Tractable Probabilistic Description Logic Programs

Thomas Lukasiewicz and Gerardo I. Simari

Abstract. We propose tractable probabilistic description logic programs (dl-programs) for the Semantic Web, which combine tractable description logics (DLs), normal programs under the answer set and the well-founded semantics, and probabilities. In detail, we first provide novel reductions of tight query processing and of deciding consistency in probabilistic dl-programs under the answer set semantics to the answer set semantics of the underlying normal dl-programs. Based on these reductions, we then introduce a novel well-founded semantics for probabilistic dl-programs, called the *total well-founded semantics*. Contrary to the previous answer set and well-founded semantics, it is defined for all probabilistic dl-programs and all probabilistic queries. Furthermore, tight (resp., tight literal) query processing under the total well-founded semantics coincides with (resp., approximates) tight (resp., tight literal) query processing under the previous well-founded (resp., answer set) semantics in all cases where the latter is defined. We then present an anytime algorithm for tight query processing in probabilistic dl-programs under the total well-founded semantics. We also show that tight literal query processing in probabilistic dl-programs under the total well-founded semantics can be done in polynomial time in the data complexity and is complete for EXP in the combined complexity. Finally, we describe an application of probabilistic dl-programs in probabilistic data integration for the Semantic Web.

1 Introduction

During recent years, formalisms for dealing with probabilistic uncertainty have started to play an important role in research related to the Web and the *Semantic Web*, which is an extension of the current Web by standards and technologies that help machines to understand the information on the Web so that they can support richer discovery, data integration, navigation, and automation of tasks [8, 7]. As

Thomas Lukasiewicz · Gerardo I. Simari
Department of Computer Science, University of Oxford
e-mail: {thomas.lukasiewicz,gerardo.simari}@cs.ox.ac.uk

Z. Ma & L. Yan (Eds.): *Advances in Probabilistic Databases*, STUDFUZZ 304, pp. 131–159.
DOI: 10.1007/978-3-642-37509-5_6

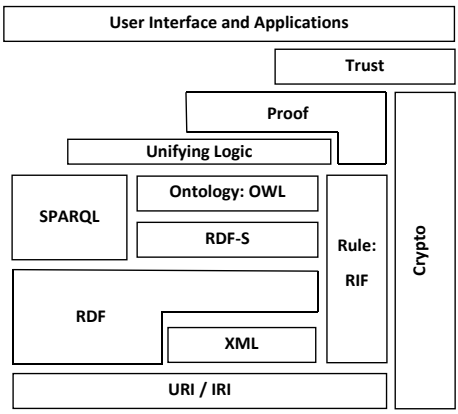

Fig. 1 Hierarchical layers of the Semantic Web

an example of the role of uncertainty in today's Web, note that the order in which Google returns the answers to a Web search query is computed by using probabilistic techniques. Besides Web search and information retrieval, other important Web and Semantic Web applications of formalisms for dealing with probabilistic uncertainty are especially data integration [85] and ontology mapping [67].

The Semantic Web consists of several hierarchical layers, as shown in Fig. 1, where the *Ontology layer*, in the form of the *OWL Web Ontology Language* [86, 38, 87], is currently the highest layer of sufficient maturity. OWL consists of three increasingly expressive sublanguages, namely, *OWL Lite*, *OWL DL*, and *OWL Full*, where OWL Lite and OWL DL are essentially very expressive description logics (DLs) with an RDF syntax [38]. As shown in [37], ontology entailment in OWL Lite (resp., OWL DL) reduces to knowledge base (un)satisfiability in the DL $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$). As a next step in the development of the Semantic Web, one currently aims especially at sophisticated reasoning capabilities for the *Rules*, *Logic*, and *Proof layers* of the Semantic Web.

In particular, there is a large body of work on integrating rules and ontologies, which is a key requirement of the layered architecture of the Semantic Web. One type of integration is to build rules on top of ontologies, i.e., for rule-based systems that use vocabulary from ontology knowledge bases. Another form of integration is to build ontologies on top of rules, where ontological definitions are supplemented by rules or imported from rules. Both types of integration have been realized in recent hybrid integrations of rules and ontologies, called *description logic programs* (or *dl-programs*), which are of the form $KB = (L, P)$, where L is a DL knowledge base, and P is a finite set of rules involving queries to L in a loose coupling [25, 26].

Other research efforts are directed towards formalisms for *uncertainty reasoning in the Semantic Web*: An important recent forum for uncertainty in the Semantic Web is the annual *Workshop on Uncertainty Reasoning for the Semantic Web (URSW)* at the *International Semantic Web Conference (ISWC)*; there was also a

W3C Incubator Group on *Uncertainty Reasoning for the World Wide Web*. There are especially extensions of DLs [31], ontology languages [15, 83], and dl-programs [53] by probabilistic uncertainty (to encode ambiguous information, such as "John is a student (resp., teacher) with the probability 0.7 (resp., 0.3)", which is very different from vague/fuzzy information, such as "John is tall with the degree of truth 0.7").

In particular, the probabilistic dl-programs in [53] are one of the most promising approaches to uncertainty reasoning for the Semantic Web, since they faithfully generalize two well-established logic programming and uncertainty formalisms, namely, answer set programming and Bayesian networks, respectively. They also generalize Poole's independent choice logic (ICL) [70], which is a powerful representation and reasoning formalism for single- and also multi-agent systems. The ICL combines logic and probability, and generalizes many important uncertainty formalisms, in particular, influence diagrams, Bayesian networks, Pearl's causal models, Markov decision processes, and normal form games. Moreover, it allows for natural notions of causes and explanations as in Pearl's causal models [27]. It is also closely related to other approaches to probabilistic logic programming, such as P-log [5] and Bayesian logic programs [42].

Since the Web contains a huge amount of data, as an important feature, Web and Semantic Web formalisms should allow for efficient algorithms. However, no such algorithms were known so far for the probabilistic dl-programs in [53]. In this work, we aim at filling this gap. We propose an approach to probabilistic dl-programs that is defined on top of tractable DLs (rather than $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ as in [53]), and show that this approach allows for tight query processing with polynomial data complexity. In the course of this, we also provide some other new results around probabilistic dl-programs, which are briefly summarized as follows:

- We provide novel reductions of deciding consistency and of tight query processing in probabilistic dl-programs under the answer set semantics to computing the answer sets of the underlying normal dl-programs. These reductions significantly simplify previous reductions proposed in [53], which additionally require to decide the solvability of a (in general quite large) system of linear inequalities and to solve two linear optimization problems relative to them, respectively.
- We define a novel well-founded semantics of probabilistic dl-programs, called the *total well-founded semantics*, since it defines tight answers for *all* probabilistic queries. This is in contrast to the previous well-founded semantics of probabilistic dl-programs in [53], which defines tight answers only for a quite restricted class of probabilistic queries. The total well-founded semantics is also defined for all probabilistic dl-programs, contrary to the answer set semantics, which is only defined for consistent probabilistic dl-programs.
- As for other nice semantic features of the total well-founded semantics, the tight answers under the total well-founded semantics coincide with the tight answers under the well-founded semantics of [53], if the latter are defined. For literal queries, the tight answers under the total well-founded semantics approximate the tight answers under the answer set semantics, if the latter are defined.

- We provide an anytime algorithm for tight query processing in probabilistic dl-programs under the total well-founded semantics, along with a precise characterization of its anytime error. Furthermore, we show that tight query processing under the total well-founded semantics can be done in polynomial time in the data complexity and is complete for EXP in the combined complexity.
- We describe an application of probabilistic dl-programs in probabilistic data integration for the Semantic Web, where probabilistic dl-programs allow for dealing with probabilistic uncertainty and inconsistencies. We especially discuss different types of probabilistic data integration that can be realized with our approach.

The rest of this paper is organized as follows. In Sections 2 and 3, we recall tractable DLs and dl-programs under the answer set and the well-founded semantics, respectively. Section 4 recalls probabilistic dl-programs. In Section 5, we provide a new reduction of tight query processing in probabilistic dl-programs under the answer set semantics to the answer set semantics of normal dl-programs. Section 6 introduces the total well-founded semantics of probabilistic dl-programs. In Section 7, we provide an anytime algorithm for tight query processing in probabilistic dl-programs under the total well-founded semantics, as well as tractability and complexity results. Section 8 describes an application of probabilistic dl-programs in probabilistic data integration for the Semantic Web. In Section 9, we discuss related work. Section 10 summarizes our results, and gives an outlook on future research.

2 Description Logics

The probabilistic dl-programs of this paper assume that the underlying description logic (DL) allows for decidable conjunctive query processing. The tractability and complexity results (see Section 7.2) also assume that the underlying DL allows for conjunctive query processing in polynomial data complexity. We use *DL-Lite* here, but the tractability and complexity results also hold for the variants of *DL-Lite* in [11, 68]. We now recall the syntax and the semantics of *DL-Lite*. Intuitively, DLs model a domain of interest in terms of concepts and roles, which represent classes of individuals and binary relations between classes of individuals, respectively.

2.1 Syntax

We first define concepts and axioms, and then knowledge bases and conjunctive queries in *DL-Lite*. We assume pairwise disjoint sets $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{I}$ of *atomic concepts*, *(atomic) roles*, and *individuals*, respectively. We use $\mathbf{R}^-$ to denote the set of all inverses R^- of roles $R \in \mathbf{R}$. A *basic concept* B is either an atomic concept $A \in \mathbf{A}$ or an *existential role restriction* $\exists R$, where $R \in \mathbf{R} \cup \mathbf{R}^-$. An *axiom* is either:

- A *concept inclusion axiom* $B \sqsubseteq C$, where B is a basic concept, and C is either a basic concept B' or its negation $\neg B'$ (called *concept*),
- a *concept membership axiom* $B(x)$, where B is a basic concept and $x \in \mathbf{I}$,

- a *role membership axiom* $R(x,y)$, where $R \in \mathbf{R}$ and $x,y \in \mathbf{I}$,
- a *functionality axiom* (funct R), where $R \in \mathbf{R} \cup \mathbf{R}^-$.

Given the basic definitions above, a *(DL) knowledge base* L is a finite set of axioms; a *conjunctive query* over L is of the form

$$Q(\mathbf{x}) = \exists \mathbf{y}\,(conj(\mathbf{x},\mathbf{y})), \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are tuples of distinct variables, and $conj(\mathbf{x},\mathbf{y})$ is a conjunction of assertions $B(z)$ and $R(z_1,z_2)$, where B and R are basic concepts and roles from $\mathbf{R}$, respectively, and z, z_1, and z_2 are individuals from $\mathbf{I}$ or variables in $\mathbf{x}$ or $\mathbf{y}$.

