Rule Based Temporal Inference

Melisachew Wudage Chekol[1] and Heiner Stuckenschmidt[2]

1 **Data and Web Science Group, University of Mannheim, Mannheim, Germany**
`mel@informatik.uni-mannheim.de`
2 **Data and Web Science Group, University of Mannheim, Mannheim, Germany**
`heiner@informatik.uni-mannheim.de`

Abstract

Time-wise knowledge is relevant in knowledge graphs as the majority facts are true in some time period, for instance, (Barack Obama, president of, USA, 2009, 2017). Consequently, temporal information extraction and temporal scoping of facts in knowledge graphs have been a focus of recent research. Due to this, a number of temporal knowledge graphs have become available such as YAGO and Wikidata. In addition, since the temporal facts are obtained from open text, they can be weighted, i.e., the extraction tools assign each fact with a confidence score indicating how likely that fact is to be true. Temporal facts coupled with confidence scores result in a probabilistic temporal knowledge graph. In such a graph, probabilistic query evaluation (marginal inference) and computing most probable explanations (MPE inference) are fundamental problems. In addition, in these problems temporal coalescing, an important research in temporal databases, is very challenging. In this work, we study these problems by using probabilistic programming. We report experimental results comparing the efficiency of several state-of-the-art systems.

1998 ACM Subject Classification D.1.6 Logic Programming

Keywords and phrases temporal inference, temporal knowledge graphs, probabilistic reasoning

Digital Object Identifier 10.4230/OASIcs.ICLP.2017.4

1 Introduction

The advance of open information extraction and data mining have guided the automatic construction and completion of big knowledge graphs (KGs). This is often done by crawling the web and extracting facts and relations using machine learning techniques, for instance NELL [4]. Some of the KGs contain high quality, human curated facts for instance YAGO [20], Wikidata [30], DBpedia [1] and some contain probabilistic facts for instance Google's Knowledge Vault [10], NELL, DeepDive [27], ReVerb [15], and ProbKB [6]. Additionally, present-day knowledge graphs contain partially temporally annotated facts.

Time-wise knowledge can be found from patient and employee histories to event and streaming data. For instance, the fact that Barack Obama was the president of USA is valid only from 2009 to 2017. When such facts are derived via machine learning techniques, they are produced with some degree confidence indicating how likely they are to be true. We refer to knowledge graphs (KGs) that contain temporally annotated probabilistic facts as *probabilistic temporal knowledge graphs*. The emergence of such KGs poses new challenges in probabilistic reasoning. In this respect, recently, Markov logic networks (MLNs) is used for conflict resolution in uncertain temporal KGs [5]. In particular, the authors investigate maximum a-posteriori inference (MAP–computing the most probable temporal KG) for debugging noisy temporal data. This problem is known to be intractable. On the other hand, the main focus of this work is to investigate marginal inference (computing the probabilities

of queries) through the use of ProbLog (a probabilistic programming language). Moreover, we leverage a tractable rule language called probabilistic soft logic for computing most probable explanations (MPE) for probabilistic temporal KGs.

In parallel with fact and relation extraction, schema induction and rule learning[1] have been widely investigated [7, 25, 26]. Most rule learning systems, such as AIME+ [16], SHERLOCK [26], and ProbFOIL [25], produce weighted Horn rules that can be encoded into ProbLog and probabilistic soft logic (PSL). Besides such rules can be conveniently *temporalized* to take into account the temporal scope of facts. Thereby, alleviating the notoriously difficult problem of temporal rule learning. As an example consider a probabilistic Horn rule which represents "a person lives in the same place where the company she works for is located":

$0.5 :: \mathsf{livesin}(x, z) :- \mathsf{worksfor}(x, y), \mathsf{locatedin}(y, z).$

This rule can be temporalized as:

$0.5 :: \mathsf{livesin}(x, z, t, t') :- \mathsf{worksfor}(x, y, t_b, t_e), \mathsf{locatedin}(y, z, t'_b, t'_e), \mathsf{overlaps}(t_b, t_e, t'_b, t'_e)$,

where the overlaps tests if there is an overlap between the intervals $[t_b, t_e]$ and $[t_b, t'_e]$ and $[t, t'] = [t_b, t_e] \cap [t'_b, t'_e]$. ProbLog is equipped with built-in predicates that allow to represent the predicate overlaps, this permits us to perform inference in temporal knowledge graphs where inference rules often contain arithmetic predicates to determine temporal overlap. Similarly, PSL provides a programming interface for creating user defined functions.

A relevant problem in probabilistic temporal KGs is *temporal coalescing*. Temporal coalescing is the process of merging facts with identical non-temporal arguments and adjacent or overlapping time-intervals. This problem has been thoroughly investigated in the database community in a non-probabilistic setting (look for instance [3]). In this paper we investigate two approaches for coalescing probabilistic temporal facts. Overall, the contributions of this paper are the following: (i) we study temporal coalescing in a probabilistic setting and propose efficient algorithms, (ii) we provide coalescing-based query rewriting for marginal and MPE inference tasks, and (iii) we perform extensive experimental analysis over the Wikidata KG.

Outline. The paper is organized as follows. Next, we briefly introduce ProbLog, PSL and knowledge graphs. In Section 3, we present temporal coalescing of KGs. Section 4 describes representation of probabilistic temporal KGs in ProbLog. We briefly outline temporal KGs in probabilistic soft logic (Section 5). In Section 6, we evaluate our approach using Wikidata and four state-of-the-art systems. We review related work in Section 7 and provide conclusion in Section 8.

