Query-Time Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules

Ndapandula Nakashole,
Max-Planck Institute for Informatics,
Saarbrücken, Germany

Mauro Sozio
Institut Mines-Telecom; Telecom ParisTech;
CNRS, Paris, France

Fabian Suchanek
Max-Planck Institute for Informatics,
Saarbrücken, Germany

Martin Theobald
Max-Planck Institute for Informatics,
Saarbrücken, Germany

ABSTRACT

Recent advances in information extraction have paved the way for the automatic construction and growth of large, semantic knowledge bases from Web sources. However, the very nature of these extraction techniques entails that the resulting RDF knowledge bases may face a significant amount of incorrect, incomplete, or even inconsistent (i.e., uncertain) factual knowledge, which makes efficient query answering over this kind of uncertain RDF data a challenge. Our engine, coined URDF, augments first-order reasoning by a combination of *soft rules* (Datalog-style implications), which are grounded in a deductive fashion in order to derive new facts from existing ones, and *hard rules* (mutual-exclusiveness constraints), which enforce additional consistency constraints among both base and derived facts. At the core of our approach is an efficient approximation algorithm for this *constrained* form of the *weighted MaxSAT problem* with soft and hard rules, allowing us to dynamically resolve inconsistencies directly *at query-time*. Experiments on real-world and synthetic data confirm a high robustness and significantly improved runtime of our framework in comparison to state-of-the-art MCMC techniques.

Keywords

Uncertain RDF, Soft and Hard Rules, Deductive Grounding, MaxSAT.

1. INTRODUCTION

The recent advent of information extraction techniques has enabled the automatic construction and growth of large, semantic knowledge bases from Web sources. Knowledge bases such as DBpedia [4], YAGO [25], *Freebase.com*, and *TrueKnowledge.com* consist of many millions or even billions of facts, which are usually captured in the form of RDF-style subject-predicate-object (SPO) triples. Moreover, the Linked-Data initiative (*LinkedData.org*) encompasses these and many other RDF datasets, along with extensive cross-linkage in the form of *owl:sameAs* properties between entities in different data sources. For high coverage of entities and their properties, it is natural to use automated, often heuristic or probabilistic, methods to populate these knowledge bases, and, perhaps, also to automatically establish *owl:sameAs* links. Consequently, these data sources may contain a significant fraction of noise and incorrect triples. However, even if we accept such data errors and inconsistencies, no knowledge base can ever be complete, and even the entire Linked-Data cloud can hardly ever cover all interesting properties of relevant entities.

Research on knowledge base construction has adopted probabilistic models and statistical learning for cleaning the gathered pool of raw fact candidates (typically extracted from Wikipedia and textual or semi-structured Web pages). A powerful instrument to this end is reasoning with *consistency constraints* (see, e.g., [6, 7, 16, 23, 26]). For example, specifying that a person can have only one spouse (at a given snapshot in time) helps distinguishing marriages from romantic affairs, and removing false hypotheses for *isMarriedTo* triples. On the other hand, keeping only those triples with the highest confidence (or likelihood) of being correct, is an unduly eager and conservative approach. For example, when searching for musician couples where both partners have won a Grammy award, correct answers, such as John Lennon and Yoko Ono, may well be composed out of low-confidence input triples, as the join predicates in the query impose additional restrictions and could implicitly "de-noise" the answers. For example, the Lennon/Ono marriage is not in any of the Wikipedia infoboxes, and respective extractions from free text should have lower confidence.

When query answers are empty because critical pieces of knowledge are missing, reasoning over uncertain facts can be helpful to produce—at least—likely or speculative results. To this end, *deduction rules* are an instrument to infer answers that go beyond the extensionally represented content of the knowledge base. These rules themselves may be uncertain as well. For example, suppose we want to find the doctoral advisor of Alon Y. Halevy. Often (but not always) the senior author on the first few papers of a researcher is the doctoral advisor. Based on such a rule, we could deduce that Yehoshua Sagiv is Halevy's advisor. Moreover, deduction rules cover a wide range of RDF/S and OWL-based reasoning concepts, such as the *owl:TransitiveProperty* of predicates (e.g., for the *rdfs:subClassOf* property or over *owl:sameAs* links), which lie at the expressive intersection of Datalog-style deductive reasoning and OWL-DL. Additional consistency constraints, on the other hand, cover OWL concepts such as the *owl:FunctionalProperty* or *owl:disjointWith* properties of predicates and classes.

In summary, we emphasize the following desiderata for query-time reasoning over uncertain RDF triples:

1) to give answers to *complex SPARQL queries* over triples with highly varying confidence values;
2) to overcome the incompleteness problem, exploit *deduction rules*,

which may themselves be uncertain, and to infer answers even if some critical triples are missing;

3) to counter amplified noise and keep query-result precision high, take into account *consistency constraints* in the specific context of a user query;
4) achieve all of the above with *high efficiency*, so that queries can be answered with *interactive response time* of a few seconds.

