A Temporal-Probabilistic Database Model for Information Extraction

Maximilian Dylla
Max Planck Institute for Informatics
Saarbrücken, Germany
mdylla@mpi-inf.mpg.de

Iris Miliaraki
Max Planck Institute for Informatics
Saarbrücken, Germany
miliaraki@mpi-inf.mpg.de

Martin Theobald
University of Antwerp
Antwerp, Belgium
martin.theobald@ua.ac.be

ABSTRACT

Temporal annotations of facts are a key component both for building a high-accuracy knowledge base and for answering queries over the resulting temporal knowledge base with high precision and recall. In this paper, we present a *temporal-probabilistic database model* for cleaning uncertain temporal facts obtained from information extraction methods. Specifically, we consider a combination of *temporal deduction rules*, *temporal consistency constraints* and probabilistic inference based on the common *possible-worlds semantics* with *data lineage*, and we study the theoretical properties of this data model. We further develop a query engine that is capable of scaling to very large temporal knowledge bases, with millions of uncertain facts and hundreds of thousands of grounded rules. Our experiments over two real-world datasets demonstrate the increased robustness of our approach compared to related techniques based on constraint solving via Integer Linear Programming (ILP) and probabilistic inference via Markov Logic Networks (MLNs). We are also able to show that our runtime performance is more than competitive to ILP solvers and the fastest available, probabilistic but non-temporal, database engines.

1. INTRODUCTION

Recent advances in the field of information extraction (IE) have enabled the automatic construction and growth of large, semantic knowledge bases (KBs) from structured and semistructured Web sources. Some well-known players in this field, which mostly employ Wikipedia infoboxes as source for their extraction techniques, include DBpedia[1], Freebase[2], and YAGO2 [17]; while, for example, the Webtables project [8] or Google's Fusion Tables[3] more generally focus on large-scale IE and integration techniques for arbitrary Web domains. Knowledge bases like DBpedia 3.8 or YAGO2 today contain hundreds of millions of facts about real-world entities and their relationships. However, due to the very nature of the underlying (largely automatized) extraction and integration strategies, these KBs may still contain a significant amount of noisy facts, and they are inherently incomplete. It remains highly unlikely that we will ever be able to extract, for example, all birth places, marriages, graduation dates, etc., of people mentioned in Wikipedia with 100 percent accuracy, regardless of how sophisticated the extraction tools are. To this end, *deduction rules* and *consistency constraints* are two well studied means in databases to derive new facts from existing ones (e.g., by inferring the marriage of two people from their common engagement) and to constrain conflicting query answers that may be obtained from both the input facts and the derived facts (e.g., by restricting the possible birth places of a person to just one allowed answer).

In this work, we focus on supporting both the tasks of *building a temporal knowledge base* from the output of IE tools and on the task of *answering queries* over an existing temporal knowledge base. Initially, all of the afore mentioned KBs ignored temporal knowledge and captured only static facts. However, when looking at the extracted data, we find that a majority of facts are only valid for a specific time span (such as marriages or political positions). Thus, with respect to knowledge base building, the temporal dimension of facts is a crucial aspect when multiple extractions of the same fact are aggregated into a single fact; while, for query answering, these temporal annotations play a major role when multiple facts are put into correlation with each other by a query or constraint. Moreover, this ephemeral nature of facts is reflected already, for example, by recent extensions of the YAGO2 knowledge base, which currently provides temporal annotations—in the form of time intervals—for several millions of facts.

Despite the vast amount of recent developments in the context of *temporal databases* (TDB), on the one hand, and *probabilistic databases* (PDB), on the other hand, we are not aware of any unified *temporal and probabilistic database* approach that natively supports the kinds of data and constraints we aim to apply to our setting. As information extraction techniques inherently deliver uncertain data, they have often been considered as one of the main motivating applications for PDBs (see, e.g., [4, 6, 7, 33] for recent systems). Likewise, temporal databases (see [18, 24] for an overview of data models and query languages) have been an active field of research for many years, but so far there exists no temporal database system that also supports prob-

[1] dbpedia.org
[2] freebase.com
[3] tables.googlelabs.com

abilistic data, and vice versa. In this paper, we develop a unified database model (and query engine) that integrates these two aspects into a *closed and complete* extension of the relational data model. We motivate for combining the two worlds of temporal and probabilistic databases by introducing the following running example.

EXAMPLE 1. *Our running example is centered around the actor "Robert DeNiro", where the following temporal-probabilistic database captures a number of facts about him.*

Id	*Fact*	T	p
f_1	$BornIn(DeNiro, Greenwich)$	$[1943\text{-}08\text{-}17, 1943\text{-}08\text{-}18)$	0.9
f_2	$BornIn(DeNiro, Tribeca)$	$[1998\text{-}01\text{-}01, 1999\text{-}01\text{-}01)$	0.6
f_3	$Wedding(DeNiro, Abbott)$	$[1936\text{-}11\text{-}01, 1936\text{-}12\text{-}01)$	0.3
f_4	$Wedding(DeNiro, Abbott)$	$[1976\text{-}07\text{-}29, 1976\text{-}07\text{-}30)$	0.7
f_5	$Divorce(DeNiro, Abbott)$	$[1988\text{-}09\text{-}01, 1988\text{-}12\text{-}01)$	0.8

Fact f_1 expresses that DeNiro was born in Greenwich Village (New York) on 1943-08-17*, which is encoded into the time interval* $[1943\text{-}08\text{-}17, 1943\text{-}08\text{-}18)$ *using the ISO standardized Internet date/time format. This fact is* true *for the given time interval with probability* 0.9*, and it is* false *for this interval with probability* 0.1.[4] *Notice that another fact, f_2, states that DeNiro could have also been born in Tribeca in the interval* $[1998\text{-}01\text{-}01, 1999\text{-}01\text{-}01)$ *with probability* 0.6*. We assume that base facts are independent, i.e., these two facts are not correlated in any way at this point. Given further facts about both DeNiro's wedding with Dianne Abbott, we next aim to deduce the time interval of their marriage (as captured by the intensional relation AreMarried) by the following two deduction rules. The first rule states that a couple stays married from the begin time point of their wedding (denoted by the variable T_{b_1}) until to the last possible time point we consider (denoted by the constant t_{max}), unless there is a divorce fact at any other time interval.*

$$AreMarried(X,Y)_{[T_{b_1},t_{max})} \leftarrow \begin{pmatrix} Wedding(X,Y)_{[T_{b_1},T_{e_1})} \wedge \\ \neg Divorce(X,Y)_{[T_{b_2},T_{e_2})} \end{pmatrix} \quad (1)$$

The second rule states that a couple stays married from the begin time point of their wedding till the end time point of their divorce. f_5 describes the divorce of DeNiro and Abbott.

$$AreMarried(X,Y)_{[T_{b_1},T_{e_2})} \leftarrow \begin{pmatrix} Wedding(X,Y)_{[T_{b_1},T_{e_1})} \wedge \\ Divorce(X,Y)_{[T_{b_2},T_{e_2})} \wedge \\ T_{e_1} \leq^T T_{b_2} \end{pmatrix} \quad (2)$$

Thereby, we consider only weddings that took place before divorces. This is expressed by the temporal arithmetic predicate $\leq^T$ which compares the order of two time points. We can see that f_1 and f_2 state different birth places of DeNiro. We counter this by the following consistency constraint.

$$\neg \begin{pmatrix} BornIn(X,Y)_{[T_{b_1},T_{e_1})} \wedge \\ BornIn(X,Z)_{[T_{b_2},T_{e_2})} \wedge \end{pmatrix} Y \neq Z \quad (3)$$

When grounded, this constraint correlates the two possible BornIn facts about DeNiro, which now become mutually exclusive, regardless of their time intervals. Thus, for this constraint to hold, at least one of the two BornIn facts has to be set to false *in all the possible instances that this temporal-probabilistic DB may take. Similarly, the next consistency constraint uses a temporal condition to express that a marriage should begin after the respective person was born.*

$$\neg \begin{pmatrix} BornIn(X,Y)_{[T_{b_1},T_{e_1})} \wedge \\ AreMarried(X,Z)_{[T_{b_2},T_{e_2})} \wedge \end{pmatrix} T_{b_2} \leq^T T_{e_1} \quad (4)$$

This constraint includes both base facts and facts derived by the deduction rules for the intensional relation AreMarried. Specifically, this constraint makes all pairs of BornIn and AreMarried facts mutually exclusive, where $T_{b_2} \leq^T T_{e_1}$. $\diamond$

We summarize the contributions of this paper as follows.

- We present a *unified temporal-probabilistic database model* in which both time and probability are considered as first class citizens.
- Specifically, we define a class of *temporal deduction rules* and *temporal consistency constraints*, and we study their properties from a theoretical perspective. We show that data computations for this class of temporal deduction rules and consistency constraints are polynomial in the size of the database, while confidence computations remain $\#\mathcal{P}$-hard (in the general case) also for our setting.
- We present *efficient algorithms* for data computations in temporal-probabilistic databases, which we complement by improved strategies for confidence computations. By determining the sweetspot in the trade-off between Shannon expansions and possible-world decompositions, we are able to perform *exact probabilistic inference* for lineage formulas with more than 10^5 variables.
- We perform an *extensive experimental evaluation* over two real-world data sets obtained from current information extraction tools, which demonstrates that our approach scales to a temporal KB over millions of facts and hundreds of thousands of grounded rules, while our query answering times remain interactive. We provide detailed comparisons to several state-of-the-art techniques.

2. RELATED WORK

A large body of literature in formal logics, artificial intelligence (AI), probability theory, and databases considers temporal data management issues from various perspectives. One of the most seminal works is constituted by Allen's interval algebra [2] and has found many adoptions [36]. Since we focus on a database-oriented view of managing temporal and probabilistic data, we refer the reader to [15] for an overview of temporal aspects in logics and AI.

Probabilistic Databases. A number of PDB systems, including MystiQ [7], MayBMS [4], and Trio [6] have been released as open-source prototypes and found a wide recognition in the database community. None of these systems however natively supports temporal data. The major challenge in PDBs remains the confidence computation step for query answers, which has been shown to be $\#\mathcal{P}$-hard already for fairly simple select-project-join (SPJ) queries [9]. Seminal work by Dalvi and Suciu [10] proposed a notion of *safe query plans*, which allow confidence computations in polynomial time for a restricted class of hierarchical query plans. Sen et al. [32] investigated a class of so-called *read-once lineage formulas*, for which confidence computations are of polynomial runtime. For a comprehensive overview of research in the field of PDBs, we refer the reader to [33].

