Representing Ontology Mappings with Probabilistic Description Logic Programs (extended abstract)

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

[1]Oxford-Man Institute of Quantitative Finance, University of Oxford, UK
[2]Computing Laboratory, University of Oxford, UK
[3]Institut für Informationssysteme, Technische Universität Wien, Austria
[4]Institut für Informatik, Universität Mannheim, Germany
andrea.cali@comlab.ox.ac.uk
thomas.lukasiewicz@comlab.ox.ac.uk
{livia,heiner}@informatik.uni-mannheim.de

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 [28]. In the presence of real world ontologies, it is unrealistic to assume that mappings between ontologies are created manually by domain experts, due to the large size of existing ontologies. Recently, a number of heuristic methods, relying on linguistic and structural criteria, for matching elements from different ontologies have been proposed that support the creation of mappings between different languages by suggesting candidate mappings (e.g., [11]). Such methods often trade off precision and recall, as shown by evaluation studies [10,12].

Automatically created mappings often contain uncertain hypotheses and errors that need to be dealt with: *(i)* mapping hypotheses are often oversimplifying, supporting very simple semantic relations (mostly equivalence between individual elements); *(ii)* there may be conflicts between different hypotheses for semantic relations from different matching components and often even from the same matcher; *(iii)* semantic relations are only given with a degree of confidence in their correctness. 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.

There is a large body of work on integrating ontologies and rules. 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 [7,8]

or concepts and roles from L as unary resp. binary predicates in a tight integration [19] (see especially [8,25,19] for detailed overviews on the different types of description logic programs).

Other works explore formalisms for *uncertainty reasoning in the Semantic Web*, which are especially probabilistic extensions of description logics [16,20], web ontology languages [3,4], and description logic programs [21].

In this paper, we propose *tightly integrated probabilistic description logic programs under the answer set semantics* as a language for representing and reasoning with uncertain and possibly inconsistent ontology mappings. The approach is a tight integration of disjunctive logic programs under the answer set semantics, the expressive description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, and Bayesian probabilities. More precisely, the tight integration between ontology and rule languages of [19] is combined with probabilistic uncertainty as in the ICL [27].

The probabilistic description logic programs here are very different from the ones in [21] (and their recent tractable variant in [22]). First, they are based on the tight integration between the ontology component L and the rule component P of [19], while the ones in [21,22] realize the loose query-based integration between the ontology component L and the rule component P of [7]. This implies in particular that the vocabularies of L and P here may have common elements, while the vocabularies of L and P in [21,22] are necessarily disjoint.

2 Tightly Integrated Disjunctive DL-Programs

In this section, we recall the *tightly integrated* approach to *disjunctive description logic programs* (or simply *disjunctive dl-programs*) $KB = (L, P)$ under the answer set semantics from [19], where KB consists of a description logic knowledge base L and a disjunctive logic program P. The description logic languages that we consider here are $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which stand behind the web ontology languages OWL Lite and OWL DL [17], respectively, and for which we refer the reader, e.g., to the full version [2]. The semantics of disjunctive dl-programs is defined in a modular way as in [7,8], 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 disjunctive dl-programs are decidable, even when their components of logic programs and description logic knowledge bases are both very expressive. We refer especially to [19] for further details on the novel approach to disjunctive dl-programs and for a detailed comparison to related works.

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

Semantics. We adopt the *answer set semantics* for disjunctive dl-programs as a generalization of the answer set semantics of ordinary disjunctive logic programs. We refer the reader to the full version of this paper [2] for formal details on such a semantics.

3 Tightly Integrated Probabilistic DL-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 [21] (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 [21], they are defined as a combination of dl-programs with Poole's ICL [27], but using the tightly integrated disjunctive dl-programs of [19], rather than the loosely integrated dl-programs of [7,8]. 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 [19], 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.

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, 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 *tightly integrated probabilistic disjunctive description logic program* (or simply *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 [21].

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 the 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 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.

