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

PROBABILISTIC DESCRIPTION LOGIC PROGRAMS

THOMAS LUKASIEWICZ

INFSYS RESEARCH REPORT 1843-06-04

JUNE 2006

Institut für Informationssysteme
AB Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 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 VIENNA

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-06-04, JUNE 2006

PROBABILISTIC DESCRIPTION LOGIC PROGRAMS

JUNE 10, 2006

Thomas Lukasiewicz [1]

Abstract. Towards sophisticated representation and reasoning techniques that allow for probabilistic uncertainty in the Rules, Logic, and Proof layers of the Semantic Web, we present probabilistic description logic programs (or pdl-programs), which are a combination of description logic programs (or dl-programs) under the answer set semantics and the well-founded semantics with Poole's independent choice logic. We show that query processing in such pdl-programs can be reduced to computing all answer sets of dl-programs and solving linear optimization problems, and to computing the well-founded model of dl-programs, respectively. Moreover, we show that the answer set semantics of pdl-programs is a refinement of the well-founded semantics of pdl-programs. Furthermore, we also present an algorithm for query processing in the special case of stratified pdl-programs, which is based on a reduction to computing the canonical model of stratified dl-programs.

[1]Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, I-00198 Rome, Italy; e-mail: lukasiewicz@dis.uniroma1.it. Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria; e-mail: lukasiewicz@kr.tuwien.ac.at.

Acknowledgements: This work has been supported by a Heisenberg Professorship of the German Research Foundation (DFG). I am thankful to the reviewers of the ECSQARU-2005 and URSW-2005 abstracts of this paper for their constructive comments, which helped to improve this work.

Copyright © 2006 by the authors

Contents

1 Introduction

The *Semantic Web* [5, 14] aims at an extension of the current World Wide Web by standards and technologies that help machines to understand the information on the Web so that they can support richer discovery, data integration, navigation, and automation of tasks. The main ideas behind it are to add a machine-readable meaning to Web pages, to use ontologies for a precise definition of shared terms in Web resources, to use KR technology for automated reasoning from Web resources, and to apply cooperative agent technology for processing the information of the Web.

The Semantic Web consists of several hierarchical layers, where the *Ontology layer*, in form of the *OWL Web Ontology Language* [46, 23] (recommended by the W3C), 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 [23]. As shown in [21], 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) [6] is an XML-based markup language for rules and rule-based systems, whereas the OWL Rules Language [22] 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, the works [12, 13] have proposed *description logic programs* (or simply *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 simply *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). 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. In this way, additional knowledge (gained in the program) can be supplied to L before querying. The semantics of dl-programs was defined in [12] and [13] as an extension of the answer set semantics by Gelfond and Lifschitz [17] and the well-founded semantics by Van Gelder, Ross, and Schlipf [45], respectively, which are the two most widely used semantics for nonmonotonic logic 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 this line of research. Towards sophisticated representation and reasoning techniques that also allow for modeling probabilistic uncertainty in the Rules, Logic, and Proof layers of the Semantic Web, we present *probabilistic description logic programs* (or simply *pdl-programs*), which generalize dl-programs under the answer set and the well-founded semantics by probabilistic uncertainty. This probabilistic generalization of dl-programs is developed as a combination of dl-programs with Poole's independent choice logic (ICL) [35].

It is important to point out that Poole's ICL is a powerful representation and reasoning formalism for single- and also multi-agent systems, which combines logic and probability, and which can represent a number of important uncertainty formalisms, in particular, influence diagrams, Bayesian networks, Markov decision processes, and normal form games [35]. Furthermore, Poole's ICL also allows for natural notions

of causes and explanations as in Pearl's structural causal models [15].

To my knowledge, this is the first work that combines description logic programs with probabilistic uncertainty. The main contributions are summarized as follows:

- We present probabilistic description logic programs (or pdl-programs), which are a probabilistic generalization of dl-programs [12, 13]. They are a combination of dl-programs with Poole's independent choice logic (ICL) [35]; they properly generalize ICL programs (with finite Herbrand bases) by description logics.

- We define a probabilistic answer set semantics of pdl-programs, which is a generalization of the (strong) answer set semantics of dl-programs in [12]. We show that query processing in pdl-programs under this semantics is reducible to computing all answer sets of dl-programs and solving linear optimization problems.

- We define a probabilistic well-founded semantics of pdl-programs, which is a generalization of the well-founded semantics of dl-programs in [13]. We then show that query processing in pdl-programs under the well-founded semantics can be reduced to computing the well-founded semantics of dl-programs.

- We show that, like for the case of dl-programs, the answer set semantics of pdl-programs is a refinement of the well-founded semantics of pdl-programs. That is, whenever an answer to a query under the well-founded semantics is defined, it coincides with the answer to the query under the answer set semantics.

- We also present an algorithm for query processing in the special case of stratified pdl-programs. It is based on a reduction to computing the canonical model of stratified dl-programs, which can be done by a finite sequence of finite fixpoint iterations. This shows especially that query processing in stratified pdl-programs is conceptually easier than query processing in general pdl-programs.

The rest of this paper is organized as follows. Section 2 recalls the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$. In Section 3, we recall dl-programs under the stratified, the answer set, and the well-founded semantics. In Section 4, we introduce their probabilistic generalization to pdl-programs. Section 5 focuses on query processing in stratified pdl-programs. In Sections 6 and 7, we discuss related work, summarize the main results, and give an outlook on future research.

2 The Description Logics $\mathcal{SHIF}(\mathrm{D})$ and $\mathcal{SHOIN}(\mathrm{D})$

In this section, we recall the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$.

2.1 Syntax

We first describe the syntax of $\mathcal{SHOIN}(\mathbf{D})$. We assume a set of *elementary datatypes* and a set of *data values*. A *datatype* is either an elementary datatype or a set of data values (called *datatype oneOf*). 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 associates with every elementary datatype a subset of $\Delta^{\mathbf{D}}$ and with 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 $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ be nonempty finite and pairwise

disjoint 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 $R \in \mathbf{R}_A$.

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 $o_1, \ldots, o_n \in \mathbf{I}$, then $\{o_1, \ldots, o_n\}$ is a concept (called *oneOf*). If C, C_1, and C_2 are concepts and if $R \in \mathbf{R}_A \cup \mathbf{R}_A^-$, then also $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$, and $\neg C$ are concepts (called *conjunction*, *disjunction*, and *negation*, respectively), as well as $\exists R.C$, $\forall R.C$, $\geq n R$, and $\leq n R$ (called *exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geq 0$. If D is a datatype and $U \in \mathbf{R}_D$, then $\exists U.D$, $\forall U.D$, $\geq n U$, and $\leq n U$ are concepts (called *datatype exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geq 0$. We write $\top$ and $\bot$ to abbreviate the concepts $C \sqcup \neg C$ and $C \sqcap \neg C$, respectively, and we eliminate parentheses as usual.

An *axiom* is an expression of one of the following forms: (1) $C \sqsubseteq D$ (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., $a \neq b$) (called *equality* (resp., *inequality*) *axiom*), where $a, b \in \mathbf{I}$. A *knowledge base* L is a finite set of axioms. For decidability, number restrictions in L are restricted to simple abstract roles $R \in \mathbf{R}_A$ [24].

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.