2 Background

2.1 ProbLog

ProbLog is a probabilistic extension of Prolog [22]. A ProbLog program consists of a set of definite clauses with their corresponding probabilities. The probability of a clause indicates the likelihood of that clause, i.e., it is a measure of how likely the clause is to hold or be true. Given a ProbLog program $T = \{p_1 :: c_1, \ldots, p_n :: c_n\}$, each ground c_i (a clause with no variables) is called a *fact*. Facts allow us to represent triples of a KG and definite clauses enable to encode background knowledge or schema of a KG. A ProbLog program $T = \{p_1 :: c_1, \ldots, p_n :: c_n\}$ defines a probability distribution over ground logic programs

[1] Often first order Horn rules are produced by inductive logic programming and machine learning techniques.

$L \subseteq L_T = \{c_1, \ldots, c_n\}$ of T:

$$P(L|T) = \prod_{c_i \in L} p_i \prod_{c_i \in L_T \setminus L} (1 - p_i)$$

2.1.1 Marginal query

An important problem in ProbLog is computing the probability of a query (known as the success probability). The probability of a query q over a ProbLog program T is obtained as:

$$P(q|T) = \sum_{L \subseteq L_T} P(q|L)P(L|T), \quad P(q|L) = \begin{cases} 1 \text{ if } \exists\theta : L \models q\theta \\ 0 \text{ otherwise} \end{cases}$$

where θ denotes a possible substitution for q. $P(q|T)$ is the probability that q is provable over the distribution of logic programs of T. In this paper, we make use of ProbLog to compute the marginal probabilities of temporal queries over probabilistic temporal KGs. Another important problem in ProbLog is computing the most probable explanation of a set of facts.

2.1.2 MPE inference

MPE inference is the task of finding the most likely interpretation (joint state) of all non-evidence facts NE given some evidence facts E, i.e., $\text{argmax}_{ne}\, P(NE = ne \mid E = e)$. MPE inference in a probabilistic temporal KG corresponds to computing the most probable temporal KG with the highest probability. Another Horn-based probabilistic logic programming language is PSL. MPE inference in PSL is known to be tractable.

2.2 Probabilistic Soft Logic

Probabilistic Soft Logic (PSL) uses first-order logic to specify templates for Hinge-Loss Markov Random Fields (HR-MRFs) [2]. A PSL knowledge base KB contains a set of formulae $\{(F_i, w_i)\}$, where F_i is a disjunction of literals (an atom or its negation) and $w_i \in \mathbb{R}_{\geq 0}$ is a real-valued weight. A formula F_i is called *hard* (resp. *soft*) if its weight $w_i = \infty$ (resp. $w_i \in \mathbb{R}_{\geq 0}$). A hard formula must be true in all the possible worlds of the KB. To avoid confusion (and perform experimental comparison of ProbLog and PSL over the same dataset) of probabilities in ProbLog and weights in PSL, we assume that the weights in PSL are between 0 and 1. A PSL formula is written as: $H_1 \vee \cdots \vee H_m \leftarrow B_1 \wedge \cdots \wedge B_n$, where $H_1, \ldots, H_m$ are predicates in the head and $B_1, \ldots, B_n$ are predicates in the body of the rule. PSL defines a probability distribution over all possible interpretations I of all ground atoms. The probability density function for I is defined as:

$$f(I) = Z^{-1} exp[-\sum_{r \in R} w_r (d_r(I))^p]; \quad z = \int_I exp[-\sum_{r \in R} w_r (d_r(I))^p],$$

where R denotes the set of ground formulas; w_r denotes the weight of rule r; Z is the normalization constant; p provides two different loss functions: linear ($p = 1$) and quadratic ($p = 2$), the choice of a loss function depends on the application[2]; $d_r(I)$ is r's *distance to*

[2] The "linear loss function chooses interpretations that completely satisfy one rule at the expense of higher distance from satisfaction for conflicting rules, whereas the quadratic loss function favors interpretations that satisfy all rules to some degree" [2].

satisfaction under the interpretation I, $d_r(I) = max\{0, I(r_{body}) - I(r_{head})\}$, which is defined using Lukasiewicz's relaxation of Boolean operators $\wedge$, $\vee$, and $\neg$:

$$I : b_i \to [0,1] \qquad I(b_i \wedge b_j) = max\{0, I(b_i) + I(b_j) - 1\}$$
$$I(\neg b_i) = 1 - I(b_i) \qquad I(b_i \vee b_j) = min\{I(b_i) + I(b_j), 1\}$$

The most important inference problem in PSL is MPE inference.

2.2.1 MPE inference

A most probable explanation query corresponds to the problem of finding a most probable state of a probabilistic KB. Formally, MPE inference is the task of computing $P(y|x)$ the most probable assignment for a set of variables y given observations x: $\text{argmax}_y P(y|x)$. This problem is known to be tractable. Hence, it allows to perform MPE inference efficiently in probabilistic temporal KGs.

2.3 Knowledge Graphs

A knowledge graph is a set of triples that can be encoded in the W3C standard RDF data model [19]. Let I and L be two disjoint sets denoting the set of IRIs (identifying resources) and literals (character strings or some other type of data), respectively. We abbreviate the union of these sets ($\mathsf{I} \cup \mathsf{L}$) as IL . A triple of the form $(s, r, o) \in \mathsf{I} \times \mathsf{I} \times \mathsf{IL}$ is called an *RDF triple*[3]; s is the *subject*, r is the *predicate*, and o is the *object* of the triple. Each triple can be thought of as an edge between the subject and the object labeled by the predicate; hence a set of RDF triples is referred to as an *RDF graph*. We use the term *knowledge graph* loosely to refer to an RDF graph. Automatic extraction of facts produces highly calibrated probabilistic annotations for the facts. Additionally, some of these facts can be timestamped with time intervals. Knowledge graphs that contain such facts are called probabilistic temporal KGs.

