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

Boris Motik[a]

[a] *Oxford University Computing Laboratory, Oxford, UK*

Abstract

RDF(S) and OWL 2 can currently represent only static information. In practice, however, the truth of statements often changes with time. 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(S) and OWL 2. Unlike the existing proposals, our approach is applicable to nondeterministic entailment relations and/or entailment relations that involve existential quantification, such as the OWL 2 Direct Entailment and the OWL 2 RDF-Based Entailment. We also present an extension of SPARQL that can be used to query temporal RDF(S) and OWL 2. Moreover, we present a general query evaluation algorithm that can be used with all entailment relations used in the Semantic Web. Finally, we present two optimizations of the algorithm that are applicable to entailment relations characterized by a set of deterministic rules, such RDF(S) and OWL 2 RL/RDF Entailment.

Keywords: RDF, OWL, SPARQL, validity time, temporal databases, query answering

1. Introduction

The Resource Description Framework (RDF) [1] is the standard language for metadata management in the Semantic Web. RDF provides a simple, but powerful data model that encodes facts using *triples*; sets of triples are called *RDF graphs*. RDF Schema (RDFS) extends RDF with vocabulary for schema modeling. The Web Ontology Language (OWL), particularly in its latest incarnation called OWL 2 [2], extends RDFS to a very expressive ontology language. The semantics of RDF graphs is determined by *entailment relations*, such as Simple Entailment or OWL 2 RL/RDF Entailment.

RDF(S) and OWL 2 have been implemented in RDF management systems such as Jena [3], Jena2 [4], Sesame [5], Oracle [6], Kowari [7], 4store [8], YARS [9], Hexastore [10], and Redland [11], as well as reasoners such as FaCT++ [12], Pellet [13], and HermiT [14]. Such systems were successfully applied in numerous applications in fields as diverse as biology [15], medicine [16], geography [17], astronomy [18], agriculture [19], and defense [20]. These applications, however, often need to deal with information that is not static, but that changes with time. To understand this problem, we present an example derived from the author's collaboration with ExperienceOn[1]—a start-up IT company from Barcelona, Spain.

ExperienceOn aims to improve search in the tourism domain by providing an advanced system that can answer complex requests 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. ExperienceOn's system must be able to represent statements that are not universally true, but are associated with *validity times*—specifications of time instants at which the statements are true. For example, "Oktoberfest is being held in Munich" is true only while the festival is actually 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, ExperienceOn's system should be able to 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 many advanced application scenarios. Some applications also need to represent *transaction times*, which specify the period during which a fact is present in the system [21]. Transaction times are important in the context of data management, but they are not part of an application domain's description. Since modeling the latter is the primary concern in the Semantic Web, in this paper we focus on validity rather than transaction time. Note, however, that the techniques for the management of one can kind of time often be applied to the other kind of time as well.

[☆] This is an extended version of a paper with the same name published at ISWC 2010, pp. 550–565.

[1] `http://www.experienceon.com/`

RDF(S) and OWL 2 can represent only static facts whose truth does not change with time. This deficiency has been recognized in the Semantic Web community, and a number of approaches for the representation, reasoning, and querying of validity time were developed [22, 23, 24, 25, 26, 27, 28, 29]; we discuss various aspects of these approaches in Section 7. These approaches, however, do not provide a general framework for the management of validity time in the Semantic Web: all approaches known to us are applicable only to entailment relations that can be characterized by deterministic inference rules without existential quantifiers, such as RDFS or OWL 2 RL.

In this paper we present a novel logic-based approach for representing validity time that is applicable to all entailment relations of the Semantic Web, including RDF(S), OWL 2 Direct Semantics, and all profiles of OWL 2. To this end, in Section 3 we introduce a notion of temporal RDF graphs similar to the one used in [22]. Instead of developing a notion of temporal entailment "from scratch," we give semantics to temporal graphs by simply mapping them into first-order theories.

