Tightly Coupled Probabilistic Description Logic Programs for the Semantic Web ⋆

Andrea Calì [1], Thomas Lukasiewicz [1, 2], Livia Predoiu [3], and Heiner Stuckenschmidt [3]

[1] Computing Laboratory, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD, UK
{andrea.cali,thomas.lukasiewicz}@comlab.ox.ac.uk
[2] Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, 1040 Wien, Austria
lukasiewicz@kr.tuwien.ac.at
[3] Institut für Informatik, Universität Mannheim
68159 Mannheim, Germany
{heiner,livia}@informatik.uni-mannheim.de

Abstract. We present a novel approach to probabilistic description logic programs for the Semantic Web in which disjunctive logic programs under the answer set semantics are tightly coupled with description logics and Bayesian probabilities. The approach has several nice features. In particular, it is a logic-based representation formalism that naturally fits into the landscape of semantic web languages. Tightly coupled probabilistic description logic programs can especially be used for representing mappings between ontologies, which are a common way of approaching the semantic heterogeneity problem on the Semantic Web. In this application, they allow in particular for resolving inconsistencies and for merging mappings from different matchers based on the level of confidence assigned to different rules. Furthermore, tightly coupled probabilistic description logic programs also provide a natural integration of ontologies, action languages, and Bayesian probabilities towards web services. We explore the computational aspects of consistency checking and query processing in tightly coupled probabilistic description logic programs. We show that these problems are decidable resp. computable and that they can be reduced to consistency checking resp. cautious/brave reasoning in tightly coupled disjunctive description logic programs. Using these results, we also provide an anytime algorithm for tight query processing. Furthermore, we analyze the complexity of consistency checking and query processing in the new probabilistic description logic programs, and we present a special case of these problems with polynomial data complexity.

1 Introduction

The *Semantic Web* [3,17] 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

⋆ Some preliminary results of this paper have appeared in: *Proceedings ICLP-2007* [4], *Proceedings URSW-2007* [5], and *Proceedings FoIKS-2008* [6].

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* [47] (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. As shown in [25], 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 ontologies and rules, 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, 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 are 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 [11,12] or concepts and roles from L as unary resp. binary predicates in a tight coupling [33] (for detailed overviews on the different types of description logic programs, see especially [12,37,33,24]).

Other works explore formalisms for *uncertainty reasoning in the Semantic Web* (an important recent forum for approaches to uncertainty reasoning 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 probabilistic extensions of description logics [21,30], of web ontology languages [7,8], and of description logic programs [31,32] (to encode ambiguous information, such as "John is a student with the probability 0.7 and a teacher with the probability 0.3", which is very different from vague / fuzzy / imprecise information, such as "John is tall with the degree of truth 0.7"). In particular, the two works [31,32] extend the loosely coupled description logic programs of [11,12] by probabilistic uncertainty as in the independent choice logic (ICL) [40]. The 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, normal form games, and Pearl's structural causal models [18].

In this paper, we continue this line of research. We propose *tightly coupled probabilistic (disjunctive) description logic programs under the answer set semantics* (or *probabilistic dl-programs*), which are a tight integration of disjunctive logic programs under the answer set semantics, the description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ (behind OWL Lite resp. OWL DL), and Bayesian probabilities. To our knowledge, this

is the first such approach. As for important applications in the Semantic Web, the new description logic programs can especially be used for representing mappings between ontologies under inconsistencies and confidence values. Furthermore, tightly coupled probabilistic description logic programs are also a natural integration of action languages, ontologies, and Bayesian probabilities, especially towards Web Services.

The problem of aligning heterogeneous ontologies via semantic mappings has been identified as one of the major challenges of semantic web technologies. In order to address this problem, a number of languages for representing semantic relations between elements in different ontologies as a basis for reasoning and query answering across multiple ontologies have been proposed [45,41]. In the presence of real world ontologies, it is unrealistic to assume that mappings between ontologies are created manually by domain experts, since existing ontologies, e.g., in the area of medicine contain thousands of concepts and hundreds of relations. Recently, a number of heuristic methods for matching elements from different ontologies have been proposed that support the creation of mappings between different languages by suggesting candidate mappings (e.g., [14]). These methods rely on linguistic and structural criteria. Evaluation studies have shown that existing methods often trade off precision and recall. The resulting mapping either contains a fair amount of errors or only covers a small part of the ontologies involved [13,15]. To leverage the weaknesses of the individual methods, it is common practice to combine the results of a number of matching components or even the results of different matching systems to achieve a better coverage of the problem [14].

This means that automatically created mappings often contain uncertain hypotheses and errors that need to be dealt with, briefly summarized as follows:

- mapping hypotheses are often oversimplifying, since most matchers only support very simple semantic relations (mostly equivalence between simple elements, i.e., only between two concepts or two relations);
- there may be conflicts between different hypotheses for semantic relations from different matching components and often even from the same matcher;
- semantic relations are only given with a degree of confidence in their correctness.

If we want to use the resulting mappings, we have to find a way to deal with these uncertainties and errors in a suitable way. We argue that the most suitable way of dealing with uncertainties in mappings is to provide means to explicitly represent uncertainties in the target language that encodes the ontology mappings. In this way, integrated reasoning with the ontologies, the mappings, and the uncertainties can be performed.

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

- We present tightly coupled probabilistic (disjunctive) description logic programs under the answer set semantics, which combine the tightly coupled disjunctive description logic programs under the answer set semantics from [33] with Bayesian probabilities as in the ICL [40]. The approach assumes no structural separation between the vocabularies of the ontology and the rules component. This enables us to have description logic concepts and roles in both rule bodies and rule heads.
- We show that tightly coupled probabilistic description logic programs are especially well-suited for representing mappings between ontologies. In particular, we can have concepts and roles in both rule bodies and rule heads, which is necessary

if we want to use rules to combine ontologies. Furthermore, we can have disjunctions in rule heads and nonmonotonic negations in rule bodies, which gives a rich basis for refining and rewriting automatically created mappings for resolving inconsistencies. Finally, the integration with probability theory provides us with a sound formal framework for dealing with confidence values. In particular, we can interpret the confidence values as error probabilities and use standard techniques for combining them. We can also resolve inconsistencies via trust probabilities.

- Since the ICL is actually a formalism for probabilistic reasoning about actions in dynamic single- and multi-agent systems, tightly coupled probabilistic description logic programs are also a natural way of combining an action language with both description logics and Bayesian probabilities, especially towards Web Services.
- We show that consistency checking and query processing in tightly coupled probabilistic description logic programs are decidable resp. computable, and that they can be reduced to their classical counterparts in tightly coupled disjunctive description logic programs. We also provide an anytime algorithm for query processing.
- We analyze the complexity of consistency checking and query processing in tightly coupled probabilistic description logic programs, which turn out to be complete for the complexity classes $\mathrm{NEXP}^{\mathrm{NP}}$ and $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$, respectively. Furthermore, we show that in the stratified normal case relative to the description logic *DL-Lite*, these two problems can be solved in polynomial time in the data complexity.

The rest of this paper is organized as follows. Sections 2 and 3 recall the expressive description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, and the tightly coupled disjunctive description logic programs under the answer set semantics from [33], respectively. In Section 4, we introduce our new approach to tightly coupled probabilistic description logic programs. Sections 5 and 6 describe its application for representing ontology mappings and for probabilistic reasoning about actions involving ontologies, respectively. In Section 7, we explore the computational aspects of consistency checking and query processing in tightly coupled probabilistic description logic programs, and we provide an anytime algorithm for query processing. Section 8 describes a special case where consistency checking and query processing can be done in polynomial time in the data complexity. Section 9 discusses some most closely related works. In Section 10, we summarize our results and give an outlook on future research. Note that detailed proofs of all results of this paper are given in Appendix A.

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 [25], 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 (*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 any 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 (*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 (*conjunction*, *disjunction*, and *negation*, respectively), as well as $\exists R.\phi$, $\forall R.\phi$, $\geqslant nR$, and $\leqslant nR$ (*existential*, *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 (*datatype existential*, *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) $\phi \sqsubseteq \psi$ (*concept inclusion axiom*), where ϕ and ψ are concepts; (2) $R \sqsubseteq S$ (*role inclusion axiom*), where either $R, S \in \mathbf{R}_A \cup \mathbf{R}_A^-$ or $R, S \in \mathbf{R}_D$; (3) $\mathrm{Trans}(R)$ (*transitivity axiom*), where $R \in \mathbf{R}_A$; (4) $\phi(a)$ (*concept membership axiom*), where ϕ is a concept and $a \in \mathbf{I}$; (5) $R(a, b)$ (resp., $U(a, v)$) (*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 *(description logic) knowledge base* L is a finite set of axioms.

We next define simple abstract roles. For abstract roles $R \in \mathbf{R}_A$, we define $\mathrm{Inv}(R) = R^-$ and $\mathrm{Inv}(R^-) = R$. Let $\sqsubseteq_L^\star$ be the reflexive and transitive closure of $\sqsubseteq$ on $\bigcup\{\{R \sqsubseteq S, \mathrm{Inv}(R) \sqsubseteq \mathrm{Inv}(S)\} \mid R \sqsubseteq S \in L,\ R, S \in \mathbf{R}_A \cup \mathbf{R}_A^-\}$. An abstract role S is *simple* relative to L iff for each abstract role R with $R \sqsubseteq_L^\star S$, it holds that (i) $\mathrm{Trans}(R) \notin L$ and (ii) $\mathrm{Trans}(\mathrm{Inv}(R)) \notin L$. Informally, an abstract role S is simple iff it is neither transitive nor has transitive subroles. For decidability, number restrictions in description logic knowledge bases L are restricted to simple abstract roles [27].

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 (University Database). A university database may use a description logic 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 Ph.D. student is a student; (4) professors are not students; (5) every student is either a bachelor student or a master student or a Ph.D. student; (6) only students take exams and only exams are taken; (7) every student takes at least one exam and at most 10 exams, and every exam is taken by at most 25 students; (8) John is a student, Mary is a master student, java is an exam, and John has taken it; and (9) John is the same person as John Miller, and John and Mary are different persons. These relationships are expressed by the following axioms in L (where $\mathbf{A} = \{bachelor\_student, master\_student, student,$

$professor, exam\}, \mathbf{R}_A = \{taken\}, \mathbf{R}_D = \emptyset$, and $\mathbf{I} = \{john, john\_miller, mary, java\}$):

(1) $bachelor\_student \sqsubseteq student$; (2) $master\_student \sqsubseteq student$;
(3) $phd\_student \sqsubseteq student$; (4) $professor \sqsubseteq \neg student$;
(5) $student \sqsubseteq bachelor\_student \sqcup master\_student \sqcup phd\_student$;
(6) $\geqslant 1\ taken \sqsubseteq student$; $\geqslant 1\ taken^- \sqsubseteq exam$;
(7) $student \sqsubseteq \exists taken.exam$; $student \sqsubseteq \leqslant 10\ taken$; $exam \sqsubseteq \leqslant 25\ taken^-$;
(8) $student(john)$; $master\_student(mary)$; $exam(java)$; $taken(john, java)$;
(9) $john = john\_miller$; $john \neq mary$.