2.3.1 Probabilistic Temporal Knowledge Graphs

A temporal knowledge graph is obtained by labeling triples in the graph with a temporal element [18]. The temporal element represents the time period in which a triple is valid, i.e., the *valid time* of the triple. We consider a discrete time domain T as a linearly ordered finite sequence of *time points*; for instance, days, minutes, or milliseconds. The finite domain assumption ensures that there are finitely many possible worlds in ProbLog and Probabilistic Soft Logic (see discussion in subsequent sections). A *time interval* is an ordered pair $[t_b, t_e]$ of time points, with $t_b \leq t_e$ and $t_b, t_e \in \mathsf{T}$, which denotes the closed interval from t_b to t_e[4]. We will work with the interval-based temporal domain to define our data model.

▶ **Definition 1** (Temporal KG)**.** A temporal KG is a KG where some facts $\mathsf{g}_i = (s, r, o)$ in the graph have a valid time $[t_b, t_e]$, i.e., $\mathsf{g}_i = (s, r, o, t_b, t_e)$. We refer to g_i as a *temporal fact*.

For a temporal KG G, its *snapshot* at time t is the graph $G(t)$ (the non-temporal KG): $G(t) = \{(s, r, o) \mid (s, r, o, t, t) \in G\}$. The KG associated with a temporal KG, denoted $u(G)$, is $\bigcup_t G(t)$, the union of the graphs $G(t)$. We define *temporal entailment* as follows. For temporal

[3] We do not consider blank nodes.
[4] It is possible to extend to other interval-based representations such as $[t_b, t_e)$, left-closed, right-open interval.

KGs G_1 and G_2, $G_1 \models_t G_2$ if $G_1(t) \models G_2(t)$ for each t; $\models_t$ denotes temporal entailment [18] and $\models$ is the standard RDF entailment [19]. An extension of temporal KGs with uncertainty is studied in [5]. The authors leverage Markov logic networks to provide semantics. In this paper, we employ ProbLog and Probabilistic soft logic (PSL) for representation and reasoning tasks.

▶ **Definition 2** (Probabilistic temporal KG)**.** A probabilistic temporal knowledge graph is a tuple $K = (G, F)$ with $G = \{(g_1, p_1), \ldots, (g_n, p_n)\}$ a temporal KG where each temporal fact $g_i \in G$ is labeled with a probability p_i; and $F = \{(f_1, p_1), \ldots, (f_m, p_m)\}$ is a finite set of first order logic formulas representing background knowledge or schema and p_i denotes the probability of clause f_i.

In this paper, we restrict F to be Horn clauses that express temporal inference rules and use the Problog syntax to represent them (discussed in the next section).

▶ **Example 3.** Consider the following probabilistic temporal KG representing Michael Jordan's playing career:

(g_1)	$(\mathsf{MichaelJordan}, \mathsf{playsfor}, \mathsf{ChicagoBulls}, 1984, 1993)$	0.99
(g_2)	$(\mathsf{MichaelJordan}, \mathsf{playsfor}, \mathsf{WashingtonWizards}, 2001, 2003)$	0.7
(g_3)	$(\mathsf{ChicagoBulls}, \mathsf{locatedin}, \mathsf{Chicago}, 1966, \mathtt{now})$	1.0

The time point `now` denotes the current time instant or *until changed* from temporal databases. The temporal fact g_3 represents the fact that "the basketball team Chicago Bulls is located in the Chicago city from 1966 until `now`".

Temporalizing inference rules. Knowledge graphs often contain background knowledge to control and manage the quality of data and query answers. These background knowledge can be captured by first order logic. However, most rule extraction systems produce Horn rules (that we call inference rules), for instance, the SHERLOCK system [26] and ontological pathfinding (OP) [7] efficiently learn several thousands of first order Horn rules. Horn clauses are expressive enough to represent complex schema axioms (background knowledge). The extraction of Horn rules with temporal constraints is notoriously difficult and has been afforded limited attention from the research community. However, the majority of the rules produced by rule learning systems can be converted into temporal rules by using the following: (i) add two variables t_b and t_e that represent time points of intervals to each predicate (i.e., $r(x, y)$ becomes $r(x, y, t_b, t_e)$) and (ii) if the number of predicates in the body is more than one, introduce an arithmetic predicate which is used to test temporal overlap (i.e., $r_3(x, z) :- r_1(x, y) \wedge r_2(y, z)$ becomes $r_3(x, z, t, t') :- r_1(x, y, t_b, t_e) \wedge r_2(y, z, t'_b, t'_e) \wedge \mathsf{overlaps}(t_b, t_e, t'_b, t'_e)$ where $[t, t'] = [t_b, t_e] \cap [t'_b, t'_e]$). We refer to this process as *temporalizing inference rules*.

