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

TIGHTLY INTEGRATED PROBABILISTIC DESCRIPTION LOGIC PROGRAMS

ANDREA CALÌ and THOMAS LUKASIEWICZ

INFSYS RESEARCH REPORT 1843-07-05

MARCH 2007

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 1843-07-05, MARCH 2007

TIGHTLY INTEGRATED PROBABILISTIC DESCRIPTION LOGIC PROGRAMS

MARCH 12, 2007

Andrea Calì [1] Thomas Lukasiewicz [2]

Abstract. We present a novel approach to probabilistic description logic programs for the Semantic Web, which constitutes a tight combination of disjunctive logic programs under the answer set semantics with both description logics and Bayesian probabilities. The approach has a number of nice features. In particular, it allows for a natural probabilistic data integration, where probabilities over possible worlds may be used as trust, error, or mapping probabilities. Furthermore, it also provides a natural integration of a situation-calculus based language for reasoning about actions with both description logics and Bayesian probabilities. We show that consistency checking and query processing are decidable resp. computable, and that they can be reduced to consistency checking resp. cautious/brave reasoning in tightly integrated disjunctive description logic programs. We also analyze the complexity of consistency checking and query processing in probabilistic description logic programs in special cases. In particular, we present a special case of these problems with polynomial data complexity.

[1]Facoltà di Scienze e Tecnologie Informatiche, Libera Università di Bolzano, Piazza Domenicani 3, I-39100 Bolzano, Italy; e-mail: cali@inf.unibz.it.

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

Acknowledgements: This work has been partially supported by the STREP FET project TONES (FP6-7603) of the European Union and by a Heisenberg Professorship of the German Research Foundation (DFG).

Contents

1 Introduction

The *Semantic Web* [1, 8] 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 knowledge representation 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* [21, 12] (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 [12]. As shown in [11], ontology entailment in OWL Lite (resp., OWL DL) reduces to knowledge base (un)satisfiability in the description logic $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$). As a next step in the development of the Semantic Web, one aims especially at sophisticated representation and reasoning capabilities for the *Rules*, *Logic*, and *Proof layers* of the Semantic Web. Several recent research efforts are going in this direction.

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, that is, 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 have the form $KB = (L, P)$, where L is a description logic knowledge base and P is a finite set of rules involving either queries to L in a loose coupling [6, 5] or concepts and roles from L as unary resp. binary predicates in a tight coupling [19, 16].

Other works explore formalisms for *uncertainty reasoning in the Semantic Web* (an important recent forum for approaches to uncertainty in the Semantic Web is the annual *Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*; there also exists a W3C Incubator Group on *Uncertainty Reasoning for the World Wide Web*). There are especially extensions of description logics, web ontology languages, and dl-programs by probabilistic uncertainty (to encode ambiguous information, such as "John is a student with the probability 0.7 and a teacher with the probability 0.3", which crucially differs from vague/fuzzy information, such as "John is tall"). In particular, [15] extends dl-programs by probabilistic uncertainty. It combines dl-programs as in [6, 5] with Poole's independent choice logic (ICL) [17, 18]. 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 [17]. Moreover, Poole's ICL also allows for natural notions of causes and explanations as in Pearl's structural causal models.

In this paper, we continue this line of research. We present *tightly integrated probabilistic disjunctive description logic programs* (or simply *probabilistic dl-programs*) *under the answer set semantics*, which are a tight integration of disjunctive logic programs under the answer set semantics, the expressive description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, and Bayesian probabilities. To our knowledge, this is the first such approach. The main contributions of this paper can be summarized as follows:

- We present a novel approach to probabilistic dl-programs, which is based on the approach to disjunctive dl-programs under the answer set semantics from [16]. The latter is a tight coupling as in [19], but it assumes no structural separation between the vocabularies of the description logic and the logic

program components.

- In the same spirit as [15], this approach is developed as a combination of dl-programs with Poole's powerful ICL. However, rather than being based on a loose coupling of rules and ontologies, it is based on a tight coupling. Furthermore, rather than being based on normal dl-programs, it is based on disjunctive dl-programs.

- We present an approach to probabilistic data integration for the Semantic Web, which is based on the novel approach to probabilistic dl-programs, where probabilistic uncertainty over possible worlds may be used as trust, error, or mapping probabilities. This application takes inspiration from a number of recent probabilistic data integration approaches in the database and web community [20].

- Since Poole's ICL is actually a formalism for reasoning about actions in dynamic systems, our approach to probabilistic dl-programs also provides a natural way of combining a language for reasoning about actions with both description logics and Bayesian probabilities, especially towards Web Services.

- We show that consistency checking and query processing in probabilistic dl-programs are decidable resp. computable, and that they can be reduced to consistency checking and cautious/brave reasoning in tightly integrated disjunctive dl-programs. This directly reveals algorithms for solving the former two problems.

- We also analyze the complexity of consistency checking and query processing in probabilistic dl-programs in special cases, which turn out to be complete for the classes $\mathrm{NEXP}^{\mathrm{NP}}$ and $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$, respectively. Furthermore, we show that in the special case of stratified normal probabilistic dl-programs relative to the description logic *DL-Lite*, these two problems have both a polynomial data complexity.

Note that the results here crucially differ from the ones in [15]. First, differently from the probabilistic dl-programs here, the ones in [15] have a loose query-based coupling between the ontology component L and the rule component P. As a consequence, the alphabets of L and P are disjoint, which is a limitation in many applications[1] (see also Examples 3.1 and 5.3). Second, query processing in the probabilistic dl-programs in [15] requires a computationally expensive linear programming step. Third, [15] does not allow for disjunctions in rule heads. Fourth, [15] does not investigate possible applications in probabilistic data integration and in probabilistic reasoning about actions. Fifth, [15] provides neither complexity nor tractability results.

The rest of this paper is organized as follows. Sections 2 and 3 recall the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ resp. disjunctive dl-programs under the answer set semantics from [16]. In Section 4, we introduce our new approach to probabilistic dl-programs. Sections 5 and 6 describe its application in probabilistic data integration resp. probabilistic reasoning about actions. In Sections 7 and 8, we focus on its computational aspects. Section 9 summarizes our main results. Note that detailed proofs of all results are given in Appendix A.

