Annals of Mathematics and Artificial Intelligence manuscript No.
(will be inserted by the editor)

Tightly Integrated Probabilistic Description Logic Programs for Representing Ontology Mappings*

Thomas Lukasiewicz [1] · **Livia Predoiu** [2] · **Heiner Stuckenschmidt** [3]

Received: 9 March 2009 / Accepted: 29 May 2011

Abstract Creating mappings between ontologies is a common way of approaching the semantic heterogeneity problem on the Semantic Web. To fit into the landscape of Semantic Web languages, a suitable, logic-based representation formalism for mappings is needed. We argue that such a formalism has to be able to deal with uncertainty and inconsistencies in automatically created mappings. We analyze the requirements for such a formalism, and we propose a novel approach to probabilistic description logic programs as such a formalism, which tightly combines normal logic programs under the well-founded semantics with both tractable ontology languages and Bayesian probabilities. We define the language, and we show that it can be used to resolve inconsistencies and merge mappings from different matchers based on the level of confidence assigned to different rules. Furthermore, we explore the semantic and computational aspects of probabilistic description logic programs under the well-founded semantics. In particular, we show that the well-founded semantics approximates the answer set semantics. We also describe algorithms for consistency checking and tight query processing, and we analyze the data and general complexity of these two central computational problems. As a crucial property, the novel tightly integrated probabilistic description logic programs under the well-founded semantics allow for tractable consistency checking and for tractable tight query processing in the data complexity, and they even have a first-order rewritable (and thus LOGSPACE data complexity) special case, which is especially interesting for representing ontology mappings.

[1] Department of Computer Science, University of Oxford, UK.
E-mail: thomas.lukasiewicz@cs.ox.ac.uk.
[2] Institut für Technische und Betriebliche Informationssysteme, Universität Magdeburg, Germany.
E-mail: predoiu@ovgu.de.
[3] Institut für Informatik, Universität Mannheim, Germany.
E-mail: heiner@informatik.uni-mannheim.de.

* This article is a significantly extended and revised version of two papers that appeared in *Proc. URSW-2007* [3] and *Proc. FoIKS-2008* [5].

1 Introduction

The problem of aligning heterogeneous ontologies via semantic mappings has been identified as one of the major challenges of Semantic Web technologies. 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 [55]. In 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., [21]). 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 [18,7,19,20,22]. 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 [21].

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 individual elements);
- 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 mappings. In this paper, we address the problem of designing a mapping representation language that is capable of representing the kinds of uncertainty mentioned above. We propose an approach to such a language, which is based on an integration of ontologies and rules under probabilistic uncertainty.

There is a large body of work on integrating ontologies and rules, which is a promising way of representing mappings between ontologies. One type of integration is to build rules on top of ontologies, i.e., rule-based systems that use vocabulary from ontology knowledge bases. Another form of integration is to build ontologies on top of rules, where ontological definitions are supplemented by rules or imported from rules. Both types of integration have been realized in recent hybrid integrations of rules and ontologies, called *description logic programs* (or *dl-programs*), which have the form $KB = (L, P)$, where L is a description logic knowledge base, and P is a finite set of rules involving either queries to L in a loose integration [16,15] or concepts and roles from L as unary resp. binary predicates in a tight integration [36,37] (for more detailed overviews, see especially [16,43]).

Other works explore formalisms for *uncertainty reasoning in the Semantic Web* (an important recent forum for approaches to uncertainty in the Semantic Web is the annual *Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*; there also exists a W3C Incubator Group on *Uncertainty Reasoning for the World Wide Web*). There are especially probabilistic extensions of description logics [26,38], Web ontology languages [9,10], and dl-programs [39] (to encode ambiguous information, such as "John is a student with probability 0.7 and a teacher with probability 0.3", which is very different from vague/fuzzy infor-

mation, such as "John is tall with degree of truth 0.7"). In particular, [39] extends the loosely integrated dl-programs of [16, 15] by probabilistic uncertainty as in Poole's independent choice logic (ICL) [47]. 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, and normal form games. It also allows for natural notions of causes and explanations as in Pearl's structural causal models [24].

In this paper, we explore the use of *tightly integrated probabilistic dl-programs under the answer set semantics and under the well-founded semantics* as a language for representing and reasoning with uncertain and possibly inconsistent ontology mappings. The approach is a tight integration of normal logic programs under the answer set and under the well-founded semantics, the tractable ontology language *DL-Lite*$_\mathcal{A}$ (which is a fragment the standard Web ontology language OWL), and Bayesian probabilities. More concretely, the tight integration between ontology and rule languages of [37] is combined with probabilistic uncertainty as in the ICL [47]. The main contributions of this paper can be summarized as follows:

- We explore the use of tightly integrated probabilistic dl-programs under the answer set semantics and under the well-founded semantics as a language for representing and reasoning with ontology mappings. Here, we aim especially at a very efficient framework, and thus (i) assume *DL-Lite*$_\mathcal{A}$ as underlying ontology language, and (ii) allow nonmonotonic negation in rule bodies but disallow disjunctions in rule heads.
- We define the well-founded semantics for tightly integrated probabilistic dl-programs and provide several semantic results around it. In particular, we show that (i) consistency under the answer set semantics implies consistency under the well-founded semantics, and that (ii) the well-founded semantics is an approximation of the answer set semantics, which is exact in certain cases. We also show that under certain conditions, tight query processing yields point probabilities rather than interval probabilities.
- We analyze the computational aspects of tightly integrated probabilistic dl-programs under the well-founded semantics. In particular, we describe algorithms for consistency checking and tight query processing, which are both based on fixpoint iterations for computing the well-founded semantics of normal dl-programs. We also analyze the data and the general complexity of consistency checking and correct query processing, which both turn out to be complete for P in the data complexity and for EXP in general, and thus have a lower complexity than their counterparts under the answer set semantics.
- As a crucial property, the novel tightly integrated probabilistic dl-programs under the well-founded semantics allow for tractable consistency checking and for tractable tight query processing in the data complexity. Furthermore, we delineate even a special case of first-order rewritability (and thus $\mathrm{LOGSPACE}$ data complexity), which is especially interesting for ontology mapping, since it informally models the case of mapping several input ontologies into an output ontology via an acyclic normal program.

The rest of this paper is structured as follows. In Section 2, we analyze the requirements of an ontology mapping language. Section 3 briefly reviews tractable ontology languages as a basis for representing ontologies to be connected by mappings. In Sections 4 and 5, we describe tightly integrated dl-programs as a basis for representing mappings between ontologies as logical rules, and explain how the rule language supports the refinement and repair of oversimplifying or inconsistent mappings. Sections 6 and 7 present a probabilistic extension thereof, and show that it can be used to represent and combine confidence values of different matchers in terms of error probabilities and to resolve inconsistencies by using trust probabilities. Sections 8 and 9 address the computational aspects of reasoning in the

novel language. In particular, Section 9 delineates a first-order rewritable case. Section 11 discusses related work, and in Section 10, we summarize the main results and give an outlook on future research. The proofs of all results in this paper are given in Appendix A.

2 Representation Requirements

The problem of ontology matching can be defined as follows [21]. Ontologies are theories encoded in a certain language L. For efficiency reasons, we assume here that ontologies are encoded in the tractable description logic *DL-Lite*$_\mathcal{A}$ [46], which is a fragment of OWL.

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, n) 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);
- n is a degree of confidence in the correctness of the correspondence.

In this paper, we develop a formal language for representing and combining correspondences that are produced by different matching components or systems. From the above general description of automatically generated correspondences between ontologies, we can derive a number of requirements for such 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 different ontologies has to be tightly integrated with the semantics of the used ontology language (in this case *DL-Lite*$_\mathcal{A}$). 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 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 at least for relevant cases.

Throughout the paper, we use real data from the Ontology Alignment Evaluation Initiative (OAEI)[1] to illustrate the different aspects of mapping representation. In particular, we use examples from the benchmark and the conference dataset. The former consists of five OWL ontologies (tests 101 and 301–304) describing scientific publications and related information, while the latter 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.

3 Tractable Ontology Languages

As underlying ontology language, we use the tractable description logic *DL-Lite*$_\mathcal{A}$ [46], which adds datatypes to a restricted combination of the tractable description logics *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$. All these description logics belong to the *DL-Lite* family [6] of tractable description logics, which are a class of restricted description logics for which the main reasoning tasks

are possible in polynomial time in general and some of them even in LOGSPACE in the data complexity. The *DL-Lite* description logics are fragments of OWL and the most common tractable ontology languages in the Semantic Web context. They are especially directed towards data-intensive applications. Intuitively, description logics model a domain of interest in terms of concepts and roles, which represent especially classes of individuals and binary relations between classes of individuals, respectively. A 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.

We now recall the syntax and the semantics of *DL-Lite*$_\mathcal{A}$.

3.1 Syntax

As for the elementary ingredients of *DL-Lite*$_\mathcal{A}$, let $\mathbf{D}$ be a finite set of *atomic datatypes* d, which are associated with pairwise disjoint sets of *data values* $\mathbf{V}_d$. Let $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ be pairwise disjoint sets of *atomic concepts*, *atomic roles*, *atomic attributes*, and *individuals*, respectively, and let $\mathbf{V}$ denote the union of all $\mathbf{V}_d$ with $d \in \mathbf{D}$.

Roles, concepts, attributes, and datatypes are as follows:

- A *basic role* Q is either an atomic role $P \in \mathbf{R}_A$ or its inverse P^-. A *(general) role* R is either a basic role Q or the negation of a basic role $\neg Q$.
- A *basic concept* B is either an atomic concept $A \in \mathbf{A}$, or an existential restriction on a basic role Q, denoted $\exists Q$, or the domain of an atomic attribute $U \in \mathbf{R}_D$, denoted $\delta(U)$. A *(general) concept* C is either the *universal concept* $\top_C$, or a basic concept B, or the negation of a basic concept $\neg B$, or an existential restriction on a basic role Q of the form $\exists Q.C$, where C is a concept.

[1] http://oaei.ontologymatching.org

- A *(general) attribute* V is either an atomic attribute $U \in \mathbf{R}_D$ or the negation of an atomic attribute $\neg U$.
- A *basic datatype* E is the range of an atomic attribute $U \in \mathbf{R}_D$, denoted $\rho(U)$. A *(general) datatype* F is either the *universal datatype* $\top_D$ or an atomic datatype $d \in \mathbf{D}$.

An *axiom* is an expression of one of the following forms:

- $B \sqsubseteq C$ (*concept inclusion axiom*), where B is a basic concept, and C is a concept;
- $Q \sqsubseteq R$ (*role inclusion axiom*), where Q is a basic role, and R is a role;
- $U \sqsubseteq V$ (*attribute inclusion axiom*), where U is an atomic attribute, and V is an attribute;
- $E \sqsubseteq F$ (*datatype inclusion axiom*), where E is a basic datatype, and F is a datatype;
- $(\mathsf{funct}\ Q)$ (*role functionality axiom*), where Q is a basic role;
- $(\mathsf{funct}\ U)$ (*attribute functionality axiom*), where U is an atomic attribute;
- $A(a)$ (*concept membership axiom*), where A is an atomic concept and $a \in \mathbf{I}$;
- $P(a, b)$ (*role membership axiom*), where P is an atomic role and $a, b \in \mathbf{I}$;
- $U(a, v)$ (*attribute membership axiom*), where U is an atomic attribute, $a \in \mathbf{I}$, and $v \in \mathbf{V}$.

A *constraint* is either a concept inclusion axiom of the form $B \sqsubseteq \neg B'$, a role inclusion axiom of the form $Q \sqsubseteq \neg Q'$, an attribute inclusion axiom of the form $U \sqsubseteq \neg U'$, a role functionality axiom $(\mathsf{funct}\ Q)$, or an attribute functionality axiom $(\mathsf{funct}\ U)$.

We next define knowledge bases, which consist of a restricted finite set of inclusion and functionality axioms, called TBox, and a finite set of membership axioms, called ABox. We also define queries to such knowledge bases.

We first define the restriction on inclusion and functionality axioms. A basic role Q (resp., atomic attribute U) is an *identifying property* in a set of axioms $\mathcal{S}$ iff $\mathcal{S}$ contains a functionality axiom $(\mathsf{funct}\ Q)$ (resp., $(\mathsf{funct}\ U)$). Given an inclusion axiom α of the form $X \sqsubseteq Y$ (resp., $X \sqsubseteq \neg Y$), a basic role (resp., atomic attribute) Y appears *positively* (resp., *negatively*) in the right-hand side of α. A basic role (resp., atomic attribute) is *primitive* in $\mathcal{S}$ iff it does not appear positively in the right-hand side of an inclusion axiom in $\mathcal{S}$ and it does not appear in an expression of the form $\exists Q.C$ in $\mathcal{S}$.

We can now define knowledge bases. A *TBox* is a finite set $\mathcal{T}$ of inclusion and functionality axioms such that every identifying property in $\mathcal{T}$ is primitive. Intuitively, identifying properties cannot be specialized in $\mathcal{T}$, i.e., they cannot appear positively in the right-hand side of inclusion axioms in $\mathcal{T}$. An *ABox* $\mathcal{A}$ is a finite set of membership axioms. A *knowledge base* $L = \mathcal{T} \cup \mathcal{A}$ is the union of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

Example 1 (Scientific Database). We use a knowledge base $L = \mathcal{T} \cup \mathcal{A}$ in *DL-Lite*$_\mathcal{A}$ to specify some simple information about scientists and their publications. Consider the following sets of atomic concepts, atomic roles, atomic attributes, individuals, and data values:

$$\begin{aligned}
\mathbf{A} &= \{\mathit{Scientist}, \mathit{Article}, \mathit{ConferencePaper}, \mathit{JournalPaper}\},\\
\mathbf{R}_A &= \{\mathit{hasAuthor}, \mathit{isAuthorOf}, \mathit{hasFirstAuthor}\},\\
\mathbf{R}_D &= \{\mathit{name}, \mathit{title}, \mathit{yearOfPublication}\},\\
\mathbf{I} &= \{i_1, i_2\},\\
\mathbf{V} &= \{\text{“}\mathit{mary}\text{”}, \text{“}\mathit{Semantic\ Web\ search}\text{”}, 2008\}.
\end{aligned}$$

The TBox $\mathcal{T}$ contains the subsequent axioms, which informally express that (1) conference and journal papers are articles, (2) conference papers are not journal papers, (3) *isAuthorOf* relates scientists and articles, (4) *isAuthorOf* is the inverse of *hasAuthor*, i.e., (*scientist*, *article*) belongs to *isAuthorOf* iff (*article*, *scientist*) belongs to *hasAuthor*, and (5) *hasFirstAu-*

thor is a functional binary relationship:

(1) $\mathit{ConferencePaper} \sqsubseteq \mathit{Article}$, $\mathit{JournalPaper} \sqsubseteq \mathit{Article}$,
(2) $\mathit{ConferencePaper} \sqsubseteq \neg \mathit{JournalPaper}$,
(3) $\exists \mathit{isAuthorOf} \sqsubseteq \mathit{Scientist}$, $\exists \mathit{isAuthorOf}^- \sqsubseteq \mathit{Article}$,
(4) $\mathit{isAuthorOf}^- \sqsubseteq \mathit{hasAuthor}$, $\mathit{hasAuthor}^- \sqsubseteq \mathit{isAuthorOf}$,
(5) $(\mathsf{funct}\ \mathit{hasFirstAuthor})$.