3 Coalescing Probabilistic Temporal Knowledge Graphs

Coalescing is a technique used in temporal databases for duplicate elimination [9, 3]. Coalescing has a number of advantages: reduces the size of the probabilistic temporal KG, avoids incorrect answers in query evaluation. For instance, consider the query '*did Michael Jordan play for Chicago bulls from 1984 to 1996?*' on the temporal facts of Fig. 1 before coalescing. The result is no, however, the same query on the coalesced fact (brown part of the figure), returns yes. Uncoalesced facts can arise in various cases: during query evaluation via projection or union operations, by not enforcing coalescing in update or insertion operations,

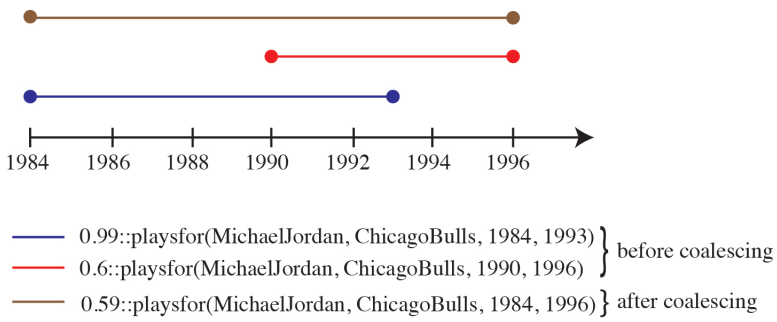


Figure 1 Coalescing probabilistic temporal facts.

and through information extraction from diverse sources or accuracy of the extractor. In this section, we discuss temporal coalescing of KGs, in the next section, we address coalescing for query evaluation.

A temporal knowledge graph G is called duplicate-free, if for all pairs of facts $p(s, o, t_b, t_e)$, $p(s, o, t'_b, t'_e) \in G$, it holds that: $[t_b, t_e] \cap [t'_b, t'_e] = \emptyset$. In other words, if the non-temporal terms of two temporal facts are the same, then their temporal terms must be disjoint (non-overlapping). Temporal coalescing is the process of merging facts with identical non-temporal arguments and adjacent or overlapping time-intervals. Often, a temporal database is assumed to be duplicate-free and coalescing is done by merging the time intervals of facts with the same non-temporal arguments. However, in a probabilistic setting, to perform coalescing is not straightforward because it is not clear what the probability of the new (coalesced) fact should be and it could be dependent of the application. Performing coalescing on a probabilistic temporal knowledge graph removes duplicates.

▶ **Definition 4** (Coalescing). Formally, two probabilistic temporal facts $p_1 :: r(s, o, t_b, t_e)$, $p_2 :: r'(s', o', t'_b, t'_e)$ can be coalesced if $s = s'$, $r = r'$, $o = o'$ and the overlap of $[t_b, t_e]$ and $[t'_b, t'_e]$ is non-empty. The probability of the coalesced fact $p_3 :: r(s, o, [t, t'])$ with $[t, t'] = [t_b, t_e] \cup [t'_b, t'_e]$ is computed using Table 1.

Rule-based coalescing. One approach to coalescing is to use the following rule-based technique. In order to coalesce all the facts of a probabilistic temporal KG, we can construct Horn rules for each relation in the KG. Thus, in ProbLog, rule-based coalescing can be done as follows: for each relation r_i in a probabilistic temporal KG K, use the following rule:

$$r_i(x, y, t, t') :- r_i(x, y, t_b, t_e), r_i(x, y, t'_b, t'_e), t \textbf{ is } min(t_b, t'_b), t' \textbf{ is } max(t_e, t'_e), t_b \leq t'_e,\ t'_b \leq t_e.$$

The expression $t_b \leq t'_e,\ t'_b \leq t_e$ tests temporal overlap of the intervals $[t_b, t_e]$ and t'_b, t'_e. Besides, **is**, *min*, and *max* are built-in predicates representing assignment, minimum, and maximum functions respectively. The probability of the coalesced facts is the product when done in ProbLog, however, this can be replaced by using Table 1 (to compute probabilities). This approach uses one rule for each relation. If a KG has 80000 relations, we need the same number of coalescing rules to coalesce the KG. Hence, this operation can be very expensive, however, it is done only once. Furthermore, this operation can be done more efficiently outside the ProbLog setting.

▶ **Example 5.** Consider coalescing the probabilistic temporal facts shown in Figure 1 using the rule-based approach. This operation merges the two facts into one with the probability of the new fact being the product of the two.

Table 1 Computing probabilities of coalesced facts based on Allen's interval relations. The computation holds for the inverse relations. g_{new} is obtained by merging (taking the union) of the intervals of the two facts and $p_{new} = p_1 + p_2 - p_1 * p_2$, based on the product rule of probability.

Interval relation	Temporal facts	Coalesced fact
Equal	(g_1, p_1) (g_2, p_2)	$(g_1, max(p_1, p_2))$
During	(g_1, p_1) (g_2, p_2)	(g_2, p_2)
Starts	(g_1, p_1) (g_2, p_2)	(g_2, p_2)
Finishes	(g_1, p_1) (g_2, p_2)	(g_2, p_2)
overlaps	(g_1, p_1) (g_2, p_2)	(g_{new}, p_{new})
Meets	(g_1, p_1) (g_2, p_2)	(g_{new}, p_{new})
Before	(g_1, p_1) (g_2, p_2)	{}

Algorithmic coalescing. Another approach for coalescing probabilistic temporal facts is based on the following. In short, the algorithm continues as follows: for each relation r search all temporal facts with the same non-temporal (s, r, o) elements, order these facts according to their time period, for overlapping time periods build the maximum time period (union of time periods), delete the temporal facts whose time periods are contained in the maximum time period, and finally assign probability to the coalesced temporal facts using Table 1. There are two important tasks in probabilistic temporal KGs: marginal and MPE inference. In order to perform these reasoning, we use ProbLog.