[1] As noted by David Poole (personal communication).

2 Description Logics

In this section, we recall the expressive description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which stand behind the web ontology languages OWL Lite and OWL DL [11], respectively. Intuitively, description logics model a domain of interest in terms of concepts and roles, which represent classes of individuals and binary relations between classes of individuals, respectively. A description logic knowledge base encodes especially subset relationships between concepts, subset relationships between roles, the membership of individuals to concepts, and the membership of pairs of individuals to roles.

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 domain* $\Delta^{\mathbf{D}}$ and a mapping $\cdot^{\mathbf{D}}$ that assigns to each elementary datatype a subset of $\Delta^{\mathbf{D}}$ and to each 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 pairwise disjoint denumerable sets of *atomic concepts*, *abstract roles*, *datatype 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 $\phi \in \mathbf{A}$ 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 ϕ, ϕ_1, and ϕ_2 are concepts and if $R \in \mathbf{R}_A \cup \mathbf{R}_A^-$, then also $(\phi_1 \sqcap \phi_2)$, $(\phi_1 \sqcup \phi_2)$, and $\neg\phi$ are concepts (called *conjunction*, *disjunction*, and *negation*, respectively), as well as $\exists R.\phi$, $\forall R.\phi$, $\geqslant nR$, and $\leqslant nR$ (called *exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geqslant 0$. If D is a datatype and $U \in \mathbf{R}_D$, then $\exists U.D$, $\forall U.D$, $\geqslant nU$, and $\leqslant nU$ are concepts (called *datatype exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geqslant 0$. We write $\top$ and $\bot$ to abbreviate the concepts $\phi \sqcup \neg\phi$ and $\phi \sqcap \neg\phi$, respectively, and we eliminate parentheses as usual.

An *axiom* has one of the following forms: (1) $\phi \sqsubseteq \psi$ (called *concept inclusion axiom*), where ϕ and ψ 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) $\phi(a)$ (called *concept membership axiom*), where ϕ 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., $a \neq b$) (*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 [13].

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 A university database may use a knowledge base L to characterize students and exams. For example, suppose that (1) every bachelor student is a student; (2) every master student is a student; (3) every student is either a bachelor student or a master student; (4) professors are not students; (5) only students give exams and only exams are given; (6) $john$ is a student, $mary$ is a master student, $java$ is an exam, and $john$ has given it. These relationships are expressed by the following axioms in L:

$$
\begin{array}{ll}
(1) & bachelor\_student \sqsubseteq student; \quad (2) \;\; master\_student \sqsubseteq student; \\
(3) & student \sqsubseteq bachelor\_student \sqcup master\_student; \quad (4) \;\; professor \sqsubseteq \neg student; \\
(5) & \geqslant 1 \; given \sqsubseteq student; \;\; \geqslant 1 \; given^{-1} \sqsubseteq exam; \\
(6) & student(john); \;\; master\_student(mary); \;\; exam(java); \;\; given(john, java) \, .
\end{array}
$$

2.2 Semantics

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ relative 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 $\phi \in \mathbf{A}$ 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 $\Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}}$. We extend $\cdot^{\mathcal{I}}$ to all concepts and roles, and we define the *satisfaction* of an axiom F in an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, denoted $\mathcal{I} \models F$, as usual [11]. We say $\mathcal{I}$ *satisfies* the axiom F, or $\mathcal{I}$ is a *model* of F, iff $\mathcal{I} \models F$. We say $\mathcal{I}$ *satisfies* a knowledge base L, or $\mathcal{I}$ is a *model* of L, denoted $\mathcal{I} \models L$, iff $\mathcal{I} \models F$ for all $F \in L$. We say L is *satisfiable* iff L has a model. An axiom F is a *logical consequence* of L, denoted $L \models F$, iff every model of L satisfies F.

3 Description Logic Programs

In this section, we recall a novel approach to *description logic programs* (or *dl-programs*) $KB = (L, P)$ from [16], where KB consists of a description logic knowledge base L and a disjunctive logic program P. Their semantics is defined in a modular way as in [6, 5], but it allows for a much tighter integration of L and P. Note that differently from [19], we do not assume any structural separation between the vocabularies of L and P. The main idea behind their semantics is to interpret P relative to Herbrand interpretations that are coherent with L, while L is interpreted relative to general interpretations over a first-order domain. Thus, we modularly combine the standard semantics of logic programs and of description logics, which allows for building on the standard techniques and results of both areas. As another advantage, the novel dl-programs are decidable, even when their components of logic programs and description logic knowledge bases are both very expressive. See especially [16] for further details on the new approach to dl-programs and for a detailed comparison to related works.

3.1 Syntax

We assume a first-order vocabulary Φ with nonempty finite sets of constant and predicate symbols, but no function symbols. We use Φ_c to denote the set of all constant symbols in Φ. We also assume pairwise disjoint denumerable sets $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ of atomic concepts, abstract roles, datatype roles, and individuals, respectively, as in Section 2. We assume that (i) Φ_c is a subset of $\mathbf{I}$, and (ii) Φ and $\mathbf{A}$ (resp., $\mathbf{R}_A \cup \mathbf{R}_D$) may have unary (resp., binary) predicate symbols in common.

Let $\mathcal{X}$ be a set of variables. A *term* is either a variable from $\mathcal{X}$ or a constant symbol from Φ. An *atom* is of the form $p(t_1, \ldots, t_n)$, where p is a predicate symbol of arity $n \geqslant 0$ from Φ, and $t_1, \ldots, t_n$ are terms. A *literal* l is an atom p or a negated atom $not\, p$. A *disjunctive rule* (or simply *rule*) r is an expression of form

$$\alpha_1 \vee \cdots \vee \alpha_k \leftarrow \beta_1, \ldots, \beta_n, not\, \beta_{n+1}, \ldots, not\, \beta_{n+m}\,, \tag{1}$$

where $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_{n+m}$ are atoms and $k, m, n \geqslant 0$. We call $\alpha_1 \vee \cdots \vee \alpha_k$ the *head* of r, while the conjunction $\beta_1, \ldots, \beta_n, not\, \beta_{n+1}, \ldots, not\, \beta_{n+m}$ is its *body*. We define $H(r) = \{\alpha_1, \ldots, \alpha_k\}$ and $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{\beta_1, \ldots, \beta_n\}$ and $B^-(r) = \{\beta_{n+1}, \ldots, \beta_{n+m}\}$. A *disjunctive program* P is a finite set of disjunctive rules of the form (1). We say P is *positive* iff $m = 0$ for all disjunctive rules (1) in P. We say P is a *normal program* iff $k \leqslant 1$ for all disjunctive rules (1) in P.

