Undefined 0 (0) 1
IOS Press

An Infrastructure for Probabilistic Reasoning with Web Ontologies

Jakob Huber [a] Mathias Niepert [b] Jan Noessner [c] Christian Meilicke [a] Heiner Stuckenschmidt [a]
[a] *Data- and Web Science Group, University of Mannheim*
[b] *Department of Computer Science, University of Washington*
[c] *Nuance Communications, Sunnyvale, USA*

Abstract. We present an infrastructure for probabilistic reasoning with ontologies that is based on our Markov logic engine RockIt. Markov logic is a template language that combines first-order logic with log-linear graphical models. We show how to translate OWL-EL as well as RDF schema to Markov logic and how to use RockIt for applying MAP inference on the given set of formulas. The resulting system is an infrastructure for log linear logics that can be used for probabilistic reasoning with both extended OWL-EL and RDF schema. We describe our system and illustrate its benefits by presenting two application scenarios. These scenarios are ontology matching, and knowledge base verification, with a special focus on temporal reasoning. Our results indicate that our system, which is based on a well-founded probabilistic semantics, is capable of solving relevant problems as good as or better than state of the art systems that have specifically been designed for the respective problem.

Keywords: Ontologies, Reasoning, Markov Logic, RDF Schema, Log-linear Logics

1. Motivation

Originally, the idea of the Semantic Web was built on formal ontologies and logical reasoning leading to the development of description logic based ontology languages, in particular the Web Ontology Language OWL. The advantages of logical reasoning on the web has been argued by many researchers in terms of verifying information models and extending query results to implicit information. While in particular, the development of light-weight ontology languages and the use of database techniques for ontology-based data access has given another boost to the use of logical models (e.g. [7]), it has also become more and more clear that logical reasoning alone is not enough for many Semantic Web applications. In particular, use cases that require a combined processing of structured and unstructured data (i.e. free text) can often not be adequately handled by logical methods alone. On the other hand, abandoning logical reasoning and completely relying on machine learning techniques for handling unstructured data seems like throwing out the child with the bathwater. Therefore, research on the (Semantic) Web has focused on methods that combine structured representations with statistical inference. One possible approach is the extension of machine learning methods to complex structural features [10], the other is to extend logical reasoning in Semantic Web languages towards statistical inference. In this paper, we present a software infrastructure that implements the second way - an extension of logical reasoning in Semantic Web languages with statistical inference [27]. In particular, we focus on the task of computing the most probable consistent ontology from a set of possibly inconsistent axioms some of which are not certain and have a weight attached that can be used to resolve inconsistencies given a probabilistic interpretation. This corresponds to the maximum a posteriori (MAP) inference in statistical relational models as opposed to marginal inference, which aims at computing the probability of a certain assertion given evidence in terms of a knowledge base. Our approach is based on an encoding of description logics into Markov logic networks that enables reasoning about description logics that support consequence-driven reasoning.

Related Work The extension of web languages towards statistical inference has been addressed by a number of researchers leading to various extensions of the OWL language with notions of probability using different approaches including non-monotonic reasoning, probabilistic logic programming, Bayesian networks and Markov logic. The probabilistic description logic $\mathcal{PSROIQ}$ [26] is an expressive description logic that combines conditional probabilities about concepts and statements expressing uncertain knowledge about instances. The logic supports probabilistic knowledge base consistency checking and lexicographic entailment. Both are probabilistic versions of non-monotonic reasoning where consistency can be achieved by preferring more specific assertions over less specific ones and calculating upper and lower bounds for the probabilities of the resulting statements. The corresponding methods have been implemented in the PRONTO system [22]. Riguzzi and others adapt the semantics of probabilistic logic programs as introduced in [11] to description logics defining the DISPONTE approach for probabilistic ontologies. DISPONTE can answer queries over probabilistic ontologies using binary decision diagrams. The corresponding methods have been implemented in the BUNDLE reasoner [33,5]. Both reasoners have been integrated into the Pellet description logic system. DISPONTE clearly aims at computing probabilistic query answers which corresponds to marginal inference rather than MAP inference. PR-OWL [8,9] is a language based on the multi-entity Bayesian network (MEBN) logic [24]. MEBNs specify a first-order language for modeling probabilistic knowledge bases as parametrized fragments of Bayesian networks. PR-OWL adds new elements to the OWL standard which enables to create MEBNs as part of an ontological model and thus supporting probabilistic reasoning, although providing only a limited integration of logical and probabilistic reasoning. Parts of the PR-OWL language were implemented in the tool UNBBAYES-MEBN[dCLC+08] which provides a GUI and a translation of MEBN to classical Bayesian networks. Based on the translation to Bayesian networks, PR-OWL can in principle compute both MAP and marginal inference. To the best of our knowledge, INCERTO[1] is the only reasoner which uses Markov logic for probabilistic description logics. It directly translates the description logic axioms to Markov logic so that concepts correspond to unary predicates, roles correspond to binary predicates, and individuals to constants. The description logic axioms are translated to first-order logic formulas. The objective of INCERTO is to learn the weights of axioms through the analysis of individuals. Furthermore, they provide exact and approximate marginal inference. However, computing the most probable coherent ontology is not supported. Due to the direct translation of the logical model into Markov logic and the implicit closed world assumption introduced on the Markov logic level, it is unclear whether the system can guarantee complete reasoning with respect to the description logic used.

[1] https://code.google.com/p/incerto/

In this paper, we first provide a comprehensive overview of our Markov logic engine RockIt, which to be best of our knowledge currently is the most performant and scalable Markov logic reasoner with respect to MAP inference. We give a brief introduction to Markov logic as an underlying formalism and explain how RockIt computes the most probable model by an efficient encoding of the problem as a linear integer program (see Section 3). After providing some details about the optimized implementation of RockIt, we introduce the notion of log-linear logics which provides the formal foundation for a sound and complete encoding of description logics and related formalisms into Markov logic, and describe two concrete logics that corresponds to probabilistic versions of OWL-EL and RDFS (see Section 4). We then present successful applications of the approach to relevant Semantic Web problems, in particular knowledge base verification and ontology matching (see Section 5). We close with a discussion of the benefits and limitations of our infrastructure in Section 6.