Example 2.1 An online store (such as *amazon.com*) may use a description logic knowledge base to classify and characterize its products. For example, suppose that (1) textbooks are books, (2) personal computers and cameras are electronic products, (3) books and electronic products are products, (4) every product has at least one related product, (5) only products are related to each other, (6) tb_ai and tb_lp are textbooks, which are related to each other, (7) pc_ibm and pc_hp are personal computers, which are related to each other, and (8) ibm and hp are providers for pc_ibm and pc_hp, respectively. This knowledge is expressed by the following description logic knowledge base L_1 in $\mathcal{SHIF}(\mathbf{D})$:

(1) $Textbook \sqsubseteq Book$; (2) $PC \sqcup Camera \sqsubseteq Electronics$;
(3) $Book \sqcup Electronics \sqsubseteq Product$; (4) $Product \sqsubseteq \geq 1\, related$;
(5) $\geq 1\, related \sqcup \geq 1\, related^- \sqsubseteq Product$;
(6) $Textbook(tb\_ai)$; $Textbook(tb\_lp)$; $related(tb\_ai, tb\_lp)$;
(7) $PC(pc\_ibm)$; $PC(pc\_hp)$; $related(pc\_ibm, pc\_hp)$;
(8) $provides(ibm, pc\_ibm)$; $provides(hp, pc\_hp)$.

2.2 Semantics

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):

- $\{o_1, \ldots, o_n\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\}$;

- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, and $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;
- $(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$;
- $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y\colon (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$;
- $(\geq nR)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \mid (x,y) \in R^{\mathcal{I}}\}) \geq n\}$;
- $(\leq nR)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \mid (x,y) \in R^{\mathcal{I}}\}) \leq n\}$;
- $(\exists U.D)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x,y) \in U^{\mathcal{I}} \wedge y \in D^{\mathbf{D}}\}$;
- $(\forall U.D)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y\colon (x,y) \in U^{\mathcal{I}} \rightarrow y \in D^{\mathbf{D}}\}$;
- $(\geq nU)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \mid (x,y) \in U^{\mathcal{I}}\}) \geq n\}$;
- $(\leq nU)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \mid (x,y) \in U^{\mathcal{I}}\}) \leq n\}$.

The *satisfaction* of a description logic 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 $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$; (6) $\mathcal{I} \models U(a,v)$ iff $(a^{\mathcal{I}}, v^{\mathbf{D}}) \in U^{\mathcal{I}}$; (7) $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$; and (8) $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. The interpretation $\mathcal{I}$ *satisfies* the axiom F, or $\mathcal{I}$ is a *model* of F, iff $\mathcal{I} \models F$. The interpretation $\mathcal{I}$ *satisfies* a knowledge base L, or $\mathcal{I}$ is a *model* of L, denoted $\mathcal{I} \models L$, iff $\mathcal{I} \models F$ for all $F \in L$. We say that L is *satisfiable* (resp., *unsatisfiable*) iff L has a (resp., no) model. An axiom F is a *logical consequence* of L, denoted $L \models F$, iff every model of L satisfies F. A negated axiom $\neg F$ is a *logical consequence* of L, denoted $L \models \neg F$, iff every model of L does not satisfy F.

3 Description Logic Programs

In this section, we recall *description logic programs* (or *dl-programs*) [12, 13], which are a combination of 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. We describe the canonical semantics of positive and stratified dl-programs, as well as the answer set semantics and the well-founded semantics of general dl-programs.

3.1 Syntax

We now define the syntax of dl-programs. We first define the syntax of ordinary normal rules and of ordinary normal and positive programs.

We assume a function-free first-order vocabulary Φ with nonempty finite sets of constant and predicate symbols, and a set of variables $\mathcal{X}$. A *term* is a constant symbol from Φ or a variable from $\mathcal{X}$. If p is a predicate symbol of arity $k \geq 0$ from Φ and $t_1, \ldots, t_k$ are terms, then $p(t_1, \ldots, t_k)$ is an *atom*. A *negation-as-failure literal* is an atom a or a default-negated atom $not\, a$. A *normal rule* r is of the form

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

where $a, b_1, \ldots, b_m$ are atoms. We refer to a as the *head* of r, denoted $H(r)$, while the conjunction $b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m$ is called the *body* of r; its *positive* (resp., *negative*) part is $b_1, \ldots, b_k$ (resp., $not\, b_{k+1}, \ldots, not\, b_m$). We 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* P is a finite set of normal rules. We say that P is *positive* iff no rule in P contains default-negated atoms.

We next define the syntax of dl-programs. Informally, they consist of a description logic knowledge base L and a generalized normal program P, which may contain queries to L. 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 $\neg R(t_1, t_2)$, where R is a role and t_1, t_2 are terms.

A *dl-atom* has the form $DL[S_1 op_1 p_1, \ldots, S_m op_m\ p_m; Q](\mathbf{t})$, where each S_i is a concept resp. role, $op_i \in \{\uplus, \uplus\}$ resp. $op_i = \uplus$, p_i is a unary resp. binary predicate symbol, $Q(\mathbf{t})$ is a dl-query, and $m \geq 0$. We call $p_1, \ldots, p_m$ its *input predicate symbols*. Intuitively, $op_i = \uplus$ (resp., $op_i = \uplus$) increases S_i (resp., $\neg S_i$) by the extension of p_i. A *dl-rule* r is of the form (1), where any $b \in B(r)$ is either an ordinary atom or a dl-atom. A *description logic program* (or *dl-program*) $KB = (L, P)$ consists of a description logic knowledge base L and a finite set of dl-rules P. *Ground terms*, *atoms*, *literals*, etc., are defined as usual. The *Herbrand base* of P, denoted HB_P, is the set of all ground atoms with standard predicate symbols that occur in P and constant symbols in Φ. We denote by $ground(P)$ the set of all ground instances of dl-rules in P relative to HB_P.

Example 3.1 Consider the dl-program $KB_1 = (L_1, P_1)$, where L_1 is the description logic knowledge base from Example 2.1, and P_1 is the following set of dl-rules:

(1) $pc(pc\_1)$; $pc(pc\_2)$; $pc(pc\_3)$;
(2) $brand\_new(pc\_1)$; $brand\_new(pc\_2)$;
(3) $vendor(dell, pc\_1)$; $vendor(dell, pc\_2)$; $vendor(dell, pc\_3)$;
(4) $avoid(X) \leftarrow DL[Camera](X), not\ offer(X)$;
(5) $offer(X) \leftarrow DL[PC \uplus pc; Electronics](X), not\ brand\_new(X)$;
(6) $provider(V) \leftarrow vendor(V, X), DL[PC \uplus pc; Product](X)$;
(7) $provider(V) \leftarrow DL[provides](V, X), DL[PC \uplus pc; Product](X)$;
(8) $similar(X, Y) \leftarrow DL[related](X, Y)$;
(9) $similar(X, Z) \leftarrow similar(X, Y), similar(Y, Z)$.

The above dl-rules express that (1) pc_1, pc_2, and pc_3 are additional personal computers, (2) pc_1 and pc_2 are brand new, (3) $dell$ is the vendor of pc_1, pc_2, and pc_3, (4) a customer avoids all cameras that are not on offer, (5) all electronic products that are not brand new are on offer, (6) every vendor of a product is a provider, (7) every entity providing a product is a provider, (8) all related products are similar, and (9) the binary similarity relation on products is transitively closed.

3.2 Semantics of Positive DL-Programs

We now define positive dl-programs and their canonical semantics. We first define interpretations and the satisfaction of dl-programs in interpretations.

In the sequel, let $KB{=}(L, P)$ be a dl-program. An *interpretation* I relative to P is any $I \subseteq HB_P$. We say that I is a *model* of $a \in HB_P$, denoted $I \models a$, iff $a \in I$. We say I is a *model* of $a \in HB_P$ under L,

denoted $I \models_L a$, iff $I \models a$. We say 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 $op_i = \cup$. 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 monotonic ground dl-atoms, but observe that one can also define dl-atoms that are not monotonic; see [12]. We say that I is a *model* of a ground dl-rule r under L, denoted $I \models_L r$, 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 every $r \in ground(P)$. We say KB is *satisfiable* (resp., *unsatisfiable*) iff it has some (resp., no) model.