A *disjunctive description logic program* (or *disjunctive dl-program*) $KB = (L, P)$ consists of a description logic knowledge base L and a disjunctive program P. We say KB is *positive* iff P is positive. It is a *normal dl-program* iff P is a normal program.

Example 3.1 Consider the disjunctive dl-program $KB = (L, P)$, where L is the description logic knowledge base from Example 2.1, and P is the following set of rules, which express that (1) $bill$ is either a master student or a Ph.D. student, (2) the relation of propaedeuticity enjoys the transitive property, (3) if a student has given an exam, then he/she has given all exams that are propaedeutic to it, and (4) $unix$ is propaedeutic for $java$, and $java$ is propaedeutic for $programming\_languages$:

$$(1) \quad master\_student(bill) \vee phd\_student(bill)\,;$$
$$(2) \quad propaedeutic(X, Z) \leftarrow propaedeutic(X, Y),\ propaedeutic(Y, Z)\,;$$
$$(3) \quad given(X, Z) \leftarrow given(X, Y),\ propaedeutic(Z, Y)\,;$$
$$(4) \quad propaedeutic(unix, java);\ propaedeutic(java, programming\_languages)\,.$$

The above disjunctive dl-program also shows the advantages and flexibility of the tight integration between rules and ontologies (compared to the loose integration in [6, 5]): Observe that the predicate symbol $given$ in P is also a concept in L, and it freely occurs in both rule bodies and rule heads in P (which is both not possible in [6, 5]). Moreover, we can easily use L to express additional constraints on the predicate symbols in P. For example, we may use the two axioms $\geqslant 1\, propaedeutic \sqsubseteq exam$ and $\geqslant 1\ propaedeutic^{-1} \sqsubseteq exam$ in L to express that $propaedeutic$ in P relates only exams.

3.2 Semantics

We now define the answer set semantics of disjunctive dl-programs as a generalization of the answer set semantics of ordinary disjunctive logic programs. In the sequel, let $KB = (L, P)$ be a disjunctive dl-program.

A *ground instance* of a rule $r \in P$ is obtained from r by replacing every variable that occurs in r by a constant symbol from Φ_c. We denote by $ground(P)$ the set of all ground instances of rules in P. The *Herbrand base* relative to Φ, denoted HB_Φ, is the set of all ground atoms constructed with constant and predicate symbols from Φ. We use DL_Φ to denote the set of all ground atoms in HB_Φ that are constructed from atomic concepts in $\mathbf{A}$, abstract roles in $\mathbf{R}_A$, and concrete roles in $\mathbf{R}_D$.

An *interpretation* I is any subset of HB_Φ. Informally, every such I represents the Herbrand interpretation in which all $a \in I$ (resp., $a \in HB_\Phi - I$) are true (resp., false). We say an interpretation I is a *model* of a description logic knowledge base L, denoted $I \models L$, iff $L \cup I \cup \{\neg a \mid a \in HB_\Phi - I\}$ is satisfiable. Observe that a negative concept membership $\neg C(a)$ can be encoded as the positive concept membership $(\neg C)(a)$. The following theorem shows that also negative role memberships $\neg R(b, c)$ can be reduced to positive concept memberships and concept inclusions.

Theorem 3.2 *Let L be a description logic knowledge base, and let $R(b, c)$ be a role membership axiom. Then, $L \cup \{\neg R(b, c)\}$ is satisfiable iff $L \cup \{B(b), C(c), \exists R.C \sqsubseteq \neg B\}$ is satisfiable, where B and C are two fresh atomic concepts.*

We say an interpretation I is a *model* of a ground atom $a \in HB_\Phi$, or I *satisfies* a, denoted $I \models a$, iff $a \in I$. We say I is a *model* of a ground rule r, denoted $I \models r$, iff $I \models \alpha$ for some $\alpha \in H(r)$ whenever $I \models B(r)$, that is, $I \models \beta$ for all $\beta \in B^+(r)$ and $I \not\models \beta$ for all $\beta \in B^-(r)$. We say an interpretation I is a *model* of a set of rules P iff $I \models r$ for every $r \in ground(P)$. We say I is a *model* of a disjunctive dl-program $KB = (L, P)$, denoted $I \models KB$, iff I is a model of both L and P.

We now define the answer set semantics of disjunctive dl-programs by generalizing the ordinary answer set semantics of disjunctive logic programs. We generalize the definition via the FLP-reduct [7] (which coincides with the answer set semantics defined via the Gelfond-Lifschitz reduct [10]). Given a dl-program

$KB = (L, P)$, the *FLP-reduct* of KB relative to an interpretation $I \subseteq HB_\Phi$, denoted KB^I, is the dl-program (L, P^I), where P^I is the set of all $r \in ground(P)$ such that $I \models B(r)$. An interpretation $I \subseteq HB_\Phi$ is an *answer set* of KB iff I is a minimal model of KB^I. A dl-program KB is *consistent* (resp., *inconsistent*) iff it has an (resp., no) answer set.

We finally define the notions of *cautious* (resp., *brave*) *reasoning* from disjunctive dl-programs under the answer set semantics as follows. A ground atom $a \in HB_\Phi$ is a *cautious* (resp., *brave*) *consequence* of a disjunctive dl-program KB under the answer set semantics iff every (resp., some) answer set of KB satisfies a.

3.3 Properties

We now summarize some important properties of disjunctive dl-programs under the above answer set semantics. In the ordinary case, every answer set of a disjunctive program P is also a minimal model of P, and the converse holds when P is positive. The following theorem shows that this carries over to disjunctive dl-programs.