The ABox $\mathcal{A}$ contains the following axioms, which express that (6) i_1 is a scientist whose name is "*mary*" and who is the author of i_2, (7) i_2 is an article entitled "*Semantic Web search*", and (8) i_2 has been published in the year 2008:

(6) $\mathit{Scientist}(i_1)$, $\mathit{name}(i_1, \text{"}\mathit{mary}\text{"})$, $\mathit{isAuthorOf}(i_1, i_2)$,
(7) $\mathit{Article}(i_2)$, $\mathit{title}(i_2, \text{"}\mathit{Semantic\ Web\ search}\text{"})$,
(8) $\mathit{yearOfPublication}(i_2, 2008)$.

3.2 Semantics

The semantics of *DL-Lite*$_\mathcal{A}$ is defined in terms of standard first-order interpretations as usual. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of (i) a nonempty domain $\Delta^\mathcal{I} = (\Delta^\mathcal{I}_O, \Delta^\mathcal{I}_V)$, which is the disjoint union of the *domain of objects* $\Delta^\mathcal{I}_O$ and the *domain of values* $\Delta^\mathcal{I}_V = \bigcup_{d \in \mathbf{D}} \Delta^\mathcal{I}_d$, where the $\Delta^\mathcal{I}_d$'s are pairwise disjoint domains of values for the datatypes $d \in \mathbf{D}$, and (ii) a mapping $\cdot^\mathcal{I}$ that assigns to each datatype $d \in \mathbf{D}$ its domain of values $\Delta^\mathcal{I}_d$, to each data value $v \in \mathbf{V}_d$ an element of $\Delta^\mathcal{I}_d$ (such that $v \neq w$ implies $v^\mathcal{I} \neq w^\mathcal{I}$), to each atomic concept $A \in \mathbf{A}$ a subset of $\Delta^\mathcal{I}_O$, to each atomic role $P \in \mathbf{R}_A$ a subset of $\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O$, to each atomic attribute $P \in \mathbf{R}_D$ a subset of $\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_V$, to each individual $a \in \mathbf{I}$ an element of $\Delta^\mathcal{I}_O$ (such that $a \neq b$ implies $a^\mathcal{I} \neq b^\mathcal{I}$). Note that different data values (resp., individuals) are associated with different elements of $\Delta^\mathcal{I}_V$ (resp., $\Delta^\mathcal{I}_O$) (*unique name assumption*). The extension of $\cdot^\mathcal{I}$ to all concepts, roles, attributes, and datatypes, and the *satisfaction* of an axiom α in the interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, denoted $\mathcal{I} \models \alpha$, are defined as follows:

- $(\top_D)^\mathcal{I} = \Delta^\mathcal{I}_V$ and $(\top_C)^\mathcal{I} = \Delta^\mathcal{I}_O$,
- $(\neg U)^\mathcal{I} = (\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_V) \setminus U^\mathcal{I}$,
- $(\neg Q)^\mathcal{I} = (\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O) \setminus Q^\mathcal{I}$,
- $(\rho(U))^\mathcal{I} = \{v \in \Delta^\mathcal{I}_V \mid \exists o\colon (o, v) \in U^\mathcal{I}\}$,
- $(\delta(U))^\mathcal{I} = \{o \in \Delta^\mathcal{I}_O \mid \exists v\colon (o, v) \in U^\mathcal{I}\}$,
- $(P^-)^\mathcal{I} = \{(o, o') \in \Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O \mid (o', o) \in P^\mathcal{I}\}$,
- $(\exists P)^\mathcal{I} = \{o \in \Delta^\mathcal{I}_O \mid \exists o'\colon (o, o') \in P^\mathcal{I}\}$,
- $(\exists Q.C)^\mathcal{I} = \{o \in \Delta^\mathcal{I}_O \mid \exists o'\colon (o, o') \in Q^\mathcal{I},\ o' \in C^\mathcal{I}\}$,
- $(\neg B)^\mathcal{I} = \Delta^\mathcal{I}_O \setminus B^\mathcal{I}$.

The *satisfaction* of an axiom α in $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, denoted $\mathcal{I} \models \alpha$, is defined as follows:

- $\mathcal{I} \models B \sqsubseteq C$ iff $B^\mathcal{I} \subseteq C^\mathcal{I}$,
- $\mathcal{I} \models Q \sqsubseteq R$ iff $Q^\mathcal{I} \subseteq R^\mathcal{I}$,
- $\mathcal{I} \models E \sqsubseteq F$ iff $E^\mathcal{I} \subseteq F^\mathcal{I}$,
- $\mathcal{I} \models U \sqsubseteq V$ iff $U^\mathcal{I} \subseteq V^\mathcal{I}$,
- $\mathcal{I} \models (\mathsf{funct}\ Q)$ iff $(o, q), (o, q') \in Q^\mathcal{I}$ implies $q = q'$,
- $\mathcal{I} \models (\mathsf{funct}\ U)$ iff $(o, v), (o, v') \in U^\mathcal{I}$ implies $v = v'$,
- $\mathcal{I} \models A(a)$ iff $a^\mathcal{I} \in A^\mathcal{I}$,

– $\mathcal{I} \models P(a,b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$,
– $\mathcal{I} \models U(a,v)$ iff $(a^{\mathcal{I}}, v^{\mathcal{I}}) \in U^{\mathcal{I}}$.

We say $\mathcal{I}$ *satisfies* the axiom α, or $\mathcal{I}$ is a *model* of α, iff $\mathcal{I} \models \alpha$. We say $\mathcal{I}$ *satisfies* a knowledge base L, or $\mathcal{I}$ is a *model* of L, denoted $\mathcal{I} \models L$, iff $\mathcal{I} \models \alpha$ for all $\alpha \in L$. We say L is *satisfiable* (resp., *unsatisfiable*) iff L has a (resp., no) model. An axiom α is a *logical consequence* of L, denoted $L \models \alpha$, iff every model of L satisfies α.

As shown in [46], in particular, deciding the satisfiability of knowledge bases in *DL-Lite*$_{\mathcal{A}}$ and deciding logical consequences of membership axioms from knowledge bases in *DL-Lite*$_{\mathcal{A}}$ can both be done in LOGSPACE in the size of the ABox in the data complexity.

*Example 2 (Scientific Database cont'd).*It is not difficult to verify that the knowledge base L of Example 1 is satisfiable, and that the two axioms *JournalPaper* $\sqsubseteq \neg$*ConferencePaper* and *hasAuthor*(i_2, i_1) are logical consequences of L.

4 Tightly Integrated Normal DL-Programs

In this section, we recall the *tightly integrated* approach to *description logic programs* (or simply *dl-programs*) $KB = (L, P)$ under the answer set semantics and under the well-founded semantics from [37]. However, in contrast to [37], where KB consists of a description logic knowledge base L and a disjunctive logic program P, here, for efficiency reasons, we restrict our attention to the special case where L is a description logic knowledge base in *DL-Lite*$_{\mathcal{A}}$, and P is a normal logic program. The semantics of such tightly integrated dl-programs is defined in a modular way as in [16, 15], but it allows for a much tighter integration 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 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 dl-programs are decidable, even when their components of logic programs and description logic knowledge bases are both very expressive. We refer especially to [37] for further details on the novel approach to dl-programs and for a detailed comparison to related works.

4.1 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 $\mathbf{V}$ and pairwise disjoint (denumerable) sets $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ of atomic concepts, atomic roles, atomic attributes, and individuals, respectively, as in Section 3. 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 *classical literal* (or simply *literal*) l is an atom a or a negated atom $\neg a$. A *negation-as-failure literal* (or *NAF-literal*) is an atom a or a default-negated atom $not\ a$. A *normal rule* (or simply *rule*) r is an expression of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_n, not\ \beta_{n+1}, \ldots, not\ \beta_{n+m}\ , \tag{1}$$

where $\alpha, \beta_1, \ldots, \beta_{n+m}$ are atoms and $m, n \geqslant 0$. We call α the *head* of r, denoted $H(r)$, while the conjunction $\beta_1, \ldots, \beta_n, not\ \beta_{n+1}, \ldots, not\ \beta_{n+m}$ is its *body*. We define $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{\beta_1, \ldots, \beta_n\}$ and $B^-(r) = \{\beta_{n+1}, \ldots, \beta_{n+m}\}$. A *fact* is a rule of the form (1) with $n = m = 0$. A *normal program* P is a finite set of normal rules of the form (1). We say P is *positive* iff $m = 0$ for all normal rules (1) in P. A *tightly integrated normal description logic program* (or simply *normal dl-program*) $KB = (L, P)$ consists of a description logic knowledge base L in *DL-Lite*$_\mathcal{A}$ and a normal program P. We say KB is a *positive dl-program* iff P is positive.

Example 3 Consider the normal dl-program $KB = (L, P)$, where L is the description logic knowledge base from Example 1, and P is the following collection of rules, which model a part of the paper assignment in a reviewing process: (1) candidate reviewers for a paper are all those referees who are experts in an area of the paper and who are not in a conflict situation on this paper, (2) an expert in an area is someone who has written at least three papers in that area, (3) someone is in a conflict situation on a paper if she is a co-author of an author of the paper, (4) any two authors of the same paper are co-authors, and (5) the paper p_0 is in the Semantic Web area, and John is a referee, who has reviewed the three papers p_1, p_2, and p_3, which are all in the Semantic Web area.

(1) $cand(X, Q) \leftarrow paperArea(Q, A),\ referee(X),\ expert(X, A),\ not\ conflict(X, Q)$;
(2) $expert(X, A) \leftarrow isAuthorOf(X, Q_1),\ isAuthorOf(X, Q_2),\ isAuthorOf(X, Q_3),$
$inArea(Q_1, A),\ inArea(Q_2, A),\ inArea(Q_3, A),\ Q_1 \neq Q_2,\ Q_2 \neq Q_3,\ Q_1 \neq Q_3$;
(3) $conflict(X, Q) \leftarrow co\text{-}author(X, Y),\ isAuthorOf(Y, Q)$;
(4) $co\text{-}author(X, Y) \leftarrow isAuthorOf(X, Q),\ isAuthorOf(Y, Q)$;
(5) $paperArea(p_0, \text{“}SemanticWeb\text{”});\ referee(john);\ isAuthorOf(john, p_1);$
$isAuthorOf(john, p_2);\ isAuthorOf(john, p_3);\ inArea(p_1, \text{“}SemanticWeb\text{”});$
$inArea(p_2, \text{“}SemanticWeb\text{”});\ inArea(p_3, \text{“}SemanticWeb\text{”})$.

The above normal dl-program also shows the advantages and flexibility of the tight integration between rules and ontologies (compared to the loose integration in [16,15]): Observe that the predicate symbol $isAuthorOf$ 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 [16,15]). Furthermore, we can easily use the description logic knowledge base L to express additional constraints on the predicate symbols in P. For example, we may use the two concept inclusion axioms $\exists conflict \sqsubseteq Scientist$ and $\exists conflict^{-1} \sqsubseteq Article$ in L to express that the relationship for conflict situations in P relates only scientists and articles.

4.2 Answer Set Semantics

The answer set semantics of normal dl-programs generalizes the answer set semantics of ordinary normal programs. In the sequel, let $KB = (L, P)$ be a normal dl-program.

A *ground instance* of a rule $r \in P$ is obtained from r by replacing every variable that occurs in r by a constant symbol from Φ_c. We denote by $ground(P)$ the set of all ground instances of rules in P. The *Herbrand base* relative to Φ, denoted HB_Φ, is the set of all ground atoms constructed with constant and predicate symbols from Φ. We use DL_Φ to denote the set of all ground atoms in HB_Φ that are constructed from atomic concepts in $\mathbf{A}$, atomic roles in $\mathbf{R}_A$, and atomic attributes 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 $I \models B(r)$, i.e., $I \models \beta$ 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 dl-program $KB = (L, P)$, denoted $I \models KB$, iff I is a model of both L and P. A dl-program $KB = (L, P)$ is *satisfiable* iff there exists a model I of KB.

We now define the answer set semantics of normal dl-programs by generalizing the ordinary answer set semantics of normal programs. We generalize the definition via the FLP-reduct [23], which is equivalent to the standard definition via the Gelfond-Lifschitz reduct [25]. Given a dl-program $KB = (L, P)$, the *FLP-reduct* of KB relative to $I \subseteq HB_\Phi$, denoted KB^I, is the dl-program (L, P^I), where P^I is the set of all $r \in ground(P)$ with $I \models B(r)$. Note that the *Gelfond-Lifschitz reduct* of KB relative to $I \subseteq HB_\Phi$ is the positive dl-program $(L, \hat{P}^I)$, where $\hat{P}^I$ is obtained from $ground(P)$ by (i) deleting every rule r such that $I \models \beta$ for some $\beta \in B^-(r)$ and (ii) deleting the negative body from each remaining rule. An interpretation $I \subseteq HB_\Phi$ is an *answer set* of KB iff I is a minimal model of KB^I. A dl-program KB is *consistent* (resp., *inconsistent*) iff it has an (resp., no) answer set.

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

Example 4 Consider again the normal dl-program $KB = (L, P)$ from Example 3. Then, it is not difficult to verify that KB has a unique answer set I_{KB}, and thus KB is consistent. Furthermore, this answer set I_{KB} is in particular a model of all facts in P and of the literal $\neg conflict(john, p_0)$. Consequently, I_{KB} also satisfies $cand(john, p_0)$, which implies that $cand(john, p_0)$ is both a cautious and a brave consequence of KB under the answer set semantics. Informally, John is a candidate reviewer for paper p_0, since we are unaware of any conflict situation of John on the paper p_0.

We next summarize some important semantic properties of normal dl-programs under the above answer set semantics. In the ordinary case, every answer set of a normal program P is also a minimal model of P, and the converse holds when P is positive. This also holds for normal dl-programs. Furthermore, like ordinary positive programs, every satisfiable positive dl-program has a unique answer set, which coincides with the least model.

As another important semantic property, the answer set semantics of normal dl-programs faithfully extends its ordinary counterpart. That is, the answer set semantics of a normal dl-program with empty description logic knowledge base coincides with the ordinary answer set semantics of its normal program.

Furthermore, the answer set semantics of normal 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 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 $\vee$.

4.3 Well-Founded Semantics

We next recall the well-founded semantics for normal dl-programs [37], which generalizes the well-founded semantics for ordinary normal programs via unfounded sets [57]. Intuitively, compared to ordinary normal programs, normal dl-programs KB additionally have a knowledge base L in *DL-Lite*$_{\mathcal{A}}$. In the definition of the well-founded semantics for KB, one part of L is considered in a similar way as ordinary normal rules, while the other part (namely, the set of all constraints in L) is considered only in a final step.

For knowledge bases L in *DL-Lite*$_{\mathcal{A}}$, we denote by L^+ the knowledge base obtained from L by removing all constraints. For literals $l = a$ (resp., $l = \neg a$), we use $\neg.l$ to denote $\neg a$ (resp., a), and for sets of literals S, we define $\neg.S = \{\neg.l \mid l \in S\}$ and $S^+ = \{a \in S \mid a \text{ is an atom}\}$. We use $Lit_\Phi = HB_\Phi \cup \neg.HB_\Phi$ to denote the set of all ground literals with predicate and constant symbols from Φ. A set of ground literals $S \subseteq Lit_\Phi$ is *consistent* iff $S \cap \neg.S = \emptyset$. A *(three-valued) interpretation* relative to Φ is any consistent set of ground literals $I \subseteq Lit_\Phi$.

We first define the notion of an unfounded set for normal dl-programs $KB = (L, P)$. Let $I \subseteq Lit_\Phi$ be consistent. A set $U \subseteq HB_\Phi$ is an *unfounded set* of KB relative to I iff

$(*)$ for every $a \in U$,
  (a) for every $r \in ground(P)$ with $H(r) = a$, either
    (i) $\neg b \in I \cup \neg.U$ for some $b \in B^+(r)$, or
    (ii) $b \in I$ for some $b \in B^-(r)$; and
  (b) $L^+ \cup S^+ \not\models a$ for every consistent $S \subseteq Lit_\Phi$ with $I \cup \neg.U \subseteq S$.

