Representing and Querying Validity Time in RDF and OWL: A Logic-Based Approach

Boris Motik

Oxford University Computing Laboratory, Oxford, UK

Abstract. RDF(S) and OWL 2 currently support only static ontologies. In practice, however, the truth of statements often changes with time, and Semantic Web applications often need to represent such changes and reason about them. In this paper we present a logic-based approach for representing validity time in RDF and OWL. Unlike the existing proposals, our approach is applicable to entailment relations that are not deterministic, such as the Direct Semantics or the RDF-Based Semantics of OWL 2. We also extend SPARQL to temporal RDF graphs and present a query evaluation algorithm. Finally, we present an optimization of our algorithm that is applicable to entailment relations characterized by a set of deterministic rules, such RDF(S) and OWL 2 RL/RDF entailment.

1 Introduction

RDF(S) and OWL 2 currently support only *static* ontologies. In practice, however, the truth of statements often changes with time, and Semantic Web applications often need to represent such changes and reason about them. We discuss these issues on an example derived from the author's collaboration with ExperienceOn (abbreviated EO)—an IT start-up company from Barcelona, Spain.

EO aims to improve search in the tourism domain by providing an advanced system that can answer complex queries such as "trips to the second week of Oktoberfest." Users will input their questions in natural language, and NLP technology will translate such questions into one or more queries over a knowledge base containing information about flights, lodging, events, geography, and so on. EO's system must be able to represent statements that are not universally true, but are associated with *validity times*. For example, "Oktoberfest is being held in Munich" is true only while the festival is being held; similarly, statements describing airline flight schedules are valid only in certain time intervals. Validity time must be tightly integrated with reasoning; for example, from the knowledge about Oktoberfest and German geography, EO's system should conclude that "Oktoberfest is being held in Bavaria" is true for the duration of the festival. Validity time should also be integrated with a query language, allowing one to retrieve "flights from London to Munich during Oktoberfest." Validity time thus affects virtually all aspects of knowledge representation and reasoning in scenarios such as EO's. Some applications also need to represent *transaction times*, which specify when facts were added to the database. In this paper we focus on validity time since it is more relevant to knowledge modeling.

P.F. Patel-Schneider et al. (Eds.): ISWC 2010, Part I, LNCS 6496, pp. 550–565, 2010.

Validity time has been extensively studied in databases and artificial intelligence [4,19]. Neither RDF nor OWL, however, supports validity time, and SPARQL does not provide temporal query primitives. These deficiencies have been recognized by the community, and several proposals have emerged. A comprehensive framework for representing validity time in RDF was presented in [7], and it encompasses notions of temporal graphs and entailment, a characterization of temporal entailment via closures [6], an encoding of temporal graphs into regular RDF graphs, and a sketch of a temporal query language. This approach was extended in [9] with more general temporal constraints. In [15], the authors extended the approach from [7] with unknown time points, defined a temporal query language based on graph matching, and presented a way for indexing temporal graphs. A general framework for annotating RDF data was presented in [17]; the very general notion of annotations can be used to represent validity time. A temporal extension of SPARQL was presented in [18]. Approaches to extending *description logics* (DLs) [3]—the family of formalisms underpinning OWL 2 DL—with temporal features were surveyed in [2]. A temporal extension of OWL based on concrete domains was presented in [10].

None of these proposals is applicable to all variants of RDF and OWL. For example, the notion of closures from [7] relies on the fact that the inference rules of RDF(S) are deterministic—an assumption that does not hold in expressive languages such as OWL. In this paper we present a novel approach for representing validity time that is applicable to all Semantic Web languages, including RDF(S) and all profiles of OWL 2. In particular, in Section 3 we develop a first-order interpretation of temporal graphs, which we use to define temporal graph entailment. Our approach coincides with the one from [7] on RDF(S), but it is applicable to all languages of the RDF and OWL family.

In Section 4, we argue that a temporal query language defined in the obvious way would allow for queries that have very large and often even infinite answers. We present a query language whose queries always have finite answers, we integrate our query primitives into the formalization of SPARQL from [14], and we present a general query evaluation algorithm. In Section 5 we optimize our general evaluation algorithm for the case of deterministic inference rules.

We implemented our approach in EO's system. Given the nature of EO's business, we cannot make the system publicly available; however, EO is successfully using our approach to answer temporal queries, which we take as indication that our approach is suitable for practice.

The proofs of all technical results can be found in the extended version of this paper available from the author's online publication list.

2 Preliminaries

We assume the reader to be familiar with the syntax and semantics of OWL 2 DL [12]; for simplicity, we write OWL 2 DL axioms using the description logic syntax [3]. We use the standard definitions of constants, variables, terms, predicates, atoms, multi-sorted first-order logic, and skolemization [5]. For α an

OWL 2 DL axiom or an ontology, let $\theta(\alpha)$ be the translation of α into a first-order formula. We assume that the equality predicate $\approx$ is treated in $\theta(\alpha)$ as a standard first-order predicate explicitly axiomatized as a congruence; this does not affect the consequences of $\theta(\alpha)$ [5]. Moreover, we assume that θ maps the blank nodes (also called anonymous individuals) in α into free first-order variables, so the semantics of α is $\exists y_1, \ldots, y_n : \theta(\alpha)$ where $y_1, \ldots y_n$ are the blank nodes of α.

Let $\mathcal{U}$, $\mathcal{B}$, and $\mathcal{L}$ be infinite sets of *URI references*, *blank nodes*, and *literals*, respectively, and let $\mathcal{UBL} = \mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$. A *triple* is an assertion of the form $\langle s, p, o \rangle$ with $s, p, o \in \mathcal{UBL}$.[1] An *RDF graph* (or just *graph*) G is a finite set of triples. The semantics of RDF is determined by *entailment relations*.

Simple entailment, *RDF entailment*, *RDFS entailment*, and *D-entailment* are defined in [8], and *OWL 2 RL/RDF entailment* and *OWL 2 RDF-Based entailment* are defined in [16]. The logical consequences of each entailment relation X from this list can be characterized by a (possibly infinite) set of first-order implications Γ_X. For example, for RDF entailment, Γ_{RDF} contains the rules in [8, Section 7], and for OWL 2 RL/RDF entailment, Γ_{RL} contains the rules in [11, Section 4.3]. The semantics of a graph G w.r.t. X can be defined by transforming G into a first-order theory as follows. We assume that each blank node corresponds to a first-order variable (i.e., for simplicity, we do not distinguish blank nodes from variables). Let $\mathsf{b}_X(G)$ be the set of all blank nodes in G. For a triple $A = \langle s, p, o \rangle$, let $\pi_X(A) = T(s, p, o)$, where T is a ternary first-order predicate. For a graph G, let $\pi_X(G) = \bigwedge_{A \in G} \pi_X(A)$. The first-order theory corresponding to G is then $\nu_X(G) = \{\exists \mathsf{b}_X(G) : \pi_X(G)\} \cup \Gamma_X$. Let $\xi_X(G)$ be obtained from $\nu_X(G)$ by skolemizing the existential quantifiers $\exists \mathsf{b}_X(G)$—that is, by removing $\exists \mathsf{b}_X(G)$ and replacing each blank node in $\pi_X(G)$ with a fresh URI reference. Theory $\nu_X(G)$ is equisatisfiable with $\xi_X(G)$. A graph G_1 *X-entails* a graph G_2, written $G_1 \models_X G_2$, if and only if $\nu_X(G_1) \models \exists \mathsf{b}_X(G_2) : \pi_X(G_2)$; the latter is the case if and only if $\xi_X(G_1) \models \exists \mathsf{b}_X(G_2) : \pi_X(G_2)$.

