Querying Probabilistic Ontologies with SPARQL

Joerg Schoenfisch

Data & Web Science Research Group
University of Mannheim, Germany
http://dws.informatik.uni-mannheim.de/

joerg@informatik.uni-mannheim.de

Abstract: In recent years a lot of efforts was put into the field of Semantic Web research to specify knowledge as precisely as possible. However, optimizing for precision alone is not sufficient. The handling of uncertain or incomplete information is getting more and more important and it promises to significantly improve the quality of query answering in Semantic Web applications. My plan is to develop a framework that extends the rich semantics offered by ontologies with probabilistic information, stores this in a probabilistic database and provides query answering with the help of query rewriting. In this proposal I describe how these three aspects can be combined. Especially, I am focusing on how uncertainty is incorporated into the ABox and how it is handled by the database and the rewriter during query answering.

1 Introduction

In recent years several standards for knowledge representation, e.g. RDF[1] or OWL[2], were recommended by the W3C, and methodologies, like OntoClean [GW09], for putting these recommendations to work evolved. Those formalisms and methods focus on specifying knowledge as precisely as possible. OntoClean, for example, categorizes expressions in an ontology into rigid and non-rigid ones and discourages the usage of non-rigid knowledge to avoid false reasoning results.

However, as Paulheim and Pan already stated, optimizing the Semantic Web for precision alone is not sufficient [PP12]. It is trivial to design a system that always achieves very high precision, but subsequently it will suffer from very low recall. The impact for linked data or ontologies is so severe that IBM decided not to use structured knowledge bases in this way for their famous Jeopardy-winning Watson system, as it would only have been able to answer about two percent of the posed questions [KBP+12]. Instead, they developed a system that allows for imprecise or uncertain knowledge to improve recall and achieve a better overall performance.

The handling of uncertain or incomplete information is getting more and more important,

[1] http://www.w3.org/standards/techs/rdf
[2] http://www.w3.org/standards/techs/owl

not only in the domain of expert systems or query-answering systems like Watson, but also in other real world applications:

- *Data integration* is the mapping of two or more heterogeneous data sources, e.g. ontologies without a common upper ontology or database with a different schema, into one combined knowledge base. In many cases it is not possible to find perfect matches between all concepts and properties of the different sources. However, quite often there are concepts that almost correspond to each other or have similar meanings. Mapping those with some specific confidence value yields an increased benefit for the resulting integrated knowledge base.
- *Information extraction* is another field where uncertain information plays an important role. Information extraction tries to generate meaningful, structured knowledge from merely unstructured data in the form of natural text. Although natural language processing provides good tools to derive information from unstructured text, natural language has many ambiguities that require context or background knowledge to discern the correct meaning of a term or statement (cf. [TDD+13]).

For all these cases query answering can be improved by supporting uncertain information properly. I suggest a framework that combines the rich semantic knowledge available within ontologies and linked data with the benefit of an increased recall provided by the incorporation of imprecise knowledge when answering queries. This framework shall also scale well for large amounts of information as for example in Linked Open Data. My contribution will thus be the combination of the probabilistic results from the database with the information from a Semantic Web reasoner.

There are already various implementations for formalisms that incorporate uncertain or probabilistic knowledge, e.g. probabilistic relational databases [HAKO09], bayesian networks [WMGH08], or markov logic networks [JS12]. The former are focused on querying large relational data, however, without rich semantics, whereas the latter belong to the field of probabilistic programming and are more concerned with statistical relational learning.

Storing and querying large amounts of semantic data is already possible in various implementations, e.g. Virtuoso[3], Stardog[4], OWLIM[5] or Oracle[6], but these applications do not support any notion of uncertainty.

Probabilistic databases are around for about a decade and are in a very mature state [TDD+13]. With their focus on large databases and their foundation in relational database management systems, they are a natural choice for the combination of probabilistic knowledge bases and querying those with SPARQL.

The OWL 2 recommendation defines a special profile, OWL 2 QL[7], that is designed to enable query answering in relational databases with large amounts of instance data in mind.