Temporal Databases. Managing temporal data in databases has been an active research field for many years, with a

[4] Facts are always *false* outside their attached time intervals.

focus on designing appropriate data models and SQL-based query languages for time-annotated tuples. Jensen's seminal work [18] provides the basis for most temporal database approaches today. Recently, Dignös et al. [12] described how to enable the usage of *sequenced semantics* in temporal databases using lineage information. Intuitively, the sequenced semantics reduces a temporal operation to corresponding non-temporal operations over the individual snapshots of the database at each point in time. However, the *non-sequenced semantics* [24], which we follow in this paper, is strictly more expressive than the sequenced one (Rules (1) and (2) are non-sequenced). Also, none of the currently available temporal database systems supports probabilistic data. Dedalus [3], which is based on Datalog, is a language for programming and reasoning about distributed systems, but does not support constraints.

Constraints in Databases. An important work by Koch and Olteanu [22] introduced constraints into a PDB setting (which we also adopt for our work). Intuitively, this work aimed to "condition" a PDB by removing all the possible worlds that do not satisfy any of the given consistency constraints. In contrast to our work, conditioning in [22] is performed as an offline process and, as such, constraints can be formulated only over the base tuples and not over any of the derived tuples or the query answers. The class of constraints we consider is formally referred to as *denial constraints* [5] and specifies sets of tuples that cannot co-exist in the database. Denial constraints have been broadly studied in the context of database repairs in deterministic DBs.

Temporal Probabilistic Databases. Relatively few works exist so far in the intersection of temporal and probabilistic DBs. Dekhtyar et al. [11] introduced temporal-probabilistic relations by adding time to a PDB. Relations are defined via discrete probability distributions over time, and they also consider consistency constraints. However, [11] does not provide a closed PDB model since they do not support lineage information and approximate probabilities using only upper and lower bounds. Wang et al. [38] performed a simple form of probabilistic deduction of temporal annotations using time histograms. They employ lineage but allow no constraints and have limited deduction capabilities. LIVE [31], which is an offspring of the Trio [6] PDB system, employs a temporal DB model by considering *transaction-time* annotations (i.e., the time point when a fact is recorded or modified) thus focusing on efficiently supporting data modifications and versioning. In contrast, *valid-time* annotations, which we consider, refer to a time interval during which a fact is considered to be valid [24]. Research in uncertain spatio-temporal databases, such as [14], focuses on stochastically modeling trajectories through space and time, rather than utilizing concepts known from PDBs (such as a possible-worlds model with data lineage [33]), as we do.

Temporal Information Extraction. Traditionally, work on temporal aspects of text focuses on detecting temporal relationships in text rather than the extraction of temporal factual knowledge. Commonly, these relationships are specified by TimeML [28], where SemEval's temporal track [35] is used as training data. Ling and Weld [23] propose a method for binding time-intervals of events occurring in text by leveraging MLNs [29]. Still, our experiments show the superior scalability of our approach. Also, in [34, 37], ILPs were employed to enforce constraints on temporal facts extracted from text, but do not allow the deduction of new facts. The emerging interest in temporal aspects in the context of Information Retrieval (IR) is also demonstrated by the recent TREC Knowledge-Base-Acceleration (KBA) [16] and TAC Knowledge-Base-Population (KBP) [20] challenges. While they strongly motivate for the usage of temporal IE techniques, such as time annotations and inference via temporal constraints, none of these provide a solution for managing and querying this kind of data in a scalable DB architecture.

3. PRELIMINARIES

We start by briefly reviewing the key concepts from both temporal and probabilistic databases, which serve as the basis for our data model. That is, all the assumptions made in this section will hold also for our combined temporal-probabilistic database model, as it is defined in Sec. 4.

3.1 Temporal Database

We consider the *time domain* Ω^T as a linearly ordered, finite sequence of *time points*. A *time interval* consists of a contiguous and finite set of time points over Ω^T, which we denote by a half-open interval $[t_b, t_e)$ where $t_b, t_e \in \Omega^T, t_b < t_e$. We employ the two constants t_{min}, t_{max} to denote the earliest and latest time point in Ω^T, respectively[5]. A *temporal relation* R^T is represented as $R^T(A_1, \ldots, A_m, T_b, T_e)$, or equivalently as $R^T(A_1, \ldots, A_m)_{[T_b,T_e)}$, where R^T denotes the relation name, $A_1, \ldots, A_m$ is a finite set of attributes of R^T with domains Ω_i, and T_b, T_e are attributes denoting the begin and end time points of the time intervals that are attached to each tuple. This definition of temporal relations employs *tuple timestamping* [24], i.e., each tuple is annotated by a single time interval that specifies its valid-time. The *database schema* $\mathcal{S}$ then consists of all the relation schemas R_i^T. A *relation instance* $\mathbf{R}^T$ of schema R^T is a finite set of tuples that each contain for every A_i a value $a_i \in \Omega_i$, and for each pair T_b, T_e a time interval $[t_b, t_e) \subseteq \Omega^T$. Further, we will use $\mathcal{F}$ to denote the set of all *base tuples* (henceforth we refer to it as the set of "base facts") captured in all the input instances $\mathbf{R}_i^T$ over $\mathcal{S}$. These we will also call the *extensional relations*. Finally, we define a *temporal database* $(\mathcal{S}, \mathcal{F}, \Omega^T)$ as a triple consisting of a database schema $\mathcal{S}$, a finite set of base tuples $\mathcal{F}$, and the time domain Ω^T.

3.2 Probabilistic Database

In its most general form, a *probabilistic database* [33] is defined as a discrete probability distribution over a finite number of database instances (the "possible worlds"). In the case of a tuple-independent probabilistic database, this distribution can very compactly be described (i.e., be factorized) by attaching a probability value to each base fact f in $\mathcal{F}$. Formally, we thus define a *tuple-independent probabilistic database* $\mathcal{DB}^p = (\mathcal{S}, \mathcal{F}, p)$ as a triple consisting of a database schema $\mathcal{S}$, a finite set of base facts $\mathcal{F}$, and a *probability measure* $p : \mathcal{F} \rightarrow (0, 1]$, which assigns a probability value $p(f)$ to each uncertain base fact $f \in \mathcal{F}$. A *probabilistic relation* R^p is represented as $R^p(A_1, \ldots, A_m, p)$, where R^p denotes the relation name, $A_1, \ldots, A_m$ denote a finite set of attributes of R^p with domains Ω_i, and p is an attribute which denotes the probability value that is attached to each fact in a relation instance $\mathbf{R}^p$ of R^p. The probability value $p(f)$ denotes the confidence in the existence of the fact in the database, i.e.,

[5] We do not consider the finiteness of Ω^T to be of any limitation to our model in practice, since we can always choose t_{min}, t_{max} as the earliest and latest time points.

a higher value $p(f)$ denotes a higher confidence in f being valid. Formally, this uncertainty is modeled by associating a Boolean random variable with each base fact $f \in \mathcal{F}$. For ease of notation, we will replace the identifiers for the random variables by the fact identifiers.

Assuming independence among all the Boolean random variables associated with the base facts $\mathcal{F}$, the *probability* $P(\mathcal{W})$ of a *possible world* $\mathcal{W} \subseteq \mathcal{F}$ is then defined as follows.

$$P(\mathcal{W}) := \prod_{f \in \mathcal{W}} p(f) \prod_{f \notin \mathcal{W}} (1 - p(f)) \tag{5}$$

That is, in the absence of any further constraints restricting the possible worlds, any subset $\mathcal{W}$ of base facts in $\mathcal{F}$ forms a valid possible world (i.e., a possible instance) of the PDB. Hence, there are exponentially many such possible worlds.

3.3 Conditioning a Probabilistic Database

As illustrated above, there are intriguing analogies between probabilistic and temporal databases, which both employ annotations to relations to express at which time interval—or with which confidence—a fact is considered to be valid. We now review a third concept, called *conditioning* [22], which allows for additionally incorporating consistency constraints into a possible-worlds-based representation formalism.

Formally, we can compute the *marginal probability* of any Boolean formula ϕ over variables in $\mathcal{F}$ as the sum of the probabilities of all the possible worlds $\mathcal{W} \subseteq \mathcal{F}$ that entail ϕ, which is denoted as $\mathcal{W} \models \phi$.

$$P(\phi) := \sum_{\mathcal{W} \models \phi, \mathcal{W} \subseteq \mathcal{F}} P(\mathcal{W}) \tag{6}$$

Conditioning a formula ϕ on another formula ψ, which also ranges over variables in $\mathcal{F}$, then simply denotes the process of computing the following conditional probability:

$$P(\phi \mid \psi) := \frac{P(\phi \wedge \psi)}{P(\psi)} \tag{7}$$

Intuitively, conditioning removes the possible worlds from a probabilistic database, in which the constraints are not satisfied, and thus reweighs the remaining worlds to again form a probability distribution.

4. TEMPORAL-PROBABILISTIC DATABASE MODEL

We now combine and extend the above two, prevalent database models into a unified temporal-probabilistic database model in order to facilitate both rule-based and probabilistic inference over uncertain temporal knowledge. Our extensions employ the following two key concepts: *temporal deduction rules*, in the form of Datalog-style logical implications, allow us to represent all of the core operations known from relational algebra over these temporal and probabilistic relations, and *temporal consistency constraints*, in the form of denial constraints, allow us to constrain the possible worlds at which query answers are considered to be valid.

4.1 Temporal Deduction Rules

Temporal deduction rules are represented analogously to Datalog notation (see Rules (1) and (2) of Example 1). Syntactically, these rules have the shape of logical implications, with exactly one positive head literal and a conjunction of both positive and negative literals in the body.