We say $KB{=}(L, P)$ is *positive* iff no dl-rule in P contains default-negated atoms. Like ordinary positive programs, every positive dl-program KB is satisfiable and has a unique least model, denoted M_{KB}, that naturally characterizes its semantics.

3.3 Semantics of Stratified DL-Programs

We next define stratified dl-programs and their canonical semantics. They are intuitively composed of hierarchic layers of positive dl-programs linked via default negation. Like ordinary stratified normal programs, they are always satisfiable and can be assigned a canonical minimal model via a number of iterative least models.

For any dl-program $KB = (L, P)$, we denote by DL_P the set of all ground dl-atoms that occur in $ground(P)$. An *input atom* of $a \in DL_P$ is a ground atom with an input predicate of a and constant symbols in Φ. A *(local) stratification of* $KB = (L, P)$ is a mapping $\lambda \colon HB_P \cup DL_P \to \{0, 1, \ldots, k\}$ such that

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

(ii) $\lambda(a) \geq \lambda(b)$ for each input atom b of each $a \in DL_P$,

where $k \geq 0$ is the *length* of λ. For $i \in \{0, \ldots, k\}$, let $KB_i = (L, P_i) = (L, \{r \in ground(P) \mid \lambda(H(r)) = i\})$, and let HB_{P_i} (resp., $HB^{\star}_{P_i}$) be the set of all $b \in HB_P$ such that $\lambda(b) = i$ (resp., $\lambda(b) \leq i$). A dl-program $KB = (L, P)$ is *(locally) stratified* iff it has a stratification λ of some length $k \geq 0$. We define its iterative least models $M_i \subseteq HB_P$ with $i \in \{0, \ldots, k\}$ as follows:

(i) M_0 is the least model of KB_0;

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

The canonical model of the stratified dl-program KB, denoted M_{KB}, is then defined as M_k. Observe that M_{KB} is well-defined, since it does not depend on a particular λ. Furthermore, M_{KB} is in fact a minimal model of KB.

3.4 Answer Set Semantics of DL-Programs

The *answer set semantics* of general dl-programs is defined by a reduction to the least model semantics of positive dl-programs as follows. We use a transformation that removes all default-negated atoms in dl-rules and that generalizes the Gelfond-Lifschitz transformation [17]. More precisely, for dl-programs $KB = (L, P)$, the *(strong) dl-transform* of P relative to L and an interpretation $I \subseteq HB_P$, denoted sP^I_L, 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. A *(strong) answer set* of KB is an interpretation $I \subseteq HB_P$ such that I is the unique least model of (L, sP_L^I).

The answer set semantics of dl-programs $KB = (L, P)$ without dl-atoms coincides with the ordinary answer set semantics of P. Answer sets of a general dl-program KB are also minimal models of KB. Positive and locally stratified dl-programs have exactly one answer set, which coincides with their canonical minimal model.

3.5 Well-Founded Semantics of DL-Programs

In the sequel, let $KB = (L, P)$ be a dl-program. 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 define $Lit_P = HB_P \cup \neg.HB_P$. A set $S \subseteq Lit_P$ is *consistent* iff $S \cap \neg.S = \emptyset$. A *three-valued interpretation* relative to P is any consistent $I \subseteq Lit_P$. We define the well-founded semantics of KB by generalizing its standard definition based on unfounded sets [45].

We first define unfounded sets of 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 $b \in B^-(r)$.

For every dl-program $KB = (L, P)$ and consistent $I \subseteq Lit_P$, the set of unfounded sets of KB relative to I is closed under union, and thus KB has a greatest unfounded set relative to I, denoted $U_{KB}(I)$. Intuitively, if I is compatible with KB, then all atoms in $U_{KB}(I)$ can be safely switched to false and the resulting interpretation is still compatible with KB. We define the operators T_{KB} and W_{KB} on all consistent $I \subseteq Lit_P$ 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 $b \in B^-(r)$;
- $W_{KB}(I) = T_{KB}(I) \cup \neg.U_{KB}(I)$.

The operators T_{KB}, U_{KB}, and W_{KB} are all monotonic. Thus, in particular, W_{KB} has a least fixpoint, denoted $lfp(W_{KB})$. The *well-founded semantics* of $KB = (L, P)$, 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., $\neg a$) is in $WFS(KB)$. Intuitively, starting with $I = \emptyset$, rules are applied to obtain new positive (resp., negated) facts via $T_{KB}(I)$ (resp., $\neg.U_{KB}(I)$). This process is repeated until no longer possible.

The well-founded semantics of dl-programs $KB = (L, P)$ without dl-atoms coincides with the ordinary well-founded semantics of P. In general, $WFS(KB)$ is a partial model of KB. Here, a consistent $I \subseteq Lit_P$ is a *partial model* of KB iff it can be extended to a (two-valued) model $I' \subseteq HB_P$ of KB. Like in the ordinary case, the well-founded semantics for positive and locally stratified dl-programs is total and coincides with their canonical minimal model. The well-founded semantics for dl-programs also approximates their answer set semantics. That is, every well-founded (resp., unfounded) atom $a \in HB_P$ is true (resp., false) in every answer set.

4 Probabilistic Description Logic Programs

In this section, we define probabilistic dl-programs (or pdl-programs) as a combination of dl-programs with Poole's independent choice logic (ICL) [35]. Poole's ICL is based on ordinary acyclic logic programs under different "choices", where every choice along with an acyclic logic program produces a first-order model, and one then obtains a probability distribution over the set of all first-order models by placing a probability distribution over the different choices. Here, we use stratified dl-programs under their canonical semantics, as well as dl-programs under the well-founded and the answer set semantics, instead of the above ordinary acyclic logic programs under their canonical semantics (which coincides with their stable model semantics and their answer set semantics, respectively).

4.1 Syntax

We now define the syntax of pdl-programs and probabilistic queries addressed to them. We first define probabilistic formulas and probabilities on choice spaces.

We assume a function-free first-order vocabulary Φ with nonempty finite sets of constant and predicate symbols, and a set of variables $\mathcal{X}$, as in Section 3.1. We use HB_Φ (resp., HU_Φ) to denote the Herbrand base (resp., universe) over Φ. In the sequel, we assume that HB_Φ is nonempty. We define *classical formulas* by induction as follows. The propositional constants *false* and *true*, denoted $\bot$ and $\top$, respectively, and all atoms are classical formulas. If ϕ and ψ are classical formulas, then $\neg\phi$ and $(\phi \wedge \psi)$ are also classical formulas. A *conditional constraint* is of the form $(\psi|\phi)[l,u]$ with reals $l,u \in [0,1]$ and classical formulas ϕ and ψ. We define *probabilistic formulas* inductively as follows. Every conditional constraint is a probabilistic formula. If F and G are probabilistic formulas, then also $\neg F$ and $(F \wedge G)$. We use $(F \vee G)$, $(F \Leftarrow G)$, and $(F \Leftrightarrow G)$ to abbreviate $\neg(\neg F \wedge \neg G)$, $\neg(\neg F \wedge G)$, and $(\neg(\neg F \wedge G) \wedge \neg(F \wedge \neg G))$, respectively, and adopt the usual conventions to eliminate parentheses. *Ground terms*, *ground formulas*, *substitutions*, and *ground instances* of probabilistic formulas are defined as usual.

