Well-Founded Semantics for Description Logic Programs in the Semantic Web *

Thomas Eiter[1], Giovambattista Ianni[1,2] Roman Schindlauer[1], and Hans Tompits[1]

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

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

Abstract. In previous work, towards the integration of rules and ontologies in the Semantic Web, we have proposed a combination of logic programming under the answer set semantics with the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which underly the Web ontology languages OWL Lite and OWL DL, respectively. More precisely, we have introduced *description logic programs* (or *dl-programs*), which consist of a description logic knowledge base L and a finite set of description logic rules P, and we have defined their answer set semantics. In this paper, we continue this line of research. Here, as a central contribution, we present the well-founded semantics for dl-programs, and we analyze its semantic properties. In particular, we show that it generalizes the well-founded semantics for ordinary normal programs. Furthermore, we show that in the general case, the well-founded semantics of dl-programs is a partial model that approximates the answer set semantics, whereas in the positive and the stratified case, it is a total model that coincides with the answer set semantics. Finally, we also provide complexity results for dl-programs under the well-founded semantics.

1 Introduction

The *Semantic Web* [6,7,14] aims at extending the current World Wide Web by standards and techniques that enable the *automated processing* of Web content. Among other issues, the main ideas to achieve this goal is to add a machine-readable meaning to Web pages, to use ontologies for a precise definition of shared information terms, and to make use of KR technology for automated reasoning from Web resources.

The Semantic Web is conceived in hierarchical layers, where the *Ontology layer*, in form of the *OWL Web Ontology Language* [35,21] (recommended by the W3C),

* This work was partially supported by the Austrian Science Fund under grants Z29-N04 and P17212-N04, by the European Commission through the IST REWERSE Network of Excellence 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), and by the German Research Foundation through a Heisenberg Fellowship. We thank Ulrike Sattler for providing valuable information on complexity-related issues about OWL-DL related description logics.

is currently the highest layer of sufficient maturity. OWL consists of three increasingly expressive sublanguages, namely *OWL Lite*, *OWL DL*, and *OWL Full*. OWL Lite and OWL DL are essentially very expressive description logics with an RDF syntax [21]. As shown in [19], ontology entailment in OWL Lite (resp., OWL DL) reduces to knowledge base (un)satisfiability in the description logic $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$).

On top of the Ontology layer, the *Rules*, *Logic*, and *Proof layers* of the Semantic Web will be developed next, which should offer sophisticated representation and reasoning capabilities. As a first effort in this direction, *RuleML* (Rule Markup Language) [8] is an XML-based markup language for rules and rule-based systems, while the OWL Rules Language [20] is a first proposal for extending OWL by Horn clause rules.

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

Towards this goal, in [13], we have proposed a combination of logic programs under the answer set semantics with description logics, introducing *description logic programs* (or *dl-programs*), 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

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

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—this is needed if parts of L should not be accessible (for example, if L contains an ontology about risk assessment in credit assignment), and only extensional reasoning services are available.

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, description logic 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

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

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.

The semantics of dl-programs was defined in [13] as an extension of the answer set semantics [15] for ordinary normal programs, which is one of the most widely used semantics for nonmonotonic logic programs. More precisely, in [13], we 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 $\mathcal{SHOIN}(\mathbf{D})$.

In this paper, we continue our work on description logic programs and extend the *well-founded semantics* to this class of programs. Introduced by Van Gelder, Ross, and Schlipf [34], the well-founded semantics is another most widely used semantics for ordinary nonmonotonic logic programs. 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. While the answer set semantics resolves conflicts by virtue of permitting multiple intended models as alternative scenarios, the well-founded semantics remains agnostic in the presence of conflicting information, assigning the truth value *false* to a maximal set of atoms that cannot become true during the evaluation of a given program. Furthermore, well-founded semantics assigns a coherent meaning to *all* programs, while some programs are not consistent under answer set semantics, i.e., lack an answer set.