2. System Architecture

We present a software architecture (see Figure 1) that bundles a series of systems which enable probabilistic reasoning in the context of the Semantic Web. The system is available at `http://executor.informatik.uni-mannheim.de/`. The core of the presented system is the RockIt Markov logic engine. It is specialized for executing maximum a posteriori (MAP) inference in Markov Logic Networks. For this purpose, it takes advantage of a database (MySQL) and an integer linear program solver (Gurobi) (see Section 3). Based on this core system, we provide interfaces for specific probabilistic reasoning tasks. In par-

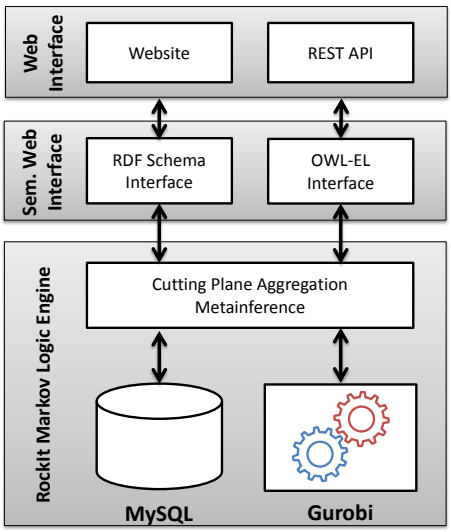

Fig. 1. Overview of the system architecture

ticular, the "Semantic Web Interface" supports probabilistic reasoning for OWL-EL and RDFS (see Section 4). The web interface has the purpose to bundle different reasoning interfaces and to make them available in a convenient way, i.e., through a user interface and an API.

The reasoners are made available by a web interface that allows executing different tasks. Hence, they can be used without installing them on a local machine which would involve setting up a database management system as well as an integer linear program solver. We offer a web site and REST interfaces to access the services. The web site provides a description, usage examples of the different reasoners and forms to upload files and to initiate the reasoning process. Additionally, the REST interfaces support the most important features and allow developers of other applications to use our services and to integrate them in their systems. As the infrastructure can be accessed by multiple users at the same time, we implemented a waiting line that ensures that the system processes at most one reasoning task at a time. This is necessary as it decreases the chance of overloading but it also makes the runtimes comparable.

The system creates a unique identification key for each added process that can be used to identify a specific process in the process database. For each process we keep track of relevant statistics and useful information. Therefore, we record the time and the date when a process was added to the waiting list, when the execution was started and when it terminated. Moreover, we

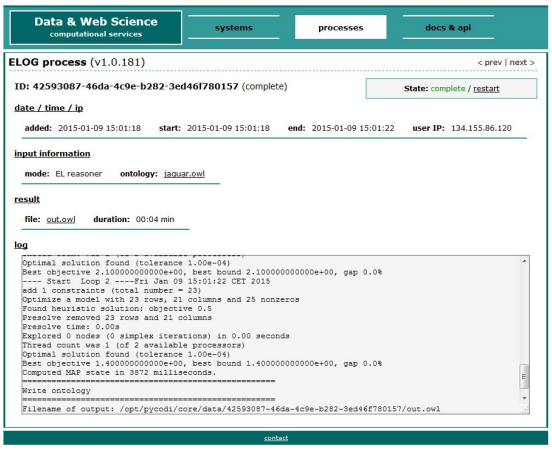

Fig. 2. View of a terminated process.

store the input information which includes the selected reasoning method as well as the chosen parameter settings and input files (see Figure 2). We also present the console output of the underlying reasoning system during the execution in real-time which helps to keep track of the execution of a process.

The web interface is written in Python and uses the web application framework "Pylons"[2]. Its internal architecture allows integrating other (reasoning) system. In fact, it is possible to add any system that is compatible with the Linux distribution Debian.

3. The RockIt Engine

The foundation of our system is the RockIt reasoning engine [30], a state of the art reasoning engine for Markov logic networks. RockIt is specialized on performing maximum a posteriori (MAP) inference. Each MAP query corresponds to an optimization problem with linear constraints and a linear objective function and, hence, RockIt formulates and solves the problem as an instance of integer linear programming (ILP). This is done in several iterations where the novel cutting plane aggregation approach (CPA) is tightly integrated with cutting plane inference (CPI) which is a meta-algorithm operating between the grounding algorithm and the ILP solver [32]. Instead of immediately adding one constraint for each ground formula to the ILP formulation, the ILP is initially formulated so as to enforce the given evidence to hold in any solu-

[2] `www.pylonsproject.org/projects/pylons-framework`

tion. Based on the solution of this more compact ILP one determines the violated constraints, adds these to the ILP, and resolves. This process is repeated until no constraints are violated by an intermediate solution. In the following, we provide a brief introduction to Markov logic as well as RockIt's approach to efficient maximum a posteriori (MAP) inference.

3.1. Markov Logic

Markov logic is a first-order template language combining first-order logic with log-linear graphical models. We first review function-free first-order logic [14]. Here, a term is either a constant or a variable. An atom $p(t_1, ..., t_n)$ consists of a predicate p/n of arity n followed by n terms t_i. A literal ℓ is an atom a or its negation $\neg a$. A clause is a disjunction $\ell_1 \vee ... \vee \ell_k$ of literals. The variables in clauses are always assumed to be universally quantified. The Herbrand base $\mathcal{H}$ is the set of all possible ground (instantiated) atoms. Every subset of the Herbrand base is a Herbrand interpretation.

A Markov logic network $\mathcal{M}$ is a finite set of pairs $(F_i, w_i), 1 \leq i \leq n$, where each F_i is a clause in function-free first-order logic and $w_i \in \mathbb{R}$. Together with a finite set of constants $C = \{c_1, ..., c_n\}$ it defines the ground Markov logic network $\mathcal{M}_\mathcal{C}$ with one binary variable for each grounding of predicates occurring in $\mathcal{M}$ and one feature for each grounding of formulas in $\mathcal{M}$ with feature weight w_i. Hence, a Markov logic network defines a log-linear probability distribution over Herbrand interpretations (possible worlds)

$$P(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\mathbf{x})\right) \tag{1}$$

where $n_i(\mathbf{x})$ is the number of satisfied groundings of clause F_i in the possible world $\mathbf{x}$ and Z is a normalization constant.