[3] http://virtuoso.openlinksw.com/
[4] http://stardog.com/
[5] https://www.ontotext.com/owlim
[6] http://www.oracle.com/technetwork/database/options/spatialandgraph/overview/rdfsemantic-graph-1902016.html
[7] http://www.w3.org/TR/owl2-profiles/#OWL_2_QL

$$\exists sisterOf \sqsubseteq Woman \quad [1.00] \quad (1)$$
$$\exists motherOf \sqsubseteq Woman \quad [1.00] \quad (2)$$
$$\exists marriedTo \sqcap Woman \sqsubseteq Wife \quad [1.00] \quad (3)$$
$$marriedTo \equiv marriedTo^{-1} \quad [1.00] \quad (4)$$
$$\exists motherOf \sqsubseteq \exists marriedTo \quad [0.65] \quad (5)$$

$$marriedTo(Alice, Bob) \quad [0.30] \quad (6)$$
$$sisterOf(Alice, Carol) \quad [0.80] \quad (7)$$
$$motherOf(Alice, Dave) \quad [0.95] \quad (8)$$

Figure 1: Example used to illustrate my approach in the proposed framework. Numbers in square brackets represent probabilities.

This is achieved by limiting the expressivity of the ontology, leading to a reduced complexity of the reasoning task for the knowledge base. Due to this, answers to a query can be computed through a technique called query rewriting. The main benefit of query rewriting is the fact that it needs no information about the instance data (ABox) of a knowledge base, but only of its schema (TBox). This enables reasoning during query time and changing the data without the need for a separate reasoning step afterwards which results in answers to a query directly reflecting changes in the knowledge base.

The recommended language to query semantic data is SPARQL. It looks similar to SQL but it is designed to query graphs (the underlying structure of RDF or OWL) instead of relational data.

The example shown in Figure 1 will be used throughout Section 3 to illustrate my approach in the framework. The ontology consists of 5 axioms in the TBox and 3 assertions in the ABox stating the following:

1. A sister of someone is a woman
2. A mother of someone is a woman
3. Someone who is married and a woman is a wife
4. The married property is symmetric
5. Alice is married to Bob with a probability of 0.30
6. The probability of Alice being Carol's sister is 0.80
7. The probability of Alice being Dave's mother is 0.95

I will show in detail how a query is rewritten using the certain TBox axioms 1 – 4. How the uncertain Axiom 5 can be incorporated during rewriting is briefly discussed at the end of Section 3 and is part of future work. I describe which of the assertions 6 – 8 are relevant answers to the query and how to compute the probabilities of the final result from their given probabilities.

The rest of this paper is structured as follows: In Section 2, I am discussing related work.

Section 3 details the approach of my framework. The current status and short term goals of my work are stated in Section 4. Section 5 outlines future work.

2 Related Work

Related work can be divided roughly into two fields: probabilistic querying, and Semantic Web reasoning. Probabilistic deductive databases, e.g. as proposed by Lakshmanan et al. [LS94], are also closely related to my approach. However, they employ technologies different from query rewriting and do not focus on large scale data.

2.1 Probabilistic Querying

The two main areas of research in probabilistic querying are probabilistic programming and probabilistic databases [TDD+13]. Both have developed rather independently during the last decade but they share a common goal in making probabilistic inference scalable.

Probabilistic programming mainly focuses on learning statistical models from existing data. Its main concern is determining whether there exists a solution to a query rather than finding all solutions as probabilistic databases do. There are several formalisms, e.g. markov logic networks [RD06], bayesian networks [Jen96], or languages and tools like ProbLog [KTD07], to describe probabilistic models. These formalisms and the tools using them, for example MarkoViews [JS12] and BayesStore [WMGH08], can handle large amounts of uncertain data, however, none of them incorporates Semantic Web reasoning.

Similarly, probabilistic databases like Trio [Wid04] or MayBMS [HAKO09] are designed for querying large probabilistic knowledge bases, but they are only handling relational data without rich semantic information. Probabilistic databases usually employ a *possible world semantics*: all possible values for uncertain values are represented at once. To enable efficient inferencing, they exploit independence assumptions, most commonly tuple-independence (all database tuples or rows are independent from each other) or block-independent (groups of tuples are independent from each other). The research on probabilistic databases draws from many years of experience with relational databases. From those, they adopt relational algebra – and thus SQL – or Datalog.

Theobald et al. [TDD+13] give a more thorough overview of these two research areas, their methodologies, differences, and commonalities.

2.2 Semantic Web Reasoning in the Context of Query Answering

In the context of query answering, there are two commonly used approaches for Semantic Web reasoning with large scale instance data: *materialization* and *query rewriting*.

For materialization all knowledge that can be inferred from known facts is explicitly stored when the data is loaded. Thus, no reasoning is needed later on, as long as the data is not modified. If the data is modified the materialization has to be computed anew which can be quite expensive. For example OWLIM uses this approach.

