A Framework for Representing Ontology Mappings under Probabilities and Inconsistency

Andrea Calì [1], Thomas Lukasiewicz [2, 3], Livia Predoiu [4], and Heiner Stuckenschmidt [4]

[1] Computing Laboratory, University of Oxford, UK
andrea.cali@comlab.ox.ac.uk
[2] Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma, Italy
lukasiewicz@dis.uniroma1.it
[3] Institut für Informationssysteme, Technische Universität Wien, Austria
lukasiewicz@kr.tuwien.ac.at
[4] Institut für Informatik, Universität Mannheim, Germany
{heiner,livia}@informatik.uni-mannheim.de

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 mapping language and present a formalism that combines tightly integrated description logic programs with independent choice logic for representing probabilistic information. We define the language, 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. We also analyze the computational aspects of consistency checking and query processing in tightly integrated probabilistic description logic programs.

1 Introduction

The problem of aligning heterogeneous ontologies via semantic mappings has been identified as one of the major challenges of semantic web technologies. In order to address this problem, a number of languages for representing semantic relations between elements in different ontologies as a basis for reasoning and query answering across multiple ontologies have been proposed [21]. In the presence of real world ontologies, it is unrealistic to assume that mappings between ontologies are created manually by domain experts, since existing ontologies, e.g., in the area of medicine contain thousands of concepts and hundreds of relations. Recently, a number of heuristic methods for matching elements from different ontologies have been proposed that support the creation of mappings between different languages by suggesting candidate mappings (e.g., [7]). 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 [6,8]. 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 [7].

This means that automatically created mappings often contain uncertain hypotheses and errors that need to be dealt with, as 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 mapping, 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, that is, rule-based systems that use vocabulary from ontology knowledge bases. Another form of integration is to build ontologies on top of rules, where ontological definitions are supplemented by rules or imported from rules. Both types of integration 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 [5] or concepts and roles from L as unary resp. binary predicates in a tight integration [16] (see especially [5,18,16] for detailed overviews on the different types of description logic programs).

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 [12], web ontology languages [2,3], and description logic programs [15] (to encode ambiguous information, such as "John is a student with the probability 0.7 and a teacher with the probability 0.3", which is very different from vague/fuzzy information, such as "John is tall with degree of truth 0.7"). In particular, [15] extends the loosely integrated description logic programs of [5] by probabilistic uncertainty as in Poole's independent choice logic (ICL) [20]. The ICL is a powerful representation and reasoning formalism for single- and also multi-agent systems, which combines logic and probability, and which can represent a number of important uncertainty formalisms, in particular, influence diagrams, Bayesian networks, Markov decision processes, normal form games, and Pearl's causal models [10].

In this paper, we propose a language for representing and reasoning with uncertain and possibly inconsistent mappings, where the tight integration between ontology and rule languages (namely, the tightly integrated disjunctive description logic programs of [16]) is combined with probabilistic uncertainty (as in the ICL). The resulting language has the following useful features, which will be explained in more detail later:

- The semantics is based on a tight integration of the rule and the ontology language. 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 disjunctions in rule heads and 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.
– In [1], we show that consistency checking and query processing in the new rule language are decidable resp. computable, and can be reduced to their classical counterparts in tightly integrated disjunctive description logic programs. We also analyze the complexity of consistency checking and query processing in special cases.
– In [1], we show that there are tractable subsets of the language that are of practical relevance. In particular, we show that when ontologies are represented in *DL-Lite*, reasoning in the language can be done in polynomial time in the data complexity.

2 Representation Requirements

The problem of ontology matching can be defined as follows [7]. Ontologies are theories encoded in a certain language L. In this work, we assume that ontologies are encoded in OWL DL or OWL Lite. For each ontology O in language L, we denote by $Q(O)$ the matchable elements of the ontology O. Given two ontologies O and O', the task of matching is now to determine correspondences between the matchable elements in the two ontologies. Correspondences are 5-tuples (id, e, e', r, 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.