In order to answer a MAP query given evidence $\mathbf{E} = \mathbf{e}$, one has to solve the maximization problem

$$\arg\max_{\mathbf{x}} \; P(\mathbf{X} = \mathbf{x} \mid \mathbf{E} = \mathbf{e})$$

where the maximization is performed over possible worlds (Herbrand interpretations) $\mathbf{x}$ compatible with the evidence.

3.2. ILP Formulation of Markov Logic Networks

Since we are employing cutting plane inference (CPI), ROCKIT retrieves in each iteration the ground clauses violated by the current solution. Hence, in each iteration of the algorithm, ROCKIT maintains a set of ground clauses $\mathcal{G}$ that have to be translated to an ILP instance. RockIt associates one binary ILP variable x_ℓ with each ground atom ℓ occurring in some $g \in \mathcal{G}$. For a ground clause $g \in \mathcal{G}$ let $L^+(g)$ be the set of ground atoms occurring unnegated in g and $L^-(g)$ be the set of ground atoms occurring negated in g. Now, we encode the given evidence by introducing linear constraints of the form $x_\ell \leq 0$ or $x_\ell \geq 1$ depending on whether the evidence sets the corresponding ground atom ℓ to false or true. For every ground clause $g \in \mathcal{G}$ with weight $w > 0,\ w \in \mathbb{R}$, we add a novel binary variable z_g and the following constraint to the ILP:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \geq z_g.$$

Please note that if any of the ground atoms ℓ in the ground clause is set to false (true) by the given evidence, we do not include it in the linear constraint.

For every g with weight $w_g < 0,\ w \in \mathbb{R}$, we add a novel binary variable z_g and the following constraint to the ILP:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \leq (|L^+(g)| + |L^-(g)|) z_g.$$

For every g with weight $w_g = \infty$, that is, a hard clause, we add the following linear constraint to the ILP:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \geq 1$$

Finally, the objective of the ILP is:

$$\max \sum_{g \in \mathcal{G}} w_g z_g,$$

where we sum over weighted ground clauses only, w_g is the weight of g, and $z_g \in \{0, 1\}$ is the binary variable previously associated with ground clause g. We compute a MAP state by solving the ILP whose solution corresponds one-to-one to a MAP state $\mathbf{x}$ where $x_i = \texttt{true}$ if the corresponding ILP variable is 1 and $x_i = \texttt{false}$ otherwise.

3.3. Constraint Aggregation

The ILP defined in the previous section can be simplified by aggregating groups of similar constraints.

Table 1
A set of ground clauses that can be aggregated.

g	ℓ_i	c	w
g_1	$x_1 \vee$	$\neg y_1 \vee y_2$	1.0
g_2	$x_2 \vee$	$\neg y_1 \vee y_2$	1.0
g_3	$\neg x_3 \vee$	$\neg y_1 \vee y_2$	1.0
g_4	$\neg x_4 \vee$	$\neg y_1 \vee y_3$	1.0
g_5	$x_5 \vee$	$\neg y_1$	0.5

$\rightsquigarrow$

ℓ_i	c	w
$x_1 \vee$		
$x_2 \vee$	$\neg y_1 \vee y_2$	1.0
$\neg x_3 \vee$		
$\neg x_4 \vee$	$\neg y_1 \vee y_3$	1.0
$x_5 \vee$	$\neg y_1$	0.5

The resulting ILP has fewer variables, fewer constraints, and its context-specific symmetries are more exposed to the ILP solver's symmetry detection heuristics. We present the general idea implemented in ROCKIT and illustrate it by an example. Formal details can be found in [30].

First we define which subsets G of $\mathcal{G}$ can be aggregated to a single constraint.

Definition 1. *Let $G \subseteq \mathcal{G}$ be a set of n weighted ground clauses and let c be a ground clause. We say that G can be aggregated with respect to c if (a) all ground clauses in G have the same weight and (b) for every $g_i \in G, 1 \leq i \leq |G|$, we have that $g_i = \ell_i \vee c$ where ℓ_i is a (unnegated or negated) literal for each $i, 1 \leq i \leq |G|$.*

Table 1 illustrates an example. The subset $G = \{g_1, g_2, g_3\}$ can be aggregated with respect to $\neg y_1 \vee y_2$, while g_4 and g_5 can be aggregated only as singleton sets, which does not yield in an advantage over the standard way of generating the ILP. The standard way of translating g_1, g_2 and g_3, ignoring the possibility of an aggregation, would result into the following ILP.

$$\textbf{max } 1.0z_1 + 1.0z_2 + 1.0z_3 \textbf{ subject to}$$
$$x_1 + (1 - y_1) + y_2 \geq z_1$$
$$x_2 + (1 - y_1) + y_2 \geq z_2$$
$$(1 - x_3) + (1 - y_1) + y_2 \geq z_3$$

Note that z_1, z_2, and z_3 are binary variables within this ILP. By using instead of that a single integer variable z, all ground clauses in G can be taken into account as a whole within the following ILP.

$$\textbf{max } 1.0z \textbf{ subject to}$$
$$x_1 + x_2 + (1 - x_3) + 3((1 - y_1) + y_2) \geq z$$
$$z \leq 3$$

This translation follows a general pattern for positive weights and a similar pattern for negative weights. As a result of this simplification the number of constraints can be reduced significantly resulting in a significantly better runtime behavior of the system. The general approach is called cutting plane aggregation (CPA).

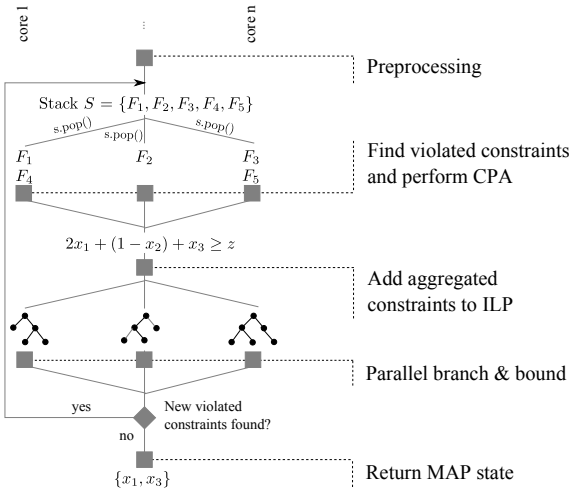

Fig. 3. ROCKIT parallelizes constraint finding, constraint aggregation, and ILP solving.