Query rewriting, on the other side, does not change or add data but rather transforms a query over a knowledge base in such a way that it produces sound and complete answers without the need for any reasoning inside the database. In the case of OWL 2 QL and similar description logics this means that the query is expanded with information from the TBox such that implicit knowledge can be extracted without the need for any consideration of the ABox during reasoning. This is achieved by limiting the expressiveness of the description logic which is used to describe the data. Calvanese et al. [CDL05] give a thorough overview in the their paper about the $DL\text{-}Lite$ family of description logics. The OWL 2 QL profile is based on $DL\text{-}Lite_R$. PerfectRef, one of the first rewriting algorithms for semantic web reasoning, is also introduced in their paper. Applications that use query rewriting for reasoning are for example Stardog or the Maestro system [CDL$^+$11]. None of the applications for large scale semantic web reasoning incorporates uncertain information.

In my framework I want to adopt a query rewriting algorithm. There are already some implementations that combine those algorithms with traditional relational databases, e.g. Owlgres [SS08], Oracle RDF Semantic Graph, or YARR [SO13]. However they are outdated (Owlgres), or not readily available (Oracle, YARR).

3 Approach

As outlined in the introduction I suggest to use a mature probabilistic relational database system and combine it with an existing query rewriting algorithm for reasoning. I use SPARQL as query language, which is recommend by the W3C for Semantic Web applications. Figure 2 shows an illustration of my approach. The two main parts of the implementation will be the query rewriter and the probabilistic database system. The query rewriter will use the knowledge from the TBox of one or more consistent ontologies to extend the queries. The information of the ABox will be stored in the database, where the computation of the results takes place. Ontologies have to be provided in OWL format, queries will be posed in SPARQL. The communication between the rewriter and the database is done via SQL (with probabilistic extensions). The results will be presented to the user as tables, as is common for databases or triple stores.

As query rewriter I am going to use the implementation of IQAROS[8] described by Venetis et al. [VSS12]. IQAROS loads the TBox of an ontology in OWL format and rewrites queries formulated in Datalog. To be able to pose queries in SPARQL and compute answers through the probabilistic database, I will implement two additional translation steps: one from SPARQL to Datalog before the rewriting takes place, and another from Datalog to SQL with probabilistic extensions for the probabilistic database.

[8] IQAROS is one of the few openly available implementations. It can be downloaded under this link: `https://code.google.com/p/iqaros/`

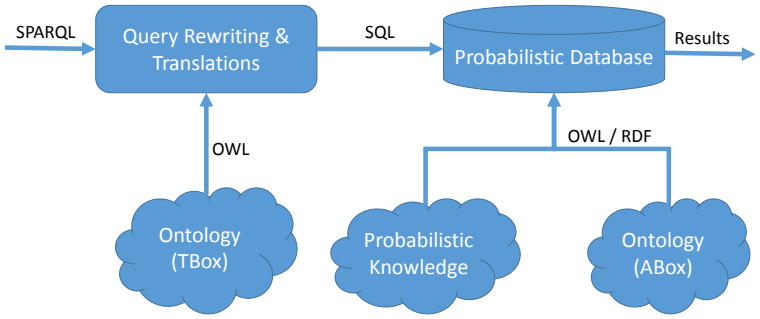

Figure 2: Illustration of the different steps of my approach and the technologies involved.

Given the example from the introduction (Fig. 1), a SPARQL query asking for all wifes looks like the following:

```
SELECT ?person WHERE { ?person a Wife . }
```

Translating SPARQL queries with simple triple patterns like that to Datalog is straightforward. Other features like optional joins or aggregates, e.g. COUNT or MAX, are more complex. These constructs are not part of Datalog, however, they are supported by SQL and probabilistic databases and thus can be used with some special handling in the translation and rewriting steps. This is the query translated to Datalog:

```
q(?person) :- Wife(?person)
```

The relevant axioms of the TBox that will be used by the rewriter to extend the query are then the following: a) a wife is married and a woman (Axiom 3), b) a person who is the mother of someone is a woman (Axiom 2), c) a person who is the sister of someone is a woman (Axiom 1), d) the relation marriedTo is symmetric (Axiom 4). The extended query looks like this:

$$q(?person) :- Wife(?person) \tag{9}$$
$$q(?person) :- marriedTo(?person, ?x), Woman(?person) \tag{10}$$
$$q(?person) :- marriedTo(?x, ?person), Woman(?person) \tag{11}$$
$$q(?person) :- marriedTo(?person, ?x), sisterOf(?person, ?y) \tag{12}$$
$$q(?person) :- marriedTo(?x, ?person), sisterOf(?person, ?y) \tag{13}$$
$$q(?person) :- marriedTo(?person, ?x), motherOf(?person, ?y) \tag{14}$$
$$q(?person) :- marriedTo(?x, ?person), motherOf(?person, ?y) \tag{15}$$

