Time-aware Reasoning in Uncertain Knowledge Bases

Yafang Wang, Mohamed Yahya, and Martin Theobald
{ywang,myahya,mtb}@mpi-inf.mpg.de

Max-Planck Institute for Informatics
Saarbruecken, Germany

Abstract. Time information is ubiquitous on the Web, and considering temporal constraints among facts extracted from the Web is key for high-precision query answering over time-variant factual data. In this paper, we present a simple and efficient representation model for time-dependent uncertainty in combination with first-order inference rules and recursive queries over RDF-like knowledge bases. In the spirit of *data lineage*, the intensional (i.e., rule-based) structure of query answers is reflected by Boolean formulas that capture the logical dependencies of each derived answer fact back to its extensional roots (i.e., base facts). Our approach incorporates simple *weight aggregations* for *begin*, *end* and *during* evidences for base facts, but also generalizes the common *possible-worlds semantics* known from probabilistic databases to histogram-like confidence distributions for derived facts. In particular, we show that adding time to the latter probabilistic setting adds only a light overhead in comparison to a time-unaware probabilistic setting.

1 Introduction

Recent progress in information extraction has led to major breakthroughs in automatically building large ontological knowledge bases from high-quality Web sources, such as online news sites, or encyclopedias like Wikipedia. Projects such as DBpedia [1], KnowItAll [6] and its underlying extraction frameworks TextRunner [23] and Kylin/KOG [22], ReadTheWeb [4], as well as our own YAGO project [17], have successfully shown how to build structured knowledge representations from unstructured or weakly structured Web collections with high precision and recall. A major shortcoming that these knowledge bases still face is the lack of time information in the both the representation model and the types of queries they support. Thus, these information extraction techniques generally work well if we consider the quality of each of the extracted facts individually, but time constraints start playing a major role when the knowledge base is queried, i.e., when we would like to reason about multiple facts with respect to their temporal context. For example, a query looking for all teammates of David Beckham during his time at Real Madrid would only be meaningful if we have explicit information about *when* each of the team members of Real Madrid played for the club.

While extraction tools like TARSQI [20] generally perform well in detecting time adverbials in text, they also introduce a certain amount of errors. Even if

we would restrict ourselves to structured and mostly trustworthy sources such as Wikipedia infoboxes, achieving 100% precision in temporal fact extraction will likely remain an illusive goal. Key factors are the incorrect detection and resolution of temporal annotations caused by the high diversity of temporal expressions used in free text, as well as plain inconsistencies among different sources. As an illustration, one news article might report *"The hype and speculation have escalated ever since the January announcement that Beckham would join the Galaxy from Spanish giants Real Madrid on a deal that will earn him a reported $250 million over five years."*, while another article mentions *"Former England captain David Beckham left London Thursday to begin his new stint playing for the Los Angeles Galaxy"*[1]. From these headlines, we could extract the fact *(David Beckham joins Los Angeles Galaxy)* with different time annotations. First, the granularity of these time annotations is different ("January" is a month, while "Thursday" is day), and second, the latter is only a relative time annotation that needs to be resolved and matched with the publication date of the news article. Furthermore, time information often is incomplete. In the first example, there is no information about the year when Beckham announced to *join* Los Angeles Galaxy, while in the second one, not even a month or week for when Beckham *left* Galaxy is stated. Thus, failures in recognizing and resolving temporal expressions are an important factor in introducing uncertainty and even inconsistency to temporal knowledge bases.

Moreover, for reasoning and query answering, new temporal facts need to be derived from existing temporal facts. Knowing, for example, the facts that a player *joined* and *left* a club, we could derive a time interval for when this player actually *played* for the club. Furthermore, teammates of the player and their corresponding time intervals could be derived as well, which calls for a principled approach to reasoning in temporal knowledge bases with uncertainty. For this purpose, we started building Timely-YAGO (T-YAGO for short) [21], which enriches our previously built knowledge base YAGO [17] by *validity intervals* for facts. Similar to work done on temporal databases [12], validity intervals provide simple, yet effective, support for query semantics built on interval intersections and unions in T-YAGO. Simple interval operations are however only of limited use for query processing (or reasoning) with *uncertainty*, i.e., with probabilistic models or otherwise statistically quantified degrees of uncertainty. In this paper, we adopt the common possible-worlds semantics known from probabilistic databases and extend it towards histogram-like confidence distributions that capture the validity of facts across time. Query processing is done via a Datalog-like, rule-based inference engine, which employs the *lineage* of derived facts for confidence computations to remain consistent with the possible-worlds model.

1.1 Example Setting

Consider the example knowledge base in Figure 1, which illustrates a number of facts about football players, coaches, and their teams. Initially, we are facing the situation where facts with temporal annotations have been extracted from different documents, which might yield different observations for when Beckham

[1] Citations taken from actual news articles

and Ronaldo have joined and left Real Madrid, respectively, each with a different frequency. Then the question arises how these different temporal annotations should be reconciled into a concise representation model. Suppose we are uncertain about the exact time point when Beckham *joined* and *left* Real Madrid. Then, what should be the time interval for when Beckham actually *played* for Real Madrid? Further, we observe that both Beckham and Ronaldo played for Real Madrid. Yet, what is their chance of being *teammates*, and when did they overlap? And what cup did Ronaldo *win* when he played for Real Madrid; or who is the *coach* of the England National Team, and when? Although there are various systems that manage uncertain data, none of them could readily solve the problems stated here. We aim to answer these questions in the following.