3.4. Implementation

Figure 3 depicts the computational pipeline of the system. After pre-processing the input MLN and loading it into the relational database system, ROCKIT performs CPI iterations until no new violated constraints are found. The violated constraints are computed with joins in the relational database system where each table stores the predicate groundings of the intermediate solutions. In each CPI iteration, ROCKIT performs CPA on the violated constraints. We can parallelize the aggregation steps by processing each first-order formula in a separate thread. To this end, each first-order formula is initially placed on a stack S. ROCKIT creates one thread per available core and, when idle, makes each of the threads (i) pop a first-order formula from the stack S, (ii) compute the formula's violated groundings, and (iii) perform CPA on these groundings. The aggregated groundings are compiled into ILP constraints and added to the ILP formulation. When the stack S is empty and all threads idle we solve the current ILP in parallel, obtain a solution, and begin the next CPI iteration.

There are different possible strategies for finding the ground clauses c that minimize the number of counting constraints per first-order formula which is required for CPA. This problem can be solved optimally, how-

ever, we implemented a greedy algorithm that only *estimates* the optimal aggregation scheme, since experiments showed that optimal algorithms dominate the ILP solving itself. The algorithm stores, for each first-order clause, the violated groundings of the form $\ell_1 \vee \ldots \vee \ell_n$ in a table with n columns where each column represents one literal position of the clause. For each column k, ROCKIT computes the set of *distinct* rows R_k of the table that results from the projection onto the columns $\{1, ..., n\} \setminus \{k\}$. Let $d = \arg\min_k\{|R_k|\}$. The clause groundings are then aggregated with respect to the rows in R_d.

ROCKIT employs MYSQL's in-memory tables for the computation of violated constraints and the aggregation. Most tables are hash indexed to facilitate highly efficient join processing. We use GUROBI[3] as ROCKIT's internal ILP solver due to its ability to parallelize its branch, bound, and cut algorithm, its remarkable performance on standard ILP benchmarks [23], and its symmetry detection heuristics.

4. The Semantic Web Interface

The ROCKIT system introduced in the previous section is a generic Markov logic engine that does not directly target languages relevant for the Semantic Web community. In this section we show how Semantic Web languages, in particular description logics and RDF Schema based formalisms, can be transformed to Markov logic while preserving their original semantics. This translation approach, to which we refer as log-linear logics, makes the power of the ROCKIT system available for Semantic Web applications.

4.1. Log-Linear Logics

Log-linear (description) logics have been introduced by Niepert and others as a framework for encoding non-trivial logics into probabilistic reasoning using the notion of log-linear models [29]. A log-linear logic knowledge base $\mathcal{KB}$ consists of a deterministic knowledge base $\mathcal{C}^{\mathsf{D}}$ and an uncertain knowledge base $\mathcal{C}^{\mathsf{U}}$. $\mathcal{C}^{\mathsf{D}}$ contains axioms in some logic $\mathcal{L}$ that supports a valid entailment relation $\models_{\mathcal{L}}$ and notion of contradiction $\bot_{\mathcal{L}}$. The uncertain knowledge base is defined as $\mathcal{C}^{\mathsf{U}} = \{(c, w_c)\}$ where c is also an axiom in $\mathcal{L}$ and w_c is a real-valued weight assigned to c. $\mathcal{C}^{\mathsf{D}}$ represents definite knowledge about the world, whereas $\mathcal{C}^{\mathsf{U}}$ contains axioms for which only a degree of confidence is available. The certain knowledge base is assumed to be non-contradictory, i.e. $\mathcal{C}^{\mathsf{D}} \not\models_{\mathcal{L}} \bot_{\mathcal{L}}$. The semantics of a log-linear logic is based on joint probability distributions over the uncertain knowledge base. In particular, the weights of the axioms in $\mathcal{C}^{\mathsf{U}}$ determine a log-linear probability distribution in a similar way as it is the case for Markov logic models introduced earlier.

For a given log-linear knowledge base $(\mathcal{C}^{\mathsf{D}}, \mathcal{C}^{\mathsf{U}})$ and some (certain) knowledge base $\mathcal{C}'$ over the same signature, the probability of $\mathcal{C}'$ is defined as

$$P(\mathcal{C}') = \begin{cases} \frac{1}{Z} \exp\left(\sum_{\{(c,w_c)\in\mathcal{C}^{\mathsf{U}}:\mathcal{C}'\models c\}} w_c\right) & \text{if } \mathcal{C}' \not\models_{\mathcal{L}} \bot_{\mathcal{L}} \\ & \text{and } \mathcal{C}' \models_{\mathcal{L}} \mathcal{C}^{\mathsf{D}}; \\ 0 & \text{otherwise} \end{cases}$$

where Z is the normalization constant of the log-linear probability distribution P.

The notion of a log-linear logic provides a means for interfacing Semantic Web languages with the ROCKIT engine in the following way:

- We define a Markov logic predicate for each type of axiom supported by the language. For this purpose, the axioms in the knowledge base typically have to be normalized in an appropriate way.
- We encode the entailment relation $\models_{\mathcal{L}}$ using Markov logic rules over the predicates representing the different axiom types.
- We define the notion of contradiction $\bot_{\mathcal{L}}$ by adding a set of first order rules that results into an inconsistent set of first-order formulas whenever, with respect to the semantics of $\mathcal{L}$, a logically undesired behavior occurs.

This approach naturally limits the applicability of the framework to languages that can be normalized to a finite set of axiom types and whose entailment relation can be modeled using Markov logic rules. In Semantic Web research such languages have been studied in connection with the notion of consequence-driven reasoning [21], providing us with a set of possible languages we can extend towards probabilistic reasoning. In the following, we discuss two of these languages that we have implemented in our infrastructure.