We next define *OWL 2 Direct entailment* (written *DL* due to its relationship with description logic). A graph G *encodes an OWL 2 DL ontology* if G can be transformed into an OWL 2 DL ontology $\mathcal{O}(G)$ as specified in [13]. For such G, let $\mathsf{b}_{DL}(G)$ be the set of blank nodes occurring in $\mathcal{O}(G)$; let $\pi_{DL}(G) = \theta(\mathcal{O}(G))$; let $\nu_{DL}(G) = \exists \mathsf{b}_{DL}(G) : \theta(\mathcal{O}(G))$; and let $\xi_{DL}(G)$ be obtained from $\nu_{DL}(G)$ by skolemizing the existential quantifiers $\exists \mathsf{b}_{DL}(G)$. Formula $\nu_{DL}(G)$ is equisatisfiable with $\xi_{DL}(G)$. For G_1 and G_2 graphs that encode OWL 2 DL ontologies, G_1 *DL-entails* G_2, written $G_1 \models_{DL} G_2$, iff $\nu_{DL}(G_1) \models \exists \mathsf{b}_{DL}(G_2) : \pi_{DL}(G_2)$; the latter is the case if and only if $\xi_{DL}(G_1) \models \exists \mathsf{b}_{DL}(G_2) : \pi_{DL}(G_2)$.

SPARQL is the standard W3C language for querying RDF graphs, and the 1.1 version (currently under development) will support different entailment relations. In this paper we focus on *group patterns*—the core of SPARQL that deals with pattern matching and is largely independent from constructs such as aggregates and sorting. We formalize group patterns as in [14], and we treat answers as sets rather than multisets as this simplifies the presentation without changing

[1] RDF actually requires $s \in \mathcal{U} \cup \mathcal{B}$, $p \in \mathcal{U}$, and $o \in \mathcal{UBL}$, but this is not important in our framework so we assume $s, p, o \in \mathcal{UBL}$ for the sake of simplicity.

the nature of our results. Let $\mathcal{V}$ be an infinite set of *variables* disjoint from $\mathcal{UBL}$. A *mapping* is a partial function $\mu : \mathcal{V} \to \mathcal{UBL}$. The domain (resp. range) of μ is written $\mathsf{dm}(\mu)$ (resp. $\mathsf{rg}(\mu)$). We define $\mu(t) = t$ for $t \in \mathcal{UBL} \cup \mathcal{V} \setminus \mathsf{dm}(\mu)$. Mappings μ_1 and μ_2 are *compatible* if $\mu_1(x) = \mu_2(x)$ for each $x \in \mathsf{dm}(\mu_1) \cap \mathsf{dm}(\mu_2)$; in such a case, $\mu_1 \cup \mu_2$ is also a mapping. The following algebraic operations on sets of mappings Ω_1 and Ω_2 are used to define the semantics of group patterns.

$$\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \text{ and } \mu_1 \text{ and } \mu_2 \text{ are compatible}\}$$
$$\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \text{each } \mu_2 \in \Omega_2 \text{ is not compatible with } \mu_1\}$$

A *built-in expression* is constructed using the elements of $\mathcal{V} \cup \mathcal{U} \cup \mathcal{L}$ as specified in [14]; furthermore, for each built-in expression R and each mapping μ, we can determine whether R evaluates to true under μ, written $\mu \models R$, as specified in [14]. A *basic graph pattern* (BGP) is a set of triples of the form $\langle s, p, o\rangle$ where $s, p, o \in \mathcal{UBL} \cup \mathcal{V}$. A *group pattern* (GP) is an expression of the form B, P_1 **and** P_2, P_1 **union** P_2, P_1 **opt** P_2, or P_1 **filter** R, where B is a BGP, P_1 and P_2 are group patterns, and R is a built-in expression. For A a built-in expression or a group pattern and μ a mapping, $\mathsf{var}(A)$ is the set of variables occurring in A, and $\mu(A)$ is the result of replacing each variable x in A with $\mu(x)$.

The *answer* to a group pattern P on a graph G depends on an entailment relation X. For each X, we assume that a function exists that maps each graph G to the set $\mathsf{ad}_X(G) \subseteq \mathcal{UBL}$ called the *answer domain* of G; this set determines the elements of $\mathcal{UBL}$ that can occur in answers to group patterns on G under X-entailment. To see why this is needed, let $B = \{\langle x, \mathit{rdf{:}type}, \mathit{rdf{:}Property}\rangle\}$; due to the axiomatic triples [8], $\emptyset \models_{RDF} \mu(B)$ whenever $\mu(x) \in \{\mathit{rdf{:}\_1}, \mathit{rdf{:}\_2}, \ldots\}$. Without any restrictions, the answer to B under RDF entailment would thus be infinite even in the empty graph. To prevent this, $\mathsf{ad}_{RDF}(G)$ excludes $\mathit{rdf{:}\_1}$, $\mathit{rdf{:}\_2}$, ... that do not occur in G, which makes $\mathsf{ad}_{RDF}(G)$ finite and thus ensures finiteness of answers. Similar definitions are used for X other than RDF.

SPARQL treats blank nodes as objects with distinct identity. To understand this, let $G = \{\langle a, b, c\rangle, \langle d, e, \_{:}1\rangle\}$ where $\_{:}1$ is a blank node, let $P = \langle a, b, x\rangle$, and let $\mu = \{x \mapsto \_{:}1\}$. Even though $G \models_{RDF} \mu(P)$, the answer to P on G under RDF entailment does not contain μ. Roughly speaking, $\_{:}1$ is distinct from c even though $\_{:}1$ is semantically a "placeholder" for an arbitrary URI reference. We capture this idea using skolemization: we replace the blank nodes in G with fresh URI references, thus giving each blank node a unique identity. Our answers are isomorphic to the answers of the official SPARQL specification, so skolemization allows us to simplify the technical presentation without losing generality. We formalize this idea by evaluating group patterns in $\xi_X(G)$ instead of $\nu_X(G)$. Table 1 defines the answer $[\![P]\!]^X_G$ to a group pattern P in a graph G w.r.t. X.

3 Representing Validity Time in RDF and OWL

To incorporate validity time into RDF, one could simply equip each triple with a validity time instant; however, it would be impractical or even impossible to explicitly list all such time instants. To this end, Chomicki distinguishes an

Table 1. Semantics of Group Patterns

$$
\begin{aligned}
[\![B]\!]^X_G &= \{\mu \mid \mathsf{dm}(\mu) = \mathsf{var}(B), \mathsf{rg}(\mu) \subseteq \mathsf{ad}_X(G), \xi_X(G) \models \exists \mathsf{b}_X(\mu(B)) : \pi_X(\mu(B))\} \\
[\![P_1 \text{ and } P_2]\!]^X_G &= [\![P_1]\!]^X_G \bowtie [\![P_2]\!]^X_G \\
[\![P_1 \text{ union } P_2]\!]^X_G &= [\![P_1]\!]^X_G \cup [\![P_2]\!]^X_G \\
[\![P_1 \text{ opt } P_2]\!]^X_G &= [\![P_1]\!]^X_G \bowtie [\![P_2]\!]^X_G \cup [\![P_1]\!]^X_G \setminus [\![P_2]\!]^X_G \\
[\![P_1 \text{ filter } R]\!]^X_G &= \{\mu \in [\![P_1]\!]^X_G \mid \mu \models R\}
\end{aligned}
$$

abstract from a *concrete* temporal database [4]. The former is a sequence of "static" databases each of which contains the facts true at some time instant. Since the time line is unbounded, an abstract temporal database is infinite, so a concrete temporal database is used as a finite specification of one or more abstract temporal databases. We next apply thus approach to RDF and OWL.

We use a discrete notion of time, since the ability to talk about predecessors/successors of time instants is needed in Section 4. Thus, the set $\mathcal{TI}$ of *time instants* is the set of all integers, $\leq$ is the usual total order on $\mathcal{TI}$, and $+1$ and -1 are the usual successor and predecessor functions on $\mathcal{TI}$. The set of *time constants* is $\mathcal{TC} = \mathcal{TI} \cup \{-\infty, +\infty\}$; we assume that $\mathcal{UBL} \cap \mathcal{TC} = \emptyset$. Time constants $-\infty$ and $+\infty$ are special in that they can occur in first-order formulae only in atoms of the form $-\infty \leq t$, $-\infty \leq +\infty$, and $t \leq +\infty$ for t a time instant or a variable; all such atoms are syntactic shortcuts for true. This allows us to simplify the notation for bounded and unbounded time intervals; for example, to say that the interval described by formula $t_1 \leq x^{\mathsf{t}} \leq t_2$ has no lower bound, we write $t_1 = -\infty$, which makes the formula equivalent to $x^{\mathsf{t}} \leq t_2$.