In Section 4, we turn our attention to the problem of querying temporal graphs. We argue that a naïve extension of SPARQL with temporal features is likely to be problematic. For example, let G be the temporal graph in which some triple A holds from a fixed time instant t_1 onwards, and let Q_1 be a query for "all time instants at which A holds"; the answer to Q_1 on G is infinite since the time line is unbounded. Furthermore, even if we restrict temporal triples to finite intervals, the answers to queries such as Q_1 can be much larger than the graphs to which the queries are applied. An obvious way to overcome this problem is to restrict answers to time instants explicitly occurring in the temporal graph; in our example, the answer to Q_1 on G would then contain only t_1. Under such a definition, however, evaluating the same query over semantically equivalent temporal graphs could produce different answers. For example, consider a temporal graph G' in which triple A holds between time instants t_1 and t_2, and from t_3 onwards, where $t_1 < t_3 < t_2$. The constraints on t_1, t_2, and t_3 ensure that G and G' are semantically equivalent; however, the answer to Q_1 on G' would contain t_1, t_2, and t_3, which is different from the answer to Q_1 on G. We address these problems in two steps:

- We extend the formalization of SPARQL by Pérez et al. [30] with primitives that allow for querying temporal graphs. For each query expressed in our language, the answers on semantically equivalent temporal graphs are guaranteed to coincide. Our query language explicitly incorporates the notions of maximality/minimality by allowing for queries such as Q_2 = "the maximal interval in which A holds" and Q_3 = "the minimal time instant at which A holds."

- We define a syntactic notion of safety that guarantees finiteness of temporal query answers. Queries such as Q_2 and Q_3 are safe, but queries such as Q_1 are not; intuitively, only queries that ask for maximal intervals can bind variables in a query.

In Section 5 we then turn our attention to algorithms for dealing with temporal graphs.

- In Section 5.1 we present an algorithm that can decide certain basic types of temporal entailments. Our algorithm reduces a temporal entailment problem to linearly many nontemporal entailment problems. Thus, the algorithm can be applied to an arbitrary temporal entailment relation for which the nontemporal counterpart is available, including relations requiring nondeterministic reasoning and/or reasoning with existential quantifiers, such as OWL 2 Direct Entailment or OWL 2 RDF-Based Entailment.

- In Section 5.2 we present an algorithm for computing answers to safe temporal queries. The query answering algorithm uses the temporal entailment algorithm as a black box and thus handles arbitrary temporal entailment relations.

In Section 6 we consider several optimizations of the general query answering algorithm.

- In Section 6.1 we present a query answering algorithm that is applicable to Simple Entailment. The algorithm consists of two parts: a temporal graph is preprocessed into a particular normal form, after which the evaluation of temporal queries is reduced to simple evaluation of conjunctive queries over the normalization. The evaluation of conjunctive queries is implemented in virtually all RDF management systems, so integrating our algorithm into these systems should be straightforward.

- In Section 6.2 we present an algorithm for reasoning with temporal graphs that is applicable to deterministic entailment relations such as RDF(S) and OWL 2 RL/RDF. Practical RDF systems usually reason with such entailment relations on nontemporal graphs by materializing all consequences of a suitable datalog program. We show how to modify an arbitrary such program into one that correctly processes the required temporal information. Our approach can thus easily be integrated into existing materialization-based RDF management systems.

2. Preliminaries

In this section we recapitulate some basic definitions of first-order logic, RDF, OWL, and SPARQL.

2.1. First-Order Logic and Sorts

We use the notions of constants, variables, function symbols, terms, predicates, atoms, and formulae from first-order logic [31]. For σ a (partial) mapping of variables to terms and ψ a first-order formula, $\sigma(\psi)$ is the result of replacing in ψ, for each variable x such that $\sigma(x)$ is defined, each free occurrence of x with $\sigma(x)$. A first-order interpretation I is a tuple $I = (\triangle^I, \cdot^I)$, where $\triangle^I$ is a nonempty domain set of I, and $\cdot^I$ is a function that assigns an interpretation X^I to each predicate, function symbol, and constant X. Let φ and χ be arbitrary first-order formulae without free variables. We write $I \models \varphi$ to denote that φ is satisfied in a first-order interpretation I; then, I is a *model* of φ. Furthermore, we write $\varphi \models \chi$ to denote that $I \models \chi$ for each I such that $I \models \varphi$; then, φ *entails* χ.