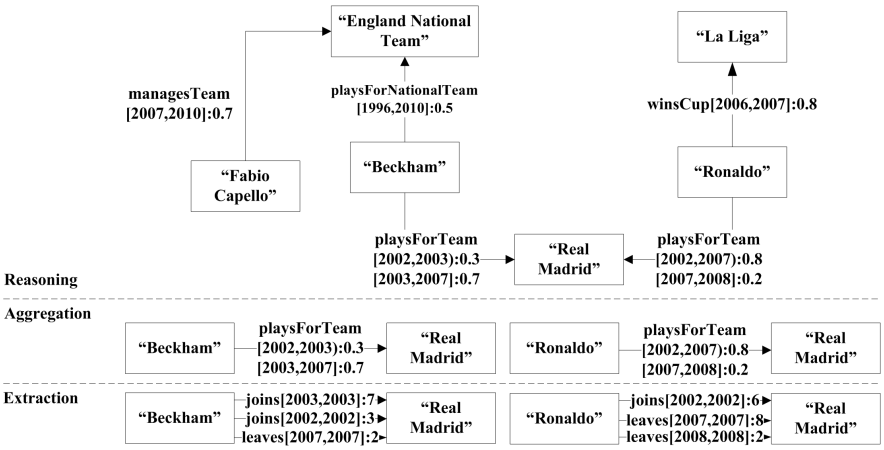

Fig. 1. Example for extracting facts with time annotations and a resulting temporal knowledge base.

1.2 Contributions and Outline

We propose an approach for representing and reconciling facts with temporal annotations for time-aware reasoning over uncertain and potentially inconsistent temporal knowledge bases. We briefly summarize the main contributions of this paper as follows:

- **Closed and Complete Representation Model for Temporal Knowledge Bases.** We develop a histogram-based data model for representing uncertainty about the validity of facts across time. In particular, we distinguish between *event* and *state* relations and show how to combine both into a unified framework for query processing (Section 2).
- **Temporal Fact Extraction with Histogram Aggregation.** We show how to reconcile multiple (potentially inconsistent) observations of facts with temporal annotations into a concise histogram at extraction time (Section 3).
- **Possible-Worlds based Reasoning over Temporal Knowledge Bases.** We employ data lineage in the form of Boolean formulas that capture the logical dependencies between base and derived facts, in a recursive, Datalog-like reasoning environment (Section 4).

– **System and Experiments.** We evaluate our system on a real-world temporal knowledge (Timely YAGO) with more than 270,000 (aggregated) temporal facts, using handcrafted rules for query processing and reasoning in the football domain (Section 5).

2 Data and Representation Model

Knowledge Base. We define a *knowledge base* $\mathcal{KB} = \langle \mathcal{F}, \mathcal{C} \rangle$ as a pair consisting of base facts $\mathcal{F}$ and first-order inference rules $\mathcal{C}$. In Semantic Web applications, facts are often encoded in the Resource Description Framework (RDF) format, while the Web Ontology Language (OWL)—or more commonly one of its decidable subsets OWL-DL or OWL-lite—is used to express further constraints over the knowledge base. Just like RDF, our set of base facts $\mathcal{F}$ constitutes a directed, labeled multi-graph, in which nodes are entities, and labeled edges represent relationships between the entities. For example, an RDF graph can have an edge between the entity *Beckham* and the entity *Real Madrid*. This edge would be labeled with the relation name *playsForTeam*. More formally, an RDF graph is defined as a set of entities Ent and a set of relations Rel, where every $R \in Rel$ is such that $R \subseteq Ent \times Ent$, and a set of triplets (or facts) $\mathcal{F} \subseteq (Rel \times Ent \times Ent)$. Unlike RDF, we also associate a *time histogram* H_f with each fact $f \in \mathcal{F}$. The time histogram H_f captures the (discrete) probability distribution of f being valid at a particular time point $t \in H_f$.

Inference Rules. We focus on a decidable subset of first-order logic for our inference rules $\mathcal{C}$. More specifically, we focus on Datalog-like Horn clauses, which can be employed for inferring new facts (i.e., reasoning) at query time. For example, a rule like

$$playsForTeam(x, y) \leftarrow joinsTeam(x, y) \wedge leavesTeam(x, y) \tag{1}$$

can be used to infer that an entity x has played for a particular team y. In the following, we will denote variables by lowercase identifiers and constants by uppercase names (with all variables implicitly being *universally quantified*).

Time Points, Intervals, and Histograms. A time point t denotes a smallest time unit of fixed granularity. We have a discrete series of ordered time points $0, \ldots, N$ (with a special designator N which marks the end of the time range we consider). These time points could represent any desired—but fixed—granularity (e.g., years, days, or seconds, or even transaction-based counters).

A *validity interval* is represented as a left-closed, right-open or right-closed, interval, which is bounded by two time points (e.g., $[1990, 2010)$), thus denoting a discrete and finite set of time points. This way, we are able to support both *range-queries* (i.e., *"Is this fact valid in the range of [1999, 2006)?"*) and *snapshot queries* (i.e., *"Is this fact valid at time point 2006?"*). A snapshot query can then simply be seen as a special-case range query by using an interval consisting of just a single time point (e.g., $[2006, 2006]$). Every interval has a corresponding *confidence value* associated with it, which denotes the probability of the fact being valid for the given interval. Multiple, non-overlapping intervals can be concatenated to form a *time histogram*. Intervals in a time histogram do not necessarily have to be contiguous. A gap between two consecutive intervals is equivalent to an interval with a confidence value of 0.