Example 1. An online store (such as *amazon.com*) may use a DL 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 *DL-Lite* knowledge base L:

(1) $Textbook \sqsubseteq Book$;
(2) $PC \sqcup Camera \sqsubseteq Electronics$;
(3) $Book \sqcup Electronics \sqsubseteq Product$;
(4) $Product \sqsubseteq \exists related$;
(5) $related \sqcup related^- \sqsubseteq Product$;
(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

The semantics of *DL-Lite* is defined as usual in first-order logics. An *interpretation* $\mathscr{I} = (\Delta^{\mathscr{I}}, \cdot^{\mathscr{I}})$ consists of a nonempty *domain* $\Delta^{\mathscr{I}}$ and a mapping $\cdot^{\mathscr{I}}$ that assigns to each $A \in \mathbf{A}$ a subset of $\Delta^{\mathscr{I}}$, to each $o \in \mathbf{I}$ an element of $\Delta^{\mathscr{I}}$ (such that $o_1 \neq o_2$ implies $o_1^{\mathscr{I}} \neq o_2^{\mathscr{I}}$, commonly referred to as the *unique name assumption*), and to each $R \in \mathbf{R}$ a subset of $\Delta^{\mathscr{I}} \times \Delta^{\mathscr{I}}$. We extend $\cdot^{\mathscr{I}}$ to all concepts and roles as follows:

- $(\neg B)^{\mathscr{I}} = \Delta^{\mathscr{I}} \setminus B^{\mathscr{I}}$, for all basic concepts B;
- $(\exists R)^{\mathscr{I}} = \{x \mid \exists y\colon (x,y) \in R^{\mathscr{I}}\}$, for all roles $R \in \mathbf{R} \cup \mathbf{R}^-$;
- $(R^-)^{\mathscr{I}} = \{(y,x) \mid (x,y) \in R^{\mathscr{I}}\}$, for all atomic roles $R \in \mathbf{R}$.

Next, we define the *satisfaction* of an axiom F by $\mathscr{I}$, denoted $\mathscr{I} \models F$, as usual:

- $\mathscr{I} \models B \sqsubseteq C$ iff $B^{\mathscr{I}} \subseteq C^{\mathscr{I}}$, for all basic concepts B and concepts C;
- $\mathscr{I} \models B(a)$ iff $a^{\mathscr{I}} \in B^{\mathscr{I}}$, for all basic concepts B;
- $\mathscr{I} \models R(a,b)$ iff $(a^{\mathscr{I}}, b^{\mathscr{I}}) \in R^{\mathscr{I}}$, for all atomic roles $R \in \mathbf{R}$;
- $\mathscr{I} \models (\text{funct } R)$ iff $(x,y) \in R^{\mathscr{I}} \wedge (x,z) \in R^{\mathscr{I}} \Rightarrow y = z$, for all roles $R \in \mathbf{R} \cup \mathbf{R}^-$.

An interpretation $\mathscr{I}$ *satisfies* the axiom F, or $\mathscr{I}$ is a *model* of F, iff $\mathscr{I} \models F$. Furthermore, $\mathscr{I}$ *satisfies* a knowledge base L, or $\mathscr{I}$ is a *model* of L, denoted $\mathscr{I} \models L$, iff $\mathscr{I} \models F$ 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.

A tuple $\mathbf{c}$ of individuals from $\mathbf{I}$ is an *answer* for a conjunctive query $Q(\mathbf{x}) = \exists \mathbf{y}\,(conj(\mathbf{x},\mathbf{y}))$ to a knowledge base L iff for every model $\mathscr{I} = (\Delta^{\mathscr{I}}, \cdot^{\mathscr{I}})$ of L, there exists a tuple $\mathbf{o}$ of elements from $\Delta^{\mathscr{I}}$ such that all assertions in $conj(\mathbf{c},\mathbf{o})$ are satisfied in $\mathscr{I}$. In *DL-Lite*, computing all such answers has a polynomial data complexity.

3 Description Logic Programs

We adopt the description logic programs (or dl-programs) of [25, 26], which consist of a DL knowledge base L and a generalized normal program P, which may contain queries to L (called dl-queries). Note that these dl-programs can be extended by queries to other formalisms, such as RDF theories [24]. We first define the syntax of dl-programs and then their answer set and their well-founded semantics. Note that in contrast to [25, 26], we assume here that dl-queries may be conjunctive queries to L.

3.1 Syntax

We assume a function-free first-order vocabulary Φ with finite nonempty sets of constant and predicate symbols Φ_c and Φ_p, respectively, and a set of variables $\mathscr{X}$. We make the following assumptions:

- Φ_c is a subset of $\mathbf{I}$ (since the constants in Φ_c may occur in concept and role assertions of dl-queries); and
- Φ and $\mathbf{A}$ (resp., $\mathbf{R}$) have no unary (resp., binary) predicate symbols in common (and thus dl-queries are the only interface between L and P).

A *term* is a constant symbol from Φ or a variable from $\mathscr{X}$. If p is a predicate symbol of arity $k \geqslant 0$ from Φ, and $t_1, \ldots, t_k$ are terms, then $p(t_1, \ldots, t_k)$ is an *atom*. A *literal* is an atom a or a default-negated atom $not\, a$. A *(normal) rule* r is of the form

$$a \leftarrow b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m, \tag{2}$$

where $a, b_1, \ldots, b_m$ are atoms and $m \geqslant k \geqslant 0$. We call a the *head* of r, denoted $H(r)$, while the conjunction $b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$ is the *body* of r; its *positive* (resp., *negative*) part is $b_1, \ldots, b_k$ (resp., $not\ b_{k+1}, \ldots, not\ b_m$). We define $B(r)$ as the union of $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. A *(normal) program* P is a finite set of (normal) rules. We say P is *positive* iff it is "*not*"-free.

A *dl-query* $Q(\mathbf{t})$ is a conjunctive query of the form (1). A *dl-atom* has the form

$$DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q(\mathbf{t})]\,,$$

where each S_i is a concept or role, p_i is a unary resp. binary predicate symbol, $Q(\mathbf{t})$ is a dl-query, and $m \geqslant 0$. We call $p_1, \ldots, p_m$ its *input predicate symbols*. Intuitively,

$\uplus$ increases S_i by the extension of p_i. A *(normal) dl-rule* r is of the form (2), where any $b \in B(r)$ may be a dl-atom. A *(normal) dl-program* $KB = (L,P)$ consists of a DL knowledge base L and a finite set of (normal) dl-rules P. We say $KB = (L,P)$ is *positive* iff P is positive. *Ground terms*, *atoms*, *literals*, etc., are defined as usual. We denote by $ground(P)$ the set of all ground instances of dl-rules in P relative to Φ_c.

Example 2. Consider the dl-program $KB = (L,P)$, where L is the DL knowledge base from Example 1, and P 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) $avoid(X) \leftarrow DL[PC \uplus pc; PC](X), not\ offer(X)$;
(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) $similar(X,Y) \leftarrow pc(X), pc(Y), X \neq Y$;
(10) $similar(X,Z) \leftarrow similar(X,Y), similar(Y,Z), X \neq Z$;
(11) $buyPC(X) \leftarrow DL[PC \uplus pc; PC](X), not\ avoid(X), not\ exclude(X)$;
(12) $exclude(X) \leftarrow DL[PC \uplus pc; PC](X), buyPC(Y), similar(X,Y)$.

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 PCs 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, (9) all PCs are similar to each other (but a PC is not similar to itself), (10) the binary similarity relation on products is transitively closed, and (11), (12) a customer buys a PC if he does not avoid it and has not decided to buy a similar one already.

3.2 Answer Set Semantics

The *Herbrand base* HB_Φ is the set of all ground atoms constructed from constant and predicate symbols in Φ. An *interpretation* I is any $I \subseteq HB_\Phi$. We say I is a *model* of $a \in HB_\Phi$ under a DL knowledge base L, denoted $I \models_L a$, iff $a \in I$. We say I is a *model* of a ground dl-atom $a = DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q(\mathbf{c})]$ under L, denoted $I \models_L a$, iff $L \cup \bigcup_{i=1}^{m} A_i(I) \models Q(\mathbf{c})$, where $A_i(I) = \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$. We say I is a *model* of a ground dl-rule r iff $I \models_L H(r)$ whenever $I \models_L B(r)$, i.e., $I \models_L a$ 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 $r \in ground(P)$.

Like ordinary positive programs, each positive dl-program KB has a unique least model, denoted M_{KB}, which naturally characterizes its semantics. The *answer set semantics* of general dl-programs is then defined by a reduction to the least model

semantics of positive ones, using a reduct that generalizes the ordinary Gelfond-Lifschitz reduct [30] and removes all default-negated atoms in dl-rules: For dl-programs $KB = (L, P)$, the *dl-reduct* of P relative to L and an interpretation $I \subseteq HB_\Phi$, denoted P_L^I, is the set of all dl-rules obtained from $ground(P)$ by:

- deleting each dl-rule r such that $I \not\models_L a$ for some $a \in B^-(r)$, and
- deleting from each remaining dl-rule r the negative body.

An *answer set* of KB is an interpretation $I \subseteq HB_\Phi$ such that I is the unique least model of (L, P_L^I). A dl-program is *consistent* iff it has an answer set.

Example 3. Consider the dl-program $KB = (L, P)$ from Example 2, which in turn relies on the *DL-Lite* knowledge base L from Example 1. Intuitively, any answer set of KB contains the atoms $offer(pc_3)$, $offer(pc\_ibm)$, and $offer(pc\_hp)$ (since none of these PCs is brand new), $avoid(pc_1)$ and $avoid(pc_2)$ (since they are not on offer), $provider(dell)$, and a set of atoms for *similar* consisting of all possible irreflexive pairs of objects in the set $\{pc\_1, pc\_2, pc\_3, pc\_ibm, pc\_hp\}$.

On the other hand, all answer sets will differ with respect to the *buyPC* atom. Rule (11) in Example 2 states that $buyPC(X)$ is true only if X corresponds to a PC (via the query to the *DL-Lite* knowledge base, augmented with the *pc* predicate), there is no $avoid(X)$ atom in the answer set, and there is no other $buyPC(Y)$ atom in the knowledge base, where Y is similar to X. Therefore, there will be three answer sets, each containing one of $buyPC(pc_3)$, $buyPC(pc\_ibm)$, and $buyPC(pc\_hp)$, as well as $exclude(X)$ atoms for the other two objects.

The answer set semantics of dl-programs has several nice features. In particular, for dl-programs $KB = (L, P)$ without dl-atoms, it coincides with the ordinary answer set semantics of P. Answer sets of a general dl-program KB are also minimal models of KB. Furthermore, positive and locally stratified dl-programs have exactly one answer set, which coincides with their canonical minimal model.

3.3 Well-Founded Semantics

Rather than associating with every dl-program a (possibly empty) set of two-valued interpretations, the well-founded semantics associates with every dl-program a unique three-valued interpretation.