Theorem 3.3 *Let $KB = (L, P)$ be a disjunctive dl-program. Then, (a) every answer set of KB is a minimal model of KB, and conversely (b) if KB is positive, then every minimal model of KB is an answer set of KB.*

The next theorem shows that the answer set semantics of disjunctive dl-programs faithfully extends its ordinary counterpart. That is, the answer set semantics of a disjunctive dl-program with empty description logic knowledge base coincides with the ordinary answer set semantics of its disjunctive program.

Theorem 3.4 *Let $KB = (L, P)$ be a disjunctive dl-program with $L = \emptyset$. Then, the set of all answer sets of KB coincides with the set of all ordinary answer sets of P.*

The following theorem shows that the answer set semantics of disjunctive dl-programs also faithfully extends (from the perspective of answer set programming) the first-order semantics of description logic knowledge bases. That is, $\alpha \in HB_\Phi$ is true in all answer sets of a positive disjunctive dl-program $KB = (L, P)$ iff α is true in all first-order models of $L \cup ground(P)$. In particular, $\alpha \in HB_\Phi$ is true in all answer sets of $KB = (L, \emptyset)$ iff α is true in all first-order models of L. Note that the theorem holds also when α is a ground formula constructed from HB_Φ using the operators $\wedge$ and $\vee$.

Theorem 3.5 *Let $KB = (L, P)$ be a positive disjunctive dl-program, and let α be a ground atom from HB_Φ. Then, α is true in all answer sets of KB iff α is true in all first-order models of $L \cup ground(P)$.*

4 Probabilistic Description Logic Programs

In this section, we present a *tightly integrated* approach to *probabilistic disjunctive description logic programs* (or simply *probabilistic dl-programs*) *under the answer set semantics*. Differently from [15] (in addition to being a tightly integrated approach), the probabilistic dl-programs here also allow for disjunctions in rule heads. Similarly to the probabilistic dl-programs in [15], they are defined as a combination of dl-programs with Poole's ICL [17, 18], but using the tightly integrated disjunctive dl-programs of Section 3, rather than the loosely integrated dl-programs of [6, 5]. Poole's ICL is based on ordinary acyclic logic programs P under different "choices", where every choice along with P 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. We use the tightly integrated disjunctive dl-programs under the answer set semantics of Section 3, instead of ordinary acyclic logic programs under their canonical semantics (which coincides with their answer set semantics). We first introduce the syntax of probabilistic dl-programs 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 introduce choice spaces and probabilities on choice spaces.

A *choice space* C is a set of pairwise disjoint and nonempty sets $A \subseteq HB_\Phi - DL_\Phi$. Any $A \in C$ is an *alternative* of C and any element $a \in A$ 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 (i) a mapping $\mu\colon \bigcup C \to [0,1]$ 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) defines a probability over the values of each random variable of C, and (ii) assumes independence between the random variables.

A *probabilistic dl-program* $KB = (L, P, C, \mu)$ consists of a disjunctive dl-program (L, P), a choice space C such that no atomic choice in C coincides with the head of any rule 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 representation of a probability distribution on a finite set of disjunctive dl-programs. We say KB is *normal* iff P is normal. A *probabilistic query* to KB has the form $\exists(c_1(\boldsymbol{x}) \vee \cdots \vee c_n(\boldsymbol{x}))[r, s]$, where $\boldsymbol{x}, r, s$ is a tuple of variables, $n \geqslant 1$, and each $c_i(\boldsymbol{x})$ is a conjunction of atoms constructed from predicate and constant symbols in Φ and variables in $\boldsymbol{x}$. A *correct* (resp., *tight*) *answer* to such a query is a ground substitution θ (acting on $\boldsymbol{x}, r, s$) such that $(c_1(\boldsymbol{x}) \vee \cdots \vee c_n(\boldsymbol{x}))[r, s]\,\theta$ is a consequence (resp., tight consequence) of KB, where the notions of *consequence* and *tight consequence* are defined in the next paragraph. Note that the above probabilistic queries can also be easily extended to conditional expressions as in [15].

Example 4.1 Consider $KB = (L, P, C, \mu)$, where L and P are as in Examples 2.1 and 3.1, respectively, except that the following two (probabilistic) rules are added to P:

$$\begin{aligned} given(X, operating\_systems) &\leftarrow master\_student(X), given(X, unix), choice_m\,; \\ given(X, operating\_systems) &\leftarrow bachelor\_student(X), given(X, unix), choice_b\,. \end{aligned}$$

Let $C = \{\{choice_m, not\_choice_m\}, \{choice_b, not\_choice_b\}\}$, and let the probability μ on C be given by $\mu\colon choice_m, not\_choice_m, choice_b, not\_choice_b \mapsto 0.9, 0.1, 0.7, 0.3$. Here, the new (probabilistic) rules express that if a master (resp., bachelor) student has given the exam $unix$, then there is a probability of 0.9 (resp., 0.7) that he/she has also given $operating\_systems$. Note that probabilistic facts can be encoded by rules with only atomic choices in their body. Our wondering about the entailed tight interval for the probability that $john$ has given an exam on $java$ can be expressed by the probabilistic query $\exists(given(john, java))[R, S]$. Our wondering about which exams $john$ has given with which tight probability interval can be expressed by $\exists(given(john, E))[R, S]$.

4.2 Semantics

We now define an answer set semantics of probabilistic dl-programs, and we introduce the notions of consistency, consequence, and tight consequence.

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 Pr is an *answer set* of KB iff (i) every interpretation $I \subseteq HB_\Phi$ 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) $Pr(\bigwedge_{p \in B} p) = \sum_{I \subseteq HB_\Phi,\, B \subseteq I} Pr(I) = \mu(B)$ for every total choice B of C. Informally, 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.

We define the notions of consequence and tight consequence as follows. Given a probabilistic query $\exists(q(\boldsymbol{x}))[r, s]$, the *probability* of $q(\boldsymbol{x})$ in a probabilistic interpretation Pr under a variable assignment σ, denoted $Pr_\sigma(q(\boldsymbol{x}))$ is defined as the sum of all $Pr(I)$ such that $I \subseteq HB_\Phi$ and $I \models_\sigma q(\boldsymbol{x})$. We say $(q(\boldsymbol{x}))[l, u]$ (where $l, u \in [0, 1]$) is a *consequence* of KB, denoted $KB \mid\!\sim (q(\boldsymbol{x}))[l, u]$, iff $Pr_\sigma(q(\boldsymbol{x})) \in [l, u]$ for every answer set Pr of KB and every variable assignment σ. We say $(q(\boldsymbol{x}))[l, u]$ (where $l, u \in [0, 1]$) is a *tight consequence* of KB, denoted $KB \mid\!\sim_{tight} (q(\boldsymbol{x}))[l, u]$, iff l (resp., u) is the infimum (resp., supremum) of $Pr_\sigma(q(\boldsymbol{x}))$ subject to all answer sets Pr of KB and all σ.