4.2. The OWL-EL Interface

Computing the MAP state for a log-linear knowledge base that contains OWL-EL axioms requires first

[3] http://www.gurobi.com/

to map the different axioms to a first-order representation. As it has been shown that any OWL-EL knowledge-base can be normalized into an equivalent knowledge base that only contains 6 types of axioms, we can map it into Markov logic predicates using the following set of mapping rules.

$$\begin{aligned} C_1 \sqsubseteq D &\mapsto sub(C_1, D) \\ C_1 \sqcap C_2 \sqsubseteq D &\mapsto int(C_1, C_2, D) \\ C_1 \sqsubseteq \exists r.C_2 &\mapsto rsup(C_1, r, C_2) \\ \exists r.C_1 \sqsubseteq D &\mapsto rsub(C_1, r, D) \\ r \sqsubseteq s &\mapsto psub(r, s) \\ r_1 \circ r_2 \sqsubseteq r_3 &\mapsto pcom(r_1, r_2, r_3). \end{aligned}$$

The first-order predicates in this listing are typed, meaning that $r, s, r_i, (1 \leq i \leq 3)$, are role names, C_1, C_2 basic concept descriptions, and D basic concept descriptions or the bottom concept. Note that axioms involving complex concept descriptions can be transformed into a normalized representation by applying a finite set of normalization rules introducing new concept and role names [3].

The translation of $\mathcal{C}^{\mathsf{D}}$ and $\mathcal{C}^{\mathsf{U}}$ results in unweighted (hard) and weighted first order formulas. The hard formulas are used, together with the completion rules described in the following paragraph, to decide whether for a potential MAP state $\mathcal{C}'$ we have $\mathcal{C}' \not\models_{EL} \bot_{EL}$ and $\mathcal{C}' \models_{EL} \mathcal{C}^{\mathsf{D}}$. To our knowledge there exists no commonly accepted proposal related to adding weights or probabilities to OWL axioms. In our implementation we are using annotation properties to add weights to axioms.

The completion rules of the formalism, in this case OWL-EL, which corresponds to the description logics $\mathcal{EL}^{++}$, are listed in Table 2. By adding these rules as constraints to the Markov logic formalization, we are able to support complete reasoning mechanisms for $\mathcal{EL}^{++}$, i.e., we re-define $\models_{EL}$ by adding the respective reasoning rules as first-order formulas in our Markov logic formalization. Given a concrete reasoning problem, the resulting MAP state will always contain the most probably non contradictory subset of $\mathcal{C}^{\mathsf{U}}$ that entails the previously known axioms $\mathcal{C}^{\mathsf{D}}$.

Note that rule F_9 does not belong to the completion rules for $\mathcal{EL}^{++}$. This rule takes the notion of incoherence into account. An incoherent ontology is an ontology that contains an unsatisfiable concept, i.e., a concept that is subsumed by $\bot$. Usually, an unsatisfiable concept indicates that the ontology contains a contradictory set of axioms. An incoherent ontology is not necessarily inconsistent, Thus, we added rule F_9 which allows us to extend the notion of contradiction $\bot_{EL}$ from inconsistency to incoherence.

Table 2
Completion rules for OWL-EL.

F_1	$\forall c : sub(c, c)$
F_2	$\forall c : sub(c, \top)$
F_3	$\forall c, c', d : sub(c, c') \wedge sub(c', d) \Rightarrow sub(c, d)$
F_4	$\forall c, c_1, c_2, d : sub(c, c_1) \wedge sub(c, c_2) \wedge int(c_1, c_2, d) \Rightarrow sub(c, d)$
F_5	$\forall c, c', r, d : sub(c, c') \wedge rsup(c', r, d) \Rightarrow rsup(c, r, d)$
F_6	$\forall c, r, d, d', e : rsup(c, r, d) \wedge sub(d, d') \wedge rsub(d', r, e) \Rightarrow sub(c, e)$
F_7	$\forall c, r, d, s : rsup(c, r, d) \wedge psub(r, s) \Rightarrow rsup(c, s, d)$
F_8	$\forall c, r_1, r_2, r_3, d, e : rsup(c, r_1, d) \wedge rsup(d, r_2, e) \wedge pcom(r_1, r_2, r_3) \Rightarrow rsup(c, r_3, e)$
F_9	$\forall c : \neg sub(c, \bot)$

For more technical details on applying the principle of log-linear logic to OWL-EL, we refer the reader to [29]. The reasoner that we have implemented following the described approach is called ELog.[4]

4.3. The RDF Schema Interface

While the OWL-EL interface was mainly designed for terminological reasoning, the focus of the RDFS interface is rather on reasoning tasks related to the A-Box. A RDF [18] knowledge base contains statements of the form $(\texttt{subject}, \texttt{predicate}, \texttt{object})$. Such statements can be seen as the only axiom type present in an RDF Schema knowledge base. Thus, we only need to define one general first-order logic predicate to which all RDF statements can be transformed:

$$(\mathtt{s}, \mathtt{p}, \mathtt{o}) \mapsto triple(s, p, o)$$

We decided to use this modeling as it is flexible and covers all possible RDF statements. In order to attach weights to RDF statements, we rely in our implementation on the concept of reification and annotation properties. Our system relies on the RDF(S) entailment rules [6] as a standard rule set. Using the predicate $triple(s, p, o)$, we define the entailment relation $\models_{RDFS}$ by mapping the RDF(S) entailment rules to first-order formulas. For instance, we state the rule

[4] http://executor.informatik.uni-mannheim.de/systems/elog/

rdfs11 which expresses the transitivity of the property *rdfs:subclassOf* as follows:

$$triple(x, \textit{rdfs:subClassOf}, y) \wedge$$
$$triple(y, \textit{rdfs:subClassOf}, z) \Rightarrow$$
$$triple(x, \textit{rdfs:subClassOf}, z).$$

We define formulas for the other RDF(S) completion rules in the same way. Moreover, it is possible to extend the basic set of completion rules by defining additional rules and constraints in order to cover a wide range of application scenarios. In partiuclar, pure RDF Schema reasoning does not have a notion of contradiction. We therefore introduce notions of contradiction by adding special rules. For example, if one might want the reasoner to cover the semantics of the property *owl:disjointWith*, the rule set can be extended:

$$triple(c1, owl:disjointWith, c2) \wedge$$
$$triple(x, rdf:type, c1) \Rightarrow$$
$$\neg triple(x, rdf:type, c2).$$

This is only one of the required rules to cover the semantics of `owl:disjointWith`. Our formalism can be extended by such rules depending on the requirements of a specific use case. By executing the MAP inference, the reasoner determines for each potential MAP state $\mathcal{C}'$ if $\mathcal{C}' \not\models_{\mathcal{RDFS}^+} \bot_{\mathcal{RDFS}^+}$ and $\mathcal{C}' \models_{\mathcal{RDFS}^+} \mathcal{C}^{\mathsf{D}}$ holds. Thereby, the system resolves not only the detected inconsistencies but is also able to infer new facts. Hence, the MAP state corresponds to the most probable consistent subset of facts and all assertions that can be entailed from that subset.