A *(classical) literal* is either an atom a or its negation $\neg a$. For sets $S \subseteq HB_\Phi$, we define $\neg S = \{\neg a \mid a \in S\}$. We define $Lit_\Phi = HB_\Phi \cup \neg HB_\Phi$. A set of ground classical literals $S \subseteq Lit_\Phi$ is *consistent* iff $S \cap \{a, \neg a\} = \emptyset$ for all $a \in HB_\Phi$. A *three-valued interpretation* is any consistent $I \subseteq Lit_\Phi$. We define the well-founded semantics of dl-programs $KB = (L, P)$ via a generalization of the operator γ^2 for ordinary normal programs. We define the operator γ_{KB} as follows. For every $I \subseteq HB_\Phi$, we define $\gamma_{KB}(I)$ as the least model of the positive dl-program $KB^I = (L, P_L^I)$. The operator γ_{KB} is anti-monotonic, and thus the operator γ_{KB}^2 (defined by $\gamma_{KB}^2(I) = \gamma_{KB}(\gamma_{KB}(I))$, for every $I \subseteq HB_\Phi$) is monotonic and has a least and a greatest fixpoint, denoted $lfp(\gamma_{KB}^2)$ and $gfp(\gamma_{KB}^2)$, respectively. Then, the *well-founded semantics* of the dl-program KB, denoted $WFS(KB)$, is defined as $lfp(\gamma_{KB}^2) \cup \neg(HB_\Phi - gfp(\gamma_{KB}^2))$.

Example 4. Consider once again the dl-program $KB = (L, P)$ from Example 2, where L is as in Example 1. The well-founded semantics of KB contains all the atoms in all answer sets in Example 3, but no atoms among $buyPC(pc_3)$, $buyPC(pc\_ibm)$, $buyPC(pc\_hp)$, $exclude(pc_3)$, $exclude(pc\_ibm)$, and $exclude(pc\_hp)$ (and their negations); such atoms are thus *undefined* under the well-founded semantics of KB.

As an important property, the well-founded semantics for dl-programs approximates their answer set semantics. That is, for all consistent dl-programs KB and literals $\ell \in Lit_\Phi$, every $\ell \in WFS(KB)$ is true in every answer set of KB.

4 Probabilistic Description Logic Programs

In this section, we recall probabilistic dl-programs from [53], which are defined as a combination of dl-programs with Poole's independent choice logic (ICL)[70]. The 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. Informally, differently from the ICL, probabilistic dl-programs consist of a dl-program (L, P) and a probability distribution μ over a set of total choices B. Every total choice B along with the dl-program (L, P) then defines a set of Herbrand interpretations of which the probabilities sum up to $\mu(B)$. We now first define the syntax of probabilistic dl-programs and probabilistic queries, and then their answer set semantics.

4.1 Syntax

We now define the syntax of probabilistic dl-programs and probabilistic queries addressed to them. We first define choice spaces 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 above for dl-programs. 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. An *event* α is any Boolean combination of atoms (i.e., constructed from atoms via the Boolean operators "$\wedge$" and "$\neg$"). A *conditional event* is of the form $\beta|\alpha$, where α and β are events. Ground terms, ground events, and substitutions are defined as usual.

A *choice space* C is a set of pairwise disjoint and nonempty sets $A \subseteq HB_\Phi$. Any $A \in C$ is called an *alternative* of C, and any element $a \in A$ is called an *atomic choice* of C. Intuitively, every alternative $A \in C$ represents a random variable and every atomic choice $a \in A$ one of its possible values. A *total choice* of C is a set $B \subseteq HB_\Phi$ such that $|B \cap A| = 1$ for all $A \in C$ (and thus $|B| = |C|$). Intuitively, every total choice B of C represents an assignment of values to all the random variables. A *probability* μ on a choice space C is a probability function on the set of all total choices of C. Intuitively, every probability μ is a probability distribution over the set of all variable assignments. Since C and all its alternatives are finite, μ can be defined by

- a mapping $\mu : \bigcup C \to [0,1]$ such that $\sum_{a \in A} \mu(a) = 1$ for all $A \in C$, and
- $\mu(B) = \Pi_{b \in B} \mu(b)$ for all total choices B of C.

Intuitively, the first condition defines a probability over the values of each random variable of C, and the second assumes independence between the random variables.

A *probabilistic dl-program* $KB = (L, P, C, \mu)$ consists of

- a dl-program (L, P),
- a choice space C such that (i) $\bigcup C \subseteq HB_\Phi$ and (ii) no atomic choice in C coincides with the head of any $r \in ground(P)$, and
- a probability μ on C.

Intuitively, since the total choices of C select subsets of P, and μ is a probability distribution on the total choices of C, every probabilistic dl-program is the compact encoding of a probability distribution on a finite set of normal dl-programs. Observe here that P is fully general and not necessarily stratified or acyclic. A *probabilistic query* has the form

$$\exists(\beta \mid \alpha)[r, s]\,,$$

where $\beta \mid \alpha$ is a conditional event, and r and s are variables.

Example 5. Consider the probabilistic dl-program $KB = (L, P, C, \mu)$, where L and P are as in Examples 1 and 2, except that the dl-rules (4), (5), and (11) are replaced by the dl-rules (4′), (5′), and (11′), respectively, and the dl-rule (13) is added:

$$\begin{array}{ll}
(4') & avoid(X) \leftarrow DL[PC \uplus pc; Electronics](X), not\ offer(X), avoid_pos; \\
(5') & offer(X) \leftarrow DL[PC \uplus pc; Electronics](X), not\ brand_new(X), offer_pos; \\
(11') & buyPC(X) \leftarrow DL[PC \uplus pc; PC](X), not\ avoid(X), not\ exclude(X), one_buy_pos; \\
(13) & buyAccessory(X) \leftarrow DL[Electronics](X), not\ avoid(X), buyPC(Y), \\
 & \qquad\qquad not\ DL[PC \uplus pc; PC](X), acc_buy_pos.
\end{array}$$

Let C be defined as

$$\{\{avoid\_pos, avoid\_neg\}, \{offer\_pos, offer\_neg\}, \\ \{one\_buy\_pos, one\_buy\_neg\}, \{acc\_buy\_pos, acc\_buy\_neg\}\},$$

and let μ be given as follows:

- $\mu(avoid\_pos) = 0.7$, $\mu(avoid\_neg) = 0.3$,
- $\mu(offer\_pos) = 0.7$, $\mu(offer\_neg) = 0.3$,
- $\mu(one\_buy\_pos) = 0.95$, $\mu(one\_buy\_neg) = 0.05$,
- $\mu(acc\_buy\_pos) = 0.8$, $\mu(acc\_buy\_neg) = 0.2$.

The new dl-rules (4′) and (5′) express that the original dl-rules (4) and (5) now only hold with probability 0.7. Furthermore, (11′) expresses that a customer buys a single PC with probability 0.95, while rule (13) states that a customer buying a PC may choose to also buy a non-PC electronics item as an accessory with probability 0.8.

In a probabilistic query, one may ask for the tight probability bounds that a customer buys a webcam w, if (i) PC p is bought, (ii) p is on offer, and (iii) w is not on offer; the result to this query may, for instance, help to decide whether it is useful to automatically show a product w to a customer buying the product p):

$$\exists(buyAccessory(w) \mid \neg offer(w) \wedge buyPC(p) \wedge offer(p))[R, S]\,.$$

4.2 Answer Set Semantics

We next define an answer set semantics of probabilistic dl-programs, the consistency of such programs, and tight answers to probabilistic queries.

Given a probabilistic dl-program $KB=(L,P,C,\mu)$, a *probabilistic interpretation* Pr is a probability function on the set of all $I \subseteq HB_\Phi$. We say that Pr is an *answer set* of KB iff the following two conditions hold:

- every interpretation $I \subseteq HB_\Phi$ with $Pr(I)>0$ is an answer set of the dl-program $(L,P \cup \{p \leftarrow \mid p \in B\})$ for some total choice B of C (which implies $B \subseteq I$), and
- $Pr\left(\bigwedge_{p\in B} p\right) = \mu(B)$ for every total choice B of C.

Informally, these conditions state that Pr is an answer set of $KB=(L,P,C,\mu)$ iff (i) every interpretation $I \subseteq HB_\Phi$ of positive probability under Pr is an answer set of the dl-program (L,P) under some total choice B of C, and (ii) Pr coincides with μ on the total choices B of C. We say KB is *consistent* iff it has an answer set Pr.

Given a ground event α, the *probability* of α in a probabilistic interpretation Pr, denoted $Pr(\alpha)$, is the sum of all $Pr(I)$ such that $I \subseteq HB_\Phi$ and $I \models \alpha$. We say that $(\beta|\alpha)[l,u]$ (where $l,u \in [0,1]$) is a *tight consequence* of a consistent probabilistic dl-program KB under the answer set semantics iff l (resp., u) is the infimum (resp., supremum) of $Pr(\alpha \wedge \beta)\,/\,Pr(\alpha)$ subject to all answer sets Pr of KB with $Pr(\alpha)>0$ (note that this infimum (resp., supremum) is naturally defined as 1 (resp., 0) iff no such Pr exists). The *tight answer* for a probabilistic query $Q=\exists(\beta|\alpha)[r,s]$ to KB under the answer set semantics is the set of all ground substitutions θ (for the variables in Q) such that $(\beta|\alpha)[r,s]\theta$ is a tight consequence of KB under the answer set semantics. For ease of presentation, since the tight answers for probabilistic queries of the form $Q=\exists(\beta|\alpha)[r,s]$ with non-ground $\beta|\alpha$ can be reduced to the tight answers for probabilistic queries $Q=\exists(\beta|\alpha)[r,s]$ with ground $\beta|\alpha$, we consider only the latter type of probabilistic queries from now on.

5 Novel Answer Set Characterizations

In this section, we give novel characterizations of (i) the consistency of probabilistic dl-programs and (ii) tight query processing in consistent probabilistic dl-programs under the answer set semantics in terms of the answer sets of normal dl-programs.

As shown in [53], a probabilistic dl-program $KB=(L,P,C,\mu)$ is consistent iff the system of linear constraints $LC_\top$ (see Fig. 2) over y_r $(r \in R)$ is solvable. Here, 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. Observe, however, that $LC_\top$ is defined over a set of variables R that corresponds to the set of all answer sets of the underlying normal dl-programs, and thus R is in general quite large.

Example 6. Consider the probabilistic dl-program $KB=(L,P,C,\mu)$, where L and P are as in Examples 1 and 2, except that the dl-rules (4) and (11) are replaced by the following dl-rules (4′) and (11′), respectively:

(4′) $avoid(X) \leftarrow DL[PC \uplus pc; PC](X), not\ offer(X), avoid\_pos;$
(11′) $buyPC(X) \leftarrow DL[PC \uplus pc; PC](X), not\ avoid(X), not\ exclude(X), buy\_pos.$

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

Fig. 2 System of linear constraints LC_α

The choice space C is defined as

$$\{\{avoid\_pos, avoid\_neg\}, \{buy\_pos, buy\_neg\}\},$$

and μ is given as follows:

- $\mu(avoid\_pos) = 0.5$, $\mu(avoid\_neg) = 0.5$,
- $\mu(buy\_pos) = 0.8$, $\mu(buy\_neg) = 0.2$.