Our aim with URDF is to address the above desiderata in an integrated manner. Our implementation is based on top-down, on-demand grounding of rules expressed in first-order logic, together with a *constrained* form of weighted MaxSAT solving. Considering hard constraints jointly with MaxSAT reasoning over propositional clauses poses additional challenges; to our knowledge, these have not been addressed in prior work for interactive, query-time reasoning.

2. DEFINITIONS AND NOTATIONS

We are given a set X of Boolean variables, each variable taking either the value *true* or *false*. The negation of a variable $x \in X$ (denoted as $\bar{x}$), has the value *true* if and only if x is assigned *false*. We shall refer to a variable x and its negation $\bar{x}$ as a *literal*. A *Horn clause* C is a set of literals containing at most one positive literal. Given a truth assignment to variables, we shall say that a Horn clause is satisfied if it contains at least one literal whose value is *true* (clauses are assumed to be in disjunctive form). Every clause C is associated with a positive weight $w(C) \in \mathbb{R}$. A *Horn formula* is defined as a conjunction of Horn clauses (and hence Horn formulas are in conjunctive normal form, CNF). A Horn formula is *satisfiable* if there is a truth assignment to all literals such that all clauses are satisfied. As we deal with Horn formulas that might not be satisfiable, we seek to find a truth assignment that maximizes the total weight of satisfied clauses. An example of a Horn formula is the following

$$(x_1 \vee \bar{x}_2) \wedge (\bar{x}_2 \vee x_3) \quad \text{equiv. to} \quad (x_1 \leftarrow x_2) \wedge (x_3 \leftarrow x_2),$$

where $\leftarrow$ denotes logical implication.

Given a set of relation types R and a set E of entities, a *fact* f is defined as a triplet of the form $f = (e_1, e_2, r)$, which expresses an instance of a binary relation of type $r \in R$ for two entities $e_1, e_2 \in E$ (i.e., we could denote the fact that "John is married to Yoko", where John and Yoko are both entities). Moreover, facts may be uncertain. Hence every fact f is also associated with a positive weight $w(f) \in \mathbb{R}$, expressing the degree of confidence in the fact being correct. Moreover, every fact is also associated with a Boolean variable $x_f \in X$, whose value indicates whether the corresponding fact is true or false. In the following, we shall simplify the notation and do not distinguish between variables and facts anymore. Hence, assigning the value *true* to a fact f corresponds to assigning *true* to the corresponding variable x_f.

We define a *knowledge base* $\mathcal{KB}$ as a triplet $\mathcal{KB} = \{\mathcal{F}, \mathcal{C}, \mathcal{S}\}$, where $\mathcal{F}$ is a set of facts, $\mathcal{C}$ is a set of Horn clauses, and $\mathcal{S}$ is a collection of disjoint sets of facts. We shall refer to $\mathcal{C}$ and $\mathcal{S}$ as *soft deduction rules* and *hard consistency constraints*, respectively (or *soft* and *hard rules* for short).

2.1 Soft Deduction Rules

We consider weighted Horn clauses with exactly one positive head literal as soft rules. A soft rule could state, for example, that *"if Yoko and John are married and John lives in NY, then also Yoko lives in NY"*, with a confidence of 0.38 of being correct, which we write as follows:

$$\mathit{lives}(\mathit{Yoko}, \mathit{NY}) \leftarrow \mathit{married}(\mathit{Yoko}, \mathit{John}) \wedge \mathit{lives}(\mathit{John}, \mathit{NY})_{[0.38]}$$

To tackle the inherent incompleteness of $\mathcal{F}$, we lift soft rules into first-order rules with universally quantified variables, which serve as input to our knowledge base. Soft rules then have the shape of first-order Horn clauses and can be written as Datalog-style implications. To express the first-order rule that *"married couples usually live in the same place"*, for example, we use the following compact notation:

$$\mathit{livesIn}(p_1, \mathit{loc}_1) \leftarrow \mathit{marriedTo}(p_1, p_2) \wedge \mathit{livesIn}(p_2, \mathit{loc}_1)_{[0.38]}$$

2.2 Hard Consistency Constraints

The set of facts $\mathcal{F}$ may contain inconsistent information. Hence, we introduce hard consistency constraints that take the shape of a collection of disjoint sets of mutually-exclusive facts $\mathcal{S} = S_1, \ldots, S_{|\mathcal{S}|}$. We enforce the constraint that for each hard rule S_k, at most one fact $f \in S_k$ may be assigned the value *true*. For example, if we observe two or more different birth dates for a person, clearly something went wrong either during extraction or when reasoning with soft rules. We can formally identify such an inconsistency by formulating a consistency constraint as follows:

$$(\mathit{date}_1 = \mathit{date}_2) \leftarrow \mathit{bornOn}(p_1, \mathit{date}_1) \wedge \mathit{bornOn}(p_1, \mathit{date}_2)$$

Grounding this hard rules could, for example, then yields the following set of mutually exclusive facts

$$\{\mathit{bornOn}(\mathit{John}, \mathit{1931}), \mathit{bornOn}(\mathit{John}, \mathit{1940}), \mathit{bornOn}(\mathit{John}, \mathit{1957})\}$$

which specifies that John could be born either in 1931, 1940, 1957, or in none of the above years. In contrast to soft rules, hard rules may not be violated by any truth assignment to the corresponding Boolean variables.

