J. 2009. Hybrid reasoning with rules and ontologies. In *Semantic Techniques for the Web: The REWERSE perspective*, F. Bry and J. Małuszyński, Eds. LNCS. Springer, Chapter 1. To appear.

DRABENT, W., HENRIKSSON, J., AND MALUSZYNSKI, J. 2007. Hybrid reasoning with rules and constraints under well-founded semantics. In *Proceedings RR-2007*. LNCS, vol. 4524. Springer, 348–357.

DRABENT, W. AND MALUSZYNSKI, J. 2007. Well-founded semantics for hybrid rules. In *Proceedings RR-2007*. LNCS, vol. 4524. Springer, 1–15.

EITER, T., GOTTLOB, G., ORTIZ, M., AND ŠIMKUS, M. 2008. Query answering in the description logic Horn-$\mathcal{SHIQ}$. In *Proceedings JELIA-2008*. LNCS, vol. 5293. Springer, 166–179.

EITER, T., IANNI, G., KRENNWALLNER, T., AND POLLERES, A. 2008. Rules and ontologies for the Semantic Web. In *Reasoning Web*. LNCS, vol. 5224. Springer, 1–53.

EITER, T., IANNI, G., KRENNWALLNER, T., AND SCHINDLAUER, R. 2009. Exploiting conjunctive queries in description logic programs. *Ann. Math. Artif. Intell.*. doi:10.1007/s10472-009-9111-3. In press.

EITER, T., IANNI, G., LUKASIEWICZ, T., SCHINDLAUER, R., AND TOMPITS, H. 2008. Combining answer set programming with description logics for the Semantic Web. *Artif. Intell. 172,* 12/13, 1495–1539.

EITER, T., IANNI, G., SCHINDLAUER, R., AND TOMPITS, H. 2005. Nonmonotonic description logic programs: Implementation and experiments. In *Proceedings LPAR-2004*. LNCS, vol. 3452. Springer, 511–517.

EITER, T., LUKASIEWICZ, T., SCHINDLAUER, R., AND TOMPITS, H. 2004. Combining answer set programming with description logics for the Semantic Web. In *Proceedings KR-2004*. AAAI Press, 141–151.

FABER, W. 2005. Unfounded sets for disjunctive logic programs with arbitrary aggregates. In *Proceedings LPNMR-2005*. LNCS, vol. 3662. Springer, 40–52.

FENSEL, D., HENDLER, J., LIEBERMAN, H., AND WAHLSTER, W., Eds. 2002. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generat. Comput. 9,* 365–385.

GLIMM, B., LUTZ, C., HORROCKS, I., AND SATTLER, U. 2008. Answering conjunctive queries in the $\mathcal{SHIQ}$ description logic. *J. Artif. Intell. Res. 31,* 150–197.

GROSOF, B. N., HORROCKS, I., VOLZ, R., AND DECKER, S. 2003. Description logic programs: Combining logic programs with description logics. In *Proceedings WWW-2003*. ACM Press, 48–57.

HAARSLEV, V. AND MÖLLER, R. 2001. RACER system description. In *Proceedings IJCAR-2001*. LNCS, vol. 2083. Springer, 701–705.

HORROCKS, I. AND PATEL-SCHNEIDER, P. F. 2004. Reducing OWL entailment to description logic satisfiability. *J. Web Sem. 1,* 4, 345–357.

HORROCKS, I., PATEL-SCHNEIDER, P. F., BOLEY, H., TABET, S., GROSOF, B., AND DEAN, M. 2004. SWRL: A Semantic Web rule language combining OWL and RuleML. W3C Member Submission. Available at `http://www.w3.org/Submission/SWRL/`.

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. *J. Web Sem. 1,* 1, 7–26.

HORROCKS, I., SATTLER, U., AND TOBIES, S. 1999. Practical reasoning for expressive description logics. In *Proceedings LPAR-1999*. LNCS, vol. 1705. Springer, 161–180.

HUSTADT, U., MOTIK, B., AND SATTLER, U. 2005. Data complexity of reasoning in very expressive description logics. In *Proceedings IJCAI-2005*. 466–471.

KAZAKOV, Y. 2008. RIQ and SROIQ are harder than SHOIQ. In *Proceedings KR-2008*. AAAI Press, 274–284.