The *skolemization* of a first-order formula φ is a formula ψ obtained by replacing existentially quantified variables occurring in φ with function terms as described in [31]. For example, the skolemization of formula φ shown below is the formula ψ, where c is a fresh constant and f is a fresh function symbol.

$$\begin{aligned}\varphi &= [\exists x : A(x)] \wedge [\forall x : A(x) \to \exists y : [R(x,y) \wedge A(y)]] \\ \psi &= A(c) \wedge [\forall x : A(x) \to [R(x, f(x)) \wedge A(f(x))]]\end{aligned}$$

For arbitrary first-order formulae φ and χ without free variables, and for ψ the skolemization of φ, we have $\psi \models \varphi$ (but $\varphi \models \psi$ does not hold in general), and $\varphi \models \chi$ if and only if $\psi \models \chi$ [31].

For reasons that will become apparent in Section 3, we assume in this paper that the equality predicate $\approx$ is treated as an ordinary first-order predicate with an explicit axiomatization. With $\Gamma_\approx$ we denote the axioms of equality from [31] instantiated for all the symbols in the signature. (Note that $\Gamma_\approx$ is infinite if the signature is infinite.)

For reasons that will also become apparent in Section 3, in this paper we use the *many-sorted* variant of first-order logic. Please refer to [32] for a formal definition of many-sorted logic. Roughly speaking, variables in many-sorted logic range over fixed subsets of the domain sets.

2.2. RDF and OWL

The syntax of RDF [1] is defined w.r.t. infinite sets $\mathcal{U}$, $\mathcal{B}$, and $\mathcal{L}$ of *URI references*, *blank nodes*, and *literals*, respectively. Let $\mathcal{UBL} = \mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$. An *RDF triple* (usually abbreviated to just *triple*) is an assertion of the form $\langle s, p, o\rangle$ such that $s, p, o \in \mathcal{UBL}$.[2] An *RDF graph* (usually abbreviated to just *graph*) G is a finite set of triples. With $\mathsf{ul}(G)$ and $\mathsf{b}(G)$ we denote the subsets of the elements of $\mathcal{U} \cup \mathcal{L}$ and $\mathcal{B}$, respectively, that occur in G. Blank nodes are usually written as $\_{:}x$. For σ a (partial) mapping of the blank nodes in G into $\mathcal{UBL}$, with $\sigma(G)$ we denote the temporal graph obtained form G by replacing with $\sigma(\_{:}x)$ each blank node $\_{:}x$ for which $\sigma(\_{:}x)$ is defined.

The semantics of RDF is defined by a model theory [33] that maps a graph G to RDF interpretations. In this paper, however, we use a different, but nevertheless equivalent formulation: we map G to a first-order theory $\nu(G)$ that captures the consequences of G. For convenience, we assume that each $t \in \mathcal{U} \cup \mathcal{L}$ corresponds to a first-order constant t, and that each blank node $\_{:}x \in \mathcal{B}$ corresponds to a first-order variable $\_{:}x$. Our mapping encodes the truth of triples using a ternary first-order predicate T—that is, each triple $A = \langle s, p, o\rangle$ is mapped into a first-order atom $\chi(A) = T(s, p, o)$; furthermore, a graph G is mapped into a formula $\chi(G) = \exists \mathsf{b}(G) : \bigwedge_{A\in G} \chi(A)$, where $\exists \mathsf{b}(G)$ abbreviates $\exists \_{:}x_1 \ldots \exists \_{:}x_k$ for $\{\_{:}x_1, \ldots, \_{:}x_k\} = \mathsf{b}(G)$.