Intuitively, all the atoms of the unfounded set U of KB relative to I can be safely set to false under I. Here, compared to unfounded sets of ordinary normal programs, the condition (b) is new, which intuitively says that a will never become true via the knowledge base L^+, if we expand I (to S) in a way such that all unfounded atoms are kept false. In $L^+ \cup S^+$, we only have to consider S^+, since the negated atoms in S (as long as consistent with $L^+ \cup S^+$) do not enlarge the set of positive atoms logically entailed by $L^+ \cup S^+$.

The set of unfounded sets of KB relative to I is closed under union, which implies that KB has a greatest unfounded set relative to I. Based on this result, we now generalize the operators T_P, U_P, and W_P from ordinary normal programs to normal dl-programs as follows. We define the operators T_{KB}, U_{KB}, and W_{KB} on all consistent $I \subseteq Lit_\Phi$ by:

- $a \in T_{KB}(I)$ iff either
  (a) $a \in HB_\Phi$ and some $r \in ground(P)$ exists such that
    (i) $H(r) = a$,
    (ii) $b \in I$ for all atoms $b \in B^+(r)$, and
    (iii) $\neg b \in I$ for all atoms $b \in B^-(r)$, or
  (b) $L^+ \cup I^+ \models a$;
- $U_{KB}(I)$ is the greatest unfounded set of KB relative to I; and
- $W_{KB}(I) = T_{KB}(I) \cup \neg.U_{KB}(I)$.

Intuitively, $T_{KB}(I)$ is the set of all (positive) ground atoms that follow either (a) from P under I in one step or (b) from L^+ under I, while $W_{KB}(I)$ additionally collects all negated ground atoms $\neg a$ such that a belongs to the greatest unfounded set of KB relative to I. Here, compared to the well-founded semantics of ordinary normal programs, the condition (b) $L^+ \cup I^+ \models a$ in the definition of $T_{KB}(I)$ is new. Intuitively, in addition to being implied by P under I, positive ground atoms may also be implied by L^+ under I.

It is not difficult to verify that the three operators T_{KB}, U_{KB}, and W_{KB} are all monotonic. Thus, in particular, W_{KB} has a least fixpoint, denoted $lfp(W_{KB})$. The well-founded

semantics of normal dl-programs can thus be defined as follows. Let $KB = (L, P)$ be a normal dl-program. The *well-founded semantics* of KB, denoted $WFS(KB)$, is defined as $lfp(W_{KB})$, if $L \cup lfp(W_{KB})$ is satisfiable, and it is undefined, otherwise. We then say that KB is *consistent* and *inconsistent under the well-founded semantics* (or *w-consistent* and *w-inconsistent*), respectively. An atom $a \in HB_\Phi$ is *well-founded* (resp., *unfounded*) relative to KB iff a (resp., $\neg a$) belongs to $WFS(KB)$. A literal $\ell \in HB_\Phi \cup \neg.HB_\Phi$ is a *consequence* of a normal dl-program KB under the well-founded semantics iff $\ell \in WFS(KB)$.

Example 5 Consider again the normal dl-program $KB = (L, P)$ from Example 3. Then, it is not difficult to verify that KB has a total well-founded semantics (i.e., each atom is either well-founded or unfounded), which coincides with the unique answer set I_{KB} from Example 4, and thus KB is w-consistent. Furthermore, in particular, all facts in P and the atom $cand(john, p_0)$ are well-founded relative to KB, while the atom $conflict(john, p_0)$ is unfounded, i.e., the literal $\neg conflict(john, p_0)$ belongs to the well-founded semantics of KB.

We next summarize some important semantic properties of the well-founded semantics for normal dl-programs. As a first such property, the well-founded semantics of normal dl-programs faithfully extends the well-founded semantics of ordinary normal programs.

Furthermore, the well-founded semantics for dl-programs can also be characterized in terms of the least and the greatest fixpoint of a monotonic operator γ^2_{KB} similar as the ordinary well-founded semantics [1]. This characterization can then be used to derive further properties of the well-founded semantics for dl-programs. For a dl-program $KB = (L, P)$, the application of the operator γ_{KB} on $I \subseteq HB_\Phi$, denoted $\gamma_{KB}(I)$, is the least model of (L^+, P^I). It can be shown that γ_{KB} is anti-monotonic [37], like its counterpart for ordinary normal programs [1]. Hence, the operator $\gamma^2_{KB}(I) = \gamma_{KB}(\gamma_{KB}(I))$, for all $I \subseteq HB_\Phi$, is monotonic and thus has a least and a greatest fixpoint, denoted $lfp(\gamma^2_{KB})$ and $gfp(\gamma^2_{KB})$, respectively. These fixpoints characterize the well-founded semantics of KB as follows.

Theorem 1 (see [37]) *Let $KB = (L, P)$ be a normal dl-program. Then, (a) KB is w-consistent iff $L \cup (lfp(\gamma^2_{KB}) \cap DL_\Phi) \cup \neg.(DL_\Phi \setminus gfp(\gamma^2_{KB}))$ is satisfiable, and (b) in that case, $a \in HB_\Phi$ is well-founded (resp., unfounded) w.r.t. KB iff $a \in lfp(\gamma^2_{KB})$ (resp., $a \notin gfp(\gamma^2_{KB})$).*

The following theorem shows that for normal dl-programs, consistency under the answer set semantics implies consistency under the well-founded semantics. The converse, however, does not hold in general, unless the well-founded semantics is defined and total (i.e., two-valued) as, e.g., in the positive case. This is due to the fact that it may not always be possible to complete the partial model of the well-founded semantics to a total model.

Theorem 2 (see [37]) *Let $KB = (L, P)$ be a normal dl-program. If KB is consistent, then KB is w-consistent.*

The next theorem shows that the well-founded semantics for normal dl-programs approximates their answer set semantics. That is, every well-founded ground atom is true in every answer set, and every unfounded ground atom is false in every answer set.

Theorem 3 (see [37]) *Let $KB = (L, P)$ be a consistent normal dl-program. Then, every answer set of KB includes all atoms $a \in HB_\Phi$ that are well-founded relative to KB and no atom $a \in HB_\Phi$ that is unfounded relative to KB.*

Recall that a ground atom a is a cautious (resp., brave) consequence under the answer set semantics of a normal dl-program KB iff a is true in every (resp., some) answer set of KB.

Hence, under the answer set semantics, every well-founded and no unfounded ground atom is a cautious (resp., brave) consequence of KB.

If the well-founded semantics of a normal dl-program $KB = (L, P)$ is total, i.e., contains either a or $\neg a$ for every $a \in HB_\Phi$, then it specifies the only answer set of KB. Like in the case of ordinary normal programs, the well-founded semantics for satisfiable positive dl-programs is total and coincides with their least model semantics. This result can be proved using the characterization of the well-founded semantics given in terms of γ^2_{KB}.

5 Representing Ontology Mappings

In this section, we show how tightly integrated normal dl-programs $KB = (L, P)$ can be used for representing mappings (without confidence values) between two ontologies. Intuitively, L encodes the union of the two ontologies, while P encodes the mappings between them.

Tightly integrated normal 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 $\mathbf{V}_1$, $\mathbf{A}_1$, $\mathbf{R}_{A,1}$, $\mathbf{R}_{D,1}$, $\mathbf{I}_1$ and $\mathbf{V}_2$, $\mathbf{A}_2$, $\mathbf{R}_{A,2}$, $\mathbf{R}_{D,2}$, $\mathbf{I}_2$, respectively, such that $\mathbf{V}_1 \cap \mathbf{V}_2 = \emptyset$, $\mathbf{A}_1 \cap \mathbf{A}_2 = \emptyset$, $\mathbf{R}_{A,1} \cap \mathbf{R}_{A,2} = \emptyset$, $\mathbf{R}_{D,1} \cap \mathbf{R}_{D,2} = \emptyset$, 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 from L_1 and L_2 is then represented by a simple rule $\alpha \leftarrow \beta$ in P, where α is an atom over $\mathbf{A}_1 \cup \mathbf{R}_{A,1} \cup \mathbf{R}_{D,1}$, and β is a conjunction of NAF-literals over $\mathbf{A}_2 \cup \mathbf{R}_{A,2} \cup \mathbf{R}_{D,2}$. For example, the 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, 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 since the signatures of L_1 and L_2 are disjoint, the resulting normal dl-program $KB = (L, P)$ is actually acyclic (see Section 9).

Example 6 Taking an example from the conference data set of the OAEI challenges 2006 to 2009, 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):

$$EarlyRegisteredParticipant(X) \leftarrow Participant(X)\,;$$
$$LateRegisteredParticipant(X) \leftarrow Participant(X)\,.$$

Informally, these two mapping relationships express that every instance of the concept $Participant$ of the ontology O_1 is also an instance of the concepts $EarlyRegisteredParticipant$ and $LateRegisteredParticipant$, respectively, of the ontology O_2.

We now encode the two ontologies and the mappings by a tightly integrated normal 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 and O_2, respectively, and P encodes the mappings. But we cannot directly use the two mapping relationships as two rules in P, since this would introduce an inconsistency in KB. In detail, 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, a model of KB contains both $EarlyRegisteredParticipant(alice)$ and $LateRegisteredParticipant(alice)$. Such a model, however, is invalidated by the ontology O_2, which requires the concepts $EarlyRegisteredParticipant$ 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 [42], we present a method for detecting such inconsistent mappings. There are different approaches for resolving this inconsistency. The simplest one is to drop mapping rules until no inconsistency is present anymore. In this paper, we suggest to make use of our expressive mapping language and, in particular, to use nonmonotonic negation in order to resolve such inconsistencies. One way of doing so is to add body literals to the mapping rules, namely, negated concepts that are disjoint to the concept in the heads of the rules. This way, we add background knowledge available in the ontologies. In the example above, we then replace the two mapping rules by the following ones:

$$EarlyRegisteredParticipant(X) \leftarrow Participant(X),\ not\ LateRegisteredParticipant(X),$$
$$LateRegisteredParticipant(X) \leftarrow Participant(X),\ not\ EarlyRegisteredParticipant(X).$$

These new mapping rules resolve the inconsistency. Assuming that there is a participant Alice in O_1 ($Participant(alice)$), we then obtain the two answer sets

$$\{EarlyRegisteredParticipant(alice), Participant(alice)\},$$
$$\{LateRegisteredParticipant(alice), Participant(alice)\}.$$

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 two atoms can only be deduced bravely. More generally, with such rules, instances that are only available in the ontology O_1 cannot be classified with certainty. Similarly, the partial model of the well-founded semantics is given by $\{Participant(alice)\}$.

Another way of resolving inconsistencies in the mapping rules is to extend the bodies of the original mapping rules by additional conditions as follows:

$$EarlyRegisteredParticipant(X) \leftarrow Participant(X),\ Registerdbefore Deadline(X),$$
$$LateRegisteredParticipant(X) \leftarrow Participant(X),\ not\ Registeredbefore Deadline(X).$$

This refinement of the mapping rules resolves the inconsistency as well, and it is also easier to process because there is no cycle involving nonmonotonic negation. Note that both kinds of modifications cannot be performed fully automatically at the moment, since it is currently not possible to automatically detect and resolve all mappings introducing an inconsistency in a sound and complete way in each application domain. However, it is conceivable to perform simple modifications automatically, i.e., cases where the inconsistency can be found easily and there also is an explicit axiom in the ontology that can be used to add a modification, and to perform more intricate modifications in a semi-automatic manner.

In the next section, we present a probabilistic extension of tightly integrated normal dl-programs that allows us to directly use confidence estimations of matching engines to resolve inconsistencies and to combine the results of different matchers.

6 Tightly Integrated Probabilistic DL-Programs

In this section, we present a *tightly integrated* approach to *probabilistic description logic programs* (or simply *probabilistic dl-programs*) under the answer set and under the well-founded semantics. Similarly to the probabilistic dl-programs in [39], they are defined as a combination of dl-programs with Poole's ICL [47], but using the tightly integrated dl-programs of [37] (see Section 4), rather than the loosely integrated dl-programs of [16, 15]. Poole's ICL is based on ordinary acyclic logic programs P under different "choices", where every choice along with P produces a first-order model, and one then obtains a probability

distribution over the set of all first-order models by placing a probability distribution over the different choices. We use the tightly integrated dl-programs under the answer set semantics and under the well-founded semantics of [37], respectively, instead of ordinary acyclic logic programs under their canonical semantics (which coincides with their answer set and their well-founded semantics). We first introduce the syntax of probabilistic dl-programs and then their answer set and their well-founded semantics.

6.1 Syntax

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

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

A *tightly integrated probabilistic description logic program* (or simply *probabilistic dl-program*) $KB = (L, P, C, \mu)$ consists of a normal dl-program (L, P), a choice space C such that no atomic choice in C coincides with the head of any rule in $ground(P)$, and a probability μ on C. Intuitively, since the total choices of C select subsets of P, and μ is a probability distribution on the total choices of C, every probabilistic dl-program is the compact representation of a probability distribution on a finite set of normal dl-programs. Observe here that P is fully general and not necessarily stratified or acyclic. We say KB is *positive* iff P is positive. An *event* α is any Boolean combination of atoms (i.e., constructed from atoms via the Boolean operators "$\wedge$" and "$\neg$"). A *conditional event* is of the form $\beta|\alpha$, where α and β are events. 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 $[0,1]$.

Example 7 Consider the tightly integrated probabilistic dl-program $KB = (L, P, C, \mu)$ that is given as follows. Let L and P be defined as in Examples 1 and 3, respectively, except that the following two probabilistic rules are added to P:

$$
\begin{array}{l}
\mathit{conflict}(X,Q) \leftarrow \mathit{sameUniversity}(X,Y), \mathit{isAuthorOf}(Y,Q), \mathit{choice}_u\,; \\
\mathit{conflict}(X,Q) \leftarrow \mathit{sameTown}(X,Y), \mathit{isAuthorOf}(Y,Q), \mathit{choice}_t\,; \\
\mathit{conflict}(\mathit{mary}, p_0) \leftarrow \mathit{choice}_m\,; \\
\mathit{isAuthorOf}(\mathit{bill}, p_0);\ \mathit{sameUniversity}(\mathit{john}, \mathit{bill})\,; \\
\mathit{sameTown}(\mathit{john}, \mathit{bill});\ \mathit{sameTown}(\mathit{jane}, \mathit{bill})\,.
\end{array}
$$

Let $C = \{\{\mathit{choice}_u, \mathit{not\_choice}_u\}, \{\mathit{choice}_t, \mathit{not\_choice}_t\}, \{\mathit{choice}_m, \mathit{not\_choice}_m\}\}$, and let the probability μ on C be given by $\mu\colon \mathit{choice}_u,\ \mathit{not\_choice}_u,\ \mathit{choice}_t,\ \mathit{not\_choice}_t,\ \mathit{choice}_m,\ \mathit{not\_choice}_m \mapsto 0.8,\ 0.2,\ 0.6,\ 0.4,\ 0.7,\ 0.3$. Here, the first (resp., second) rule encodes that if a person works at the same university (resp., lives in the same town) as the

author of a paper, then this person is in a conflict situation on that paper with the probability 0.8 (resp., 0.6). The third rule encodes that Mary is in a conflict situation on paper p_0 with the probability 0.7. That is, probabilistic facts can be encoded by rules with only atomic choices in their body. A probabilistic query asking about the entailed tight interval for the probability that John is in a conflict situation on paper p_0 can be expressed by $\exists(\mathit{conflict}(\mathit{john}, p_0))[r, s]$. A probabilistic query asking about all persons along with the tight interval for the probability with which they are in a conflict situation on paper p_0 can be encoded by $\exists(\mathit{conflict}(E, p_0))[r, s]$. Another query assuming the additional condition that John lives in the same town as Bill is $\exists(\mathit{conflict}(E, p_0)|\mathit{sameTown}(\mathit{john}, \mathit{bill}))[r, s]$.