2.3 Problem Definition

As we deal with Horn formulas that might not be satisfiable, we seek to find a truth assignment that maximizes the total weight of the satisfied clauses. We call this problem *weighted MaxSAT with soft and hard rules* which we formally define as follows.
We are given a set of facts $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, a collection $\mathcal{C} = C_1, \ldots, C_m$ of clauses in Horn form (soft rules), and a collection of sets $\mathcal{S} = S_1, \ldots, S_t$ that partition $\mathcal{F}$ (hard rules). Each clause $C \in \mathcal{C}$ is associated with a positive weight $w(C)$. We wish to find a truth assignment to each variable $f \in \mathcal{F}$ such that for each set in $\mathcal{S}$ at most one fact is assigned the value true*, and the sum of the weights of the satisfied clauses is maximized.*

Given a knowledge base $\mathcal{KB} = \{\mathcal{F}, \mathcal{C}, \mathcal{S}\}$ (in grounded form), we define an instance of the MaxSAT problem with soft and hard rules as follows. Every fact $f_i \in \mathcal{F}$ is associated with a Boolean variable. In addition, we introduce a unit clause $C_i = \{f_i\}$, whose weight is equal to the confidence of the corresponding fact. For convenience of notation, for each fact f_i, we also introduce a unit hard rule $S_i = \{f_i\}$ into $\mathcal{S}$. As the MaxSAT problem with hard and soft rules is a generalization of the classic MaxSAT problem with Horn clauses, which is NP-Hard [11], it follows that also our problem is NP-Hard. Because of the intractability to compute an optimum solution for the above problem, we resort to devise an approximation algorithm.

We remark that instead of hard rules, one could also enforce consistency constraints by introducing soft rules with high weights. However, in combination with an approximation algorithm, this approach may involve non-trivial issues as illustrated by the following example. Consider the following two facts $\mathit{bornOn}(\mathit{John}, \mathit{1931})$ and $\mathit{bornOn}(\mathit{John}, \mathit{1940})$, whose weights are 0 and $w \gg 0$, respectively. In order to enforce the hard rule that John can only have one date of birth, we could introduce the following soft rule $(\bar{x} \vee \bar{y})$ where x and y are the Boolean variables associated with facts $\mathit{bornOn}(\mathit{John}, \mathit{1931})$ and $\mathit{bornOn}(\mathit{John}, \mathit{1940})$, respectively. However, it is not clear how to determine the weight W of such a soft rule. If W is not large enough, then we could not enforce the hard consistency constraint. On the other hand, if W is too large, then any $(1+\epsilon)-$approximation algorithm (for $\epsilon > 0$) might assign true to $bornOn(John, 1931)$, as the ratio between such a solution and the optimum is $\frac{W}{W+w}$.

3. ALGORITHM

Our algorithm is inspired by Johnson's approximation algorithm for the original weighted MaxSAT [12] problem (considering only weighted clauses but no additional consistency constraints). This is based on the *method of conditional probabilities* [1] which allows to convert a randomized approximation algorithm into a deterministic one, while preserving the approximation guarantee. From a practical point of view, one advantage of this approach is that often the resulting deterministic algorithms are efficient.

The first step is to compute a real value p_i in $[0, 1]$ for each fact f_i, satisfying the following property: the sum of all p_i's corresponding to the facts within a same hard rule is at most 1. Each p_i is interpreted as the probability that f_i is assigned *true* and is computed in such a way that p_i is large when the confidence in fact f_i being true is high (i.e., $w(f_i)$ is large) and small otherwise. Due to space constraints, we omit the details of the computation of the p_i's (see our technical report for more details [27]). Then at each step t, we consider a hard rule S_t and we determine a truth assignment for all facts in S_t which maximizes a carefully defined objective function. Our potential function can be interpreted as the expected total weight of satisfied clauses with each unassigned fact f_i being assigned true with probability p_i (independently from the facts not belonging to S_t).

Formally, we denote with W_t the value of our potential function at step t. At the beginning of our algorithm all facts are unassigned and the value of our potential function (W_0) is defined as

$$W_0 = \sum_{C \in \mathcal{C}} w(C) \cdot \mathcal{P}(\text{C is satisfied}).$$

At step t, let $\hat{\mathbf{f}}_{t-1} = \hat{f}_1, \ldots, \hat{f}_{t-1}$ be the truth assignment for the facts $\mathbf{f}_{t-1} = f_1, \ldots, f_{t-1}$ and let S_t be a hard rule considered at step t. We denote with $S_t = false$ the truth assignment that assigns *false* to all facts in S_t. For every f in S_t, we define

$$W_{t,f=true} = \sum_{C \in \mathcal{C}} w(C) \cdot \mathcal{P}(\text{C is sat.}|\hat{\mathbf{f}}_{t-1} = \mathbf{f}_{t-1}, f = true)$$

where $\mathcal{P}(A|B)$ denotes a conditional probability. When all facts in S_t are assigned *false* our potential function is denoted as

$$W_{t,S_t=false} = \sum_{C \in \mathcal{C}} w(C) \cdot \mathcal{P}(\text{C is sat.}|\hat{\mathbf{f}}_{t-1} = \mathbf{f}_{t-1}, S_t = false).$$

Our algorithm determines the truth assignment that maximizes the current potential function by choosing the highest value among all $W_{t,f=true}$'s and $W_{t,S_t=false}$ and then assigns the corresponding truth values to the facts in S_t. At each iteration, all clauses that become satisfied by this truth assigment are removed from the set of clauses.