5 Probabilistic Data Integration

A central aspect of the Semantic Web is data integration. In this section, we show how probabilistic dl-programs can be used for data integration with probabilities. Thus, probabilistic dl-programs are a very promising formalism for probabilistic data integration in the Rules, Logic, and Proof layers of the Semantic Web.

5.1 Overview

A *data integration system* (in its most general form) [14] $I = (G, S, M)$ consists of (i) a *global* (or *mediated*) *schema* G, which represents the domain of interest of the system, (ii) a *source schema* S, which represents the data sources that take part in the system, and (iii) a *mapping* M, which establishes a relation between 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 very data structure in S is defined through a view over G the mapping is *LAV (local-as-view)*. A mixed approach, called *GLAV* [9, 2], associates views over G to views over S.

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: (i) the symbols in Φ_G are of arity at least 1 and represent the global predicates, (ii) the symbols in Φ_S are of arity at least 1 and represent source predicates, and (iii) the symbols in Φ_c are constants. Let $\mathbf{A}_G$, $\mathbf{R}_{A,G}$, and $\mathbf{R}_{D,G}$ be pairwise disjoint denumerable sets of atomic concepts, abstract roles, and datatype roles, respectively, for the global schema, and let $\mathbf{A}_S$, $\mathbf{R}_{A,S}$, and $\mathbf{R}_{D,S}$ (disjoint from $\mathbf{A}_G$, $\mathbf{R}_{A,G}$, and $\mathbf{R}_{D,G}$) 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, μ_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.

Note that our very general setting allows 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. The only limitation is having a disjunction of atoms in the head; this does not allow us to fully capture GLAV data integration systems.

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 informative 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.

5.2 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 most simple 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 5.1 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 every place, and C may use several different relations $forecast\_weather(place, date, weather)$, $forecast\_temperature(place, date, temperature)$, and $forecast\_wind(place, date, wind)$. Suppose that the global schema G has the relation $forecast\_rome(date,$

$weather, temperature, wind)$, which may e.g. be posted on the web by the tourist information 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 $KB_M = (\emptyset, P_M, C_M, \mu_M)$:

$$\begin{aligned}
P_M &= \{forecast_rome(D,W,T,M) \leftarrow forecast(rome,D,W,T,M),\ inst_A;\\
&\quad forecast_rome(D,W,T,M) \leftarrow forecast_rome(D,W,T,M),\ inst_B;\\
&\quad forecast_rome(D,W,T,M) \leftarrow forecast_weather(rome,D,W),\\
&\qquad forecast_temperature(rome,D,T),\ forecast_wind(rome,D,M),\ inst_C\};\\
C_M &= \{\{inst_A, inst_B, inst_C\}\};\\
\mu_M &: inst_A,\ inst_B,\ inst_C \mapsto 0.6,\ 0.3,\ 0.1\,.
\end{aligned}$$

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 the institutes 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 5.2 Suppose that we want to integrate the data provided by the different sensors in a sensor network. For example, suppose that we have a sensor network measuring the concentration of ozone in several different positions of a certain town, which may e.g. 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. Suppose that the global schema contains 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 $KB_M = (\emptyset, P_M, C_M, \mu_M)$:

$$\begin{aligned}
P_M &= \{aux(P,D,T,K,R) \leftarrow reading_i(D,T,K,R),\ sensor(i,P) \mid i \in \{1,\ldots,n\}\} \cup\\
&\quad \{reading(P,D,T,K,R) \leftarrow aux(P,D,T,K,R),\ not_error_i \mid i \in \{1,\ldots,n\}\} \cup\\
&\quad \{reading(P,D,T,K,R) \leftarrow not\ aux(P,D,T,K,R),\ error_i \mid i \in \{1,\ldots,n\}\};\\
C_M &= \{\{error_i, not_error_i\} \mid i \in \{1,\ldots,n\}\};\\
\mu_M &: error_1, not_error_1, \ldots, error_n, not_error_n \mapsto e_1,\ 1{-}e_1,\ \ldots,\ e_n,\ 1{-}e_n\,.
\end{aligned}$$

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 $1 - e_j e_k$ (since it may be erroneous with the probability $e_j e_k$). 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 $\{error_j error_k,\ not\_error_j error_k, not\_error_j\ not\_error_k\}$.

Finally, 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 5.3 Suppose we want to integrate the schemas of two libraries, and that the global schema contains the concept $logic\_programming$, while the source schemas contain only the concepts $rule\text{-}based\_systems$ resp. $deductive\_databases$ in their ontologies. These three concepts are overlapping to some extent, but they do not exactly coincide. For example, a randomly chosen book from $rule\text{-}based\_systems$ (resp., $deductive\_databases$) may belong to the area $logic\_programming$ 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 $KB_M = (\emptyset, P_M, C_M, \mu_M)$:

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

Observe that the above rules express a probabilistic mapping between the concepts of two ontologies, and thus they show especially the advantages of tightly integrated probabilistic dl-programs in probabilistic data integration (since such a mapping cannot be expressed via the loosely integrated probabilistic dl-programs in [15]).

6 Probabilistic Reasoning about Actions

Poole's ICL [17, 18] is in fact a situation-calculus based language for reasoning about actions under probabilistic uncertainty. As a consequence, our approach to probabilistic dl-programs also constitutes a natural way of integrating Bayesian probabilities and description logics in reasoning about actions, especially towards Web Services.