DEFINITION 1. *A* temporal deduction rule *over a database schema $\mathcal{S}$ is a logical rule of the form*

$$R_0^T(\bar{X}_0)_{[T_{b_0}, T_{e_0}]} \leftarrow \begin{pmatrix} \bigwedge_{i=1,\ldots,n} & R_i^T(\bar{X}_i)_{[T_{b_i}, T_{e_i}]} \wedge \\ \bigwedge_{j=1,\ldots,m} & \neg R_j^T(\bar{X}_j)_{[T_{b_j}, T_{e_j}]} \wedge \\ & \Phi(\bar{X}) \end{pmatrix} \tag{8}$$

where:

(1) R_0^T denotes the head literal's intensional temporal relation in $\mathcal{S}$, while R_i^T, R_j^T refer to extensional or intensional temporal relations in $\mathcal{S}$;

(2) $n \geq 1$, $m \geq 0$, thus requiring at least one positive relational literal;

(3) $\Phi(\bar{X})$ is a conjunction of literals over the arithmetic predicates $=$, $\neq$, and $\leq^T$, whose arguments are in $\bar{X}$;

(4) $\bar{X}_0$, $\bar{X}_i$, $\bar{X}_j$ denote tuples of non-temporal variables and constants, such that $Var(\bar{X}_0) \subseteq \bigcup_i Var(\bar{X}_i)$ and $Var(\bar{X}_j) \subseteq \bigcup_i Var(\bar{X}_i)$;

(5) $\bar{X}$ denotes a tuple of both temporal and non-temporal variables and constants, such that $Var(\bar{X}) \subseteq \bigcup_i Var(\bar{X}_i) \cup \bigcup_i \{T_{b_i}, T_{e_i}\}$;

(6) T_{b_0}, T_{e_0} are the head's temporal arguments, such that $T_{b_0}, T_{e_o} \in \bigcup_i \{T_{b_i}, T_{e_i}\} \cup \{t_{min}, t_{max}\}$;

Predicates. From condition *(1)*, the predicates occurring in the head literals of all deduction rules define a set of so-called *intensional relations*, which must not overlap with any of the extensional ones. The predicates in the body literals of a deduction rule may however relate to both extensional and intensional relations. In *(2)*, we require at least one non-negated relational literal ($n \geq 1$) to exist. As denoted by *(3)*, we allow the two arithmetic predicates $=$ and $\neq$, and we additionally support the single *temporal arithmetic predicate* $\leq^T$ which compares the order of two time points in the sequence Ω^T.

Variables. The positive non-temporal variables $\bar{X}_i$ bind the non-temporal variables in the head literal $\bar{X}_0$ (see *(4)*), in the negated body literals $\bar{X}_j$ (see *(4)*), and in the literals relating to arithmetic predicates $\bar{X}$ (see *(5)*). Hence, our Datalog rules are always *safe* with respect to the non-temporal arguments [1]. Regarding temporal arguments, *(5)*, *(6)* require the temporal variables of the head literal and the arithmetic literals in the body of a rule to be bound by a positive relational literal in the body. We emphasize that the temporal arguments of a negated relational body literal, i.e., T_{b_j}, T_{e_j}, may be unbound. A proof of the safeness of our temporal arguments is available in Appendix A.

Expressiveness. With respect to the non-temporal arguments of the rules, we consider only *safe, non-recursive Datalog programs*, which allows us to capture all of the core operations that are expressible in relational algebra (apart from grouping and aggregations) [1]. Moreover, by utilizing conjunctions of $\leq^T$ predicates over the temporal arguments of each rule, we are able to express all the 13 relationships between time intervals defined by Allen [2], such as *overlaps*, *disjoint* or *startsWith*, via this form of interval arithmetic.

Lineage. Instead of evaluating the deduction rules over each individual possible world, we utilize *data lineage* to represent the logical dependencies between the base facts and the deduced facts, i.e., the query answers. In analogy to [13,

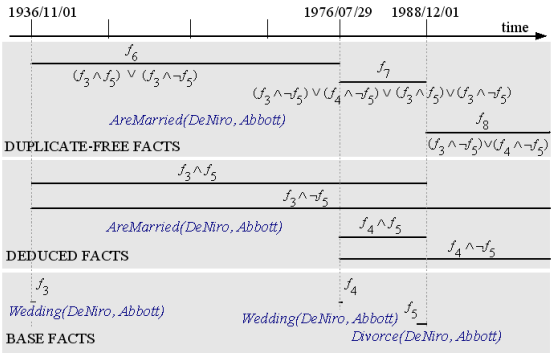


Figure 1: Deducing & deduplicating facts with time intervals

33], we consider lineage as a Boolean formula that relates each deduced fact with the base facts via the three Boolean connectives $\wedge$, $\vee$ and $\neg$, in order to reflect the semantics of the relational operations that were applied to derive that fact. Specifically, we employ

- a conjunction ($\wedge$) that connects the relational literals in the body of a deduction rule;
- a negation ($\neg$) for a negated relational literal in the body of a deduction rule;
- a disjunction ($\vee$) whenever the same fact is obtained from the head literals of two or more deduction rules.

We refer the reader to [13] for a formal definition of lineage in combination with Datalog rules. We write $\phi(f)$ to refer to the lineage of a fact f. Moreover, if f is a base fact, we set $\phi(f) := f$, where f stands for the Boolean (random) variable that represents this fact.

Grounding. In the following, we formally consider a grounding function G that, when applied to a conjunction of first-order literals Ψ, resolves the first-order literals to base facts or lineage formulas in the case of intensional relations. This is achieved by binding the first-order literals' variables to constants obtained from base facts $\mathcal{F}$.

Definition 2. *Let Ψ be a conjunctive first-order formula and $\mathcal{F}$ a set of facts. Then, the* grounding function $G(\Psi, \mathcal{F})$ *instantiates Ψ over $\mathcal{F}$ into a set of propositional lineage formulas $\{\psi_0, \ldots, \psi_n\}$.*

Example 2. *Applying G to the body of Rule* (2) *and the set of facts $\{f_3, f_4, f_5\}$ yields the lineage formulas $\{f_3 \wedge f_5, f_4 \wedge f_5\}$. By further instantiating Rule* (2)*'s head with the arguments of $f_4 \wedge f_5$, we obtain the new fact $f' = AreMarried(DeNiro, Abbott)_{[1976\text{-}07\text{-}29,1988\text{-}12\text{-}01)}$. Regarding the lineage, we have $\phi(f') = f_4 \wedge f_5$. All facts which can be deduced from Rules* (1) *and* (2) *are shown together with their lineages and time intervals in the middle part of Fig. 1.* $\diamond$

For a non-recursive (non-temporal) Datalog program $\mathcal{D}$, the iterative application of G over the rule bodies in $\mathcal{D}$ can be computed in polynomial time in $|\mathcal{F}|$ and $|\mathcal{D}|$ [1]. In Appendix B, we provide a polynomial-time reduction of our temporal deduction rules to non-temporal Datalog.

Duplicates. All four deduced facts in Fig. 1 state the marriage of *DeNiro* and *Abbott*, but with different lineages and overlapping time intervals. Following prior work on temporal DBs with data lineage [12], we next define *duplicate-free* temporal relations. We refer to the arguments of a fact f in a relation instance $\mathbf{R}^T$ as $R^T(\bar{a})_{[t_b,t_e)}$, where R^T is the relation name and $\bar{a}$ and t_b, t_e are constants denoting the non-temporal and temporal arguments of f, respectively.

Definition 3. *A temporal relation instance $\mathbf{R}^T$ is called a* duplicate-free relation *[12], if for all pairs of facts $f = R^T(\bar{a})_{[t_b,t_e)}$, $f' = R^T(\bar{a}')_{[t'_b,t'_e)}$ in $\mathbf{R}^T$ it holds that:*

$$\bar{a} = \bar{a}' \quad \Rightarrow \quad [t_b, t_e) \cap [t'_b, t'_e) = \emptyset$$

In other words, facts with equivalent non-temporal arguments must have non-overlapping time-intervals.

Deduplicating Facts. In order to convert a temporal relation with duplicates (as shown in the middle box of Fig. 1) into a duplicate-free temporal relation (as shown on the top of Fig. 1), we provide the following definition.

Definition 4. *Let a temporal relation R^T, non-temporal constants $\bar{a}$, a time-point $t \in \Omega^T$, and a set of facts $\mathcal{F}$ be given. Then, L is defined as the set of lineages of fact $R^T(\bar{a})$ that are valid at time point t:*

$$L(R^T, \bar{a}, t, \mathcal{F}) := \{\phi(f) \mid f = R^T(\bar{a})_{[t_b,t_e)} \in \mathcal{F}, t_b \leq t < t_e\}$$

We create duplicate free facts $f' = R^T(\bar{a})_{[t_b,t_e)}$, such that for any pair of time-points $t_0, t_1 \in [t_b, t_e)$ it holds that:

$$L(R^T, \bar{a}, t_0, \mathcal{F}) = L(R^T, \bar{a}, t_1, \mathcal{F}) \tag{9}$$

Furthermore, we define the new facts' lineage to be:

$$\phi(f') := \bigvee_{\psi_i \in L(R^T, \bar{a}, t_b, \mathcal{F})} \psi_i \tag{10}$$

In other words, for a given relation instance and a fact's non-temporal arguments, L is the set of all facts' lineages that share the same non-temporal arguments and which are valid at time point t. In Eq. (9), we require a new duplicate-free fact to have identical lineage sets L at all time points in its time interval. We remark that for the equality check of Eq. (9), we focus on syntactical equivalence checks between the lineage formulas obtained from our grounding algorithm. We however refrain from logical equivalence checks between lineage formulas (which are $\mathcal{NP}$-hard). This definition is similar to change preservation in [12], which also iteratively compares sets of lineages along a query plan.

Example 3. *Applying Definition 4 to the facts in the middle of Fig. 1 yields the facts shown at the top of the figure. For instance, if we inspect the time points* 1976-07-28 *and* 1976-07-29*, we notice that $\{f_3 \wedge f_5, f_3 \wedge \neg f_5\} \neq \{f_3 \wedge f_5, f_3 \wedge \neg f_5, f_4 \wedge f_5, f_4 \wedge \neg f_5\}$, so two differing facts f_6 and f_7 have to be kept in the relation. Thus, the resulting duplicate-free facts are the following.*

Id	*Fact*	T	p
f_6	$AreMarried(DeNiro, Abbott)$	[1936-11-01, 1976-07-29)	0.3
f_7	$AreMarried(DeNiro, Abbott)$	[1976-07-29, 1988-12-01)	0.79
f_8	$AreMarried(DeNiro, Abbott)$	[1988-12-01, t_{max})	0.158

Following Eq. (10)*, their lineages are as follows.*

$$\phi(f_6) = (f_3 \wedge f_5) \vee (f_3 \wedge \neg f_5)$$
$$\phi(f_7) = (f_3 \wedge f_5) \vee (f_3 \wedge \neg f_5) \vee (f_4 \wedge f_5) \vee (f_4 \wedge \neg f_5)$$
$$\phi(f_8) = (f_3 \wedge \neg f_5) \vee (f_4 \wedge \neg f_5) \diamond$$

An algorithmic approach to obtain the above facts and time intervals is described in Sec. 6.2, while confidence computations based on lineage are discussed in Sec. 6.3.