Our algorithm stops when all facts have been assigned a truth value. We remark that our algorithm is completely deterministic (i.e., it always produces the same output given the same input). Algorithm 1 shows pseudocode for our algorithm, while the approximation guarantee of our algorithm is stated in Theorem 1.

Algorithm 1 Weighted MaxSAT with Soft and Hard Rules

```
For each hard rule compute a prob. distribution over its facts.
For each hard rule $S_t \in \mathcal{S}$:
  · Let $f$ be the fact with largest $W_{t,f=true}$ among all facts in $S_t$.
  · · If $W_{t,f=true} \geq W_{t,S_t=false}$ then assign true to $f$,
      else assign false to all facts in $S_t$.
  · Remove all satisfied clauses.
```

THEOREM 1. *Given a set $\mathcal{C}$ of Horn clauses and a set $\mathcal{S}$ of hard rules, let λ be the minimum number of negated literals in any Horn clause that has at least two literals. Our algorithm is a p-approximation algorithm for the MaxSAT problem with soft and hard rules, where p is obtained by solving the equation $p = 1 - p^{\lambda}$. The running time of our algorithm is $O(mn)$ in the worst case, with $m = \sum_{c \in \mathcal{C}} |c|$ and $n = \sum_{s \in \mathcal{S}} |s|$.*

Moreover, we obtain the following result as a corollary.

COROLLARY 1. *Our algorithm has an approximation guarantee of $1/2$ for general Horn clauses.*

Due to space constraints, we defer the proof of Theorem 1 to our technical report [27]. We are not aware of any closed form formula to express the solutions of the equation $p = 1 - p^{\lambda}$ as a function of λ. In the case $\lambda = 2$ we obtain $p \approx 0.618$, while in the case $\lambda = 3$ we obtain a ratio of roughly 0.68. We can also show an approximation guarantee of 0.83 in some cases of interest (see our technical report [27]). We remark that the algorithm reaches its worst case running time when every fact occurs in every grounded soft rule, i.e., when $|C| = |\mathcal{F}|, \forall C \in \mathcal{C}$. In practice, this case is highly unlikely, and in fact our experiments confirm the efficiency of our algorithm in real-world applications (see Figure 1a).

4. REASONING FRAMEWORK

In the absence of any soft and hard rules, URDF conforms to a standard (conjunctive) SPARQL engine where all facts in $\mathcal{F}$ are assumed to be true. Our key observation for query answering in combination with MaxSAT solving is that still often only a small subset of facts in $\mathcal{F}$—often several orders of magnitude less facts than those contained in the entire knowledge base—are relevant for answering a specific query and for finding the corresponding truth assignments to the facts that are related to the query. For this purpose, we define the *dependency graph* $\mathcal{D}_Q \subseteq \mathcal{F}$ of a query Q as follows.

DEFINITION 1. *Given a knowledge base $\mathcal{KB} = \{\mathcal{F}, \mathcal{C}, \mathcal{S}\}$ and a conjunctive query Q, then:*

- *All possible groundings $f_i \in \mathcal{F}$ of the query atoms $q_j \in Q$ are facts in the dependency graph $\mathcal{D}_Q$ of Q.*
- *If a grounded fact $f_n \in \mathcal{F}$ is in $\mathcal{D}_Q$, then all grounded facts $f_1, \ldots, f_{n-1} \in \mathcal{F}$ of all grounded soft rules $C \in \mathcal{C}$, in which f_n is the head, are also in $\mathcal{D}_Q$.*
- *If a grounded fact $f_i \in \mathcal{F}$ is in $\mathcal{D}_Q$, then all grounded facts $f_1, \ldots, f_k \in \mathcal{F}$ of the grounded hard rule $S \in \mathcal{S}$, which are mutually exclusive to f_i, are also in $\mathcal{D}_Q$.*

Definition 1 already yields a recursive algorithm to compute the dependency graph, which is similar to SLD resolution [3] used in Datalog and Prolog. In our case, SLD resolution is extended by an additional grounding step for of hard rules, i.e., whenever we ground a fact f_i, we also iteratively ground all hard rules that are related to it, using f_i as a new subquery[1]. The URDF reasoning steps are summarized in Algorithm 2.