Another important aspect of the well-founded semantics is that it is geared towards efficient *query answering* and also plays a prominent role in deductive databases (see, e.g., [26] for a proposal for object-oriented deductive databases, which is applied to the Florid system implementing F-logic). As an important computational property, a query to an ordinary normal program is evaluable under well-founded semantics in polynomial time (under data complexity), while query answering under the answer set semantics is intractable in general. Finally, efficient implementations of the well-founded semantics exist, such as the XSB system [28] and Smodels [27].

The main contributions of this paper can be summarized as follows:

(1) We define the well-founded semantics for dl-programs by generalizing Van Gelder *et al.*'s [34] fixpoint characterization of the well-founded semantics for ordinary normal programs based on *greatest unfounded sets*. 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. Furthermore, 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. Finally, the well-founded semantics also tolerates abbreviations for dl-atoms.

(2) Generalizing a result by Baral and Subrahmanian [5], 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.

(3) 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.

(4) Finally, we give a precise characterization of the 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, literal inference under the well-founded semantics has a lower complexity than under the answer set semantics. More precisely, relative to program complexity [11], literal inference under the well-founded semantics for dl-programs over $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) is complete for EXP (resp., $\mathrm{P}^{\mathrm{NEXP}}$), while cautious literal inference under the strong answer set semantics for dl-programs over $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) is complete for co-NEXP (resp., co-$\mathrm{NP}^{\mathrm{NEXP}}$) [13].

2 Preliminaries

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

Normal Programs. We assume a function-free first-order vocabulary Φ with nonempty finite sets of constant and predicate symbols, and a set $\mathcal{X}$ of variables. 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 $\mathit{not}\, a$. A *normal rule* (or *rule*) r is of the form

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

where $a, b_1, \ldots, b_m$ are atoms. We refer to a as the *head* of r, denoted $H(r)$, while the conjunction $b_1, \ldots, b_k, \mathit{not}\, b_{k+1}, \ldots, \mathit{not}\, b_m$ is the *body* of r; its *positive* (resp., *negative*) part is $b_1, \ldots, b_k$ (resp., $\mathit{not}\, b_{k+1}, \ldots, \mathit{not}\, b_m$). We define $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. 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.

The well-founded semantics has many different equivalent definitions [34,5]. We recall here the one based on unfounded sets.

Let P be a program. *Ground terms*, *atoms*, *literals*, etc., are defined as usual. We denote by HB_P the *Herbrand base* of P, i.e., the set of all ground atoms with predicate and constant symbols from P (if P contains no constant, then choose an arbitrary one from Φ), and by $\mathit{ground}(P)$ the set of all ground instances of rules in P (w.r.t. 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 $\mathit{Lit}_P = HB_P \cup \neg.HB_P$ to denote the set of all ground literals with predicate and constant symbols from P. A set $S \subseteq \mathit{Lit}_P$ is *consistent* iff $S \cap \neg.S = \emptyset$. A *three-valued interpretation* relative to P is any consistent $I \subseteq \mathit{Lit}_P$.

A set $U \subseteq HB_P$ is an *unfounded set* of P relative to I, if for every $a \in U$ and every $r \in \mathit{ground}(P)$ 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 operators T_P and W_P on consistent $I \subseteq \mathit{Lit}_P$ are then defined by:

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

The operator W_P is monotonic, and thus has a least fixpoint, denoted $\mathit{lfp}(W_P)$, which is the *well-founded semantics* of P, denoted $\mathit{WFS}(P)$. An atom $a \in HB_P$ is *well-founded* (resp., *unfounded*) w.r.t. P, if a (resp., $\neg a$) is in $\mathit{lfp}(W_P)$. 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.