A *choice space* C is a set of pairwise disjoint and nonempty sets $A \subseteq HB_\Phi$. Any member $A \in C$ is an *alternative* of C and any element $a \in A$ an *atomic choice* of C. A *total choice* of C is a set $B \subseteq HB_\Phi$ such that $|B \cap A| = 1$ for all $A \in C$. A *probability* μ on a choice space C is a probability function on the set of all total choices of C. Since C and all its alternatives are finite, μ can be defined by (i) a mapping $\mu\colon \bigcup C \rightarrow [0,1]$ such that $\sum_{a \in A} \mu(a) = 1$ for all $A \in C$, and (ii) $\mu(B) = \Pi_{b \in B} \mu(b)$ for all total choices B of C. Intuitively, (i) associates a probability with each atomic choice of C, and (ii) assumes independence between the alternatives of C.

A *probabilistic dl-program* (or *pdl-program*) $KB = (L, P, C, \mu)$ consists of a dl-program (L, P), a choice space C such that no atomic choice in C coincides with the head of any dl-rule in $ground(P)$ (note that this condition ensures that stratified pdl-programs are always consistent; cf. Section 4.2), and a probability μ on C. Intuitively, since the total choices of C select subsets of P, every probabilistic dl-program is the compact representation of a probability distribution on a finite set of dl-programs. A *probabilistic query* to KB has the form $?F$ or the form $?(\beta|\alpha)[R,S]$, where F is a probabilistic formula, β, α are classical formulas, and R, S are variables. The *correct answer* to $?F$ is the set of all substitutions θ such that $F\theta$ is a consequence of KB. The *tight answer* to $?(\beta|\alpha)[R,S]$ is the set of all substitutions θ such that $?(\beta|\alpha)[R,S]\theta$ is a tight consequence of KB. In the following paragraphs, we define the notions of *consequence* and *tight consequence* under the stratified, the answer set, and the well-founded semantics.

Example 4.1 Consider the pdl-program $KB_1 = (L_1, P_1, C_1, \mu_1)$, where L_1 and P_1 are as in Example 2.1 resp. 3.1 except that the dl-rules (4) and (5) are replaced by the dl-rules (4′) and (5′), respectively, and the

dl-rules (10) and (11) are added:

(4′) $avoid(X) \leftarrow DL[Camera](X), not\ offer(X), avoid\_pos$;
(5′) $offer(X) \leftarrow DL[PC \uplus pc; Electronics](X), not\ brand\_new(X), offer\_pos$;
(10) $buy(C, X) \leftarrow needs(C, X), view(X), not avoid(X), v\_buy\_pos$;
(11) $buy(C, X) \leftarrow needs(C, X), buy(C, Y), also\_buy(Y, X), a\_buy\_pos$.

Let $C_1 = \{\{avoid\_pos,\ avoid\_neg\},\ \{offer\_pos,\ offer\_neg\},\ \{v\_buy\_pos,\ v\_buy\_neg\},\ \{a\_buy\_pos,\ a\_buy\_neg\}\}$, and let μ_1 be given by $\mu_1(avoid\_pos) = 0.9$, $\mu_1(avoid\_neg) = 0.1$, $\mu_1(offer\_pos) = 0.9$, $\mu_1(offer\_neg) = 0.1$, $\mu_1(v\_buy\_pos) = 0.7$, $\mu_1(v\_buy\_neg) = 0.3$, $\mu_1(a\_buy\_pos) = 0.7$, and $\mu_1(a\_buy\_neg) = 0.3$.

Here, the new dl-rules (4′) and (5′) express that the dl-rules (4) and (5) actually only hold with the probability 0.9. Furthermore, (10) expresses that a customer buys a needed product that is viewed and not avoided with the probability 0.7, while (11) says that a customer buys a needed product x with probability 0.7, if she bought another product y, and every customer that previously had bought y also bought x.

In a probabilistic query, one may ask for the tight probability bounds that a customer c buys a needed product x, if (i) c bought another product y, (ii) every customer that previously had bought y also bought x, (iii) x is not avoided, and (iv) c has been shown product x (the result to this query may, e.g., help to decide whether it is useful to make a customer automatically also view product x when buying y):

$$?(buy(c, x) \mid needs(c, x) \wedge buy(c, y) \wedge also\_buy(y, x) \wedge view(x) \wedge \neg avoid(x))[R, S]\,.$$

4.2 Semantics of Stratified PDL-Programs

A *stratified pdl-program* is a pdl-program $KB{=}(L, P, C, \mu)$ such that the dl-program (L, P) is stratified. In the following, we define the semantic notions of consequence and tight consequence for stratified pdl-programs.

Example 4.2 Consider again the pdl-program $KB_1 = (L_1, P_1, C_1, \mu_1)$ given in Example 4.1. It is not difficult to see that KB_1 is stratified.

A *total world* I is a subset of HB_Φ. We use $\mathcal{I}_\Phi$ to denote the set of all total worlds over Φ. A *variable assignment* σ maps each variable $X \in \mathcal{X}$ to an element of HU_Φ. We extend σ to all terms by $\sigma(c) = c$ for all constant symbols c from Φ. The *truth* of classical formulas ϕ in I under a variable assignment σ, denoted $I \models_\sigma \phi$ (or $I \models \phi$ when ϕ is ground), is inductively defined by:

- $I \models_\sigma p(t_1, \ldots, t_k)$ iff $p(\sigma(t_1), \ldots, \sigma(t_k)) \in I$;
- $I \models_\sigma \neg\phi$ iff not $I \models_\sigma \phi$; and $I \models_\sigma (\phi \wedge \psi)$ iff $I \models_\sigma \phi$ and $I \models_\sigma \psi$.

A *total probabilistic interpretation* Pr is a probability function on $\mathcal{I}_\Phi$ (that is, since $\mathcal{I}_\Phi$ is finite, a mapping $Pr\colon \mathcal{I}_\Phi \to [0, 1]$ such that all $Pr(I)$ with $I \in \mathcal{I}_\Phi$ sum up to 1). The *probability* of a classical formula ϕ in Pr under a variable assignment σ, denoted $Pr_\sigma(\phi)$ (or $Pr(\phi)$ when ϕ is ground), is defined as the sum of all $Pr(I)$ such that $I \in \mathcal{I}_\Phi$ and $I \models_\sigma \phi$. For classical formulas ϕ and ψ with $Pr_\sigma(\phi) > 0$, we use $Pr_\sigma(\psi|\phi)$ to abbreviate $Pr_\sigma(\psi \wedge \phi)\,/\,Pr_\sigma(\phi)$. The *truth* of probabilistic formulas F in Pr under σ, denoted $Pr \models_\sigma F$, is inductively defined as follows:

- $Pr \models_\sigma (\psi|\phi)[l, u]$ iff $Pr_\sigma(\phi) = 0$ or $Pr_\sigma(\psi|\phi) \in [l, u]$;

- $Pr \models_\sigma \neg F$ iff not $Pr \models_\sigma F$; and $Pr \models_\sigma (F \wedge G)$ iff $Pr \models_\sigma F$ and $Pr \models_\sigma G$.