[1] We employ a form of tabling (i.e., memoization) in order to cache redundant subgoals. This table also serves to break potential cy-

Algorithm 2 URDF Reasoning Framework

```
Require: A knowledge base $\mathcal{KB}$ with base facts $\mathcal{F}$, first-order soft
    rules $\mathcal{C}$ and first-order hard rules $\mathcal{S}$, a conjunctive query $Q$
1: Initialize the dependency graph $\mathcal{D}_Q = \emptyset$.
2: Ground all literals $q_i \in Q$ via SLD resolution and add their
    intersection to $\mathcal{D}_Q$.
3: Let $\mathcal{C}_Q, \mathcal{S}_Q$ denote the sets of soft and hard rules grounded for
    answering $Q$.
4: Expand $\mathcal{D}_Q$ by all facts $f$ in grounded rules $\mathcal{C}_Q$ and $\mathcal{S}_Q$.
5: Construct a CNF formula over grounded clauses $\mathcal{C}_Q$ and indi-
    vidual facts $\mathcal{D}_Q \subseteq \mathcal{F}$.
6: Solve the constrained weighted MaxSAT over the CNF and sets
    $\mathcal{S}_Q$ (Algorithm 1).
7: return $\mathcal{D}_Q$ with truth assignments to all facts $f \in \mathcal{D}_Q$
```

Given a set of first order rules, this form of deductive grounding has a well-known polynomial runtime for non-recursive Datalog programs, and for linear, recursive programs, respectively. It however has an exponential complexity (in the number of facts) already for Datalog programs with a single, non-linear, recursive rule [9]. Line 3 denotes the rules that were grounded during this resolution phase in order to construct a Boolean formula in conjunctive normal form (CNF). These grounded rules are already available from the previous SLD resolution and can be kept in a simple buffer of the algorithm. The CNF construction in Line 5 itself is linear in the size of the grounded rules $\mathcal{C}_Q$ and $\mathcal{S}_Q$, because all grounded soft rules are already in clause form, while the grounded hard rules can be input into our MaxSAT solver directly as plain sets of mutually exclusive facts. The next step in Line 6 requires the execution of Algorithm 1 for the weighted MaxSAT problem (with both soft and hard rules) described in Section 3.

We remark that the above form of dependency graph construction guarantees truth assignments to query answers that are consistent with the truth assigments that would be found by the MaxSAT solver for the entire sets of facts $\mathcal{F}$, clauses $\mathcal{C}$, and constraints $\mathcal{S}$. That is, the truth assignments to facts in the dependency graph $\mathcal{D}_Q \subseteq \mathcal{F}$ for any query Q after MaxSAT solving are equivalent to the truth assignments that would be obtained for these facts when running the MaxSAT solver over the entire set of facts $\mathcal{F}$ in the knowledge base (modulo ties and possible MaxSAT approximation errors).

5. EXPERIMENTS

The following experiments were run on an AMD Opteron Quad-Core 2.6 GHz server with 16 GB RAM, using Oracle-11g as storage back-end for the underlying knowledge base. Physical I/O's were cached (thus aiming to eliminate variances due to disk operations) by running each query once and then taking the average runtime over 5 immediately repeated runs. Memory consumption was generally not a delimiting factor, with up to 501 MB overall memory consumption for our URDF Java implementation (including the high overhead of the Java VM) and less than 10 MB for the Alchemy package (see Subsection 5.2), implemented in C++.

5.1 YAGO Knowledge Base, Rules and Queries

The semantic graph of YAGO [25] serves as basis for our experiments. YAGO is a large common-sense knowledge base that has been extracted automatically from Wikipedia articles. YAGO contains more than 2 million entities and 19 million facts. The facts include a class hierarchy of 200,000 classes with about 100 distinct relation types. Moreover, we employ 16 (partially recursive) hand-crafted soft deduction rules of common-sense reasoning about people and their relationships, together with 10 queries of different shapes. We enforce functional properties of the predicates *bornIn*, *bornOnDate*, *diedIn*, *diedOnDate* and *marriedTo* as consistency constraints. As weights for base facts, we employ the confidence weights provided by YAGO, while the weight for a soft rule is calculated as a conditional probability for the entire rule to hold (including the head literal), given that the body of the rule holds (when grounded over YAGO, see [27] for a detailed description of rules and queries). Queries were chosen such that many query predicates are defined via deduction rules, which led to a recursion depth of up to 7 in our experiments.

5.2 Markov Logic: MAP and MC-SAT

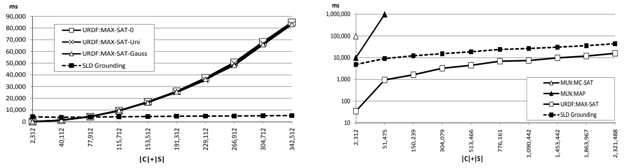
Alchemy[2] provides a series of algorithms for statistical relational learning and probabilistic inference based on the Markov Logic Networks [23]. It implements various MCMC sampling techniques, including MAP inference [24] (which is a memory-efficient stochastic MaxSAT solver based on MaxWalkSAT) and MC-SAT [20]. MAP inference yields truth assignments to facts (which allows for precision comparisons with URDF), whereas MC-SAT computes probability distributions over the underlying Markov network (which URDF does not do). Thus, we merely refer to MC-SAT for runtime comparisons as a state-of-art technique for MCMC sampling. We found grounding the above rules and queries over the entire YAGO knowledge base in Alchemy under an open-world assumption not to be feasible due to the nearly quadratic deflation of the resulting MLN structure. Hence, we provide the facts and rules grounded by URDF (via SLD resolution) directly as input to Alchemy, which effectively leads to a closed-world grounding of rules in the corresponding MLN structure. MLN running times (for MAP and MC-SAT) thus mostly correspond to the time needed for inference by Alchemy over this much smaller network structure (plus some overhead for parsing the formulas and grounding the network in closed-world mode).