4.2 Temporal Consistency Constraints

We now introduce a second type of rules to our setting, referred as *temporal consistency constraints*. Syntactically, these constraints resemble denial constraints [5] in non-temporal databases (see Constraints (3) and (4) of Example 1). Consistency constraints do not generate new facts, but they condition the marginal (i.e., posterior) probabilities of both the base and the derived facts from the deduction rules.

DEFINITION 5. *A* temporal consistency constraint *over a database schema $\mathcal{S}$ is a logical rule of the form*

$$\neg\left(\bigwedge_i R_i^T(\bar{X}_i)_{[T_{b_i},T_{e_i})} \ \wedge \Phi(\bar{X})\right)$$

where:

(1) R_i^T denote extensional or intensional temporal relations in $\mathcal{S}$;

(2) $\Phi(\bar{X})$ is a conjunction of literals relating to the arithmetic predicates $=$, $\neq$, and $\leq^T$ with arguments in $\bar{X}$;

(3) The $\bar{X}_i$ denote tuples of non-temporal variables and constants, and T_{b_i}, T_{e_i} are temporal variables and constants, such that $Var(\bar{X}) \subseteq \bigcup_i (Var(\bar{X}_i) \cup \{T_{b_i}, T_{e_i}\})$.

Predicates and Variables Constraints may range over both intensional and extensional temporal relations (see *(1)*). In Constraint (4), for example, *AreMarried* refers to an intensional relation. Similarly to the deduction rules, we allow conjunctions of literals over the arithmetic predicates $=$, $\neq$, and $\leq^T$ (see *(2)*). Additionally, *(3)* enforces that all variables of arithmetic literals, i.e., the ones contained in $\bar{X}$, must also be bound by at least one relational literal. For example, in the Constraint (3), the variables of the arithmetic predicate $(X = Y)$ are both arguments of a relational *BornIn* literal as well. We implicitly assume all-quantification of all variables.

Conditioning. Similar to [22], we apply conditioning to adjust the probabilities of query answers with respect to our temporal consistency constraints. However, as opposed to [22], we include conditioning into the query processing step which is applied after grounding the deduction rules.

DEFINITION 6. *Let $\mathcal{C} = \{C_0, \ldots, C_n\}$ be a set of temporal consistency constraints, and let $\mathcal{F}^+$ be a set containing both base facts $\mathcal{F}$ and facts deducible from a set of temporal deduction rules $\mathcal{D}$. Then, we define C to be the conjunction of all grounded constraints over $\mathcal{F}^+$.*

$$C := \bigwedge_{\psi_i \in G(C_i,\mathcal{F}^+), C_i \in \mathcal{C}} \psi_i \tag{11}$$

The confidence of a fact $f \in \mathcal{F}^+$ with lineage $\phi(f)$ and with respect to the grounded constraints C is then calculated as:

$$P(\phi(f) \mid C) := \frac{P(\phi(f) \wedge C)}{P(C)} \tag{12}$$

If no constraints have been returned by the grounding function G, we define $C := \mathit{true}$. Consequently, we obtain $P(\phi(f) \wedge C) = P(\phi(f))$ and $P(C) = P(\mathit{true}) := 1$, such that we fall back to the unconditioned semantics. Also, if the constraints are unsatisfiable (i.e., no possible worlds exist over which the conjunction C is satisfied), then we define $C := \mathit{false}$ and $P(\phi(f)) := 0$ for all facts $f \in \mathcal{F}^+$. We remark that in grounding the consistency constraints, we are making the same closed-world assumption that we also make for grounding the deduction rules, and which is also common in Datalog. That is, all relational atoms that do not match any of the grounded facts are assumed to be *false*.

EXAMPLE 4. *Grounding the Constraint* (3) *against the facts of Example 1 yields $\neg(f_1 \wedge f_2)$, where the arithmetic literal is omitted from the grounded formula since it evaluates to* true. *Altogether, grounding Constraints* (3) *and* (4) *results in $C = \neg(f_1 \wedge f_2) \wedge \neg(f_1 \wedge \phi(f_6)) \wedge \neg(f_2 \wedge \phi(f_6)) \wedge \neg(f_2 \wedge \phi(f_7))$, where all arithmetic predicates have already been evaluated correspondingly.* $\diamond$

4.3 Temporal-Probabilistic Database

By gathering all the preconditions we established, we now define a temporal-probabilistic database as follows.

DEFINITION 7. *A* temporal-probabilistic database $\mathcal{DB}^{Tp} = (\mathcal{S}, \mathcal{F}, \mathcal{D}, \mathcal{C}, \Omega^T, p)$ *is a six-tuple consisting of a database schema $\mathcal{S}$, a set of base facts $\mathcal{F}$, a set of temporal deduction rules $\mathcal{D}$, a set of temporal consistency constraints $\mathcal{C}$, a time domain Ω^T, and a probability measure p.*

In Sec. 5, we investigate the basic properties of this model.

4.4 Queries

In analogy to the afore defined rules, we consider a *query* over a temporal-probabilistic database $\mathcal{DB}^{Tp}$ as a conjunction of literals, each of which may relate to both relational and arithmetic predicates. Thus, for a query, similar conditions as for the deduction rules regarding safeness hold. That is, every non-temporal variable that occurs in a negated relational literal or in an arithmetic literal must also occur in at least one of the positive relational literals. Every temporal variable that occurs in an arithmetic literal must also occur in at least one of the relational literals.

EXAMPLE 5. *Using Example 1, we can query for the following conjunction of literals, thus asking for pairs of facts about the birth places of married people:*

$$\mathit{BornIn}(X,Y)_{[T_{b_1},T_{e_1})} \wedge \mathit{AreMarried}(X,Z)_{[T_{b_2},T_{e_2})}$$

5. PROPERTIES

In this section, we describe the properties of our probabilistic-temporal database model with respect to grounding, confidence computations, and its representational power.

THEOREM 1. *For a temporal-probabilistic database $\mathcal{DB}^{Tp} = (\mathcal{S}, \mathcal{F}, \mathcal{D}, \mathcal{C}, \Omega^T, p)$ with a fixed set of temporal deduction rules $\mathcal{D}$, grounding $\mathcal{D}$ has polynomial data complexity in $|\mathcal{F}|$ and $|\Omega^T|$.*

The proof is available in Appendix B and provides a reduction to non-temporal Datalog with inequalities. The reduction also properly preserves the lineages of the derived facts.

COROLLARY 1. *For a temporal-probabilistic database $\mathcal{DB}^{Tp} = (\mathcal{S}, \mathcal{F}, \mathcal{D}, \mathcal{C}, \Omega^T, p)$ with a fixed set of temporal deduction rules $\mathcal{D}$ and temporal consistency constraints $\mathcal{C}$, grounding $\mathcal{C}$ has polynomial data complexity in $|\mathcal{F}|$ and $|\Omega^T|$.*

Due to space constraints, we resort to sketching this proof. Formally, we may ground all constraints according to Eq. (11) by utilizing the polynomial-time grounding function G (see Definition 2). Since $|F^+|$ (see Definition 6) is polynomial in $|\mathcal{F}|$ and $|\Omega^T|$ (due to Theorem 1), C can be constructed in polynomial time in $|\mathcal{F}|$ and $|\Omega^T|$.

LEMMA 1. *For a temporal-probabilistic database* $\mathcal{DB}^{Tp} = (\mathcal{S}, \mathcal{F}, \mathcal{D}, \mathcal{C}, \Omega^T, p)$ *with a fixed set of temporal deduction rules* $\mathcal{D}$ *and temporal consistency constraints* $\mathcal{C}$*, confidence computations are* $\#\mathcal{P}$*-hard in* $|\mathcal{F}|$ *and* $|\Omega^T|$*.*

We sketch this proof by encoding a $\#\mathcal{P}$ query [9, 33] for a tuple-independent (but non-temporal) probabilistic database in our data model. First, we assign every tuple the same time-interval. Then, we translate the relational query into Datalog rules as in [1]. By grounding these rules against the temporal-probabilistic database, we receive the same lineage formulas as in the original probabilistic database, where confidence calculations are known to be $\#\mathcal{P}$-hard.

THEOREM 2. *A temporal-probabilistic database* $\mathcal{DB}^{PT} = (\mathcal{S}, \mathcal{F}, \mathcal{D}, \mathcal{C}, \Omega^T, p)$ *with lineage is* closed *and* complete *under all algebraic operations which are expressible by the temporal deduction rules.*

Generally, a representation system is *complete*, if it can represent any finite instance of data, which is in our case temporal and probabilistic. Since completeness implies closure [33], we provide a proof for the completeness of our temporal-probabilistic database model in Appendix C.

6. ALGORITHMS

In this section, we describe our algorithms for data and confidence computations in temporal-probabilistic databases. Specifically, we describe the grounding step, the deduplication of facts with overlapping time intervals, lineage decompositions, and the final confidence computation step.

6.1 Grounding

We first cover how we ground deduction rules and constraints. Our main grounding procedure (Algorithm 1) is

Algorithm 1 Ground($\mathcal{F}, \mathcal{D}, \mathcal{C}$)

Input: Facts $\mathcal{F}$, deduction rules $\mathcal{D}$, constraints $\mathcal{C}$
Output: Closure of facts $\mathcal{F}^+$ with lineage, grounded constraints C

1: Plan := Map[$R^T \to \{d \mid d \in \mathcal{D}, \mathit{headRelation}(d) = R^T\}$]
2: $\mathcal{F}^+ := \mathcal{F}$ ▷ Contains base and deduced facts.
3: **for** $R^T \in \mathit{Keys}$(Plan) in bottom-up order **do**
4: $\quad \mathcal{F}_{R^T} := \emptyset$ ▷ Facts in relation R^T.
5: $\quad$ **for** $\mathit{body} \to \mathit{head} \in$ Plan(R^T) **do**
6: $\quad\quad \mathcal{F}_{R^T} := \mathcal{F}_{R^T} \cup \mathit{Instantiate}(\mathit{head}, G(\mathit{body}, \mathcal{F}^+))$ ▷ Def. 2
7: $\quad$ **for** $\bar{a} \in \{\bar{a} \mid R^T(\bar{a})_{[T_b,T_e)} \in \mathcal{F}_{R^T}\}$ **do**
8: $\quad\quad \mathcal{F}^+ := \mathcal{F}^+ \cup$ Deduplicate($\mathcal{F}_{R^T,\bar{a}}$) ▷ Alg. 2
9: $C := \mathit{true}$ ▷ All facts are deduced now.
10: **for** $c \in \mathcal{C}$ **do**
11: $\quad C := C \wedge \bigwedge_{g \in G(c,\mathcal{F}^+)} g$ ▷ Eq. (11)
12: **return** $\mathcal{F}^+$, C

divided into two phases. The first phase (Lines 1-8) grounds all deduction rules in a bottom-up manner while tracing the lineage of derived facts. The second phase (Lines 9-11) instantiates constraints against all facts (i.e., both the base and the deduced facts).