Several *entailment relations* can be used with RDF: *Simple Entailment*, *RDF Entailment*, *RDFS Entailment*, and *D-Entailment* are defined in [33], *OWL 2 RDF-Based Entailment* is defined in [34], and *OWL 2 RL/RDF Entailment* is defined in [35]. An entailment relation X places constraints on RDF interpretations; in our framework, we can capture these constraints using a first-order theory Γ_X (i.e., a set of first-order formulae) in which T and $\approx$ are the only predicates. For example, for Simple Entailment, $\Gamma_{simple} = \emptyset$; for RDF Entailment, Γ_{RDF} contains the rules in [33, Section 7]; and for OWL 2 RL/RDF Entailment, Γ_{RL} contains the rules in [35, Section 4.3]. We assume that Γ_X does not contain the elements of $\mathcal{B}$, and that, if the equality predicate $\approx$ is used in Γ_X, then Γ_X contains the explicit axiomatization of equality $\Gamma_\approx$. Note that Γ_X is not required to be finite. The semantics of a graph G under entailment relation X is then captured by the first-order theory $\nu_X(G) = \{\chi(G)\} \cup \Gamma_X$. A graph G_1 *X-entails* a graph G_2, written $G_1 \models_X G_2$, if $\nu_X(G_1) \models \nu_X(G_2)$; the latter condition is clearly equivalent to $\nu_X(G_1) \models \chi(G_2)$. Finally, for convenience, we define the following notation:

$$\mathsf{ul}_X(G) = \mathsf{ul}(G) \qquad \mathsf{b}_X(G) = \mathsf{b}(G)$$
$$\pi_X(G) = \bigwedge_{A\in G} \chi(A)$$

As defined above, $\pi_X(G)$, $\mathsf{b}_X(G)$, and $\mathsf{ul}_X(G)$ do not depend on X if X is an entailment relation mentioned at the beginning of this paragraph. However, we next present different definitions of $\pi_X(G)$, $\mathsf{b}_X(G)$, and $\mathsf{ul}_X(G)$ for the case when X is OWL 2 Direct Entailment. In this way, we obtain notation that allows us to uniformly refer to an arbitrary entailment relation of RDF and OWL 2.

The *OWL 2 Direct Entailment* (written *DL* due to its relationship with description logics [36]) employs a completely different approach to interpreting graphs. A graph G is said to *encode an OWL 2 DL ontology* if G can be transformed into an OWL 2 DL ontology $\mathcal{O}(G)$ as specified in [37]. This process is rather complex, and we cannot discuss it in detail. Roughly speaking, various *triple patterns* are applied to G in order to extract parts of $\mathcal{O}(G)$. Axioms of $\mathcal{O}(G)$ are "recognized" by triple patterns listed in [37, Tables 16 and 17]. Each of these triple patterns contains either a single *main triple* or a single triple of the form $\langle \_{:}x, rdf{:}type, y\rangle$ where $\_{:}x$ is a blank node and y is an OWL 2 resource specifying the type of the axiom.

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

We call a triple of either of these two forms a *lead triple*; intuitively, a lead triple is a triple that "represents" the axiom in G. Different conditions are required to hold at various stages during the transformation process. If at any point in time a condition becomes invalidated, the transformation fails and the input RDF graph cannot be used with OWL 2 Direct Entailment. In contrast, if the transformation of G into $\mathcal{O}(G)$ is successful, ontology $\mathcal{O}(G)$ is interpreted as specified in [38]. To obtain a common semantic framework, however, in this paper we equivalently map $\mathcal{O}(G)$ into a first-order formula. More precisely, for an OWL 2 DL axiom α, let $\theta(\alpha)$ be the translation of α into a first-order formula [36] in which blank nodes occur as free variables; note that this translation uses the equality predicate $\approx$ to interpret features such as number restrictions and nominals. Let $\mathsf{ul}_{DL}(G)$ and $\mathsf{b}_{DL}(G)$ be the subsets of the elements of $\mathcal{U} \cup \mathcal{L}$ and $\mathcal{B}$, respectively, that occur in $\mathcal{O}(G)$. The semantics of a graph G under OWL 2 Direct Entailment is then captured by the first-order theory $\nu_{DL}(G)$ defined as follows:

$$\pi_{DL}(G) = \bigwedge_{\alpha \in \mathcal{O}(G)} \theta(\alpha)$$

$$\nu_{DL}(G) = \{\exists \mathsf{b}_{DL}(G) : \pi_{DL}(G)\} \cup \Gamma_{\approx}$$

To clarify, in the above definition we assume that $\Gamma_{\approx}$ axiomatizes the equality predicate $\approx$ for all symbols in the signature, not just for those occurring in $\mathcal{O}(G)$. For G_1 and G_2 graphs that encode OWL 2 DL ontologies, G_1 *DL-entails* G_2, written $G_1 \models_{DL} G_2$, if $\nu_{DL}(G_1) \models \nu_{DL}(G_2)$.

A remark about notation is in order. Note that, for an arbitrary entailment relation X and graphs G, G_1, and G_2 that can be interpreted under X, $\pi_X(G)$ is a first-order formula whose free variables (if any) are contained in $\mathcal{B}$; furthermore, $\nu_X(G)$ is a first-order theory that contains a formula of the form $\exists \mathsf{b}_{DL}(G) : \pi_X(G)$ plus possibly other formulae without blank nodes; finally, $G_1 \models_X G_2$ if and only if $\nu_X(G_1) \models \exists \mathsf{b}_X(G_2) : \pi_X(G_2)$.

The result of *skolemizing* the blank nodes in $\nu_X(G)$ is the first-order theory $\xi_X(G)$ obtained from $\nu_X(G)$ by removing $\exists \mathsf{b}_X(G)$ and replacing each blank node in $\pi_X(G)$ with a fresh URI reference. We assume that each blank node is uniquely associated with a fresh URI reference used for skolemization; furthermore, we define $\mathsf{usl}_X(G)$ as $\mathsf{ul}_X(G)$ extended with the URI references obtained by the skolemization of $\nu_X(G)$. For arbitrary G_1 and G_2, by the properties of skolemization we have $G_1 \models_X G_2$ if and only if $\xi_X(G_1) \models \exists \mathsf{b}_X(G_2) : \pi_X(G_2)$.

The *X-skolemization* of a graph G is the graph G' obtained from G by replacing each blank node in $\mathsf{b}_X(G)$ with the corresponding URI reference from $\mathsf{usl}_X(G)$. Note that, for X other than DL, graph G' does not contain blank nodes; however, for $X = DL$, graph G' can contain blank nodes: $\mathsf{b}_X(G)$ contains only the blank nodes that occur in $\mathcal{O}(G)$, but not the ones that encode syntactic parts of OWL 2 DL axioms (such as *owl*:*SomeValuesFrom* restrictions). In all cases, however, we have $\nu_X(G') = \xi_X(G')$; furthermore, $\xi_X(G)$ is always semantically equivalent with $\nu_X(G')$. Thus, it does not matter whether we transform G into $\nu_X(G)$ and then skolemize the latter, or we skolemize G and then transform the resulting graph G' into $\nu_X(G')$: the resulting theories are semantically equivalent.

2.3. SPARQL

We next present an overview of SPARQL—the standard W3C language for querying RDF graphs. 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 along the lines of [30]. Let $\mathcal{V}$ be an infinite set of *variables* disjoint with $\mathcal{UBL}$. A *mapping* is a partial function $\mu : \mathcal{V} \to \mathcal{UBL}$. The domain and the range of μ are given by $\mathsf{dom}(\mu)$ and $\mathsf{rng}(\mu)$, respectively. We define $\mu(t) = t$ for each $t \in \mathcal{UBL} \cup \mathcal{V} \setminus \mathsf{dom}(\mu)$. Mappings μ_1 and μ_2 are *compatible* if $\mu_1(x) = \mu_2(x)$ for each $x \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$; in such a case, $\mu_1 \cup \mu_2$ is also a mapping. We use the following algebraic operations on sets of mappings Ω_1 and Ω_2 to define the semantics of group patterns.