Example 2.1 Consider the propositional program $P = \{p \leftarrow \mathit{not}\, q;\ q \leftarrow p;\ p \leftarrow \mathit{not}\, r\}$. For $I = \emptyset$, we have $T_P(I) = \emptyset$ and $U_P(\emptyset) = \{r\}$: 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, $W_P(\emptyset) = \{\neg r\}$. Since $T_P(\{\neg r\}) = \{p\}$ and $U_P(\{\neg r\}) = \{r\}$, it follows $W_P(\{\neg r\}) = \{p, \neg r\}$. Since $T_P(\{p, \neg r\}) = \{p, q\}$ and $U_P(\{p, \neg r\}) = \{r\}$, it then follows $W_P(\{p, \neg r\}) = \{p, q, \neg r\}$. Thus, $\mathit{lfp}(W_P) = \{p, q, \neg r\}$. That is, r is unfounded relative to P, and the other atoms are well-founded.

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

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

An *axiom* is of one of the following forms: (1) $C \sqsubseteq D$, where C and D are concepts (*concept inclusion*); (2) $R \sqsubseteq S$, where either $R, S \in \mathbf{R}_A$ or $R, S \in \mathbf{R}_D$ (*role inclusion*); (3) $\mathrm{Trans}(R)$, where $R \in \mathbf{R}_A$ (*transitivity*); (4) $C(a)$, where C is a concept and $a \in \mathbf{I}$ (*concept membership*); (5) $R(a, b)$ (resp., $U(a, v)$), where $R \in \mathbf{R}_A$ (resp., $U \in \mathbf{R}_D$) and $a, b \in \mathbf{I}$ (resp., $a \in \mathbf{I}$ and $v \in \mathrm{dom}(\mathbf{D})$) (*role membership*); and (6) $a = b$ (resp., $a \neq b$), where $a, b \in \mathbf{I}$ (*equality* (resp., *inequality*)). A *knowledge base* L is a finite set of axioms. (For decidability, number restrictions in L are restricted to simple $R \in \mathbf{R}_A$ [22]).

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

For the semantics of $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, we refer the reader to [19].

Example 2.2 A small computer store obtains its hardware from several vendors. It uses the following description logic 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.

$$\begin{array}{l}
\geq 1\ \mathit{supplier} \sqsubseteq \mathit{Shop};\ \top \sqsubseteq \forall \mathit{supplier}.\mathit{Part};\ \geq 2\ \mathit{supplier} \sqsubseteq \mathit{Discount};\\
\mathit{Part}(\mathit{harddisk});\ \mathit{Part}(\mathit{cpu});\ \mathit{Part}(\mathit{case});\\
\mathit{Shop}(s_1);\ \mathit{Shop}(s_2);\ \mathit{Shop}(s_3);\\
\mathit{provides}(s_1, \mathit{case});\ \mathit{provides}(s_2, \mathit{cpu});\ \mathit{provides}(s_3, \mathit{case});\\
\mathit{provides}(s_1, \mathit{cpu});\ \mathit{provides}(s_2, \mathit{harddisk});\ \mathit{provides}(s_3, \mathit{harddisk});\\
\mathit{supplier}(s_3, \mathit{case}).
\end{array}$$

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

3 Description Logic Programs

In this section, we recall *description logic programs* (or simply *dl-programs*) from [13], which combine description logics and normal programs. They consist of a knowledge base L in a description logic and a finite set of description logic rules P. Such rules are similar to usual rules in logic programs with negation as failure, but may also contain *queries to* L, possibly default negated. In [13], 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 only for ease of presentation, since every dl-program with classical negation can be translated into one without, like in the ordinary case. The latter, however, is a technical necessity for the well-founded semantics of dl-programs, but not a severe restriction, since most queries to L are in fact monotonic (naturally, a dl-program may still contain NAF-literals).

A dl-program consists of a description logic knowledge base L and a generalized normal program P, which may contain queries to L. Roughly, in such a query, it is asked whether a certain description logic axiom or its negation logically follows from L or not. Formally, 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 $\neg R(t_1, t_2)$,[3] where R is a role and t_1, t_2 are terms.