5.2.1 Basic Query Processing over YAGO.

The first setting reports running times and result precision for the URDF reasoner compared to MAP inference and MC-SAT over the basic YAGO setting. As for approximation quality, we measure the *relative precision* of the URDF MaxSAT solver compared to the MAP baseline: if the MaxSAT weight computed by URDF (*MAX-SAT-W*) is at least as large as the weight achieved by MAP inference (*MAP-W*), and none of the hard constraints are violated by either approach, we conclude that we found an equally good or even better solution. Grounding time (*SLD*) denotes the time needed to ground the query atoms, soft and hard rules, and to expand the dependency graph via a top-down SLD resolution. In the following, $\#\mathcal{C}$ and $\#\mathcal{S}$ denote the number of grounded soft and hard rules (including unit clauses), while $|\mathcal{C}|$ and $|\mathcal{S}|$ denote the number of occurrences of facts in all grounded soft and hard rules, respectively. The overall query response time (in ms) for URDF is the sum of SLD grounding and MaxSAT solving. Table 1 shows that the URDF MaxSAT solver achieves more than two orders of magnitude runtime improvement over MAP and MC-SAT, at 90 percent precision compared to the MAP baseline for queries Q_1–Q_{10}. That is, for 9 out of the 10 queries URDF finds the same MaxSAT weight as MAP, while only for query Q_7, the weight returned by URDF is marginally worse. We also see that running MC-SAT is generally more expensive than MAP inference. In this basic setting,

cles in SLD resolution if the same rule is attempted to be grounded repeatedly with the same bindings of variables to constants.

[2] http://alchemy.cs.washington.edu/



(a) (b)

Figure 1: Asymptotic MaxSAT behavior (a) and MaxSAT vs. MAP and MC-SAT (b).

SLD grounding clearly dominates query response times for URDF. However, for all queries we achieve interactive response times with an overall runtime of at most 3 seconds per query.

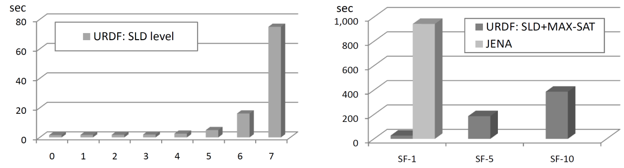
5.2.2 *Synthetic Rule Expansions.*

To systematically investigate the asymptotic behavior of our algorithm for large dependency graphs and more complex rules, we employ synthetic expansions over the basic YAGO setting. In the first expansion setting (Figure 1)(a), we expand each grounded soft rule by *10 distinct facts* as noise per expansion step, for each of the query results depicted in Table 1. For the noisy facts, we apply uniform weights and weights following Gaussian distributions (with σ^2=1) around the original weights (μ) of the YAGO facts. We thus simulate very complex CNF formulas with more than 10^5 occurrences of facts in soft rules. In the following plots, the grounding time also includes the time for expanding the CNF formulas with these noisy facts and thus is not constant for all runs. Figure 1(a) confirms that the runtime of the MaxSAT algorithm is not affected by the weighting schemes and remains equally efficient.

In the second expansion setting (Figure 1)(b), we do not only vary the number of facts per soft rule, but also the number of grounded soft and hard rules by replicating rules with different combinations of noisy facts. That is, for each original grounded fact, we introduce *10 mutually exclusive facts* as noise, and, for each soft rule, we expand the CNF by introducing *10 randomly expanded soft rules* at each expansion step. Overlap among soft rules is achieved by uniform-randomly drawing the noisy facts from a static pool of 1,000 synthetic facts for all expansions. We keep Gaussian weights for the expanded facts. This setting yields very large CNF formulas with $|\mathcal{C}| + |\mathcal{S}|$ ranging from 2,312 to 2,321,488. More specifically, we create constraints with up to 21,277 soft rules and 2,311,551 occurrences of facts in all soft rules, over an overall amount of only 9,937 distinct facts for the last expansion step. In this step, each fact on average occurs in more than 232 rules, each with an average rule length of 108 facts. URDF solves the CNF formulas for this expansion in 15.7 seconds, while remaining at a higher precision in computing the MaxSAT weight as MAP. We measure the approximation precision only for the first three expansion steps, yielding MaxSAT weights of 853.74, 975.02, $1,099.34$ for URDF and 854.58, 937.74, $1,069.70$ for MAP, respectively. MAP does not scale to larger CNFs than in these first three expansions, while MC-SAT cannot be run beyond the first expansion step anymore.