In more detail, in Line 1 we use a map from intensional relations (keys) to the sets of deduction rules with the corresponding relational predicate as head literal (values). Next, the loop of Line 3 iterates over the intensional relations in a bottom-up manner [1]. That is, intensional relations, whose rules have only extensional relations in their bodies, are processed first. Then, we iterate over all deduction rules with relation R^T (Line 5) and instantiate their head literals into new facts by using the variable bindings obtained from the grounded body literals $G(\mathit{body}, \mathcal{F}^+)$ (see Definition 2). The following loop (Line 7) iterates over all tuples of non-temporal arguments $\bar{a}$ and calls Deduplicate($\mathcal{F}_{R^T,\bar{a}}$) for each. The call makes the set facts of $\mathcal{F}_R$ conform with Definition 3 and is described in Sec. 6.2. Here, $\mathcal{F}_{R^T,\bar{a}}$ are all facts in $\mathcal{F}_R^T$ whose non-temporal arguments are $\bar{a}$, or formally $\mathcal{F}_{R^T,\bar{a}} := \{f \mid f = R^T(\bar{a}', \bar{t}) \in \mathcal{F}_R^T, \bar{a}' = \bar{a}\}$. Finally, in Line 11 we produce the conjunction of the groundings of all constraints $c \in \mathcal{C}$ against the base and the deduced facts $\mathcal{F}^+$. The runtime complexity of Algorithm 1 is polynomial in $|\mathcal{F}|$, since Lines 1 to 8 correspond to grounding a non-recursive Datalog program, and in Lines 9 to 11 the function G runs in polynomial time in $|\mathcal{F}^+|$ and $|c|$.

EXAMPLE 6. *We initialize Algorithm 1 using* $\mathcal{F} := \{f_1, \ldots, f_5\}$*,* $\mathcal{D}$ *comprising Rules* (1) *and* (2)*, and* $\mathcal{C}$ *containing Constraints* (3) *and* (4) *of Example 1. First, we initialize* Plan *with* [*AreMarried* $\to \{(1), (2)\}$]*, and we set* $\mathcal{F}^+ = \{f_1, \ldots, f_5\}$*. The loop in Line 3 performs only one iteration where* $R^T =$ *AreMarried. Assuming that the loop in Line 5 selects* (2) *first, we instantiate the head to two new facts* $\mathit{AreMarried}(\mathit{DeNiro}, \mathit{Abbott})_{[1936-11-01,1988-12-01)}$ *and* $\mathit{Marriage}(\mathit{DeNiro}, \mathit{Abbott})_{[1976-07-29,1988-12-01)}$ *with lineages* $f_3 \wedge f_5$ *and* $f_4 \wedge f_5$*, respectively. Their lineage and that of the two facts deduced by Rule* (1) *are shown in Example 3. Fig. 1's top part corresponds to the output of* Deduplicate *(Line 8). When we reach Line 9,* $\mathcal{F}^+ = \{f_1, \ldots, f_8\}$*. Next, the constraints are grounded as shown already in Example 4.*

6.2 Deduplicating Facts

Algorithm 2 outputs, for a given set of deduced facts, duplicate-free facts as described in Definition 4. Fig. 1 again illustrates this by our running example. For a set of facts $\mathcal{F}_{R^T,\bar{a}}$ with relation R^T and non-temporal arguments $\bar{a}$ it performs a loop over the limits of the facts' time intervals. In

Algorithm 2 Deduplicate($\mathcal{F}_{R^T,\bar{a}}$)

Input: Facts $\mathcal{F}_{R^T,\bar{a}}$ of relation R^T, non-temporal arguments $\bar{a}$
Output: Deduplicated facts according to Definition 4

1: Limits := $\{t_b, t_e \mid R^T(\bar{a})_{[t_b,t_e)} \in \mathcal{F}_{R,\bar{a}}\}$
2: Begin := Map[$t \to \{f \mid f = R^T(\bar{a})_{[t,t_e)} \in \mathcal{F}_{R^T,\bar{a}}\}$]
3: End := Map[$t \to \{f \mid f = R^T(\bar{a}, t_b, t) \in \mathcal{F}_{R^T,\bar{a}}\}$]
4: $t_{last} := \mathit{first}$(Limits)
5: Active := $\emptyset$
6: Result := $\emptyset$
7: **for** $t \in$ Limits in ascending order **do**
8: $\quad$ **if** Active $\neq \emptyset$ **then**
9: $\quad\quad f_{new} := R^T(\bar{a})_{[t_{last},t)}$
10: $\quad\quad \phi(f_{new}) := \bigvee_{f \in \text{Active}} \phi(f)$ ▷ Eq. (10)
11: $\quad\quad$ Result := Result $\cup \{f_{new}\}$
12: $\quad\quad t_{last} := t$
13: $\quad\quad$ Active$_{last}$:= Active
14: $\quad$ Active := (Active $\setminus$ End(t)) $\cup$ Begin(t)
15: **return** Result

more detail, the set *Limits* contains all endpoints of intervals in $\mathcal{F}_{R^T,\bar{a}}$. *Begin* and *End* are maps pointing from a time point t to the set of facts beginning from t and ending at t, respectively. During the execution of the loop (Line 7), the set *Active* contains all facts whose interval contains $[t_{last}, t)$

(Line 14). In Lines 9 and 10, we produce a new fact f_{new} whose lineage is the disjunction of the lineage of all facts in *Active*. The runtime complexity of Algorithm 2 is in $O(|\mathcal{F}_{R^T,\bar{a}}|log|\mathcal{F}_{R^T,\bar{a}}|)$, since the loop requires the sorting of *Limits*. The worst-case size of the output is $2 \cdot |\mathcal{F}_{R^T,\bar{a}}| - 1$, which occurs when all facts' time-intervals are stacked.

EXAMPLE 7. *We apply Algorithm 2 to the facts deduced by Rules* (1) *and* (2)*, hence constructing the lineage for the facts presented in Example 3. First, we initialize* Limits *as* $\{1936\text{-}11\text{-}01, 1976\text{-}07\text{-}29, 1988\text{-}12\text{-}01, t_{max}\}$*, set* Begin *to* $[1936\text{-}11\text{-}01 \rightarrow \{f_3 \wedge f_5, f_3 \wedge \neg f_5\}, 1976\text{-}07\text{-}29 \rightarrow \{f_4 \wedge f_5, f_4 \wedge \neg f_5\}]$*, and assign* $[1988\text{-}12\text{-}01 \rightarrow \{f_3 \wedge f_5, f_4 \wedge f_5\}, t_{max} \rightarrow \{f_3 \wedge \neg f_5, f_4 \wedge \neg f_5\}]$ *to* End*. In the first iteration of the loop,* Active *is empty. In the next iteration, we add* $f_6 = AreMarried(DeNiro, Abbott)_{[1936\text{-}11\text{-}01,1976\text{-}07\text{-}29)}$ *to* Result *and set its lineage to* $\phi(f_6) = (f_3 \wedge f_5) \vee (f_3 \wedge \neg f_5)$*. The last two iterations produce* f_7*,* f_8 *as shown in Example 3.* $\diamond$

6.3 Confidence Computations

Having obtained the lineage from the previously discussed grounding steps, we now calculate the confidence of a fact at each time interval by following a number of techniques known from probabilistic databases [33]. We first define the function $F(\phi)$ to deliver the set of base facts that are contained in a lineage formula ϕ, e.g., $F(f_1 \wedge f_2) = \{f_1, f_2\}$. Next, we define $P(\phi)$ over the structure of ϕ as follows:

Definition	Condition
$P(f) := p(f)$	$f \in \mathcal{F}$
$P(\bigwedge_i \phi_i) := \prod_i P(\phi_i)$	$i \neq j \Rightarrow F(\phi_i) \cap F(\phi_j) = \emptyset$
$P(\bigvee_i \phi_i) := 1 - \prod_i (1 - P(\phi_i))$	$i \neq j \Rightarrow F(\phi_i) \cap F(\phi_j) = \emptyset$
$P(\phi \vee \psi) := P(\phi) + P(\psi)$	$\phi \wedge \psi \equiv false$
$P(\neg\phi) := 1 - P(\phi)$	

(13)

The first line captures the case of a base fact, for which we return its attached probability. The next two lines handle *independent-and* and *independent-or* operations for conjunctions and disjunctions of disjoint sets of base facts, respectively. Then, we handle disjunctions for sub-formulas that denote probabilistically disjoint events (known as *disjoint-or* [33]). The last line handles negation.

EXAMPLE 8. *In Example 3 the lineage of* f_6 *was defined as* $(f_3 \wedge f_5) \vee (f_3 \wedge \neg f_5)$*. Now, let us compute its confidence:*

$$\begin{aligned} &P((f_3 \wedge f_5) \vee (f_3 \wedge \neg f_5)) \\ &= P(f_3 \wedge f_5) + P(f_3 \wedge \neg f_5) \\ &= P(f_3) \cdot P(f_5) + P(f_3) \cdot (1 - P(f_5)) \\ &= 0.3 \cdot 0.8 + 0.3 \cdot 0.2 = 0.3 \end{aligned}$$

The first equality utilizes a disjoint-or operation, followed by applications of the definitions for independent-and and negation. Last, it looks up the confidence values from Example 1.

Eq. (13)'s definition of $P(\phi)$ runs in linear time in the size of ϕ. However, in general, computing $P(\phi)$ is $\#\mathcal{P}$-hard [9, 33] and this becomes evident if we consider Eq. (14), called *Shannon expansion*, which is always applicable:

$$P(\phi) := P(f) \cdot P(\phi_{[f \rightarrow true]}) + (1 - P(f)) \cdot P(\phi_{[f \rightarrow false]}) \quad (14)$$

Here, the notation $\phi_{[f \rightarrow true]}$ for a fact $f \in F(\phi)$ denotes that we replace all occurrences of f in ϕ by *true* (and *false*, respectively). Shannon expansion is based on the equivalence

$$\phi \equiv (f \wedge \phi_{[f \rightarrow true]}) \vee (\neg f \wedge \phi_{[f \rightarrow false]}) \quad (15)$$

such that the disjunction fulfills the disjoint-or condition with respect to f. Repeated applications of Shannon expansions may result in an exponential increase of ϕ.