A *dl-atom* has the form

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

where each S_i is a concept or role, $op_i \in \{\uplus, \cup\!\!\!\!-\}$, p_i is a unary resp. binary predicate symbol, and $Q(\mathbf{t})$ is a dl-query. We call $p_1, \ldots, p_m$ its *input predicate symbols*. Intuitively, $op_i = \uplus$ (resp., $op_i = \cup\!\!\!\!-$) increases S_i (resp., $\neg S_i$) by the extension of p_i. A *dl-rule* r is of form (1), where any $b \in B(r)$ may be a dl-atom. A *dl-program* $KB = (L, P)$

[3] Note that $\mathcal{SHOIN}(\mathbf{D})$ does not provide terminological role negation; we use the expression $\neg(\exists R.\{b\})(a)$ in order to add and query $\neg R(a, b)$ for a specific pair of individuals.

consists of a description logic 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 Consider the dl-program $KB_1 = (L_1, P_1)$, with L_1 as in Example 2.2 and P_1 given as follows, choosing vendors for needed parts w.r.t. 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) $buy\_cand(V, P) \leftarrow vendor(V), not\ avoid(V), DL[provides](V, P), needed(P),$
$\qquad not\ exclude(P)$
(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.

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

– $A_i(I) = \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$, for $op_i = \uplus$; and
– $A_i(I) = \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$, for $op_i = \cup\!\!\!\!-$.

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.

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 consider only ground dl-atoms which are monotonic relative to a dl-program, but one can also define dl-atoms that are not monotonic; see [13].

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 w.r.t. 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 [15].

Let KB^I denote the dl-program (L, sP_L^I). Since KB^I is positive, it has the least model M_{KB^I}. A *strong answer set* (or simply *answer set*) of KB is an interpretation $I \subseteq HB_P$ such that $I = M_{KB^I}$.

Example 3.2 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).

The strong answer set semantics of dl-programs $KB = (L, P)$ without dl-atoms coincides with the ordinary answer set semantics of P [15]. Furthermore, 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. Note that *stratified dl-programs* are composed of hierarchic layers of positive dl-programs that are linked via default negation [13].

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.

In the sequel, let $KB = (L, P)$ be a dl-program. We first define the notion of an unfounded set for dl-programs. 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) $I^+ \models_L b$ for some dl-atom $b \in B^-(r)$.

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 false. The following examples illustrate the concept of an unfounded set for dl-programs.

Example 4.1 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 $J = \{q(a)\}$, 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 J.

Example 4.2 Consider the dl-program $KB_3 = (L_2, P_3)$ where P_3 results 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 $J = \{q(a)\}$, however, both S_1 and S_2 as well as $S_3 = \{r(a)\}$ are unfounded sets of KB_3.

Example 4.3 The unfounded set of $KB_1 = (L_1, P_1)$ in Example 3.1 relative to $I_0 = \emptyset$ contains $buy\_cand(s_1, harddisk)$, $buy\_cand(s_2, case)$, and $buy\_cand(s_3, cpu)$ 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 implies that KB has a greatest unfounded set relative to I.

Lemma 4.4 *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.*

We now generalize the operators T_P, U_P, and W_P to dl-programs as follows. We define the operators T_{KB}, U_{KB}, and W_{KB} on all consistent $I \subseteq Lit_P$ by:

- $a \in T_{KB}(I)$ 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
- $W_{KB}(I) = T_{KB}(I) \cup \neg.U_{KB}(I)$.

The following result shows that the three operators are all monotonic.

Lemma 4.5 *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 4.6 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 4.7 Consider KB_2 of Example 4.1. 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 4.8 Now consider KB_3 of Example 4.2. 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 4.9 Consider again $U_{KB_1}(I_0 = \emptyset)$ of Example 4.3. $W_{KB_1}(I_0)$ 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 describe some semantic properties of the well-founded semantics for dl-programs. 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*, i.e., $J^+ \cup (\neg.J)^+ = HB_P$. 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.*

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 a characterization of the well-founded semantics given in the next section.