We adapted our reasoning system to the special case of probabilistic temporal reasoning. Additionally to adding weights to RDF statements, it is possible to assign a temporal interval, which consists of a start point and end point, resulting in statements of the form $\langle triple(s, p, o)[start, end]\rangle$. The interval expresses the validity time of a fact [16,17]. Considering the temporal relations of events, it is possible to extend the set of constraints in order to detect inconsistencies. For this purpose, the reasoning system incorporates the temporal predicates (e.g. before, during, ...) of Allen's interval algebra [1] that can be used to express such constraints. The proposed approach is implemented by the reasoner T-RDFS-LOG.[5]

[5] `http://executor.informatik.uni-mannheim.de/systems/t-rdfs-log/`

5. Applications

In [31] we have applied the OWL-EL interface to improve automatically generated ontology alignments, while in [19] we have shown how to use the methods available via the RDF schema interface to debug a probabilistic temporal knowledge base. Within the following two sections we summarize these application scenarios and the experimental results to illustrate the general applicability of our framework.

5.1. Ontology Matching

Ontology matching is concerned with the task of constructing an alignment $\mathcal{A}$ between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. An alignment is a set of mappings usually interpreted as equivalence axioms between concepts and properties of $\mathcal{O}_1$ and $\mathcal{O}_2$. Most matching systems annotate mappings with a confidence value that expresses the trust in the correctness of the generated mapping. Thus, we have a perfect setting for applying ELog: The alignment $\mathcal{A}$ can be interpreted as a set of uncertain, weighted equivalence axioms $\mathcal{C}^{\mathsf{U}}$, while $\mathcal{O}_1 \cup \mathcal{O}_2$ comprises the certain axioms $\mathcal{C}^{\mathsf{D}}$.

Some of the concepts in $\mathcal{O}_1$ or $\mathcal{O}_2$ might become unsatisfiable due to the axioms of $\mathcal{A}$. $\mathcal{A}$ is called an incoherent alignment if this happens. The following is an example for an incoherent alignment.

$$\mathcal{O}_1 = \{\texttt{Jaguar}_1 \sqsubseteq \texttt{Cat}_1, \texttt{Cat}_1 \sqsubseteq \texttt{Animal}_1\},$$
$$\mathcal{O}_2 = \{\texttt{Jaguar}_2 \sqsubseteq \texttt{Brand}_2, \texttt{Animal}_2 \sqsubseteq \neg\texttt{Brand}_2\}$$
$$\mathcal{A} = \{\langle \texttt{Jaguar}_1 \equiv \texttt{Jaguar}_2, 0.9\rangle,$$
$$\langle \texttt{Animal}_1 \equiv \texttt{Animal}_2, 0.95\rangle\}$$

In this example the concepts $\texttt{Jaguar}_1$ and $\texttt{Jaguar}_2$ are unsatisfiable in the merged ontology. There are several possible ways to resolve this incoherence by removing different subsets from $\mathcal{A}$. Applying the formalism introduced in Section 4.2 results in computing the coherent subset $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \{\texttt{Animal}_1 \equiv \texttt{Animal}_2\}$ as MAP state. The alignment $\{\texttt{Animal}_1 \equiv \texttt{Animal}_2\}$ is thus the must probable alignment given the information available in $\mathcal{O}_1$ and $\mathcal{O}_2$.

Several systems and algorithms have been developed to deal with the problem of debugging incoherent alignments. We present experiments comparing our generic approach to two of these system which are LogMap [20] and Alcomo [28]. LogMap is a matching system that consists beside other components of an algorithm for generating (nearly) coherent alignments.

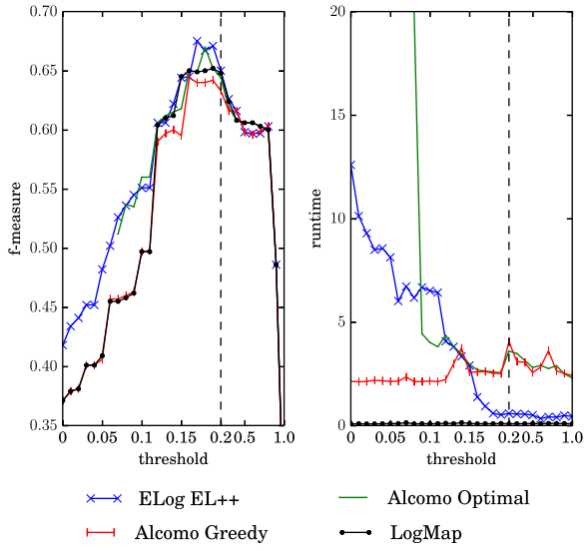

Fig. 4. Mapping quality and runtime for different debugging approaches.

Alcomo is a system that has specifically been developed for the purpose of debugging ontology alignments. It can be used with different settings that differ with respect to completeness and runtime efficiency.

We have used the testcases of the well known OAEI conference track together with the alignments submitted by the OAEI 2013 participants [15]. We merged these alignments (details can be found in [31]) and conducted experiments for different thresholds. For low thresholds the input alignments were large and highly incoherent while high thresholds resulted in precise and small input alignments. We applied our generic approach as well as LogMap and Alcomo (in two settings) to these alignments. The results of our experiments are depicted in Figure 4.

On the left side of the figure we have shown the F-measure of the debugged alignments for different input thresholds. ELog achieves for all thresholds the best f-measure. In some cases the results of the other systems are similar, however, ELog outperforms the other systems with respect to the threshold range that results into the best f-measure. Only Alcomo (optimal setting) can achieve the same result in one case. ELog can debug the given alignments in acceptable time (see right side of the figure), however, Alcomo (greedy setting) and LogMap are significantly faster for large input alignments. This is mainly related to the fact that both systems do not solve the underlying optimization problem, but generate an approximate solution by applying a greedy algorithm. This explains also why both algorithms generate results that are slightly worse compared to the results of ELog. Only Alcomo in the optimal setting, using a search tree to find the optimal solution, is an exception. However, Alcomo (optimal setting) does not terminate for larger testcases. We conclude that our generic approach can be applied successfully to the problem of alignment debugging and, even more, outperforms existing systems that have specifically been designed for the given problem.