Event and State Relations. In the following, we distinguish between *event* and *state* relations. In an event relation, a fact is valid at *exactly one* time point. By default, facts in an event relation are thus associated with a validity interval consisting of only one time point. For capturing uncertainty, however, validity intervals (and entire histograms) may cover more than one time point, as in the following example:

$$winsCup(Beckham, ChampionsLeague)[1999, 2001) : 0.8$$

For simplicity, we assume a *uniform* distribution for the probability of a fact within the interval in this case. For example, for the interval $[1999, 2001)$, which covers 2 time points with a confidence of 0.8, each time point in the interval would have a probability of 0.4. The confidences of all intervals (and implicitly also the confidences of the corresponding time points) must form a proper probability distribution, i.e., the sum of all intervals' confidences may be at most 1.

For a state relation, a fact is valid at *every* time point of an interval. Hence, all time points in the interval are (implicitly) associated with the probability of the interval, as in the following example:

$$playsForTeam(Beckham, United)[1992, 2003) : 0.3; [2003, 2007) : 0.4$$

Here, for the interval $[1992, 2003)$, which covers 12 time points with a confidence of 0.3, the fact is valid at each time point with probability of 0.3; and for the interval $[2003, 2007)$, the fact is valid at each time point with probability of 0.4. For facts in a state relation, the confidences of all intervals must form a proper probability distribution.

For both event and state relations, the sum p of confidences for the intervals in a histogram may be less than 1. In general, a fact is *invalid* for all time points outside the range of time points captured by the histogram with probability $1 - p$. Moreover, different operations for slicing and coalescing intervals apply, depending on whether a fact belongs to either an event or a state relation.

Slicing and Coalescing. In analogy to temporal databases [12], different operations for reorganizing time intervals (and thus histograms) apply. For an *event* relation, we can slice an interval into any set of disjoint and contiguous subintervals by applying our uniformity assumption of confidences. Further, we can coalesce any two contiguous intervals into a single interval, only if the individual time points in both intervals have the same probability. In this case, the confidence of the coalesced interval is the sum of the confidences of the two input intervals. For a *state* relation, however, slicing intervals into subintervals is generally not allowed. Further, we can coalesce any two contiguous subintervals into a single interval, only if they have the same confidence. In this case, the confidence of the coalesced interval in a state relation is the same as the confidence of the two input intervals.

Closed and Complete Representation Model. We remark that this model is a generalization of the possible-worlds data model used in various probabilistic database approaches (see, e.g., [2,5,11]), which now lets us express uncertainty about a fact's validity across time. In particular, this model allows for arbitrary Boolean combinations of both state and event facts for query processing, such that the distribution of confidences of any derived fact is guaranteed

to form a proper probability distribution again (*closedness*). Moreover, any discrete and finite distribution of confidences can be captured by this model, also for both base facts and derived facts (*completeness*). A detailed definition of these operations for query processing is provided in Section 4.

3 Temporal Fact Extraction and Histogram Aggregation

In our temporal model for extraction, each fact is associated with its possible earliest and latest time information. For example, from the sentence *"Beckham signed up for Real Madrid in 2003."*, we infer that Beckham joined Real in the *year* 2003. Using *days* as our primary granularity for reasoning, we determine the possible earliest (begin) time point of starting his contract to be *2003-1-1* and the latest (end) time point as *2003-12-31* (using date-formatted time points for better readability). The begin and end time points then constitute an initial time interval *[2003-1-1, 2003-12-31]* for this occurrence (evidence) of the fact *joins(Beckham, Real)* in the document. But then the question arises, how we should reconcile multiple of these (potentially inconsistent) occurrences, which we are likely to observe in different documents during the extraction phase, and how to represent these in a concise histogram for query processing.

Fact	Time Expression	Begin Time	End Time	Frequency	Event Type
joins	"July, 2003"	2003-7-1	2003-7-31	2	*begin*
(Beckham, Real)	"Summer, 2003"	2003-6-1	2003-9-30	3	
leaves	"June, 2007"	2007-6-1	2007-6-30	1	*end*
(Beckham, Real)	"Early June, 2007"	2007-6-1	2007-6-10	2	
hasContract	"Season 2003	2003-7-1	2007-6-30	2	*during*
(Beckham, Real)	to 2007"				

Table 1. Examples of time expressions and their corresponding intervals.

The extraction stage produces facts which may be valid at both a single time point (e.g., a day or a year) and entire intervals (e.g., multiple days or years). Staying in our football example, we aim to *aggregate* multiple occurrences of such events into a single *state* fact *playsForTeam(Beckham, Real)[2003-1-1, 2007-12-31]*. This state fact for *playsForTeam* can be inferred, for example, from two event facts *joins(Beckham, Real)[2003-1-1, 2003-12-31]* and *leaves(Beckham, Real)[2007-1-1, 2007-12-31)*. Besides events that indicate the *begin* and *end* of an interval, we can also directly extract events that happened *during* the period when Beckham played for Real, such as *hasContract(Beckham, Real)[2003-7-1, 2007-6-30]*. Table 1 depicts a few examples of time expressions along with their corresponding intervals and possible observation frequencies as they occur at extraction time.

From these facts, we aim to derive the histogram for *playsForTeam(Beckham, Real)*. Notice, that even in case a player might have played for a team multiple times (which occurs frequently), our approach allows for aggregating multiple overlapping occurrences of *begin*, *end* and *during* events into a single histogram.