EXAMPLE 9. *If we apply Eq.* (13) *and Eq.* (14) *to calculate the confidences of* f_1*,* f_2 *(as of Example 1) and* f_6*,* f_7*,* f_8 *(from Example 3) by following Eq.* (12)*, while respecting the constraints of Example 4 , we obtain:*

	f_1	f_2	f_6	f_7	f_8
P	0.827	0.041	0.039	0.68	0.136

Comparing these confidences to their unconditioned version, thus ignoring the constraints (see Example 1 and Example 3), we note that the (in reality) incorrect facts f_2*,* f_6*, and* f_8 *now indeed have significantly lower confidences (an effect we exploit in the experiments in Sec. 7.1).*

6.4 Lineage Decompositions

In practice, the necessity for Shannon expansions hinders the scalability of the confidence computation step, for which we can do exact probabilistic inference, to larger data sets. Hence, we propose two methods for lineage decomposition (executed before invoking the confidence computations), which drastically reduce the number of Shannon expansions. Before describing our algorithm, we provide a function for determining the number of required Shannon expansions at the top-level operator of a lineage formula ϕ.

DEFINITION 8. *Let* $\phi \equiv \phi_0$ *op* ... *op* ϕ_n *be a lineage formula with operator op* $\in \{\wedge, \vee\}$*, and let* ϕ_0 *to* ϕ_n *be sub-formulas of* ϕ*. Then, we define the function* S *to return the number of necessary Shannon expansions as follows:*

$$S(\phi) = |\{f \mid \exists i \neq j : f \in F(\phi_i), f \in F(\phi_j)\}|$$

The above definition applies to $\wedge, \vee$ only, because $\neg$ cannot induce new Shannon expansions. Moreover, if $S(\phi) = 0$ then Eq. (13)'s second or third line is applicable. If we attempt to calculate $P(\phi)$ naively, then $S(\phi)$ characterizes the exponent added by ϕ's top level operator to $P(\phi)$'s runtime.

EXAMPLE 10. *As an example, we apply* S *to* $\phi(f_7)$ *(see Example 3). That is* $op = \vee$ *and* $S((f_3 \wedge f_5) \vee (f_3 \wedge \neg f_5) \vee (f_4 \wedge f_5) \vee (f_4 \wedge \neg f_5)) = |\{f_3, f_4, f_5\}| = 3$*. Thus, a naive computation of* $P(\phi(f_7))$ *takes* 2^3 *iterations.*

Algorithm 3 implements two strategies for lowering the number of Shannon expansions, which are repeatedly invoked by the loop in Line 1. Empirically, we found that reducing the Shannon expansions outperforms the alternative of eagerly removing all of them, which is done in comparable knowledge compilation techniques [19, 27]. This is why we employ the threshold θ (set to 4 in our evaluation). The first strategy (Lines 2-7) aims to identify sets S_i of lineage subformulas ϕ'_0 to ϕ'_n such that there are no Shannon expansions between each pair of sets S_i, S_j (as enforced by $F(S_i) \cap F(S_j) = \emptyset$). If the condition in Line 6 returns *true*, then $S(op_i(op_{\phi'_j \in S_i} \phi'_j)) = 0$, which makes Eq. (13)'s second or third line applicable. Furthermore, $\sum_i S(op_{\phi'_j \in S_i} \phi'_j) = S(\phi')$ which provides an exponential speed-up for the confidence computations since $2^{S(\phi')} > \sum_i 2^{S(op_{\phi'_j \in S_i} \phi'_j)}$. If the condition in Line 6 evaluates to *false*, i.e., $m = 0$, we invoke the second strategy to remove

Algorithm 3 Decompose(ϕ, θ)

Input: Lineage formula ϕ, threshold θ
Output: Adapted formula ϕ with reduced Shannon expansions

```
1: while ∃φ' ∈ φ : S(φ) > θ do
2:     Select φ' ∈ φ
3:       s.t. φ' ≡ φ'_0 op ... op φ'_n, op ∈ {∧,∨}, S(φ') > θ
4:     Determine S_0,...,S_m, S_i ⊆ {φ'_0,...,φ'_n}
5:       s.t. ⋃̇_i S_i = {φ'_0,...,φ'_n}, ∀i ≠ j : F(S_i) ∩ F(S_j) = ∅
6:     if m > 0 then
7:         replace φ' in φ by op_i(op_{φ'_j ∈ S_i} φ'_j)
8:     else
9:         f := arg max_f |{φ'_i | f ∈ F(φ'_i), 0 ≤ i ≤ n}|
10:        replace φ' in φ by (f ∧ φ'_[f→true]) ∨ (¬f ∧ φ'_[f→false])
11: return φ
```

Shannon expansions. In Line 9, we pick the fact f which occurs in the maximal number of lineage subformulas ϕ'_0 to ϕ'_n, such that $\phi_{[f\to true]}$, $\phi_{[f\to false]}$ have high chances to be simplified. Then, in Line 10, we materialize a Shannon expansion for f by following Eq. (15).

The inner part of the while loop can be implemented in (almost) linear time in the size of ϕ by utilizing the *union-find* data structure for Line 4. In theory, there are cases where the while loop is executed an exponential number of times, but our experiments show great performance speed-ups achieved by Algorithm 3.

EXAMPLE 11. *We decompose Example 3's* $\phi(f_7) = (f_3 \wedge f_5) \vee (f_3 \wedge \neg f_5) \vee (f_4 \wedge f_5) \vee (f_4 \wedge \neg f_5)$ *by Algorithm 3 with* $\theta = 0$. *As discussed in Example 10,* $S(\phi(f_7)) = 3$, *so we set* $\phi' = \phi(f_7)$. *In Line 4, we receive* $m = 0$ *because* f_5 *occurs in all lineage subformulas. Hence, in Line 9, we choose* f_5 *and rewrite* $\phi(f_7)$ *to* $(f_5 \wedge (f_3 \vee f_4)) \vee (\neg f_5 \wedge (f_3 \vee f_4))$ *in Line 10. Now, all Shannon expansions are gone, yielding* $P(\phi(f_7)) = (0.8 \cdot (1 - (1 - 0.3) \cdot (1 - 0.7))) + (0.2 \cdot (1 - (1 - 0.3) \cdot (1 - 0.7))) = 0.8$ *which can be computed via Eq.* (13).

7. EXPERIMENTS

Our evaluation focuses on four different aspects. First, we compare the quality of our extraction and inference techniques against state-of-the-art IE systems. Next, we focus on our runtime performance for query answering tasks over a large temporal-probabilistic database instance, before we study how we perform for the KB building task by materializing large queries. Finally, we analyze the detailed runtimes of the individual algorithmic tasks involved in these steps.

Setup. We implement our engine (coined *TPDB*) in Java 7. TPDB employs a PostgreSQL 8.4 database for grounding conjunctive queries (i.e., queries, bodies of rules, and consistency constraints) via SQL statements and storing both the extensional relations and the intermediate facts. We translate the "$\leq^T$" predicate into its SQL counterpart, using "$<$" over time points encoded as integers. Since computing confidences is a major bottleneck for a PDB, we perform lineage decompositions and probabilistic inference (see Sec. 6.4) entirely in main memory. We run all experiments on a 8-core Intel Xeon 2.4Ghz machine with 48 GB RAM, repeat each query on each setting four times and report the average of the last three runs. Due to space constraints, all queries, rules, and constraints are provided as supplementary material[6].

[6] http://www.mpi-inf.mpg.de/~mdylla/tpdbSupplementary.pdf

7.1 Extraction Quality

We first compare TPDB against four state-of-the-art methods for temporal fact extraction, two employing constraint solving via ILP and two using probabilistic inference via MLNs [29]. For ILP we use the *Gurobi* software[7] implementing the constraints from [34, 37] as well as cutting plane inference [30] from "Markov thebeast"[8] (*Beast*), which both perform MAP inference, i.e., they return the single most likely possible world, rather than computing marginals as TPDB and MLNs do. As for MLNs, we compare against *Alchemy*[9] and *Tuffy*[10], that latter of which improves the grounding step of Alchemy by using a PostgreSQL optimizer [26]. Since ILPs and MLNs do not natively support time, we encode time intervals by adding two arguments on the relational predicates denoting their begin and end time points. All systems operate on the same set of deduction rules and consistency constraints.

Data Set and Ground Truth. Our dataset consists of 1,817 base facts which we extracted from free-text biographies of 272 celebrities crawled from Wikipedia, `imdb.com`, and `biography.com`. The facts correspond to nine temporal relations, namely *AttendedSchool*, *Born*, *Died*, *Divorce*, *IsDating*, *Founded*, *GraduatedFrom*, *MovedTo*, and *Wedding*. Our extractor uses textual patterns from [25] to recognize potential facts, regular expressions to find dates, and the Stanford NER tagger[11] to identify named entities. We assigned each pattern and extraction step a probability, such that the confidence of each extracted fact is obtained by multiplying the probabilities of all steps involved in its extraction process. As for the ground truth, we manually labeled 100 randomly chosen facts from each relation by determining their correct time intervals from the text and labeled them as false if the extracted fact was erroneous. We equally divided the labeled facts into a training set (for developing deduction rules and constraints) and a test set (used for precision and recall). We employed a set of 11 hand-crafted temporal deduction rules and 21 temporal consistency constraints, including the ones shown in Example 1. The free-text sources, both the extraction rules and facts are available online[6].