6.2 Answer Set Semantics

We now define an answer set semantics of 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. Note that their semantics is based on subjective probabilities defined on a set of possible worlds.

Given a probabilistic dl-program $KB = (L, P, C, \mu)$, a *probabilistic interpretation* Pr is a probability function on the set of all $I \subseteq HB_\Phi$. We say Pr is a *(probabilistic) answer set* of KB iff (i) every interpretation $I \subseteq HB_\Phi$ with $Pr(I) > 0$ is an answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$ for some total choice B of C, and (ii) $Pr(\bigwedge_{p \in B} p) = \sum_{I \subseteq HB_\Phi,\, B \subseteq I} Pr(I) = \mu(B)$ for every total choice B of C. Informally, Pr is an answer set of $KB = (L, P, C, \mu)$ iff (i) every interpretation $I \subseteq HB_\Phi$ of positive probability under Pr is an answer set of the dl-program (L, P) under some total choice B of C, and (ii) Pr coincides with μ on the total choices B of C. We say KB is *consistent* iff it has an answer set Pr. The latter is equivalent to $(L, P \cup \{p \leftarrow \mid p \in B\})$ being consistent for every total choice B of C with $\mu(B) > 0$, which implies that deciding whether a probabilistic dl-program is consistent can be reduced to deciding whether a normal dl-program is consistent.

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, denoted $KB \mid\!\sim (\beta|\alpha)[l, u]$, 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, denoted $KB \mid\!\sim_{\mathit{tight}} (\beta|\alpha)[l, u]$, 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 $\beta|\alpha$, we consider only the latter type of probabilistic queries in the following.

The next result shows that computing tight answers for probabilistic queries $\exists(\beta|\alpha)[r, s]$ with ground $\beta|\alpha$ to consistent probabilistic dl-programs $KB = (L, P, C, \mu)$ can be reduced to brave and cautious reasoning from normal dl-programs. Informally, the calculation is based on evaluating whether $\alpha \wedge \beta$ and $\alpha \wedge \neg\beta$ are true in every (resp., some) answer set of the normal dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every total choice B of C with $\mu(B) > 0$.

Theorem 4 (see [4]) *Let $KB = (L, P, C, \mu)$ be a consistent probabilistic dl-program, and let $Q = \exists(\beta|\alpha)[r, s]$ be a probabilistic query with ground conditional event $\beta|\alpha$. Let a (resp., b) be the sum of all $\mu(B)$ such that (i) B is a total choice of C with $\mu(B) > 0$ and (ii) $\alpha \wedge \beta$ is true in every (resp., some) answer set of the normal 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 C with $\mu(B) > 0$ and (ii) $\alpha \wedge \neg\beta$ is true in every (resp., some) answer set of the normal 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 by:*

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

Example 8 Consider again the probabilistic dl-program $KB = (L, P, C, \mu)$ of Example 7. Observe that the total choices B_i, $i \in \{1, \ldots, 8\}$, of C, which encode the possible worlds, along with their probabilities $\mu(B_i)$ are given as follows:

$$\begin{array}{ll}
B_1 = \{choice_u, choice_t, choice_m\}, & \mu(B_1) = 0.8 \cdot 0.6 \cdot 0.7 = 0.336, \\
B_2 = \{choice_u, not_choice_t, choice_m\}, & \mu(B_2) = 0.8 \cdot 0.4 \cdot 0.7 = 0.224, \\
B_3 = \{not_choice_u, choice_t, choice_m\}, & \mu(B_3) = 0.2 \cdot 0.6 \cdot 0.7 = 0.084, \\
B_4 = \{not_choice_u, not_choice_t, choice_m\}, & \mu(B_4) = 0.2 \cdot 0.4 \cdot 0.7 = 0.056, \\
B_5 = \{choice_u, choice_t, not_choice_m\}, & \mu(B_5) = 0.8 \cdot 0.6 \cdot 0.3 = 0.144, \\
B_6 = \{choice_u, not_choice_t, not_choice_m\}, & \mu(B_6) = 0.8 \cdot 0.4 \cdot 0.3 = 0.096, \\
B_7 = \{not_choice_u, choice_t, not_choice_m\}, & \mu(B_7) = 0.2 \cdot 0.6 \cdot 0.3 = 0.036, \\
B_8 = \{not_choice_u, not_choice_t, not_choice_m\}, & \mu(B_8) = 0.2 \cdot 0.4 \cdot 0.3 = 0.024\,.
\end{array}$$

It is not difficult to verify that KB has exactly one (probabilistic) answer set, i.e., every normal dl-program $(L, P \cup \{p \leftarrow \mid p \in B_i\})$, $i \in \{1, \ldots, 8\}$, has exactly one ordinary answer set. Hence, KB is in particular consistent (under the answer set semantics). Consider now the probabilistic query $Q_1 = \exists(conflict(mary, p_0))[r, s]$. Then, $a = 0.7$ (resp., $b = 0.7$), since $conflict(mary, p_0)$ is true in every (resp., some) answer set of B_i iff $i \in \{1, \ldots, 4\}$. Moreover, $c = 0.3$ (resp., $d = 0.3$), since $\neg conflict(mary, p_0)$ is true in every (resp., some) answer set of B_i iff $i \in \{5, \ldots, 8\}$. Hence, the tight answer for Q_1 to KB under the answer set semantics is given by $\theta = \{r/0.7,\ s/0.7\}$. The tight answers for the probabilistic queries $Q_2 = \exists(conflict(jane, p_0))[r, s]$, $Q_3 = \exists(conflict(john, p_0))[r, s]$, and $Q_4 = \exists(cand(john, p_0))[r, s]$ to KB under the answer set semantics are $\theta = \{r/0.6,\ s/0.6\}$ (as $conflict(jane, p_0)$ is true in the answer set of B_i iff $i \in \{1, 3, 5, 7\}$), $\theta = \{r/0.92,\ s/0.92\}$ (as $conflict(john, p_0)$ is true in the answer set of B_i iff $i \in \{1, 2, 3, 5, 6, 7\}$), and $\theta = \{r/0.08,\ s/0.08\}$ (as $cand(john, p_0)$ is true in the answer set of B_i iff $i \in \{4, 8\}$), respectively. Consider finally the query $Q_5 = \exists(conflict(mary, p_0) | sameTown(john, bill))[r, s]$ asking for the probability that there is a conflict for Mary with paper p_0 given that Jane and Bill live in the same town. Then, $a = 0.7$ (resp., $b = 0.7$), since $sameTown(john, bill) \wedge conflict(mary, p_0)$ is true in the answer set of B_i iff $i \in \{1, \ldots, 4\}$. Moreover, $c = 0.3$ (resp., $d = 0.3$), since $sameTown(john, bill) \wedge \neg conflict(mary, p_0)$ is true in the answer set of B_i iff $i \in \{5, \ldots, 8\}$. Thus, the tight answer for Q_5 to KB under the answer set semantics is $\theta = \{r/0.7,\ s/0.7\}$.

Example 9 Consider the probabilistic dl-program $KB = (L, P, C, \mu)$, where $L = \{c \sqsubseteq e\}$, $P = \{c(X) \leftarrow not\ d(X),\ a_2;\ d(X) \leftarrow not\ c(X);\ e(X) \leftarrow not\ c(X);\ e(o) \leftarrow a_1;\ f(o) \leftarrow a_1\}$, $C = \{\{a_1, a_1'\}, \{a_2, a_2'\}\}$, and $\mu(a_1) = 0.9$ and $\mu(a_2) = 0.8$. Then, the total choices of C are

given by $B_1 = \{a_1, a_2\}$, $B_2 = \{a_1, a_2'\}$, $B_3 = \{a_1', a_2\}$, and $B_4 = \{a_1', a_2'\}$. They are associated with the probabilities $\mu(B_1) = 0.9 \cdot 0.8 = 0.72$, $\mu(B_2) = 0.9 \cdot 0.2 = 0.18$, $\mu(B_3) = 0.1 \cdot 0.8 = 0.08$, and $\mu(B_4) = 0.1 \cdot 0.2 = 0.02$, and the sets of answer sets $ASS(B_1) = \{\{c(o), e(o), f(o)\}, \{d(o), e(o), f(o)\}\}$, $ASS(B_2) = \{\{d(o), e(o), f(o)\}\}$, $ASS(B_3) = \{\{c(o), e(o)\}, \{d(o), e(o)\}\}$, and $ASS(B_4) = \{\{d(o), e(o)\}\}$. This shows in particular that KB is consistent. Consider now the probabilistic query $Q = \exists(e(o))[r, s]$. Then, $a = 1$ (resp., $b = 1$), since $e(o)$ is true in each (resp., some) answer set of B_1, B_2, B_3, and B_4. Furthermore, $c = 0$ (resp., $d = 0$), since $\neg e(o)$ is false in each (resp., some) answer set of B_1, B_2, B_3, and B_4. Hence, the tight answer for Q to KB under the answer set semantics is given by $\theta = \{r/1, s/1\}$. Consider next the probabilistic query $Q' = \exists(f(o))[r, s]$. Then, $a = 0.9$ (resp., $b = 0.9$), since $f(o)$ is true in each (resp., some) answer set of B_1 and B_2. Furthermore, $c = 0.1$ (resp., $d = 0.1$), since $\neg f(o)$ is true in each (resp., some) answer set of B_3 and B_4. Hence, the tight answer for Q' to KB under the answer set semantics is given by $\theta = \{r/0.9, s/0.9\}$.

6.3 Well-Founded Semantics

In this section, we define the well-founded semantics for probabilistic dl-programs. More specifically, we define the notions of consistency and of tight answers to probabilistic queries for the well-founded semantics. We then provide several results around these notions of consistency and tight answers to probabilistic queries. In particular, we show that tight answers under the well-founded semantics approximate tight answers under the answer set semantics, and we also delineate special cases where this approximation is exact.

We first define the notion of consistency under the well-founded semantics for probabilistic dl-programs as follows. Informally, a probabilistic dl-program $KB = (L, P, C, \mu)$ is consistent under the well-founded semantics iff the normal dl-program for every total choice B of C with $\mu(B) > 0$ is consistent under the well-founded semantics. Recall that KB is consistent under the answer set semantics iff the normal dl-program for every total choice B of C with $\mu(B) > 0$ is consistent under the answer set semantics.

Definition 1 A probabilistic dl-program $KB = (L, P, C, \mu)$ is *consistent under the well-founded semantics* (or simply *w-consistent*) iff $(L, P \cup \{p \leftarrow \mid p \in B\})$ is w-consistent for every total choice B of C with $\mu(B) > 0$.

Example 10 It is not difficult to verify that the well-founded semantics of the probabilistic dl-program KB of Example 7 exists and coincides with the answer set semantics (cf. Example 8). Hence, KB is consistent under the well-founded semantics (or w-consistent).

The following theorem shows that for probabilistic dl-programs, consistency under the answer set semantics implies consistency under the well-founded semantics. The converse, however, does not hold in general, unless the well-founded semantics is defined and total (i.e., two-valued) as, e.g., in the positive case. Both results follow from similar results for normal dl-programs under the answer set and the well-founded semantics (cf. Theorem 2).

Theorem 5 *Let $KB = (L, P, C, \mu)$ be a probabilistic dl-program. If KB is consistent, then KB is w-consistent.*

Example 11 The w-consistency of the probabilistic dl-program KB of Example 7 (cf. Example 10) is immediate by its consistency under the answer set semantics (cf. Example 8).

We next define the notion of tight answers for probabilistic queries to probabilistic dl-programs under the well-founded semantics as follows. Here, we take inspiration from the characterization of tight answers for probabilistic queries to probabilistic dl-programs under the answer set semantics in Theorem 4 and the approximation of the answer set semantics by the well-founded semantics for normal dl-programs in Theorem 3. Informally, rather than evaluating whether $\alpha \wedge \beta$ and $\alpha \wedge \neg\beta$ are true in every (resp., some) answer set of the normal dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every total choice B of C with $\mu(B) > 0$, we evaluate whether $\alpha \wedge \beta$ and $\alpha \wedge \neg\beta$ are true (resp., not false) in the well-founded semantic of $(L, P \cup \{p \leftarrow \mid p \in B\})$, for every total choice B of C with $\mu(B) > 0$.

Definition 2 Let $KB = (L, P, C, \mu)$ be a w-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 C with $\mu(B) > 0$ and (ii) $\alpha \wedge \beta$ is true (resp., false) in $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$. Let c (resp., d^-) be the sum of all $\mu(B)$ such that (i) B is a total choice of C with $\mu(B) > 0$ and (ii) $\alpha \wedge \neg\beta$ is true (resp., false) in $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$. Let $b = 1 - b^-$ and $d = 1 - d^-$. Then, *the tight answer θ for Q to KB under the well-founded semantics* is defined as follows:

$$\theta = \begin{cases} \{r/1,\ s/0\} & \text{if } b = 0 \text{ and } d = 0; \\ \{r/0,\ s/0\} & \text{if } b = 0 \text{ and } d \neq 0; \\ \{r/1,\ s/1\} & \text{if } b \neq 0 \text{ and } d = 0; \\ \{r/\frac{a}{a+d},\ s/\frac{b}{b+c}\} & \text{otherwise.} \end{cases} \tag{3}$$

Example 12 Consider again the probabilistic dl-program $KB = (L, P, C, \mu)$ of Example 7. Then, the tight answers for the probabilistic queries $Q_1 = \exists(\mathit{conflict}(mary, p_0))[r, s]$, $Q_2 = \exists(\mathit{conflict}(jane, p_0))[r, s]$, $Q_3 = \exists(\mathit{conflict}(john, p_0))[r, s]$, $Q_4 = \exists(\mathit{cand}(john, p_0))[r, s]$, and $Q_5 = \exists(\mathit{conflict}(mary, p_0)|\mathit{sameTown}(john, bill))[r, s]$ to KB under the well-founded semantics are given by $\theta = \{r/0.7,\ s/0.7\}$, $\theta = \{r/0.6,\ s/0.6\}$, $\theta = \{R/0.92,\ S/0.92\}$, $\theta = \{R/0.08,\ S/0.08\}$, and $\theta = \{r/0.7,\ s/0.7\}$, respectively. For example, as for Q_1, $a = 0.7$ (resp., $b = 0.7$), since $\mathit{conflict}(mary, p_0)$ is true (resp., false) in the well-founded semantics of B_1, B_2, B_3, and B_4 (resp., B_5, B_6, B_7, and B_8). Furthermore, $c = 0.3$ (resp., $d = 0.3$), since $\neg\mathit{conflict}(mary, p_0)$ is true (resp., false) in the well-founded semantics of B_5, B_6, B_7, and B_8 (resp., B_1, B_2, B_3, and B_4). Finally, the tight answers for the two queries $Q_6 = \exists(\mathit{conflict}(mary, p_0)|\mathit{conflict}(jane, p_0))[r, s]$ and $Q_7 = \exists(\mathit{conflict}(jane, p_0)|\mathit{conflict}(mary, p_0))[r, s]$ are $\theta = \{r/0.7,\ s/0.7\}$ and $\theta = \{r/0.6,\ s/0.6\}$, respectively.