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

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

– *Support for mappings refinement:* The language should be expressive enough to allow the user to refine oversimplifying correspondences suggested by the matching system. This is important to be able to provide a 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 is 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 form the Ontology Alignment Evaluation Initiative[1] to illustrate the different aspects of mapping representation. In particular, we use examples from the benchmark and the conference data set. The benchmark dataset consists of five OWL ontologies (tests 101 and 301 to 304) describing scientific publications and related information. The conference dataset consists of about 10 OWL ontologies describing concepts related to conference organization and management. In both cases, we give examples of mappings that have been created by the participants of the 2006 evaluation campaign. In particular, we use mappings created by state-of-the-art ontology matching systems like falcon, hmatch, and coma++.

3 Description Logics

In this section, we recall the expressive description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which stand behind the web ontology languages OWL Lite and OWL DL [13], respectively. Intuitively, description logics model a domain of interest in terms of concepts and roles, which represent classes of individuals and binary relations between classes of individuals, respectively. A description logic knowledge base encodes especially subset relationships between concepts, subset relationships between roles, the membership of individuals to concepts, and the membership of pairs of individuals to roles.

3.1 Syntax. We first describe the syntax of $\mathcal{SHOIN}(\mathbf{D})$. We assume a set of *elementary datatypes* and a set of *data values*. A *datatype* is either an elementary datatype or a set of data values (*datatype oneOf*). A *datatype theory* $\mathbf{D} = (\Delta^{\mathbf{D}}, \cdot^{\mathbf{D}})$ consists of a *datatype domain* $\Delta^{\mathbf{D}}$ and a mapping $\cdot^{\mathbf{D}}$ that assigns to each elementary datatype a subset of $\Delta^{\mathbf{D}}$ and to each data value an element of $\Delta^{\mathbf{D}}$. The mapping $\cdot^{\mathbf{D}}$ is extended to all datatypes by $\{v_1, \ldots\}^{\mathbf{D}} = \{v_1^{\mathbf{D}}, \ldots\}$. Let $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ be pairwise disjoint (denumerable) sets of *atomic concepts*, *abstract roles*, *datatype roles*, and *individuals*, respectively. We denote by $\mathbf{R}_A^-$ the set of *inverses* R^- of all $R \in \mathbf{R}_A$.

A *role* is any element of $\mathbf{R}_A \cup \mathbf{R}_A^- \cup \mathbf{R}_D$. *Concepts* are inductively defined as follows. Every $\phi \in \mathbf{A}$ is a concept, and if $o_1, \ldots, o_n \in \mathbf{I}$, then $\{o_1, \ldots, o_n\}$ is a concept (*oneOf*). If ϕ, ϕ_1, and ϕ_2 are concepts and if $R \in \mathbf{R}_A \cup \mathbf{R}_A^-$, then also $(\phi_1 \sqcap \phi_2)$, $(\phi_1 \sqcup \phi_2)$, and $\neg\phi$ are concepts (*conjunction*, *disjunction*, and *negation*, respectively), as well as $\exists R.\phi$, $\forall R.\phi$, $\geqslant nR$, and $\leqslant nR$ (*exists*, *value*, *atleast*, and *atmost restriction*,

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

respectively) for an integer $n \geqslant 0$. If D is a datatype and $U \in \mathbf{R}_D$, then $\exists U.D$, $\forall U.D$, $\geqslant nU$, and $\leqslant nU$ are concepts (*datatype exists*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geqslant 0$. We write $\top$ and $\bot$ to abbreviate the concepts $\phi \sqcup \neg\phi$ and $\phi \sqcap \neg\phi$, respectively, and we eliminate parentheses as usual.

An *axiom* has one of the following forms: (1) $\phi \sqsubseteq \psi$ (*concept inclusion axiom*), where ϕ and ψ are concepts; (2) $R \sqsubseteq S$ (*role inclusion axiom*), where either $R, S \in \mathbf{R}_A \cup \mathbf{R}_A^-$ or $R, S \in \mathbf{R}_D$; (3) $\mathrm{Trans}(R)$ (*transitivity axiom*), where $R \in \mathbf{R}_A$; (4) $\phi(a)$ (*concept membership axiom*), where ϕ is a concept and $a \in \mathbf{I}$; (5) $R(a, b)$ (resp., $U(a, v)$) (*role membership axiom*), where $R \in \mathbf{R}_A$ (resp., $U \in \mathbf{R}_D$) and $a, b \in \mathbf{I}$ (resp., $a \in \mathbf{I}$ and v is a data value); and (6) $a = b$ (resp., $a \neq b$) (*equality* (resp., *inequality*) *axiom*), where $a, b \in \mathbf{I}$. A *(description logic) knowledge base* L is a finite set of axioms. For decidability, number restrictions in L are restricted to simple abstract roles [14].