4 ProbLog-based Representation of Probabilistic Temporal KGs

In order to compute the probabilities of temporal queries, we represent probabilistic temporal KGs in ProbLog. This can be done by introducing a predicate for each temporal fact. To elaborate, we use a simple correspondence between temporal facts (s, r, o, t_b, t_e) and ProbLog predicates of the form $r(s, o, t_b, t_e)$ such that s, r, o, t_b, and t_e are temporal KG symbols. Thus, whenever a temporal fact (s, r, o, t_b, t_e) is satisfied, the corresponding predicate $r(s, o, t_b, t_e)$ is satisfied and the converse also holds. More formally, given a probabilistic temporal KG $K = (G, F)$ with $G = \{(g_1, p_1), \ldots, (g_n, p_n)\}$ and $F = \{(f_1, p_1), \ldots, (f_m, p_m)\}$, a ProbLog representation K_p of K is obtained as follows: (i) replace each probabilistic temporal fact $(g_i = (s, r, o, t_b, t_e), p_i)$ with $p_i :: r(s, o, t_b, t_e) \in G_p$, and (ii) replace each Horn formula (f_j, p_j) with $p_j :: f_j \in F_p$ where $p_j \in (0, 1]$. Thus, $K_p = (G_p, F_p)$ is a ProbLog program with G_p being facts and F_p is a set of definite clauses.

▶ **Example 6.** Consider the following temporal ProbLog KG $K_p = (G_p, F_p)$ based on Wikidata[5], G_p contains the facts g_4–g_6 obtained by translating the temporal facts g_1–g_3 of Example 3 into ProbLog. F_p contains the formulas f_1–f_3 shown below. The formula f_1 expresses the fact that if someone *plays for* (resp. *works for*, resp. *coaches*) a club located in a city over an overlapping period of time, then that person likely lives in the same city as where the club is located. f_2 represents if a person plays for a team and that the team participates in a league over an overlapping period of time, then that person plays in that league. Finally, f_3 is used to express disjointness of temporal relations, to be precise, a person cannot play (resp. work, resp. coach) for two different teams at the same time.

$$(f_1)\quad 0.5 :: \mathsf{livesin}(x,z,t,t') :- \mathsf{playsfor/worksfor/coaches}(x,y,t_b,t_e), \mathsf{locatedin}(y,z,t'_b,t'_e), t \textbf{ is } \mathsf{max}(t_b,t'_b), t' \textbf{ is } \mathsf{min}(t'_b,t'_e), t \leq t'.$$

$$(f_2)\quad 0.7 :: \mathsf{playsInLeague}(x,z,t,t') :- \mathsf{playsfor}(x,y,t_b,t_e), \mathsf{teamPlaysInLeague}(y,z,t'_b,t'_e), t \textbf{ is } \mathsf{max}(t_b,t'_b), t' \textbf{ is } \mathsf{min}(t'_b,t'_e), t \leq t'.$$

$$(f_3)\quad 0.9 :: false :- \mathsf{playsfor/worksfor/coach}(x,y,t_b,t_e), \mathsf{playsfor/worksfor/coach}(x,z,t_b,t_e), y \neq z, t \textbf{ is } \mathsf{max}(t_b,t'_b), t' \textbf{ is } \mathsf{min}(t'_b,t'_e), t \leq t'.$$

$$(g_4)\quad 0.99 :: \mathsf{playsfor}(\mathsf{MichaelJordan}, \mathsf{ChicagoBulls}, 1984, 1993)$$

$$(g_5)\quad 0.7 :: \mathsf{playsfor}(\mathsf{MichaelJordan}, \mathsf{WashingtonWizards}, 2001, 2003)$$

$$(g_6)\quad 1.0 :: \mathsf{locatedin}(\mathsf{ChicagoBulls}, Chicago, 1966, \mathsf{now})$$

In f_1–f_3, max and min are ProbLog built-in predicates, **is** is an assignment operator. The arthimetic expression t **is** $\mathsf{max}(t_b, t'_b), t'$ **is** $\mathsf{min}(t'_b, t'_e), t \leq t'$ tests if there is an overlap between the intervals $[t_b, t_e]$ and $[t'_b, t'_e]$.

The semantics of probabilistic temporal KGs in ProbLog can be given in terms of Herbrand interpretations. Let C be the set of IRIs and Literals that appear in some probabilistic temporal KG $K = (G, F)$ and let $K_p = (G_p, F_p)$ be its ProbLog representation, the Herbrand base of F_p can be constructed by replacing all the variables in F_p with the constants in C. For a finite set C and a set of time points T, each temporal fact in K_p can be mapped, using a substitution θ, into a subset of the Herbrand base of F with respect to C and T. A Herbrand interpretation is a subset of the Herbrand base. A Herbrand interpretation H is a Herbrand model of F_p iff it satisfies all groundings of the formulas in F_p. As in ProbLog, since the schema $F_p = \{p_i :: f_i\}$ of K_p is fixed and there is a one-to-one mapping between ground f_i clauses and Herbrand interpretations, a probabilistic temporal KG also defines a probability distribution over its Herbrand interpretations [22]. The probabilities of the facts and Horn rules determine a probability distribution in ProbLog. Formally, given a probabilistic temporal KG $K = (G, F)$ and some $K' = (G', F')$ over the same set of constants (IRIs, literals and time points) such that $K' \subseteq K$, we have the following:

$$P(K'|K) = \prod_{g_i \in G' \wedge G' \cup F' \models_t g_i} p_i \prod_{g_i \in G \setminus G'} (1 - p_i),$$

where $G' \cup F' \models_t g_i$ is temporal entailment at time point t.

4.1 Marginal Inference

An important task in probabilistic knowledge bases is computing the probability of a set of facts, i.e., given a query q and a KG K, marginal inference computes the probability of the answers of q over K. In this paper, we study temporal conjunctive queries. A temporal conjunctive query is a conjunction of a set of temporal facts.