Definition 1. *A* temporal triple *has the form* $\langle s, p, o\rangle[t]$ *or* $\langle s, p, o\rangle[t_1, t_2]$*, such that* $s, p, o \in \mathcal{UBL}$*,* $t \in \mathcal{TI}$*,* $t_1 \in \mathcal{TI} \cup \{-\infty\}$*, and* $t_2 \in \mathcal{TI} \cup \{+\infty\}$*. A* temporal graph G *is a finite set of temporal triples.*

In this work, we focus mainly on the conceptual aspects of temporal graphs and we do not discuss practical issues such as serialization syntax. We interpret temporal graphs in multi-sorted first-order logic. Let t be a distinct *temporal sort* interpreted over $\mathcal{TI}$; we write x^{t} to stress that a variable x ranges over $\mathcal{TI}$. For each n-ary predicate P, let $\hat{P}$ be the $n+1$-ary predicate where positions 1–n have the same sort as in P, and position $n+1$ is of sort t. For t a term of sort t and $P(u_1, \ldots, u_n)$ an atom, let $P(u_1, \ldots, u_n)\langle t\rangle = \hat{P}(u_1, \ldots, u_n, t)$, and let $\varphi\langle t\rangle$ be obtained by replacing each atom A with $A\langle t\rangle$ in a first-order formula φ.

Intuitively, atom $\hat{P}(u_1, \ldots, u_n, t)$ encodes the truth of atom $P(u_1, \ldots, u_n)$ at time instant t: the former is true iff the latter is true at time t, so our approach is similar to the temporal arguments approach [19]. Similarly, $\varphi\langle t\rangle$ determines the truth of φ at time instant t. As explained in Section 2, $\approx$ is an ordinary predicate with an explicit axiomatization, so $\hat{\approx}$ is well defined and it gives us a notion of equality that changes with time. Finally, to understand why a multi-sorted interpretation is needed, consider a graph G that encodes the OWL 2 DL axiom $\top \sqsubseteq \{c\}$. Such G is satisfiable only in first-order interpretations consisting of a

single object, which contradicts the requirement that a domain should contain $\mathcal{TI}$. Multi-sorted logic cleanly separates temporal instants from other objects in the domain, so axioms such as $\top \sqsubseteq \{c\}$ do not quantify over time instants, which solves the problem. We next define the semantics of temporal graphs.

Definition 2. *Let X be an entailment relation from Section 2 other than DL, and let Γ_X be the first-order theory that characterizes X. For G a temporal graph, $\mathsf{u}_X(G)$, $\mathsf{b}_X(G)$, and $\mathsf{tc}_X(G)$ are the subsets of $\mathcal{U} \cup \mathcal{L}$, $\mathcal{B}$, and $\mathcal{TC}$, respectively, that occur in G. Mappings π_X and ν_X are extended to temporal graphs as shown below, where O is a fresh unary predicate. Furthermore, $\xi_X(G)$ is obtained from $\nu_X(G)$ by skolemizing the existential quantifiers in $\exists \mathsf{b}_X(G)$, and $\mathsf{ub}_X(G)$ is $\mathsf{u}_X(G)$ extended with the URI references introduced via skolemization.*

$$\pi_X(\langle s, p, o\rangle[t]) = \hat{T}(s, p, o, t)$$
$$\pi_X(\langle s, p, o\rangle[t_1, t_2]) = \forall x^{\mathsf{t}} : (t_1 \leq x^{\mathsf{t}} \leq t_2) \rightarrow \hat{T}(s, p, o, x^{\mathsf{t}})$$
$$\pi_X(G) = \bigwedge_{u \in \mathsf{b}_X(G)} O(u) \wedge \bigwedge_{A \in G} \pi_X(A)$$
$$\nu_X(G) = \{\exists \mathsf{b}_X(G) : \bigwedge_{u \in \mathsf{u}_X(G)} O(u) \wedge \pi_X(G)\} \cup \{\forall x^{\mathsf{t}} : \varphi\langle x^{\mathsf{t}}\rangle \mid \varphi \in \Gamma_X\}$$

A temporal graph G_1 entails *a temporal graph G_2 under entailment relation X, written $G_1 \models_X G_2$, if and only if $\nu_X(G_1) \models \exists \mathsf{b}_X(G_2) : \pi_X(G_2)$.*

Intuitively, predicate O in $\nu_X(G)$ "contains" all elements of $\mathsf{u}_X(G) \cup \mathsf{b}_X(G)$ that occur in G, which ensures that, whenever $G_1 \models_X G_2$, all blank nodes in G_2 can be mapped to $\mathsf{u}_X(G_1) \cup \mathsf{b}_X(G_1)$. We discuss the rationale behind such a definition at end of this section; for the moment, we just note that, when applied to RDF(S), our definition of entailment coincides with the one from [7].

We next present a small example. Let G_1 be the temporal graph containing temporal triples (1)–(3). Triples in (1) state that there is a flight from LHR to MUC; this information may have been gathered from two distinct sources, so validity times of the two triples overlap. Triple (2) states that Munich hosts Oktoberfest. Finally, triple (3) states that, if x hosts y, then x has y as an attraction; that this statement is not universally true might be due to the fact that attractions are relevant only during holiday seasons. One can easily verity that $G_1 \models_{RDFS} \langle{:}Munich, {:}hasAttraction, {:}Oktoberfest\rangle[130, 180]$.

$$\langle{:}LHR, {:}flightTo, {:}MUC\rangle[50, 120] \qquad \langle{:}LHR, {:}flightTo, {:}MUC\rangle[100, 150] \quad (1)$$
$$\langle{:}Munich, {:}hosts, {:}Oktoberfest\rangle[80, 180] \quad (2)$$
$$\langle{:}hosts, rdfs{:}subPropertyOf, {:}hasAttraction\rangle[130, 300] \quad (3)$$

OWL 2 Direct entailment is not characterized by a fixed set of first-order implications, so we define temporal OWL 2 Direct entailment separately.

Definition 3. *A* temporal OWL 2 DL axiom *has the form $\alpha[t]$ or $\alpha[t_1, t_2]$ for α an OWL 2 DL axiom, $t \in \mathcal{TI}$, $t_1 \in \mathcal{TI} \cup \{-\infty\}$, and $t_2 \in \mathcal{TI} \cup \{+\infty\}$. A*

temporal OWL 2 DL ontology $\mathcal{O}$ *is a finite set of temporal OWL 2 DL axioms. Temporal axioms and ontologies are mapped into formulae as* $\theta(\alpha[t]) = \theta(\alpha)\langle t\rangle$, $\theta(\alpha[t_1, t_2]) = \forall x^{\mathsf{t}} : (t_1 \leq x^{\mathsf{t}} \leq t_2) \rightarrow \theta(\alpha)\langle x^{\mathsf{t}}\rangle$, *and* $\theta(\mathcal{O}) = \bigwedge_{A \in \mathcal{O}} \theta(A)$.

A temporal graph G encodes a temporal OWL 2 DL ontology $\mathcal{O}(G)$ *if* $\mathcal{O}(G)$ *can be extracted from* G *using the mapping from [13] modified as follows:*

- *Each* $\langle s, p, o\rangle$ *in Tables 3–8 and 10–15 is replaced with* $\langle s, p, o\rangle[-\infty, +\infty]$.
- *Each triple pattern from Tables 16 and 17 without a main triple*[2] *producing an axiom* α *is changed as follows: each* $\langle s, p, o\rangle$ *in the pattern is replaced with* $\langle s, p, o\rangle[-\infty, +\infty]$, *and the triple pattern produces* $\alpha[-\infty, +\infty]$.
- *Each triple pattern from Tables 16 and 17 with a main triple* $\langle s_m, p_m, o_m\rangle$ *producing an axiom* α *is replaced with the following two triple patterns.*
  - *The first one is obtained by replacing each triple* $\langle s, p, o\rangle$ *in the pattern other than the main one with* $\langle s, p, o\rangle[-\infty, +\infty]$, *replacing the main triple with* $\langle s_m, p_m, o_m\rangle[t]$, *and making the triple pattern produce* $\alpha[t]$.
  - *The second one is obtained by replacing each triple* $\langle s, p, o\rangle$ *in the pattern other than the main one with* $\langle s, p, o\rangle[-\infty, +\infty]$, *replacing the main triple with* $\langle s_m, p_m, o_m\rangle[t_1, t_2]$, *and making the triple pattern produce* $\alpha[t_1, t_2]$.