We then obtain four total choices:

$$\{avoid\_pos, buy\_pos\}, \{avoid\_pos, buy\_neg\},$$
$$\{avoid\_neg, buy\_pos\}, \{avoid\_neg, buy\_neg\}.$$

The set of linear constraints $LC_\top$ for KB then comprises four constraints corresponding to each of the total choices, one constraint corresponding to the sum to 1, and one constraint for each variable, expressing its non-negativeness. There is one variable for each possible answer set given all possible total choices. Intuitively, we can see that multiple answer sets can arise only if *buy_pos* is true. The case where *avoid_pos* is also true was investigated in Example 3, where we saw that there are three answer sets in this situation. The remaining case corresponds to *avoid_neg* being true; here, rule $(4')$ no longer applies, and therefore rule $(11')$'s *not avoid*(X) atom will hold for all answers to the query in the dl-atom $DL[PC \uplus pc; PC](X)$, which contains five objects: pc_1, pc_2, pc_3, *pc_hp*, and *pc_ibm*. Therefore, we have five more answer sets in this case, and thus a total of 10 variables in $LC_\top$.

The following theorem shows that the consistency of probabilistic dl-programs can be expressed in terms of answer sets of normal dl-programs only, without having to additionally decide whether or not a system of linear constraints is solvable.

Theorem 1 (Consistency). *Let $KB = (L, P, C, \mu)$ be a probabilistic dl-program. Then, KB is consistent iff, for every total choice B of C such that $\mu(B) > 0$, the dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent.*

Similarly, as shown in [53], computing tight answers for probabilistic queries can be reduced to computing all answer sets of normal dl-programs and solving two linear optimization problems. More specifically, let $KB = (L, P, C, \mu)$ be a consistent probabilistic dl-program, and let $Q = \exists(\beta \mid \alpha)[r, s]$ be a probabilistic query with ground $\beta \mid \alpha$. Then, the tight answer for Q to KB is given by $\theta = \{r/l, s/u\}$, where l (resp.,

u) is the optimal value of the subsequent linear program (3) over y_r $(r \in R)$, if (3) has a solution, and it is given by $\theta = \{r/1, s/0\}$, if (3) has no solution.

$$\min \text{ (resp., max) } \textstyle\sum_{r \in R, r \models \alpha \wedge \beta} y_r \quad \text{subject to } LC_\alpha \text{ (see Fig. 2).} \tag{3}$$

But the linear program (3) is defined over the same (generally quite large) set of variables as the system of linear constraints $LC_\top$ above. The following theorem shows that the tight answers can also be expressed in terms of answer sets of normal dl-programs only, without additionally solving two linear optimization problems.

Theorem 2 (Tight Query Processing). *Let $KB = (L, P, C, \mu)$ be a consistent probabilistic dl-program, and let $Q = \exists(\beta|\alpha)[r,s]$ be a probabilistic query with ground $\beta|\alpha$. Let a (resp., b) be the sum of all $\mu(B)$ such that (i) B is a total choice of C and (ii) $\alpha \wedge \beta$ is true in every (resp., some) answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$. Let c (resp., d) be the sum of all $\mu(B)$ such that (i) B is a total choice of C and (ii) $\alpha \wedge \neg\beta$ is true in every (resp., some) answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$. Then, the tight answer θ for Q to KB under the answer set semantics is given as follows:*

$$\theta \;=\; \begin{cases} \{r/1, s/0\} & \text{if } b = 0 \text{ and } d = 0; \\ \{r/0, s/0\} & \text{if } b = 0 \text{ and } d \neq 0; \\ \{r/1, s/1\} & \text{if } b \neq 0 \text{ and } d = 0; \\ \{r/\frac{a}{a+d}, s/\frac{b}{b+c}\} & \text{otherwise.} \end{cases}$$

6 Total Well-Founded Semantics

In this section, we define a novel well-founded semantics for probabilistic dl-programs, called the *total well-founded semantics*, since it is defined for all probabilistic queries to probabilistic dl-programs, as opposed to the well-founded semantics of [53], which is only defined for a very limited class of probabilistic queries. Furthermore, the total well-founded semantics is defined for all probabilistic dl-programs, as opposed to the answer set semantics, which is only defined for consistent ones.

More concretely, given a probabilistic dl-program $KB = (L, P, C, \mu)$ and a probabilistic query $Q = \exists(\beta|\alpha)[r,s]$ with ground $\beta|\alpha$, the tight answer θ for Q to KB under the well-founded semantics of [53] *exists* iff both ground events $\alpha \wedge \beta$ and α are defined in every $S = WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ such that B is a total choice of C. Here, a ground event ϕ is *defined* in S iff either $I \models \phi$ for every interpretation $I \subseteq HB_\Phi$ such that (i) $S \cap HB_\Phi \subseteq I$ and (ii) $\neg S \cap I = \emptyset$, or $I \not\models \phi$ for every interpretation $I \subseteq HB_\Phi$ such that (i) $S \cap HB_\Phi \subseteq I$ and (ii) $\neg S \cap I = \emptyset$. If α is false in every $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ such that B is a total choice of C, then the tight answer is defined as $\theta = \{r/1, s/0\}$; otherwise, the tight answer (if it exists) is defined as $\theta = \{r/\frac{u}{v}, s/\frac{u}{v}\}$, where u (resp., v) is the sum of all $\mu(B)$ such that:

- B is a total choice of C, and
- $\alpha \wedge \beta$ (resp., α) is true in $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$.

We define the total well-founded semantics as follows, taking inspiration from the novel answer set characterization of tight answers in the previous section.

Definition 1 (Total Well-Founded Semantics). Let $KB=(L,P,C,\mu)$ be a probabilistic dl-program, and let $Q=\exists(\beta|\alpha)[r,s]$ be a probabilistic query with ground $\beta|\alpha$. Let a (resp., b^-) be the sum of all $\mu(B)$ such that (i) B is a total choice of C and (ii) $\alpha\wedge\beta$ is true (resp., false) in $WFS(L,P\cup\{p\leftarrow\,|\,p\in B\})$. Let c (resp., d^-) be the sum of all $\mu(B)$ such that (i) B is a total choice of C and (ii) $\alpha\wedge\neg\beta$ is true (resp., false) in $WFS(L,\,P\cup\{p\leftarrow\,|\,p\in B\})$. Let $b=1-b^-$ and $d=1-d^-$. Then, *the tight answer θ for Q to KB under the total well-founded semantics* (denoted $TWFS(KB)$) is defined as follows:

$$\theta \;=\; \begin{cases} \{r/1,\,s/0\} & \text{if } b=0 \text{ and } d=0;\\ \{r/0,\,s/0\} & \text{if } b=0 \text{ and } d\neq 0;\\ \{r/1,\,s/1\} & \text{if } b\neq 0 \text{ and } d=0;\\ \{r/\frac{a}{a+d},\,s/\frac{b}{b+c}\} & \text{otherwise.} \end{cases}$$

The following theorem shows that for probabilistic queries $Q=\exists(\ell)[r,s]$, where ℓ is a ground literal, the tight answers under the total well-founded semantics approximate the tight answers under the answer set semantics (if they exist). This is a nice semantic feature of the total well-founded semantics. It allows for an efficient approximation of tight answers to such queries under the answer set semantics by the bottom-up fixpoint iteration of the well-founded semantics of normal dl-programs.

Theorem 3. *Let $KB=(L,P,C,\mu)$ be a consistent probabilistic dl-program, and let $Q=\exists(\ell)[r,s]$ be a probabilistic query with ground literal ℓ. Let $\theta=\{r/l,\,s/u\}$ (resp., $\theta'=\{r/l',\,s/u'\}$) be the tight answer for Q to KB under the total well-founded semantics (resp., answer set semantics). Then, $[l',u']\subseteq[l,u]$.*

The next theorem shows that the total well-founded semantics generalizes the well-founded semantics of [53], i.e., the tight answers under the former coincide with the tight answers under the latter, if the tight answers under the latter exist.

Theorem 4. *Let $KB=(L,P,C,\mu)$ be a probabilistic dl-program, and let $Q=\exists(\beta\,|\,\alpha)[r,s]$ be a probabilistic query with ground $\beta|\alpha$. Then, the tight answer for Q to KB under the total well-founded semantics coincides with the tight answer for Q to KB under the well-founded semantics of [53] (if it exists).*

7 Algorithms and Complexity

In this section, we provide an anytime algorithm for tight query processing in probabilistic dl-programs under the total well-founded semantics, and conclude with tractability and complexity results.

7.1 An Anytime Algorithm for Tight Query Processing

By Definition 1, computing the tight answer for a probabilistic query to a probabilistic dl-program $KB = (L, P, C, \mu)$ under $TWFS(KB)$ can be reduced to computing the well-founded models of all normal dl-programs $(L, P \cup \{p \leftarrow \mid p \in B\})$ such that B is a total choice of C. Here, the number of all total choices B is generally a non-neglectable source of complexity. We thus propose:

- to compute the tight answer only up to an error within a given threshold $\varepsilon \in [0,1]$,
- to process the B's along decreasing probabilities $\mu(B)$, and
- to eventually stop the computation after a given time interval.

Given a (not necessarily consistent) probabilistic dl-program $KB = (L, P, C, \mu)$, a probabilistic query $Q = \exists(\beta|\alpha)[r,s]$ with ground $\beta|\alpha$, and an error threshold $\varepsilon \in [0,1]$, algorithm tight_answer (see Fig. 3) computes some $\theta = \{r/l', s/u'\}$ such that $|l - l'| + |u - u'| \leqslant \varepsilon$, where $\{r/l, s/u\}$ is the tight answer for Q to KB under $TWFS(KB)$. More concretely, it computes the bounds l' and u' by first initializing the variables a, b, c, and d (which play the same role as in Definition 1). It then computes the well-founded semantics S of the normal dl-program $(L, P \cup \{p \leftarrow \mid p \in B_i\})$ for every total choice B_i of C, checks whether $\alpha \wedge \beta$ and $\alpha \wedge \neg\beta$ are true or false in S, and updates a, b, c, and d accordingly. If the possible error in the bounds falls below ε, then it stops and returns the bounds computed so far. Thus, in the special case where $\varepsilon = 0$, the algorithm computes in particular the tight answer for Q to KB under $TWFS(KB)$. The following theorem shows that algorithm tight_answer is sound.