The initial query remains unchanged (9). The next two queries (10 and 11) use the definition of wife (Axiom 3) and the symmetry of marriedTo (Axiom 4). The following two

queries (14 and 15) use the same two axioms and additionally the definition that every sister is a woman (Axiom 1). The last two queries (12 and 13) are similar to the two before, but use the axiom about motherOf (Axiom 2).

Because of the restricted expressivity of OWL 2 QL, all rewriting algorithms only produce unions of conjunctive queries or non-recursive Datalog programs. The expressive power of those queries is equal to that of relational algebra, i.e. essentially SQL. Thus the rewritten Datalog query can also be fully translated to SQL. The translation of Query 9 looks as follows:

```
SELECT abox.instance AS wife, abox.conf() AS prob
FROM abox
WHERE abox.class = 'Wife'
```

The function *conf()* is provided by the probabilistic database and returns the probability for a single row of the result.

Query 10 is translated the following way:

```
SELECT abox.instance AS wife,
       (abox.conf() * married.prob) AS wife_prob
FROM abox
LEFT JOIN (SELECT relation.subject AS person,
                  conf() AS prob
           FROM relation
           WHERE relation.subject = abox.instance AND
                 property = 'marriedTo'
           GROUP BY relation.subject) married
    ON married.person = abox.instance
WHERE abox.class = 'Woman'
```

For the sake of brevity I omit the rest of the queries. They are constructed in a way similar to Query 10.

The last query also shows how the probabilities in a single query are aggregated. I assume that all atoms of a query are mutually independent and thus their probabilities for the individual answers can simply be multiplied to compute that of the final result. For the given example, this assumptions is valid. However, it is easy to formulate a query that contains atoms with dependencies, for example $q(?child) :- motherOf(?x, ?child)$, $Woman(?x)$. Clearly, $motherOf(?x, ?child)$ depends on $Woman(?x)$, as every mother has to be a woman, and thus their probabilities may not simply be multiplied to compute the result. How to handle dependencies in detail is part of my current work (cf. Section 4).

After translating all queries, I concatenate them using `UNION ALL` and send the resulting single query to the probabilistic database. For this small example it will return the following result:

```
 wife          | conf
=====================
 Alice         | 0.240
 Alice         | 0.285
```

For each row of a result, there exists an explanation (cf. [KPHS07]) why it is part thereof. A single explanation contains all axioms and assertions involved in the computation of one result row. The explanations for the results above are the following:

$$\begin{gathered}\exists marriedTo \sqcap Woman \equiv Wife \\ \exists sisterOf \sqsubseteq Woman \\ marriedTo(Alice, Bob), [0.30] \\ sisterOf(Alice, Carol), [0.80]\end{gathered} \tag{16}$$

$$\begin{gathered}Wife \equiv \exists marriedTo \sqcap Woman \\ \exists motherOf \sqsubseteq Woman \\ marriedTo(Alice, Bob), [0.30] \\ motherOf(Alice, Dave), [0.95]\end{gathered} \tag{17}$$

In the case of Alice, there are two different explanations for her being a wife, each with its own probability. When aggregating those rows, it is again important to consider if they are independent or not. For independent rows, I can simply multiply the complementary probability of each row and use the complementary probability of this product as the resulting probability of the aggregation. In this case, however, the two results are dependent because their explanations both contain the assertion $marriedTo(Alice, Bob)$, $[0.30]$. In every world $marriedTo(Alice, Bob), [0.30]$ has to be true, however there are three possible combinations for $sisterOf(Alice, Carol), [0.80]$ and $motherOf(Alice, Dave), [0.95]$ for which Alice is a wife:

$$\begin{gathered}P(marriedTo(Alice, Bob)) = 0.30 \\ P(sisterOf(Alice, Carol)) = 0.80 \\ P(motherOf(Alice, Dave)) = 0.95\end{gathered} \tag{18}$$

$$\begin{gathered}P(marriedTo(Alice, Bob)) = 0.30 \\ P(sisterOf(Alice, Carol)) = 0.80 \\ P(\neg motherOf(Alice, Dave)) = 0.05\end{gathered} \tag{19}$$

$$\begin{gathered}P(marriedTo(Alice, Bob)) = 0.30 \\ P(\neg sisterOf(Alice, Carol)) = 0.20 \\ P(motherOf(Alice, Dave)) = 0.95\end{gathered} \tag{20}$$