KNORR, M., ALFERES, J. J., AND HITZLER, P. 2007. A well-founded semantics for hybrid MKNF knowledge bases. In *Proceedings DL-2007*. CEUR Workshop Proceedings, vol. 250. CEUR-WS.org, 347–354.

KNORR, M., ALFERES, J. J., AND HITZLER, P. 2008. A coherent well-founded model for hybrid MKNF knowledge bases. In *Proceedings ECAI-2008*. Frontiers in Artificial Intelligence and Applications, vol. 178. IOS Press, 99–103.

KNORR, M. AND HITZLER, P. 2007. A comparison of disjunctive well-founded semantics. In *Proceedings FAInt-2007*. CEUR Workshop Proceedings, vol. 277. CEUR-WS.org.

KRÖTZSCH, M., RUDOLPH, S., AND HITZLER, P. 2008. Description logic rules. In *Proceedings ECAI-2008*. Frontiers in Artificial Intelligence and Applications, vol. 178. IOS Press, 80–84.

LEONE, N., PFEIFER, G., FABER, W., EITER, T., GOTTLOB, G., PERRI, S., AND SCARCELLO, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log. 7,* 3, 499–562.

LEVY, A. Y. AND ROUSSET, M.-C. 1998. Combining Horn rules and description logics in CARIN. *Artif. Intell. 104,* 1–2, 165–209.

LIFSCHITZ, V. 1991. Nonmonotonic databases and epistemic queries. In *Proceedings IJCAI-91*. Morgan Kaufmann, 381–386.

LUKASIEWICZ, T. 2007. A novel combination of answer set programming with description logics for the Semantic Web. In *Proceedings ESWC-2007*. LNCS, vol. 4519. Springer, 384–398.

MOTIK, B., HORROCKS, I., ROSATI, R., AND SATTLER, U. 2006. Can OWL and logic programming live together happily ever after? In *Proceedings ISWC-2006*. LNCS, vol. 4273. Springer, 501–514.

MOTIK, B. AND ROSATI, R. 2007a. Closing Semantic Web Ontologies. Tech. rep., University of Manchester. March 2007. Available at `http://web.comlab.ox.ac.uk/people/Boris.Motik/pubs/mr06closing-report.pdf`.

MOTIK, B. AND ROSATI, R. 2007b. A faithful integration of description logics with logic programming. In *Proceedings IJCAI-2007*. AAAI Press/IJCAI, 477–482.

MOTIK, B., SATTLER, U., AND STUDER, R. 2005. Query answering for OWL-DL with rules. *J. Web Sem. 3,* 1, 41–60.

PELOV, N., DENECKER, M., AND BRUYNOOGHE, M. 2007. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming 7,* 3, 301–353.

PEREIRA, L. M. AND ALFERES, J. J. 1992. Well founded semantics for logic programs with explicit negation. In *Proceedings ECAI-1992*. John Wiley & Sons, 102–106.

PRATT-HARTMANN, I. 2008. Data-complexity of the two-variable fragment with counting quantifiers. Tech. rep. Forthcoming in *Information and Computation*.

ROSATI, R. 2006. $\mathcal{DL}+log$: Tight integration of description logics and disjunctive Datalog. In *Proceedings KR-2006*. AAAI Press, 68–78.

SCHINDLAUER, R. 2006. Answer-set programming for the Semantic Web. Ph.D. thesis, Vienna University of Technology, Austria. Available at `http://www.kr.tuwien.ac.at/staff/roman/papers/thesis.pdf`.

TERRACINA, G., LEONE, N., LIO, V., AND PANETTA, C. 2008. Experimenting with recursive queries in database and logic programming systems. *Theory and Practice of Logic Programming 8,* 2, 129–165.

TOBIES, S. 2001. Complexity results and practical algorithms for logics in knowledge representation. Ph.D. thesis, RWTH Aachen, Germany.

VAN GELDER, A. 1989. The alternating fixpoint of logic programs with negation. In *Proceedings PODS-1989*. ACM Press, 1–10.

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

WANG, K. AND ZHOU, L. 2005. Comparisons and computation of well-founded semantics for disjunctive logic programs. *ACM Trans. Comput. Log. 6,* 2, 295–327.