Theorem 5. *Let KB be a probabilistic dl-program, let $Q = \exists(\beta|\alpha)[r,s]$ be a probabilistic query with ground $\beta|\alpha$, and let $\theta = \{r/l, s/u\}$ be the tight answer for Q to KB under $TWFS(KB)$. Let $\theta' = \{r/l', s/u'\}$ be the output computed by tight_answer for the error threshold $\varepsilon \in [0,1]$. Then, $|l - l'| + |u - u'| \leqslant \varepsilon$.*

Algorithm tight_answer is actually an *anytime algorithm*, since we can always interrupt it, and return the bounds computed thus far. The following theorem shows that these bounds deviate from the tight bounds with an exactly measurable error (note that the possible error is also decreasing along the iterations of the while-loop). For this reason, algorithm tight_answer also iterates through the total choices B_i of C in a way such that the probabilities $\mu(B_i)$ are decreasing, so that the error in the computed bounds is very likely to be low already after few iteration steps.

Theorem 6. *Let KB be a probabilistic dl-program, let $Q = \exists(\beta|\alpha)[r,s]$ be a probabilistic query with ground $\beta|\alpha$, let $\varepsilon \in [0,1]$ be an error threshold, and let $\theta = \{r/l, s/u\}$ be the tight answer for Q to KB under $TWFS(KB)$. Suppose we run tight_answer on KB, Q, and ε, and we interrupt it after line (9). Let the returned $\theta' = \{r/l', s/u'\}$ be as specified in lines (11) to (14). Then, if $v = 0$, then $\theta = \theta'$. Otherwise,*

$$|l - l'| + |u - u'| \leqslant \frac{v}{a+d} + \frac{v}{b+c}.$$

The algorithm is based on two finite fixpoint iterations for computing the well-founded semantics of normal dl-programs, which are in turn based on a finite fixpoint

Algorithm tight_answer

Input: probabilistic dl-program $KB=(L,P,C,\mu)$, probabilistic query $Q=\exists(\beta|\alpha)[r,s]$ with ground $\beta|\alpha$, and error threshold $\varepsilon\in[0,1]$.

Output: $\theta=\{r/l', s/u'\}$ such that $|l-l'|+|u-u'|\leqslant\varepsilon$, where $\{r/l, s/u\}$ is the tight answer for Q to KB under the total well-founded semantics.

Notation: $B_1,\ldots,B_k$ is a sequence of all total choices B of C with $\mu(B_1)\geqslant\cdots\geqslant\mu(B_k)$.

1. $a:=0$; $b:=1$; $c:=0$; $d:=1$; $v:=1$; $i:=1$;
2. **while** $i\leqslant k$ and $v>0$ and $\frac{v}{a+d}+\frac{v}{b+c}>\varepsilon$ **do begin**
3. $S:=WFS(L,P\cup\{p\leftarrow\mid p\in B_i\})$;
4. **if** $\alpha\wedge\beta$ is true in S **then** $a:=a+\mu(B_i)$
5. **else if** $\alpha\wedge\beta$ is false in S **then** $b:=b-\mu(B_i)$;
6. **if** $\alpha\wedge\neg\beta$ is true in S **then** $c:=c+\mu(B_i)$
7. **else if** $\alpha\wedge\neg\beta$ is false in S **then** $d:=d-\mu(B_i)$;
8. $v:=v-\mu(B_i)$;
9. $i:=i+1$
10. **end**;
11. **if** $b=0$ and $d=0$ **then return** $\theta=\{r/1, s/0\}$
12. **else if** $b=0$ and $d\neq 0$ **then return** $\theta=\{r/0, s/0\}$
13. **else if** $b\neq 0$ and $d=0$ **then return** $\theta=\{r/1, s/1\}$
14. **else return** $\theta=\{r/\frac{a}{a+d}, s/\frac{b}{b+c}\}$.

Fig. 3 Algorithm tight_answer

iteration for computing the least model of positive dl-programs. More specifically, to compute the well-founded semantics of KB, i.e.,

$$WFS(KB)=lfp(\gamma_{KB}^2)\cup\neg(HB_\Phi-gfp(\gamma_{KB}^2)),$$

we compute $lfp(\gamma_{KB}^2)$ and $gfp(\gamma_{KB}^2)$ as the limits of the two finite fixpoint iterations

$$U_0=\emptyset,\text{ and }U_{i+1}=\gamma_{KB}^2(U_i),\text{ for }i\geqslant 0,\text{ and}$$
$$O_0=HB_\Phi,\text{ and }O_{i+1}=\gamma_{KB}^2(O_i),\text{ for }i\geqslant 0,$$

respectively. Here, the operator γ_{KB}, which is defined by $\gamma_{KB}(I)=M_{KB^I}$ (with $KB^I=(L,P_L^I)$) for all $I\subseteq HB_\Phi$, is computed as the limit of the finite fixpoint iteration

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

since $\gamma_{KB}(I)=lfp(T_{KB^I})$ for all $I\subseteq HB_\Phi$, where T_{KB^I} is the immediate consequence operator for positive dl-programs, which is defined as follows for every $J\subseteq HB_\Phi$:

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

All the above three fixpoint iterations are finite, since HB_Φ is finite.

7.2 Complexity

The following theorem shows that tight query processing in probabilistic dl-programs $KB = (L, P, C, \mu)$ in *DL-Lite* (i.e., L is in *DL-Lite*) under $TWFS(KB)$ can be done in polynomial time in the data complexity. This follows from Theorem 5 and the polynomial data complexity of (a) computing the well-founded semantics of a normal dl-program (see above) and of (b) conjunctive query processing in *DL-Lite*. Here, $|C|$ is bounded by a constant, since C and μ define the probabilistic information of P, which is fixed as a part of the program in P, while the ordinary facts in P (along with the concept and role membership axioms in L) are the variable input.

Theorem 7. *Given a probabilistic dl-program KB in DL-Lite and a probabilistic query $Q = \exists(\ell)[r,s]$ with ground literal ℓ, the tight answer $\theta = \{r/l, s/u\}$ for Q to KB under $TWFS(KB)$ can be computed in polynomial time in the data complexity.*

The next theorem shows that computing tight answers is EXP-complete in the combined complexity. The lower bound follows from the EXP-hardness of Datalog in the combined complexity, and the upper bound follows from Theorem 5.

Theorem 8. *Given a probabilistic dl-program KB in DL-Lite and a probabilistic query $Q = \exists(\beta|\alpha)[r,s]$ with ground $\beta|\alpha$, computing the tight answer $\theta = \{r/l, s/u\}$ for Q to KB under $TWFS(KB)$ is* EXP*-complete in the combined complexity.*

8 Probabilistic Data Integration

Integrating data from different sources is a crucial issue in the Semantic Web. In this section, we show how probabilistic dl-programs can be employed as a formalism for data integration in the Semantic Web. We first give some general definitions.

A *data integration system* (in its most general form, see [47]) $I = (G, S, M)$ consists of the following components:

- a *global* (or *mediated*) *schema* G, which represents the domain of interest,
- a *source schema* S, which represents the data sources of the system, and
- a *mapping* M, which relates the source schema and the global schema.

Here, G is purely virtual, while the data are stored in S. The mapping M can be specified in different ways, which is a crucial aspect in a data integration system. In particular, when every data structure in G is defined through a view over S, the mapping is said to be *GAV (global-as-view)*, while when every data structure in S is defined through a view over G the mapping is *LAV (local-as-view)*. A mixed approach, called *GLAV* [28, 12], associates views over G to views over S.

8.1 Modeling Data Integration Systems

In our framework, we assume that the global schema G, the source schema S, and the mapping M are each encoded by a probabilistic dl-program. More formally, we partition the vocabulary Φ into the sets Φ_G, Φ_S, and Φ_c such that:

- the symbols in Φ_G are of arity at least 1 and represent the global predicates,
- the symbols in Φ_S are of arity at least 1 and represent source predicates, and
- the symbols in Φ_c are constants.

Let $\mathbf{A}_G$ and $\mathbf{R}_G$ be disjoint denumerable sets of atomic concepts and abstract roles, respectively, for the global schema, and let $\mathbf{A}_S$ and $\mathbf{R}_S$ (disjoint from $\mathbf{A}_G$ and $\mathbf{R}_G$) be similar sets for the source schema. We also assume a denumerable set of individuals $\mathbf{I}$ that is disjoint from the set of all concepts and roles and a superset of Φ_c. A *probabilistic data integration system* $PI = (KB_G, KB_S, KB_M)$ consists of a probabilistic dl-program $KB_G = (L_G, P_G, C_G, \mu_G)$ for the global schema, a probabilistic dl-program $KB_S = (L_S, P_S, C_S, \mu_S)$ for the source schema, and a probabilistic dl-program $KB_M = (\emptyset, P_M, C_M, \mu_M)$ for the mapping:

- KB_G (resp., KB_S) is defined over the predicates, constants, concepts, roles, and individuals of the global (resp., source) schema, and it encodes ontological, rule-based, and probabilistic relationships in the global (resp., source) schema.
- KB_M is defined over the predicates, constants, concepts, roles, and individuals of the global and the source schema, and it encodes a probabilistic mapping between the predicates, concepts, and roles of the source and those of the global schema.

Our probabilistic dl-rules permit a specification of the mapping that can freely use global and source predicates together in rules, thus having a formalism that generalizes LAV and GAV in some way. Moreover, with a simple technicality, we are able to partly model GLAV systems. In GLAV data integration systems, the mapping is specified by means of rules of the form $\psi \leftarrow \varphi$, where ψ is a conjunction of atoms of G, and φ is a conjunction of atoms of S. We introduce an auxiliary atom α that contains all the variables of ψ; moreover, let $\psi = \beta_1 \wedge \ldots \wedge \beta_m$. We model the GLAV mapping rule with the following rules:

$$\begin{aligned} \beta_1 &\leftarrow \alpha; \\ &\vdots \\ \beta_m &\leftarrow \alpha; \\ \alpha &\leftarrow \varphi. \end{aligned}$$

What our framework does not allow is having rules that are *unsafe*, i.e., having existentially-quantified variables in their head.

Note also that correct and tight answers to probabilistic queries on the global schema are formally defined relative to the probabilistic dl-program $KB = (L, P, C, \mu)$, where $L = L_G \cup L_S$, $P = P_G \cup P_S \cup P_M$, $C = C_G \cup C_S \cup C_M$, and $\mu = \mu_G \cdot \mu_S \cdot \mu_M$. Informally, KB is the result of merging KB_G, KB_S, and KB_M. In a similar way, the probabilistic dl-program KB_S of the source schema S can be defined by merging the probabilistic dl-programs $KB_{S_1}, \ldots, KB_{S_1}$ of $n \geqslant 1$ source schemas $S_1, \ldots, S_n$.

The fact that the mapping is probabilistic allows for a high flexibility in the treatment of the uncertainty that is present when pieces of data come from heterogeneous sources whose content may be inconsistent and/or redundant relative to the global schema G, which in general incorporates constraints. Some different types of probabilistic mappings that can be modeled in our framework are summarized below.