5.2. *Knowledge Base Verification*

Knowledge base verification is concerned with the task of identifying erroneous statements in a knowledge base. In the context of a probabilistic knowledge base, this scenario is essentially an optimization problem. This means that a reasoner must identify the most probable consistent subset of facts given a set of constraints. This corresponds to the computation of the MAP state. The following example illustrates a inconsistent knowledge base $\mathcal{F} = \{f_1, f_2, f_3\}$:

$$f_1 = \langle \texttt{triple}(\texttt{Einstein}, \texttt{birthYear}, \texttt{1879}), 0.5 \rangle$$

$$f_2 = \langle \texttt{triple}(\texttt{Einstein}, \texttt{birthYear}, \texttt{1955}), 1.0 \rangle$$

$$f_3 = \langle \texttt{triple}(\texttt{Einstein}, \texttt{deathYear}, \texttt{1955}), 0.8 \rangle$$

For this knowledge base, we define the following constraint set $\mathcal{R} = \{r_1, r_2\}$:

$$r_1 = \texttt{A person has at most one birth date.}$$

$$r_2 = \texttt{The birth of a person happens before her death.}$$

Given the constraint set $\mathcal{R}$, it is possible to detect inconsistencies in $\mathcal{F}$. r_1 causes a clash between the facts f_1 and f_2 while r_2 causes a clash between the facts f_2 and f_3. Hence, f_2 has to be removed in order to obtain a consistent knowledge base despite having the highest weight of all facts.

Different approaches have been proposed to solve such problems. In [4,2] OWL 2.0 is extended in order to enable temporal reasoning for supporting temporal queries. The authors define SWRL rules that are compatible with a reasoner that supports DL-safe rules in order to detect inconsistencies. However, their system can only detect if a knowledge base is consistent but cannot resolve the existing conflicts. [34,13,12] proposed different approaches to resolve temporal conflicts at query time. In particular, they define temporal constraint as Datalog rules. However, these approaches do not incorporate terminological knowledge while re-

Table 3

Precision (P), Recall (R) and F-measure (F) for the repaired dataset. We apply the RDFS reasoner to the input datasets that contain different fractions of erroneous statements. The reasoner removes those statements with a high precision as the loss of recall is reasonable with respect to the gain of precision.

Input	Repaired Dataset			ΔF
P	P	R	F	
0.99	1.00	1.00	1.00	0.002
0.91	0.97	0.98	0.97	0.022
0.80	0.92	0.96	0.94	0.050
0.67	0.84	0.93	0.88	0.084
0.57	0.78	0.90	0.83	0.106
0.50	0.72	0.87	0.79	0.119

solving the conflicts and do also not support weighted constraints.

Contrary to this, our reasoner T-RDFS-LOG (see Section 4.3) is well-suited for verifying probabilistic temporal knowledge bases as it allows extending a standard rule set, i.e., the RDFS entailment rules, by defining domain-specific constraints. Hence, it is a flexible system that can be adapted to many domains. Based on rules and constraints, the reasoner is able to cleanse a knowledge base by removing the statements that do not belong to the MAP state.

In [19] we have evaluated this application scenario in an artificial setting that is based on DBPedia [25] data. Therefore, we extracted over $150k$ facts (RDF triples) describing entities of the domain of academics. For this domain, we define a set of specific constraints that are used to detect and to resolve inconsistencies in the knowledge base. We also generate erroneous facts and add them to the knowledge base in order to investigate the influence of the percentage of wrong statements in the knowledge base on the precision of our application. Hence, we create multiple datasets whose recall is 1.0 while its precision depends on the number of added statements. In order to obtain a probabilistic knowledge base, we assign to all statements a random weight in the range from 0.0 to 1.0. The results of the experiments indicate that our approach is able to remove erroneous statements with a high precision. Table 3 shows that the F-measure of the knowledge base increases independently of the share of added erroneous statements which is depicted in the column ΔF. The precision of the debugging process is $\approx 80\%$ while the recall is $\approx 65\%$ which means the at least 4 out of 5 are proper removals while more than half of the wrong statements get detected and removed.

With respect to the runtime, we measured that it takes $\approx$7 minutes for the initial dataset ($150k$ facts, precision = 1.0) and $\approx$18 minutes for the largest extended dataset ($300k$ facts, precision = 0.5) to determine the consistent subset. We executed our experiments on a virtual machine running Ubuntu 12.04 that has access to two threads of the CPU (2.4 GHz) and 16 GB RAM. Hence, T-RDFS-LOG is capable to verify knowledge bases by exploiting terminological knowledge and temporal relations of events.

6. Conclusions

We presented an infrastructure for probabilistic reasoning about Semantic Web data. The infrastructure is based on the state of the art Markov logic engine ROCKIT that supports very efficient MAP inference. We presented the ROCKIT system and showed how the concept of log-linear logics can be used to translate Semantic Web data into Markov logic and use ROCKIT to solve relevant problems on the web. We also demonstrated the benefits of specific probabilistic extensions of Semantic Web languages. In particular a temporal extension of RDFS as well as the EL fragment of the Web Ontology language for verifying extracted information from the web and for matching heterogeneous web ontologies. The reasoning infrastructure presented is available for download[6] and can also be accessed through a web interface that we provide for testing and smaller reasoning problems.[7]

While we have already used the infrastructure for the applications described in the paper and also for other purposes not mentioned due to lack of space, there is still a lot of potential for improving the system. First of all, we have only experimented with a restricted set of logical languages that do not fully exploit all possibilities of the concept of log-linear logics. There is a need for a more systematic investigation of the concept of log-linear logics and their possible use in the context of Semantic Web applications and beyond.