The syntax of $\mathcal{SHIF}(\mathbf{D})$ is as the above syntax of $\mathcal{SHOIN}(\mathbf{D})$, but without the oneOf constructor and with the atleast and atmost constructors limited to 0 and 1.

3.2 Semantics. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ relative to a datatype theory $\mathbf{D} = (\Delta^\mathbf{D}, \cdot^\mathbf{D})$ consists of a nonempty (*abstract*) *domain* $\Delta^\mathcal{I}$ disjoint from $\Delta^\mathbf{D}$, and a mapping $\cdot^\mathcal{I}$ that assigns to each atomic concept $\phi \in \mathbf{A}$ a subset of $\Delta^\mathcal{I}$, to each individual $o \in \mathbf{I}$ an element of $\Delta^\mathcal{I}$, to each abstract role $R \in \mathbf{R}_A$ a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and to each datatype role $U \in \mathbf{R}_D$ a subset of $\Delta^\mathcal{I} \times \Delta^\mathbf{D}$. We extend $\cdot^\mathcal{I}$ to all concepts and roles, and we define the *satisfaction* of an axiom F in an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, denoted $\mathcal{I} \models F$, as usual [13]. We say $\mathcal{I}$ *satisfies* the axiom F, or $\mathcal{I}$ is a *model* of F, iff $\mathcal{I} \models F$. We say $\mathcal{I}$ *satisfies* a knowledge base L, or $\mathcal{I}$ is a *model* of L, denoted $\mathcal{I} \models L$, iff $\mathcal{I} \models F$ for all $F \in L$. We say L is *satisfiable* iff L has a model. An axiom F is a *logical consequence* of L, denoted $L \models F$, iff every model of L satisfies F.

4 Description Logic Programs

In this section, we recall the novel approach to *description logic programs* (or *dl-programs*) $KB = (L, P)$ from [16], where KB consists of a description logic knowledge base L and a disjunctive logic program P. Their semantics is defined in a modular way as in [5], 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 their 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. See especially [16] for further details on the new 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 $\mathbf{V}$ (relative to a datatype theory $\mathbf{D} = (\Delta^\mathbf{D}, \cdot^\mathbf{D})$) and pairwise disjoint (denumerable) sets $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ of atomic concepts, abstract roles, datatype roles, and individuals, respectively, as in

Section 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 *literal* l is an atom p or a default-negated atom $not\, p$. A *disjunctive rule* (or simply *rule*) r is an expression of the form

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

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

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

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

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

An *interpretation* I is any subset of HB_Φ. Informally, every such I represents the Herbrand interpretation in which all $a \in I$ (resp., $a \in HB_\Phi - I$) are true (resp., false). We say an interpretation I is a *model* of a description logic knowledge base L, denoted $I \models L$, iff $L \cup I \cup \{\neg a \mid a \in HB_\Phi - I\}$ is satisfiable. We say I is a *model* of a ground atom $a \in HB_\Phi$, or I *satisfies* a, denoted $I \models a$, iff $a \in I$. We say I is a *model* of a ground rule r, denoted $I \models r$, iff $I \models \alpha$ for some $\alpha \in H(r)$ whenever $I \models B(r)$, that is, $I \models \beta$ for all $\beta \in B^+(r)$ and $I \not\models \beta$ for all $\beta \in B^-(r)$. We say I is a *model* of a set of rules P iff $I \models r$ for every $r \in ground(P)$. We say I is a *model* of a disjunctive dl-program $KB = (L, P)$, denoted $I \models KB$, iff I is a model of both L and P.