Merging Observations. Before presenting the forward and backward aggregation of event frequencies into a histogram, we first introduce the basic algorithm for reorganizing the bins of an output histogram, given two or more input histograms, as depicted in Algorithm 1. That is, at each time point where

the confidence of an input histogram changes (i.e., at every interval boundary of an input interval), the confidence in the output histogram may change as well, and a new bin in the output histogram is created. Initially, the input histograms correspond to the basic intervals we extracted for the *begin*, *end* and *during* events (see Table 1).

This (binary) reorganization operation of bins is associative and commutative, hence multiple input histograms can be reorganized into a single output histogram in arbitrary order. Runtime and the number of bins in the output histogram are *linear* in the number of bins in the input histograms. Notice that the smallest-possible width of a histogram bin is a single time point.

Algorithm 1 Reorganizing histograms.

Require: Two input histograms H_1, H_2
Let T be the disjoint union of begin and end time points from intervals in H_1 and H_2, respectively (in ascending order)
Let H_3 be an empty output histogram
Set t_b := -1
For all $t_e \in T$ **do**
  If $t_b > -1$
    Insert a new interval $[t_b, t_e)$ into H_3
  Set $t_b := t_e$
Return: H_3

Forward and Backward Aggregation of Frequencies. As we have finished the histogram reorganization from the basic *begin*, *end* and *during* events, we continue to aggregate and normalize the frequencies for our fact in the target relation *playsForTeam*. Intuitively, the confidence of the *playsForTeam* should increase while we aggregate frequencies of intervals that indicate a *begin* event; it should increase at the begin of a *during* interval but decrease at the end of a *during* interval; and it should decrease for intervals relating to *end* events. The amount of occurrences for *begin* and *end* events may however be imbalanced, such that we also need to normalize the frequencies of each of these two types individually, before combining them into a single histogram. To obtain an increasing confidence from *begin* events, we cumulate frequencies of each bin from the first bin to the last bin (forward aggregation). In contrast, to obtain a decreasing confidence from *end* events, we cumulate frequencies of each bin from the last bin to the first one (backward aggregation).

As shown in Figure 2, we first define the reorganized histograms H_1 and H_2 by aggregating the frequencies of all *begin* and *end* events of Table 1 according to their type. Forward aggregation then iterates over all bins of H_1 by cumulating the bins' weights as $H_1[i] = \sum_{0 \leq j \leq i} H_1[j]$, starting with the first bin $H_1[0]$. On the contrary, the backward aggregation iterates over all bins of H_2 by cumulating the bins' weights as $H_2[i] = \sum_{e \geq j \geq i} H_2[j]$, starting from the last bin $H_2[e]$. In the next step, both H_1 and H_2 are normalized to the weight of H_3, i.e., the aggregated histogram of all *during* events, before all three histograms are again aggregated and normalized to form the final confidence distribution of the *playsForTeam* fact (step 3 in Figure 2). In case no *during* event could be extracted from the sources, an artificial *during* interval with the earliest and latest time points of *begin* and *end* events with weight 1 can be created as H_3,

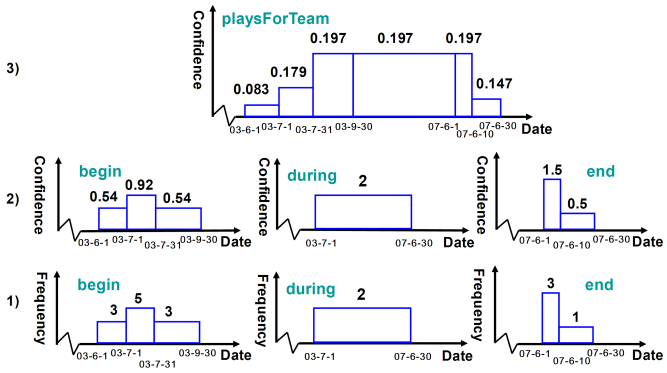

Fig. 2. Reorganizing and merging histograms based on the input facts from Table 1.

in order to normalize H_1 and H_2. These various levels of aggregation are summarized in Algorithm 2. Figure 2 provides an illustration of these three iterative reorganization and aggregation steps based on the facts in Table 1.

We remark that this aggregation of observation frequencies is just one possible way of deriving an initial histogram at extraction time. In the following, we call facts like *playsForTeam(Beckham, Real)*, which are obtained from such a forward/backward aggregation step, the *base facts*. Confidences in a probabilistic sense are traced back to only those base facts at reasoning time. Further, we assume these base facts to be *independent*.

Algorithm 2 Merging histograms.

Require: Forward-cumulated *begin* histogram H_1, backward-cumulated *end* histogram H_2, and aggregated *during* histogram H_3
Let H_4 be an empty output histogram
Reorganize H_1, H_2, H_3, and H_4 using Algorithm 1
Normalize H_1 and H_2 such that $\sum_i H_1[i] = \sum_i H_3[i]$ and $\sum_i H_2[i] = \sum_i H_3[i]$
For all $i \in H_4$ **do**
  Set $H_4[i] := H_1[i] + H_2[i] + H_3[i]$
Normalize H_4 such that $\sum_i H_4[i] = 1$
Return: H_4

4 Rule-based Reasoning, Lineage, and Possible Worlds

Our approach for reasoning in semantic knowledge bases is based on Datalog-like inference rules (Horn clauses), which can be employed to either enforce integrity constraints (Horn clauses with only negated literals) or provide means for actual inference and query answering (Horn clauses with exactly one positive literal). Recall that Horn clauses with exactly one positive literal can equivalently be rewritten as implications, where the positive literal becomes the head of the rule and the body is a conjunction of the remaining literals. Our key observation is that the logical dependencies of query answers (i.e., the *possible worlds* the entire knowledge base can take) are determined only by the way rules were processed in order to ground the query atoms (potentially recursively) down to the base facts.