Calculating the probability for each of this worlds by multiplying the individual atoms and summing them up gives the following final result:

```
 wife          | conf
=====================
 Alice         | 0.297
```

I plan to use the probabilistic database system MayBMS [HAKO09]. MayBMS is a mature and stable implementation that extends PostgreSQL. It defines its own query language which is an extension of SQL with uncertainty-aware constructs. MayBMS offers a solid and scalable foundation for my approach.

To store an ABox inside MayBMS, there is either the possibility to specifically create a schema for a given ontology, i.e. tables for every property, class, etc., or to develop a generic schema that can store any ontology. If the schema is tailored specifically for an ontology, this could provide performance benefits. However, for the framework to be as flexible as possible I will create a schema that is able to store arbitrary ontologies and probabilistic information.

It is an open question how to describe uncertain information in an ontology. To my knowledge there is no standard or recommendation that describes how this should be done. RDF and OWL offer the possibility for reification (statements about another statement), but this construct is cumbersome, hard to read and understand, and badly supported in editors. There was an incubator group, URW3-XG[9], at W3C but no working group was established to follow up on their work and propose a recommendation.

Furthermore, there are several possibilities which constructs of the ontology are allowed to be probabilistic. Technically, probabilistic information can be attached to either ABox elements only, TBox elements only, or both. Supporting uncertainty in the ABox is the easiest part, as the probabilistic database already supports this and does most of its handling. The rest of the computation can be done inside the query as shown in the example. Uncertainty in the TBox, however, needs to be incorporated into the query rewriting step, which will be more complicated. An example for uncertainty in the TBox is Axiom 5. It adds another possibility for Alice to be married. Probabilistic TBox axioms need special handling during the rewriting as every query derived using that axiom is directly influenced by its probability. In the beginning of my work, I will solely focus on probabilistic information in the ABox.

4 Current Status

So far, I implemented the translation step from SPARQL to Datalog for simple queries, e.g. without aggregates, optional graph patterns, sorting, etc. This enables me to pose simple SPARQL queries to IQAROS and get a rewritten query in Datalog.

Currently, I am detailing the theoretical work on how to aggregate the probabilities after the results are returned from the database. It is only valid to multiply the probabilities, or respectively the complementary probabilities, if all the explanations for a result are mutually independent. The assumptions used during Section 3 need further considerations,

[9] http://www.w3.org/2005/Incubator/urw3/

especially when extending uncertainty to the TBox.

My next steps are the following:

1. Defining a mechanism to include uncertainty into an ontology. As a first prototype, I will use annotations which are already available and can be used on almost all constructs.
2. Creating a database schema to store the ontology. In its first version, as a proof of concept, this schema will contain class assertions and object property assertions only. However, it can be easily extended later on. I plan to keep the TBox as a separate file and not to store it in the database. This way it will be easier to make changes with an editor of choice, e.g. Protégé[10].
3. Implementing the translation from Datalog to SQL.
4. The last step towards a working prototype is the aggregation of the results returned from the database. Here, the final probabilities are computed and can then be displayed to the user.

I plan to finish these goals within the next few months, my long term goals for the p.h.d. are stated in the section future work.

5 Future Work

There are 4 directions in which I plan to continue to work after reaching my midterm goals: Extending the framework, benchmarking its performance, and using it during a case study.

- On the one side I will extend the notion of uncertainty to the TBox and allow the incorporation of imprecise information. For this, I will have to adapt the query rewriting algorithm within IQAROS to account for probabilities on axioms. On the other side, I want to implement support for additional constructs of SPARQL, e.g. unions, optional graph patterns, or aggregates.
- Together with the theoretical work on the handling of uncertainty in the reasoning step, it is also interesting to investigate the influence of the closed or open world assumption on the query results.
- Furthermore, I want to test my framework for performance and scalability. One possibility for this is to use one of the established benchmark frameworks for Semantic Web reasoning, e.g. LUBM[11], SP^{2}Bench[12], or BSBM[13]. Those benchmarks all use generators to synthetically generate knowledge bases. These generators could be adapted to also create imprecise information.