Example 13 Consider again the probabilistic dl-program $KB = (L, P, C, \mu)$ and the probabilistic queries Q and Q' of Example 9. Then, under the well-founded semantics, the total choices B_1, B_2, B_3, and B_4 of C are associated with the literal sets $WFS(B_1) = \{e(o),\ f(o)\}$, $WFS(B_2) = \{\neg c(o),\ d(o),\ e(o),\ f(o)\}$, $WFS(B_3) = \{\neg f(o)\}$, and $WFS(B_4) = \{\neg c(o),\ d(o),\ e(o),\ \neg f(o)\}$. This shows in particular that KB is w-consistent. Consider now $Q = \exists(e(o))[r, s]$. Then, $a = 0.92$ (resp., $b = 1$), since $e(o)$ is true (resp., false) in the well-founded semantics of B_1, B_2, and B_4 (resp., no total choice). Furthermore, $c = 0$ (resp., $d = 0.08$), since $\neg e(o)$ is true (resp., false) in the well-founded semantics of no total choice (resp., B_1, B_2, and B_4). Hence, the tight answer for Q to KB under the well-founded semantics is given by $\theta = \{r/0.92,\ s/1\}$. Consider next $Q' = \exists(f(o))[r, s]$. Then, $a = 0.9$ (resp., $b = 0.9$), since $f(o)$ is true (resp., false) in the well-founded semantics of B_1 and B_2 (resp., B_3 and B_4). Furthermore, $c = 0.1$ (resp., $d = 0.1$), since $\neg f(o)$ is true (resp., false) in the well-founded semantics of B_3 and B_4 (resp., B_1 and B_2). Hence, the tight answer for Q' to KB under the well-founded semantics is given by $\theta = \{r/0.9,\ s/0.9\}$.

The following theorem shows that, for probabilistic dl-programs $KB = (L, P, C, \mu)$ where the well-founded semantics is total, the tight answers for probabilistic queries to KB under the well-founded semantics (a) coincide with the tight answers to KB under the answer set semantics (if they exist) and (b) are either point intervals or empty. Here, the well-founded semantics of KB is *total*, i.e., $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ is total (i.e., two-valued) for every total choice B of C with $\mu(B) > 0$. Note that, in particular, the well-founded semantics of every positive probabilistic dl-program KB is total.

Theorem 6 *Let $KB = (L, P, C, \mu)$ be a consistent probabilistic dl-program, and let $Q = \exists(\beta|\alpha)[r, s]$ be a probabilistic query with ground $\beta|\alpha$. Let the well-founded semantics of KB be total, and let $\theta = \{r/l,\ s/u\}$ and $\theta' = \{r/l',\ s/u'\}$ be the tight answers for Q to KB under the well-founded semantics and under the answer set semantics, respectively. Then, (a) $[l', u'] = [l, u]$, and (b) either $l = u$, or $l = 1$ and $u = 0$.*

Example 14 Consider again the probabilistic dl-program KB of Example 7. The well-founded semantics of KB is total. Indeed, (a) it coincides with the answer set semantics (cf. Example 10), and (b) all tight answers in Example 12 are point intervals.

The next theorem has slightly weaker preconditions, but also a slightly weaker statement than Theorem 6. It says that for probabilistic queries $Q = \exists(\ell)[r, s]$ to KB, where ℓ is a ground literal that is defined for KB, the tight answers to KB under the well-founded semantics (a) coincide with the tight answers to KB under the answer set semantics (if they exist) and (b) are either point intervals or empty. Here, ℓ is *defined* for $KB = (L, P, C, \mu)$ iff either ℓ or $\neg.\ell$ belongs to $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ for every total choice B of C with $\mu(B) > 0$. In particular, ℓ is always defined when the well-founded semantics of KB is total.

Theorem 7 *Let $KB = (L, P, C, \mu)$ be a consistent probabilistic dl-program, and let $Q = \exists(\ell)[r, s]$ be a probabilistic query with ground literal ℓ that is defined for KB. Let $\theta = \{r/l,\ s/u\}$ and $\theta' = \{r/l',\ s/u'\}$ be the tight answers for Q to KB under the well-founded semantics and under the answer set semantics, respectively. Then, (a) $[l', u'] = [l, u]$, and (b) either $l = u$, or $l = 1$ and $u = 0$.*

Example 15 Consider again the probabilistic dl-program KB and the probabilistic query Q' of Example 9. Then, observe that the ground literal $f(o)$ is defined for KB (cf. Example 13). Indeed, the tight answer for $Q' = \exists(f(o))[r, s]$ to KB under both the answer set and the well-founded semantics is given by $\theta = \{r/0.9,\ s/0.9\}$ (cf. Examples 9 and 13).

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

Theorem 8 *Let $KB = (L, P, C, \mu)$ be a consistent probabilistic dl-program, and let $Q = \exists(\ell)[r, s]$ be a probabilistic query with ground literal ℓ. Let $\theta = \{r/l,\ s/u\}$ and $\theta' = \{r/l',\ s/u'\}$ be the tight answers for Q to KB under the well-founded semantics and under the answer set semantics, respectively. Then, $[l', u'] \subseteq [l, u]$.*

Example 16 Consider again the probabilistic dl-program KB and the probabilistic query Q of Example 9. Recall that the tight answer for Q to KB under the answer set semantics is given by $\theta = \{r/1,\ s/1\}$ (cf. Example 9), while the tight answer for Q to KB under the well-founded semantics is given by $\theta = \{r/0.92,\ s/1\}$ (cf. Example 13).

Note that the approximation is even exact in cases where the well-founded semantics associated with every total choice B of C with $\mu(B) > 0$ coincides with the set of all ground literals that can be bravely concluded under the answer set semantics associated with B.

7 Representing Ontology Mappings with Confidence Values

We now show how tightly integrated probabilistic dl-programs $KB = (L, P, C, \mu)$ 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, C, and μ encode the mappings between them, where confidence values can be encoded as error probabilities, and inconsistencies can also be resolved via trust probabilities (in addition to using nonmonotonic negations under the answer set and the well-founded semantics in P).

The probabilistic extension of tightly integrated normal dl-programs $KB = (L, P)$ to tightly integrated probabilistic dl-programs $KB' = (L, P, C, \mu)$ provides us with a means to explicitly represent and use the confidence values provided by matching systems.

In particular, we interpret each confidence value n as an *error probability*, stating that the probability that its mapping introduces an error is $1 - n$. Conversely, the probability that a mapping correctly describes the semantic relation between elements of the different ontologies is $1-(1-n) = n$. This means that we use the confidence value n 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 $(1 - n_1) \cdot (1 - n_2)$. This means that we can get a new probability for the correctness of the rule found by two matchers which is $1-(1-n_1)\cdot(1-n_2)$. 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 $1 - (1 - n_1) \cdot (1 - n_2) \geqslant n_1$ and $1 - (1 - n_1) \cdot (1 - n_2) \geqslant n_2$.

We construct the choice space along with its probability values for encoding the confidence values as follows. First, we enumerate all the mappings that are found by each matcher. Then, for each mapping, an alternative of the choice space is constructed, consisting of two atomic choices, which represent the probability that the mapping holds and that it does not hold, respectively. The values of the atomic choices are directly taken from the confidence values produced by the matchers by interpreting them as error probabilities, as described in the paragraph above. In this way, we obtain for each matcher m the subset of the choice space $\{\{m_i, not\_m_i\} | i \in \{1, \ldots, l\}\}$, where l denotes the number of all mappings of matcher m. Unifying all these subsets then yields the overall choice space.

In addition, we can associate a *trust probability* with each matcher m; it represents our confidence into the quality of the mappings produced by m, which can vary between matchers and certain application domains with specific features. For example, amongst others, the usage or avoidance of specific naming conventions for the entities in an ontology can result in mappings of different quality and thus different trust. This is especially useful when merging inconsistent results of different matchers, where we can weigh each matching system and its result with a trust probability. 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. Note that all trust probabilities sum up to 1.

Such trust probabilities can be assessed in advance, since they do not change as long as the set of considered matchers and the set of considered features of application domains do not change. It is conceivable to perform an evaluation like the one at the Ontology Alignment

Evaluation Initiative with a systematic benchmark series[2], to identify the areas in which a matcher produces strong or weak results, and to thus determine a trust probability for each matcher. As for their encoding in choice space and probability values, we introduce a new alternative, where each atomic choice along with its probability value corresponds to one trust probability; this new alternative is then easily added to an already existing choice space (after eventually renaming atomic choices that already occur in our choice space).

We now show how the probabilities are calculated, giving two examples from the benchmark data set of the OAEI 2008 campaign. In particular, we use the publication ontology in test 101 (O_1), the publication ontology in test 302 (O_2), and the publication ontology in test 301 (O_3). The mappings that we use have been discovered by one of the two matchers hmatch [8] and falcon AO [30] and possibly refined manually. Whenever we deal with a manually refined mapping by hmatch or falcon AO, we explicitly say so. The first example shows how we can deal with possibly inconsistent mappings in our framework.

Example 17 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 number for identifying the mapping. These additional conjuncts are atomic choices of the choice space C and link probabilities (which are specified in the probability μ on the choice space 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):

$$Book(X) \leftarrow Collection(X),\ hmatch_1\,;$$
$$Proceedings_2(X) \leftarrow Proceedings_1(X),\ hmatch_2\,.$$

We define the choice space according to the interpretation of confidence described above. The resulting choice space is $C = \{\{hmatch_i, not\_hmatch_i\} \mid i \in \{1,2\}\}$. It comes along with the probability μ on C, which assigns the corresponding confidence value n (from the matching system) to each atomic choice $hmatch_i$ and the complement $1-n$ 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 found by the falcon system:

$$InCollection(X) \leftarrow Collection(X),\ falcon_1\,;$$
$$Proceedings_2(X) \leftarrow Proceedings_1(X),\ falcon_2\,.$$

Here, the confidence encoding yields the choice space $C' = \{\{falcon_i, not\_falcon_i\} \mid i \in \{1, 2\}\}$ 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 would not be a good idea for two reasons. The first one is that we may encounter an inconsistency as shown in Section 5. For example, in this case, the ontology O_2 imposes that the concepts $InCollection$ and $Book$ are 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

[2] http://oaei.ontologymatching.org/2010/benchmarks/

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 the rule has two probabilities at the same time.

Suppose that 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:

$$Book(X) \leftarrow Collection(X),\ hmatch_1,\ sel\_hmatch_1\,;$$
$$InCollection(X) \leftarrow Collection(X),\ falcon_1,\ sel\_falcon_1\,;$$
$$Proceedings_2(X) \leftarrow Proceedings_1(X),\ hmatch_2\,;$$
$$Proceedings_2(X) \leftarrow Proceedings_1(X),\ falcon_2\,.$$

The new choice space C'' and the new probability μ'' on C'' are obtained from $C \cup C'$ and $\mu \cdot \mu'$ (which is the product of μ and μ', i.e., $(\mu \cdot \mu')(B \cup B') = \mu(B) \cdot \mu'(B')$ for all total choices B of C and B' of C'), respectively, by adding the alternative $\{sel\_\ hmatch_1, sel\_falcon_1\}$ and the two 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 $Proceedings_2(X) \leftarrow Proceedings_1(X)$ 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 under the answer set semantics 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 under the answer set semantics is $\theta = \{r/0.341, s/0.341\}$ (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 and 0.341, respectively. Note that we may obtain proper intervals when there are total choices with multiple answer sets.

Since the above probabilistic dl-program is positive, by Theorem 7, its well-founded semantics coincides with the answer set semantics. Hence, the well-founded semantics yields the same tight answer for the probabilistic query $Q = \exists(Book(pub))[r, s]$ as above.

In the next example, we show how two ontologies can be mapped to a third one. The resulting probabilistic dl-program has the acyclicity property, which allows for first-order rewritability and thus for LOGSPACE data complexity of deciding consistency and tight query processing under the well-founded semantics (see Section 9).

Example 18 The publication ontology in test 101 (O_1) and the publication ontology in test 301 (O_2) are mapped to the publication ontology in test 302 (O_3). Below we show some mappings that have been detected by the matching systems hmatch and falcon AO. Again, the mappings are described as rules in P, which contain an atomic choice of the choice space C as a conjunct in their bodies. Each alternative of the choice space indicates the

matching system that has created the mapping. The function μ encodes the probabilities of the mappings. Note that the common concept $Publication$ of the ontologies O_1, O_2, and O_3 is renamed to the concepts $Publication_1$, $Publication_2$, and $Publication_3$, respectively):

$$Published(X) \leftarrow Publication_1(X),\ not\, Unpublished(X),\ hmatch_1\ ;$$
$$Published(X) \leftarrow Publication_2(X),\ not\, Unpublished(X),\ hmatch_2\ ;$$
$$Publication_3(X) \leftarrow Published(X),\ falcon_1\ ;$$
$$Publication_3(X) \leftarrow Unpublished(X),\ falcon_2\ .$$

The part of the choice space mentioned in the rules above is $C = \{\{hmatch_1, not\_hmatch_1\}, \{hmatch_2,\ not\_hmatch_2\},\ \{falcon_1,\ not\_falcon_1\},\ \{falcon_2,\ not\_falcon_2\}\}$. It comes along with the function μ on C, which assigns the corresponding confidence value n (from the matching system) to each atomic choice $hmatch_i$ and the complement $1-n$ to the atomic choice not_hmatch_i. In this example, we have $\mu(hmatch_1) = 0.72$, $\mu(hmatch_2) = 0.71$, $\mu(falcon_1) = 0.85$, and $\mu(falcon_2) = 0.92$.

Suppose that $book$ is an instance of the concept $Publication_1$ in ontology O_1. Then, the probabilistic query $\exists(Publication_3(book)|Published(book))[r, s]$ yields the tight answer $\theta = \{r/0.85,\ s/0.85\}$, under both the answer set and the well-founded semantics.

8 Algorithms and Complexity

In this section, we show how the consistency and the tight query processing problem in probabilistic dl-programs under the well-founded semantics can be solved by an anytime algorithm on top of procedures for solving the consistency and the query processing problem in normal dl-programs under the well-founded semantics. More concretely, we first describe finite fixpoint iterations for solving these two problems in normal dl-programs. Based on this, we then describe the anytime algorithm for the two problems in probabilistic dl-programs, and we also characterize their data and combined complexity.

8.1 Fixpoint Iteration

The well-founded semantics of normal dl-programs KB can be computed by two finite fixpoint iterations, via the operator $\gamma^2_{KB} = \gamma_{KB} \circ \gamma_{KB}$. The computation of the operator γ_{KB} can in turn be done by one finite fixpoint iteration for computing the least model of a positive dl-program KB', via its immediate consequence operator $T_{KB'}$.

In detail, to compute the well-founded semantics of a normal dl-program $KB = (L, P)$, i.e., by Theorem 1, $WFS(KB) = lfp(\gamma^2_{KB}) \cup \neg.(HB_\Phi \setminus gfp(\gamma^2_{KB}))$, if it exists, we compute the least and the greatest fixpoint of γ^2_{KB} as the limits of the two fixpoint iterations

$$lfp(\gamma^2_{KB}) = U_\infty = \bigcup_{i \geqslant 0} U_i,\ \text{where } U_0 = \emptyset,\ \text{and } U_{i+1} = \gamma^2_{KB}(U_i),\ \text{for } i \geqslant 0,\ \text{and}$$
$$gfp(\gamma^2_{KB}) = O_\infty = \bigcap_{i \geqslant 0} O_i, \text{where } O_0 = HB_\Phi,\ \text{and } O_{i+1} = \gamma^2_{KB}(O_i),\ \text{for } i \geqslant 0,$$

respectively, which are both reached within $|HB_\Phi|$ many steps, where γ^2_{KB} stands for applying γ_{KB} twice. Note that the U_i's are an increasing sequence, while the O_i's are a decreasing sequence, i.e., $U_0 \subseteq U_1 \subseteq \cdots \subseteq U_\infty$ and $O_0 \supseteq O_1 \supseteq \cdots \supseteq O_\infty$, respectively.