4 Representing Ontology Mappings with Confidence Values

We now show how a tightly integrated probabilistic dl-program $KB = (L, P, C, \mu)$ 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 the ontologies, where confidence values can be encoded as error probabilities, and inconsistencies can also be resolved via trust probabilities (in addition to using disjunctions and nonmonotonic negations in P).

The probabilistic extension of tightly integrated disjunctive 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 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.

Example 4.1. 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 (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 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 $\mu'(falcon_1) = 0.94$, $\mu'(not\_falcon_1) = 0.06$, $\mu'(falcon_2) = 0.96$, and $\mu'(not\_falcon_2) = 0.04$.

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

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

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

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 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 is given by $\theta = \{R/0, S/0\}$, if pub is not an instance of the concept $Collection$ in the ontology O_1 (since there is no mapping rule that maps another concept than $Collection$ to the concept $Book$). If pub is an instance of the concept $Collection$, however, then the tight answer to Q is given by $\theta = \{R/0.341, S/0.341\}$ (as $\mu''(hmatch_1) \cdot \mu''(sel\_hmatch_1) = 0.62 \cdot 0.55 = 0.341$). Informally, pub belongs to the concept $Book$ with the probabilities 0 resp. 0.341. Note that we may obtain real intervals when there are total choices with multiple answer sets.

5 Tractability Results

We define stratified normal dl- and stratified normal probabilistic dl-programs as follows. A normal dl-program $KB = (L, P)$ is *stratified* iff (i) L is defined in *DL-Lite* [5] and (ii) $trans(P)$ is locally stratified. A probabilistic dl-program $KB = (L, P, C, \mu)$ is *normal* iff P is normal. A normal probabilistic dl-program $KB = (L, P, C, \mu)$ is *stratified* iff every of KB's represented dl-programs is stratified.

The following result shows that stratified normal probabilistic dl-programs allow for consistency checking and query processing with a polynomial data complexity.

Theorem 5.1. *Given Φ and a stratified normal probabilistic dl-program KB, (a) deciding if KB has an answer set, and (b) computing $l, u \in [0, 1]$ for a given ground atom q such that $KB \mid\sim_{tight}(q)[l, u]$ can be done in polynomial time in the data complexity.*

Acknowledgements. Andrea Calì is supported by the EPSRC project EP/E010865/1 *Schema Mappings and Automated Services for Data Integration and Exchange*. 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.
2. 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.
3. P. C. G. da Costa. *Bayesian Semantics for the Semantic Web*. Doctoral Dissertation, George Mason University, Fairfax, VA, USA, 2005.
4. 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.
5. 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.
6. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.
7. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. In *Proc. KR-2004*, pp. 141–151. AAAI Press, 2004.

8. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artif. Intell.*, in press.
9. 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.
10. 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*.
11. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, Heidelberg, Germany, 2007.
12. J. Euzenat, H. Stuckenschmidt, and M. Yatskevich. Introduction to the ontology alignment evaluation 2005. In *Proc. K-CAP-2005 Workshop on Integrating Ontologies*.
13. 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.
14. 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.
15. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.
16. 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.
17. 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.
18. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. LPAR-1999*, pp. 161–180. *LNCS* 1705, Springer, 1999.
19. 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.
20. T. Lukasiewicz. Expressive probabilistic description logics. *Artif. Intell.*, 172(6/7):852–883, 2008.
21. T. Lukasiewicz. Probabilistic description logic programs. *Int. J. Approx. Reason.*, 45(2):288–307, 2007.
22. T. Lukasiewicz. Tractable probabilistic description logic programs. In *Proc. SUM-2007*, pp. 143–156. *LNCS* 4772, Springer, 2007.
23. 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.
24. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Repairing ontology mappings. In *Proc. AAAI-2007*, pp. 1408–1413. AAAI Press, 2007.
25. 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.
26. P. Wang and B. Xu. Debugging ontology mapping: A static method. *Computation and Intelligence*, 2007. To appear.
27. D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artif. Intell.*, 94(1/2):7–56, 1997.
28. L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping languages for terminological knowledge. In *Proc. IJCAI-2005*, pp. 576–581, 2005.
29. 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.