5.2.3 *Inductively Learned Rules.*

The previous experiments were run over the entire YAGO knowledge base, using 16 handcrafted rules in a fully recursive fashion. To measure the cost of deduction over such a large knowledge base with more general rules, we conduct runs over 42 recursive rules learned inductively by a variant of FOIL [21], using YAGO as background knowledge. Figure 2(a) depicts the runtimes (in seconds) for grounding queries Q_1–Q_{10} over YAGO using these rules, however breaking SLD resolution at different deduction levels.



(a) (b)

Figure 2: SLD deduction levels (a) and LUBM results (b).

5.2.4 *LUBM Benchmark.*

Figure 2(b) shows a comparison of URDF runtimes (in seconds, including both SLD and MaxSAT) versus the Jena Semantic Web toolkit[3] (using PostgreSQL 8.3 as storage backend) over the LUBM benchmark setting [10] at scale factors (SF) 1, 5 and 10. Performance for URDF is similar to that observed on YAGO, where MaxSAT times are again much faster than SLD grounding times. URDF outperforms Jena even in the grounding step by a large margin, while Jena failed to respond to many queries at increasing scale factors (see also [10] for detailed results on LUBM).

6. RELATED WORK

Reasoning with inconsistency and uncertainty has been gaining significant attention in the Database, Semantic Web, and Machine Learning communities lately. In contrast to related works on uncertain and probabilistic databases, we consider a more dynamic way of querying and reasoning with deduction rules for intensional relations. All database approaches for uncertain data we are aware of— for example Trio [29], MayBMS [2], and MystiQ [5]—limit queries to materialized data. Some systems support views [2, 29], but only in materialized form which is equivalent to comprehensive and thus expensive bottom-up grounding in Datalog. Moreover, we also found recent approaches that adopt inference methods for probabilistic graphical networks to database systems, such as [18] or [28], to be primarily designed for batch processing and not to be well suited for interactive querying. A similar observation also holds for Probabilistic Logic Programming (PLP) and Answer Set Programming (ASP). PLP combines logic programming (including Datalog) with probabilistic reasoning. ProbLog [22] employs SLD resolution [3] for grounding first-order rules, together with Monte Carlo sampling or binary decision diagrams for probabilistic inference over Boolean formulas obtained from SLD proofs. ASP, on the other hand, is a more generic paradigm for solving combinatorial problems with logic programs. In [19], the authors study tractable subclasses of ASP with cardinality constraints and weights. In Probabilistic Description Logic [14] (PDL) (see [15] for an overview), probability ranges can be attached to subsumption (or instance-of) statements. PDL generalizes the description logic $\mathcal{SHOIN}^{(D)}$ and can thus express functional rules. However, PDL cannot deal with truly inconsistent input statements. Thus, in order to apply PDL to a knowledge base with noisy confidence scores in place of the probabilities, the probability ranges would have to be reconciled upfront—which amounts to solving the inconsistencies before starting the reasoner.

Statistical Relational Learning (SRL) has been gaining a strong momentum in both the Database and Machine Learning communities, with Markov Logic Networks (MLNs) [23] probably being one of the most generic approaches for combining first-order logic and probabilistic graphical models. In these classes of graphical models, sampling methods based on Markov Chain Monte Carlo

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

					URDF/MLN	URDF	MLN	MLN	URDF	MLN
	$\#\mathcal{C}$	$\lvert\mathcal{C}\rvert$	$\#\mathcal{S}$	$\lvert\mathcal{S}\rvert$	SLD(ms)	MaxSAT(ms)	MAP(ms)	MC-SAT(ms)	MaxSAT-W	MAP-W
Q_1	49	109	22	25	243	1	80	65	19.92	19.92
Q_2	14	20	10	12	53	1	814	17	9.69	9.69
Q_3	32	40	27	28	25	7	814	17	20.56	20.56
Q_4	46	82	23	30	178	1	861	221	20.77	20.77
Q_5	176	203	167	167	3,062	13	1,564	60,970	161.93	161.93
Q_6	318	342	307	310	584	4	111	173	292.40	292.40
Q_7	100	220	41	48	222	4	1,344	1,032	**27.06**	**27.91**
Q_8	195	199	192	193	93	7	1,877	36,330	188.21	188.21
Q_9	44	71	27	35	143	7	1,407	142	26.37	26.37
Q_{10}	89	89	89	89	71	4	283	5,267	86.83	86.83
$\sum$	1,063	1,375	905	937	4,674	49	9,155	104,234	853.74	854.58

Table 1: Basic YAGO results.