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 the mapping $\cdot^\mathcal{I}$ to all concepts and roles as usual (where $\#S$ denotes the cardinality of a set S):

- $(R^-)^\mathcal{I} = \{(y, x) \mid (x, y) \in R^\mathcal{I}\}$;
- $\{o_1, \ldots, o_n\}^\mathcal{I} = \{o_1^\mathcal{I}, \ldots, o_n^\mathcal{I}\}$; $(\neg\phi)^\mathcal{I} = \Delta^\mathcal{I} \setminus \phi^\mathcal{I}$;
- $(\phi_1 \sqcap \phi_2)^\mathcal{I} = \phi_1^\mathcal{I} \cap \phi_2^\mathcal{I}$; $(\phi_1 \sqcup \phi_2)^\mathcal{I} = \phi_1^\mathcal{I} \cup \phi_2^\mathcal{I}$;
- $(\exists R.\phi)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y\colon (x, y) \in R^\mathcal{I} \wedge y \in \phi^\mathcal{I}\}$;
- $(\forall R.\phi)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \forall y\colon (x, y) \in R^\mathcal{I} \rightarrow y \in \phi^\mathcal{I}\}$;
- $(\geqslant nR)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \#(\{y \mid (x, y) \in R^\mathcal{I}\}) \geqslant n\}$;
- $(\leqslant nR)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \#(\{y \mid (x, y) \in R^\mathcal{I}\}) \leqslant n\}$;
- $(\exists U.D)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y\colon (x, y) \in U^\mathcal{I} \wedge y \in D^\mathbf{D}\}$;
- $(\forall U.D)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \forall y\colon (x, y) \in U^\mathcal{I} \rightarrow y \in D^\mathbf{D}\}$;
- $(\geqslant nU)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \#(\{y \mid (x, y) \in U^\mathcal{I}\}) \geqslant n\}$;
- $(\leqslant nU)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \#(\{y \mid (x, y) \in U^\mathcal{I}\}) \leqslant n\}$.

The *satisfaction* of an axiom F in $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ relative to $\mathbf{D} = (\Delta^\mathbf{D}, \cdot^\mathbf{D})$, denoted $\mathcal{I} \models F$, is defined as follows: (1) $\mathcal{I} \models \phi \sqsubseteq \psi$ iff $\phi^\mathcal{I} \subseteq \psi^\mathcal{I}$; (2) $\mathcal{I} \models R \sqsubseteq S$ iff $R^\mathcal{I} \subseteq S^\mathcal{I}$; (3) $\mathcal{I} \models \mathrm{Trans}(R)$ iff $R^\mathcal{I}$ is transitive; (4) $\mathcal{I} \models \phi(a)$ iff $a^\mathcal{I} \in \phi^\mathcal{I}$; (5) $\mathcal{I} \models R(a, b)$ iff $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$; (6) $\mathcal{I} \models U(a, v)$ iff $(a^\mathcal{I}, v^\mathbf{D}) \in U^\mathcal{I}$; (7) $\mathcal{I} \models a = b$ iff $a^\mathcal{I} = b^\mathcal{I}$; and (8) $\mathcal{I} \models a \neq b$ iff $a^\mathcal{I} \neq b^\mathcal{I}$. 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 description logic 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.

Example 2.2 (University Database cont'd). Consider again the description logic knowledge base L of Example 2.1. It is not difficult to verify that L is satisfiable, and that $professor \sqsubseteq \neg master\_student$, $student(mary)$, $(bachelor\_student \sqcup master\_student)(john)$, and $taken(john, java)$ are logical consequences of L.

3 Tightly Coupled Disjunctive DL-Programs

In this section, we recall the *tightly coupled* approach to *disjunctive description logic programs* (or simply *disjunctive dl-programs*) $KB = (L, P)$ under the answer set semantics from [33], where KB consists of a description logic knowledge base L and a disjunctive logic program P. The semantics of KB is defined in a modular way as in [11,12], but it allows for a much tighter coupling of L and P. Note that we do not assume any structural separation between the vocabularies of L and P. The main idea behind the semantics of KB is to interpret P relative to Herbrand interpretations that are compatible 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 disjunctive dl-programs are decidable, even when their components of logic programs and description logic knowledge bases are both very expressive. See especially [33] for further details on the novel approach to disjunctive dl-programs and for a detailed comparison to related works.

3.1 Syntax

We assume a first-order vocabulary Φ with finite nonempty 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 a set of data values $\mathbf{V}$ (relative to a datatype theory $\mathbf{D} = (\Delta^{\mathbf{D}}, \cdot^{\mathbf{D}})$) and 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 $\mathbf{I} \cup \mathbf{V}$, and that (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 default-negated atom $not\ p$. A *disjunctive rule* (or simply *rule*) r is an expression of the 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 (logic) 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 (logic) program* iff $k \leqslant 1$ for all disjunctive rules (1) in P.

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

Example 3.1 (University Database cont'd). 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 (which is encoded by a rule that has the form of a disjunction of ground atoms), and Mary has taken an exam in unix; (2) every student who has taken an exam in knowledge bases is either a master student or a Ph.D. student (which is encoded by a rule with a disjunction in its head); (3) every student who is not known to be a master student or a Ph.D. student is a bachelor student (which is encoded by a rule with default negations in its body); (4) the relation of being a prerequisite enjoys the transitive property; (5) if a student has taken an exam, then he/she has taken every exam that is a prerequisite for it; and (6) unix is a prerequisite for java, and java is a prerequisite for programming languages:

$$
\begin{array}{l}
(1)\ \mathit{master\_student}(\mathit{bill}) \vee \mathit{phd\_student}(\mathit{bill});\ \ \mathit{taken}(\mathit{mary}, \mathit{unix})\,; \\
(2)\ \mathit{master\_student}(X) \vee \mathit{phd\_student}(X) \leftarrow \mathit{taken}(X, \mathit{knowledge\_bases})\,; \\
(3)\ \mathit{bachelor\_student}(X) \leftarrow \\
\qquad\qquad \mathit{student}(X),\ \mathit{not}\ \mathit{master\_student}(X),\ \mathit{not}\ \mathit{phd\_student}(X)\,; \\
(4)\ \mathit{prerequisite}(X, Z) \leftarrow \mathit{prerequisite}(X, Y),\ \mathit{prerequisite}(Y, Z)\,; \\
(5)\ \mathit{taken}(X, Z) \leftarrow \mathit{taken}(X, Y),\ \mathit{prerequisite}(Z, Y)\,; \\
(6)\ \mathit{prerequisite}(\mathit{unix}, \mathit{java});\ \mathit{prerequisite}(\mathit{java}, \mathit{programming\_languages})\,.
\end{array}
$$

The above disjunctive dl-program also shows the advantages and flexibility of the tight coupling between rules and ontologies (compared to the loose coupling in [11,12]): Observe that the predicate symbol taken in P is also a role in L, and it freely occurs in both rule bodies and rule heads in P (which is both not possible in [11,12]). Moreover, we can easily use L to express additional constraints on the predicate symbols in P. For example, we may use the two axioms $\geqslant 1\, \mathit{prerequisite} \sqsubseteq \mathit{exam}$ and $\geqslant 1\ \mathit{prerequisite}^{-1} \sqsubseteq \mathit{exam}$ in L to express that $\mathit{prerequisite}$ in P relates only exams.

3.2 Semantics

We now define the answer set semantics of (tightly coupled) 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 $\mathit{ground}(P)$ 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 datatype 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. We say 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 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 [16] (which is equivalent to the definition via the Gelfond-Lifschitz reduct [20]). Given a disjunctive dl-program $KB = (L, P)$, the *FLP-reduct* of KB relative to an interpretation $I \subseteq HB_\Phi$, denoted KB^I, is the disjunctive 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 disjunctive dl-program KB is *consistent* (resp., *inconsistent*) iff it has an (resp., no) answer set.

Example 3.2 (University Database cont'd). Consider again the disjunctive dl-program $KB = (L, P)$ of Example 3.1. It is not difficult to verify that KB is consistent and that it has two answer sets, which contain both in particular the ground atoms

$$student(john),\ student(john\_miller),\ bachelor\_student(john)\,,$$
$$bachelor\_student(john\_miller),\ master\_student(mary),\ student(mary)\,,$$
$$student(bill),\ exam(java),\ taken(john, java),\ taken(john\_miller, java)\,,$$
$$taken(mary, unix),\ prerequisite(java, programming\_languages)\,,$$
$$prerequisite(unix, java),\ prerequisite(unix, programming\_languages)\,,$$

as well as either $master\_student(bill)$ or $phd\_student(bill)$.

We finally define the notion 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.

Example 3.3 (University Database cont'd). Consider again the disjunctive dl-program $KB = (L, P)$ of Example 3.1. By Example 3.2, the ground atom $student(bill)$ is a cautious consequence of KB, while $phd\_student(bill)$ is a brave consequence of KB.

3.3 Semantic Properties

We now summarize some important semantic 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. This result holds also for disjunctive dl-programs.

As another important semantic property, 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. Furthermore, 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, a ground atom $\alpha \in HB_\Phi$ 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, a ground atom $\alpha \in HB_\Phi$ is true in all answer sets of $KB = (L, \emptyset)$ iff α is

true in all first-order models of L. Note that this result holds also when α is a ground formula constructed from HB_{Φ} using the operators $\wedge$ and $\vee$.

Another important feature of disjunctive dl-programs $KB = (L, P)$ concerns the *unique name assumption*, which says that any two distinct constant symbols in Φ_c represent two distinct domain objects (and which is quite usual in logic programming). It turns out that we do not have to make the unique name assumption here, since the description logic knowledge base of a disjunctive dl-program may very well contain or imply equalities between individuals. Intuitively, since we have no unique name assumption in L, we also do not have to make the unique name assumption in P.

Example 3.4. The unique answer set of the disjunctive dl-program $KB = (L, P) = (\{a = b\}, \{p(a)\})$, where $a, b \in \Phi_c \cap \mathbf{I}$ and $p \in \Phi \cap \mathbf{A}$, contains both ground atoms $p(a)$ and $p(b)$, since L contains the equality axiom $a = b$, and P contains the fact $p(a)$.

The tight coupling of ontologies and rules semantically behaves very differently from the loose coupling. This makes the former more (and the latter less) suitable for representing ontology mappings (see Section 5) and for combining sophisticated reasoning formalisms from Artificial Intelligence (such as reasoning about actions) with ontologies (see Section 6). The following example illustrates this difference.

Example 3.5 (Client Database). The normal dl-program $KB = (L, P)$, where

$$\begin{array}{rcl} L & = & \{\mathit{person}(a),\ \mathit{person} \sqsubseteq \mathit{male} \sqcup \mathit{female}\} \text{ and} \\ P & = & \{\mathit{client}(X) \leftarrow \mathit{male}(X),\ \mathit{client}(X) \leftarrow \mathit{female}(X)\} \end{array}$$

implies $\mathit{client}(a)$, while the normal dl-program $KB' = (L', P')$ as in [11,12]

$$\begin{array}{rcl} L' & = & \{\mathit{person}(a),\ \mathit{person} \sqsubseteq \mathit{male} \sqcup \mathit{female}\} \text{ and} \\ P' & = & \{\mathit{client}(X) \leftarrow DL[\mathit{male}](X),\ \mathit{client}(X) \leftarrow DL[\mathit{female}](X)\} \end{array}$$

does *not* imply $\mathit{client}(a)$, since the two queries are evaluated independently from each other, and neither $\mathit{male}(a)$ nor $\mathit{female}(a)$ follows from L'. To obtain the conclusion $\mathit{client}(a)$ in [11,12], one has to directly use the rule $\mathit{client}(X) \leftarrow DL[\mathit{male} \sqcup \mathit{female}](X)$.