We now define the answer set semantics of disjunctive dl-programs by generalizing the ordinary answer set semantics of disjunctive logic programs. We generalize the definition via the FLP-reduct [9] (which coincides with the answer set semantics defined via the Gelfond-Lifschitz reduct [11]). Given a dl-program $KB = (L, P)$, the *FLP-reduct* of KB relative to an interpretation $I \subseteq HB_\Phi$, denoted KB^I, is the dl-program (L, P^I), where P^I is the set of all $r \in ground(P)$ such that $I \models B(r)$. An interpretation $I \subseteq HB_\Phi$ is an *answer set* of KB iff I is a minimal model of KB^I. A dl-program KB is *consistent* (resp., *inconsistent*) iff it has an (resp., no) answer set.

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

4.3 Semantic Properties. We now summarize some important semantic properties of disjunctive dl-programs under the above answer set semantics. In the ordinary case, every answer set of a disjunctive program P is also a minimal model of P, and the converse holds when P is positive. This result holds also for disjunctive dl-programs.

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

Theorem 4.1 (see [16]). *Let $KB{=}(L, P)$ be a disjunctive dl-program with $L{=}\emptyset$. Then, the set of all answer sets of KB coincides with the set of all ordinary answer sets of P.*

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

Theorem 4.2 (see [16]). *Let $KB = (L, P)$ be a positive disjunctive dl-program, and let α be a ground atom from HB_Φ. Then, α is true in all answer sets of KB iff α is true in all first-order models of $L \cup ground(P)$.*

4.4 Representing Mappings. Tightly integrated disjunctive dl-programs $KB = (L, P)$ provide a natural way for representing mappings between two heterogeneous ontologies 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 (representing O_1 resp. O_2) with signatures $\mathbf{A}_1$, $\mathbf{R}_{A,1}$, $\mathbf{R}_{D,1}$, $\mathbf{I}_1$ and $\mathbf{A}_2$, $\mathbf{R}_{A,2}$, $\mathbf{R}_{D,2}$, $\mathbf{I}_2$, respectively, such that $\mathbf{A}_1 \cap \mathbf{A}_2 = \emptyset$, $\mathbf{R}_{A,1} \cap \mathbf{R}_{A,2} = \emptyset$, $\mathbf{R}_{D,1} \cap \mathbf{R}_{D,2} = \emptyset$, and $\mathbf{I}_1 \cap \mathbf{I}_2 = \emptyset$. Note that this can easily be achieved for any pair of ontologies by a suitable renaming. A mapping between elements e and e' from L_1 and L_2, respectively, is then represented by a simple rule $e'(\overrightarrow{x}) \leftarrow e(\overrightarrow{x})$ in P, where $e \in \mathbf{A}_1 \cup \mathbf{R}_{A,1} \cup \mathbf{R}_{D,1}$, $e' \in \mathbf{A}_2 \cup \mathbf{R}_{A,2} \cup \mathbf{R}_{D,2}$, and $\overrightarrow{x}$ is a suitable variable vector. Note that the fact that we demand that the signatures of L_1 and L_2 are disjoint guarantees that the rule base that represents mappings between different ontologies is stratified as long as there are no cyclic mapping relations.

Taking some examples from the conference data set of the OAEI challenge 2006, we find e.g. the following mappings that were created by automatic matching systems:[2]

$$\begin{aligned} &NegativeReview(X) \leftarrow Review(X)\,; \\ &NeutralReview(X) \leftarrow Review(X)\,; \\ &PositiveReview(X) \leftarrow Review(X)\,. \end{aligned}$$

Another example of created mapping relations are the following:[3]

$$\begin{aligned} &EarlyRegisteredParticipant(X) \leftarrow participant(X)\,; \\ &LateRegisteredParticipant(X) \leftarrow participant(X)\,. \end{aligned}$$

[2] Results of the hmatch system for mapping the SIGKDD on the EKAW Ontology.
[3] Results of the hmatch system for mapping the CRS on the EKAW Ontology.