(MCMC) provide a family of efficient approximation algorithms for probabilistic inference. Specifically, the algorithms employed in Markov Logic for maximum-a-posteriori (MAP) inference [24] and MC-SAT [20] constitute two state-of-the-art extensions to Max WalkSAT-based, stochastic MaxSAT solvers [13] and Gibbs-style sampling techniques, respectively. Our approach diverges from Markov Logic in two basic aspects: grounding and inference. Grounding a first-order Markov network in an open-world semantics works by binding all entities (constants) to variables in the first-order rules that match the type signature of the respective predicates. For binary predicates, this may result in grounded networks which are often nearly quadratic in the number of entities in the knowledge base. Unlike Markov Logic, we specifically focus on query-time reasoning, with a deductive (closed-world) grounding of soft rules.

In [26], we presented a "self organizing framework for information extraction", where the main problem has been formulated also as a MaxSAT problem. The main difference between this work and [26] is that here we deal with hard and soft rules in a more principled way, while in [26] hard rules are enforced by introducing soft rules with large weights (see Section 2 for a discussion about why a more principled approach is needed). Moreover, we present an algorithm with an approximation guarantee of 0.83 in some cases of interest. The MaxSAT problem is well studied in the theoretical computer science community (see for example [8]). Here, we focus on the effectiveness of our algorithms in solving real-life problems. In [17], we presented an initial demo of our interactive reasoning framework.

7. CONCLUSIONS

We presented a query-time reasoning approach for uncertain RDF data and SPARQL queries over a combination of soft deduction rules and hard consistency constraints. URDF employs a generalized weighted MaxSAT algorithm that guarantees consistent query answers with regard to the hard constraints. Our experiments confirm that our MaxSAT approximation algorithm yields interactive response times over formulas with many thousands of grounded rules and facts, while empirically performing much better than the provided lower bound of 1/2 for the approximation guarantee. We also saw that, in many cases, the grounding time for the Datalog-style soft deduction rules is the actual delimiting factor for query response times. Our future work will investigate how to further scale up inference by a combination of dynamic grounding over the highly transient parts of the knowledge base with materialized facts for the more static parts, as well as by distributing our grounding and MaxSAT-based inference strategies.

8. REFERENCES

[1] N. Alon and J. Spencer. *The Probabilistic Method.* John Wiley, 1992.

[2] L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing incomplete information with probabilistic world-set decompositions. In *ICDE*, 2007.

[3] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *J. ACM*, 29(3):841–862, 1982.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: a nucleus for a web of open data. In *ISWC*, pages 722–735, 2007.

[5] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Ré, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *SIGMOD*, pages 891–893, 2005.

[6] A. Carlson, J. Betteridge, R. C. Wang, E. R. H. Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM*, pages 101–110, 2010.

[7] M.-W. Chang, L.-A. Ratinov, N. Rizzolo, and D. Roth. Learning and inference with constraints. In *AAAI*, pages 1513–1518, 2008.

[8] M. X. Goemans and D. P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discr. Math.*, 7(4):656–666, 1994.

[9] G. Gottlob and C. Papadimitriou. On the complexity of single-rule Datalog queries. *Inf. Comput.*, 183:104–122, 2003.

[10] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2-3):158–182, 2005.

[11] B. Jaumard and B. Simeone. On the complexity of the maximum satisfiability problem for Horn formulas. *Information Processing Letters*, 26(1):1–4, 1987.

[12] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, pages 256–278, 1974.

[13] H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. In *The Satisfiability Problem: Theory and Applications*, pages 573–586. American Mathematical Society, 1996.

[14] T. Lukasiewicz. Probabilistic description logic programs. *Int. J. Approx. Reasoning*, 45(2):288–307, 2007.

[15] T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *J. of Web Semantics*, (6):291–308,2008.

[16] A. McCallum, K. Schultz, and S. Singh. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *NPIS*, pages 1249–1257. 2009.

[17] T. Meiser, M. Dylla, and M. Theobald. Interactive reasoning in uncertain RDF knowledge bases. In *CIKM*, pages 2557–2560, 2011.

[18] F. Niu, C. Ré, A. Doan, and J. W. Shavlik. Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. *PVLDB*, 4(6):373–384, 2011.

[19] R. Pichler, S. Rümmele, S. Szeider, and S. Woltran. Tractable answer-set programming with weight constraints: Bounded treewidth is not enough. In *KR*, 2010.

[20] H. Poon, P. Domingos, and M. Sumner. A general method for reducing the complexity of relational inference and its application to MCMC. In *AAAI*, pages 1075–1080, 2008.

[21] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[22] L. D. Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, pages 2462–2467, 2007.

[23] M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, 2006.

[24] P. Singla and P. Domingos. Memory-efficient inference in relational domains. In *AAAI*, 2006.

[25] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *WWW*, 2007.

[26] F. M. Suchanek, M. Sozio, and G. Weikum. SOFIE: A self-organizing framework for information extraction. In *WWW*, 2009.

[27] M. Theobald, M. Sozio, F. M. Suchanek, and N. Nakashole. URDF: An efficient reasoning framework for uncertain knowledge bases with hard and soft rules. Technical report, Max-Planck-Institute for Informatics, 2010.

[28] M. L. Wick, A. McCallum, and G. Miklau. Scalable probabilistic databases with factor graphs and MCMC. *PVLDB*, 3(1):794–804, 2010.

[29] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.