4 Tightly Coupled Probabilistic DL-Programs

In this section, we present a *tightly coupled* approach to *probabilistic disjunctive description logic programs* (or simply *probabilistic dl-programs*) under the answer set semantics. Differently from [31] (in addition to being a tightly coupled approach), the probabilistic dl-programs here also allow for disjunctions in rule heads. Similarly to the probabilistic dl-programs in [31], they are defined as a combination of dl-programs with the ICL [40], but using the tightly coupled disjunctive dl-programs of [33] (see Section 3), rather than the loosely coupled dl-programs of [11,12]. The 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 [33], 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.

Observe that tightly coupled probabilistic dl-programs generalize a simplified version of Poole's (multi-agent) ICL, where we have only "nature" as agent, and the Herbrand base is finite. Since the latter can represent (discrete and finite) Bayesian networks [39] and (binary and finite) structural causal models [18], it thus follows immediately that tightly coupled probabilistic dl-programs can also represent (discrete and finite) Bayesian networks and (binary and finite) structural causal models. Furthermore, since Poole's ICL can represent influence diagrams, Markov decision processes, and normal form games [40], it follows that a multi-agent version of tightly coupled probabilistic dl-programs (where we additionally have a finite set of agents (including "nature"), which each control certain alternatives on the choice space, and the probability on the choice space only concerns the alternatives controlled by nature), can also represent influence diagrams, Markov decision processes, and normal form games.

4.1 Syntax

We now define the syntax of (tightly coupled) probabilistic dl-programs and probabilistic queries to them. We first introduce choice spaces and probabilities on choice spaces.

A *choice space* $\mathcal{C}$ is a set of pairwise disjoint and nonempty sets $A \subseteq HB_\Phi - DL_\Phi$. Any $A \in \mathcal{C}$ is an *alternative* of $\mathcal{C}$ and any element $a \in A$ an *atomic choice* of $\mathcal{C}$. Intuitively, every alternative $A \in \mathcal{C}$ represents a random variable and every atomic choice $a \in A$ one of its possible values. A *total choice* of $\mathcal{C}$ is a set $B \subseteq HB_\Phi$ such that $|B \cap A| = 1$ for all $A \in \mathcal{C}$ (and thus $|B| = |\mathcal{C}|$). Intuitively, every total choice B of $\mathcal{C}$ represents an assignment of values to all the random variables. A *probability* μ on a choice space $\mathcal{C}$ is a probability function on the set of all total choices of $\mathcal{C}$. Intuitively, every probability μ is a probability distribution over the set of all joint variable assignments. Since $\mathcal{C}$ and all its alternatives are finite, μ can be defined by (i) a mapping $\mu\colon \bigcup \mathcal{C} \to [0,1]$ such that $\sum_{a \in A} \mu(a) = 1$ for all $A \in \mathcal{C}$, and (ii) $\mu(B) = \Pi_{b \in B} \mu(b)$ for all total choices B of $\mathcal{C}$. Intuitively, (i) defines a probability over the values of each random variable of $\mathcal{C}$, and (ii) assumes independence between the random variables.

Example 4.1 (University Database cont'd). A choice space $\mathcal{C}$ for the University Database may be defined by $\mathcal{C} = \{\{choice_u,\ not\_choice_u\},\ \{choice_o,\ not\_choice_o\}\}$, which represents two random variables X_u and X_o with the binary domains $\{choice_u,\ not\_choice_u\}$ and $\{choice_o,\ not\_choice_o\}$, respectively. A probability μ on $\mathcal{C}$ may be given as follows. We first define μ for the atomic choices by $\mu\colon choice_u,\ not\_choice_u,\ choice_o,\ not\_choice_o \mapsto 0.9,\ 0.1,\ 0.7,\ 0.3$, and then we naturally extend μ to all total choices by assuming independence between the alternatives. For example, the total choice $B = \{choice_u,\ not\_choice_o\}$ (which represents the joint variable assignment $X_u = choice_u, X_o = not\_choice_o$) has the probability $\mu(B) = 0.9 \cdot 0.3 = 0.27$.

Observe that, as in Poole's ICL [40], all atomic choices in the alternatives of choices spaces are ground. But one may also use non-ground atomic choices in the alternatives by assuming that every non-ground alternative abbreviates all its ground instances, which allows for a more compact representation of choice spaces and their probabilities.

Example 4.2 (University Database cont'd). The non-ground choice space $\mathcal{C} = \{\{p(X), not\_p(X)\}\}$ along with the probability $\mu\colon p(X), not\_p(X) \mapsto 0.9, 0.1$ (assuming independence between the alternatives) abbreviates the ground choice space $\mathcal{C}' = \{\{p(c), not\_p(c)\} \mid c \in \Phi_c\}$ along with the probability $\mu'\colon p(c), not\_p(c) \mapsto 0.9, 0.1$ for every $c \in \Phi_c$ (assuming independence between the alternatives). Informally, for every constant $c \in \Phi_c$, we have one random variable X_c with the binary domain $\{p(c), not\_p(c)\}$ and the probability $\mu'\colon p(c), not\_p(c) \mapsto 0.9, 0.1$ on it. Similarly, one may also have one random variable X_c for every constant $c \in \Phi_s$, where Φ_s is a certain subset of Φ_c only, such as the set of all constants in Φ_c encoding exams.

A *tightly coupled probabilistic disjunctive description logic program* (or simply *probabilistic dl-program*) $KB = (L, P, \mathcal{C}, \mu)$ consists of a (tightly coupled) disjunctive dl-program (L, P), a choice space $\mathcal{C}$ such that no atomic choice in $\mathcal{C}$ coincides with the head of any rule in $ground(P)$, and a probability μ on $\mathcal{C}$. Intuitively, since the total choices of $\mathcal{C}$ select subsets of P, and μ is a probability distribution on the total choices of $\mathcal{C}$, every probabilistic dl-program is the compact representation of a probability distribution on a finite set of disjunctive dl-programs. Observe here that P is fully general (it may have disjunctions in rule heads and default negations in rule bodies, and it may not necessarily be (locally) stratified or acyclic). We say KB is *normal* iff P is normal. An *event* α is any Boolean combination of atoms (that is, 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. A *probabilistic query* to KB has the form $\exists(\beta|\alpha)[r, s]$, where $\beta|\alpha$ is a conditional event, and r and s are either two variables or two reals from $[0, 1]$. Note that dealing with probabilities of conditional events in both knowledge bases and queries is commonly regarded as being an indispensable feature of probabilistic reasoning formalisms in Artificial Intelligence [38,19,44]. In our approach, conditional events are explicitly allowed in queries, and probabilities of conditional events in the knowledge base can be modeled in the same way as in Poole's ICL [40].

Example 4.3 (University Database cont'd). A probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$ is given by the choice space $\mathcal{C}$ and the probability μ on $\mathcal{C}$ of Example 4.1, and the disjunctive dl-program (L, P), which is nearly the same as the one given in Example 3.1, except that P now also contains the following two (probabilistic) rules:

$$taken(X, operating\_systems) \leftarrow master\_student(X), taken(X, unix), choice_u\,;$$
$$taken(X, databases) \leftarrow master\_student(X), taken(X, operating\_systems), choice_o\,.$$

Here, the new (probabilistic) rules express that if a master student has taken an exam in $unix$ (resp., $operating\_systems$), then there is a probability of 0.9 (resp., 0.7) that he/she has also taken an exam in $operating\_systems$ (resp., $databases$). Note that probabilistic facts can be encoded by rules with only atomic choices in their body.

Querying for the entailed tight interval for the probability that Bill is a student (resp., master student) can be expressed by the probabilistic query $\exists(student(bill))[R, S]$ (resp., $\exists(master\_student(bill))[R, S]$). Querying for the entailed tight interval for the probability that Mary has taken an exam in databases (resp., Mary has taken an exam in databases given that she has taken an exam in operating systems) can be expressed by the probabilistic query $\exists(taken(mary,\ databases))[R, S]$ (resp., $\exists(taken(mary,$

$databases)|taken(mary, operating\_systems))[R, S]$). In the latter case, we model a conditioning of all probability distributions that are compatible with KB on the observation that Mary has taken an exam in operating systems (which is not encoded in KB). Querying for the exams that John has taken along with their tight probability intervals can be expressed by the probabilistic query $\exists(taken(john, E))[R, S]$.

4.2 Semantics

We now define an answer set semantics of (tightly coupled) probabilistic dl-programs, and we introduce the notions of consistency, consequence, tight consequence, and correct and tight answers for probabilistic queries to probabilistic dl-programs.

Given a probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$, 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 $\mathcal{C}$, 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 $\mathcal{C}$. Informally, Pr is an answer set of $KB = (L, P, \mathcal{C}, \mu)$ 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 $\mathcal{C}$, and (ii) Pr coincides with μ on the total choices B of $\mathcal{C}$. We say KB is *consistent* iff it has an answer set Pr.

Example 4.4 (University Database cont'd). Consider again the probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$ of Example 4.3. Let S_1, S_2, S_3, resp. S_4 be answer sets of $(L, P \cup \{choice_u, choice_o\})$, $(L, P \cup \{choice_u, not\_choice_o\})$, $(L, P \cup \{not\_choice_u, choice_o\})$, resp. $(L, P \cup \{not\_choice_u, not\_choice_o\})$. Then, $Pr\colon S_1, S_2, S_3, S_4 \mapsto 0.63, 0.27, 0.07, 0.03$ is an answer set of KB, which also shows that KB is consistent.

If additionally S_1', S_2', S_3', resp. S_4' are answer sets of $(L, P \cup \{choice_u, choice_o\})$, $(L, P \cup \{choice_u, not\_choice_o\})$, $(L, P \cup \{not\_choice_u, choice_o\})$, resp. $(L, P \cup \{not\_choice_u, not\_choice_o\})$ different from S_1, S_2, S_3, resp. S_4, then $Pr\colon S_1, S_1', S_2, S_2', S_3, S_3', S_4, S_4' \mapsto 0.63, 0, 0.22, 0.05, 0.06, 0.01, 0.02, 0.01$ is an answer set of KB.

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 $I \models \alpha$. Given two ground events α and β, and two reals $l, u \in [0, 1]$, we say $(\beta|\alpha)[l, u]$ is a *consequence* of a consistent probabilistic dl-program KB under the answer set semantics iff $Pr(\alpha \wedge \beta) \,/\, Pr(\alpha) \in [l, u]$ for all answer sets Pr of KB with $Pr(\alpha) > 0$. We say $(\beta|\alpha)[l, u]$ 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* (resp., *correct answer*) for a probabilistic query $Q = \exists(\beta|\alpha)[r, s]$ to KB under the answer set semantics, where r and s are two variables (resp., two reals from $[0, 1]$), is the set of all ground substitutions θ (for the variables in Q) such that $(\beta|\alpha)[r, s]\theta$ is a tight consequence (resp., consequence) of KB under the answer set semantics. For ease of presentation, since tight (and correct) answers for probabilistic queries $Q = \exists(\beta|\alpha)[r, s]$ with non-ground

$\beta|\alpha$ are easily reducible to tight answers for probabilistic queries $Q = \exists(\beta|\alpha)[r, s]$ with ground $\beta|\alpha$,[1] we consider only the latter type of probabilistic queries in the following.