8.2 Types of Probabilistic Mappings

In addition to expressing probabilistic knowledge about the global schema and about the source schema, the probabilities in probabilistic dl-programs can especially be used for specifying the probabilistic mapping in the data integration process. We distinguish three different types of probabilistic mappings, depending on whether the probabilities are used as *trust*, *error*, or *mapping probabilities*.

The simplest way of probabilistically integrating several data sources is to weight each data source with a *trust probability* (which all sum up to 1). This is especially useful when several redundant data sources are to be integrated. In such a case, pieces of data from different data sources may easily be inconsistent with each other.

Example 7. Suppose that we want to obtain a weather forecast for a certain place by integrating the potentially different weather forecasts of several weather forecast institutes. For ease of presentation, suppose that we only have three weather forecast institutes A, B, and C. In general, one trusts certain weather forecast institutes more than others. In our case, we suppose that our trust in the institutes A, B, and C is expressed by the trust probabilities 0.6, 0.3, and 0.1, respectively. That is, we trust most in A, medium in B, and less in C. In general, the different institutes do not use the same data structure to represent their weather forecast data. For example, institute A may use a single relation

$$forecast(place, date, weather, temperature, wind)$$

to store all the data, while B may have one relation

$$forecast\_place(date, weather, temperature, wind)$$

for each place, and C may use several different relations

$$forecast\_weather(place, date, weather),$$
$$forecast\_temperature(place, date, temperature),$$
$$forecast\_wind(place, date, wind).$$

Suppose the global schema G has the relation

$$forecast\_rome\_global(date, weather, temperature, wind),$$

which may for instance be posted on the Web by the tourist information of the city of Rome. The probabilistic mapping of the source schemas of A, B, and C to the global schema G can then be specified by the following $KB_M = (\emptyset, P_M, C_M, \mu_M)$:

$$\begin{aligned} P_M = \{ & forecast\_rome\_global(D,W,T,M) \leftarrow forecast(rome,D,W,T,M),\ inst_A; \\ & forecast\_rome\_global(D,W,T,M) \leftarrow forecast\_rome(D,W,T,M),\ inst_B; \\ & forecast\_rome\_global(D,W,T,M) \leftarrow forecast\_weather(rome,D,W), \\ & \quad forecast\_temperature(rome,D,T), forecast\_wind(rome,D,M),\ inst_C\}; \\ C_M = \{ & \{inst_A, inst_B, inst_C\}\}; \\ \mu_M : \ & inst_A, inst_B, inst_C \mapsto 0.6, 0.3, 0.1\,. \end{aligned}$$

The mapping assertions state that the first, second, and third rule above hold with the probabilities 0.6, 0.3, and 0.1, respectively. This is motivated by the fact that three institutes may generally provide conflicting weather forecasts, and our trust in A, B, and C are given by the trust probabilities 0.6, 0.3, and 0.1, respectively.

A more complex way of probabilistically integrating several data sources is to associate each data source (or each derivation) with an *error probability*.

Example 8. Suppose that we want to integrate the data provided by the different sensors in a sensor network. For example, assume a sensor network measuring the concentration of ozone in several different positions of a certain town, which may for instance be the basis for the common hall to reduce or forbid individual traffic. Suppose that each sensor $i \in \{1,\ldots,n\}$ with $n \geqslant 1$ is associated with its position through $sensor(i,position)$ and provides its measurement data in a single relation $reading_i(date,time,type,result)$. Each such reading may be erroneous with the probability e_i. That is, any tuple returned (resp., not returned) by a sensor $i \in \{1,\ldots,n\}$ may not hold (resp., may hold) with probability e_i. Let the global schema contain a single relation $reading(position,date,time,type,result)$. Then, the probabilistic mapping of the source schemas of the sensors $i \in \{1,\ldots,n\}$ to the global schema G can be specified by the following probabilistic dl-program $KB_M = (\emptyset, P_M, C_M, \mu_M)$:

$$
\begin{aligned}
P_M \;=\; & \{aux_i(P,D,T,K,R) \leftarrow reading_i(D,T,K,R), sensor(i,P) \mid i \in \{1,\ldots,n\}\} \cup \\
& \{reading(P,D,T,K,R) \leftarrow aux_i(P,D,T,K,R), not\_error_i \mid i \in \{1,\ldots,n\}\} \cup \\
& \{reading(P,D,T,K,R) \leftarrow not\, aux_i(P,D,T,K,R), error_i \mid i \in \{1,\ldots,n\}\}; \\
C_M \;=\; & \{\{error_i, not\_error_i\} \mid i \in \{1,\ldots,n\}\}; \\
\mu_M :\; & error_1, not\_error_1, \ldots, error_n, not\_error_n \;\mapsto\; e_1, 1-e_1, \ldots, e_n, 1-e_n .
\end{aligned}
$$

Note that if there are two sensors j and k for the same position, and they both return the same tuple as a reading, then this reading is correct with the probability $1-e_je_k$ (since it may be erroneous with the probability e_je_k). Note also that this modeling assumes that the errors of the sensors are independent from each other, which can be achieved by eventually unifying atomic choices. For example, if the sensor j depends on the sensor k, then j is erroneous when k is erroneous, and thus the atomic choices $\{error_j, not\_error_j\}$ and $\{error_k, not\_error_k\}$ are merged into the new atomic choice $\{error_j error_k, not\_error_j error_k, not\_error_j not\_error_k\}$.

When integrating several data sources, it may be the case that the relationships between the source schema and the global schema are *purely probabilistic*.

Example 9. Suppose that we want to integrate the schemas of two libraries, and that the global schema contains the predicate symbol *logic_programming*, while the source schemas contain only the concepts *rule-based_systems* and *deductive_databases* in their ontologies. These three concepts are overlapping to some extent, but they do not exactly coincide. For example, a randomly chosen book from *rule-based_systems* (resp., *deductive_databases*) may belong to *logic_programming* with

the probability 0.7 (resp., 0.8). The probabilistic mapping from the source schemas to the global schema can then be expressed by the following $KB_M = (\emptyset, P_M, C_M, \mu_M)$:

$$\begin{aligned} P_M &= \{logic\_programming(X) \leftarrow DL[rule\text{-}based\_systems(X)], choice_1; \\ &\quad logic\_programming(X) \leftarrow DL[deductive\_databases(X)], choice_2\}; \\ C_M &= \{\{choice_1, not\_choice_1\}, \{choice_2, not\_choice_2\}\}; \\ \mu_M &: \ choice_1, not\_choice_1, choice_2, not\_choice_2 \mapsto 0.7, 0.3, 0.8, 0.2. \end{aligned}$$

8.3 Deterministic Mappings on Probabilistic Data

Finally, we briefly describe an approach to use probabilistic dl-programs to model probabilistic data, such as those in [17].

Example 10. Suppose that the weather in Oxford can be sunny, cloudy, or rainy with probabilities 0.2, 0.45, and 0.35, respectively, and similar probabilities are assigned for other cities. This setting is analogous to the "classical" one of probabilistic data, where there is a probability distribution over ground facts. In such a case, the choice space is $C = \{\{weather(oxford, sunny), weather(oxford, cloudy), weather(oxford, rainy)\}, \ldots\}$, and the probability is μ: $weather(oxford, sunny), weather(oxford, cloudy), weather(oxford, rainy) \mapsto 0.2, 0.45, 0.35$. A mapping rule such as

$$candidate\_destination(L) \leftarrow weather(L, sunny)$$

can now express the fact that a destination is a candidate for a day-trip if it has sunny weather. While the mapping is purely deterministic, the probability distributions on the sets of atomic choices of the choice space enforce, by virtue of the mapping, a probability distribution on the ground facts of the global schema. Our framework is able to capture this situation, allowing for query answering over uncertain data.

9 Related Work

In this section, we discuss more closely related work on (a) the combination of logic programs, DLs, and probabilistic uncertainty, (b) the combination of logic programs and probabilistic uncertainty, (c) the combination of logic programs and DLs, and (d) the combination of DLs (or ontology languages) and probabilistic uncertainty. Note that more detailed overviews on uncertainty reasoning for the Semantic Web, which covers (a), (c), and (d), are given in [59, 56].

9.1 Probabilistic Description Logic Programs

To our knowledge, the work of [53] was the first one combining (normal) dl-programs (under the loose integration) with probabilistic uncertainty. Instead of being based on the loosely integrated normal dl-programs $KB = (L, P)$ of [25, 26], probabilistic dl-programs can also be developed as a generalization of the *tightly*

integrated ones in [50] (see [14, 57]). Rather than having dl-queries to L in rule bodies in P (which also allow for passing facts as dl-query arguments from P to L) and assuming that Φ and $\mathbf{A}$ (resp., $\mathbf{R}$) have no unary (resp., binary) predicate symbols in common (and so that dl-queries are the only interface between L and P), the tightly integrated normal dl-programs of [50] have no dl-queries, but Φ and $\mathbf{A}$ (resp., $\mathbf{R}$) may very well have unary (resp., binary) predicate symbols in common, and so the integration between L and P is of a much tighter nature. Nearly all the results of this paper carry over to such tightly integrated probabilistic dl-programs. As an important feature for the Semantic Web, they also allow for expressing in P probabilistic relations between the concepts and roles in L, since we can freely use concepts and roles from L as unary resp. binary predicate symbols in P.

The (loosely coupled) probabilistic fuzzy dl-programs in [58] combine fuzzy DLs, fuzzy logic programs (with stratified default-negation), and probabilistic uncertainty in a uniform framework for the Semantic Web. Intuitively, they allow for defining several rankings on ground atoms using fuzzy vagueness, and then for merging these rankings using probabilistic uncertainty (by associating with each ranking a probabilistic weight and building the weighted sum of all rankings). Less closely related, since they deal with fuzzy vagueness alone, rather than probabilistic ambiguity and imprecision, are the loosely and tightly coupled fuzzy dl-programs that have been introduced in [52] and [60], respectively, and extended by a top-k retrieval technique in [61]. Related works by Straccia combine (positive) dl-programs with lattice-based uncertainty [81] and with fuzzy vagueness [78].

9.2 Probabilistic Logic Programs

Ng and Subrahmanian [63, 62] proposed the first approach to probabilistic logic programs that addresses the problem of combining logic programs [49] with probability theory by adopting semantics in the style of Nilsson [65] and Halpern [35] for probabilistic logic. All semantics proposed for quantitative logic programming prior to this work had been non-probabilistic, of which [84] and [75] are examples; on the other hand, this approach aims at developing a probabilistic model theory and fixpoint theory. The general form of rules in their formalism is:

$$F_0 : \mu_0 \leftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n ,$$

where the F_i's are *basic formulas* (conjunctions or disjunctions of atoms) and the μ_i's are probabilistic annotations in the form of intervals that may contain expressions with variables. In [62], heads of rules are restricted to annotated atoms, where negation is still supported via $[0,0]$ annotations, but conditional probabilities are not expressible. The formalism is a general framework for expressing probabilistic information, and the authors study its semantics and relationship with probability theory, model theory, fixpoint theory, and proof theory. They also develop a procedure for answering queries about probabilities of events, which is different from query processing in classical logic programming, since most general unifiers are not always unique, and thus *maximally general* unifiers must be computed.