[5] `https://www.wikidata.org/`

▶ **Definition 7.** A temporal conjunctive query q is a Horn formula of the form $q :- g_1, \ldots, g_n$ where $g_i = r_i(X_i, Y_i, t_b, t_e)$ is a temporal predicate with X_i and Y_i being temporal variables or constants; and t_b, t_e are time points or variables. Given a temporal ProbLog KG $K_p = (G_p, F_p)$ and a query q over K_p, the marginal probability of q is obtained by:

$$P(q|K_p) = \sum_{K' \subseteq K_p} P(q|K') \cdot P(K'|K_p).$$

▶ **Example 8.** Consider the following queries on the temporal facts (before coalescing) of Example 5:

1. How long is Michael Jordan's playing career? $q(t_b, t_e) :- \mathsf{playsfor}(\mathsf{MichaelJordan}, Y, t_b, t_e)$.
2. Select the teams that Michael Jordan owns and played for since 1995.
   $q(Y) :- \mathsf{playsfor}(\mathsf{MichaelJordan}, Y, t_b, t_e), \mathsf{owns}(\mathsf{MichaelJordan}, Y, t'_b, t'_e))$.

In probabilistic databases and statistical relational learning, often the probabilities of queries are computed by grounding, i.e., by replacing all the variables in the queries using constants in the database. The grounding is used to generate a propositional sentence (lineage of a query) for exact inference or a graphical model for approximate computation. Similarly, temporal conjunctive queries can be grounded by evaluating queries using the techniques from temporal databases and instantiating the variables in the queries using their answers. This results in a set of ground queries, the probability of which can be computed using ProbLog. Instead, in this paper, we evaluate temporal conjunctive queries by rewriting them in ProbLog. Temporal conjunctive queries require checking interval intersection to determine the overlap of intervals in the query predicates. In order to do this, we rewrite queries. Here, we consider only the following case and leave out the rest as a future work.

One predicate query: queries that contain only a single temporal predicate (when the size of the body of the query is one), i.e., $q(W) :- r(X, Y, t_b, t_e)$, with $W \subseteq \{X, Y\}$. Here, we consider the rewriting of queries with non-temporal variable projections. Hence, q can be rewritten $q_r(W)$ by using a self join (by rewriting the same predicate with different temporal variables) as shown below:

$$\begin{aligned} q(W) &:- r(X, Y, t_b, t_e) \\ q_r(W) &:- r(X, Y, t_b, t_e), r(X, Y, t'_b, t'_e), \mathsf{overlaps}(t_b, t_e, t'_b, t'_e). \\ \mathsf{overlaps}(t_b, t_e, t'_b, t'_e) &:- t_b \leq t'_e,\ t'_b \leq t_e. \end{aligned}$$

We use the predicate overlaps() to check if the intersection of the intervals is non-empty.

4.2 MPE inference

In addition to marginal inference, we can compute most probable explanations over temporal ProbLog KGs. MPE queries are one of the most important tasks in probabilistic reasoning. These queries are useful for computing the most probable temporal KG of a probabilistic temporal KG. Formally, given a rewritten temporal conjunctive query q, some evidence e (set of temporal facts), and a temporal ProbLog KG $K_p = (G_p, F_p)$, the most likely temporal KG of q is obtained by: $\text{argmax}_q P(q|e)$.

5 Probabilistic Temporal KGs in PSL

Since PSL like ProbLog uses Horn clauses to model KBs, we present a compact description of probabilistic temporal KGs in PSL. On the other hand, ProbLog is based on Sato's

distributional semantics and PSL is defined over hinge-loss Markov random fields. A PSL knowledge base consists of a set of (weighted) Horn rules and a set of soft evidence facts. Soft refers to probabilities. Note that while in PSL the weight of the rules can be a positive real number, in this paper, we assume the weights to be between 0 and 1. This weight conversion can be done, for instance, by the *Logit* function. We make this assumption in order to perform experiments on the same datasets for ProbLog and PSL.

A temporal PSL knowledge graph $K_{psl} = (G_{psl}, F_{psl})$ contains a set of (temporal) facts $G_{psl} = \{(g_1, w_1), \ldots, (g_n, w_n)\}$ and a set of (temporal) inference rules $F_{psl} = \{(F_1, w_1), \ldots, (F_m, w_m)\}$. Given a temporal PSL knowledge graph K_{psl}, and a set of deduction rules F_{psl}, the semantics of K_{psl} is given based on a probability density function. Formally, for a given $K_{psl} = (G_{psl}, F_{psl})$ and some K'_{psl} over the same signature, the probability density function $P(K'_{psl})$ is given by:

$$P(K'_{psl}) = \begin{cases} Z^{-1} \exp\left[\sum_{\{(g_i,w_i)\in G : K'_{psl} \models_t g_i\}} w_i (d_{g_i}(K'_{psl}))^p\right] & \text{if } K'_{psl} \models_t K_{psl} \\ 0 \qquad \text{otherwise} \end{cases}$$

where Z is the normalization constant of the probability density function P; w_i is the weight of the temporal fact g_i; $d_{g_i}(K'_{psl})$ is the distance to satisfaction of g_i in K'_{psl}; and p is a loss function. Note that in MPE inference Z is not computed. The most relevant reasoning task in PSL is MPE inference which is defined in the same way as in ProbLog.