Example 6.1 Consider a mobile robot that should pick up some objects. We now sketch how this scenario can be modeled using a probabilistic dl-program $KB = (L, P, C, \mu)$. The ontology component L encodes background knowledge about the domain. For example, concepts may encode different kinds of objects and different kinds of positions, while roles may express different kinds of relations between positions (in a 3×3 grid), which is expressed by the following description logic axioms in L:

$$\begin{aligned} &ball \sqsubseteq light\_object;\ \ light\_object \sqsubseteq object;\ \ heavy\_object \sqsubseteq object; \\ &central\_position \sqsubseteq position;\ \ object(obj_1);\ \ heavy\_object(obj_2); \\ &ball(obj_3);\ \ light\_object(obj_4);\ \ position(pos_1);\ \ldots;\ position(pos_9); \\ &central\_position(pos_5);\ \ west\_of(pos_1, pos_2);\ \ldots;\ \ \exists west\_of.\top \sqsubseteq position; \\ &\exists west\_of^{-}.\top \sqsubseteq position;\ \ north\_of(pos_1, pos_4);\ \ldots;\ neighbor(pos_1, pos_2);\ldots\,. \end{aligned}$$

The rules component P encodes the dynamics (within a finite time frame). For example, the following rule in L says that if the robot performs a pickup of object O, both the robot and the object O are at the same position, and the pickup of O succeeds (which is an atomic choice associated with a certain probability), then the robot is carrying O at the next time point (here, action function symbols are removed through grounding):

$$\begin{aligned} &carrying(O, T+1) \leftarrow do(pickup(O), T),\ at(robot, Pos, T),\ at(O, Pos, T), \\ &\quad pickup\_succeeds(O, T),\ object(O),\ position(Pos)\,. \end{aligned}$$

The subsequent rule in P says that if the robot is carrying a heavy object O, performs no pickup and no putdown operation, and keeps carrying O (which is an atomic choice associated with a certain probability), then the robot also keeps carrying O at the next time point (we can then use a similar rule for light object with a different probability):

$$carrying(O, T+1) \leftarrow carrying(O, T),\ not\ do(pickup(O), T),\ not\ do(putdown(O), T),\\ keeps\_carrying(O, T),\ heavy\_object(O),\ position(Pos)\,.$$

In order to encode the probabilities for the above rules, the choice space C contains all ground instances of $\{keeps\_carrying(O, T),\ not\_keeps\_carrying(O, T)\}$ and $\{pickup\_succeeds(O, T),\ not\_pickup\_succeeds(O, T)\}$. We then define a probability μ on each atomic choice $A \in C$ (for example, $\mu(keeps\_carrying(obj_1, 1)) = 0.9$ and $\mu(not\_keeps\_carrying(obj_1, 1)) = 0.1$) and extend it to a probability μ on the set of all total choices of C by assuming independence between the atomic choices of C.

7 Algorithms and Complexity

In this section, we characterize the consistency and the query processing problem in probabilistic dl-programs in terms of the consistency and the cautious/brave reasoning problem in disjunctive dl-programs (which are all decidable [16]). These characterizations show that the consistency and the query processing problem in probabilistic dl-programs are decidable and computable, respectively, and they directly reveal algorithms for solving these problems. We also give a precise picture of the complexity of deciding consistency and correct answers when the choice space C is bounded by a constant (which always holds for data integration using trust probabilities (where $|C| = 1$), and which is generally also reasonable when using error probabilities).

7.1 Algorithms

The following theorem shows that a probabilistic dl-program $KB = (L, P, C, \mu)$ is consistent iff $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of C. This implies that deciding whether a probabilistic dl-program is consistent can be reduced to deciding whether a disjunctive dl-program is consistent.

Theorem 7.1 *Let $KB = (L, P, C, \mu)$ be a probabilistic dl-program. Then, KB is consistent iff $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of C.*

The next theorem shows that computing tight answers for probabilistic queries $\exists(q)[r, s]$ to KB, where $q \in HB_\Phi$, can be reduced to computing all answer sets of disjunctive dl-programs and then solving two linear optimization problems. The theorem holds also when q is a ground formula constructed from HB_Φ.

Theorem 7.2 *Let $KB = (L, P, C, \mu)$ be a consistent probabilistic dl-program, and let q be a ground atom from HB_Φ. Then, l (resp., u) such that $KB \mid\!\sim_{tight} (q)[l, u]$ is the optimal value of the following linear program over y_r $(r \in R)$, where R is the set of all answer sets of $(L, P \cup \{p \leftarrow \mid p \in B\})$ for all total choices B of C:*

min (resp., max) $\sum_{r \in R,\, r \models q} y_r$ *subject to LC in Fig. 1.*

The following theorem shows that computing tight answers for $\exists(q)[r, s]$ to KB, where $q \in HB_\Phi$, can be reduced to brave and cautious reasoning from disjunctive dl-programs. Informally, to obtain the tight lower (resp., upper) bound, we have to sum up all $\mu(B)$ such that q is a cautious (resp., brave) consequence of $(L, P \cup \{p \leftarrow \mid p \in B\})$. The theorem holds also when q is a ground formula constructed from HB_Φ.

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

Figure 1: System of linear constraints LC for Theorem 7.2.

Theorem 7.3 *Let $KB = (L, P, C, \mu)$ be a consistent probabilistic dl-program, and let q be a ground atom from HB_Φ. Then, l (resp., u) such that $KB \mid\!\sim_{tight} (q)[l, u]$ is the sum of all $\mu(B)$ such that (i) B is a total choice of C and (ii) q is true in all (resp., some) answer sets of $(L, P \cup \{p \leftarrow \mid p \in B\})$.*

7.2 Complexity

The following theorem shows that deciding whether a probabilistic dl-program is consistent is complete for $\mathrm{NEXP}^{\mathrm{NP}}$ (and so has the same complexity as deciding consistency in ordinary disjunctive logic programs) when the size of its choice space is bounded by a constant. Here, the lower bound follows from the $\mathrm{NEXP}^{\mathrm{NP}}$-hardness of deciding whether an ordinary disjunctive logic program has an answer set [4].

Theorem 7.4 *Given Φ and a probabilistic dl-program $KB = (L, P, C, \mu)$, where L is defined in $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$, and the size of C is bounded by a constant, deciding whether KB is consistent is complete for $\mathrm{NEXP}^{\mathrm{NP}}$.*

The following theorem shows that deciding correct answers for probabilistic queries $\exists(q)[r, s]$, where $q \in HB_\Phi$, to a probabilistic dl-program is complete for $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$ when the size of the choice space is bounded by a constant. The theorem holds also when q is a ground formula constructed from HB_Φ.