A total probabilistic interpretation Pr is a *model* of a probabilistic formula F iff $Pr \models_\sigma F$ for every variable assignment σ. We say that Pr is the *canonical model* of a stratified pdl-program $KB = (L, P, C, \mu)$ iff every world $I \in \mathcal{I}_\Phi$ with $Pr(I) > 0$ is the canonical model of $(L, P \cup \{p \leftarrow \mid p \in B\})$ for some total choice B of C such that $Pr(I) = \mu(B)$. Observe that every stratified pdl-program KB has a unique canonical model Pr. A probabilistic formula F is a *consequence* of KB, denoted $KB \mid\!\sim F$, iff the canonical model of KB is also a model of F. A conditional constraint $(\psi|\phi)[l,u]$ is a *tight consequence* of KB, denoted $KB \mid\!\sim_{tight} (\psi|\phi)[l,u]$, iff l (resp., u) is the infimum (resp., supremum) of $Pr_\sigma(\psi|\phi)$ subject to the canonical model Pr of KB and all variable assignments σ with $Pr_\sigma(\phi) > 0$. Note that query processing in stratified pdl-programs is discussed in Section 5 below.

Example 4.3 Consider again the pdl-program $KB_1 = (L_1, P_1, C_1, \mu_1)$ given in Example 4.1. Since (L_1, P_1) is stratified, also KB_1 is stratified. The choice space C_1 has 16 total choices, and each of these total choices is associated with a probability under μ_1. For example, one total choice of C_1 is given by $B_1 = \{avoid\_pos, offer\_pos, v\_buy\_pos, a\_buy\_pos\}$; it has the probability $\mu_1(B_1) = 0.9 \times 0.9 \times 0.7 \times 0.7 = 0.3969$. Every total choice B of C_1 specifies a canonical model of a stratified dl-program, namely, the canonical model of $(L, P \cup \{p \leftarrow \mid p \in B\})$, which is associated with the probability $\mu_1(B)$. Hence, the canonical model Pr_1 of KB_1 consists of 16 canonical models of stratified dl-programs along with their probabilities. For example, the canonical model I_{B_1} for the above total choice B_1 satisfies (among others) the ground atoms in lines (1) to (3) of Example 3.1 as well as $offer(pc3)$, $offer(pc\_ibm)$, and $offer(pc\_hp)$; the canonical model I_{B_1} has the associated probability $\mu_1(B_1) = 0.3969$. The canonical model Pr_1 of KB_1 thus represents exactly one probability distribution over first-order models, and it allows to assign a probability to any ground classical formula. For example, the ground atoms $offer(pc3)$, $offer(pc\_ibm)$, and $offer(pc\_hp)$ have each the probability 0.9 under the canonical model Pr_1 of KB_1 (beside I_{B_1} there are other I_B that satisfy $offer(pc3)$, $offer(pc\_ibm)$, and $offer(pc\_hp)$).

4.3 Answer Set Semantics of PDL-Programs

We now introduce the notions of consistency, consequence, and tight consequence under the answer set semantics for general pdl-programs.

A total probabilistic interpretation Pr is an *answer set model* of a pdl-program $KB = (L, P, C, \mu)$ iff (i) every total world $I \in \mathcal{I}_\Phi$ with $Pr(I) > 0$ is an answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$ for some total choice B of C, and (ii) $Pr(\bigwedge B) = Pr(\bigwedge_{p \in B} p) = \mu(B)$ for every total choice B of C. We say that KB is *consistent* iff it has an answer set model Pr. A probabilistic formula F is an *answer set consequence* of KB, denoted $KB \mid\!\sim^{as} F$, iff every answer set model of KB is also a model of F. A conditional constraint $(\psi|\phi)[l,u]$ is a *tight answer set consequence* of KB, denoted $KB \mid\!\sim^{as}_{tight} (\psi|\phi)[l,u]$, iff l (resp., u) is the infimum (resp., supremum) of $Pr_\sigma(\psi|\phi)$ subject to all answer set models Pr of KB and all variable assignments σ with $Pr_\sigma(\phi) > 0$. Here, we assume that $l = 1$ and $u = 0$, when $Pr_\sigma(\phi) = 0$ for all answer set models Pr of KB and all σ.

Every stratified pdl-program KB is consistent and has exactly one answer set model, which coincides with the canonical model of KB. Deciding whether a general pdl-program KB is consistent can be reduced to deciding whether dl-programs have an answer set. The following theorem shows that computing tight answers to queries $?(\beta|\alpha)[R,S]$ to KB, where β and α are ground, can be reduced to computing all answer sets of dl-programs and then solving two linear optimization problems. It follows from a standard result on transforming linear fractional programs into equivalent linear programs by Charnes and Cooper [7].

$$\begin{aligned}
\sum_{r\in R,\, r\not\models\bigwedge B} -\mu(B)\, y_r + \sum_{r\in R,\, r\models\bigwedge B} (1-\mu(B))\, y_r &= 0 \quad \text{(for all total choices } B \text{ of } C\text{)}\\
\sum_{r\in R,\, r\models\alpha} y_r &= 1\\
y_r &\geq 0 \quad \text{(for all } r\in R\text{)}
\end{aligned}$$

Figure 1: System of linear constraints LC for Theorem 4.4.

Theorem 4.4 *Let $KB=(L,P,C,\mu)$ be a consistent pdl-program, and let β and α be ground classical formulas such that $Pr(\alpha)>0$ for some answer set model Pr of KB. Then, l (resp., u) such that $KB \mid\!\sim^{as}_{tight} (\beta|\alpha)[l,u]$ is the optimal value of the following linear program over the variables y_r $(r\in R)$, where R is the union of all sets of answer sets of $(L, P\cup\{p\leftarrow \mid p\in B\})$ for all total choices B of C:*

minimize (resp., maximize) $\sum_{r\in R,\, r\models\beta\wedge\alpha} y_r$ *subject to LC in Fig. 1.*

4.4 Well-Founded Semantics of PDL-Programs

We finally define the notions of consequence and tight consequence under the well-founded semantics for general pdl-programs. We first define partial probabilistic interpretations and the evaluation of probabilistic formulas in them.

A *partial world* I is a consistent subset of $Lit_\Phi = HB_\Phi \cup \neg.HB_\Phi$. We identify I with the three-valued interpretation $I\colon HB_\Phi \to \{\mathbf{true}, \mathbf{false}, \mathbf{undefined}\}$ that is defined by $I(a)=\mathbf{true}$ iff $a\in I$, $I(a)=\mathbf{false}$ iff $\neg a\in I$, and $I(a)=\mathbf{undefined}$ iff $I\cap\{a,\neg a\}=\emptyset$). We use $\mathcal{I}^p_\Phi$ to denote the set of all partial worlds over Φ. Every classical formula ϕ in a partial world I under a variable assignment σ is associated with a *three-valued truth value* from $\{\mathbf{true}, \mathbf{false}, \mathbf{undefined}\}$, denoted $I_\sigma(\phi)$ (or simply $I(\phi)$ when ϕ is ground), which is inductively defined by:

- $I_\sigma(p(t_1,\ldots,t_k)) = I(p(\sigma(t_1),\ldots,\sigma(t_k)))$;
- $I_\sigma(\neg\phi)=\mathbf{true}$ iff $I_\sigma(\phi)=\mathbf{false}$, and $I_\sigma(\neg\phi)=\mathbf{false}$ iff $I_\sigma(\phi)=\mathbf{true}$;
- $I_\sigma(\phi\wedge\psi)=\mathbf{true}$ iff $I_\sigma(\phi)=I_\sigma(\psi)=\mathbf{true}$, and
  $I_\sigma(\phi\wedge\psi)=\mathbf{false}$ iff $I_\sigma(\phi)=\mathbf{false}$ or $I_\sigma(\psi)=\mathbf{false}$.