Example 4.5 (University Database cont'd). Consider again the probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$ of Example 4.3. It is then not difficult to verify that the tight answers to the two probabilistic queries $\exists(student(bill))[R, S]$ and $\exists(master\_student(bill))[R, S]$ are given by $\theta = \{R/1, S/1\}$ and $\theta = \{R/0, S/1\}$, respectively. Furthermore, the tight answers to the two probabilistic queries $\exists(taken(mary,\ databases))[R, S]$ and $\exists(taken(mary, databases)|taken(mary, operating\_systems))[R, S]$ are given by $\theta = \{R/0.63, S/0.63\}$ and $\theta = \{R/0.7, S/0.7\}$, respectively.

5 Representing Ontology Mappings

In this section, we show that tightly coupled probabilistic dl-programs are well-suited for representing ontology mappings. We first describe the requirements of a formal language for representing and combining correspondences produced by different matching components or systems. We then show how tightly coupled disjunctive dl-programs can be used for representing (possibly inconsistent) ontology mappings (without confidence values). We finally show how tightly coupled probabilistic dl-programs can be used for representing (possibly inconsistent) ontology mappings with confidence values.

5.1 Representation Requirements

The problem of ontology matching can be defined as follows [14]. Ontologies are theories encoded in a certain language L. In this work, we assume that ontologies are encoded in OWL DL or OWL Lite. For each ontology O in language L, we denote by $Q(O)$ the matchable elements of the ontology O. Given two ontologies O and O', the task of matching is now to determine correspondences between the matchable elements in the two ontologies. Correspondences are 5-tuples (id, e, e', r, p) such that

- id is a unique identifier for referring to the correspondence;
- $e \in Q(O)$ and $e' \in Q(O')$ are matchable elements from the two ontologies;
- $r \in R$ is a semantic relation (in this work, we consider the case where the semantic relation can be interpreted as an implication);
- p is a degree of confidence in the correctness of the correspondence.

Note that the set of matchable elements $Q(O)$ are determined by the degree of overlap between the ontologies and by the matching tools which are used to discover mappings. That is, if the ontologies do not describe overlapping domains, obviously the set of matchable elements of each ontology is empty. If, furthermore, a matching tool is used which is only able to detect very simple mappings, e.g. only between concepts or relations, the set of matchable elements of an ontology only consists of such matchable

[1] Every probabilistic query $Q = \exists(\beta|\alpha)[r, s]$ with non-ground $\beta|\alpha$ is reduced to all ground instances $Q\theta$ of Q relative to Φ_c: The tight answer to $Q = \exists(\beta|\alpha)[r, s]$, where r and s are two variables, is given by the set of all $\theta \circ \theta'$ such that (i) θ is a ground substitution for $\beta|\alpha$ and (ii) θ' is the tight answer to $Q\theta$. As Φ_c is finite, also the set of all ground instances of Q is finite.

elements, i.e. in this case of exactly those concepts and relations that have a counterpart in another ontology. Note that such matchable elements are allowed to be any construct of the language L which is used to represent the ontologies. I.e. arbitrary formulae can be matchable elements. However, it is very difficult to detect mappings between arbitrary matchable elements as this amounts to learning a relation between arbitrary logic formulae in OWL DL or OWL Lite. Thus, most matching tools that exist nowadays cannot detect more complex mappings than simple mappings between concepts or relations.

From the above general description of automatically generated correspondences between ontologies, we can derive a number of requirements for a formal language for representing the results of multiple matchers as well as the contained uncertainties:

– *Tight integration of mapping and ontology language:* The semantics of the language used to represent the correspondences between elements in different ontologies has to be tightly integrated with the semantics of the ontology language used (in this case OWL). This is important if we want to use the correspondences to reason across different ontologies in a semantically coherent way. In particular, this means that the interpretation of the mapped elements depends on the definitions in the ontologies.

– *Support for mappings refinement:* The language should be expressive enough to allow the user to refine oversimplifying correspondences suggested by the matching system. This is important to be able to provide a more precise account of the true semantic relation between elements in the mapped ontologies. In particular, this requires the ability to describe correspondences that include several elements from the two ontologies.

– *Support for repairing inconsistencies:* Inconsistent mappings are a major problem for the combined use of ontologies because they can cause inconsistencies in the mapped ontologies. These inconsistencies can make logical reasoning impossible, since everything can be derived from an inconsistent ontology. The mapping language should be able to represent and reason about inconsistent mappings in an approximate fashion.

– *Representation and combination of confidence:* The confidence values provided by matching systems are an important indicator for the uncertainty that has to be taken into account. The mapping representation language should be able to use these confidence values when reasoning with mappings. In particular, it should be able to represent the confidence in a mapping rule and to combine confidence values on a sound formal basis.

– *Decidability and efficiency of instance reasoning:* An important use of ontology mappings is the exchange of data across different ontologies. In particular, we normally want to be able to ask queries using the vocabulary of one ontology and receive answers that do not only consist of instances of this ontology but also of ontologies connected through ontology mappings. To support this, query answering in the combined formalism consisting of ontology language and mapping language has to be decidable and there should be efficient algorithms for answering queries.

Throughout this section, we use real data form the Ontology Alignment Evaluation Initiative[2] to illustrate the different aspects of mapping representation. In particular, we use examples from the benchmark and the conference data set. The benchmark

[2] http://oaei.ontologymatching.org/2006/

dataset consists of five OWL ontologies (tests 101 and 301–304) describing scientific publications and related information. The conference dataset consists of about 10 OWL ontologies describing concepts related to conference organization and management. In both cases, we give examples of mappings that have been created by the participants of the 2006 evaluation campaign. In particular, we use mappings created by state-of-the-art ontology matching systems like falcon and hmatch.

5.2 Deterministic Ontology Mappings

We now show how tightly coupled disjunctive dl-programs $KB = (L, P)$ can be used for representing (possibly inconsistent) mappings (without confidence values) between two ontologies. Intuitively, L encodes the union of the two ontologies, while P encodes the mappings between the ontologies, where disjunctions in rule heads and nonmonotonic negations in rule bodies in P can be used to resolve inconsistencies.

Tightly coupled disjunctive dl-programs $KB = (L, P)$ naturally represent two heterogeneous ontologies O_1 and O_2, and mappings between O_1 and O_2 as follows. The description logic knowledge base L is the union of two independent description logic knowledge bases L_1 and L_2, which encode the ontologies O_1 and O_2, respectively. Here, we assume that L_1 and L_2 have signatures $\mathbf{A}_1$, $\mathbf{R}_{A,1}$, $\mathbf{R}_{D,1}$, $\mathbf{I}_1$ and $\mathbf{A}_2$, $\mathbf{R}_{A,2}$, $\mathbf{R}_{D,2}$, $\mathbf{I}_2$, respectively, such that $\mathbf{A}_1 \cap \mathbf{A}_2 = \emptyset$, $\mathbf{R}_{A,1} \cap \mathbf{R}_{A,2} = \emptyset$, $\mathbf{R}_{D,1} \cap \mathbf{R}_{D,2} = \emptyset$, and $\mathbf{I}_1 \cap \mathbf{I}_2 = \emptyset$. Note that this can easily be achieved for any pair of ontologies by a suitable renaming. A mapping between elements e_1 and e_2 from L_1 and L_2, respectively, is then represented by a simple rule $e_2(\mathbf{x}) \leftarrow e_1(\mathbf{x})$ in P, where $e_1 \in \mathbf{A}_1 \cup \mathbf{R}_{A,1} \cup \mathbf{R}_{D,1}$, $e_2 \in \mathbf{A}_2 \cup \mathbf{R}_{A,2} \cup \mathbf{R}_{D,2}$, and $\mathbf{x}$ is a suitable variable vector. Informally, such a rule encodes that every instance of (the concept or role) e_1 in O_1 is also an instance of (the concept or role) e_2 in O_2. Note that demanding the signatures of L_1 and L_2 to be disjoint guarantees that the rule base that represents mappings between different ontologies is stratified as long as there are no cyclic mappings. Note furthermore that the restriction to such simple mapping rules is not imposed by us but by the limitations of the used matchers, which they share with most matchers existing nowadays.

Example 5.1. Taking an example from the conference data set of the OAEI challenge 2006, we find e.g. the following mappings that have been created by the hmatch system for mapping the CRS Ontology (O_1) on the EKAW Ontology (O_2):

$$O_2 : \mathit{EarlyRegisteredParticipant}(X) \leftarrow O_1 : \mathit{Participant}(X) ;$$
$$O_2 : \mathit{LateRegisteredParticipant}(X) \leftarrow O_1 : \mathit{Participant}(X) .$$

Informally, these two mapping relationships express that every instance of the concept $\mathit{Participant}$ of the ontology O_1 is also an instance of the concepts $\mathit{EarlyRegistered}$-$\mathit{Participant}$ and $\mathit{LateRegisteredParticipant}$, respectively, of the ontology O_2.

We now encode the two ontologies and the mappings by a tightly coupled disjunctive dl-program $KB = (L, P)$, where L is the union of two description logic knowledge bases L_1 and L_2 encoding the ontologies O_1 resp. O_2, and P encodes the mappings. However, we cannot directly use the two mapping relationships as two rules in P, since this would introduce an inconsistency in KB. More specifically, recall that a model of KB has to satisfy both L and P. Here, the two mapping relationships interpreted

as rules in P would require that if there is a participant Alice ($Participant(alice)$) in the ontology O_1, an answer set of KB contains $EarlyRegisteredParticipant(alice)$ and $LateRegisteredParticipant(alice)$ at the same time. Such an answer set, however, is invalidated by the ontology O_2, which requires the concepts $EarlyRegistered$-$Participant$ and $LateRegisteredParticipant$ to be disjoint. Therefore, these mappings are useless, since they do not actively participate in the creation of any model of KB.

In [34], we present a method for detecting such inconsistent mappings. There are different approaches for resolving this inconsistency. The most straightforward one is to drop mappings until no inconsistency is present any more. Peng and Xu [46] have proposed a more suitable method for dealing with inconsistencies in terms of a relaxation of the mappings. In particular, they propose to replace a number of conflicting mappings by a single mapping that includes a disjunction of the conflicting concepts. In the example above, we would replace the two mapping rules by the following one:

$$O_2 : EarlyRegisteredParticipant(X) \vee \\ O_2 : LateRegisteredParticipant(X) \leftarrow O_1 : Participant(X) .$$

This new mapping rule can be represented in our framework and resolves the inconsistency. More specifically, for a particular participant Alice ($Participant(alice)$) in the ontology O_1, it imposes the existence of two answer sets

$$\{O_2 : EarlyRegisteredParticipant(alice), O_1 : Participant(alice)\} ; \\ \{O_2 : LateRegisteredParticipant(alice), O_1 : Participant(alice)\} .$$

None of these answer sets is invalidated by the disjointness constraints imposed by the ontology O_2. However, we can deduce only $Participant(alice)$ cautiously, the other atoms can be deduced bravely. More generally, with such rules, instances that are only available in the ontology O_2 cannot be classified with certainty.