So far, our infrastructure only provides efficient reasoning support for MAP inference. Other important problems, in particular marginal inference and weight learning are only supported to a limited extend. While - as we have shown in our previous work - MAP infer-

[6] https://code.google.com/p/rockit/

[7] http://executor.informatik.uni-mannheim.de/

ence can be used to solve a number of important problems, the restriction to this particular kind of reasoning limits the applicability of the infrastructure. In particular, there is a lack of support for answering probabilistic queries over Semantic Web data. We will address these limitations in future work.

References

[1] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] Eleftherios Anagnostopoulos, Sotiris Batsakis, and Euripides GM Petrakis. Chronos: A reasoning engine for qualitative temporal information in owl. *Procedia Computer Science*, 22:70–77, 2013.

[3] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the el envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 364–369, 2005.

[4] Sotiris Batsakis, Kostas Stravoskoufos, and Euripides GM Petrakis. Temporal reasoning for supporting temporal queries in owl 2.0. In *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 558–567. Springer, 2011.

[5] Elena Bellodi, Evelina Lamma, Fabrizio Riguzzi, and Simone Albani. A distribution semantics for probabilistic ontologies. In *Proceedings of the 7th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2011)*, volume 778 of *CEUR Workshop Proceedings*, pages 75–86, Bonn, Germany, 2011. CEUR-WS.org.

[6] Dan Brickley and Ramanathan Guha. RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[7] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The dl-lite approach. In *Reasoning Web, Semantic Technologies for Information Systems*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356, 2009.

[8] Paulo CG Costa. *Bayesian semantics for the Semantic Web*. PhD thesis, George Mason University, 2005.

[9] Paulo Cesar G. da Costa and Kathryn B. Laskey. PR-OWL: A framework for probabilistic ontologies. In *Formal Ontology in Information Systems, Proceedings of the Fourth International Conference, FOIS 2006*, volume 150 of *Frontiers in Artificial Intelligence and Applications*, pages 237–249, Baltimore, Maryland, USA,, 2006. IOS Press.

[10] Claudia d'Amato, Nicola Fanizzi, Marko Grobelnik, Agnieszka Lawrynowicz, and Vojtech Svatek. Inductive reasoning and machine learning for the semantic web. *Semantic Web Journal*, 5(1):3–4, 2014.

[11] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence IJCAI 2007*, pages 2462–2467, Hyderabad, India, 2007.

[12] Maximilian Dylla, Iris Miliaraki, and Martin Theobald. A temporal-probabilistic database model for information extraction. *Proceedings of the VLDB Endowment*, 6(14):1810–1821, 2013.

[13] Maximilian Dylla, Mauro Sozio, and Martin Theobald. Resolving temporal conflicts in inconsistent rdf knowledge bases. In *14. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, pages 474–493, 2011.

[14] Michael R. Genesereth and Nils J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann, 1988.

[15] Bernardo Cuenca Grau, Zlatan Dragisic, Kai Eckert, Jérôme Euzenat, Alfio Ferrara, Roger Granada, Valentina Ivanova, Ernesto Jiménez-Ruiz, Andreas Oskar Kempf, Patrick Lambrix, Andriy Nikolov, Heiko Paulheim, Dominique Ritze, François Scharffe, Pavel Shvaiko, Cássia Trojahn dos Santos, and Ondrej Zamazal. Results of the ontology alignment evaluation initiative 2013. In *Proceedings of the 8th Ontology Matching Workshop*, pages 61–100, 2013.

[16] Claudio Gutierrez, Carlos Hurtado, and Alejandro Vaisman. Temporal rdf. In *The Semantic Web: Research and Applications*, pages 93–107. Springer, 2005.

[17] Claudio Gutierrez, Carlos A Hurtado, and Alejandro Vaisman. Introducing time into rdf. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):207–218, 2007.

[18] Patrick Hayes. RDF semantics. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[19] Jakob Huber, Christian Meilicke, and Heiner Stuckenschmidt. Applying markov logic for debugging probabilistic temporal knowledge bases. In *Proceedings of the 4th Workshop on Automated Knowledge Base Construction (AKBC)*, 2014.

[20] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *Proceedings of the 10th International Semantic Web Conference (ISWC)*, pages 273–288, 2011.

[21] Yevgeny Kazakov. Consequence-driven reasoning for horn shiq ontologies. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, pages 2040–2045, Pasadena, California, USA, 2009.

[22] Pavel Klinov. *Practical reasoning in probabilistic description logic*. PhD thesis, The University of Manchester, Manchester, UK, 2011.

[23] Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, Emilie Danna, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Andrea Lodi, Hans D. Mittelmann, Ted K. Ralphs, Domenico Salvagnin, Daniel E. Steffy, and Kati Wolter. Miplib 2010. *Math. Program. Comput.*, 3(2):103–163, 2011.

[24] Kathryn B. Laskey. MEBN: A language for first-order bayesian knowledge bases. *Artificial Intelligence*, 172(2-3):140–178, 2008.

[25] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Chris Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2014.

[26] Thomas Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7), 2008.

[27] Thomas Lukasiewicz and Umberto Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):291–308, 2008.

[28] Christian Meilicke. *Alignment incoherence in ontology match-*

ing. PhD thesis, University Mannheim, 2011.

[29] Mathias Niepert, Jan Noessner, and Heiner Stuckenschmidt. Log-linear description logics. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 11)*, pages 2153–2158, Barcelone, Spain, 2011. AAAI Press.

[30] Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2013.

[31] Jan Noessner, Heiner Stuckenschmidt, Christian Meilicke, and Mathias Niepert. Completeness and optimality in ontology alignment debugging. In *Proceedings of the 9th International Workshop on Ontology Matching collocated with the 13th International Semantic Web Conference (ISWC 2014)*, page 25, 2014.

[32] Sebastian Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, pages 468–475. AUAI Press, 2008.

[33] Fabrizio Riguzzi, Elena Bellodi, Evelina Lamma, and Riccardo Zese. Bundle: A reasoner for probabilistic ontologies. In *Reasoning and Rule Systems - 7th International Conference, RR 2013*, Lecture Notes in Computer Science, pages 183–197, Mannheim, Germany,, 2013. Springer.

[34] Yafang Wang, Mohamed Yahya, and Martin Theobald. Time-aware reasoning in uncertain knowledge bases. In *Proceedings of the Fourth International VLDB workshop on Management of Uncertain Data (MUD 2010)*, pages 51–65, 2010.