A *partial probabilistic interpretation* Pr is a probability function on $\mathcal{I}^p_\Phi$. The *probability* of a classical formula ϕ in Pr under a variable assignment σ, denoted $Pr_\sigma(\phi)$ (or simply $Pr(\phi)$ when ϕ is ground), is $\mathbf{undefined}$, if $I_\sigma(\phi)$ is $\mathbf{undefined}$ for some $I\in\mathcal{I}^p_\Phi$ with $Pr(I)>0$; and $Pr_\sigma(\phi)$ is defined as the sum of all $Pr(I)$ such that $I\in\mathcal{I}^p_\Phi$ and $I_\sigma(\phi)=\mathbf{true}$, otherwise. For classical formulas ϕ and ψ such that $Pr_\sigma(\phi)>0$, the *conditional probability* of ψ given ϕ in Pr under σ, denoted $Pr_\sigma(\psi|\phi)$, is defined as $Pr_\sigma(\psi\wedge\phi)\,/\,Pr_\sigma(\phi)$. Note that, alternatively, we may also define $Pr_\sigma(\phi)$ as the interval $[\sum_{I_\sigma(\phi)=\mathbf{true}} Pr(I), 1-\sum_{I_\sigma(\phi)=\mathbf{false}} Pr(I)]$. However, even though this definition ensures that $Pr_\sigma(\phi)$ is always defined, it cannot easily be generalized to conditional probabilities. Every probabilistic formula F in Pr under σ is associated with a *three-valued truth value* from $\{\mathbf{true}, \mathbf{false}, \mathbf{undefined}\}$, denoted $Pr_\sigma(F)$, which is inductively defined as follows:

- $Pr_\sigma((\psi|\phi)[l,u]) = \mathbf{true}$ iff $Pr_\sigma(\phi) = 0$ or $Pr_\sigma(\psi|\phi) \in [l,u]$, and
  $Pr_\sigma((\psi|\phi)[l,u]) = \mathbf{false}$ iff $Pr_\sigma(\phi) > 0$ and $Pr_\sigma(\psi|\phi) \notin [l,u]$;
- $Pr_\sigma(\neg F) = \mathbf{true}$ iff $Pr_\sigma(F) = \mathbf{false}$, and $Pr_\sigma(\neg F) = \mathbf{false}$ iff $Pr_\sigma(F) = \mathbf{true}$;
- $Pr_\sigma(F \wedge G) = \mathbf{true}$ iff $Pr_\sigma(F) = Pr_\sigma(G) = \mathbf{true}$, and
  $Pr_\sigma(F \wedge G) = \mathbf{false}$ iff $Pr_\sigma(F) = \mathbf{false}$ or $Pr_\sigma(G) = \mathbf{false}$.

The *well-founded model* of a pdl-program $KB = (L, P, C, \mu)$, denoted Pr^{wf}_{KB}, is defined as follows: (i) $Pr^{wf}_{KB}(I_B) = \mu(B)$, where I_B is the well-founded model of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every total choice B of C, and (ii) $Pr^{wf}_{KB}(I) = 0$ for all other $I \in \mathcal{I}^p_\Phi$. A probabilistic formula F is a *well-founded consequence* of KB, denoted $KB \mid\!\sim^{wf} F$, iff F is $\mathbf{true}$ in Pr^{wf}_{KB} under every variable assignment σ. A conditional constraint $(\psi|\phi)[l,u]$ is a *tight well-founded consequence* of KB, denoted $KB \mid\!\sim^{wf}_{tight} (\psi|\phi)[l,u]$, iff (i) $Pr^{wf}_{KB}(\phi)$ under every variable assignments σ is different from $\mathbf{undefined}$, and (ii) l (resp., u) is the infimum (resp., supremum) of $Pr_\sigma(\psi|\phi)$ subject to $Pr = Pr^{wf}_{KB}$ and all variable assignments σ with $Pr_\sigma(\phi) > 0$.

The well-founded model of a stratified pdl-program KB is total and coincides with the canonical model of KB. As an advantage of the well-founded semantics, every general pdl-program KB has a unique well-founded model, but not necessarily an answer set model. Furthermore, the unique well-founded model can be easily computed by fixpoint iteration [13]. As a drawback, the well-founded model associates only with *some* classical formulas under σ a probability, while every answer set model associates with *all* classical formulas under σ a probability. The following theorem shows that the answer set semantics is a refinement of the well-founded semantics. That is, if an answer to a query under the well-founded semantics is defined, then it coincides with the answer under the answer set semantics. The theorem follows from the result that the well-founded semantics of dl-programs approximates their answer set semantics [13]. The advantages of both semantics can thus be combined in query processing by first trying to compute the well-founded answer, and only if this does not exist the answer under the answer set semantics.

Theorem 4.5 *Let $KB = (L, P, C, \mu)$ be a consistent pdl-program, and let $(\psi|\phi)[l,u]$ be a ground conditional constraint. If $Pr^{wf}_{KB}(\phi), Pr^{wf}_{KB}(\psi \wedge \phi) \neq \mathbf{undefined}$, then*

(a) $KB \mid\!\sim^{wf} (\psi|\phi)[l,u]$ iff $KB \mid\!\sim^{as} (\psi|\phi)[l,u]$, and

(b) $KB \mid\!\sim^{wf}_{tight} (\psi|\phi)[l,u]$ iff $KB \mid\!\sim^{as}_{tight} (\psi|\phi)[l,u]$.

Proof (sketch). Let $\alpha \in \{\psi, \psi \wedge \phi\}$. It is sufficient to show that $Pr^{wf}_{KB}(\alpha)$ is equal to $Pr(\alpha)$ for all answer set models Pr of KB. Observe that $Pr^{wf}_{KB}(\alpha)$ is the sum of all $\mu(B)$ such that (i) B is a total choice of C, (ii) $\mu(B) > 0$, and (iii) $I_B(\alpha) = \mathbf{true}$, where I_B denotes the well-founded model of $(L, P \cup \{p \leftarrow \mid p \in B\})$. By induction on the structure of classical formulas, it is not difficult to see that $I_B(\alpha) = \mathbf{true}$ iff $I \models \alpha$ for all answer sets I of $(L, P \cup \{p \leftarrow \mid p \in B\})$. This already shows that $Pr^{wf}_{KB}(\alpha)$ is equal to $Pr(\alpha)$ for all answer set models Pr of KB. □

5 Query Processing in Stratified PDL-Programs

The canonical model of an ordinary positive (resp., stratified) normal program P has a fixpoint characterization in terms of an immediate consequence operator T_P, which generalizes to positive (resp., stratified)

dl-programs. This can be used for a bottom-up computation of the canonical model of a positive (resp., stratified) dl-program, and thus also for computing the canonical model of a stratified pdl-program and for query processing in stratified pdl-programs.

5.1 Fixpoint Iteration in Positive DL-Programs

We first describe a fixpoint characterization of the canonical model of a positive dl-program. For any dl-program $KB = (L, P)$, we define the operator T_{KB} on the subsets of HB_P as follows. For every $I \subseteq HB_P$, let

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

If KB is positive, then T_{KB} is monotonic. Hence, T_{KB} has a least fixpoint, denoted $\mathit{lfp}(T_{KB})$. Furthermore, $\mathit{lfp}(T_{KB})$ can be computed by a finite fixpoint iteration (given finiteness of P and the number of constant symbols in Φ). For every $I \subseteq HB_P$, we define $T^i_{KB}(I) = I$, if $i = 0$, and $T^i_{KB}(I) = T_{KB}(T^{i-1}_{KB}(I))$, if $i > 0$.