Theorem 7.5 *Given Φ, a probabilistic dl-program $KB = (L, P, C, \mu)$, where L is defined in $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$, and the size of C is bounded by a constant, a ground atom q from HB_Φ, and $l, u \in [0, 1]$, deciding whether $(q)[l, u]$ is a consequence of KB is complete for $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$.*

8 Tractability Results

In this section, we describe a special class of probabilistic dl-programs for which the problems of deciding consistency and of query processing have both a polynomial data complexity. These programs are normal, stratified, and defined relative to *DL-Lite* [3], which allows for deciding knowledge base satisfiability in polynomial time.

We first recall *DL-Lite*. Let $\mathbf{A}$, $\mathbf{R}_A$, and $\mathbf{I}$ be pairwise disjoint sets of atomic concepts, abstract roles, and individuals, respectively. A *basic concept in DL-Lite* is either an atomic concept from $\mathbf{A}$ or an exists restriction on roles $\exists R.\top$ (abbreviated as $\exists R$), where $R \in \mathbf{R}_A \cup \mathbf{R}_A^-$. A *literal in DL-Lite* is either a basic concept b or the negation of a basic concept $\neg b$. *Concepts in DL-Lite* are defined by induction as follows. Every basic concept in *DL-Lite* is a concept in *DL-Lite*. If b is a basic concept in *DL-Lite*, and ϕ_1 and ϕ_2 are concepts in *DL-Lite*, then $\neg b$ and $\phi_1 \sqcap \phi_2$ are also concepts in *DL-Lite*. An *axiom in DL-Lite* is either (1) a concept inclusion axiom $b \sqsubseteq \psi$, where b is a basic concept in *DL-Lite* and ϕ is a concept in *DL-Lite*, or (2) a *functionality axiom* $(\mathsf{funct}\ R)$, where $R \in \mathbf{R}_A \cup \mathbf{R}_A^-$, or (3) a concept membership axiom $b(a)$, where b

is a basic concept in *DL-Lite* and $a \in \mathbf{I}$, or (4) a role membership axiom $R(a,c)$, where $R \in \mathbf{R}_A$ and $a, c \in \mathbf{I}$. A *knowledge base in DL-Lite* L is a finite set of axioms in *DL-Lite*.

Every knowledge base in *DL-Lite* L can be transformed into an equivalent one in *DL-Lite* $trans(L)$ in which every concept inclusion axiom is of form $b \sqsubseteq \ell$, where b (resp., ℓ) is a basic concept (resp., literal) in *DL-Lite* [3]. We then define $trans(P) = P \cup \{b'(X) \leftarrow b(X) \mid b \sqsubseteq b' \in trans(L),\ b' \text{ is a basic concept}\} \cup \{\exists R(X) \leftarrow R(X,Y) \mid R \in \mathbf{R}_A \cap \Phi\} \cup \{\exists R^-(Y) \leftarrow R(X,Y) \mid R \in \mathbf{R}_A \cap \Phi\}$. Intuitively, we make explicit all the relationships between the predicates in P that are implicitly encoded in L.

We define stratified normal dl- and stratified normal probabilistic dl-programs as follows. A normal dl-program $KB = (L, P)$ is *stratified* iff (i) L is defined in *DL-Lite* and (ii) $trans(P)$ is locally stratified. A probabilistic dl-program $KB = (L, P, C, \mu)$ is *normal* iff P is normal. A normal probabilistic dl-program $KB = (L, P, C, \mu)$ is *stratified* iff every of KB's represented dl-programs is stratified.

The following result shows that stratified normal probabilistic dl-programs allow for consistency checking and query processing with a polynomial data complexity. It follows from Theorems 7.1 and 7.3 and that consistency checking and cautious/brave reasoning in stratified normal dl-programs have all a polynomial data complexity [16].

Theorem 8.1 *Given Φ and a stratified normal probabilistic dl-program KB, (a) deciding whether KB has an answer set, and (b) computing $l, u \in [0,1]$ for a given ground atom q such that $KB \mid\!\sim_{tight}(q)[l,u]$ has both a polynomial data complexity.*

9 Conclusion

We have presented a tight combination of disjunctive logic programs under the answer set semantics, description logics, and Bayesian probabilities. We have described applications in probabilistic data integration and in reasoning about actions. We have shown that consistency checking and query processing are decidable resp. computable, and that they can be reduced to consistency checking and cautious/brave reasoning in disjunctive dl-programs. We have also analyzed the complexity of consistency checking and query processing in probabilistic dl-programs in special cases. In particular, we have presented a special case of these problems with polynomial data complexity.

Appendix A: Proofs

Proof of Theorem 7.1. Recall first that KB is consistent iff KB has an answer set Pr, which is a probabilistic interpretation Pr such that (i) every interpretation $I \subseteq HB_\Phi$ such that $Pr(I) > 0$ is an answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$ for some total choice B of C, and (ii) $Pr(\bigwedge_{p \in B} p) = \mu(B)$ for every total choice B of C.

($\Rightarrow$) Suppose that KB is consistent. We now show that $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of C. Towards a contradiction, suppose the contrary. That is, $(L, P \cup \{p \leftarrow \mid p \in B\})$ is not consistent for some total choice B of C. It thus follows that $Pr(\bigwedge_{p \in B} p) = 0$. But this contradicts $Pr(\bigwedge_{p \in B} p) = \mu(B)$. This shows that $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of C.