We can solve this issue by refining the rules again and making use of nonmonotonic negation. In particular, we can extend the body of the original mappings with the following additional requirement:

$$O_2 : EarlyRegisteredParticipant(X) \leftarrow \\ O_1 : Participant(X) \wedge O_1 : RegisterdbeforeDeadline(X) ; \\ O_2 : LateRegisteredParticipant(X) \leftarrow \\ O_1 : Participant(X) \wedge not\ O_1 : RegisteredbeforeDeadline(X) .$$

This refinement of the mapping rules resolves the inconsistency and also provides a more correct mapping because background information has been added. A drawback of this approach is the fact that it requires manual post-processing of mappings because the additional background information is not obvious. In the next section, we present a probabilistic extension of tightly integrated disjunctive dl-programs that allows us to directly use confidence estimations of matching engines to resolve inconsistencies and to combine the results of different matchers.

5.3 Ontology Mappings with Confidence Values

We next show how tightly coupled probabilistic dl-programs $KB = (L, P, \mathcal{C}, \mu)$ can be used for representing (possibly inconsistent) mappings with confidence values between

two ontologies. Intuitively, L encodes the union of the two ontologies, while $P, \mathcal{C}$, and μ encode the mappings between the ontologies, where confidence values can be encoded as error probabilities, and inconsistencies can also be resolved via trust probabilities (in addition to using disjunctions and nonmonotonic negations in P). Note, however, that again we need to employ an additional method for detecting inconsistent mappings as mentioned in section 5.2. Here, we show how the previously detected incosistencies can be resolved by taking into account the uncertainty represented by the confidence values the matchers produce. Furthermore, we also show how we can combine possibly inconsistent results of different matchers by adding the representation of trust to the matchers by means of bayesian probabilities. The trust values can be adjusted manually, but it is also conceivable to adjust them automatically by providing the background knowledge of the domain and by using a statistical preevaluation on some benchmarking ontologies of different domains.

The probabilistic extension of tightly coupled disjunctive dl-programs $KB = (L, P)$ to tightly coupled probabilistic dl-programs $KB' = (L, P, \mathcal{C}, \mu)$ provides us with a means to explicitly represent and use the confidence values provided by matching systems. In particular, we can interpret the confidence value as an *error probability* and state that the probability that a mapping introduces an error is $1 - p$. Conversely, the probability that a mapping correctly describes the semantic relation between elements of the different ontologies is $1 - (1 - p) = p$. This means that we can use the confidence value p as a probability for the correctness of a mapping. The indirect formulation is chosen, because it allows us to combine the results of different matchers in a meaningful way. In particular, if we assume that the error probabilities of two matchers are independent, we can calculate the joint error probability of two matchers that have found the same mapping rule as $(1 - p_1) \cdot (1 - p_2)$. This means that we can get a new probability for the correctness of the rule found by two matchers which is $1 - (1 - p_1) \cdot (1 - p_2)$. This way of calculating the joint probability meets the intuition that a mapping is more likely to be correct if it has been discovered by more than one matcher because $1 - (1 - p_1) \cdot (1 - p_2) \geqslant p_1$ and $1 - (1 - p_1) \cdot (1 - p_2) \geqslant p_2$.

In addition, when merging inconsistent results of different matching systems, we weigh each matching system and its result with a (e.g., user-defined) *trust probability*, which describes our confidence in its quality. All these trust probabilities sum up to 1. For example, the trust probabilities of the matching systems m_1, m_2, and m_3 may be 0.6, 0.3, and 0.1, respectively. That is, we trust most in m_1, medium in m_2, and less in m_3.

Example 5.2. We illustrate this approach using an example from the benchmark data set of the OAEI 2006 campaign. In particular, we consider the case where the publication ontology in test 101 (O_1) is mapped on the ontology of test 302 (O_2). Below we show some mappings that have been detected by the matching system hmatch that participated in the challenge. The mappings are described as rules in P, which contain a conjunct indicating the matching system that has created it and a subscript for identifying the mapping. These additional conjuncts are atomic choices of the choice space $\mathcal{C}$ and link probabilities (which are specified in the probability μ on the choice space $\mathcal{C}$) to the rules (where the common concept $Proceedings$ of both ontologies O_1 and O_2 is

renamed to the concepts $Proceedings_1$ and $Proceedings_2$, respectively):

$$O_2 : Book(X) \leftarrow O_1 : Collection(X) \wedge hmatch_1 ;$$
$$O_2 : Proceedings_2(X) \leftarrow O_1 : Proceedings_1(X) \wedge hmatch_2 .$$

We define the choice space according to the interpretation of confidence described above. The resulting choice space is $\mathcal{C} = \{\{hmatch_i, not\_hmatch_i\} \mid i \in \{1, 2\}\}$. It comes along with the probability μ on $\mathcal{C}$, which assigns the corresponding confidence value p to each atomic choice $hmatch_i$ and the complement $1 - p$ to the atomic choice not_hmatch_i. In our case, we have $\mu(hmatch_1) = 0.62$, $\mu(not\_hmatch_1) = 0.38$, $\mu(hmatch_2) = 0.73$, and $\mu(not\_hmatch_2) = 0.27$.

The benefits of this explicit treatment of uncertainty becomes clear when we now try to merge this mapping with the result of another matching system. Below are two examples of rules that describe correspondences for the same ontologies that have been found by the falcon system:

$$O_2 : InCollection(X) \leftarrow O_1 : Collection(X) \wedge falcon_1 ;$$
$$O_2 : Proceedings_2(X) \leftarrow O_1 : Proceedings_1(X) \wedge falcon_2 .$$

Here, the confidence encoding yields the choice space $\mathcal{C}' = \{\{falcon_i, not\_falcon_i\} \mid i \in \{1, 2\}\}$ along with the probabilities $\mu'(falcon_1) = 0.94$, $\mu'(not\_falcon_1) = 0.06$, $\mu'(falcon_2) = 0.96$, and $\mu'(not\_falcon_2) = 0.04$.

Note that directly merging these two mappings as they are would not be a good idea for two reasons. The first one is that we might encounter an inconsistency problem like shown in Section 5.2. For example, in this case, the ontology O_2 imposes that the concepts $InCollection$ and $Book$ are to be disjoint. Thus, for each publication pub belonging to the concept $Collection$ in the ontology O_1, the merged mappings infer $Book(pub)$ and $InCollection(pub)$. Therefore, the first rule of each of the mappings cannot contribute to a model of the knowledge base. The second reason is that a simple merge does not account for the fact that the mapping between the $Proceedings_1$ and $Proceedings_2$ concepts has been found by both matchers and should therefore be strengthened. Here, the mapping rule has the same status as any other rule in the mapping and each instance of Proceedings has two probabilities at the same time.

Suppose we associate with hmatch and falcon the trust probabilities 0.55 and 0.45, respectively. Based on the interpretation of confidence values as error probabilities, and on the use of trust probabilities when resolving inconsistencies between rules, we can now define a merged mapping set that consists of the following rules:

$$O_2 : Book(X) \leftarrow O_1 : Collection(X) \wedge hmatch_1 \wedge sel\_hmatch_1 ;$$
$$O_2 : InCollection(X) \leftarrow O_1 : Collection(X) \wedge falcon_1 \wedge sel\_falcon_1 ;$$
$$O_2 : Proceedings(X) \leftarrow O_1 : Proceedings(X) \wedge hmatch_2 ;$$
$$O_2 : Proceedings(X) \leftarrow O_1 : Proceedings(X) \wedge falcon_2 .$$

The new choice space $\mathcal{C}''$ and the new probability μ'' on $\mathcal{C}''$ are obtained from $\mathcal{C} \cup \mathcal{C}'$ and $\mu \cdot \mu'$ (which is the product of μ and μ', that is, $(\mu \cdot \mu')(B \cup B') = \mu(B) \cdot \mu'(B')$ for all total choices B of $\mathcal{C}$ and B' of $\mathcal{C}'$), respectively, by adding the alternative $\{sel\_hmatch_1, sel\_falcon_1\}$ and the probabilities $\mu''(sel\_hmatch_1) = 0.55$ and $\mu''(sel\_\ falcon_1) = 0.45$ for resolving the inconsistency between the first two rules.

It is not difficult to verify that, due to the independent combination of alternatives, the last two rules encode that the rule $O_2 : Proceedings(X) \leftarrow O_1 : Proceedings(X)$ holds with the probability $1 - (1 - \mu''(hmatch_2)) \cdot (1 - \mu''(falcon_2)) = 0.9892$, as desired. Informally, any randomly chosen instance of $Proceedings$ of the ontology O_1 is also an instance of $Proceedings$ of the ontology O_2 with the probability 0.9892. In contrast, if the mapping rule would have been discovered only by falcon or hmatch, respectively, such an instance of $Proceedings$ of the ontology O_1 would be an instance of $Proceedings$ of the ontology O_2 with the probability 0.96 or 0.73, respectively.

A probabilistic query Q asking for the probability that a specific publication pub in the ontology O_1 is an instance of the concept $Book$ of the ontology O_2 is given by $Q = \exists(Book(pub))[R, S]$. The tight answer θ to Q is given by $\theta = \{R/0, S/0\}$, if pub is not an instance of the concept $Collection$ in the ontology O_1 (since there is no mapping rule that maps another concept than $Collection$ to the concept $Book$). If pub is an instance of the concept $Collection$, however, then the tight answer to Q is given by $\theta = \{R/0.341, S/0.341\}$ (as $\mu''(hmatch_1) \cdot \mu''(sel\_hmatch_1) = 0.62 \cdot 0.55 = 0.341$). Informally, pub belongs to the concept $Book$ with the probabilities 0 resp. 0.341.

6 Probabilistic Reasoning about Actions Involving Ontologies

The ICL [40] is in fact a language for probabilistic reasoning about actions in single- and multi-agent systems. As a consequence, our approach to (tightly coupled) probabilistic dl-programs also constitutes a natural way of integrating reasoning about actions, description logics and Bayesian probabilities, especially towards Web Services. The following example illustrates this application of probabilistic dl-programs.