Theorem 5.1 *For every positive dl-program $KB = (L, P)$, it holds that $\mathit{lfp}(T_{KB}) = M_{KB}$. Furthermore, $\mathit{lfp}(T_{KB}) = \bigcup_{i=0}^{n} T^i_{KB}(\emptyset) = T^n_{KB}(\emptyset)$ for some $n \geq 0$.*

5.2 Fixpoint Iteration in Stratified DL-Programs

We next describe a fixpoint characterization for stratified dl-programs. Using Theorem 5.1, we can characterize the canonical model M_{KB} of a stratified dl-program $KB = (L, P)$ as follows. Let $\widehat{T}^i_{KB}(I) = T^i_{KB}(I) \cup I$ for all $i \geq 0$.

Theorem 5.2 *Suppose that $KB = (L, P)$ has a stratification λ of length $k \geq 0$. Let $M_i \subseteq HB_P$, $i \in \{-1, 0, \ldots, k\}$, be defined by $M_{-1} = \emptyset$ and $M_i = \widehat{T}^{n_i}_{KB_i}(M_{i-1})$ for $i \geq 0$, where $n_i \geq 0$ such that $\widehat{T}^{n_i}_{KB_i}(M_{i-1}) = \widehat{T}^{n_i+1}_{KB_i}(M_{i-1})$. Then, $M_k = M_{KB}$.*

5.3 Query Processing in Stratified PDL-Programs

Algorithm *canonical_model* (see Fig. 2) computes the canonical model Pr of a given stratified pdl-program $KB = (L, P, C, \mu)$. It is essentially based on a reduction to computing the canonical model of stratified dl-programs in line 4, which can be done using the above finite sequence of finite fixpoint iterations.

Example 5.3 Consider again the stratified pdl-program $KB_1 = (L_1, P_1, C_1, \mu_1)$ of Example 4.1. The computation of *canonical_model* in Fig. 2 on KB_1 is summarized as follows. For each of the 16 total choices B of C_1, the canonical model I_B of the stratified dl-program $(L_1, P_1 \cup \{p \leftarrow \mid p \in B\})$ is computed, and I_B is associated with the probability $\mu_1(B)$, while all the other $I \in \mathcal{I}_\Phi$ are associated with the probability 0. For example, for the total choice $B_1 = \{\mathit{avoid\_pos},\ \mathit{offer\_pos},\ \mathit{v\_buy\_pos},\ \mathit{a\_buy\_pos}\}$ of C_1, the canonical model I_{B_1} of the stratified dl-program $(L_1, P_1 \cup \{p \leftarrow \mid p \in B_1\})$ satisfies exactly the ground atoms in lines (1) to (3) of Example 3.1 as well as $\mathit{offer}(\mathit{pc3})$, $\mathit{offer}(\mathit{pc\_ibm})$, $\mathit{offer}(\mathit{pc\_hp})$, $\mathit{provider}(\mathit{dell})$, $\mathit{provider}(\mathit{ibm})$, $\mathit{provider}(\mathit{hp})$, $\mathit{similar}(\mathit{tb\_ai}, \mathit{tb\_lp})$, and $\mathit{similar}(\mathit{pc\_ibm}, \mathit{pc\_hp})$, and I_{B_1} is associated with the probability $\mu_1(B_1) = 0.9 \times 0.9 \times 0.7 \times 0.7 = 0.3969$.

Algorithm *canonical_model*

Input: stratified pdl-program $KB = (L, P, C, \mu)$.
Output: canonical model Pr of KB.

1. **for every** interpretation $I \in \mathcal{I}_\Phi$ **do**
2. $\quad Pr(I) := 0$;
3. **for every** total choice B of C **do begin**
4. $\quad$ compute the canonical model I of the
5. $\quad\quad$ stratified dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$;
6. $\quad Pr(I) := \mu(B)$
7. **end**;
8. **return** Pr.

Figure 2: Algorithm *canonical_model*.

Algorithm *tight_answer*

Input: stratified pdl-program $KB = (L, P, C, \mu)$ and probabilistic query $?(\beta|\alpha)[R, S]$.
Output: tight answer $\theta = \{R/l, S/u\}$ for $?(\beta|\alpha)[R, S]$ to KB.

1. $Pr :=$ *canonical_model*(KB);
2. $l := 1$;
3. $u := 0$;
4. **for every** ground instance $\beta'|\alpha'$ of $\beta|\alpha$ **do begin**
5. $\quad l := \min(l, Pr(\beta'|\alpha'))$;
6. $\quad u := \max(u, Pr(\beta'|\alpha'))$
7. **end**;
8. **return** $\theta = \{R/l, S/u\}$.

Figure 3: Algorithm *tight_answer*.

Algorithm *tight_answer* (see Fig. 3) computes tight answers $\theta = \{R/l, S/u\}$ for a given probabilistic query $?(\beta|\alpha)[R, S]$ to a given stratified pdl-program KB. It computes the canonical model of KB in line 1 and the tight answer in lines 2–8.

Example 5.4 To compute the tight answer for the query $?(\mathit{offer}(pc3)|\top)[R, S]$ to the stratified pdl-program $KB_1 = (L_1, P_1, C_1, \mu_1)$ of Example 4.1, Algorithm *tight_answer* in Fig. 3 first computes the canonical model Pr of KB_1, using Algorithm *canonical_model*. Then, since $\beta|\alpha = \mathit{offer}(pc3)|\top$ is ground and unconditional, Algorithm *tight_answer* computes only the probability of $\mathit{offer}(pc3)$ in Pr. Finally, since the latter is given by 0.9, the algorithm returns $\theta = \{R/0.9, S/0.9\}$.

6 Related Work

In this section, we discuss related work on the combination of logic programs with description logics and on uncertainty reasoning for the Semantic Web. Note that an overview of the large body of previous work on the combination of logic programs with probabilistic uncertainty is contained in [30, 26].

6.1 Description Logic Programs

Related work on the combination of description logics and logic programs 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 give some representatives for (a)–(d). Further works and details are given in [12, 13].

The works by Donini *et al.* [10], Levy and Rousset [27], and Rosati [37, 38] are representatives of hybrid approaches using description logics as input. Donini *et al.* [10] 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 [27] presents a combination of Horn rules with the description logic $\mathcal{ALCNR}$. In contrast to Donini *et al.* [10], 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 [37] 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.* [44], Alsaç and Baral [1], Swift [43], Grosof *et al.* [19], and Hufstadt *et al.* [25]. 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. 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.

An important approach to rule-based reasoning under the well-founded semantics for the Semantic Web is due to Damásio [9]. He aims at Prolog tools for implementing different semantics for RuleML [6]. So far, an XML parser library as well as a RuleML compiler have been developed, with routines to convert RuleML rule bases to Prolog and vice versa. The compiler supports paraconsistent well-founded semantics with explicit negation; it is planned to be extended to use XSB [36].

6.2 Uncertainty Reasoning for the Semantic Web

Related approaches to uncertainty reasoning for the Semantic Web can be roughly divided into (a) description logic programs under non-probabilistic uncertainty, (b) probabilistic generalizations of description logics, and (c) probabilistic generalizations of web ontology languages, such as DAML+OIL and OWL.

As for (a), previous works by Straccia combine (positive) description logic programs with *lattice-based uncertainty* [41] and with *fuzzy vagueness* [42]. Whereas, to my knowledge, the present paper is the first one combining (normal) description logic programs with *probabilistic uncertainty*.