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

A *built-in expression* is constructed using the elements of $\mathcal{V} \cup \mathcal{U} \cup \mathcal{L}$ as specified in [30]; 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 [30]. 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 inductively defined as a basic group pattern or an expression of the form P_1 and P_2, P_1 union P_2, P_1 opt P_2, or P_1 filter R, where P_1 and P_2 are group patterns, and R is a built-in expression. Let A be a built-in expression or a group pattern, and let μ a be a mapping; then $\mathsf{var}(A)$ is the set of variables occurring in A, and $\mu(A)$ is the result of replacing in A, for each variable x where $\sigma(x)$ is defined, each occurrence of x with $\sigma(x)$.

We next define the notion of answers to group patterns on an RDF graph. Please note that the 1.0 version of SPARQL does not specify how to evaluate BGPs under entailment relations other than Simple Entailment. The 1.1 version of SPARQL is expected to correct this, but the details have not yet been finalized. In this paper, we adopt definitions that, we believe, are likely candidates for the formalization of SPARQL 1.1. Our definitions address the following two technical problems.

First, we must ensure that answers are always finite. Consider $B = \{\langle x, \textit{rdf:type}, \textit{rdf:Property}\rangle\}$. Due to the axiomatic triples of RDF Entailment [33], $\emptyset \models_{RDF} \mu(B)$ for each μ such that $\mu(x) \in \{\textit{rdf:\_1}, \textit{rdf:\_2}, \ldots\}$. Thus, if the answer to B on the empty RDF graph under RDF Entailment is to be finite, it cannot contain such mappings. As a remedy, we assume that, for each entailment relation X and each graph G, one can determine the *answer domain* $\mathsf{ad}_X(G) \subseteq \mathcal{UBL}$ of G w.r.t. X. Then, for each group pattern P and each mapping μ contained in the answer to P on G w.r.t. X, we require $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G)$; thus, $\mathsf{ad}_X(G)$ identifies the subset of $\mathcal{UBL}$ that is allowed to occur in an answer to P in G w.r.t. X. If $\mathsf{ad}_X(G)$ is finite, then each answer to P on G w.r.t X will be finite as well. In our example, to ensure that answers to group patterns are finite on finite graphs, we can define the answer domain for RDF Entailment as $\mathsf{ad}_{RDF}(G) = \mathsf{ul}_{RDF}(G) \cup \mathsf{b}_{RDF}(G)$, which effectively restrict the answers to refer only to the elements of $\mathcal{UBL}$ that occur in G.

Second, we must deal with the fact that blank nodes are treated in SPARQL as objects with distinct identity. To understand this, consider G, P, and μ defined as follows, where $\_{:}y$ is a blank node:

$$G = \{\langle a, b, c\rangle, \langle d, e, \_{:}y\rangle\} \tag{1}$$

$$P = \{\langle a, b, x\rangle\} \tag{2}$$

$$\mu = \{x \mapsto \_{:}y\} \tag{3}$$

Even though $G \models_{RDF} \mu(P)$, the answer to P on G under RDF entailment should not contain μ. Roughly speaking, $\_{:}y$ is distinct from c even though $\_{:}y$ 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 current SPARQL 1.1. 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 $[\![P]\!]_G^X$ to a group pattern P in a graph G w.r.t. an entailment relation X.