[10] http://protege.stanford.edu/
[11] http://swat.cse.lehigh.edu/projects/lubm/
[12] http://dbis.informatik.uni-freiburg.de/forschung/projekte/SP2B/
[13] http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

- Additionally to synthetic benchmarks, I want to apply my framework to riskmanagement as an alternative approach to markov logic networks, which we are currently using in a project concerned with this topic [vSOS14].
- I will further distinguish my work from the approaches used in probabilistic deductive databases. A deeper analysis of the work of Hernich et al. [HKLG13] about the relation of Datalog and description logics is also needed.

Acknowledgements

I want to thank my supervisor Christian Meilicke for his support and the fruitful discussions about the planned framework and my approach. Additionally, I also want to thank the reviewers for their valuable comments.

References

[CDL05] Diego Calvanese, Giuseppe De Giacomo, and Domenico Lembo. DL-Lite: Tractable description logics for ontologies. *AAAI*, 5:602–607, 2005.

[CDL+11] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The Mastro system for ontology-based data access. *Semantic Web*, 2:43–53, 2011.

[GW09] Nicola Guarino and Christopher A. Welty. An overview of OntoClean. *Handbook on ontologies*, pages 1–20, 2009.

[HAKO09] Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. MayBMS: a probabilistic database management system. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 1071–1074. ACM Press, 2009.

[HKLG13] André Hernich, Clemens Kupke, Thomas Lukasiewicz, and Georg Gottlob. Well-founded semantics for extended datalog and ontological reasoning. In *Proceedings of the 32nd symposium on Principles of database systems*, pages 225–236, New York, New York, USA, 2013. ACM Press.

[Jen96] Finn V. Jensen. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996.

[JS12] Abhay Jha and Dan Suciu. Probabilistic databases with MarkoViews. *Proceedings of the VLDB Endowment*, 5(11):1160–1171, 2012.

[KBP+12] Aditya Kalyanpur, Branimir K. Boguraev, Siddharth Patwardhan, J. William Murdock, Adam Lally, Christopher A. Welty, John M. Prager, Bonaventura Coppola, Achille Fokoue-Nkoutche, Lei Zhang, Yue Pan, and Zhao Ming Qui. Structured data and inference in DeepQA. *IBM Journal of Research and Development*, 56(3):351—-364, 2012.

[KPHS07] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding all justifications of OWL DL entailments. *The Semantic Web*, pages 267—-280, 2007.

[KTD07] Angelika Kimmig, Hannu Toivonen, and Luc De Raedt. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In Rajeev Sangal, Harish Mehta, and R K Bagga, editors, *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, volume 7, pages 2462–2467, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[LS94] Laks V. S. Lakshmanan and Fereidoon Sadri. Probabilistic Deductive Databases. *SLP*, pages 254–268, June 1994.

[PP12] Heiko Paulheim and Jeff Z. Pan. Why the semantic web should become more imprecise. *What will the Semantic Web look like*, 2012.

[RD06] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, January 2006.

[SO13] Joerg Schoenfisch and Jens Ortmann. YARR!: Yet Another Rewriting Reasoner. *CEUR Workshop Proceedings*, 1015:19–25, 2013.

[SS08] Markus Stocker and Michael Smith. Owlgres: A Scalable OWL Reasoner. *OWLED*, 432, 2008.

[TDD+13] Martin Theobald, Luc De Raedt, Maximilian Dylla, Angelika Kimmig, and Iris Miliaraki. 10 years of probabilistic querying-what next? *Advances in Databases and Information Systems*, pages 1–13, 2013.

[vSOS14] Janno von Stülpnagel, Jens Ortmann, and Joerg Schoenfisch. IT Risk Management with Markov Logic Networks. *Proceeedings of the 26th Conference on Advanced Information Systems Engineering (CAiSE 2014)*, to appear, 2014.

[VSS12] Tassos Venetis, Giorgos Stoilos, and Giorgos Stamou. Incremental query rewriting for OWL 2 QL. *complexity*, 7:5, 2012.

[Wid04] Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. *Technical Report*, pages 1–22, 2004.

[WMGH08] Daisy Zhe Wang, Eirinaios Michelakis, Minos Garofalakisy, and Joseph M. Hellerstein. BayesStore: Managing large, uncertain data repositories with probabilistic graphical models. In *Proceedings of the VLDB Endowment*, volume 1, pages 340–351, 2008.