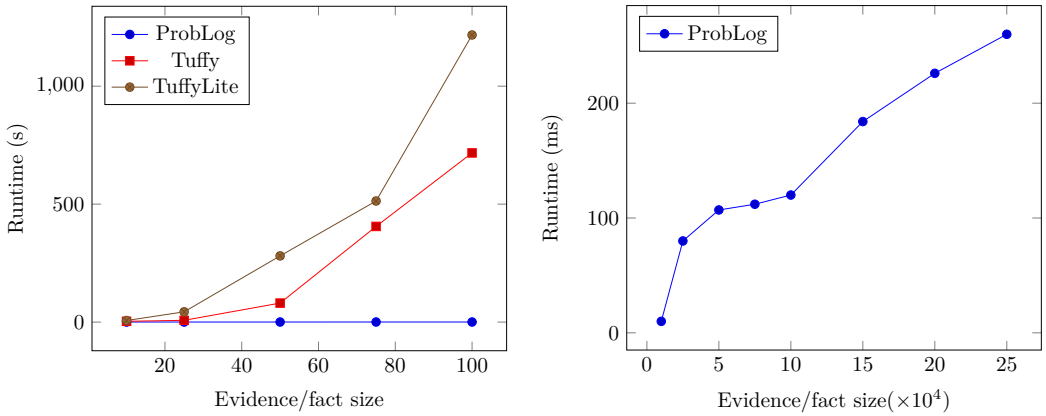
6 Experiments

We conducted two different experiments: (i) marginal inference and (ii) MPE inference. For both experiments, we carried out performance test in terms of running times by comparing four state-of-the-art solvers. We ran the experiments on a 2GHz 24-core processor with 386GB of RAM running Debian 8.

- **Tools.** We used *four* different tools for our experiments: ProbLog, PSL, Tuffy [23], and TuffyLite [21]. Tuffy and TuffyLite are based on Markov logic networks (MLNs – attaches weights to first order formulas). For marginal inference experiments, we employ ProbLog, Tuffy, and TuffyLite whereas for MPE inference, we use PSL and ProbLog. Note that PSL does not support marginal inference.
- **Temporal rules.** We designed 40 different temporal inference rules (definite clauses) based on the fluents (time-varying relations) in Wikidata.
- **Evidence.** We used as evidence different size fragments of Wikidata knowledge graph. In particular, we extract a part of the KG that contains structured temporal information (obtained from various sources using open information extraction). We extracted over 6.3 million temporal facts from Wikidata. We extracted temporal facts for various fluent relations including: *playsFor*, *educatedAt*, *memberOf*, *occupation*, *spouse*, and so on.

6.1 Marginal Temporal Query Evaluation

In this experiment, we test the scalability of marginal temporal query evaluation. We carried out the experiments using ProbLog, Tuffy, and TuffyLite. We found out that Tuffy and TuffyLite hardly scale when the size (arity) of the predicates is 3 or more (we stopped the execution after one hour timeout). On one occasion, while running Tuffy on Wikidata, we



(a). Tuffy, TuffyLite, and ProbLog. (b). ProbLog.

Figure 2 Marginal inference: runtime comparison over fixed query and varying data size.

noticed that its grounded database is 400GB large, thereby our execution eventually ran out of memory. To elaborate, the ground size in Tuffy and TuffyLite can be extremely large, for instance, for a test data size of 75 temporal facts and 40 inference rules, the grounding database contains 4,112,784 tuples. Comparatively, ProbLog uses a different grounding technique that reduces that ground size reasonably. Therefore, it is recommended to use ProbLog for marginal inference tasks in probabilistic temporal KGs when the background knowledge (inference rules) can be expressed as Horn rules. The results of the experiment are shown in Fig. 2. The reported runtimes are averaged over 5 runs for all of the solvers. As it can be seen in Fig. 2(a), due to scalability, we could only test Tuffy and TuffyLite over 10 to 100 facts. While the runtime of ProbLog increases linearly with data, the runtime of Tuffy and TuffyLite is nonlinear; it increases sharply as the size of data increases. If the inference rules of a knowledge graph can be expressed by Horn rules, then it is advisable to use ProbLog because it scales much better than Markov logic solvers (Tuffy and TuffyLite). On the other hand, if expressivity is required at the expense of scalability, Tuffy and TuffyLite can be chosen. In Fig. 2(b), we report the runtimes of marginal inference for ProbLog on large datasets ($\times 10^4$ magnitude).

6.2 MPE Inference

In this experiment, we compare the running times of ProbLog and PSL on different data sizes. Due to intractability of inference in Markov logic, we exclude Tuffy and TuffyLite. The runtimes, averaged over 5 runs, are reported in Fig. 3. As can be seen, ProbLog is faster than PSL. In addition, while the runtime of ProbLog increases linearly with respect to the datasize, the runtime of PSL does not increase linearly with respect to the size of the input data. This is due to each added incorrect temporal fact might be involved in a conflict resulting in a non trivial optimization problem. Furthermore, contrary to PSL, ProbLog handles well the temporal predicates that are used to test the overlap of temporal intervals.

7 Related Work

Temporal databases. Temporal databases have been extensively studied (see surveys [24, 28]). However, relatively few works exist on probabilistic temporal databases [11, 8]. A relational database is used to model and query temporal data, integrity constraints and deduction rules

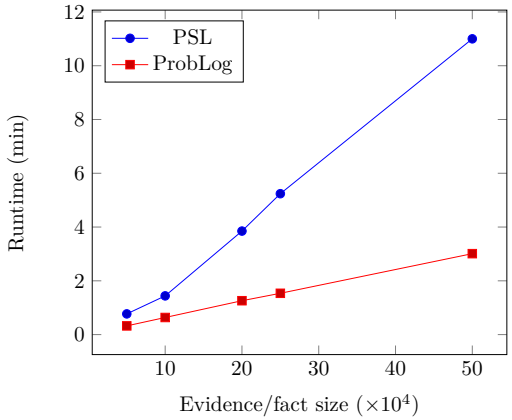


Figure 3 MPE inference: runtime comparison of ProbLog and PSL over fixed query and varying data size.

can also be specified [11]. However, these rules must be deterministic (unweighted) unlike what we do here. On the otherhand, contrary to this study where we use the valid time model, uncertain spatio-temporal databases focus on stochastically modelling trajectories through space and time (see [14] for instance).