As for the computation of γ_{KB} (and thus of γ^2_{KB}), recall that the application of γ_{KB} on $I \subseteq HB_\Phi$, denoted $\gamma_{KB}(I)$, is the least model of the positive dl-program $KB^{I,+} = (L^+, P^I)$, where L^+ is obtained from L by removing all constraints (see Section 4.3). The least model

of $KB^{I,+}$ in turn coincides with the least fixpoint of the immediate consequence operator $T_{KB^{I,+}}$, denoted $lfp(T_{KB^{I,+}})$, where $T_{KB^{I,+}}$ is defined as follows for every $I \subseteq HB_{\Phi}$:

$$T_{KB^{I,+}}(I) = \{H(r) \mid r \in ground(P^I),\ I \models b \text{ for all } b \in B(r)\} \cup \{a \in DL_{\Phi} \mid L^+ \cup (I \cap DL_{\Phi}) \models a\}\,.$$

Hence, to compute $\gamma_{KB}(I)$, for all $I \subseteq HB_{\Phi}$, we thus compute the least fixpoint of $T_{KB^{I,+}}$ as the limit of the fixpoint iteration

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

which is also reached within $|HB_{\Phi}|$ many steps. Note here that the S_i's are an increasing sequence, i.e., $S_0 \subseteq S_1 \subseteq \cdots \subseteq S_{\infty}$.

Finally, to assure that $WFS(KB)$ actually exists, by Theorem 1, it remains to verify that $L \cup (lfp(\gamma_{KB}^2) \cap DL_{\Phi}) \cup \neg.(DL_{\Phi} \setminus gfp(\gamma_{KB}^2))$ is satisfiable.

8.2 Anytime Algorithm

We now describe how the consistency problem for probabilistic dl-programs under the well-founded semantics can be solved. Furthermore, we provide an anytime algorithm for tight query processing in probabilistic dl-programs under the well-founded semantics.

By Definition 1, a probabilistic dl-program $KB = (L, P, C, \mu)$ is w-consistent iff every normal dl-programs $(L, P \cup \{p \leftarrow \mid p \in B\})$ such that B is a total choice of C with $\mu(B) > 0$ is w-consistent. The latter can be decided by trying to compute the well-founded semantics of the normal dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ as described in Section 8.1.

By Definition 2, computing the tight answer for a probabilistic query to a w-consistent probabilistic dl-program $KB = (L, P, C, \mu)$ under the well-founded semantics can be reduced to computing the well-founded semantics of all normal dl-programs $(L, P \cup \{p \leftarrow \mid p \in B\})$ such that B is a total choice of C with $\mu(B) > 0$. Here, the number of all total choices B is generally non-neglectable. We thus propose (i) to compute the tight answer only up to an error within a given threshold $\epsilon \in [0, 1]$, (ii) to process the B's along decreasing probabilities $\mu(B) > 0$, and (iii) to eventually stop the computation after a given time interval.

Given a w-consistent probabilistic dl-program $KB = (L, P, C, \mu)$, a probabilistic query $Q = \exists(\beta|\alpha)[r, s]$ with ground conditional event $\beta|\alpha$, 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 well-founded 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 Definition 2). It then computes the well-founded semantics S of the normal dl-program $(L, P \cup \{p \leftarrow \mid p \in B_i\})$ for every total choice B_i of C with $\mu(B_i) > 0$, checks whether $\alpha \wedge \beta$ and $\alpha \wedge \neg\beta$ are true or false in S, and updates a, b, c, and d accordingly. If the possible error in the bounds falls below ϵ, then it stops and returns the bounds computed thus far. Hence, in the special case where $\epsilon = 0$, the algorithm computes in particular the tight answer for Q to KB under the well-founded semantics. The algorithm is based on two finite fixpoint iterations for computing the well-founded semantics of normal dl-programs, which are in turn based on a finite fixpoint iteration for computing the least model of positive dl-programs, as described in Section 8.1. The following theorem shows that Algorithm tight_answer is sound.

Algorithm tight_answer

Input: w-consistent probabilistic dl-program $KB = (L, P, C, \mu)$, 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 well-founded semantics.

Notation: $B_1, \ldots, B_k$ is a sequence of all total choices B of C with $\mu(B_1) \geqslant \cdots \geqslant \mu(B_k) > 0$.

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

Fig. 1 Algorithm tight_answer.

Theorem 9 *Let $KB = (L, P, C, \mu)$ be a w-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 well-founded semantics. If tight_answer returns $\theta' = \{r/l',\ s/u'\}$ for the error threshold $\epsilon \in [0, 1]$, then $|l - l'| + |u - u'| \leqslant \epsilon$.*

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 possible error is decreasing along the iterations of the while-loop). For this reason, Algorithm tight_answer also iterates through the total choices B_i of C in a way such that the probabilities $\mu(B_i)$ are decreasing, so that the error in the computed bounds is very likely to be low already after few iteration steps.

Theorem 10 *Let $KB = (L, P, C, \mu)$ be a w-consistent probabilistic dl-program, let $Q = \exists(\beta|\alpha)[r, s]$ be a probabilistic query with ground $\beta|\alpha$, 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 well-founded semantics. Suppose we run tight_answer on KB, Q, and ϵ, and interrupt it after line (9). Let the returned $\theta' = \{r/l',\ s/u'\}$ be as in lines (11) to (14). Then, if $v = 0$, then $\theta = \theta'$. Otherwise,*

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

8.3 Complexity

The following theorem shows that both deciding w-consistency and correct query processing in probabilistic dl-programs $KB = (L, P, C, \mu)$ under the well-founded semantics is complete for P in the data complexity. Recall that the complexity class P contains all decision problems that can be solved in polynomial time on a deterministic Turing machine, and

that the data complexity for probabilistic dl-programs $KB = (L, P, C, \mu)$ describes the case where all of KB but the facts in P and the concept, role, and attribute membership axioms in L are fixed. Hardness for P follows from the hardness for P of deciding, given an ordinary positive program P and a ground atom a, whether P logically entails a in the data complexity [11]. Membership in P follows from Theorem 9 and that (a) computing the well-founded semantics of ordinary normal programs can be done in polynomial time in the data complexity, and (b) instance checking and knowledge base satisfiability in *DL-Lite*$_{\mathcal{A}}$ can be done in polynomial time. Here, $|C|$ is bounded by a constant, since C and μ define the probabilistic information of P, which is fixed as a part of the program in P, while the ordinary facts in P are the variable input. Observe that, by a similar line of argumentation, tight query processing in probabilistic dl-programs under the well-founded semantics can also be done in polynomial time in the data complexity. Note that this data tractability result for deciding w-consistency and tight query processing nicely generalizes the data tractability result presented in the conference version of this paper.

Theorem 11 *(a) Given a vocabulary Φ and a probabilistic dl-program $KB = (L, P, C, \mu)$, deciding whether KB is w-consistent is* P*-complete in the data complexity. (b) Given additionally a probabilistic query $Q = \exists(\ell)[r, s]$ with ground literal ℓ and reals $l, u \in [0, 1]$, deciding whether $\theta = \{r/l,\ s/u\}$ is a correct answer for Q to KB under the well-founded semantics is* P*-complete in the data complexity.*

The next theorem shows that both deciding w-consistency and correct query processing in probabilistic dl-programs $KB = (L, P, C, \mu)$ under the well-founded semantics is complete for EXP in general. The bounds follow from a similar argumentation as in the case of data complexity, except that now, deciding, given an ordinary positive program P and a ground atom a, whether P logically entails a is hard for EXP in general [11], and computing the well-founded semantics of ordinary normal programs is in EXP in general. Notice that, by a similar line of argumentation, tight query processing in probabilistic dl-programs under the well-founded semantics can also be done in exponential time in general.

Theorem 12 *(a) Given a vocabulary Φ and a probabilistic dl-program $KB = (L, P, C, \mu)$, deciding whether KB is w-consistent is* EXP*-complete. (b) Given also a probabilistic query $Q = \exists(\beta|\alpha)[r, s]$ with ground $\beta|\alpha$ and reals $l, u \in [0, 1]$, deciding whether $\theta = \{r/l,\ s/u\}$ is a correct answer for Q to KB under the well-founded semantics is* EXP*-complete.*

Thus, deciding w-consistency and correct query processing in probabilistic dl-programs under the well-founded semantics have a lower complexity than their counterparts under the answer set semantics, which are hard for NP and co-NP in the data and for NEXP and co-NEXP in general, as they generalize deciding consistency and brave consequences of ground atoms under the answer set semantics in ordinary normal programs, which are complete for NP and co-NP in the data and for NEXP and co-NEXP in general [11].

9 First-Order Rewritability

We now show that deciding consistency and (correct and) tight query processing in probabilistic dl-programs $KB = (L, P, C, \mu)$ under the well-founded semantics is even first-order rewritable, and thus can be done in LOGSPACE in the data complexity, when we make an additional acyclicity assumption. Hence, deciding consistency and (correct and) tight query processing from such KB under the well-founded semantics can be done very efficiently by

means of commercial, SQL-expressive relational database systems. In the same time, normal and probabilistic dl-programs under the additional acyclicity assumption are still expressive enough to represent ontology mappings as described in Sections 5 and 7, respectively.

We first formalize the notion of first-order rewritability for the consistency and the tight query processing problem in probabilistic dl-programs under the well-founded semantics. The w-consistency problem in probabilistic dl-programs $KB = (L, P, C, \mu)$ is *first-order rewritable* iff it can be expressed in terms of a first-order formula ϕ over the set F of all concept, role, and attribute membership axioms in L and all facts in P, i.e., KB is w-consistent iff $I_F \models \phi$, where I_F is the model satisfying exactly F. A probabilistic query $Q = \exists(\beta|\alpha)[r, s]$ with ground $\beta|\alpha$ to a w-consistent probabilistic dl-program $KB = (L, P, C, \mu)$ is *first-order rewritable* iff the tight answer $\theta = \{r/l,\ s/u\}$ for Q to KB under the well-founded semantics can be expressed in terms of a first-order formula $\phi(l, u)$ over the set F of all concept, role, and attribute membership axioms in L and all facts in P, i.e., $\theta = \{r/l,\ s/u\}$ is the tight answer for Q to KB under the well-founded semantics iff $I_F \models \phi(l, u)$.

We next define the notion of acyclicity for ordinary normal programs, normal dl-programs, and probabilistic dl-programs as follows. Given a normal program P, we denote by $\mathcal{P}_P$ the set of all predicate symbols in P. We say P is *acyclic* iff a mapping $\kappa\colon \mathcal{P}_P \to \{0, 1, \ldots, n\}$ exists such that for every $r \in P$, the predicate symbol p of $H(r)$, and every predicate symbol q of some $b \in B(r)$, it holds that $\kappa(p) > \kappa(q)$. A normal dl-program $KB = (L, P)$ is *acyclic* iff (i) P is acyclic, and (ii) L can be partitioned into description logic knowledge bases $L_1^I, \ldots, L_m^I, L^O$ over pairwise disjoint sets of atomic concepts, atomic roles, and attributes such that the atomic concepts, atomic roles, and attributes of $L_1^I, \ldots, L_m^I$ only occur in bodies of rules in P and the ones of L^O only occur in heads of rules in P. A probabilistic dl-program $KB = (L, P, C, \mu)$ is *acyclic* iff every normal dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$ is acyclic for every total choice B of C with $\mu(B) > 0$. Intuitively, acyclic probabilistic dl-programs $KB = (L, P, C, \mu)$ allow for reading out instances of concepts, roles, and attributes from several input ontologies $L_1^I, \ldots, L_m^I$, elaborating them in an acyclic normal program P, and then merging the result into an output ontology L^O.

The following theorem shows that both deciding w-consistency and tight query processing in acyclic probabilistic dl-programs $KB = (L, P)$ under the well-founded semantics are first-order rewritable (and thus can be done in LOGSPACE in the data complexity).

Theorem 13 *(a) Given an alphabet Φ and an acyclic probabilistic dl-program $KB = (L, P, C, \mu)$, deciding whether KB is w-consistent is first-order rewritable. (b) Given additionally a probabilistic query $Q = \exists(\beta|\alpha)[r, s]$ with ground $\beta|\alpha$, computing the tight answer for Q to KB under the well-founded semantics is first-order rewritable.*

10 Related Work

In this section, we give a comparison to most closely related approaches to (i) combinations of rules and ontologies under the well-founded semantics, (ii) probabilistic description logic programs, (iii) other probabilistic languages for representing ontology mappings, and (iv) approaches to representing complex ontology mappings.

10.1 Combinations of Rules and Ontologies under the Well-Founded Semantics

To our knowledge, there are no previous approaches to probabilistic generalizations of logic programs or of combinations of rules and ontologies that use the well-founded semantics to

interpret non-monotonic negations in rule bodies. However, there are several ordinary combinations of rules and ontologies (for the Semantic Web), for which a well-founded semantics has been defined; more specifically, the works [15], [34], and [14] define a well-founded semantics for the loosely integrated dl-programs in [16,15], for the hybrid MKNF knowledge bases in [43,44], and for an integration of rules and ontologies that is close in spirit to Rosati's approach [51,52], respectively. As for the more general use of the well-founded semantics in the context of the Web, several reasoners adopt it for handling nonmonotonic negation, including $\mathcal{F}$lora-2[3] (which builds on XSB[4]) and OntoBroker[5], which are based on F-Logic [33], and IRIS and MINS,[6] towards the WSML-Rule language [12].

10.2 Probabilistic Description Logic Programs

It is important to point out that the probabilistic description logic programs here are very different from the ones in [39] (and their recent tractable variant in [40]) as well as from the ones in [49]. First, they are based on the tight integration between the ontology component L and the rule component P of [37], while the ones in [39,40] realize the loose query-based integration between the ontology component L and the rule component P of [16,15]. This implies in particular that the vocabularies of L and P here may have common elements (see also Example 3), while the vocabularies of L and P in [39,40] are necessarily disjoint. Furthermore, the probabilistic description logic programs here behave semantically very differently from the ones in [39,40]. 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), while the ones in [39,40] can especially be used as query interfaces to Web databases (including RDF theories). Second, differently from here, the works [39,40] do not explore the aspect of first-order rewritability. Third, the works [39, 40] also do not explore the use of probabilistic description logic programs for representing and reasoning with ontology mappings under probabilistic uncertainty and inconsistency. Furthermore, the programs in [49] are much less expressive in general.

10.3 Probabilistic Languages for Representing Ontology Mappings

There are several languages for representing mappings between ontologies [55,49]. However, all of them, except for Bayesian description logic programs (BDLPs) [49] represent mappings deterministically. BDLPs differ from the probabilistic dl-programs here in the expressibility of both the description logic component L and the logic programming component P. In BDLPs, differently from here, L is formulated in the description logic programming (DLP) fragment from Grosof et al. [27], and P does not support any kind of negation, which is strictly less expressive than here. Furthermore, differently from here, the semantics of BDLPs is based on a translation of L into logic programs.