As for (b), Giugno and Lukasiewicz [18] present a probabilistic generalization of the expressive description logic $\mathcal{SHOQ}(\mathbf{D})$ that stands behind DAML+OIL, which is based on lexicographic probabilistic reasoning. In earlier work, Heinsohn [20] and Jaeger [28] present probabilistic extensions to the description logic $\mathcal{ALC}$, which are essentially based on probabilistic reasoning in probabilistic logics. Koller et al. [29] present a probabilistic generalization of the CLASSIC description logic, which uses Bayesian networks as underlying probabilistic reasoning formalism. Note that fuzzy description logics, such as the ones by Straccia [39, 40], are less closely related to probabilistic description logics, since they deal with fuzzy vagueness, rather than probabilistic ambiguity and imprecision.

As for (c), there are especially the works by Costa [8], Pool and Aikin [34], and Ding and Peng [11], which present probabilistic generalizations of the web ontology language OWL. In particular, Costa's work [8] is semantically based on multi-entity Bayesian networks, while [11] has a semantics in standard Bayesian networks. In closely related work, Fukushige [16] proposes a basic framework for representing probabilistic relationships in RDF. Finally, Nottelmann and Fuhr [33] present pDAML+OIL, which is a probabilistic generalization of the web ontology language DAML+OIL, along with a mapping to stratified probabilistic datalog.

7 Conclusion

We have presented probabilistic dl-programs (or pdl-programs), which are a combination of dl-programs under the answer set and the well-founded semantics with Poole's independent choice logic. We have shown that query processing in such pdl-programs can be reduced to computing all answer sets of dl-programs and solving linear optimization problems, and to computing the well-founded semantics of dl-programs, respectively. We have also shown that the answer set semantics of pdl-programs is a refinement of the well-founded semantics of pdl-programs. Moreover, we have considered the special case of stratified pdl-programs. In particular, we have presented an algorithm for query processing in such pdl-programs, which is based on a reduction to computing the canonical model of stratified dl-programs.

An interesting topic of future research is to further enhance pdl-programs towards a possible use for Web Services. This may be done by exploiting and generalizing further features of Poole's ICL for dynamic and multi-agent systems [35]. It would also be interesting to further explore the computational aspects of query processing in pdl-programs under the stratified, answer set, and well-founded semantics.

References

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

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

[3] G. Antoniou and G. Wagner. Rules and defeasible reasoning on the Semantic Web. In *Proceedings RuleML-2003, LNCS* 2876, pp. 111–120, 2003.

[4] F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *J. Autom. Reason.*, 14:149–180, 1995.

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

[6] H. Boley, S. Tabet, and G. Wagner. Design rationale for RuleML: A markup language for Semantic Web rules. In *Proceedings SWWS-2001*, pp. 381–401, 2001.

[7] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9:181–186, 1962.

[8] P. C. G. da Costa. Bayesian semantics for the Semantic Web. Doctoral Dissertation, George Mason University, Fairfax, VA, USA, 2005.

[9] C. V. Damásio. The W^4 Project, 2002. See `http://centria.di.fct.unl.pt/~cd/projectos/w4/index.htm`.

[10] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating datalog and description logics. *J. Intell. Inf. Syst. (JIIS)*, 10(3):227–252, 1998.

[11] Z. Ding and Y. Peng. A Probabilistic extension to ontology language OWL. In *Proceedings HICSS-2004*, 2004.

[12] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. In *Proceedings KR-2004*, pp. 141–151, 2004. Extended Report RR-1843-03-13, Institut für Informationssysteme, TU Wien, 2003.

[13] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-founded semantics for description logic programs in the Semantic Web. In *Proceedings RuleML-2004*, *LNCS* 3323, pp. 81–97, 2004.

[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] A. Finzi and T. Lukasiewicz. Structure-based causes and explanations in the independent choice logic. In *Proceedings UAI-2003*, pp. 225–232, 2003.

[16] Y. Fukushige. Representing probabilistic knowledge in the Semantic Web. In *Proceedings of the W3C Workshop on Semantic Web for Life Sciences*, Cambridge, MA, USA, 2004.

[17] M. Gelfond and V. Lifschitz. Classical negation in logic programs and deductive databases. *New Generat. Comput.*, 17:365–387, 1991.

[18] R. Giugno and T. Lukasiewicz. P-$\mathcal{SHOQ}(\mathbf{D})$: A probabilistic extension of $\mathcal{SHOQ}(\mathbf{D})$ for probabilistic ontologies in the Semantic Web. In *Proceedings JELIA-2002*, *LNCS* 2424, pp. 86–97, 2002.

[19] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logics. In *Proceedings WWW-2003*, pp. 48–57, 2003.

[20] J. Heinsohn. Probabilistic description logics. In *Proceedings UAI-1994*, pp. 311–318, 1994.

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

[22] I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL Rules Language. In *Proceedings WWW-2004*, pp. 723–731, 2004.

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

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

[25] 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.

[26] G. Kern-Isberner and T. Lukasiewicz. Combining probabilistic logic programming with the power of maximum entropy. *Artif. Intell.*, 157(1–2):139–202, 2004.

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

[28] M. Jaeger. Probabilistic reasoning in terminological logics. In *Proceedings KR-1994*, pp. 305–316, 1994.

[29] D. Koller, A. Levy, and A. Pfeffer. P-CLASSIC: A tractable probabilistic description logic. In *Proceedings AAAI-1997*, pp. 390–397, 1997.

[30] T. Lukasiewicz. Probabilistic logic programming with conditional constraints. *ACM Trans. Comput. Log.*, 2(3):289–339, 2001.

[31] T. Lukasiewicz. Probabilistic description logic programs. In *Proceedings ECSQARU-2005*, *LNCS* 3571, pp. 737–749, 2005.

[32] T. Lukasiewicz. Stratified Probabilistic description logic programs. In *Proceedings URSW-2005*, pp. 87–97, 2005.

[33] H. Nottelmann and N. Fuhr. pDAML+OIL: A probabilistic extension to DAML+OIL based on probabilistic Datalog. In *Proceedings IPMU-2004*, 2004.

[34] M. Pool and J. Aikin. KEEPER and Protégé: An elicitation environment for Bayesian inference tools. In *Proceedings of the Workshop on Protégé and Reasoning* held at the *7th International Protégé Conference*, 2004.

[35] D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artif. Intell.*, 94(1–2):7–56, 1997.

[36] P. Rao, K. Sagonas, T. Swift, D. S. Warren, and J. Freire. XSB: A system for efficiently computing WFS. In *Proceedings LPNMR-1997*, *LNCS* 1265, pp. 430–440, 1997.

[37] R. Rosati. Towards expressive KR systems integrating datalog and description logics: Preliminary report. In *Proceedings DL-1999*, pp. 160–164, 1999.

[38] R. Rosati. On the decidability and complexity of integrating ontologies and rules. *J. Web Semantics*, 3(1):61–73, 2005.

[39] U. Straccia. Reasoning within fuzzy description logics. *J. Artif. Intell. Res.*, 14:137–166, 2001.

[40] U. Straccia. Towards a fuzzy description logic for the Semantic Web (preliminary report). In *Proceedings ESWC-2005*, *LNCS* 3532, pp. 167–181, 2005.

[41] U. Straccia. Uncertainty and description logic programs over lattices. In E. Sanchez, editor, *Fuzzy Logic and the Semantic Web*, Capturing Intelligence, chapter 7, pp. 115–133. Elsevier, 2006.

[42] U. Straccia. Fuzzy description logic programs. In *Proceedings IPMU-2006*, 2006.

[43] T. Swift. Deduction in ontologies via ASP. In *Proceedings LPNMR-2004*, *LNCS* 2923, pp. 275–288, 2004.

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

[45] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.

[46] W3C. OWL web ontology language overview, 2004. W3C Recommendation (10 Feb. 2004). See `www.w3.org/TR/2004/REC-owl-features-20040210/`.