Query evaluation in probabilistic databases is an active area of research [17, 31, 7, 29, 12]. With respect to temporal query evaluation over a probabilistic temporal knowledge base, to the best of our knowledge, there are two important studies [5] and [11]. While the former focuses on MPE inference, we study here marginal inference and deal with the problem of temporal coalescing. The later deals with marginal inference, the difference with this work are the following: (i) we consider weighted inference rules and constraints, (ii) we propose coalescing for temporal KGs, and (iii) we introduce rewriting for the coalescing of queries. In another study [13], the authors proposed an approach for resolving temporal conflicts in RDF knowledge bases. The idea is to use first-order logic Horn formulas with temporal predicates to express temporal and non-temporal constraints. However, these approaches are limited to a small set of temporal patterns and only allow for uncertainty in facts. Moreover, extending KGs using open domain information extraction, will often also lead to uncertainty about the correctness of schema information; a large variety of temporal inference rules and constraints, some of which will be domain specific, can also be the subject of uncertainty. Finally, Chen and Wang [6] debug erroneous facts by using a set of functional constraints although they do not deal with numerical and temporal facts at the same time.

8 Conclusion

Temporal reasoning is indispensable as advances in open information extraction has guided the automatic construction of temporal knowledge graphs. To perform temporal reasoning, one has to take care of the temporal scopes of facts. In addition, coalescing is necessary to prohibit errors and compact query answers. In this work, we addressed these issues. We provided theoretical as well as experimental results based on ProbLog and PSL.

A possible line of future work is to address scalability issues for ProbLog and PSL. Besides, coalescing needs to be addressed for other query operators such as union, negation, and selection.

Acknowledgements. We thank Janina Luitz for her helpful comments.

References

1 Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735, 2007.

2 Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. arXiv:1505.04406 [cs.LG], 2015.

3 Michael H. Böhlen, Richard T. Snodgrass, and Michael D. Soo. Coalescing in temporal databases. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 180–191, 1996.

4 Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3, 2010.

5 Melisachew Wudage Chekol, Giuseppe Pirrò, Joerg Schoenfisch, and Heiner Stuckenschmidt. Marrying uncertainty and time in knowledge graphs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 88–94, 2017.

6 Yang Chen and Daisy Zhe Wang. Knowledge expansion over probabilistic knowledge bases. In *SIGMOD*, pages 649–660. ACM, 2014.

7 Yang Chen, Daisy Zhe Wang, and Sean Goldberg. Scalekb: scalable learning and inference over large knowledge bases. *The VLDB Journal*, 25(6):893–918, 2016.

8 Alex Dekhtyar, Robert Ross, and VS Subrahmanian. Probabilistic temporal databases, i: algebra. *ACM Transactions on Database Systems (TODS)*, 26(1):41–95, 2001.

9 Anton Dignös, Michael H Böhlen, and Johann Gamper. Temporal alignment. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 433–444. ACM, 2012.

10 Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *SIGKDD*, pages 601–610, 2014.

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

12 Maximilian Dylla, Iris Miliaraki, and Michael Theobald. Top-k query processing in probabilistic databases with non-materialized views. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 122–133. IEEE, 2013.

13 Maximilian Dylla, Mauro Sozio, and Martin Theobald. Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases. In *BTW*, pages 474–493, 2011.

14 Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, and Andreas Zufle. Querying uncertain spatio-temporal data. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 354–365. IEEE, 2012.

15 Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP '11)*, Edinburgh, Scotland, UK, July 27-31 2011.

16 Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast Rule Mining in Ontological Knowledge Bases with AMIE+. *The VLDB Journal*, 24(6):707–730, 2015.

17 Eric Gribkoff and Dan Suciu. Slimshot: In-database probabilistic inference for knowledge bases. *PVLDB*, 9(7):552–563, 2016.

18 Claudio Gutierrez, Carlos Hurtado, and Alejandro Vaisman. Temporal RDF. In *Proc. of European Semantic Web Conference*, pages 93–107, 2005.

19 Patrick Hayes. RDF Semantics. W3C Recommendation, 2004.

20 Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232. ACM, 2011.

21 Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. Markov logic networks for natural language question answering. *arXiv preprint arXiv:1507.03045*, 2015.

22 Angelika Kimmig, Bart Demoen, Luc De Raedt, Vitor Santos Costa, and Ricardo Rocha. On the implementation of the probabilistic logic programming language problog. *Theory and Practice of Logic Programming*, 11(2-3):235–262, 2011.

23 Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *Proc. of the VLDB Endowment*, 4(6):373–384, 2011.

24 Gultekin Ozsoyoglu and Richard T Snodgrass. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, 1995.

25 Luc De Raedt, Anton Dries, Ingo Thon, Guy Van den Broeck, and Mathias Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1835–1843, 2015.

26 Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098. Association for Computational Linguistics, 2010.

27 Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. Incremental knowledge base construction using deepdive. *Proceedings of the VLDB Endowment*, 8(11):1310–1321, 2015.

28 Richard T Snodgrass. Temporal databases. In *Theories and methods of spatio-temporal reasoning in geographic space*, pages 22–64. Springer, 1992.

29 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.

30 Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

31 Hong Zhu, Caicai Zhang, Zhongsheng Cao, and Ruiming Tang. On efficient conditioning of probabilistic relational databases. *Knowl.-Based Syst.*, 92:112–126, 2016.