There are other probabilistic extensions of different Web languages that are conceivable to be used as mapping languages in the context of ontology mapping [48]. Examples of such probabilistic extensions are probabilistic extensions of description logics like P-CLASSIC,

[3] http://flora.sourceforge.net/

[4] http://xsb.sourceforge.net/

[5] http://www.ontoprise.de/en/home/products/ontobroker/

[6] http://iris-reasoner.org/, http://tools.sti-innsbruck.at/mins/

which is a probabilistic extension of the CLASSIC description logic [35], PR-OWL, which is an ontology that describes multi-entity Bayesian networks [10], BayesOWL, which provides a probabilistic extension of a subset of OWL [13], and P-$\mathcal{SHOQ}(\mathbf{D})$, which is a probabilistic extension of $\mathcal{SHOQ}(\mathbf{D})$ [26] (see also P-$\mathcal{SHIF}(\mathbf{D})$ and P-$\mathcal{SHOIN}(\mathbf{D})$ in [38]). P-CLASSIC and BayesOWL have the disadvantage of a too low expressivity. PR-OWL does not provide a tight formal integration between ontologies and the probabilistic model that they describe. Although P-$\mathcal{SHOQ}$ is quite expressive and provides a tight integration between the description logic and the probabilistic model, it does not have a rules component and cannot solve the instance retrieval reasoning task as efficiently as a rule language. Probabilistic extensions of rule languages for the Web besides the already mentioned BDLPS are also pOWL Lite$^-$ and pOWL Lite^{EQ} [45]. These two languages differ only by equality, which is disallowed in pOWL Lite$^-$. Both support also only the description logic programming fragment (possibly enriched with equality) that is supported by BDLPs and thus have the same expressivity drawback. Note that except for BDLPs, none of these languages have been considered for an application in the area of ontology mappings.

10.4 Representing Complex Ontology Mappings

While almost all existing matching systems are restricted to the generation of simple equivalence correspondences between classes and relations in ontologies, there is an increasing interest in the identification and representation of complex correspondences between ontologies [50, 54]. Along this line of research, a number of common alignment patterns have been defined by Scharffe and others [53]; they capture typical situations where a complex relation exists between elements from two ontologies. It is easy to see that many of the patterns proposed in [53] can directly or indirectly be represented using our formalism. Here are two examples of the more complex patterns (note that the examples are slightly modified to enhance readability, without changing the nature of the patterns):

Class Relation Mapping: The pattern links a class expression to a relation in the target ontology. The example given in [53] can be represented in tightly integrated description logic programs via the following rule:

$$marriage(X_1, X_2, D) \leftarrow Marriage(X),\ partner1(X, X_1),\\ partner2(X, X_2),\ date(X, D)\,.$$

Attribute Value Mapping: The pattern relates two attribute values in different ontologies. The example given in [53] can be represented in tightly integrated description logic programs via the following rule:

$$country(A, Ireland) \leftarrow Address(A),\ countryCode(A, \text{'IE'})\,.$$

These examples show that the proposed formalism is a good candidate for providing a formal underpinning of complex ontology mapping patterns. A more detailed and complete investigation of the formalization of proposed patterns is subject to future work.

11 Conclusion

In this paper, we have explored the use of tightly integrated probabilistic dl-programs as a rule-based framework for representing and reasoning with ontology mappings. We have

newly presented the well-founded semantics for such programs, and analyzed its semantic and computational properties. In particular, we have shown that the well-founded semantics approximates the answer set semantics. Furthermore, we have presented an anytime algorithm for tight query processing in such programs, based on fixpoint iterations for computing the well-founded semantics of normal dl-programs, and we have analyzed the data (resp., general) complexity of consistency checking and correct query processing for tightly integrated probabilistic dl-programs, which are both complete for P (resp., EXP).

As for their use in representing ontology mappings, tightly integrated probabilistic dl-programs have the following useful features:

- The semantics of the language is based on the tight integration between ontology and rule languages of [37], which assumes no structural separation between the vocabularies of the description logic and the logic program components. This enables us to have description logic concepts and roles in both rule bodies and rule heads. This is necessary if we want to use rules to combine ontologies.
- The rule language is quite expressive. In particular, we can have nonmonotonic negations in rule bodies. This gives a rich basis for refining and rewriting automatically created mappings for resolving inconsistencies.
- The integration with probability theory provides us with a sound formal framework for representing and reasoning 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 by using trust probabilities.
- Consistency checking and tight query processing under the well-founded semantics are tractable in the data complexity. In a special case, which is especially interesting for representing and reasoning with ontology mappings, these problems are even first-order rewritable (and thus can be done in LOGSPACE in the data complexity).

Hence, compared to the answer set semantics of tightly integrated probabilistic dl-programs, (its semantic approximation by) the well-founded semantics can be calculated via fixpoint iterations, and both consistency checking and query processing have a lower (data and general) complexity, as well as a practically useful first-order rewritable special case.

Note that the results of this paper are not restricted to *DL-Lite*$_\mathcal{A}$ as underlying ontology language; they also hold when any other tractable ontology language from the *DL-Lite* family [6] is used instead. Most of the results (except for the first-order rewritability ones) also carry over to more expressive tractable ontology languages, such as Horn-$\mathcal{SHIQ}$ [31].

We leave for future work the implementation of tightly integrated probabilistic dl-programs. Another interesting topic for future work is to explore whether the first-order rewritability results can be extended to an even larger class of tightly integrated probabilistic dl-programs. Furthermore, it would be interesting to investigate whether one can develop an efficient top-k query technique (as, e.g., in [56,41]) for tightly integrated probabilistic dl-programs: Rather than computing the tight probability interval for a given ground atom, such a technique returns the k most probable ground instances of a given non-ground atom.

Acknowledgments. Thomas Lukasiewicz has been supported by the German Research Foundation (DFG) under the Heisenberg Programme, by a Yahoo! Research Fellowship, by the Austrian Science Fund (FWF) under the project P18146-N04, by the EPSRC under the grant EP/E010865/1 "Schema Mappings and Automated Services for Data Integration", and by the European Research Council under the EU's 7th Framework Programme (FP7/2007-2013)/ERC grant 246858 – DIADEM. Heiner Stuckenschmidt and Livia Predoiu have been supported by an Emmy-Noether Grant of the German Research Founda-

tion (DFG). We thank the reviewers of this paper and its URSW-2007 and FoIKS-2008 abstracts for their useful and constructive comments, which have helped to improve this work.

Appendix A: Proofs

Proof of Theorem 5. Recall that $KB = (L, P, C, \mu)$ is consistent under the answer set semantics iff $(L, P \cup \{p \leftarrow \mid p \in B\})$ has an answer set for every total choice B of C with $\mu(B) > 0$. By Theorem 2, the latter implies that $(L, P \cup \{p \leftarrow \mid p \in B\})$ is consistent under the well-founded semantics for every total choice B of C with $\mu(B) > 0$, which is in turn equivalent to $KB = (L, P, C, \mu)$ being consistent under the well-founded semantics. □

Proof of Theorem 6. (a) By Theorem 3, for every total choice B of C with $\mu(B) > 0$, the answer set semantics of $(L, P \cup \{p \leftarrow \mid p \in B\})$ consists of a single answer set, and it coincides with the well-founded semantics of $(L, P \cup \{p \leftarrow \mid p \in B\})$. This implies that a, b, c, and d in Theorem 4 coincide with a, b, c, and d in Definition 2, respectively.

(b) Since $S_B = WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ is total for every total choice B of C with $\mu(B) > 0$, it follows that (1) $\alpha \wedge \beta$ is true in S_B iff $\alpha \wedge \beta$ is not false in S_B, and (2) $\alpha \wedge \neg\beta$ is true in S_B iff $\alpha \wedge \neg\beta$ is not false in S_B, for every total choice B of C with $\mu(B) > 0$. Hence, $b^- = 1 - a$ and $d^- = 1 - c$, and thus $b = a$ and $c = d$, respectively. It thus follows that either $l = 1$ and $u = 0$, or $l = u$. □

Proof of Theorem 7. (a) We follow the same line of argumentation as in the proof of Theorem 8. But since ℓ is defined for $KB = (L, P, C, \mu)$, either ℓ or $\neg.\ell$ belongs to $S_B = WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ for every total choice B of C with $\mu(B) > 0$, and thus we obtain $a = a_s$, $b_s = b$, $c = c_s$, and $d_s = d$ rather than only $a \leqslant a_s$, $b_s \leqslant b$, $c \leqslant c_s$, and $d_s \leqslant d$, respectively, which implies the stronger result $[l', u'] = [l, u]$ rather than only $[l', u'] \subseteq [l, u]$.

(b) Since ℓ is defined for $KB = (L, P, C, \mu)$, either ℓ or $\neg.\ell$ belongs to $S_B = WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ for every total choice B of C with $\mu(B) > 0$. It thus follows that (a) ℓ is true in S_B iff ℓ is not false in S_B, and (b) $\neg\ell$ is true in S_B iff $\neg\ell$ is not false in S_B, for every total choice B of C with $\mu(B) > 0$. Hence, $b^- = 1 - a$ and $d^- = 1 - c$, and thus $b = a$ and $c = d$, respectively. Hence, $l = 1$ and $u = 0$, or $l = u$. □

Proof of Theorem 8. Recall that, by Definition 2, the tight answer $\theta = \{r/l,\ s/u\}$ for Q to KB under the well-founded semantics is given as follows:

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

where (a) a (resp., b^-) is the sum of all $\mu(B)$ such that (a.i) B is a total choice of C with $\mu(B) > 0$ and (a.ii) ℓ is true (resp., false) in $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$, (b) c (resp., d^-) is the sum of all $\mu(B)$ such that (b.i) B is a total choice of C with $\mu(B) > 0$ and (b.ii) $\neg.\ell$ is true (resp., false) in $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$, and (c) $b = 1 - b^-$ and $d = 1 - d^-$. Similarly, by Theorem 4, the tight answer $\theta' = \{r/l',\ s/u'\}$ for Q to KB under the answer set semantics

is given as follows:

$$\theta' \;=\; \begin{cases} \{r/1,\ s/0\} & \text{if } b_s = 0 \text{ and } d_s = 0; \\ \{r/0,\ s/0\} & \text{if } b_s = 0 \text{ and } d_s \neq 0; \\ \{r/1,\ s/1\} & \text{if } b_s \neq 0 \text{ and } d_s = 0; \\ \{r/\frac{a_s}{a_s+d_s},\ s/\frac{b_s}{b_s+c_s}\} & \text{otherwise,} \end{cases}$$

where (a) a_s (resp., b_s) is the sum of all $\mu(B)$ such that (a.i) B is a total choice of C with $\mu(B) > 0$ and (a.ii) ℓ is true in every (resp., some) answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$, and (b) c_s (resp., d_s) is the sum of all $\mu(B)$ such that (b.i) B is a total choice of C with $\mu(B) > 0$ and (b.ii) $\neg.\ell$ is true in every (resp., some) answer set of $(L, P \cup \{p \leftarrow \mid p \in B\})$. By Theorem 3, it thus follows that $a \leqslant a_s$, $b_s \leqslant b$, $c \leqslant c_s$, and $d_s \leqslant d$. Since clearly $a_s \leqslant b_s$ and $c_s \leqslant d_s$, we thus obtain $a \leqslant a_s \leqslant b_s \leqslant b$ and $c \leqslant c_s \leqslant d_s \leqslant d$. We now consider four cases (i)–(iv) as follows. (i) If $b = 0$ and $d = 0$, then also $b_s = 0$ and $d_s = 0$, and thus $[l', u'] = [l, u] = [1, 0]$. (ii) If $b \neq 0$ and $d = 0$, then $b_s \geqslant 0$ and $d_s = 0$, and thus $[l', u']$ is either $[1, 0]$ or $[1, 1]$, which implies that $[l', u'] \subseteq [l, u] = [1, 1]$. (iii) If $b = 0$ and $d \neq 0$, then $b_s = 0$ and $d_s \geqslant 0$, and thus $[l', u']$ is either $[1, 0]$ or $[0, 0]$, which implies that $[l', u'] \subseteq [l, u] = [0, 0]$. (iv) If $b \neq 0$ and $d \neq 0$, then $b_s \geqslant 0$ and $d_s \geqslant 0$, and we consider four subcases as follows. (iv.a) If $b_s = 0$ and $d_s = 0$, then $[l', u'] = [1, 0] \subseteq [l, u]$. (iv.b) If $b_s = 0$ and $d_s \neq 0$, then $[l', u'] = [0, 0]$, $a = 0$, and $l = 0$, which implies that $[l', u'] \subseteq [l, u]$. (iv.c) If $b_s \neq 0$ and $d_s = 0$, then $[l', u'] = [1, 1]$, $c = 0$, and $u = 1$, which implies that $[l', u'] \subseteq [l, u]$. (iv.d) If $b_s \neq 0$ and $d_s \neq 0$, then $[l', u'] = [\frac{a_s}{a_s+d_s}, \frac{b_s}{b_s+c_s}]$ and $[l, u] = [\frac{a}{a+d}, \frac{b}{b+c}]$, which also implies that $[l', u'] \subseteq [l, u]$. □

Proof of Theorem 9. Since the number of all total choices B of 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. Observe then that $a' + v', c' + v' \leqslant 1$ and $b', d' \geqslant v'$. If $v' = 0$, then the algorithm has processed all the total choices B of C with $\mu(B) > 0$. Hence, in this case, by Definition 2, the algorithm returns the exact tight answer for Q to KB. Suppose now $v' > 0$ (and thus $i' \leqslant k$, $b' \neq 0$, $d' \neq 0$, and $\frac{v'}{a'+d'} + \frac{v'}{b'+c'} \leqslant \epsilon$). Then, the returned lower and upper bounds are given by $l' = \frac{a'}{a'+d'}$ and $u' = \frac{b'}{b'+c'}$, respectively. From the algorithm, it is easy to verify that the exact tight lower and upper bounds l and u are of the form

$$l = \frac{a'+v_a}{a'+v_a+d'-v_d} \text{ and } u = \frac{b'-v_b}{b'-v_b+c'+v_c},$$

respectively, where $v_a, v_b, v_c, v_d \in [0, v']$. Observe that

$$l = (1 + \tfrac{d'-v_d}{a'+v_a})^{-1} \text{ and } u = (1 + \tfrac{c'+v_c}{b'-v_b})^{-1}, \text{ if } a' + v_a > 0 \text{ and } b' - v_b > 0,$$

respectively. This implies that $l' \leqslant l \leqslant \frac{a'+v'}{a'+d'}$ and $\frac{b'-v'}{b'+c'} \leqslant u \leqslant u'$, respectively. Hence,

$$\begin{aligned} |l - l'| + |u - u'| &= (l - l') + (u' - u) \\ &\leqslant \tfrac{a'+v'}{a'+d'} - \tfrac{a'}{a'+d'} + \tfrac{b'}{b'+c'} - \tfrac{b'-v'}{b'+c'} \\ &= \tfrac{v'}{a'+d'} + \tfrac{v'}{b'+c'} \\ &\leqslant \epsilon\,. \;\; \square \end{aligned}$$

Proof of Theorem 10. Immediate by the proof of Theorem 9. □

Proof of Theorem 11. Hardness for P in both (a) and (b) follows from the P-hardness of deciding, given an ordinary positive program P and a ground atom a, whether P logically entails a in the data complexity [11]. Membership in P in both (a) and (b) follows from Theorem 9 and the fact that the fixpoint iterations for computing $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ for each total choice B of C with $\mu(B) > 0$ can be done in polynomial time in the data complexity, since instance checking and knowledge base satisfiability in *DL-Lite*$_{\mathcal{A}}$ can be done in polynomial time. □

Proof of Theorem 12. Hardness for EXP in both (a) and (b) follows from the EXP-hardness of deciding, given an ordinary positive program P and a ground atom a, whether P logically entails a [11]. Membership in EXP in both (a) and (b) follows from Theorem 9 and the fact that the fixpoint iterations for computing $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ for each total choice B of C with $\mu(B) > 0$ can be done in exponential time, since instance checking and knowledge base satisfiability in *DL-Lite*$_{\mathcal{A}}$ can be done in polynomial time. □

Proof of Theorem 13. We first show that for each total choice B of C with $\mu(B) > 0$, deciding whether a given ground atom a belongs to $WFS(L, P \cup \{p \leftarrow \mid p \in B\})$ is first-order rewritable. Since every $(L, P \cup \{p \leftarrow \mid p \in B\})$ is acyclic, there exists a mapping $\kappa\colon \mathcal{P}_P \rightarrow \{0, 1, \ldots, n\}$ such that for every rule $r \in P$, the predicate symbol p of $H(r)$, and every predicate symbol q of some $b \in B(r)$, it holds that $\kappa(p) > \kappa(q)$. We call $\kappa(p)$ the *rank* of p. Since every L_j^I is defined in *DL-Lite*$_{\mathcal{A}}$, as shown in [46], every concept, role, and attribute membership a from L_j^I can be expressed in terms of a first-order formula over the concept, role, and attribute membership axioms in L_j^I. We first show by induction on $\kappa(p) \in \{0, 1, \ldots, n\}$ that every predicate $p \in \mathcal{P}_P$ (relative to $(L \cup P) \setminus L^O$) can also be expressed in terms of a first-order formula over the concept, role, and attribute membership axioms in $L \setminus L^O$ and the *database facts* in P, constructed from predicate symbols of rank 0.