($\Leftarrow$) Suppose that $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of C. That is, there exists some answer set I_B of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every total choice B of C. Let the probabilistic

interpretation Pr be defined by $Pr(I_B) = \mu(B)$, for every total choice B of C. Then, Pr is an interpretation that satisfies (i) and (ii). That is, Pr is an answer set of KB. This shows that KB is consistent. □

Proof of Theorem 7.2. We show that every answer set Pr of KB corresponds to a solution of the system of linear constraints LC. Observe first that only the interpretations $I \subseteq HB_\Phi$ that are an answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$ for some total choice B of C can be assigned a positive probability under an answer set Pr of KB. The set of all such interpretations I corresponds to the set of all variables in R. The last two equations of LC ensure that the probability associated with each such interpretation is non-negative and that all probabilities sum up to 1. The first equation ensures that the probabilities associated with all the answer sets of each $(L, P \cup \{p \leftarrow \mid p \in B\})$ sum up to $\mu(B)$, since it is equivalent to $\sum_{r \in R,\, r \models \bigwedge B} y_r = \mu(B)$. Finally, the probability of q, which has to be minimized (resp., maximized) to obtain the tightest lower (resp., upper) bound of $Pr(q)$, is represented by the objective function $\sum_{r \in R,\, r \models q} y_r$. □

Proof of Theorem 7.3. The statement of the theorem follows from the observation that the probability $\mu(B)$ of all total choices B of C such that q is true in all (resp., some) answer sets of $(L, P \cup \{p \leftarrow \mid p \in B\})$ contributes (resp., may contribute) to the probability $Pr(q)$, while the probability $\mu(B)$ of all total choices B of C such that q is false in all answer sets of $(L, P \cup \{p \leftarrow \mid p \in B\})$ does not contribute to $Pr(q)$. □

Proof of Theorem 7.4. We first show membership in $\mathrm{NEXP}^{\mathrm{NP}}$. By Theorem 7.1, we check whether $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of C. Since C is bounded by a constant, the number of total choices of C is also bounded by a constant. As shown in [16], deciding whether a disjunctive dl-program has an answer set is in $\mathrm{NEXP}^{\mathrm{NP}}$. Hence, deciding whether KB is consistent is in $\mathrm{NEXP}^{\mathrm{NP}}$.

Hardness for $\mathrm{NEXP}^{\mathrm{NP}}$ follows from the $\mathrm{NEXP}^{\mathrm{NP}}$-hardness of deciding whether a disjunctive dl-program has an answer set [16], since by Theorem 7.1 a disjunctive dl-program $KB = (L, P)$ has an answer set iff the probabilistic dl-program $KB = (L, P, C, \mu)$ has answer set, for any choice space C and probability function μ. □

Proof of Theorem 7.5. We first show membership in co-$\mathrm{NEXP}^{\mathrm{NP}}$. We show that deciding whether $(q)[l, u]$ is not a consequence of KB is in $\mathrm{NEXP}^{\mathrm{NP}}$. By Theorem 7.3, $(q)[l, u]$ is not a consequence of KB iff there exists a set $\mathcal{B}$ of total choices B of C such that either (a.1) q is true in some answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every $B \in \mathcal{B}$, and (a.2) $\sum_{B \in \mathcal{B}} \mu(B) > u$, or (b.1) q is false in some answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every $B \in \mathcal{B}$, and (a.2) $\sum_{B \in \mathcal{B}} \mu(B) < l$. As shown in [16], deciding whether q is true in some answer set of a disjunctive dl-program is in $\mathrm{NEXP}^{\mathrm{NP}}$. It thus follows that deciding whether $(q)[l, u]$ is not a consequence of KB is in $\mathrm{NEXP}^{\mathrm{NP}}$, and thus deciding whether $(q)[l, u]$ is a consequence of KB is in co-$\mathrm{NEXP}^{\mathrm{NP}}$

Hardness for co-$\mathrm{NEXP}^{\mathrm{NP}}$ follows from the co-$\mathrm{NEXP}^{\mathrm{NP}}$-hardness of deciding whether a ground atom q is true in all answer sets of a disjunctive dl-program [16], since by Theorem 7.3 a ground atom q is true in all answer sets of a disjunctive dl-program $KB = (L, P)$ iff $(q)[1, 1]$ is a consequence of the probabilistic dl-program $KB = (L, P, C, \mu)$, for any choice space C and probability function μ. □

Proof of Theorem 8.1. As shown in [16], deciding the existence of (and computing) the answer set of a stratified normal dl-program has a polynomial data complexity. Observe then that in the case of data complexity, the choice space C is fixed. By Theorems 7.1 and 7.3, it thus follows that the problems of (a) deciding whether KB has an answer set, and (b) computing $l, u \in [0, 1]$ for a given ground atom q such that $KB \mid\!\sim_{tight} (q)[l, u]$, respectively, have both a polynomial data complexity. □

References

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

[2] A. Calì. Reasoning in data integration systems: Why LAV and GAV are siblings. In *Proceedings ISMIS-2003*, volume 2871 of *LNCS*, pages 562–571, 2003.

[3] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. *DL-Lite*: Tractable description logics for ontologies. In *Proceedings AAAI-2005*, pages 602–607, 2005.

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

[5] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *Proceedings ESWC-2006*, volume 4011 of *LNCS*, pages 273–287, 2006.

[6] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. In *Proceedings KR-2004*, pages 141–151, 2004.

[7] W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proceedings JELIA-2004*, volume 3229 of *LNCS*, pages 200–212, 2004.

[8] 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.

[9] M. Friedman, A. Y. Levy, and T. D. Millstein. Navigational plans for data integration. In *Proceedings AAAI-2006*, pages 67–73, 1999.

[10] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.

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

[12] 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 Sem.*, 1(1):7–26, 2003.

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

[14] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings PODS-2002*, pages 233–246, 2002.

[15] T. Lukasiewicz. Probabilistic description logic programs. In *Proceedings ECSQARU-2005*, volume 3571 of *LNCS*, pages 737–749. Springer, 2005. Extended version in *Int. J. Approx. Reason.*, 2007 (in press).

[16] T. Lukasiewicz. A novel combination of answer set programming with description logics for the Semantic Web. In *Proceedings ESWC-2006*, *LNCS*, 2006.

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

[18] D. Poole. Logic, knowledge representation, and Bayesian decision theory. In *Proceedings CL-2000*, volume 1861 of *LNCS*, pages 70–86, 2000.

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

[20] M. van Keulen, A. de Keijzer, and W. Alink. A probabilistic XML approach to data integration. In *Proceedings ICDE-2005*, pages 459–470, 2005.

[21] W3C. OWL web ontology language overview, 2004. W3C Recommendation (10 February 2004). Available at `www.w3.org/TR/2004/REC-owl-features-20040210/`.