For G *encoding a temporal OWL 2 DL ontology* $\mathcal{O}(G)$, $\mathsf{u}_{DL}(G)$, $\mathsf{b}_{DL}(G)$, *and* $\mathsf{tc}_{DL}(G)$ *are the sets of named individuals, blank nodes, and temporal constants, respectively, in* $\mathcal{O}(G)$. *Mappings,* π_{DL} *and* ν_{DL} *are extended to* G *as shown below, where* O *is a fresh unary predicate. Furthermore,* $\xi_{DL}(G)$ *is obtained from* $\nu_{DL}(G)$ *by skolemizing the existential quantifiers in* $\exists \mathsf{b}_{DL}(G)$, *and* $\mathsf{ub}_{DL}(G)$ *is* $\mathsf{u}_{DL}(G)$ *extended with the named individuals introduced via skolemization.*

$$\pi_{DL}(G) = \bigwedge_{u \in \mathsf{b}_{DL}(G)} O(u) \wedge \theta(\mathcal{O}(G))$$

$$\nu_{DL}(G) = \exists \mathsf{b}_{DL}(G) : \bigwedge_{u \in \mathsf{u}_{DL}(G)} O(u) \wedge \pi_{DL}(G)$$

For G_1 *and* G_2 *temporal graphs that encode temporal OWL 2 DL ontologies, we have* $G_1 \models_{DL} G_2$ *if and only if* $\nu_{DL}(G_1) \models_{DL} \exists \mathsf{b}_{DL}(G_2) : \pi_{DL}(G_2)$.

Definition 3 allows us to attach validity time to axioms (but not to parts of axioms such as class expressions), which provides us with a flexible language that can represent, for example, class hierarchies that change over time.

We next explain the intuition behind the predicate O in Definitions 2 and 3. Note that $\exists \mathsf{b}_X(G)$ occurs in ν_X before the universal quantifiers over $\mathcal{TI}$, so blank nodes in G are interpreted *rigidly*—that is, they represent the same objects throughout all time. For example, let $G_2 = \{\langle s, p, \_{:}1\rangle[-\infty, +\infty]\}$, so $\pi_{DL}(G_2) = \exists \_{:}1 : O(\_{:}1) \wedge \forall x^{\mathsf{t}} : \hat{p}(s, \_{:}1, x^{\mathsf{t}})$; since $\exists \_{:}1$ comes before $\forall x^{\mathsf{t}}$, blank node $\_{:}1$ refers to the same object at all time instants. In contrast, the existential quantifiers in $\varphi\langle x^{\mathsf{t}}\rangle$ and $\theta(\mathcal{O}(G))$ are *not rigid*—that is, they can be satisfied by different objects at different time instants. For example, let G_3 be such that $\mathcal{O}(G_3) = \{\exists p.\top(s)[-\infty, +\infty]\}$, so $\pi_{DL}(G_3) = \forall x^{\mathsf{t}} : \exists y : \hat{p}(s, y, x^{\mathsf{t}})$; since $\exists y$

[2] Please refer to [13] for the definition of a main triple.

comes after $\forall x^{\mathsf{t}}$, the value for y can be different at different time instants. Consequently, G_2 is *not* *DL*-equivalent to G_3; in fact, G_2 *DL*-entails G_3, but not vice versa. Blank nodes can thus be understood as unnamed constants, which we believe to be in the spirit of RDF and OWL. In line with this intuition, conjuncts $\bigwedge_{u \in \mathsf{b}_X(G)} O(u)$ and $\bigwedge_{u \in \mathsf{u}_X(G)} O(u)$ in Definitions 2 and 3 ensure that, if $G_2 \models_X G_3$, then the blank nodes in $\mathsf{b}_X(G_3)$ can be mapped to the rigid objects in $\mathsf{u}_X(G_2)$, but not to the nonrigid objects whose existence is implied by existential quantifiers. Without this restriction, G_3 would *DL*-entail $G_4 = \{\langle s, p, \_{:}1\rangle[1,1]\}$ (since the triple in G_4 refers only to a single time instant, the nonrigidity of $\exists p.\top$ is irrelevant), which seems at odds with the fact that G_3 does not *DL*-entail G_2. Under our semantics, G_3 does not *DL*-entail G_4 due to the O predicate, which seems more intuitive and it is also easier to implement.

4 Querying Temporal Graphs

The first step in designing a query language is to identify the types of questions that the language should support. The language of first-order logic readily reveals the following natural types of questions:

Q1. Is BGP B true in G at some time instant t?
Q2. Is BGP B true in G at all time instants between t_1 and t_2?
Q3. Is BGP B true in G at some time instant between t_1 and t_2?

Such questions can be easily encoded in first-order formulae, and an answer to a formula Q over a graph G under entailment relation X can be defined as the set of mappings μ of the free variables of Q such that $G \models_X \mu(Q)$. Such an approach, however, has an important drawback. Let $G_5 = \{\langle a, b, c\rangle[5, 12],\ \langle a, b, c\rangle[9, +\infty]\}$ and let $Q(x_1, x_2) = \forall x : x_1 \leq x \leq x_2 \rightarrow \langle a, b, c\rangle[x]$ be a question of type Q2. Evaluating $Q(x_1, x_2)$ on G_5 is not a problem if x_1 and x_2 are concrete time instants. Note, however, that $Q(x_1, x_2)$ does not ask for *maximal* x_1 and x_2 for which the formula holds. Thus, the answer to $Q(x_1, x_2)$ on G_5 is infinite since it contains each mapping μ such that $5 \leq \mu(x_1) \leq \mu(x_2) \leq +\infty$.

One can restrict answers to mappings that refer to time instants explicitly occurring in G, but this is also problematic. First, answers can contain redundant mappings. For example, $\mu_1 = \{x_1 \mapsto 5, x_2 \mapsto +\infty\}$ is the "most general mapping" in the answer to $Q(x_1, x_2)$ on G_5, but the answer also contains a "less general" mapping $\mu_2 = \{x_1 \mapsto 9, x_2 \mapsto 12\}$. Second, answers can differ on syntactically different but semantically equivalent temporal graphs. For example, $G_6 = \{\langle a, b, c\rangle[5, 10],\ \langle a, b, c\rangle[7, +\infty]\}$ is equivalent to G_5 under simple entailment; however, μ_2 is not contained in the answer to $Q(x_1, x_2)$ on G_6, and $\mu_3 = \{x_1 \mapsto 7, x_2 \mapsto 10\}$ is not contained in the answer to $Q(x_1, x_2)$ on G_5. Third, computing redundant answers can be costly: an answer to a formula such as $Q(x_1, x_2)$ in a graph with n overlapping intervals consists of mappings that refer to any two pairs of interval endpoints, so the number of mappings in the answer is exponential in n. One might try to identify the "most general" mappings, but this would be an ad hoc solution without a clear semantic justification.

We deal with these problems in two stages. First, we introduce primitives that support questions of types Q1–Q3, as well as of types Q4–Q5, thus explicitly introducing a notion of maximality into the language.

Q4. Is $[t_1, t_2]$ the maximal interval such that BGP B holds in G for each time instant in the interval?
Q5. Is t the smallest/largest instant at which BGP B holds in G?

We define our notion of answers w.r.t. $\mathcal{TC}$, which makes the answers independent from the syntactic form of temporal graphs. To ensure finiteness, we then define a syntactic notion of *safety*, which guarantees that only questions of type Q4 and Q5 can "produce" a value.