Theorem 5.3 *Let $KB = (L, P)$ be a dl-program. If KB is positive (resp., stratified), then (a) every ground atom $a \in HB_P$ is either well-founded or unfounded relative to KB, 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 5.4 The dl-program KB_2 in Example 4.1 is stratified (intuitively, the recursion through negation is acyclic) while KB_3 in Example 4.2 is not. The result computed in Example 4.7 verifies the conditions of Theorem 5.3.

The following result shows 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 $a(\mathbf{t}) = DL[S_1\, op_1 p_1, \ldots, S_m\, op_m\, p_m; Q](\mathbf{t})$ 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, the well-founded semantics tolerates abbreviations in the sense that they do not change the semantics of a dl-program.

Theorem 5.5 *Let $KB = (L, P)$ be a dl-program. Then, $WFS(KB) = WFS(KB^{dl}) \cap Lit_P$.*

6 Relationship to Strong Answer Set Semantics

In this section, we show that the well-founded semantics for dl-programs can be characterized in terms of the least and greatest fixpoint of a monotone operator γ^2_{KB} similar as the well-founded semantics for ordinary normal programs [5]. We then use this characterization to derive further properties of the well-founded semantics for dl-programs.

For a dl-program $KB=(L, P)$, define the operator γ_{KB} on interpretations $I \subseteq HB_P$ by

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

i.e., as 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 [5]. Note that this result holds only if all dl-atoms in P are monotonic.

Proposition 6.1 *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 6.2 *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 6.3 Consider the dl-program KB_1 from 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 4.9 (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 6.4 *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 6.5 *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, i.e., contains either a or $\neg a$ for every $a \in HB_P$, then it specifies the only strong answer set of KB.

Theorem 6.6 *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.*

7 Computation and Complexity

For any positive dl-program $KB = (L, P)$, its least model M_{KB} is the least fixpoint of $T_{KB}(I)$ [13]. Thus, $\gamma_{KB}(I) = M_{KB^I}$ (with $KB^I = (L, sP_L^I)$) can be computed as

$$\mathit{lfp}(T_{KB^I}) = \bigcup_{i \geq 0} T^i_{KB^I}(\emptyset) \quad (= \bigcup_{i=0}^{|HB_P|} T^i_{KB^I}(\emptyset)).$$

The least and greatest fixpoint of γ^2_{KB} can be constructed as the limits

$$\begin{aligned} U_\infty &= \textstyle\bigcup_{i \geq 0} U_i, \text{ where } U_0 = \emptyset, \text{ and } U_{i+1} = \gamma^2_{KB}(U_i), \text{ for } i \geq 0, \text{ and} \\ O_\infty &= \textstyle\bigcap_{i \geq 0} O_i, \text{ where } O_0 = HB_P, \text{ and } O_{i+1} = \gamma^2_{KB}(O_i), \text{ for } i \geq 0, \end{aligned}$$

respectively, which are both reached within $|HB_P|$ many steps.

We recall that for a given ordinary normal program, computing the well-founded model needs exponential time in general (measured in the program size [11]), and also reasoning from the well-founded model has exponential time complexity.

Furthermore, evaluating a ground dl-atom a for $KB = (L, P)$ of the form (2) given an interpretation I_p of its input predicates $p = p_1, \ldots, p_m$ (i.e., deciding $I \models_L a$ for each I that coincides on p with I_p) is complete for EXP (resp., co-NEXP) for L from $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) [13], where EXP (resp., NEXP) denotes exponential (resp., nondeterministic exponential) time; this is inherited from the complexity of the satisfiability problem for $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) [31,19].

The following result implies that the complexity of the well-founded semantics for dl-programs over $\mathcal{SHIF}(\mathbf{D})$ does not increase over the one of ordinary logic programs.