Other approaches to probabilistic logic programs have especially been proposed by Ngo and Haddawy [64], Lukasiewicz [54], Lukasiewicz and Kern-Isberner [41], Lakshmanan and Shiri [46], Dekhtyar and Subrahmanian [21], and Damasio et al. [19]. Ngo and Haddawy [64] present a model theory, fixpoint theory, and proof procedure for conditional probabilistic logic programming. Lukasiewicz [54] presents a conditional semantics for probabilistic logic programs where each rule is interpreted as specifying the conditional probability of the rule head, given the body. In closely related work, Lukasiewicz and Kern-Isberner [41] combine probabilistic logic programs (also adopting an explicit treatment of conditional probabilities) with maximum entropy, in relation to Nilsson's proposal for probabilistic logic [65]. Lakshmanan and Shiri [46] developed a semantics for logic programs in which different general axiomatic methods are given to compute probabilities of conjunctions and disjunctions, and these are used to define a semantics for probabilistic logic programs. In [21], Dekhtyar and Subrahmanian consider different conjunction and disjunction strategies, originally introduced by Lakshmanan et al. [45], and allow an explicit syntax in probabilistic logic programs so that users are able to express their knowledge of a dependency. Damasio et al. [19] present a well-founded semantics for annotated logic programs and show how to compute it.

Although there is a rich body of work on probabilistic logic programs, most such works to date have only addressed the problem of checking whether a given formula of the form $F : [\ell, u]$ is entailed by a probabilistic logic program [63, 62] or is true in a specific model (e.g., the well-founded model [19]). This usually boils down to finding out if all interpretations that satisfy the probabilistic logic program assign a probability between ℓ and u to F. An interesting extension of the concept of probabilistic entailment is proposed in [88], where the authors propose to go one step further and check to what *degree of satisfaction* the query is entailed by the program, similar to the novel approach of Bröcheler et al. [10], who propose to answer entailment queries with histograms indicating how the density of solutions are distributed in the probabilistic interval. In contrast, other works have recently focused on finding *most probable worlds* [43, 77], and answering *abductive queries* [76].

9.3 Description Logic Programs

Related work on the combination of logic programs and DLs can be divided into the following categories: (a) hybrid approaches using DLs as input to logic programs, (b) approaches reducing DLs to logic programs, (c) combinations of DLs with default and defeasible logic, and (d) approaches to rule-based well-founded reasoning in the Semantic Web. Below we give some representatives for these categories; further works and details are given in [25, 26, 50].

The works by Donini et al. [23], Levy and Rousset [48], and Rosati [73, 74] are representatives of hybrid approaches using DLs as input. Donini et al. [23] introduce a combination of (disjunction-, negation-, and function-free) Datalog with the DL $\mathcal{AL}$. An integrated knowledge base consists of a structural component in $\mathcal{AL}$ 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 [48] presents a combination of Horn rules with the DL $\mathcal{ALC}$. In contrast to Donini et al. [23], 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 [73] 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 DL reasoning to logic programming are the works by Van Belleghem et al. [6], Alsaç and Baral [1], Swift [82], Grosof et al. [34], and Hufstadt et al. [39]. Early work on dealing with default information in DLs 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 DLs 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 Damasio [18]. He aims at Prolog tools for implementing different semantics for RuleML [9]. 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 [71].

9.4 Probabilistic Description Logics and Ontology Languages

Probabilistic generalizations of the expressive DLs $\mathcal{SHOQ}(\mathbf{D})$, $\mathcal{SHIF}(\mathbf{D})$, and $\mathcal{SHOIN}(\mathbf{D})$ behind DAML+OIL, OWL Lite, and OWL DL, respectively, have been proposed by Giugno and Lukasiewicz [31] and Lukasiewicz [51]. They are based on lexicographic probabilistic reasoning. A companion paper [20] combines *DL-Lite* with Bayesian networks. In earlier works, Heinsohn [36] and Jaeger [40] present probabilistic extensions to the DL $\mathcal{ALC}$, which are essentially based on probabilistic reasoning in probabilistic logics. Koller et al. [44] present a probabilistic generalization of the CLASSIC DL, which uses Bayesian networks as underlying probabilistic reasoning formalism. Recently, an extension to the Datalog$^{\pm}$ family of ontology languages [13] has been developed in [32, 33] to add the capability of representing probabilistic uncertainty by means of an integration between Datalog$^{\pm}$ ontologies and Markov logic networks [72], focusing on scalability towards applications in data extraction and reasoning in the Web. Note that fuzzy DLs, such as the ones by Straccia [79, 80] are less closely related to probabilistic DLs, since they deal with fuzzy vagueness, rather than probabilistic ambiguity and imprecision.

Especially the works by Costa [16], Pool and Aikin [69], and Ding and Peng [22] present probabilistic generalizations of the Web ontology language OWL. In particular, Costa's work [16] is semantically based on multi-entity Bayesian networks, while [22] has a semantics in standard Bayesian networks. In closely related work, Fukushige [29] proposes a basic framework for representing probabilistic

relationships in RDF. Finally, Nottelmann and Fuhr [66] present pDAML+OIL, which is a probabilistic generalization of the Web ontology language DAML+OIL, along with a mapping to stratified probabilistic Datalog.

10 Conclusion

We have proposed tractable probabilistic dl-programs for the Semantic Web, which combine tractable DLs, normal programs under the answer set and the well-founded semantics, and probabilities. We have given novel reductions of tight query processing and deciding consistency in probabilistic dl-programs under the answer set semantics to the answer set semantics of the underlying normal dl-programs. Based on them, we have then introduced the total well-founded semantics for probabilistic dl-programs. Contrary to the previous answer set and well-founded semantics, it is defined for all probabilistic dl-programs and queries. Furthermore, tight (resp., tight literal) query processing under the total well-founded semantics coincides with (resp., approximates) tight (resp., tight literal) query processing under the previous well-founded (resp., answer set) semantics in all cases where the latter is defined. We have then presented an anytime algorithm for tight query processing in probabilistic dl-programs under the total well-founded semantics. Note that the novel reductions, the total well-founded semantics, and the anytime algorithm are not limited to *DL-Lite* as underlying DL; they hold for all probabilistic dl-programs on top of DLs with decidable conjunctive query processing. We have also shown that tight query processing in probabilistic dl-programs under the total well-founded semantics is possible in polynomial time in the data complexity and is complete for EXP in the combined complexity. Finally, we have described an application of probabilistic dl-programs in probabilistic data integration for the Semantic Web.

An interesting topic for future research is to investigate whether one can also develop an efficient top-k query technique for the presented probabilistic dl-programs: Rather than computing the tight probability interval for a given ground literal, such a technique returns k most probable ground instances of a given non-ground formula.

Acknowledgements. This work was partially supported by the European Research Council under the European Union's 7th Framework Programme (FP7/2007-2013)/ERC grant 246858 — DIADEM, the EPSRC grant EP/J008346/1 "PrOQAW: Probabilistic Ontological Query Answering on the Web", a Yahoo! Research Fellowship, and a Google Research Award. This article is a significantly extended and revised version of a paper that appeared in *Proc. SUM-2007* [55].

References

1. Alsaç, G., Baral, C.: Reasoning in description logics using declarative logic programming. Technical report, Arizona State University (2001)
2. Antoniou, G.: Nonmonotonic Rule Systems on Top of Ontology Layers. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 394–398. Springer, Heidelberg (2002)

3. Antoniou, G., Wagner, G.: Rules and Defeasible Reasoning on the Semantic Web. In: Schröder, M., Wagner, G. (eds.) RuleML 2003. LNCS, vol. 2876, pp. 111–120. Springer, Heidelberg (2003)
4. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. J. Autom. Reasoning 14(1), 149–180 (1995)
5. Baral, C., Gelfond, M., Rushton, J.N.: Probabilistic Reasoning With Answer Sets. In: Lifschitz, V., Niemelä, I. (eds.) LPNMR 2004. LNCS (LNAI), vol. 2923, pp. 21–33. Springer, Heidelberg (2003)
6. Belleghem, K.V., Denecker, M., Schreye, D.D.: A strong correspondence between description logics and open logic programming. In: Proc. ICLP 1997, pp. 346–360. MIT Press (1997)
7. Berners-Lee, T.: Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press (2003)
8. Berners-Lee, T., Fischetti, M.: Weaving the Web. Harper, San Francisco (1999)
9. Boley, H., Tabet, S., Wagner, G.: Design rationale of RuleML: A markup language for Semantic Web rules. In: Proc. SWWS 2001, pp. 381–401 (2001)
10. Broecheler, M., Simari, G.I., Subrahmanian, V.S.: Using Histograms to Better Answer Queries to Probabilistic Logic Programs. In: Hill, P.M., Warren, D.S. (eds.) ICLP 2009. LNCS, vol. 5649, pp. 40–54. Springer, Heidelberg (2009)
11. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. Autom. Reasoning 39(3), 385–429 (2007)
12. Calì, A.: Reasoning in Data Integration Systems: Why LAV and GAV Are Siblings. In: Zhong, N., Raś, Z.W., Tsumoto, S., Suzuki, E. (eds.) ISMIS 2003. LNCS (LNAI), vol. 2871, pp. 562–571. Springer, Heidelberg (2003)
13. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem (2012) (in press)
14. Calì, A., Lukasiewicz, T., Predoiu, L., Stuckenschmidt, H.: Tightly coupled probabilistic description logic programs for the Semantic Web. J. Data Sem. 12, 95–130 (2009)
15. da Costa, P.C.G., Laskey, K.B.: PR-OWL: A framework for probabilistic ontologies. In: Proc. FOIS 2006, pp. 237–249. IOS Press (2006)
16. da Costa, P.C.G.: Bayesian Semantics for the Semantic Web. PhD thesis, Fairfax, VA, USA (2005)
17. Dalvi, N., Suciu, D.: Management of probabilistic data: foundations and challenges. In: Proc. PODS 2007, pp. 1–12. ACM Press (2007)
18. Damasio, C. V.: The W^4 project (2002), `http://centria.di.fct.unl.pt/~cd/projectos/w4/index.htm`
19. Viegas Damásio, C., Moniz Pereira, L., Swift, T.: Coherent Well-founded Annotated Logic Programs. In: Gelfond, M., Leone, N., Pfeifer, G. (eds.) LPNMR 1999. LNCS (LNAI), vol. 1730, pp. 262–276. Springer, Heidelberg (1999)
20. d'Amato, C., Fanizzi, N., Lukasiewicz, T.: Tractable Reasoning with Bayesian Description Logics. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 146–159. Springer, Heidelberg (2008)
21. Dekhtyar, A., Subrahmanian, V.S.: Hybrid probabilistic programs. In: Proc. ICLP 1997, pp. 391–405. MIT Press (1997)
22. Ding, Z., Peng, Y.: A probabilistic extension to ontology language OWL. In: Proc. HICSS 2004, IEEE Computer Society (2004)
23. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: $\mathscr{AL}$-log: Integrating Datalog and description logics. J. Intell. Inf. Syst. 10(3), 227–252 (1998)

24. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: Proc. IJCAI 2005, pp. 90–96. Morgan Kaufmann (2005)
25. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artif. Intell. 172(12/13), 1495–1539 (2008)
26. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R.: Well-founded semantics for description logic programs in the Semantic Web. ACM Trans. Comput. Log. 12(2), 11 (2011)
27. Finzi, A., Lukasiewicz, T.: Structure-based causes and explanations in the independent choice logic. In: Proc. UAI 2003, pp. 225–232. Morgan Kaufmann (2003)
28. Friedman, M., Levy, A.Y., Millstein, T.D.: Navigational plans for data integration. In: Proc. AAAI 1999, pp. 67–73. AAAI Press (1999)
29. Fukushige, Y.: Representing probabilistic knowledge in the Semantic Web. In: Proceedings of the W3C Workshop on Semantic Web for Life Sciences (2004)
30. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generat. Comput. 9(3/4), 365–386 (1991)
31. Giugno, R., Lukasiewicz, T.: P-$\mathcal{SHOQ}(\mathbf{D})$: A Probabilistic Extension of $\mathcal{SHOQ}(\mathbf{D})$ for Probabilistic Ontologies in the Semantic Web. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 86–97. Springer, Heidelberg (2002)
32. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Answering Threshold Queries in Probabilistic Datalog+/– Ontologies. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 401–414. Springer, Heidelberg (2011)
33. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Conjunctive query answering in probabilistic datalog+/– ontologies. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 77–92. Springer, Heidelberg (2011)
34. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logic. In: Proc. WWW 2003, pp. 48–57. ACM Press (2003)
35. Halpern, J.Y.: An analysis of first-order logics of probability. Artif. Intell. 46(3), 311–350 (1990)
36. Heinsohn, J.: Probabilistic description logics. In: Proc. UAI 1994, pp. 311–318. Morgan Kaufmann (1994)
37. Horrocks, I., Patel-Schneider, P.F.: Reducing OWL entailment to description logic satisfiability. J. Web Sem. 1(4), 345–357 (2004)
38. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The making of a Web ontology language. J. Web Sem. 1(1), 7–26 (2003)
39. Hufstadt, U., Motik, B., Sattler, U.: Reasoning for description logics around $\mathcal{SHIQ}$ in a resolution framework. Technical report, FZI Karlsruhe (2004)
40. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Proc. KR 1994, pp. 305–316. Morgan Kaufmann (1994)
41. Kern-Isberner, G., Lukasiewicz, T.: Combining probabilistic logic programming with the power of maximum entropy. Artif. Intell. 157(1/2), 139–202 (2004)
42. Kersting, K., De Raedt, L.: Bayesian logic programs. Technical report (2001)
43. Khuller, S., Martinez, M.V., Nau, D.S., Simari, G.I., Sliva, A., Subrahmanian, V.S.: Computing most probable worlds of action probabilistic logic programs: Scalable estimation for $10^{30,000}$ worlds. Ann. Math. Artif. Intell. 51(2-4), 295–331 (2007)
44. Koller, D., Levy, A.Y., Pfeffer, A.: P-CLASSIC: A tractable probabilistic description logic. In: Proc. AAAI 1997, pp. 390–397. AAAI Press, MIT Press (1997)

45. Lakshmanan, L.V.S., Leone, N., Ross, R., Subrahmanian, V.S.: ProbView: A flexible probabilistic database system. ACM Trans. Database Syst. 22(3), 419–469 (1997)
46. Lakshmanan, L.V.S., Shiri, N.: A parametric approach to deductive databases with uncertainty. IEEE Trans. Knowl. Data Eng. 13(4), 554–570 (2001)
47. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. PODS 2002, pp. 233–246. ACM Press (2002)
48. Levy, A.Y., Rousset, M.-C.: Combining Horn rules and description logics in CARIN. Artif. Intell. 104(1/2), 165–209 (1998)
49. Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer (1987)
50. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the Semantic Web. IEEE Trans. Knowl. Data Eng. 22(11), 1577–1592 (2010)
51. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. 172(6/7), 852–883 (2008)
52. Lukasiewicz, T.: Fuzzy description logic programs under the answer set semantics for the Semantic Web. Fundam. Inform. 82(3), 289–310 (2008)
53. Lukasiewicz, T.: Probabilistic description logic programs. Int. J. Approx. Reason. 45(2), 288–307 (2007)
54. Lukasiewicz, T.: Probabilistic logic programming with conditional constraints. ACM Trans. Comput. Log. 2(3), 289–339 (2001)
55. Lukasiewicz, T.: Tractable Probabilistic Description Logic Programs. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 143–156. Springer, Heidelberg (2007)
56. Lukasiewicz, T.: Uncertainty Reasoning for the Semantic Web. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 26–39. Springer, Heidelberg (2009)
57. Lukasiewicz, T., Predoiu, L., Stuckenschmidt, H.: Tightly integrated probabilistic description logic programs for representing ontology mappings. Ann. Math. Artif. Intell. (2011)
58. Lukasiewicz, T., Straccia, U.: Description logic programs under probabilistic uncertainty and fuzzy vagueness. Int. J. Approx. Reasoning 50(6), 837–853 (2009)
59. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the Semantic Web. J. Web Sem. 6(4), 291–308 (2008)
60. Lukasiewicz, T., Straccia, U.: Tightly coupled fuzzy description logic programs under the answer set semantics for the Semantic Web. Int. J. Semantic Web Inf. Syst. 4(3), 68–89 (2008)
61. Lukasiewicz, T., Straccia, U.: Top-k Retrieval in Description Logic Programs Under Vagueness for the Semantic Web. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 16–30. Springer, Heidelberg (2007)
62. Ng, R.T., Subrahmanian, V.S.: Probabilistic logic programming. Inform. Comput. 101(2), 150–201 (1992)
63. Ng, R.T., Subrahmanian, V.S.: A semantical framework for supporting subjective and conditional probabilities in deductive databases. J. Autom. Reasoning 10(2), 191–235 (1993)
64. Ngo, L., Haddawy, P.: Probabilistic logic programming and Bayesian networks. In: Kanchanasut, K., Levy, J.-J. (eds.) ACSC 1995. LNCS, vol. 1023, pp. 286–300. Springer, Heidelberg (1995)
65. Nilsson, N.J.: Probabilistic logic. Artif. Intell. 28(1), 71–87 (1986)
66. Nottelmann, H., Fuhr, N.: pDAML+OIL: A probabilistic extension to DAML+OIL based on probabilistic Datalog. In: Proc. IPMU 2004 (2004)
67. Pan, R., Ding, Z., Yu, Y., Peng, Y.: A Bayesian Network Approach to Ontology Mapping. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 563–577. Springer, Heidelberg (2005)

68. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)
69. Pool, M., Aikin, J.: KEEPER and Protégé: An elicitation environment for Bayesian inference tools. In: Proceedings of the Workshop on Protégé and Reasoning (2004)
70. Poole, D.: The independent choice logic for modelling multiple agents under uncertainty. Artif. Intell. 94(1/2), 7–56 (1997)
71. Rao, P., Sagonas, K.F., Swift, T., Warren, D.S., Freire, J.: XSB: A system for efficiently computing WFS. In: Fuhrbach, U., Dix, J., Nerode, A. (eds.) LPNMR 1997. LNCS, vol. 1265, pp. 431–441. Springer, Heidelberg (1997)
72. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. 62(1/2), 107–136 (2006)
73. Rosati, R.: Towards expressive KR systems integrating Datalog and description logics: Preliminary report. In: Proc. DL 1999, pp. 160–164 (1999)
74. Rosati, R.: On the decidability and complexity of integrating ontologies and rules. J. Web Sem. 3(1), 61–73 (2005)
75. Shapiro, E.Y.: Logic programs with uncertainties: A tool for implementing rule-based systems. In: Proc. IJCAI 1983, pp. 529–532. William Kaufmann (1983)
76. Simari, G.I., Dickerson, J.P., Subrahmanian, V.S.: Cost-Based Query Answering in Action Probabilistic Logic Programs. In: Deshpande, A., Hunter, A. (eds.) SUM 2010. LNCS, vol. 6379, pp. 319–332. Springer, Heidelberg (2010)
77. Simari, G.I., Martinez, M.V., Sliva, A., Subrahmanian, V.S.: Focused most probable world computations in probabilistic logic programs. Ann. Math. Artif. Intell. (2012) (in press)
78. Straccia, U.: Fuzzy description logic programs. In: Proc. IPMU 2006, pp. 1818–1825 (2006)
79. Straccia, U.: Reasoning within fuzzy description logics. J. Artif. Intell. Res. 14, 137–166 (2001)
80. Straccia, U.: Towards a Fuzzy Description Logic for the Semantic Web (Preliminary Report). In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 167–181. Springer, Heidelberg (2005)
81. Straccia, U.: Uncertainty and description logic programs over lattices. In: Sanchez, E. (ed.) Fuzzy Logic and the Semantic Web, Capturing Intelligence, pp. 115–133. Elsevier (2006)
82. Swift, T.: Deduction in Ontologies via ASP. In: Lifschitz, V., Niemelä, I. (eds.) LPNMR 2004. LNCS (LNAI), vol. 2923, pp. 275–288. Springer, Heidelberg (2003)
83. Udrea, O., Yu, D., Hung, E., Subrahmanian, V.S.: Probabilistic Ontologies and Relational Databases. In: Meersman, R. (ed.) CoopIS/DOA/ODBASE 2005. LNCS, vol. 3760, pp. 1–17. Springer, Heidelberg (2005)
84. van Emden, M.: Quantitative deduction and its fixpoint theory. J. Log. Program. 3(1), 37–53 (1986)
85. van Keulen, M., de Keijzer, A., Alink, W.: A probabilistic XML approach to data integration. In: Proc. ICDE 2005, pp. 459–470. IEEE Computer Society (2005)
86. W3C. OWL Web Ontology Language Overview, 2004. W3C Recommendation (February10, 2004), `http://www.w3.org/TR/owl-features/`
87. W3C. OWL 2 Web Ontology Language Document Overview, 2009. W3C Recommendation (October 27, 2009), `http://www.w3.org/TR/owl2-overview/`
88. Yue, A., Liu, W., Hunter, A.: Measuring the Ignorance and Degree of Satisfaction for Answering Queries in Imprecise Probabilistic Logic Programs. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 386–400. Springer, Heidelberg (2008)