Practical applications will often need to express constraints on time points and intervals retrieved via Q1–Q5. For example, to retrieve "hotels with vacancy during Oktoberfest," we must require the duration of Oktoberfest to be contained in the hotels' vacancy period. Such conditions can be expressed, for example, using Allen's interval algebra [1], and they can be integrated into our query language via built-in expressions; for example, we can provide a built-in expression that takes two pairs of interval end-points and that is true iff the first interval is contained in the second. Such extensions of our query language are straightforward, so we do not discuss them further in the rest of this paper.

Definition 4. *A* temporal group pattern *(TGP) is an expression defined inductively as shown below, where B is a BGP, P_1 and P_2 are TGPs, R is a built-in expression, $t_1 \in \mathcal{TI} \cup \{-\infty\} \cup \mathcal{V}$, $t_2 \in \mathcal{TI} \cup \{+\infty\} \cup \mathcal{V}$, and $t_3 \in \mathcal{TI} \cup \mathcal{V}$. TGPs from the first two lines are called* basic*.*

B at t_3	B during $[t_1, t_2]$	B occurs $[t_1, t_2]$	
B maxint $[t_1, t_2]$	B mintime t_3	B maxtime t_3	
P_1 and P_2	P_1 union P_2	P_1 opt P_2	P_1 filter R

We redefine a mapping *as a partial function $\mu : \mathcal{V} \to \mathcal{UBL} \cup \mathcal{TC}$. Let X be an entailment relation and G a temporal graph. Let $\mathsf{ad}_X(G) = \mathsf{ad}_X(G')$, where G' is the nontemporal graph obtained by replacing all triples in G of the form $\langle s, p, o\rangle[u]$ and $\langle s, p, o\rangle[u_1, u_2]$ with $\langle s, p, o\rangle$. The answer to a basic TGP P in G under X is the set of mappings defined as specified below, where $\delta_X(\mu(P))$ is a condition from Table 2. Answers to all other TGP types are defined in Table 1.*

$$[\![P]\!]^X_G = \{\mu \mid \mathsf{dm}(\mu) = \mathsf{var}(P), \mathsf{rg}(\mu) \subseteq \mathsf{ad}_X(G) \cup \mathcal{TC}, \textit{and } \delta_X(\mu(P)) \textit{ holds}\}$$

We next present several TGPs that could be used in our running example. TGP (4) returns the maximal intervals $[y, z]$ during Oktoberfest in which a flight from airport x to the Munich airport exists; the answer to (4) on G_1 is $\{\{x \mapsto LHR, y \mapsto 80, y \mapsto 150\}\}$. TGP (5) retrieves all events z in London that have at least one time instant in common with Oktoberfest; if occurs were changed to during, the TGP would retrieve all events z in London whose duration is contained in the duration of Oktoberfest. TGP (6) retrieves the first time instant at which Munich hosted Oktoberfest; the answer to (6) on G_1 is

Table 2. Semantics of Temporal Graph Patterns

P	$\delta_X(P)$
B at t_3	$\xi_X(G) \models \exists \mathsf{b}_X(B) : \pi_X(B)\langle t_3\rangle$
B during $[t_1, t_2]$	$\xi_X(G) \models \exists \mathsf{b}_X(B)\ \forall x^{\mathrm{t}} : [t_1 \leq x^{\mathrm{t}} \leq t_2] \rightarrow \pi_X(B)\langle x^{\mathrm{t}}\rangle$
B occurs $[t_1, t_2]$	$\xi_X(G) \models \exists \mathsf{b}_X(B)\ \exists x^{\mathrm{t}} : [t_1 \leq x^{\mathrm{t}} \leq t_2 \wedge \pi_X(B)\langle x^{\mathrm{t}}\rangle]$
B maxint $[t_1, t_2]$	a function $\sigma : \mathsf{b}_X(B) \rightarrow \mathsf{ub}_X(G)$ exists such that $\xi_X(G) \models \forall x^{\mathrm{t}} : [t_1 \leq x^{\mathrm{t}} \leq t_2] \rightarrow \pi_X(\sigma(B))\langle x^{\mathrm{t}}\rangle$, and $t_1 = -\infty$ or $\xi_X(G) \not\models \pi_X(\sigma(B))\langle t_1 - 1\rangle$, and $t_2 = +\infty$ or $\xi_X(G) \not\models \pi_X(\sigma(B))\langle t_2 + 1\rangle$
B mintime t_3	a function $\sigma : \mathsf{b}_X(B) \rightarrow \mathsf{ub}_X(G)$ exists such that $\xi_X(G) \models \pi_X(\sigma(B))\langle t_3\rangle$ and $\xi_X(G) \not\models \pi_X(\sigma(B))\langle x^{\mathrm{t}}\rangle$ for each $x^{\mathrm{t}} \in \mathcal{TI}$ with $x^{\mathrm{t}} \leq t_3 - 1$
B maxtime t_3	a function $\sigma : \mathsf{b}_X(B) \rightarrow \mathsf{ub}_X(G)$ exists such that $\xi_X(G) \models \pi_X(\sigma(B))\langle t_3\rangle$ and $\xi_X(G) \not\models \pi_X(\sigma(B))\langle x^{\mathrm{t}}\rangle$ for each $x^{\mathrm{t}} \in \mathcal{TI}$ with $t_3 + 1 \leq x^{\mathrm{t}}$

Note: $\delta_X(P)$ does not hold if P is malformed (e.g., if it is of the form B at t_3 and $t_3 \notin \mathcal{TI}$); and $\sigma(B)$ is the result of replacing each blank node v in B with $\sigma(v)$.

$\{\{x \mapsto 80\}\}$. Finally, TGP (7) returns all rooms x that have price y during an event z in Munich within the time interval $[50, 100]$.

$$\{\langle x, :\mathit{flightTo}, :\mathit{MUC}\rangle,\ \langle :\mathit{Munich}, :\mathit{hosts}, :\mathit{Oktoberfest}\rangle\}\ \mathsf{maxint}\ [y, z] \tag{4}$$

$$\{\langle :\mathit{Munich}, :\mathit{hosts}, :\mathit{Oktoberfest}\rangle\}\ \mathsf{maxint}\ [x, y]\ \mathsf{and}\ \{\langle :\mathit{London}, :\mathit{hosts}, z\rangle\}\ \mathsf{occurs}\ [x, y] \tag{5}$$

$$\{\langle :\mathit{Munich}, :\mathit{hosts}, :\mathit{Oktoberfest}\rangle\}\ \mathsf{mintime}\ x \tag{6}$$

$$\{\langle x, :\mathit{hasPrice}, y\rangle,\ \langle :\mathit{Munich}, :\mathit{hosts}, z\rangle\}\ \mathsf{occurs}\ [50, 100] \tag{7}$$

We next turn our attention to the formal properties of TGPs. By Definition 4, $\mathsf{ad}_X(G)$ does not contain time constants occurring in G and answers are defined w.r.t. $\mathcal{TC}$, which ensures that answers do not depend on the syntactic form of temporal graphs. For example, temporal graphs G_5 and G_6 mentioned earlier are equivalent and $\mathsf{ad}_X(G_5) = \mathsf{ad}_X(G_6)$, so $[\![P]\!]^X_{G_5} = [\![P]\!]^X_{G_6}$ for each TGP P.

Proposition 1. *Let X be an entailment relation, and let G_1 and G_2 be temporal graphs such that $G_1 \models_X G_2$, $G_2 \models_X G_1$, and $\mathsf{ad}_X(G_1) = \mathsf{ad}_X(G_2)$. Then, for each temporal group pattern P, we have $[\![P]\!]^X_{G_1} = [\![P]\!]^X_{G_2}$.*

Since the answers are defined w.r.t. the entire set $\mathcal{TC}$, temporal basic graph patterns can have infinite answers. We next define a notion of safe TGPs and later show that such group patterns always have finite answers.