Theorem 7.1 *Given Φ and a dl-program $KB = (L, P)$ with L in $\mathcal{SHIF}(\mathbf{D})$, computing $WFS(KB)$ is feasible in exponential time. Furthermore, deciding whether for a given literal l it holds that $l \in WFS(KB)$ is* EXP*-complete.*

For dl-programs over $\mathcal{SHOIN}(\mathbf{D})$, the computation of $WFS(KB)$ and reasoning from it is expected to be more complex than for $\mathcal{SHIF}(\mathbf{D})$ knowledge bases, since already evaluating a single dl-atom is co-NEXP-hard. Computing WFS 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 $l \in WFS(KB)$ is in $\mathrm{EXP}^{\mathrm{NP}}$.

A more precise account reveals the following strict characterization of the complexity, which is believed to be lower.

Theorem 7.2 *Given Φ, a dl-program $KB = (L, P)$ with L in $\mathcal{SHOIN}(\mathbf{D})$, and a literal l, deciding $l \in WFS(KB)$ is $\mathrm{P}^{\mathrm{NEXP}}$-complete.*

The results in Theorems 7.1 and 7.2 also show that, like for ordinary normal programs, inference under the well-founded semantics is computationally less complex than under the answer set semantics, since cautious reasoning from the strong answer sets of a dl-programs using a $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$) description logic knowledge base is complete for co-NEXP (resp., co-$\mathrm{NP}^{\mathrm{NEXP}}$) [13].

We leave an account of the data complexity of dl-programs $KB = (L, P)$ (i.e., L and the rules of P are fixed, while facts in P may vary) for further work. However, we note that whenever the evaluation of dl-atoms is polynomial (i.e., in description logic terminology, A-Box reasoning is polynomial), then also the computation of the well-founded semantics for dl-programs is polynomial. Most recent results in [23] suggest that for $\mathcal{SHIF}(\mathbf{D})$, the problem is solvable in polynomial time with an NP oracle (and, presumably, complete for that complexity).

8 Related Work

Related work can be divided into (a) hybrid approaches using description logics as input to logic programs, (b) approaches reducing description logics to logic programs, (c) combinations of description logics with default and defeasible logic, and (d) approaches to rule-based well-founded reasoning in the Semantic Web. Below we discuss some representatives for (a)–(d). Further works are discussed in [13].

The works by Donini *et al.* [12], Levy and Rousset [24], and Rosati [29] are representatives of hybrid approaches using description logics as input. Donini *et al.* [12] introduce a combination of (disjunction-, negation-, and function-free) datalog with the description logic $\mathcal{ALC}$. An integrated knowledge base consists of a structural component in $\mathcal{ALC}$ and a relational component in datalog, where the integration of both components lies in using concepts from the structural component as constraints in rule bodies of the relational component.

The closely related work by Levy and Rousset [24] presents a combination of Horn rules with the description logic $\mathcal{ALCNR}$. In contrast to Donini *et al.* [12], Levy and Rousset also allow for roles as constraints in rule bodies, and do not require the safety condition that variables in constraints in the body of a rule r must also appear in ordinary atoms in the body of r. Finally, Rosati [29] presents a combination of disjunctive datalog (with classical and default negation, but without function symbols) with $\mathcal{ALC}$, which is based on a generalized answer set semantics.

Some approaches reducing description logic reasoning to logic programming are the works by Van Belleghem *et al.* [32], Alsaç and Baral [1], Swift [30], Grosof *et al*. [17], and Hufstadt *et al.* [23]. In detail, Van Belleghem *et al.* [32] analyze the close relationship between description logics and open logic programs, and present a mapping of description logic knowledge bases in $\mathcal{ALCN}$ to open logic programs. Alsaç and Baral [1] and Swift [30] reduce inference in the description logic $\mathcal{ALCQI}$ to query answering from normal logic programs (with default negation, but without disjunctions and classical negations) under the answer set semantics. Grosof *et al.* [17] show how inference in a subset of the description logic $\mathcal{SHOIQ}$ can be reduced to inference in a subset of function-free Horn programs (where negations and disjunctions are disallowed), and vice versa. The type of inference follows traditional minimal model semantics, thus not allowing for nonmonotonic reasoning. In contrast to a mapping between description logics and logic programs, we presented a full-fledged coupling under the well-founded semantics. Hufstadt *et al.* [23] show how $\mathcal{SHIQ}(\mathbf{D})$ can be reduced to disjunctive datalog and exploit this for efficient query answering. As a byproduct of their reduction, they obtain a decidable extension of $\mathcal{SHIQ}(\mathbf{D})$ with positive rules in which variables are bound to objects occurring in the extensional part of the description logic knowledge base. These rules, however, have classical first-order semantics; this can be easily emulated within the strong answer set semantics of [13]. Handling negation is not addressed in [23].