Example 6.1. Consider a mobile robot that should pick up some objects. We now sketch how this scenario can be modeled using a (tightly coupled) probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$. 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:

$$
\begin{array}{l}
ball \sqsubseteq light\_object;\ \ light\_object \sqsubseteq object;\ \ heavy\_object \sqsubseteq object\,; \\
central\_position \sqsubseteq position;\ \ central\_position \sqsubseteq \leqslant 9\, neighbor^-.position\,; \\
central\_position \sqsubseteq (\leqslant 1\, west\_of^-.position) \sqcap (\leqslant 1\, north\_of^-.position)\,; \\
\exists west\_of.\top \sqsubseteq position;\ \ \exists west\_of^-.\top \sqsubseteq position\,; \\
object(obj_1);\ \ light\_object(obj_2);\ \ heavy\_object(obj_3);\ \ ball(obj_4);\ \ obj_2 = obj_4\,; \\
position(pos_1);\ \ldots;\ position(pos_9);\ \ central\_position(pos_5)\,; \\
neighbor(pos_1, pos_2);\ \ldots;\ west\_of(pos_1, pos_2);\ \ldots;\ north\_of(pos_1, pos_4);\ \ldots\,.
\end{array}
$$

Intuitively, the description logic knowledge base L adds to P different kinds of objects and positions, as well as different kinds of relations between positions: Every answer set of KB contains in particular the following ground atoms:

$$
\begin{array}{l}
object(obj_1);\ \ ball(obj_1);\ \ light\_object(obj_1);\ \ light\_object(obj_2);\ \ object(obj_2)\,; \\
heavy\_object(obj_3);\ \ object(obj_3);\ \ ball(obj_4);\ \ light\_object(obj_4);\ \ object(obj_4)\,; \\
position(pos_1);\ \ldots;\ position(pos_9);\ \ central\_position(pos_5)\,; \\
neighbor(pos_1, pos_2);\ \ldots;\ west\_of(pos_1, pos_2);\ \ldots;\ north\_of(pos_1, pos_4);\ \ldots\,.
\end{array}
$$

The rules component P encodes the dynamics (within a finite time frame). For example, the following rule in L says that if (1) the robot performs a pickup of object O, (2) both the robot and the object O are at the same position, and (3) 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 (as usual, we use parameterized actions for a more compact representation, that is, $pickup(obj_i)$ and $putdown(obj_i)$ represent the non-parameterized actions $pickup\_obj_i$ and $putdown\_obj_i$, respectively):

$$carrying(O,T') \leftarrow do(pickup(O),T),\ at(robot,Pos,T),\ at(O,Pos,T),\\ pickup\_succeeds(O,T),\ object(O),\ position(Pos),\ succ(T,T')\,.$$

The next rule in P says that if (1) the robot is carrying a heavy object O, (2) performs no pickup and no putdown operation, and (3) 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):

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

To encode the probabilities for the above rules, the choice space $\mathcal{C}$ 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 alternative $A \in \mathcal{C}$ (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 $\mathcal{C}$ by assuming independence between the alternatives of $\mathcal{C}$.

7 Algorithms and Complexity

In this section, we characterize the consistency and the query processing problem in probabilistic dl-programs under the answer set semantics in terms of the consistency and the cautious/brave reasoning problem in disjunctive dl-programs under the answer set semantics (which are all decidable [33]). These characterizations show that the consistency and the query processing problem in probabilistic dl-programs under the answer set semantics are decidable resp. computable, and they also directly reveal algorithms for solving these problems. In particular, the second characterization can be used for an anytime algorithm for tight query processing in probabilistic dl-programs under the answer set semantics. We describe this anytime algorithm along with soundness and error estimation results. We also give a precise picture of the complexity of deciding consistency and correct answers for probabilistic dl-programs under the answer set semantics.

7.1 Algorithms

The following theorem shows that a probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$ is consistent iff the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of $\mathcal{C}$ with $\mu(B) > 0$. Thus, deciding whether a probabilistic dl-program is consistent can be reduced to deciding whether a disjunctive dl-program is consistent.

Theorem 7.1. *Let $KB{=}(L, P, \mathcal{C}, \mu)$ be a probabilistic dl-program. Then, KB is consistent iff $(L, P{\cup}\{p \leftarrow \mid p \in B\})$ is consistent for each total choice B of $\mathcal{C}$ with $\mu(B){>}0$.*

The next theorem shows that computing tight answers for probabilistic queries $Q = \exists(\beta|\alpha)[r, s]$ with ground $\beta|\alpha$ to consistent probabilistic dl-programs KB under the answer set semantics can be reduced to brave and cautious reasoning from disjunctive dl-programs. Informally, the tight lower (resp., upper) bound is computed from values a (resp., b) and c (resp., d), where (1) a (resp., b) is the sum of all $\mu(B)$ such that (1.i) B is a total choice of $\mathcal{C}$ and (1.ii) $\alpha \wedge \beta$ a cautious (resp., brave) consequence of the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$, and (2) c (resp., d) is the sum of all $\mu(B)$ such that (2.i) B is a total choice of $\mathcal{C}$ and (2.ii) $\alpha \wedge \neg\beta$ a brave (resp., cautious) consequence of the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$.

Theorem 7.2. *Let $KB = (L, P, \mathcal{C}, \mu)$ be a consistent probabilistic dl-program, and let $Q = \exists(\beta|\alpha)[r, s]$ be a probabilistic query with ground conditional event $\beta|\alpha$. Let a (resp., b) be the sum of all $\mu(B)$ such that (i) B is a total choice of $\mathcal{C}$ and (ii) $\alpha \wedge \beta$ is true in every (resp., some) answer set of the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$. Let c (resp., d) be the sum of all $\mu(B)$ such that (i) B is a total choice of $\mathcal{C}$ and (ii) $\alpha \wedge \neg\beta$ is true in some (resp., every) answer set of the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$. Then, the tight answer θ for Q to KB under the answer set semantics is given as follows:*

$$\theta \;=\; \begin{cases} \{r/1,\ s/0\} & \text{if } b = 0 \text{ and } c = 0; \\ \{r/0,\ s/0\} & \text{if } b = 0 \text{ and } c \neq 0; \\ \{r/1,\ s/1\} & \text{if } b \neq 0 \text{ and } c = 0; \\ \{r/\frac{a}{a+c},\ s/\frac{b}{b+d}\} & \text{otherwise.} \end{cases} \tag{2}$$

By the above theorem, computing the tight answer for probabilistic queries $Q = \exists(\beta|\alpha)[r, s]$ with ground $\beta|\alpha$ to consistent probabilistic dl-programs $KB = (L, P, \mathcal{C}, \mu)$ under the answer set semantics can be reduced to (1) computing the set of all answer sets of each disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ such that B is a total choice of $\mathcal{C}$ and (2) performing brave and cautious reasoning from these answer sets. The number of all total choices B is generally a non-neglectable source of complexity. We thus propose (i) to compute the tight answer for Q to KB only up to an error within a given threshold $\epsilon \in [0, 1]$, (ii) to process the B's along decreasing probabilities $\mu(B)$, and (iii) to eventually stop the calculation after a given time interval.

Given a consistent probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$, a probabilistic query $Q = \exists(\beta|\alpha)[r, s]$ with ground $\beta|\alpha$, and an error threshold $\epsilon \in [0, 1]$, Algorithm tight_answer (see Fig. 1) computes some $\theta = \{r/l',\ s/u'\}$ such that $|l - l'| + |u - u'| \leqslant \epsilon$, where $\{r/l,\ s/u\}$ is the tight answer for Q to KB under the answer set semantics. 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 Theorem 7.2). It then computes the answer set semantics S of the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B_i\})$, for every total choice B_i of $\mathcal{C}$, checks whether $\alpha \wedge \beta$ and $\alpha \wedge \neg\beta$ are true or false in all $s \in S$, 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 thus far. Hence, in the special case where

Algorithm tight_answer

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

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

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

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

Fig. 1. Algorithm tight_answer.

$\epsilon = 0$, the algorithm computes in particular the tight answer for Q to KB under the answer set semantics. The following theorem shows that tight_answer is sound.

Theorem 7.3. *Let KB be a consistent 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 the answer set semantics. Let $\epsilon \in [0, 1]$ be an error threshold. Then, Algorithm tight_answer always terminates on KB, Q, and ϵ. Let $\theta' = \{r/l',\ s/u'\}$ be the output computed by tight_answer for KB, Q, and ϵ, and let v' be the value of the variable v. Then, if $v' = 0$, then $l = l'$ and $u = u'$; otherwise, $|l - l'| + |u - u'| \leqslant \epsilon$.*

The following example illustrates how Algorithm tight_answer works.

Example 7.1 (University Database cont'd). Consider again the probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$ and the probabilistic query $Q = \exists(\beta|\alpha)[r, s] = \exists(taken(mary, databases)|taken(mary, operating\_systems))[R, S]$ of Example 4.3, and suppose $\epsilon = 0$. After the initialization in line 1, we observe that (i) $\alpha \wedge \beta$ is true in every answer set of exactly $(L,\ P \cup \{choice_u,\ choice_o\})$, (ii) $\alpha \wedge \beta$ is false in every answer set of exactly $(L, P \cup \{choice_u,\ not\_choice_o\})$, $(L, P \cup \{not\_choice_u,\ choice_o\})$, and $(L,\ P \cup \{not\_choice_u,\ not\_choice_o\})$, (iii) $\alpha \wedge \neg\beta$ is false in every answer set of exactly $(L,\ P \cup \{choice_u,\ choice_o\})$, $(L, P \cup \{not\_choice_u,\ choice_o\})$, and $(L, P \cup \{not\_choice_u,\ not\_choice_o\})$, and (iv) $\alpha \wedge \neg\beta$ is true in every answer set of exactly $(L, P \cup \{choice_u,\ not\_choice_o\})$. We thus obtain (i) $a = 0 + 0.63 = 0.63$, (ii) $b = 1 - 0.27 - 0.07 - 0.03 = 0.63$, (iii) $c = 1 - 0.63 - 0.07 - 0.03 = 0.27$, and (iv) $d = 0 + 0.27 = 0.27$, respectively, and return the tight answer $\theta = \{r/\frac{0.63}{0.63+0.27},\ s/\frac{0.63}{0.63+0.27}\} = \{r/0.7,\ s/0.7\}$.

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 it can also be shown that the computed lower and upper bounds are increasing and that the possible error is decreasing along the iterations of the while-loop of the algorithm). For this reason, Algorithm tight_answer also iterates through the total choices B_i of $\mathcal{C}$ 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 7.4. *Let KB be a consistent probabilistic dl-program, let $Q = \exists(\beta|\alpha)[r,s]$ be a probabilistic query with ground conditional event $\beta|\alpha$, let $\epsilon \in [0,1]$ be an error threshold, and let $\theta = \{r/l, s/u\}$ be the tight answer for Q to KB under the answer set semantics. Assume we run Algorithm 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), and let a', b', c', d', and v' be the values of the variables a, b, c, d, and v, respectively. Then, if $v' = 0$, then $\theta = \theta'$; otherwise, $|l - l'| + |u - u'| \leqslant \frac{v'}{a'+c'} + \frac{v'}{b'+d'}$.*

7.2 Complexity

The following theorem shows that deciding whether a probabilistic dl-program under the answer set semantics is consistent is complete for $\mathrm{NEXP}^{\mathrm{NP}}$ (and thus has the same complexity as deciding consistency of ordinary disjunctive logic programs under the answer set semantics). Note that the lower bound follows from the $\mathrm{NEXP}^{\mathrm{NP}}$-hardness of deciding whether an ordinary disjunctive logic program has an answer set.

Theorem 7.5. *Given a first-order vocabulary Φ and a probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$, where L is defined in $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$, deciding whether KB is consistent under the answer set semantics is complete for* $\mathrm{NEXP}^{\mathrm{NP}}$.

The next theorem shows that deciding correct answers for probabilistic queries $Q = \exists(\beta|\alpha)[l,u]$, where $\beta|\alpha$ is a ground conditional event, to a consistent probabilistic dl-program KB under the answer set semantics is complete for $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$.

Theorem 7.6. *Given a first-order vocabulary Φ, a consistent probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$, where L is defined in $\mathcal{SHIF}(\mathbf{D})$ or $\mathcal{SHOIN}(\mathbf{D})$, a ground conditional event $\beta|\alpha$, and reals $l, u \in [0,1]$, deciding whether $(\beta|\alpha)[l,u]$ is a consequence of KB under the answer set semantics is complete for* $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$.