Definition 5. *For P a temporal group pattern, $\mathsf{uns}(P)$ is the set of variables as shown in Table 3. Pattern P is* safe *if and only if $\mathsf{uns}(P) = \emptyset$.*

Table 3. The Definition of Safety

P	$\mathsf{uns}(P)$	P	$\mathsf{uns}(P)$
B at t_3	$\{t_3\} \cap \mathcal{V}$	B maxint $[t_1, t_2]$	$\emptyset$
B during $[t_1, t_2]$	$\{t_1, t_2\} \cap \mathcal{V}$	B mintime t_3	$\emptyset$
B occurs $[t_1, t_2]$	$\{t_1, t_2\} \cap \mathcal{V}$	B maxtime t_3	$\emptyset$
P_1 and P_2	$\mathsf{uns}(P_1) \cup [\mathsf{uns}(P_2) \setminus \mathsf{var}(P_1)]$	P_1 union P_2	$\mathsf{uns}(P_1) \cup \mathsf{uns}(P_2)$
P_1 opt P_2	$\mathsf{uns}(P_1) \cup [\mathsf{uns}(P_2) \setminus \mathsf{var}(P_1)]$	P_1 filter R	$\mathsf{uns}(P_1)$

Intuitively, $x \in \mathsf{uns}(P)$ means that there is no guarantee that $\mu(x) \in \mathcal{TC}$ implies $\mu(x) \in \mathsf{tc}_X(G)$ for each $\mu \in [\![P]\!]^X_G$. Thus, B at t_3, B during $[t_1, t_2]$, and B occurs $[t_1, t_2]$ are safe iff t_1, t_2, and t_3 are not variables: B can hold at potentially infinitely many time intervals, which could give rise to infinite answers if t_1, t_2, or t_3 were a variable. In contrast, B maxint $[t_1, t_2]$, B mintime t_3, and B maxtime t_3 are always safe as there are finitely many maximal intervals in which B holds. The nontrivial remaining cases are P_1 and P_2 and P_1 opt P_2, in which we assume that P_1 is evaluated "before" P_2—that is, that the values for variables obtained by evaluating P_1 are used to bind unsafe variables in P_2; this will be made precise shortly in our algorithm for TGP evaluation. Thus, (B_1 occurs $[x, y]$) and (B_2 maxint $[x, y]$) is not safe while (B_2 maxint $[x, y]$) and (B_1 occurs $[x, y]$) is, which may seem odd given that conjunction is commutative. Without a predefined evaluation order, however, we would need to examine every possible order of conjuncts in a conjunction to find an "executable" one, which could be impractical.

We next present an algorithm for evaluating TGPs. We start by showing how to decide three types of temporal entailment that are used as basic building blocks of our evaluation algorithm. We first present some auxiliary definitions. Let G be a temporal graph and X an entailment relation. A pair of time constants (t_1, t_2) is *consecutive* in G if $t_1, t_2 \in \mathsf{tc}_X(G)$, $t_1 < t_2$, and no $t \in \mathsf{tc}_X(G)$ exists with $t_1 < t < t_2$. The *representative* of such (t_1, t_2) is defined as $t_1 + 1$ if $t_1 \neq -\infty$, $t_2 - 1$ if $t_1 = -\infty$ and $t_2 \neq +\infty$, and 0 otherwise. Furthermore, $\mathsf{ti}_X(G) \subseteq \mathcal{TI}$ is the smallest set that contains $\mathsf{tc}_X(G) \cap \mathcal{TI}$ and the representative of each consecutive pair of time constants in G. Finally, note that by Definitions 2 and 3, $\xi_X(G)$ contains $\bigwedge_{u \in \mathsf{ub}_X(G)} O(u) \wedge \Lambda$ and zero or more formulae of the form $\forall x^{\mathsf{t}} : \varphi_i\langle x^{\mathsf{t}}\rangle$, and that Λ is a conjunction of formulae of the form $\psi_i\langle t_i\rangle$ and $\forall x^{\mathsf{t}} : (t_i^1 \leq x^{\mathsf{t}} \leq t_i^2) \rightarrow \kappa_i\langle x^{\mathsf{t}}\rangle$; then, for $t \in \mathcal{TI}$, $\Xi_X(G, t)$ is the set of all $O(u)$, all ψ_i such that $t_i = t$, all κ_i such that $t_i^1 \leq t \leq t_i^2$, and all φ_i.

Proposition 2. *Let G be a temporal graph, let X be an entailment relation, let B be a BGP such that $\mathsf{var}(B) = \emptyset$, and let $t_1 \in \mathcal{TI} \cup \{-\infty\}$, $t_2 \in \mathcal{TI} \cup \{+\infty\}$, and $t_3 \in \mathcal{TI}$. Then, the following claims hold:*

1. *$\xi_X(G)$ is satisfiable iff $\Xi_X(G, t)$ is satisfiable for each $t \in \mathsf{ti}_X(G)$.*
2. *$\xi_X(G) \models \exists \mathsf{b}_X(B) : \pi_X(B)\langle t_3\rangle$ iff $\xi_X(G)$ is unsatisfiable or some function $\sigma : \mathsf{b}_X(B) \rightarrow \mathsf{ub}_X(G)$ exists such that $\Xi_X(G, t_3) \models \pi_X(\sigma(B))$.*

Table 4. Evaluation of Temporal Group Patterns

$\mathsf{eval}_X(P,G)$ is the set of mappings defined as follows depending on the type of P:
$P = B$ at t_3 or $P = B$ during $[t_1,t_2]$ or $P = B$ occurs $[t_1,t_2]$:
$\{\mu \mid \mathsf{dm}(\mu) = \mathsf{var}(B), \mathsf{rg}(\mu) \subseteq \mathsf{ad}_X(G),$ and $\delta_X(\mu(P))$ holds$\}$
$P = B$ maxint $[t_1,t_2]$:
$\{\mu \mid \mathsf{dm}(\mu) = \mathsf{var}(P), \mathsf{rg}(\mu) \subseteq \mathsf{ad}_X(G) \cup \mathsf{ti}_X(G) \cup \{-\infty,+\infty\},$ and $\delta_X(\mu(P))$ holds$\}$
$P = B$ $\mathsf{mintime}$ t_3 :
$\{\mu \mid \mathsf{dm}(\mu) = \mathsf{var}(P), \mathsf{rg}(\mu) \subseteq \mathsf{ad}_X(G) \cup \mathsf{ti}_X(G), \delta_X(\mu(B$ at $t_3))$ holds, and $\delta_X(\mu(B$ at $t'))$ does not hold for all $t' \in \mathsf{ti}_X(G)$ such that $t' \leq \mu(t_3) - 1\}$
$P = B$ $\mathsf{maxtime}$ t_3 :
$\{\mu \mid \mathsf{dm}(\mu) = \mathsf{var}(P), \mathsf{rg}(\mu) \subseteq \mathsf{ad}_X(G) \cup \mathsf{ti}_X(G), \delta_X(\mu(B$ at $t_3))$ holds, and $\delta_X(\mu(B$ at $t'))$ does not hold for all $t' \in \mathsf{ti}_X(G)$ such that $\mu(t_3) + 1 \leq t'\}$
$P = P_1$ and P_2 :
$\{\mu_1 \cup \mu_2 \mid \mu_1 \in \mathsf{eval}_X(P_1,G)$ and $\mu_2 \in \mathsf{eval}_X(\mu_1(P_2),G)\}$
$P = P_1$ union P_2 :
$\mathsf{eval}_X(P_1,G) \cup \mathsf{eval}_X(P_2,G)$
$P = P_1$ opt P_2 :
$\mathsf{eval}_X(P_1$ and $P_2,G) \cup \{\mu \in \mathsf{eval}_X(P_1,G) \mid \mathsf{eval}_X(\mu(P_2),G) = \emptyset\}$
$P = P_1$ filter R :
$\{\mu \in \mathsf{eval}_X(P_1,G) \mid \mu \models R\}$

3. $\xi_X(G) \models \exists \mathsf{b}_X(B)\ \forall x^{\mathsf{t}} : [t_1 \leq x^{\mathsf{t}} \leq t_2] \rightarrow \pi_X(B)\langle x^{\mathsf{t}}\rangle$ *iff* $\xi_X(G)$ *is unsatisfiable or some* $\sigma : \mathsf{b}_X(G) \rightarrow \mathsf{ub}_X(G)$ *exists such that* $\Xi_X(G,t) \models \pi_X(\sigma(B))$ *for each* $t \in \mathsf{ti}_X(G)$ *with* $t_1 \leq t \leq t_2$.
4. $\xi_X(G) \models \exists \mathsf{b}_X(B)\ \exists x^{\mathsf{t}} : [t_1 \leq x^{\mathsf{t}} \leq t_2 \wedge \pi_X(B)\langle x^{\mathsf{t}}\rangle]$ *iff* $\xi_X(G)$ *is unsatisfiable or some* $\sigma : \mathsf{b}_X(B) \rightarrow \mathsf{ub}_X(G)$ *exists such that* $\Xi_X(G,t) \models \pi_X(\sigma(B))$ *for some* $t \in \mathsf{ti}_X(G)$ *with* $t_1 \leq t \leq t_2$.