Early work on dealing with default information in description logics is the approach due to Baader and Hollunder [4], where Reiter's default logic is adapted to terminological knowledge bases, differing significantly from our approach. Antoniou [2] combines defeasible reasoning with description logics for the Semantic Web. In [3], Antoniou and Wagner summarize defeasible and strict reasoning in a single rule formalism, building on the idea of using rules as a uniform basis for the Ontology, Logic, and Proof layers. Like in other work above, the considered description logics serve only as an input for the nonmonotonic reasoning mechanism running on top of it. Note that defeasible logic is in general different from well-founded semantics, the latter being able to draw more conclusions in certain situations [9]. Maher and Governatori [25] present a well-founded defeasible logic, based on the definition of unfounded sets by Van Gelder *et al.* [34], which reconstructs the well-founded semantics.

An important approach to rule-based reasoning under the well-founded semantics for the Semantic Web is due to Damásio [10]. He aims at developing Prolog tools for implementing different semantics for RuleML [8]. So far, an XML parser library as well as a RuleML compiler have been developed, providing routines to convert RuleML rule bases to Prolog and vice versa. Currently, the compiler supports paraconsistent well-founded semantics with explicit negation; it is planned to be extended to use XSB. However, as a crucial difference to our work, the approach of [10] does not address interfacing to ontologies and ontology reasoning, and thus provides no direct support for integrating rule-based and ontology reasoning, which we have done in this paper.

9 Summary and Outlook

We have presented the well-founded semantics for dl-programs, which generalizes the well-founded semantics for ordinary normal programs [34]. We have given a definition via greatest unfounded sets for dl-programs as well as an equivalent characterization using a generalized Gelfond-Lifschitz transform. We have then analyzed the semantic properties of the well-founded semantics for dl-programs. In particular, we have shown that it generalizes the well-founded semantics for ordinary normal programs. Moreover, in the general case, the well-founded semantics for dl-programs is a partial model that approximates the answer set semantics, while in the positive and stratified case, it is a total model that coincides with the answer set semantics. Finally, we have also provided detailed complexity results for dl-programs under the well-founded semantics.

An experimental prototype implementation using a datalog engine and RACER [18] is available at `http://www.kr.tuwien.ac.at/staff/roman/dlwfs/`. An interesting topic for further work is to extend the presented well-founded semantics to more general dl-programs, which may, for example, allow for disjunctions, NAF-literals, and dl-atoms in the heads of dl-rules. Furthermore, employing RuleML as a versatile and expressive syntax for our formalism could provide a standardized and well-accepted interface to other applications. Finally, to further enrich description logic programs, we plan to examine the possibility of resolution mechanisms for conflicting rules, like priority relations as in courteous logic programs [16] and defeasible logic [3].