In this paper, we focus on the case of Horn clauses with exactly one positive head literal, because it results in Boolean formulas with positive (i.e., conjunctive or disjunctive) lineage only.

Temporal Predicates. For reasoning about time intervals, we employ additional *temporal predicates* such as *overlaps*, *before*, *after*, etc. (see, e.g., Allen et al. [7] for an overview of temporal relations among intervals). These temporal predicates allow us to constrain the temporal relationships of time-annotated facts in the rules. Within the formulation of a rule, we also extend the given (binary) predicates by a third time variable t which is used as reference when reasoning with the temporal predicates (see Rules (2) and (3)). While this extension clearly remains in first-order logic, it—strictly speaking—no longer conforms with the core RDF data model.

Queries. Queries in Datalog can be expressed as Boolean combinations of literals (again, we do not allow negation). Hence, $teammates(Beckham, x)$ would retrieve all teammates of Beckham, while $teammates(x, y)$ would denote all pairs of teammates that could be inferred from the knowledge base. Literals in queries are grounded against the knowledge base. Semantically, a disjunction of two literals relates to a disjoint union of two sets of facts (obtained from grounding each literal), while a conjunction relates to a set intersection. Set operations in these reasoning settings are always duplicate eliminating.

Conjunctive vs. Disjunctive Lineage. When processing a query, predicates in the body of an inference rule are combined conjunctively, while multiple rules with the same head predicate create a disjunctive derivation of the query answer. In analogy to probabilistic databases, processing the body of a rule thus conforms to a join operation with conjunctive lineage, whereas grounding the same derived fact from multiple rules conforms to a duplicate-elimination step with disjunctive lineage [2,16]. We thus adopt a similar notion of data lineage as in [2] to compute the individual confidences of bins in the time histogram of a derived fact. In a Datalog-like setting, however, rules are potentially recursive, such that the derivation of answers typically is less uniform than for a regular SQL query or materialized view. Lineage however remains acyclic also in our setting, because all rules are grounded against base facts to find valid answers.

ID	Fact	Histogram	Relation Type
F_1	*playsForTeam(Beckham, Real)*	[2003,2008):0.8	state
F_2	*playsForTeam(Ronaldo, Real)*	[2002,2008):0.7	state
F_3	*winsCupForTeam(Ronaldo,Real)*	[2003,2004):0.6	event

Table 2. Base facts with time histograms (intervals).

$$teammates(x,y) \leftarrow playsForTeam(x,z,t_1) \ \wedge \ playsForTeam(y,z,t_2) \\ \wedge \ notEquals(x,y) \ \wedge \ overlaps(t_1,t_2) \quad (2)$$

$$teammates(x,y) \leftarrow playsForTeam(x,z,t_1) \ \wedge \ winsCupForTeam(y,z,t_2) \\ \wedge \ notEquals(x,y) \ \wedge \ overlaps(t_1,t_2) \quad (3)$$

As an example, consider we want to retrieve the probability of Beckham and Ronaldo being teammates for Rules (2) and (3) and the base facts depicted in Table 2. We will next discuss how confidence computation works in this setting.

Confidence Computation. While grounding queries via rules yields exactly one Boolean lineage formula for a derived fact, the input confidences of base facts may vary across time. Hence our algorithm needs to ensure that the correct confidences are chosen as input when calculating the confidence of a result histogram. This is achieved via reorganizing the bins of the output histogram using Algorithm 1 and slicing and coalescing the input intervals of base facts belonging to an event relation accordingly. Notice that intervals from base facts belonging to a state relation do not have to be sliced, since a fact is defined to be valid at each time point of an interval with the same probability (see Section 2).

Thus, grounding the query *teammates(Beckham,x)* over the above Rules (2) and (3) and base facts depicted in Table 2 results in the (single) grounded query answer *teammates(Beckham, Ronaldo)* with lineage $(F_1 \wedge F_2) \vee (F_1 \wedge F_3)$. However, by simply multiplying the probability of each literal in the lineage of *teammates(Beckham, Ronaldo)*, we would get $0.8 \times 0.7 \times 0.8 \times 0.6 = 0.2688$. This is not correct, since the probability of *playsForTeam(Beckham, Real)* is considered twice. Assuming independence among base facts, we can calculate the correct probability of *teammates(Beckham, Ronaldo)* for the interval $[2003, 2004)$ as $0.8 \times 0.7 \times 0.6 + 0.8 \times 0.7 \times (1 - 0.6) + 0.8 \times (1 - 0.7) \times 0.6 = 0.704$ (as can be verified by a truth table). For simplicity, we show the confidence computation only for a single interval. In general, one such computation can be triggered for each bin of a time histogram, again using Algorithm 1 for reorganizing the histogram, but with a possible-worlds-based confidence computation instead of the simple aggregation of Algorithm 2.

Our approach for confidence computations with time histograms can thus be summarized into the following two steps:

1) reorganizing bins of the output histogram using Algorithm 1, and
2) computing the confidence for a fact's validity at each bin of its histogram.