Proposition 2 reduces temporal entailment to standard entailment problems that can be solved using any decision procedure available. This provides us with a way to check conditions $\delta_X(\mu(P))$ needed to evaluate safe TGPs. Furthermore, note that Claim 3 can be straightforwardly extended to general temporal graph entailment. We use these results as building blocks for the function shown in Table 4 that evaluates safe temporal group patterns. For P a basic TGP, $\mathsf{eval}_X(P,G)$ can be computed by enumerating all mappings potentially relevant to P and then eliminating those mappings that do not satisfy the respective conditions; optimizations can be used to quickly eliminate irrelevant mappings.

Proposition 3. *Let* G *be a temporal graph, let* X *be an entailment relation such that* $\mathsf{ad}_X(G)$ *is finite, and let* P *be a safe temporal group pattern. Then* $\mathsf{eval}_X(P,G) = [\![P]\!]_G^X$ *and* $[\![P]\!]_G^X$ *is finite.*

5 Optimized Query Answering

The algorithm from Table 4 checks temporal entailment using a black box decision procedure, which can be inefficient. In this section we first present an optimization of this algorithm that is applicable to simple entailment, and then we extend this approach to any entailment relation that can be characterized by deterministic rules, such as RDF(S) and OWL 2 RL.

5.1 Simple Entailment

Simple entailment is the basic entailment relation in which BGPs can be evaluated in nontemporal graphs by simple graph lookup. Such an approach provides the basis of virtually all practical RDF storage systems and has proved itself in practice, so it would be beneficial if similar approaches were applicable to TGPs and temporal graphs. As the following example demonstrates, however, this is not the case. Let $P = \{\langle :LHR, :\mathit{flightTo}, :MUC\rangle\}$ $\mathsf{maxint}\,[x, y]$; then $[\![P]\!]^{\mathit{simple}}_{G_1} = \{\{x \mapsto 50, y \mapsto 150\}\}$. Note, however, that G_1 does not contain temporal triple $\alpha = \langle :LHR, :\mathit{flightTo}, :MUC\rangle[50, 150]$, so $[\![P]\!]^{\mathit{simple}}_{G_1}$ cannot be computed via lookup. Temporal graph G_1 is, however, equivalent to the normalized temporal graph $\mathsf{nrm}(G_1)$ obtained from G_1 by replacing (1) with α; then, P can be evaluated in $\mathsf{nrm}(G_1)$ via lookup, which simplifies query processing. TGPs of other types can additionally require adequate interval comparisons.

We next formalize this idea. We say that temporal triples $\langle s, p, o\rangle[t_1, t_2]$ and $\langle s', p', o'\rangle[t'_1, t'_2]$ *overlap* if $s = s'$, $p = p'$, $o = o'$, and $\max(t_1, t'_1) \leq \min(t_2, t'_2)$; this definition is extended to triples of the form $\langle s, p, o\rangle[t_1]$ by treating them as abbreviations for $\langle s, p, o\rangle[t_1, t_1]$. Let G be a temporal graph and let $A \in G$ be a temporal triple. The *maximal subset* of G w.r.t. A is the smallest set $G_A \subseteq G$ such that $A \in G_A$ and, if $\beta \in G_A$, $\gamma \in G$, and β and γ overlap, then $\gamma \in G_A$ as well. The *normalization* of G is the temporal graph $\mathsf{nrm}(G)$ that, for each $A \in G$ of the form $\langle s, p, o\rangle[t_1, t_2]$ or $\langle s, p, o\rangle[t_1]$, contains the temporal triple $\langle s, p, o\rangle[t'_1, t'_2]$ where t'_1 and t'_2 are the smallest and the largest temporal constant, respectively, occurring in the maximal subset G_A of G w.r.t. A.

Let G' be the list of the temporal triples in G of the form $\langle s, p, o\rangle[t_1, t_2]$ and $\langle s, p, o\rangle[t_1]$ sorted by s, p, o, t_1, and t_2. For each $A \in G$, the triples that constitute the maximal subset G_A of G occur consecutively in G', so $\mathsf{nrm}(G)$ can be computed by a simple sequential scan through G'.

We next show how to use $\mathsf{nrm}(G)$ to evaluate temporal group patterns. Let $B = \{\langle s_1, p_1, o_1\rangle, \ldots, \langle s_k, p_k, o_k\rangle\}$ be a BGP, let $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$ be variables not occurring in B, and let G be a temporal graph. Then $\langle B\rangle^G$ is the set of all mappings μ such that $\mathsf{dm}(\mu) = \mathsf{var}(B) \cup \{x_1, y_1, \ldots, x_k, y_k\}$, $\mu(\langle s_i, p_i, o_i\rangle[x_i, y_i]) \in \mathsf{nrm}(G)$ for each $1 \leq i \leq k$, and $\mu^{\downarrow} \leq \mu^{\uparrow}$, where the latter are defined as $\mu^{\downarrow} = \max\{\mu(x_1), \ldots, \mu(x_k)\}$ and $\mu^{\uparrow} = \min\{\mu(y_1), \ldots, \mu(y_k)\}$. Furthermore, $\nu|_B$ if the restriction of a mapping ν to $\mathsf{var}(B)$. Table 5 then shows how to evaluate basic TGPs under simple entailment in a normalization.

Table 5. Evaluation of Temporal Group Patterns under Simple Entailment

$\mathsf{eval}_{simple}(P, G)$ is the set of mappings defined as follows:

$P = B$ at t_3 :
$\{\nu \mid \mathsf{dm}(\nu) = \mathsf{var}(P)$ and $\exists \mu \in \langle B \rangle^G : \nu|_B = \mu|_B \wedge \mu^{\downarrow} \leq t_3 \leq \mu^{\uparrow}\}$

$P = B$ during $[t_1, t_2]$:
$\{\nu \mid \mathsf{dm}(\nu) = \mathsf{var}(P)$ and $\exists \mu \in \langle B \rangle^G : \nu|_B = \mu|_B \wedge \mu^{\downarrow} \leq t_1 \leq t_2 \leq \mu^{\uparrow}\}$

$P = B$ occurs $[t_1, t_2]$:
$\{\nu \mid \mathsf{dm}(\nu) = \mathsf{var}(P)$ and $\exists \mu \in \langle B \rangle^G : \nu|_B = \mu|_B \wedge \max(\mu^{\downarrow}, t_1) \leq \min(\mu^{\uparrow}, t_2)\}$

$P = B$ maxint $[t_1, t_2]$:
$\{\nu \mid \mathsf{dm}(\nu) = \mathsf{var}(P)$ and $\exists \mu \in \langle B \rangle^G : \nu|_B = \mu|_B \wedge \nu(t_1) = \mu^{\downarrow} \wedge \nu(t_2) = \mu^{\uparrow}\}$

$P = B$ $\mathsf{mintime}$ t_3 :
$\{\nu \mid \mathsf{dm}(\nu) = \mathsf{var}(P)$ and $\exists \mu \in \langle B \rangle^G :$
$\mu^{\downarrow} \in \mathcal{TI} \wedge \nu|_B = \mu|_B \wedge \nu(t_3) = \mu^{\downarrow} \wedge \forall \lambda \in \langle B \rangle^G : \mu|_B = \lambda|_B \rightarrow \mu^{\downarrow} \leq \lambda^{\downarrow}\}$

$P = B$ $\mathsf{maxtime}$ t_3 :
$\{\nu \mid \mathsf{dm}(\nu) = \mathsf{var}(P)$ and $\exists \mu \in \langle B \rangle^G :$
$\mu^{\uparrow} \in \mathcal{TI} \wedge \nu|_B = \mu|_B \wedge \nu(t_3) = \mu^{\uparrow} \wedge \forall \lambda \in \langle B \rangle^G : \mu|_B = \lambda|_B \rightarrow \lambda^{\uparrow} \leq \mu^{\uparrow}\}$

Proposition 4. *For each temporal graph G and each safe temporal group pattern P, we have* $\mathsf{eval}_{simple}(P, G) = [\![P]\!]_G^{simple}$.