8 Tractability Results

We now describe a special class of (tightly coupled) probabilistic dl-programs for which deciding consistency and query processing can both be done in polynomial time in the data complexity. These programs are normal, stratified, and defined relative to *DL-Lite* [9] (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 existential 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 $b \sqsubseteq \phi$, where b is a basic concept in *DL-Lite*, and ϕ is a concept in *DL-Lite*, or (2) a *functionality axiom* (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*.

Theorem 8.1 (see [9]). ...

Theorem 8.2 (see [9]). *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.*

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 in *DL-Lite* as follows. A normal dl-program $KB = (L, P)$ with L in *DL-Lite* is *stratified* iff $trans(P)$ is locally stratified. A probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$ is *normal* iff P is normal. A normal probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$ with L in *DL-Lite* is *stratified* iff every of KB's represented dl-programs is stratified.

Example 8.1. ...

The following result shows that stratified normal probabilistic dl-programs in *DL-Lite* allow for consistency checking and query processing with a polynomial data complexity. It follows from Theorems 7.1 and 7.2 and that consistency checking and cautious/brave reasoning in stratified normal dl-programs can be done in polynomial time in the data complexity [33].

Theorem 8.3. *Given a first-order vocabulary* Φ *and a stratified normal probabilistic dl-program* $KB = (L, P, \mathcal{C}, \mu)$ *with* L *in DL-Lite, (a) deciding whether* KB *has an answer set, and (b) computing* $l, u \in [0, 1]$ *for a given ground conditional event* $\beta|\alpha$ *such that* $KB \mid\!\!\sim_{tight} (\beta|\alpha)[l, u]$ *can both be done in polynomial time in the data complexity.*

9 Related Work

...

9.1 Tightly Coupled Description Logic Programs

Some other tight integrations of rules and ontologies are in particular due to Donini et al. [10], Levy and Rousset [29], Grosof et al. [22], Motik et al. [36], Heymans et al. [23], and Rosati [43,42]. SWRL [26] and WRL [2] also belong to this category. Among the above works, closest in spirit to the tightly coupled disjunctive dl-programs used in this paper is perhaps Rosati's approach [43,42]. Like here, Rosati's hybrid knowledge bases also consist of a description logic knowledge base L and a disjunctive program (with default negations) P, where concepts and roles in L may act as predicate symbols in P. However, differently from this paper, Rosati partitions the predicates of L and P into description logic predicates and logic program predicates, where the former are interpreted under the classical model-theoretic semantics, while the latter are interpreted under the answer set semantics (and thus in particular default negations of concepts and roles are not allowed in P). Furthermore, differently from this paper, he also assumes a syntactic restriction on rules (called *weak safeness*) to gain decidability, and he assumes the standard names assumption, which includes the unique name assumption.

9.2 Probabilistic Description Logic Programs

It is important to point out that the probabilistic description logic programs here are very different from the ones in [31] (and their recent tractable variant in [32]). First, they are based on the tight integration between the ontology component L and the rule component P of [33], while the ones in [31,32] realize the loose query-based integration between the ontology component L and the rule component P of [11]. This implies in particular that the vocabularies of L and P here may have common elements (see also Example 3.1), while the vocabularies of L and P in [31,32] are necessarily disjoint. Furthermore, the probabilistic description logic programs here behave semantically very differently from the ones in [31,32] (see Example 3.5). As a consequence, the probabilistic description logic programs here are especially useful for sophisticated probabilistic reasoning tasks involving ontologies (including representing and reasoning with ontology mappings under probabilistic uncertainty and inconsistency, as well as probabilistic reasoning about actions involving ontologies), while the ones in [31,32] can especially be used as query interfaces to web databases (including RDF theories). Second, differently from the programs here, the ones in [31,32] do not allow for disjunctions in rule heads. Third, differently from here, the works [31,32] do not explore the use of probabilistic description logic programs for representing and reasoning with ontology mappings under probabilistic uncertainty and inconsistency, and their use for probabilistic reasoning about actions involving ontologies.

9.3 Representing Ontology Mappings

Vielleicht hinyufuegen: To our knowlegde, such a combination is not possible in other formal logics that provide a decidable hybrid integration of rules and ontologies.

9.4 Reasoning about Actions Involving Ontologies

...

10 Conclusion

We have presented tightly coupled probabilistic (disjunctive) dl-programs under the answer set semantics, which are a tight combination of disjunctive logic programs under the answer set semantics, description logics, and Bayesian probabilities. We have described applications in representing and reasoning with ontology mappings and in probabilistic reasoning about actions involving ontologies. We have shown that consistency checking and query processing in tightly coupled probabilistic dl-programs are decidable resp. computable, and that they can be reduced to their classical counterparts in tightly coupled disjunctive dl-programs. We have also given an anytime algorithm for query processing, and we have analyzed the complexity of consistency checking and query processing. Furthermore, we have delineated a special case of these problems that can be solved in polynomial time in the data complexity.

As for representing ontology mappings, the new formalism supports the resolution of inconsistencies on a symbolic and a numeric level. While the use of disjunction and nonmonotonic negation allows the rewriting of inconsistent rules, the probabilistic extension of the language allows us to explicitly represent numeric confidence values as error probabilities, to resolve inconsistencies by using trust probabilities, and to reason about these on a numeric level. While being expressive and well-integrated with description logic ontologies, the new formalism is still decidable and has data-tractable subsets, which make it particularly interesting for practical applications.

We leave for future work the implementation of tightly coupled probabilistic dl-programs. Another interesting topic for future work is to explore whether the tractability results can be extended to an even larger class of tightly coupled probabilistic dl-programs. One way to achieve this could be to approximate the answer set semantics through the well-founded semantics (which may be defined similarly as in [32]). Furthermore, it would be interesting to investigate whether one can develop an efficient top-k query technique for the presented tightly coupled probabilistic dl-programs: Rather than computing the tight probability interval for a given ground conditional event, such a technique returns the k most probable ground instances of a given non-ground atom.

Acknowledgments. Andrea Calì has been supported by the STREP FET project TONES (FP6-7603) of the European Union. Thomas Lukasiewicz has been supported by the German Research Foundation (DFG) under the Heisenberg Programme and by the Austrian Science Fund (FWF) under the project P18146-N04. Heiner Stuckenschmidt and Livia Predoiu have been supported by an Emmy-Noether Grant of the German Research Foundation (DFG).

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

($\Rightarrow$) Suppose that KB is consistent. We now show that the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of $\mathcal{C}$ with $\mu(B) > 0$. 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 $\mathcal{C}$ with $\mu(B) > 0$. It thus follows that $Pr(\bigwedge_{p \in B} p) = 0$. But this contradicts $Pr(\bigwedge_{p \in B} p) = \mu(B) > 0$. This shows that $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of $\mathcal{C}$ with $\mu(B) > 0$.

($\Leftarrow$) Suppose that the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of $\mathcal{C}$ with $\mu(B) > 0$. 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 $\mathcal{C}$ with $\mu(B) > 0$. Let the probabilistic interpretation Pr be defined by $Pr(I_B) = \mu(B)$ for every total choice B of $\mathcal{C}$ with $\mu(B) > 0$ and by $Pr(I) = 0$ for all other $I \subseteq HB_\Phi$. Then, Pr is an interpretation that satisfies (i) and (ii). That is, Pr is an answer set of KB. Thus, KB is consistent. □

Proof of Theorem 7.2. The statement of the theorem is immediate for the three cases where $b = 0$ or $c = 0$, since $b = 0$ (resp., $c = 0$) iff $Pr(\alpha \wedge \beta) = 0$ (resp., $Pr(\alpha \wedge \neg\beta) = 0$) for all models Pr of KB. Thus, in the following, suppose $b \neq 0$ and $c \neq 0$. Observe first that the probability $\mu(B)$ of all total choices B of $\mathcal{C}$ such that δ is true in all (resp., some) answer sets of the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ definitely contributes (resp., "can be made to" contribute) to the probability $Pr(\delta)$. As for the lower bound, every $\mu(B)$ of all total choices B of $\mathcal{C}$ such that $\alpha \wedge \beta$ is true in all answer sets of $(L, P \cup \{p \leftarrow \mid p \in B\})$ definitely contributes to both $Pr(\alpha \wedge \beta)$ and $Pr(\alpha)$. Hence, we obtain the smallest value of $Pr(\alpha \wedge \beta) \,/\, Pr(\alpha) = Pr(\alpha \wedge \beta) \,/\, (Pr(\alpha \wedge \beta) + Pr(\alpha \wedge \neg\beta))$, if we additionally take the probabilities $\mu(B)$ of all total choices B of $\mathcal{C}$ such that $\alpha \wedge \neg\beta$ is true in some answer sets of $(L, P \cup \{p \leftarrow \mid p \in B\})$ and make them contribute to $Pr(\alpha)$. Similarly, as for the upper bound, every $\mu(B)$ of all total choices B of $\mathcal{C}$ such that $\alpha \wedge \neg\beta$ is true in all answer sets of $(L, P \cup \{p \leftarrow \mid p \in B\})$ definitely does not contribute to $Pr(\alpha \wedge \beta)$ but contributes to $Pr(\alpha)$. Thus, we obtain the largest value of $Pr(\alpha \wedge \beta) \,/\, Pr(\alpha) = Pr(\alpha \wedge \beta) \,/\, (Pr(\alpha \wedge \beta) + Pr(\alpha \wedge \neg\beta))$ if we additionally take the $\mu(B)$'s of all total choices B of $\mathcal{C}$ such that $\alpha \wedge \beta$ is true in some answer sets of $(L, P \cup \{p \leftarrow \mid p \in B\})$ and make them contribute to both $Pr(\alpha \wedge \beta)$ and $Pr(\alpha)$. □

Proof of Theorem 7.3. Since the number of all total choices B of $\mathcal{C}$ is finite, Algorithm tight_answer always terminates. Let a', b', c', d', i', and v' be the final values of the variables a, b, c, d, i, and v, respectively. If $v' = 0$, then the algorithm has processed all the total choices B of $\mathcal{C}$ with $\mu(B) > 0$. Hence, in this case, by Theorem 7.2, the algorithm returns the exact tight answer for Q to KB. Suppose now $v' > 0$ (and thus $i' \leqslant k$, $b' \neq 0$, and $c' \neq 0$). Then, the returned lower and upper bounds are given by $l' = \frac{a'}{a'+c'} \leqslant \frac{a'+v'}{a'+c'}$ and $u' = \frac{b'}{b'+d'} \leqslant \frac{b'+v'}{b'+d'}$, respectively. Furthermore, the exact tight lower and upper bounds l and u are of the form $l' \leqslant l = \frac{a'+v_a}{a'+v_a+c'-v_c} \leqslant \frac{a'+v'}{a'+c'}$ and $u' \leqslant u = \frac{b'+v_b}{b'+v_b+d'-v_d} \leqslant \frac{b'+v'}{b'+d'}$, respectively. So, $|l - l'| + |u - u'| \leqslant \frac{v'}{a'+c'} + \frac{v'}{b'+d'} \leqslant \epsilon$. □