While step 1) is linear in the number of input bins, each confidence computation per output bin is #P-complete for general Boolean formulas [15]. We thus employ the Luby-Karp family of sampling algorithms for approximating the confidence computation. Different versions for Luby-Karp sampling [13] are available, depending on whether the formula is in CNF, DNF, or of generic Boolean shape, each with different approximation guarantees. Thus, as a simple optimization, our implementation is able to check for the structure of the formulas at query time, and it can select the most appropriate variant of Luby-Karp, or even an exact confidence computation if this is still feasible.

In our current implementation, lineage is transient, i.e., we keep lineage information only in memory at query processing time. For future work, we aim to investigate also making lineage persistent, thus being able to "learn" new facts from existing facts in the knowledge base and storing these derived facts along with their derivation in the knowledge base for further processing and faster subsequent inference.

5 System Setup and Experiments

Our system is implemented as an extension of URDF [18], which is a framework for efficient reasoning over uncertain RDF knowledge bases developed at the Max

Planck Institute for Informatics. URDF employs SLD resolution for grounding first-order formulas (Horn clauses) against an underlying knowledge base. Unlike most Datalog engines, URDF follows a top-down grounding approach, i.e., for an incoming query, it aims to resolve answers by processing rules recursively from the head predicate down to the body predicates, which are conjunctions of predicates found either in the knowledge base or which can in turn be processed via the head predicate of a rule. URDF is implemented in Java 1.6 with about 4,000 lines of code. All experiments were run on an Intel Xeon 2.40GHz server (in single-threaded mode) with 48GB RAM. We use Oracle 10g as backend for storing the T-YAGO knowledge base, which was installed on a second AMD Opteron 2.6 GHz server with 32GB RAM.

As competitors we employ the original URDF framework (without the temporal extension) and the IRIS [3] reasoner, a default reasoning engine used in many Semantic Web applications. In terms of reasoning, IRIS [3] is an open-source Datalog engine supporting built-in predicates. It is designed to be highly configurable, allowing for different Datalog evaluation strategies and the definition of custom data types and predicates.

5.1 Timely YAGO Knowledge Base

Our experiments are based on the semantic graph of T-YAGO [21]. For T-YAGO, we extracted more than 270K temporal facts from Wikipedia and freetext, with 16 distinct relationship types. Currently it covers the football domain, including relationships such as *playsForSeniorClub*, *participatedIn* and *winsCup*, but also raw facts for the *begin*, *end*, and *during* events of these relations, such as *joinsSeniorClub* or *leavesSeniorClub*. These raw facts can be integrated with the existing facts of the corresponding relations (e.g., *playsForSeniorClub*), in order to reconcile time histograms using the aggregation rules depicted in Table 4.

The facts and time histograms are stored in two separate tables. The `facts` table contains three columns for RDF triplets (i.e., first argument, second argument, and relation name) and a column for the fact id. The `time` table is composed of two columns (i.e., start time point and end time point) corresponding to the begin and end time point of an interval, a foreign key connecting to the fact's id, and a column for the fact's confidence at the interval.

5.2 Rules and Queries

Table 4 depicts 4 aggregation rules for reasoning about the time interval of a player's or coach's career period, as well as 9 partly recursive, hand-crafted inference rules for reasoning about people's activities and relationships in the football domain. As URDF (without time) and IRIS do not support time-aware reasoning, we remove all temporal predicates in the inference rules, such as *overlaps* or *after*, when comparing their runtimes and results. Table 4 illustrates 8 queries including single-fact queries, chains, stars and cliques of interconnected facts used as our baseline for experiments.

5.3 Experimental Results

Our experiments focus on investigating the overhead of time-aware query processing, compared to a time-oblivious setting. We compare the running times and

result precision of URDF (with time) to IRIS and URDF (without time). The running time of URDF (with time) includes grounding time (using SLD resolution) and histogram creation (i.e., possible-worlds-based histogram calculation time).

Baseline Runs Without Time. Since IRIS and URDF (without time) do not support time-aware reasoning, we compare grounding time and result precision of IRIS to URDF (without time) in the first experiment. The grounding time in URDF (without time) denotes the time to ground the query atoms, using the inference rules and queries depicted in Table 4. The measured time in IRIS is the time required to ground the query using magic sets rewriting, which includes both the rule rewriting step followed by a bottom-up query evaluation over the rewritten rules. We can see that URDF already outperforms IRIS for the grounding time (both without using time-specific predicates).

Overhead of Confidence Computations with Histograms. In the second experiment, we compare the grounding time and result precision of URDF (without time) to URDF (with time). Besides the grounding time consumed by URDF (without time), URDF (with time) also includes the possible-worlds-based histogram computation time. A comparable confidence computation for facts with just a single confidence value but without a time histogram is also shown on the left-hand side for URDF (without time).

Interestingly, Table 3 shows that URDF (with time) even partly achieves better runtimes than URDF (without time) for complex queries, because URDF (with time) does not ground any answers that do not satisfy the temporal predicates. This is also the main reason for the lower precision of URDF (without time) compared to URDF (with time). The grounding time of URDF (with time) is better than URDF (without time) for Queries 4, 5, 7 and 8, even when taking also the time for building the final histogram into account. However, the time for building the histogram for Query 6 is much worse than the others, yielding 14 results with 6,552 literals in 504 disjunctions in their lineage. Also, only for Query 6 we needed to employ Luby-Karp-based sampling (using $\varepsilon = 0.05$ and $\delta = 0.05$), while all the other confidences could be computed exactly.