We explain the intuition behind this algorithm for $P = B$ maxint $[t_1, t_2]$. First, we compute $\langle B \rangle^G$ by evaluating the conjunctive query $\bigwedge \langle s_i, p_i, o_i \rangle [x_i, y_i]$ in $\mathsf{nrm}(G)$ via simple lookup. Consider now an arbitrary $\mu \in \langle B \rangle^G$. By the definition of normalization, each $\mu(x_i)$ and $\mu(y_i)$ determine the maximal validity interval of $\langle s_i, p_i, o_i \rangle$ so, to answer P, we must intersect all intervals $[\mu(x_i), \mu(y_i)]$. Note that $\mu^{\downarrow}$ and $\mu^{\uparrow}$ give the lower and the upper limit of the intersection, provided that $\mu^{\downarrow} \leq \mu^{\uparrow}$. Thus, what remains to be done is to convert μ into ν by setting $\nu(x) = \mu(x)$ for each $x \in \mathsf{var}(B)$ and ensuring that $\nu(t_1) = \mu^{\downarrow}$ and $\nu(t_2) = \mu^{\uparrow}$. Based on these ideas, EO's system translates TGPs into SQL, which allows us to use a standard database query planner to optimize query execution.

5.2 Entailments Characterized by Deterministic Rules

Let X be an entailment relation that can be characterized by a set Γ_X of deterministic rules of the form (8).

$$A_1 \wedge \ldots \wedge A_n \rightarrow B \tag{8}$$

To evaluate a SPARQL group pattern in a graph under X-entailment, most existing (nontemporal) RDF systems first compute the closure of the graph w.r.t. Γ_X. We next show how to compute the temporal closure $\mathsf{cls}_X(G)$ of a temporal graph G using the rules from Γ_X. After computing the closure, we can normalize it and then apply the algorithm from Section 5.1.

Definition 6. *For X and Γ_X as stated above, let Σ_X be the set containing the rule* (9) *for each rule* (8) *in Γ_X.*

$$A_1[x_1, y_1] \wedge \ldots \wedge A_n[x_n, y_n] \wedge \max(x_1, \ldots, x_n) \leq \min(y_1, \ldots, y_n) \rightarrow B[\max(x_1, \ldots, x_n), \min(y_1, \ldots, y_n)] \quad (9)$$

Let G be a temporal graph consisting of triples of the form $\langle s, p, o\rangle[t_1, t_2]$.[3] *The* skolemization *of G is a temporal graph obtained from G be replacing each blank node with a fresh URI reference. Furthermore, the* temporal closure *of G is the (possibly infinite) temporal graph $\mathsf{cls}_X(G)$ obtained by exhaustively applying the rules in Σ_X to the skolemization of G.*

By applying this approach to G_1 under RDFS entailment, one can see that (2) and (3) produce $\langle{:}\mathit{Munich}, {:}\mathit{hasAttraction}, {:}\mathit{Oktoberfest}\rangle[130, 180]$.

The following proposition shows that, instead of evaluating TGPs in G under X-entailment, one can evaluate them in $\mathsf{cls}_X(G)$ under simple entailment.

Proposition 5. *Let X and G be as stated in Definition 6. For each temporal group pattern P, we have $[\![P]\!]^X_G = [\![P]\!]^{simple}_{G'}$, where $G' = \mathsf{cls}_X(G)$.*

6 Implementation and Outlook

In this paper we presented an approach for representing validity time in RDF and OWL, an extension of SPARQL that allows for querying temporal graphs, and two query answering algorithms. We implemented our approach in EO's knowledge representation system. The system is based on a proprietary extension of RDF that supports n-ary relations; it uses an ontology language based on OWL 2 RL; and it implements a proprietary query language based on the primitives and the notion of safety outlined in Section 4. The PostgreSQL database is used for data persistence and query processing. Ontology reasoning is implemented by translating the ontology into a datalog program, which is then compiled into a plSQL script that implements the seminaïve datalog evaluation strategy. Datalog rules are modified as described in Section 5.2 in order to deal with validity time; furthermore, the resulting set of facts obtained by applying the rules is normalized to allow for efficient query answering. Finally, temporal queries are translated into SQL and then evaluated using PostgreSQL's query engine; the translation essentially encodes the query evaluation algorithm from Section 5.1. The source of the system is not open, and EO has no plans for licensing the system to third-party developers. Therefore, we do not present a performance evaluation since such results could not be validated by the community. We merely note that EO is successfully using our approach with datasets consisting of tens of millions of triples, which we take as confirmation that our approach is amenable to practical implementation.

An important open theoretical question is to determine the worst-case complexity bounds of the query answering problem for our query language. Furthermore, one should see whether the general algorithm from Section 4 can be

[3] For simplicity we assume that G does not contain triples of the form $\langle s, p, o\rangle[t_1]$.

successfully used with expressive languages such as OWL 2 DL. We believe this to be possible provided that the algorithm is adequately optimized.

References

1. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM 26(11), 832–843 (1983)
2. Artale, A., Franconi, E.: A survey of temporal extensions of description logics. Annals of Mathematics and Artificial Intelligence 30(1-4), 171–210 (2000)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications, 2nd edn. Cambridge University Press, Cambridge (August 2007)
4. Chomicki, J.: Temporal Query Languages: A Survey. In: Proc. ICTL, pp. 506–534 (1994)
5. Fitting, M.: First-Order Logic and Automated Theorem Proving, 2nd edn. Texts in Computer Science. Springer, Heidelberg (1996)
6. Gutiérrez, C., Hurtado, C.A., Mendelzon, A.O.: Foundations of Semantic Web Databases. In: Proc. PODS, pp. 95–106 (2004)
7. Gutierrez, C., Hurtado, C.A., Vaisman, A.A.: Introducing Time into RDF. IEEE Transactions on Knowledge and Data Engineering 19(2), 207–218 (2007)
8. Hayes, P.: RDF Semantics, W3C Recommendation (February 10, 2004)
9. Hurtado, C.A., Vaisman, A.A.: Reasoning with Temporal Constraints in RDF. In: Alferes, J.J., Bailey, J., May, W., Schwertel, U. (eds.) PPSWR 2006. LNCS, vol. 4187, pp. 164–178. Springer, Heidelberg (2006)
10. Milea, V., Frasincar, F., Kaymak, U.: Knowledge Engineering in a Temporal Semantic Web Context. In: Proc. ICWE, pp. 65–74 (2008)
11. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language: Profiles. W3C Recommendation (October 27, 2009)
12. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax, W3C Recommendation (October 27, 2009)
13. Patel-Schneider, P.F., Motik, B.: OWL 2 Web Ontology Language: Mapping to RDF Graphs, W3C Recommendation (October 27, 2009)
14. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Transactions on Database Systems 34(3) (2009)
15. Pugliese, A., Udrea, O., Subrahmanian, V.S.: Scaling RDF with Time. In: Proc. WWW, pp. 605–614 (2008)
16. Schneider, M.: OWL 2 Web Ontology Language: RDF-Based Semantics, W3C Recommendation (October 27, 2009)
17. Straccia, U., Lopes, N., Lukácsy, G., Polleres, A.: A General Framework for Representing and Reasoning with Annotated Semantic Web Data. In: Proc. AAAI (2010)
18. Tappolet, J., Bernstein, A.: Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 308–322. Springer, Heidelberg (2009)
19. Vila, L.: A Survey on Temporal Reasoning in Artificial Intelligence. AI Communications 7(1), 4–28 (1994)