Both of these sets of correspondences are examples of mappings that introduce inconsistency in the target ontology. The reason is that the three concepts $NegativeReview$, $NeutralReview$, and $PositiveReview$, as well as the two concepts $EarlyRegistered$-$Participant$ and $LateregisteredParticipant$ are defined to be disjoint in the corresponding ontologies. Using the rules as shown above will make an instance of the concept $Review$ (resp., $participant$) a member of disjoint classes. In [17], we have presented a method for detecting such inconsistent mappings. There are different approaches for resolving this inconsistency. The most straightforward one is to drop mappings until no inconsistency is present anymore. Peng and Xu [19] have proposed a more suitable method for dealing with inconsistencies in terms of a relaxation of the mappings. In particular, they propose to replace a number of conflicting mappings by a single mapping that includes a disjunction of the conflicting concepts. In the first example above, we would replace the three rules by the following one:

$$NegativeReview(X) \vee NeutralReview(X) \vee PositiveReview(X) \leftarrow Review(X)\,.$$

This new mapping rule can be represented in our framework and resolves the inconsistency. In this particular case, it also correctly captures the meaning of the concepts.

In principle, the second example can be solved using the same approach. In this case, however, the actual semantics of the concepts can be captured more accurately by refining the rules and making use of the full expressiveness of the mapping language. In particular, we can resolve the inconsistency by extending the body of the mapping rules with additional requirements:

$$EarlyRegisteredParticipant(X) \leftarrow participant(X) \wedge RegisterdbeforeDeadline(X)\,;$$
$$LateRegisteredParticipant(X) \leftarrow participant(X) \wedge not\ RegisteredbeforeDeadline(X)\,.$$

This refinement of the mapping rules resolves the inconsistency and also provides a more correct mapping. A drawback of this approach is the fact that it requires manual post-processing of mappings. In the next section, we present a probabilistic extension of tightly integrated disjunctive dl-programs that allows us to directly use confidence estimations of matching engines to resolve inconsistencies and to combine the results of different matchers.

5 Probabilistic Description Logic Programs

In this section, we present a *tightly integrated* approach to *probabilistic disjunctive description logic programs* (or simply *probabilistic dl-programs*) *under the answer set semantics*. Differently from [15] (in addition to being a tightly integrated approach), the probabilistic dl-programs here also allow for disjunctions in rule heads. Similarly to the probabilistic dl-programs in [15], they are defined as a combination of dl-programs with Poole's ICL [20], but using the tightly integrated disjunctive dl-programs of [16] (see Section 4), rather than the loosely integrated dl-programs of [5]. Poole's ICL is based on ordinary acyclic logic programs P under different "choices", where every choice along with P produces a first-order model, and one then obtains a probability distribution over the set of all first-order models by placing a probability distribution over the different choices. We use the tightly integrated disjunctive dl-programs under the answer set semantics of [16], instead of ordinary acyclic logic programs under

their canonical semantics (which coincides with their answer set semantics). We first introduce the syntax of probabilistic dl-programs and then their answer set semantics.

5.1 Syntax. We now define the syntax of probabilistic dl-programs and 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, every probability μ is a probability distribution over the set of all variable assignments. Since C and all its alternatives are finite, μ can be defined by (i) a mapping $\mu\colon \bigcup C \rightarrow [0,1]$ such that $\sum_{a \in A} \mu(a) = 1$ for all $A \in C$, and (ii) $\mu(B) = \Pi_{b \in B} \mu(b)$ for all total choices B of C. Intuitively, (i) defines a probability over the values of each random variable of C, and (ii) assumes independence between the random variables.

A *probabilistic dl-program* $KB = (L, P, C, \mu)$ consists of a disjunctive dl-program (L, P), a choice space C such that no atomic choice in C coincides with the head of any rule in $ground(P)$, and a probability μ on C. Intuitively, since the total choices of C select subsets of P, and μ is a probability distribution on the total choices of C, every probabilistic dl-program is the compact representation of a probability distribution on a finite set of disjunctive dl-programs. Observe here that P is fully general and not necessarily stratified or acyclic. We say KB is *normal* iff P is normal. A *probabilistic query* to KB has the form $\exists(c_1(\boldsymbol{x}) \vee \cdots \vee c_n(\boldsymbol{x}))[r, s]$, where $\boldsymbol{x}, r, s$ is a tuple of variables, $n \geqslant 1$, and each $c_i(\boldsymbol{x})$ is a conjunction of atoms constructed from predicate and constant symbols in Φ and variables in $\boldsymbol{x}$. Note that the above probabilistic queries can also be easily extended to conditional expressions as in [15].

5.2 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.