	Without time information				With time histograms			T-URDF/URDF
	IRIS ms	URDF ms	PWs-conf ms	# results	T-URDF ms	PWs-conf ms	# results	precision
Q_1	6893	35	2	8	45	<1	8	8/8
Q_2	821	11	<1	5	12	<1	5	8/8
Q_3	7127	1905	1191	766	2113	1	184	184/766
Q_4	6686	699	188	239	308	5	58	58/239
Q_5	7628	3099	314	190	1423	51	114	114/190
Q_6	4317	693	20345	14	1054	87600	14	8/8
Q_7	6909	6712	574	183	3277	5	17	17/183
Q_8	7125	6396	190	133	4441	1	25	25/133
$\sum$	47506	19550	<22805	1538	12673	<87665	425	avg=0.545

Table 3. Experimental results.

6 Related Work

Temporal reasoning has a fairly long history through works in logics and AI, most notably in the seminal work by Allen et al. [7]. To the best of our knowledge, our

approach is the first to integrate reasoning, temporal probabilistic RDF data, and lineage. Recently, there has been an effort to expand information extraction along the temporal dimension, coined T-YAGO [21]. T-YAGO extends the rule-based approach used for extracting YAGO [17] to temporal facts from infoboxes and category information in Wikipedia, resulting in fairly large collections of facts with temporal annotations presented using the RDF data model. In [14], a pre-trained probabilistic model is used to extract temporal information from natural language sentences and annotate facts with time intervals. However, this approach does not support reasoning about relations or query answering. Work on temporal databases dates back to the early 1980's [12]. Different semantics for associating time with facts have been defined. In the context of this paper, we use the valid-time semantics, where the time recorded for facts in a database captures the reality of the world being modeled by the database. Extensions for traditional data models have been explored to accommodate temporal data in an efficient manner, both in terms of space and query processing. There has also been an extensive effort to develop query languages for querying temporal data. Most of these efforts were attempts to modify SQL to reduce the complexity of temporal queries. There is a wealth of research on probabilistic databases and the management of uncertain data. [11] is a state-of-the-art probabilistic database management system achieving scalability. [2,16] present a framework for dealing with uncertain data and data lineage. This approach allows for the decoupling of data and confidence computations when processing queries over uncertain data [16], allowing for a wider range of query plans to be used while still maintaining the correctness of confidence computations. For dealing with probabilistic reasoning in the context of information retrieval, [8] presents a probabilistic version of Datalog, which is one of the first works to introduce a notion of intensional query semantics. [9,10,19] present a probabilistic extension to RDF and how SPARQL queries over such an extension can be supported. However, no notion of temporal reasoning has been considered in these contexts.

7 Conclusions

We believe that adding time to a knowledge base is a crucial component for high-precision query answering. Time-aware information extraction increases the demand for coping with imprecise or otherwise uncertain data and is an excellent showcase for uncertain data management. Moreover, in our approach, we show that adding time histograms involves only a light overhead over a comparable probabilistic setting that does not consider time. Time-aware reasoning may even spare unnecessary computations for false-positive answers at an early stage and thus reduce the overall runtime for query answering. Currently, the way we aggregate occurrence frequencies into our initial time histograms for the base facts still is fairly abrasive from a probabilistic point of view. Our long-term goal thus is to find appropriate generative models which allow for incorporating the actual occurrences of facts in the documents into the probabilistic interpretation. The temporal extension however is already fully integrated into our URDF reasoning framework, which provides a unified and versatile reasoning platform, including, for example, also spatial reasoning extensions.

APPENDIX: Rules and Queries

Aggregation Rules

A_1: $joinsYouthClub(a,b) \wedge duringYouthClub(a,b) \wedge leavesYouthClub(a,b)$
$\rightarrow playsForYouthClub(a,b)$

A_2: $joinsSeniorClub(a,b) \wedge duringSeniorClub(a,b) \wedge leavesSeniorClub(a,b)$
$\rightarrow playsForSeniorClub(a,b)$

A_3: $joinsNationalTeam(a,b) \wedge duringNationalTeam(a,b) \wedge leavesNationalTeam(a,b)$
$\rightarrow playsForNationalTeam(a,b)$

A_4: $beginManagesTeam(a,b) \wedge duringManagesTeam(a,b) \wedge endManagesTeam(a,b)$
$\rightarrow managesTeam(a,b)$

Inference Rules

Players playing for teams are summarized into $playsForTeam$.

C_1: $playsForYouthClub(a,b) \rightarrow playsForTeam(a,b)$
$playsForSeniorClub(a,b) \rightarrow playsForTeam(a,b)$
$playsForNationalTeam(a,b) \rightarrow playsForTeam(a,b)$

If two players play for the same team at the same time, they are teammates.
C_2: $playsForTeam(a,b,t_1) \wedge playsForTeam(c,b,t_2) \wedge overlaps(t_1,t_2) \wedge notEquals(a,c)$
$\rightarrow teammates(a,c)$

If one player plays for the same team after another player, then the former is a successor of the latter.
C_3: $playsForTeam(a,b,t_1) \wedge playsForTeam(c,b,t_2) \wedge after(t_1,t_2) \wedge notEquals(a,c)$
$\rightarrow successor(a,c)$

If one player plays for the same team before another player, then the former is an ancestor of the latter.
C_4: $playsForTeam(a,b,t_1) \wedge playsForTeam(c,b,t_2) \wedge before(t_1,t_2) \wedge notEquals(a,c)$
$\rightarrow ancestor(a,c)$