References

1. G. Alsaç and C. Baral. Reasoning in description logics using declarative logic programming. Tech. report, Computer Science and Engineering Dept., Arizona State University, 2001.
2. G. Antoniou. Nonmonotonic rule systems on top of ontology layers. In *Proc. ISWC-2002*, *LNCS* 2342, pp. 394–398, 2002.
3. G. Antoniou and G. Wagner. Rules and defeasible reasoning on the Semantic Web. In *Proc. RuleML-2003*, *LNCS* 2876, pp. 111–120, 2003.
4. F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180, 1995.
5. C. Baral and V. S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *J. Automated Reasoning*, 10(3):399–420, 1993.
6. T. Berners-Lee. *Weaving the Web*. Harper, San Francisco, CA, 1999.
7. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
8. H. Boley, S. Tabet, and G. Wagner. Design rationale for RuleML: A markup language for Semantic Web rules. In *Proc. SWWS-2001*, pp. 381–401, 2001.
9. G. Brewka. On the relationship between defeasible logic and well-founded semantics. In *Proc. LPNMR-2001*, *LNCS* 2713, pp. 121–132, 2001.
10. C. V. Damásio. The W^4 Project, 2002. `http://centria.di.fct.unl.pt/~cd/projectos/w4/index.htm`.
11. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
12. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems (JIIS)*, 10(3):227–252, 1998.

13. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. In *Proc. KR-2004*, pp. 141–151, 2004. Preliminary Report RR-1843-03-13, Institut für Informationssysteme, TU Wien, 2003.
14. D. Fensel, W. Wahlster, H. Lieberman, and J. Hendler, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2002.
15. M. Gelfond and V. Lifschitz. Classical negation in logic programs and deductive databases. *New Generation Computing*, 17:365–387, 1991.
16. B. N. Grosof. Courteous logic programs: Prioritized conflict handling for rules. IBM Research Report RC 20836, IBM Research Division, T.J. Watson Research, 1997.
17. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logics. In *Proc. WWW-2003*, pp. 48–57, 2003.
18. V. Haarslev and R. Möller. RACER system description. In *Proc. IJCAR-2001*, *LNCS* 2083, pp. 701–705, 2001.
19. I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proc. ISWC-2003*, *LNCS* 2870, pp. 17–29, 2003.
20. I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL Rules Language. In *Proc. WWW-2004*, pp. 723–731, 2004.
21. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
22. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. LPAR-1999*, *LNCS* 1705, pp. 161–180, 1999.
23. U. Hufstadt, B. Motik, and U. Sattler. Reasoning for description logics around $\mathcal{SHIQ}$ in a resolution framework. Technical Report 3-8-04/04, FZI Karlsruhe, 2004.
24. A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artif. Intell.*, 104(1-2):165–209, 1998.
25. M. J. Maher and G. Governatori. A semantic decomposition of defeasible logics. In *Proc. AAAI/IAAI-1999*, pp. 299–305, 1999.
26. W. May, B. Ludäscher, and G. Lausen. Well-founded semantics for deductive object-oriented database languages. In F. Bry, R. Ramakrishnan, and K. Ramamohanarao, editors, *Proc. DOOD-1997*, *LNCS* 1341, pp. 320–336, 1997.
27. I. Niemelä, P. Simons, and T. Syrjänen. Smodels: A system for answer set programming. In *Proc. NMR-2000*, 2000.
28. P. Rao, K. Sagonas, T. Swift, D. S. Warren, and J. Freire. XSB: A system for efficiently computing WFS. In *Proc. LPNMR-1997*, *LNCS* 1265, pp. 430–440, 1997.
29. R. Rosati. Towards expressive KR systems integrating datalog and description logics: Preliminary report. In *Proc. DL-1999*, pp. 160–164, 1999.
30. T. Swift. Deduction in ontologies via ASP. In *Proc. LPNMR-2004*, *LNCS* 2923, pp. 275–288, 2004.
31. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.
32. K. Van Belleghem, M. Denecker, and D. De Schreye. A strong correspondence between description logics and open logic programming. In *Proc. ICLP-1997*, pp. 346–360, 1997.
33. A. Van Gelder. The alternating fixpoint of logic programs with negation. In *Proc. PODS-1989*, pp. 1–10, 1989.
34. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
35. W3C. OWL web ontology language overview, 2004. W3C Recommendation (10 February 2004). Available at `www.w3.org/TR/2004/REC-owl-features-20040210/`.