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

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

infimum (resp., supremum) of $Pr_\sigma(q(\boldsymbol{x}))$ subject to all answer sets Pr of KB and all σ. A *correct* (resp., *tight*) *answer* to a probabilistic query $\exists(c_1(\boldsymbol{x}) \vee \cdots \vee c_n(\boldsymbol{x}))[r, s]$ is a ground substitution θ (for the variables $\boldsymbol{x}, r, s$) such that $(c_1(\boldsymbol{x}) \vee \cdots \vee c_n(\boldsymbol{x}))[r, s]\,\theta$ is a consequence (resp., tight consequence) of KB.

5.3 Representing and Combining Confidence Values. The probabilistic extension of disjunctive dl-programs $KB = (L, P)$ to 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 can interpret the confidence value as an *error probability* and state that the probability that a 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 can 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 $1 - (1 - n_1) \cdot (1 - n_2) \geqslant n_2$.

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

We illustrate this approach using an example from the benchmark data set of the OAEI 2006 campaign. In particular, we consider the case where the publication ontology in test 101 (O_1) is mapped on the ontology of test 302 (O_2). Below we show some mappings that have been detected by the matching system hmatch that participated in the challenge. The mappings are described as rules in P, which contain a conjunct indicating the matching system that has created it and a 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):

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

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

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

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

Note that just putting together the rules without considering the choice space would lead to the same inconsistency problems shown in the last section, because the concepts $Book$ and $InCollection$ are disjoint. Further, the fact that the mapping between the concepts $Proceeding_1$ and $Proceeding_2$ has been found by both matchers is not considered and this mapping rule would have the same status as any other rule in the mapping.

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

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

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', that is, $(\mu \cdot \mu')(B \cup B') = \mu(B) \cdot \mu'(B')$ for all total choices B of C and B' of C'), respectively, by adding the alternative $\{sel\_hmatch_1, sel\_falcon_1\}$ and the 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.

6 Summary and Outlook

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

We leave for future work the implementation of the language and the performing of experiments on the basis of large data sets, to further substantiate our claims that this formal framework is suited for realistic applications of ontology mappings.

Acknowledgments. Andrea Calì is supported by the EU STREP FET project TONES (FP6-7603). Thomas Lukasiewicz is supported by the German Research Foundation

(DFG) under the Heisenberg Programme and by the Austrian Science Fund (FWF) under the project P18146-N04. Heiner Stuckenschmidt and Livia Predoiu are supported by an Emmy-Noether Grant of the German Research Foundation (DFG).

References

1. 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. (See also Report RR-1843-07-05, Institut für Informationssysteme, TU Wien, 2007.)
2. P. C. G. da Costa. *Bayesian Semantics for the Semantic Web*. Doctoral Dissertation, George Mason University, Fairfax, VA, USA, 2005.
3. 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.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. *DL-Lite*: Tractable description logics for ontologies. In *Proc. AAAI-2005*, pp. 602–607. AAAI Press, 2005.
5. T. Eiter, T. Lukasiewicz, R. Schindlauer, H. Tompits. Combining answer set programming with description logics for the Semantic Web. In *Proc. KR-2004*, pp. 141–151. AAAI Press, 2004. (See also Report RR-1843-07-04, Institut für Informationssysteme, TU Wien, 2007.)
6. 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*.
7. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, Heidelberg, Germany, 2007.
8. J. Euzenat, H. Stuckenschmidt, and M. Yatskevich. Introduction to the ontology alignment evaluation 2005. In *Proc. K-CAP-2005 Workshop on Integrating Ontologies*.
9. 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.
10. 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.
11. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.
12. 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.
13. 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.
14. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. LPAR-1999*, pp. 161–180. *LNCS* 1705, Springer, 1999.
15. T. Lukasiewicz. Probabilistic description logic programs. *Int. J. Approx. Reason.*, 45(2):288–307, 2007.
16. 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.
17. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Repairing ontology mappings. In *Proc. AAAI-2007*, pp. 1408–1413. AAAI Press, 2007.
18. 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.
19. P. Wang and B. Xu. Debugging ontology mapping: A static method. *Computation and Intelligence*, 2007. To appear.
20. D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artif. Intell.*, 94(1/2):7–56, 1997.
21. L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping languages for terminological knowledge. In *Proc. IJCAI-2005*, pp. 576–581, 2005.