Players who have played for more than 1460 days (more than 4 years) for a team.
C_5: $playsForTeam(a,b,t_1) \wedge durationMoreThan(t_1,1460)$
$\rightarrow playedMoreThan4YearsForTeam(a,b)$

If a coach manages the team when a player is playing for the team, the coach trained this player.
C_6: $managesTeam(a,b,t_1) \wedge playsForTeam(c,b,t_2) \wedge overlaps(t_1,t_2)$
$\rightarrow isCoachOf(a,c)$

If a coach manages a team, and this is a national team, then he is a coach of a national team.
C_7: $managesTeam(a,b,t_1) \wedge playsForNationalTeam(c,b,t_2) \wedge overlaps(t_1,t_2)$
$\rightarrow isCoachOfNationalTeam(a,b)$

Queries

Single-fact queries:

For which teams (and when) did David Beckham play?
Q_1: $playsForTeam(DavidBeckham,x)$

Which teams (and when) has Alex Ferguson managed?
Q_2: $managesTeam(AlexFerguson,x)$

Who are the ancestors of David Beckham?
Q_3: $ancestor(x,DavidBeckham)$

Chain queries:

Who are the coaches of David Beckham, and which teams did they previously play for?
Q_4: $isCoachOf(x,DavidBeckham) \wedge playsForTeam(x,y)$

Who are teammates of David Beckham, who participated in the same activity as Zinedine Zidane?
Q_5: $teammates(DavidBeckham,y) \wedge participatedIn(y,z) \wedge participatedIn(ZinedineZidane,z)$

Star queries:

Who are the coaches of the England National Football Team, what cups did they win,
and which activities did they join?

Q_6: $isCoachOfNationalTeam(x,EnglandNationalFootballTeam) \wedge winsCup(x,y)$
$\wedge participatedIn(x,z)$

Who played for Manchester United for more than 4 years and was a teammate of David Beckham?
Q_7: $playedMoreThan4YearsForTeam(x,ManchesterUnited) \wedge teammates(x,DavidBeckham)$

Clique query:

Who are the successors of David Beckham who won the same cup as Beckham?
Q_8: $successor(x,DavidBeckham) \wedge winsCup(x,z) \wedge winsCup(DavidBeckham,z)$

Table 4: Aggregation rules, inference rules, and queries used for the experiments.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A nucleus for a web of open data. In: ISWC (2007)
2. Benjelloun, O., Sarma, A.D., Halevy, A.Y., Theobald, M., Widom, J.: Databases with uncertainty and lineage. VLDB J. 17(2) (2008)
3. Bishop, B., Fischer, F.: IRIS- Integrated rule inference system (2008)
4. Carlson, A., Betteridge, J., Wang, R.C., Jr., E.R.H., Mitchell, T.M.: Coupled semi-supervised learning for information extraction. In: WSDM (2010)
5. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. VLDB J. 16(4) (2007)
6. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Web-scale information extraction in KnowItAll. In: WWW (2004)
7. Fisher, M., Gabbay, D.M., (Eds.), L.V.: Handbook of temporal reasoning in artificial intelligence. Elsevier (2005)
8. Fuhr, N.: Probabilistic datalog - a logic for powerful retrieval methods. In: SIGIR (1995)
9. Fukushige, Y.: Representing probabilistic relations in RDF. In: ISWC-URSW (2005)
10. Huang, H., Liu, C.: Query evaluation on probabilistic RDF databases. In: WISE (2009)
11. Huang, J., Antova, L., Koch, C., Olteanu, D.: MayBMS: a probabilistic database management system. In: SIGMOD Conference (2009)
12. Jensen, C.S., Snodgrass, R.T.: Temporal data management. IEEE Trans. on Knowl. and Data Eng. 11(1) (1999)
13. Karp, R.M., Luby, M.: Monte-carlo algorithms for enumeration and reliability problems. In: FOCS. pp. 56–64 (1983)
14. Ling, X., Weld, D.S.: Temporal information extraction. In: AAAI'10. AAAI Press (2010)
15. Re, C., Dalvi, N.N., Suciu, D.: Efficient top-k query evaluation on probabilistic data. In: ICDE (2007)
16. Sarma, A.D., Theobald, M., Widom, J.: Exploiting lineage for confidence computation in uncertain and probabilistic databases. In: ICDE (2008)
17. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago
18. Theobald, M., Sozio, M., Suchanek, F., Nakashole, N.: URDF: Efficient reasoning in uncertain RDF knowledge bases with soft and hard rules. Tech. Rep. MPI-I-2010-5-002, Max Planck Institute Informatics (MPI-INF) (2010)
19. Udrea, O., Subrahmanian, V.S., Majkic, Z.: Probabilistic RDF. In: IRI (2006)
20. Verhagen, M., Mani, I., Sauri, R., Knippen, R., Jang, S.B., Littman, J., Rumshisky, A., Phillips, J., Pustejovsky, J.: Automating temporal annotation with TARSQI. In: ACL (2005)
21. Wang, Y., Zhu, M., Qu, L., Spaniol, M., Weikum, G.: Timely YAGO: harvesting, querying, and visualizing temporal knowledge from wikipedia. In: EDBT (2010)
22. Wu, F., Weld, D.S.: Automatically refining the wikipedia infobox ontology. In: WWW (2008)
23. Yates, A., Banko, M., Broadhead, M., Cafarella, M.J., Etzioni, O., Soderland, S.: TextRunner: Open information extraction on the web. In: HLT-NAACL (2007)