Proof of Theorem 7.4. Immediate by the proof of Theorem 7.3. □

Proof of Theorem 7.5. We first show membership in $\mathrm{NEXP}^{\mathrm{NP}}$. By Theorem 7.1, we check whether the disjunctive dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent, for every total choice B of $\mathcal{C}$ with $\mu(B) > 0$. Observe then that the number of all total choices B of $\mathcal{C}$ with $\mu(B) > 0$ is exponential in the size of $\mathcal{C}$. As shown in [33], deciding

whether a disjunctive dl-program has an answer set is in $\mathrm{NEXP}^{\mathrm{NP}}$. In summary, this shows that 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 [33], since by Theorem 7.1 a disjunctive dl-program $KB = (L, P)$ has an answer set iff the probabilistic dl-program $KB = (L, P, \mathcal{C}, \mu)$ has an answer set, for the choice space $\mathcal{C} = \{\{a\}\}$, the probability function $\mu(a) = 1$, and any ground atom $a \in HB_{\Phi}$ that does not occur in $ground(P)$. □

Proof of Theorem 7.6. We first show membership in co-$\mathrm{NEXP}^{\mathrm{NP}}$. We show that deciding whether $(\beta|\alpha)[l, u]$ is not a consequence of KB under the answer set semantics is in $\mathrm{NEXP}^{\mathrm{NP}}$. Observe that $(\beta|\alpha)[l, u]$ is not a consequence of KB under the answer set semantics iff there are sets $\mathcal{B}_{\beta\wedge\alpha}$ and $\mathcal{B}_{\beta\wedge\neg\alpha}$ of total choices B of $\mathcal{C}$ with $\mu(B) > 0$ such that either (a.1) $\beta \wedge \alpha$ is true in some answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every $B \in \mathcal{B}_{\beta\wedge\alpha}$, (a.2) $\beta \wedge \neg\alpha$ is false in some answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every $B \in \mathcal{B}_{\beta\wedge\neg\alpha}$, and (a.3) $b > u \cdot (b + d)$, where $b = \sum_{B\in\mathcal{B}_{\beta\wedge\alpha}} \mu(B)$ and $d = 1 - \sum_{B\in\mathcal{B}_{\beta\wedge\neg\alpha}} \mu(B)$, or (b.1) $\beta \wedge \alpha$ is false in some answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every $B \in \mathcal{B}_{\beta\wedge\alpha}$, (a.2) $\beta \wedge \neg\alpha$ is true in some answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every $B \in \mathcal{B}_{\beta\wedge\neg\alpha}$, and (a.3) $a > l \cdot (a + c)$, where $a = 1 - \sum_{B\in\mathcal{B}_{\beta\wedge\alpha}} \mu(B)$ and $c = \sum_{B\in\mathcal{B}_{\beta\wedge\neg\alpha}} \mu(B)$. Since the number of all total choices B of $\mathcal{C}$ with $\mu(B) > 0$ is exponential in the size of $\mathcal{C}$, guessing $\mathcal{B}_{\beta\wedge\alpha}$ and $\mathcal{B}_{\beta\wedge\neg\alpha}$ can be done in nondeterministic exponential time. As shown in [33], deciding whether $\beta \wedge \alpha$ or $\beta \wedge \neg\alpha$ is true or false in some answer set of a disjunctive dl-program is in $\mathrm{NEXP}^{\mathrm{NP}}$. In summary, guessing the sets $\mathcal{B}_{\beta\wedge\alpha}$ and $\mathcal{B}_{\beta\wedge\neg\alpha}$, and verifying that either (a.1)–(a.3) or (b.1)–(b.3) hold is in $\mathrm{NEXP}^{\mathrm{NP}}$. Hence, deciding whether $(\beta|\alpha)[l, u]$ is not a consequence of KB under the answer set semantics is in $\mathrm{NEXP}^{\mathrm{NP}}$. It thus follows that deciding whether $(\beta|\alpha)[l, u]$ is a consequence of KB under the answer set semantics 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 [33], since by Theorem 7.2 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 $KB = (L, P, \mathcal{C}, \mu)$ under the answer set semantics, for the choice space $\mathcal{C} = \{\{a\}\}$, the probability function $\mu(a) = 1$, and any $a \in HB_{\Phi}$ that does not occur in $ground(P)$. □

Proof of Theorem 8.3. As shown in [33], deciding the existence of (and computing) the answer set of a stratified normal dl-program (L, P) with L in *DL-Lite* can be done in polynomial time in the data complexity. Notice then that in the case of data complexity, the choice space $\mathcal{C}$ (and so the set of all its total choices) is fixed. By Theorems 7.1 and 7.2, it thus follows that the problems of (a) deciding whether KB has an answer set, and (b) computing the reals $l, u \in [0, 1]$ for a given ground conditional event $\beta|\alpha$ such that $KB \mid\!\sim_{tight} (\beta|\alpha)[l, u]$ can both be done in polynomial time in the data complexity. □

References

1. F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proceedings AAAI-2005*, pp. 572–577. AAAI Press / MIT Press, 2005.

2. J. Angele, H. Boley, J. de Bruijn, D. Fensel, P. Hitzler, M. Kifer, R. Krummenacher, H. Lausen, A. Polleres, and R. Studer. Web Rule Language (WRL), September 2005. W3C Member Submission. http://www.w3.org/Submission/WRL/.
3. T. Berners-Lee. *Weaving the Web*. Harper, San Francisco, CA, 1999.
4. A. Calì and T. Lukasiewicz. Tightly integrated probabilistic description logic programs for the Semantic Web. In *Proceedings ICLP-2007*, pp. 428–429. *LNCS* 4670, Springer, 2007.
5. A. Calì, T. Lukasiewicz, L. Predoiu, and H. Stuckenschmidt. A framework for representing ontology mappings under probabilities and inconsistency. In *Proceedings URSW-2007*. *CEUR Workshop Proceedings* 327, CEUR-WS.org, 2008.
6. A. Calì, T. Lukasiewicz, L. Predoiu, and H. Stuckenschmidt. Tightly integrated probabilistic description logic programs for representing ontology mappings. In *Proceedings FoIKS-2008*, pp. 178–198. *LNCS* 4932, Springer, 2008.
7. P. C. G. da Costa. *Bayesian Semantics for the Semantic Web*. Doctoral Dissertation, George Mason University, Fairfax, VA, USA, 2005.
8. P. C. G. da Costa and K. B. Laskey. PR-OWL: A framework for probabilistic ontologies. In *Proceedings FOIS-2006*, pp. 237–249. IOS Press, 2006.
9. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. *DL-Lite*: Tractable description logics for ontologies. In *Proceedings AAAI-2005*, pp. 602–607. AAAI Press / MIT Press, 2005.
10. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. Intell. Inf. Syst.*, 10(3):227–252, 1998.
11. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. In *Proceedings KR-2004*, pp. 141–151. AAAI Press, 2004.
12. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artif. Intell.*, in press.
13. J. Euzenat, M. Mochol, P. Shvaiko, H. Stuckenschmidt, O. Svab, V. Svatek, W. R. van Hage, and M. Yatskevich. First results of the ontology alignment evaluation initiative 2006. In *Proceedings ISWC-2006 Workshop on Ontology Matching*.
14. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, Heidelberg, Germany, 2007.
15. J. Euzenat, H. Stuckenschmidt, and M. Yatskevich. Introduction to the ontology alignment evaluation 2005. In *Proceedings K-CAP-2005 Workshop on Integrating Ontologies*.
16. W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proceedings JELIA-2004*, pp. 200–212. *LNCS* 3229, Springer, 2004.
17. 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.
18. A. Finzi and T. Lukasiewicz. Structure-based causes and explanations in the independent choice logic. In *Proceedings UAI-2003*, pp. 225–232. Morgan Kaufmann, 2003.
19. A. M. Frisch and P. Haddawy. Anytime deduction for probabilistic logic. *Artif. Intell.*, 69(1/2):93–122, 1994.
20. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.
21. R. Giugno and T. Lukasiewicz. P-$\mathcal{SHOQ}(\mathbf{D})$: A probabilistic extension of $\mathcal{SHOQ}(\mathbf{D})$ for probabilistic ontologies in the Semantic Web. In *Proceedings JELIA-2002*, pp. 86–97. *LNCS* 2424, Springer, 2002.
22. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings WWW-2003*, pp. 48–57. ACM Press, 2003.

23. S. Heymans, D. V. Nieuwenborgh, D. Vermeir. Nonmonotonic ontological and rule-based reasoning with extended conceptual logic programs. In *Proceedings ESWC-2005*, pp. 392–407. *LNCS* 3532, Springer, 2005.
24. S. Heymans, J. de Bruijn, L. Predoiu, C. Feier, and D. Van Nieuwenborgh. Guarded hybrid knowledge bases. In *CoRR Computing Research Repository*, 2007.
25. I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proceedings ISWC-2003*, pp. 17–29. *LNCS* 2870, Springer, 2003.
26. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web rule language combining OWL and RuleML, May 2004. W3C Member Submission. Available at http://www.w3.org/Submission/SWRL/.
27. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings LPAR-1999*, pp. 161–180. *LNCS* 1705, Springer, 1999.
28. D. Koller, A. Levy, and A. Pfeffer. P-CLASSIC: A tractable probabilistic description logic. In *Proceedings AAAI-1997*, pp. 390–397. AAAI Press / MIT Press, 1997.
29. A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artif. Intell.*, 104(1/2):165–209, 1998.
30. T. Lukasiewicz. Expressive probabilistic description logics. *Artif. Intell.*, 172(6/7):852–883, 2008.
31. T. Lukasiewicz. Probabilistic description logic programs. *Int. J. Approx. Reason.*, 45(2):288–307, 2007.
32. T. Lukasiewicz. Tractable probabilistic description logic programs. In *Proceedings SUM-2007*, pp. 143–156. *LNCS* 4772, Springer, 2007.
33. T. Lukasiewicz. A novel combination of answer set programming with description logics for the Semantic Web. In *Proceedings ESWC-2007*, pp. 384-398. *LNCS* 4519, Springer, 2007.
34. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Repairing ontology mappings. In *Proceedings AAAI-2007*, pp. 1408–1413. AAAI Press, 2007.
35. M. Milicic. Planning in action formalisms based on DLs: First results. In *Proceedings DL-2007*. CEUR-WS.org, 2007.
36. B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60, 2005.
37. B. Motik, I. Horrocks, R. Rosati, and U. Sattler. Can OWL and logic programming live together happily ever after? In *Proceedings ISWC-2006*, pp. 501–514. *LNCS* 4273, Springer, 2006.
38. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.
39. D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artif. Intell.*, 64(1):81–129, 1993.
40. D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artif. Intell.*, 94(1/2):7–56, 1997.
41. L. Predoiu and H. Stuckenschmidt. A probabilistic framework for information integration and retrieval on the Semantic Web. In *Proceedings InterDB-2007 Workshop on Database Interoperability*, 2007.
42. R. Rosati. $\mathcal{DL}+log$: Tight integration of description logics and disjunctive Datalog. In *Proceedings KR-2006*, pp. 68–78. AAAI Press, 2006.
43. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *J. Web Sem.*, 3(1):61–73, 2005.
44. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, 2nd edition. Prentice Hall, 2002.
45. L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping languages for terminological knowledge. In *Proceedings IJCAI-2005*, pp. 576–581. 2005.

46. P. Wang and B. Xu. Debugging ontology mapping: A static method. *Computation and Intelligence*, 2007. To appear.
47. W3C. OWL web ontology language overview, 2004. W3C Recommendation (10 February 2004). `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.