Basis: Every predicate symbol p of rank 0 that does not occur in L, can trivially be expressed in terms of a first-order formula over the database facts in P. As stated above, by [46], every predicate symbol p of rank 0 that occurs in some L_j^I can be expressed in terms of a first-order formula over the concept, role, and attribute membership axioms in L_j^I. In summary, every predicate symbol p of rank 0 can be expressed in terms of a first-order formula over the concept, role, and attribute membership axioms in $L \setminus L^O$ and the database facts in P.

Induction: Consider any predicate symbol $p \in \mathcal{P}_P$ along with the set of all its defining rules in P, i.e., all rules in P with p in their head. W.l.o.g., the heads $p(\mathbf{x})$ of all these rules coincide. Let $\alpha(\mathbf{x})$ denote the disjunction of the existentially quantified bodies of these rules. By the induction hypothesis, every body predicate symbol in $\alpha(\mathbf{x})$ can be expressed in terms of a first-order formula over the concept, role, and attribute membership axioms in $L \setminus L^O$ and the database facts in P. Let the first-order formula $\alpha'(\mathbf{x})$ be obtained from $\alpha(\mathbf{x})$ by replacing all atoms by their first-order formulas. Then, $\alpha'(\mathbf{x})$ expresses p in terms of the concept, role, and attribute membership axioms in $L \setminus L^O$ and the database facts in P.

We next add the description logic knowledge base L^O. As stated above, by [46], every predicate symbol p that occurs in L^O can be expressed in terms of a first-order formula $\alpha(\mathbf{x})$ over the concept, role, and attribute membership axioms in L^O. As shown above, every predicate symbol $q \in \mathcal{P}_P$ (relative to $(L \cup P) \setminus L^O$) can be expressed in terms of a first-order formula ϕ over the concept, role, and attribute membership axioms in $L \setminus L^O$ and the database facts in P. Let the first-order formula $\alpha'(\mathbf{x})$ be obtained from $\alpha(\mathbf{x})$ by replacing every atom $q(\mathbf{y})$ by the formula $q(\mathbf{y}) \vee \phi(\mathbf{y})$. Then, $\alpha'(\mathbf{x})$ is a first-order formula for p over the concept, role, and attribute membership axioms in L and the database facts in P.

Consequently, every atom and thus also every ground event α (relative to $(L, P \cup \{p \leftarrow \mid p \in B\})$) can be expressed in terms of a first-order formula α_B over the concept, role, and attribute membership axioms in L and the database facts in P. Similarly, the probability of α in KB can be expressed as the first-order formula $\phi(pr)$ consisting of the disjunction of all $prob(pr) \wedge \alpha'_{B_1} \wedge \cdots \wedge \alpha'_{B_n}$, where (i) $B_1, \ldots, B_n$ are the total choices of C with $\mu(B_i) > 0$, (ii) α'_{B_i} is either α_{B_i} or $\neg\alpha_{B_i}$, and (iii) pr is the sum of all $\mu(B_i)$, $i \in \{1, \ldots, n\}$, such that $\alpha'_{B_i} = \alpha_{B_i}$. Similarly, the probability interval $[l, u]$ for a ground conditional event $\beta|\alpha$ in KB can be expressed as the first-order formula $\phi(l, u)$ consisting of the disjunction of all $prob_l(l) \wedge prob_u(u) \wedge (\alpha \wedge \beta)'_{B_1} \wedge \cdots \wedge (\alpha \wedge \beta)'_{B_n} \wedge \alpha'_{B_1} \wedge \cdots \wedge \alpha'_{B_n}$, where (i) $B_1, \ldots, B_n$ are the total choices of C with $\mu(B_i) > 0$, (ii) $(\alpha \wedge \beta)'_{B_i}$ is either $(\alpha \wedge \beta)_{B_i}$ or $\neg(\alpha \wedge \beta)_{B_i}$, (iii) α'_{B_i} is either α_{B_i} or $\neg\alpha_{B_i}$, and (iv) $l = u = v/w$, if $w > 0$, and $l = 1$ and $u = 0$, otherwise (note that the acyclicity of KB implies that the well-founded semantics of KB is total), where v is the sum of all $\mu(B_i)$, $i \in \{1, \ldots, n\}$, such that $(\alpha \wedge \beta)'_{B_i} = (\alpha \wedge \beta)_{B_i}$, and w is the sum of all $\mu(B_i)$, $i \in \{1, \ldots, n\}$, such that $\alpha'_{B_i} = \alpha_{B_i}$. In summary, $\theta = \{r/l,\ s/u\}$ is the tight answer for Q to KB under the well-founded semantics iff $F \models \phi(l, u)$, where F is the set of all concept, role, and attribute membership axioms in L and the database facts in P.

As for the w-consistency problem, by [46], the satisfiability of L^O and every L^I_j can be expressed in terms of a first-order formula over the concept, role, and attribute membership axioms in L^O and every L^I_j, respectively. The acyclicity of KB implies that we only have to check these satisfiabilities, where all facts derived under B are added to L^O, for every total choice B of C with $\mu(B) > 0$. The first-order formula for this satisfiability check is constructed as above, and then disjunctively combined with the disjunction of the first-order formulas for all L^I_j. The first-order formula for the w-consistency problem is then the disjunction of all these first-order formulas, for every total choice B of C with $\mu(B) > 0$. □

References

1. C. Baral and V. S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *J. Autom. Reasoning*, 10(3):399–420, 1993.
2. A. Calì and T. Lukasiewicz. Tightly integrated probabilistic description logic programs for the Semantic Web. In *Proc. ICLP-2007*, pp. 428–429. *LNCS* 4670, Springer, 2007.
3. A. Calì, T. Lukasiewicz, L. Predoiu, and H. Stuckenschmidt. A framework for representing ontology mappings under probabilities and inconsistency. In *Proc. URSW-2007*. *CEUR Workshop Proceedings* 327, CEUR-WS.org, 2008.
4. A. Calì, T. Lukasiewicz, L. Predoiu, and H. Stuckenschmidt. Tightly coupled probabilistic description logic programs for the Semantic Web. *J. Data Sem.*, 12:95–130, 2009.
5. A. Calì, T. Lukasiewicz, L. Predoiu, and H. Stuckenschmidt. Tightly integrated probabilistic description logic programs for representing ontology mappings. In *Proc. FoIKS-2008*, pp. 178–198. *LNCS* 4932, Springer, 2008.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
7. C. Caracciolo, J. Euzenat, L. Hollink, R. Ichise, A. Isaac, V. Malaise, C. Meilicke, J. Pane, P. Shvaiko, H. Stuckenschmidt, O. Svab-Zamazal, and V. Svatek. Results of the ontology alignment evaluation initiative 2008. In *Proc. ISWC-2008 Workshop on Ontology Matching*.
8. S. Castano, A. Ferrara, and G. Messa. ISLab HMatch Results for OAEI 2006. In *Proc. International Workshop on Ontology Matching*, 2006.
9. P. C. G. da Costa. *Bayesian Semantics for the Semantic Web*. Doctoral Dissertation, George Mason University, Fairfax, VA, USA, 2005.
10. P. C. G. da Costa and K. B. Laskey. PR-OWL: A framework for probabilistic ontologies. In *Proc. FOIS-2006*, pp. 237–249. IOS Press, 2006.
11. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.
12. J. de Bruijn, H. Lausen, A. Polleres, and D. Fensel. The Web service modeling language WSML: An overview. In *Proc. ESWC-2006*, pp. 590–604. *LNCS* 4011, Springer, 2006.

13. Z. Ding, Y. Peng, and R. Pan. BayesOWL: Uncertainty modeling in Semantic Web ontologies. In *Soft Computing in Ontologies and Semantic Web*, pp. 3–28. Springer, 2006.
14. W. Drabent and J. Małuszyński. Well-founded semantics for hybrid rules. In *Proc. RR-2007*, pp. 1–15. *LNCS* 4524, Springer, 2007.
15. T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer. Well-founded semantics for description logic programs in the Semantic Web. *ACM Trans. Comput. Log.*, 12(2), Article 11, 2011.
16. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artif. Intell.*, 172(12/13):1495–1539, 2008.
17. T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *Proc. ESWC-2006*, pp. 273–287. *LNCS* 4011, Springer, 2006.
18. J. Euzenat, A. Ferrara, L. Hollink, A. Isaac, C. Joslyn, V. Malaise, C. Meilicke, A. Nikolov, J. Pane, M. Sabou, F. Scharffe, P. Shvaiko, V. Spiliopoulos, H. Stuckenschmidt, O. Svab-Zamazal, V. Svatek, C. Trojahn dos Santos, G. Vouros, and S. Wang. Results of the ontology alignment evaluation initiative 2009. In *Proc. ISWC-2009 Workshop on Ontology Matching*.
19. J. Euzenat, A. Isaac, C. Meilicke, P. Shvaiko, H. Stuckenschmidt, O. Svab, V. Svatek, W. R. van Hage, and M. Yatskevich. Results of the ontology alignment evaluation initiative 2007. In *Proc. ISWC-2007 Workshop on Ontology Matching*.
20. 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 *Proc. ISWC-2006 Workshop on Ontology Matching*.
21. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, Heidelberg, Germany, 2007.
22. J. Euzenat, H. Stuckenschmidt, and M. Yatskevich. Introduction to the ontology alignment evaluation 2005. In *Proc. K-CAP-2005 Workshop on Integrating Ontologies*.
23. W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proc. JELIA-2004*, pp. 200–212. *LNCS* 3229, Springer, 2004.
24. A. Finzi and T. Lukasiewicz. Structure-based causes and explanations in the independent choice logic. In *Proc. UAI-2003*, pp. 225–232. Morgan Kaufmann, 2003.
25. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generat. Comput.*, 9(3/4):365–386, 1991.
26. 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 *Proc. JELIA-2002*, pp. 86–97. *LNCS* 2424, Springer, 2002.
27. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logics. In *Proc. WWW-2003*, pp. 48–57. ACM Press, 2003.
28. I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proc. ISWC-2003*, pp. 17–29. *LNCS* 2870, Springer, 2003.
29. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. LPAR-1999*, pp. 161–180. *LNCS* 1705, Springer, 1999.
30. W. Hu, Y. Qu, and G. Cheng. Matching large ontologies: A divide-and-conquer approach. In *Data Knowl. Eng.*, 67(1):140–160, 2008.
31. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. IJCAI-2005*, pp. 466–471, 2005.
32. U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}$-description logic to disjunctive Datalog programs. In *Proc. KR-2004*, pp. 152–162. AAAI Press, 2004.
33. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.
34. M. Knorr, J. J. Alferes, and P. Hitzler. A coherent well-founded model for hybrid MKNF knowledge bases. In *Proc. ECAI-2008*, pp. 99–103. *Frontiers in Artificial Intelligence and Applications* 178, IOS Press, 2008.
35. D. Koller, A. Y. Levy, and A. Pfeffer. P-CLASSIC: A tractable probabilistic description logic. In *Proc. AAAI-2007*, pp. 390–397. AAAI Press, 1997.
36. T. Lukasiewicz. A novel combination of answer set programming with description logics for the Semantic Web. In *Proc. ESWC-2007*, pp. 384–398. *LNCS* 4519, Springer, 2007.
37. T. Lukasiewicz. A novel combination of answer set programming with description logics for the Semantic Web. *IEEE Trans. Knowl. Data Eng.*, 22(11):1577–1592, 2010.
38. T. Lukasiewicz. Expressive probabilistic description logics. *Artif. Intell.*, 172(6/7):852–883, 2008.
39. T. Lukasiewicz. Probabilistic description logic programs. *Int. J. Approx. Reason.*, 45(2):288–307, 2007.
40. T. Lukasiewicz. Tractable probabilistic description logic programs. In *Proc. SUM-2007*, pp. 143–156. *LNCS* 4772, Springer, 2007.
41. T. Lukasiewicz and U. Straccia. Top-k retrieval in description logic programs under vagueness for the Semantic Web. In *Proc. SUM-2007*, pp. 16–30. *LNCS* 4772, Springer, 2007.

42. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Repairing ontology mappings. In *Proc. AAAI-2007*, pp. 1408–1413. AAAI Press, 2007.
43. B. Motik, I. Horrocks, R. Rosati, and U. Sattler. Can OWL and logic programming live together happily ever after? In *Proc. ISWC-2006*, pp. 501–514. *LNCS* 4273, Springer, 2006.
44. B. Motik and R. Rosati. A faithful integration of description logics with logic programming. In *Proc. IJCAI-2007*, pp. 477–482. AAAI Press/IJCAI, 2007.
45. H. Nottelmann and N. Fuhr. Adding probabilities and rules to OWL Lite subsets based on probabilistic Datalog. *Int. J. Uncertainty Fuzziness Knowledge-Based Syst.*, 14(1):17–42, 2006.
46. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Sem.*, 10:133–173, 2008.
47. D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artif. Intell.*, 94(1/2):7–56, 1997.
48. L. Predoiu. Probabilistic models for the Semantic Web. In Z. Ma and H. Wang, editors, *The Semantic Web for Knowledge and Data Management: Technologies and Practices*, pp. 74–105. Information Science Reference, 2009.
49. L. Predoiu and H. Stuckenschmidt. A probabilistic framework for information integration and retrieval on the Semantic Web. In *Proc. InterDB-2007 Workshop on Database Interoperability*, 2007.
50. D. Ritze, C. Meilicke, O. Svab-Zamazal, and H. Stuckenschmidt. A pattern-based ontology matching approach for detecting complex correspondences. In *Proc. ISWC-2009 Workshop on Ontology Matching*.
51. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *J. Web Sem.*, 3(1):61–73, 2005.
52. R. Rosati. $\mathcal{DL}+log$: Tight integration of description logics and disjunctive Datalog. In *Proc. KR-2006*, pp. 68–78. AAAI Press, 2006.
53. F. Scharffe, J. de Bruijn, and D. Foxvog. Ontology mediation patterns library, V2. Deliverable D4.3.2, EU-IST Integrated Project (IP) IST-2003-506826 SEKT, February 2006.
54. F. Scharffe and D. Fensel. Correspondence patterns for ontology alignment. In *Proc. EKAW-2008*, pp. 83–92. *LNCS* 5268, Springer, 2008.
55. L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping languages for terminological knowledge. In *Proc. IJCAI-2005*, pp. 576–581, 2005.
56. U. Straccia. Towards top-k query answering in description logics: The case of *DL-Lite*. In *Proc. JELIA-2006*, pp. 439–451. *LNCS* 4160, Springer, 2006.
57. A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.
58. P. Wang and B. Xu. Debugging ontology mapping: A static method. *Comput. Inform.*, 27(1):21–36, 2008.