Metrics. To evaluate the result quality, our precision and recall metrics should reflect the overlap between the obtained facts' time intervals and the time intervals of the ground-truth. For this, we define a function V that, for a set of facts in a temporal relation instance $\mathbf{R}^T$ with identical non-temporal arguments $\bar{a}$ and a probability threshold θ_p, returns the set of time points at which these facts are valid with a probability of at least θ_p.

$$V(\mathbf{R}^T, \bar{a}, \theta_p) := \left\{ t \;\middle|\; \begin{array}{c} f = R^T(\bar{a})_{[t_b,t_e)} \in \mathbf{R}^T, \\ t \in [t_b, t_e),\ p(f) \geq \theta_p \end{array} \right\}$$

We now define precision and recall as follows.

$$\mathit{Precision}(\mathbf{R}^T, \bar{a}, \theta_p) := \frac{|V(\mathbf{R}^T, \bar{a}, \theta_p) \cap V(\mathbf{R}^T_{truth}, \bar{a}, \theta_p))|}{|V(\mathbf{R}^T, \bar{a}, \theta_p)|}$$

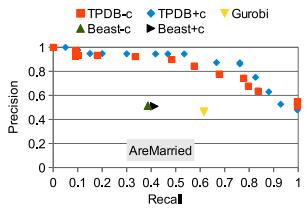
$$\mathit{Recall}(\mathbf{R}^T, \bar{a}, \theta_p) := \frac{|V(\mathbf{R}^T, \bar{a}, \theta_p) \cap V(\mathbf{R}^T_{truth}, \bar{a}, \theta_p))|}{|V(\mathbf{R}^T_{truth}, \bar{a}, \theta_p)|}$$

For example, consider the ground truth fact f_g : *Divorce*(*DeNiro*, *Abbott*)$_{[1988-09-01,1988-09-02)}$, f_5 from Example 1,

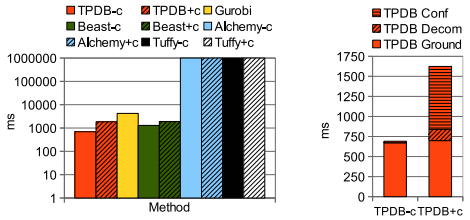
[7] http://www.gurobi.com
[8] https://code.google.com/p/thebeast/
[9] http://alchemy.cs.washington.edu/
[10] http://hazy.cs.wisc.edu/hazy/tuffy/download/
[11] http://nlp.stanford.edu/ner/

	TPDB-c	TPDB+c	Gurobi	Beast-c	Beast+c
AreMarried	0.76	**0.81**	0.52	0.44	0.46
AttendedSchool	**0.72**	**0.72**	0.54	0.66	0.68
Born	0.83	0.84	**0.87**	0.80	0.80
Died	**0.70**	0.62	0.48	0.46	0.53
Founded	**0.80**	**0.80**	0.38	0.73	**0.80**
GraduatedFrom	**0.74**	**0.74**	0.70	0.68	0.67
IsDating	0.62	**0.66**	0.54	0.51	0.50
MovedTo	0.75	0.77	**0.79**	0.69	0.74
Average	0.74	**0.75**	0.60	0.62	0.65

(a) F_1 measure (best θ_p)



(b) Precision/recall (varying θ_p)



(c) Runtimes (grounding & inference)

Figure 2: Temporal fact extraction (Sec. 7.1)

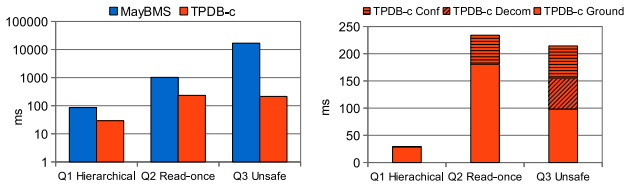


Figure 3: Query answering task (Sec. 7.2)

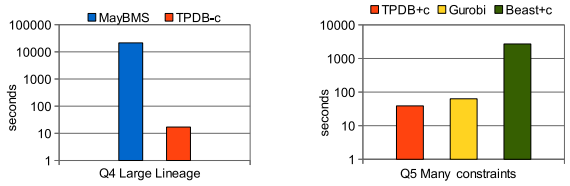
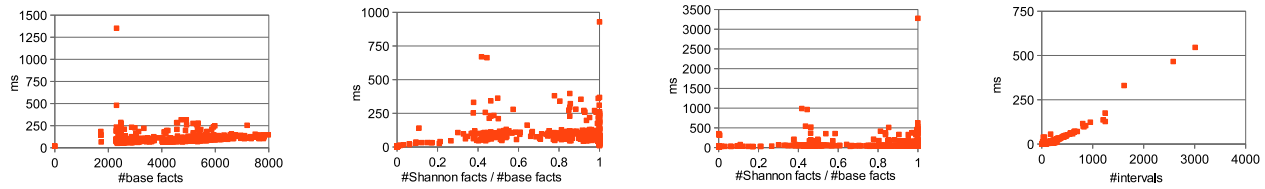


Figure 4: KB building task (Sec. 7.3)



(a) Grounding ($Q3$) (b) Decompositions ($Q3$) (c) Confidences ($Q3$) (d) Deduplication ($Q6$)

Figure 5: Detailed runtime analysis for the different steps of TPDB (Sec. 7.4)

and $\theta_p = 0.7$, we obtain $Precision(f_5) \approx 0.01$ and $Recall(f_5) = 1.0$. To establish precision and recall for sets of facts with different non-temporal arguments, we report the macro-average of the individual facts' values. F_1 measure is computed as the harmonic arithmetic mean of precision and recall.

Results. In Fig. 2(a), we report F_1 as achieved by TPDB with and without constraints (*TPDB-c* and *TPDB+c* respectively) and the ILP competitors *Gurobi* and *Beast*. We only include the best F_1 value since different probability thresholds θ_p produce difference results as shown in Fig. 2(b). The MLN competitors *Alchemy* and *Tuffy* are not included, since even after extensive tuning they either exceeded the available memory, disk space, or ran for several days without terminating. In the supplementary material, we report their performance for a 3% sample of the data set, which yielded reasonable running times.

Discussion. To demonstrate the challenges of the dataset, about 700 of the 1,817 facts violated at least one constraint and about 500 of the 700 facts contradicted more than one constraint (i.e., requiring Shannon expansions). Considering the F_1 metric, TPDB performs best, where the addition of constraints shifts the focus from recall towards precision and sometimes yields major quality gains (see Fig. 2(b)). *Gurobi* and *Beast* perform well, but the single possible world returned by their MAP inference makes the F_1 values less stable (see Fig. 2(a)). With respect to runtimes (Fig. 2(c)), *Beast* and TPDB are competitive (however at a significant gap in F_1 between 65% and 75%), whereas *Gurobi* is slightly slower (and performs worse in F_1 with only 61%). Adding constraints slows TPDB down only moderately. MLNs, however, hardly scale for predicates of arity 4, which blow up the internal data structures of both *Alchemy* and *Tuffy*.

7.2 Query Answering Task

We now focus on answering specific queries over uncertain, temporal data. We employ a large data set to study how scalable TPDB is compared to the established PDB system MayBMS [4]. Since MayBMS has no native support for temporal data, we add two integer attributes to represent the interval begin and end time point, fill them with values, but do not perform any kind of computations on them.

Dataset and Queries. We use YAGO2 [17] as our data set that captures about 100 relations holding $224.4 \cdot 10^6$ facts of which $1.6 \cdot 10^6$ are temporal ones. Because YAGO2 comes with constant confidences per relation, we uniformly sampled confidence values. We focus on three established query classes in PDBs, namely *hierarchical* [33], *read-once* [32], and general *unsafe* [9] queries. We define three query patterns, one for each class, that we instantiate with different constants. Each pattern yields 1,000 distinct queries with varying lineages and numbers of answers. We report the average runtime over the 1,000 queries for each class.

Results. Fig. 3 consists of two plots, the left one depicts the runtimes for MayBMS and TPDB and the right one how the runtime of TPDB is spent on the tasks performed by Algorithm 1 (*TPDB Ground*), Algorithm 3 (*TPDB Decom*), and on confidence computations (*TPDB Conf*).

Discussion. The confidence computations in hierarchical queries ($Q1$) are polynomial, i.e., they are completely captured by Eq. (13). Hence both systems perform very well. Considering read-once lineage ($Q2$), confidence computations are in polynomial time, however they might require a conversion of the lineage formula using the algorithm provided in [32]. This becomes evident by the time spent in con-

fidence computations (*TPDB Conf*). Also, MayBMS slows down slightly. Moving to unsafe queries ($Q3$), which are $\#\mathcal{P}$-hard, our lineage decomposition step (*TPDB Decomp*) becomes the key for achieving the major speed-ups compared to MayBMS. In $Q3$, on average there are about 4,000 base facts in the lineage formula of each answer.

7.3 Knowledge Base Building Task

We now tackle queries with result sizes that occur when we aim to materialize major parts of a temporal KB. We designed two queries for the YAGO2 dataset by focusing on two different challenges, namely producing large lineage formulas and having constraints over many facts.

Large Lineage. Fig. 4 (left part) depicts the runtimes of MayBMS and *TPDB-c* for query $Q4$, where each query answer has a $\#\mathcal{P}$-hard subquery. When grounded, the subquery's lineage alone involves about 150,000 base facts, where *TPDB-c*'s lineage decompositions pay off by a large margin.

Many Constraints. In query $Q5$, we apply 8 constraints to relations with a total size of 10^6 tuples, and run *TPDB+c*, *Gurobi* and *Beast+c*. In Fig. 4 (right part), we report the runtimes for the inference omitting grounding times. TPDB solves the problem in less than 40 seconds while *Gurobi* was not able to find the optimal solution anymore and finishes only after 60 seconds. Last,*Beast* spent about 40 minutes.

7.4 Detailed Runtime Analysis

We conclude our evaluation by exploring how runtimes are spent among the steps of TPDB, i.e., grounding, lineage decompositions, confidence computations, and deduplication.

Grounding. Fig. 5(a) depicts the grounding time (Algorithm 1) of each query in $Q3$. We observe a smooth behavior as the number of base facts grows (except for an outlier).

Decomposition. Fig. 5(b) shows the decomposition time (Algorithm 3) for each of the 1,000 queries in $Q3$. The x-axis shows the fraction of base facts participating in a Shannon expansion. Hence, more difficult lineage formulas appear towards the right-hand side of the x-axis.

Confidence. Using the same x-axis, Fig. 5(c) depicts the runtime for confidence computation (using Eqs. (13) and (14)) for each query of $Q3$. Due to the decomposition, most queries are handled very fast. Still, for more difficult queries the runtime grows as we perform more Shannon expansions.

Deduplication. For Algorithm 2, we created query pattern $Q6$ and instantiated into 1,000 queries, such that varying numbers of time-intervals are involved in the single answer of $Q6$. Fig. 5(d) depicts the runtime of TPDB over the number of time-intervals in the query answers. As pointed out in Sec. 6.2, the runtime actually follows a $n \log n$ shape.

8. CONCLUSIONS

We presented a temporal-probabilistic database model that supports both time and probability as first class citizens. We introduced an expressive class of temporal deduction rules and temporal consistency constraints allowing for a closed and complete representation formalism for temporal uncertain data, and analyzed the properties of our data model from a theoretical perspective. Also, we proposed efficient algorithms, which we evaluated on various information-extraction tasks. As future work, we aim to investigate the support for periodical temporal data and background knowledge (e.g., president elections), and extend our data model to non-independent (i.e., correlated) base facts.

APPENDIX

A. NEGATIONS IN DEDUCTION RULES

We define temporal deduction rules (Definition 1) without requiring temporal variables of negated relational literals to be bound by a positive relational literal. In general, this violates the definition of safe Datalog rules. We justify this by the following reduction. First, we introduce an additional relation $Time(T)$, which contains one base fact for every time point t in the time domain Ω^T. Since Ω^T is finite, *Time* will also be finite. Then, we rewrite Eq. (8) to

$$\begin{aligned} &R_0(\bar{X}, T_{b_0}, T_{e_0}) \leftarrow \\ &\quad \textstyle\bigwedge_i R_i(\bar{X}_i, T_{b_i}, T_{e_i}) \wedge \Phi(\bar{X}) \wedge \\ &\quad \textstyle\bigwedge_j \left(\neg R_j(\bar{X}_j, T_{b_j}, T_{e_j}) \wedge Time(T_{b_j}) \wedge Time(T_{e_j})\right) \end{aligned}$$

The above rule meets all requirements of the definition for safe Datalog rules. Also, the blow-up of the number of grounded rules is polynomial in Ω^T.

B. GROUNDING DEDUCTION RULES

We will reduce our temporal deduction rules (see Definition 1) to regular Datalog with inequalities in the rules' bodies, whose grounding has polynomial data complexity [21]. During grounding, we assume lineage tracing, as formally defined in [13], to take place.

For this, let Ω^T be isomorphic to a finite subset of the natural numbers. For each temporal (and probabilistic) relation $R(\bar{X})_{[T_b,T_e)}$, we introduce an additional deterministic relation $R^d(\bar{X}, T_b, T_e)$. For extensional relations, we copy all base facts, such that $P(R^d(\bar{a}, t_b, t_e)) = 1.0$. We then rewrite a general rule of Eq. (8) as follows. First, we create a deterministic version of the rule:

$$\begin{aligned} &R^d(\bar{X}, T_b, T_e) \leftarrow \\ &\quad \textstyle\bigwedge_i R_i^d(\bar{X}_i, T_{b,i}, T_{e,i}) \wedge \bigwedge_j \neg R_j^d(\bar{X}_j, T_{b,j}, T_{e,j}) \wedge \Phi \end{aligned}$$

Applying this rule yields all facts as deterministic facts, which are not necessarily duplicate-free. To counter this, we emulate Algorithm 2 in Datalog. Therefore, we gather all limits of time intervals by the following two rules over the deterministic relations:

$$\begin{aligned} Limit_R^d(\bar{X}, T_b) &\leftarrow R^d(\bar{X}, T_b, T_e) \\ Limit_R^d(\bar{X}, T_e) &\leftarrow R^d(\bar{X}, T_b, T_e) \end{aligned}$$

Thus, the $Limit_R^d$ relation corresponds to the set in Line 1 of Algorithm 2. To ensure that intervals will be non-overlapping, we create another deterministic relation that is instantiated for each pair of time points (denoted by the variables T_1, T_3), if there is at least one *Limit* fact in between:

$$\begin{aligned} Between_R^d(\bar{X}, T_1, T_3) \leftarrow \; & Limit_R^d(\bar{X}, T_1) \wedge Limit_R^d(\bar{X}, T_2) \wedge \\ & Limit_R^d(\bar{X}, T_3) \wedge T_1 < T_2 \wedge T_2 < T_3 \end{aligned}$$

Now, we deduce all adjacent, non-overlapping time intervals (i.e., shown in Fig. 1's upper part) by the following rule:

$$\begin{aligned} Interval_R^d(\bar{X}, T_b, T_e) \leftarrow \; & Limit_R^d(\bar{X}, T_b) \wedge Limit_R^d(\bar{X}, T_e) \\ & \wedge T_b < T_e \wedge \neg Between_R^d(\bar{X}, T_b, T_e) \end{aligned}$$

These intervals are duplicate free as demanded by Definition 3. At this point, we are able to deduce the time intervals as deterministic facts, however, we are missing rules to induce the correct lineage in order to compute probabilities.

Thus, in the final step, we create another copy of Eq. (8) where $R'(\bar{X}, T_b, T_e)$ is a new relation.

$$R'(\bar{X}, T_b, T_e) \leftarrow \bigwedge_i R_i(\bar{X}_i, T_{b,i}, T_{e,i}) \wedge \bigwedge_j \neg R_j(\bar{X}_j, T_{b,j}, T_{e,j}) \wedge \Phi(\bar{X})$$

Again, its deduced time intervals correspond to the ones on middle of Fig. 1. We add a final deduction rule which combines the deduced facts with their correct time intervals from $Interval^d_R$ and creates the logical disjunction in the lineage as required by Definition 4, thus utilizing the uncertain facts of R':

$$R(\bar{X}, T'_b, T'_e) \leftarrow \begin{pmatrix} R'(\bar{X}, T_b, T_e) \wedge Interval^d(\bar{X}, T'_b, T'_e) \\ \wedge T_b \leq T'_b \wedge T'_e \leq T_e \end{pmatrix}$$

We note that the head's time interval $[T'_b, T'_e)$ originates from $Interval^d(\bar{X}, T'_b, T'_e)$, hence the facts deduced from R' are duplicate-free. Also, all facts in $R'(\bar{X}, T_b, T_e)$, whose time interval contains $[T'_b, T'_e)$, will deduce $R(\bar{X}, T'_b, T'_e)$, so the logical disjunction of Eq. (10) is produced. Furthermore, since $Interval^d$ is deterministic, the confidence of the deduced fact entirely depends on R'.

If we apply the above procedure to all temporal deduction rules, we obtain a polynomial blow-up in the size of the data and the number of rules. Since the data complexity of non-recursive Datalog with inequalities is polynomial, also our model has polynomial data complexity.

C. CLOSURE AND COMPLETENESS

We show that, given any finite instance **D** of temporal and probabilistic relational data, we can represent it in our temporal-probabilistic database model. In general, a relation instance **R** in **D** might not be duplicate-free (Definition 3). We counter this by adding unique fact ids to **R**. Via the reduction of Appendix B, the proof in [33], stating the completeness of (non-temporal) tuple-independent probabilistic databases extended with lineage under all operations of the relational calculus, applies to our data model as well. Hence, we conclude that completeness follows also for our temporal-probabilistic database model.

4. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] J. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11), 1983.

[3] P. Alvaro, W. Marczak, N. Conway, J. M. Hellerstein, D. Maier, and R. C. Sears. Dedalus: Datalog in time and space. Technical Report UCB/EECS-2009-173, University of California, Berkeley, 2009.

[4] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, 2008.

[5] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, 1999.

[6] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with Uncertainty and Lineage. *The VLDB Journal*, 17(2), 2008.

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

[8] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the web. *PVLDB*, 1(1), 2008.

[9] N. Dalvi and D. Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, 2007.

[10] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4), 2007.

[11] A. Dekhtyar, R. Ross, and V. S. Subrahmanian. Probabilistic temporal databases, I: Algebra. *ACM Trans. Database Syst.*, 26, 2001.

[12] A. Dignös, M. H. Böhlen, and J. Gamper. Temporal alignment. In *SIGMOD*, 2012.

[13] M. Dylla, I. Miliaraki, and M. Theobald. Top-k Query Processing in Probabilistic Databases with Non-Materialized Views. In *ICDE*, 2013.

[14] T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle. Querying uncertain spatio-temporal data. In *ICDE*, 2012.

[15] M. Fisher, D. Gabbay, and L. Vila. *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

[16] J. R. Frank, M. Kleiman-Weiner, D. A. Roberts, F. Niu, C. Zhang, C. Re, and I. Soboroff. Building an entity-centric stream filtering test collection for TREC 2012. In *TREC*, 2012.

[17] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194(0), 2013.

[18] C. S. Jensen. *Temporal Database Management*. PhD thesis, Aalborg University, 2000.

[19] A. Jha and D. Suciu. Knowledge compilation meets database theory: compiling queries to decision diagrams. In *ICDT*, 2011.

[20] H. Ji and R. Grishman. Knowledge base population: successful approaches and challenges. HLT, 2011.

[21] P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint query languages. In *PODS*, 1990.

[22] C. Koch and D. Olteanu. Conditioning probabilistic databases. *PVLDB*, 1(1), 2008.

[23] X. Ling and D. S. Weld. Temporal information extraction. In *AAAI*, 2010.

[24] L. Liu and M. T. Zsu. *Encyclopedia of Database Systems*. Springer, 1st edition, 2009.

[25] N. Nakashole, G. Weikum, and F. M. Suchanek. PATTY: A taxonomy of relational patterns with semantic types. In *EMNLP*, 2012.

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

[27] D. Olteanu and J. Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *SUM*, 2008.

[28] J. Pustejovsky, J. M. Castaño, R. Ingria, R. Sauri, R. J. Gaizauskas, A. Setzer, G. Katz, and D. R. Radev. TimeML: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering*, 2003.

[29] M. Richardson and P. Domingos. Markov Logic Networks. *Mach. Learn.*, 62(1-2), 2006.

[30] S. Riedel. Improving the accuracy and efficiency of MAP inference for Markov Logic. In *UAI*, 2008.

[31] A. D. Sarma, M. Theobald, and J. Widom. LIVE: a lineage-supported versioned DBMS. In *SSDBM*, 2010.

[32] P. Sen, A. Deshpande, and L. Getoor. Read-once functions and query evaluation in probabilistic databases. *PVLDB*, 3(1), 2010.

[33] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool Publishers, 2011.

[34] P. P. Talukdar, D. Wijaya, and T. Mitchell. Coupled temporal scoping of relational facts. In *WSDM*, 2012.

[35] M. Verhagen, R. Saurí, T. Caselli, and J. Pustejovsky. SemEval-2010 Task 13: TempEval-2. In *SemEval*, 2010.

[36] L. Vila. A survey on temporal reasoning in artificial intelligence. *AI Commun.*, 7(1), 1994.

[37] Y. Wang, M. Dylla, M. Spaniol, and G. Weikum. Coupling label propagation and constraints for temporal fact extraction. In *ACL*, 2012.

[38] Y. Wang, M. Yahya, and M. Theobald. Time-aware Reasoning in Uncertain Knowledge Bases. In *MUD*, 2010.