Note that the official semantics of SPARQL treats answers as mutisets, rather than sets of mappings. The multiset variant of SPARQL can be formalized as in Table 1, but by using the multiset versions of $\bowtie$ and the set operators in the definitions of $[\![P_1 \text{ and } P_2]\!]_G^X$, $[\![P_1 \text{ union } P_2]\!]_G^X$, $[\![P_1 \text{ opt } P_2]\!]_G^X$, and $[\![P_1 \text{ filter } R]\!]_G^X$. As we show in the following sections, our temporal extension of SPARQL essentially changes just the definition of $[\![B]\!]_G^X$. Thus, we can obtain a temporal extension of SPARQL with multiset semantics by using the multiset versions of group patterns that are not basic. Analogously, by using the appropriate version of the set operators, our algorithms for computing answers to temporal queries can be modified to support the multiset semantics. Therefore, in this paper we simplify the presentation by treating answers to be sets, rather than multisets; as in [30], we do not believe that this affects the generality of our results in any significant way.

3. Representing Validity Time in RDF and OWL

Existing RDF and OWL ontologies often contain temporal information. For example, the following (nontemporal) RDF triple represents the fact that Oktoberfest 2011 starts at time instant 80 (we will discuss shortly our choice of representing time instants as integers).

$$\langle :\!\mathit{Oktoberfest\_2011}, :\!\mathit{startsAt}, 80\rangle \tag{4}$$

Despite the fact that it refers to time instants, triple (4) can be considered as being universally true at all time instants. To understand why this might be a practical problem, assume that the Oktoberfest organizing committee meets at time instant 40 and decides that Oktoberfest 2011 should start at time instant 120; however, due to popular demand, the organizing committee reconvenes at time instant 60 and decides to start the event at time instant 80. One can encode such information using an ad hoc solution; for example, one could introduce a property *:previouslyStartsAt* to record previously considered start time instants. A more principled solution, however, would be to explicitly represent the fact that triple

$$\langle :\!\mathit{Oktoberfest\_2011}, :\!\mathit{startsAt}, 120\rangle \tag{5}$$

is true between time instants 40 and 60, but that triple (4) is true from time instant 60 onwards. In other words, we would like to annotate our triples with *validity times* to specify the time instants at which the triples are true. In the rest of this section, we present a framework that can be used to achieve this goal. In particular, our framework allows for *temporal RDF triples* such as (6) and (7), which we interpret formally in first-order logic.

$$\langle :\!\mathit{Oktoberfest\_2011}, :\!\mathit{startsAt}, 120\rangle[40, 59] \tag{6}$$

$$\langle :\!\mathit{Oktoberfest\_2011}, :\!\mathit{startsAt}, 80\rangle[60, +\infty] \tag{7}$$

Please note the distinction between two kinds of temporal information in the above example. Time instants 120 and 80 do not specify the validity time of RDF triples; thus, we can process such information using an adequate datatype and a suitable set of built-in SPARQL expressions, the design of which is not the subject of this paper. In contrast, our framework focusses on techniques for the management of annotations such as $[40, 59]$ and $[60, +\infty]$ that restrict the validity of triples to specific time intervals.

A simple extension of RDF with validity time might include annotating each triple with a time instant that specifies when the triple is true. Triples, however, can be true for long periods of time, which makes listing all time instants explicitly impractical. Furthermore, triples that hold from/to eternity would need to be associated with infinitely many instants, which is practically infeasible.

We overcome these problems by applying the principles from temporal databases. In particular, Chomicki [39] distinguishes an *abstract* from a *concrete* temporal database [39]. An abstract temporal database can be understood as a sequence of "static" databases, each of which contains the facts that are true at one particular time instant. Since the number of time instants is generally assumed to

Table 1: Semantics of Group Patterns

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

be unbounded, an abstract temporal database is infinite and thus cannot be represented explicitly. In contrast, a concrete temporal database is a finite specification of abstract temporal databases. The correspondence between concrete and abstract temporal databases is zero-to-many: a concrete temporal database that contains a contradiction corresponds to no abstract temporal databases; similarly, a concrete temporal database may not fully specify the truth of facts at all time instants, so it may correspond to several abstract temporal databases. In the rest of this section we apply this approach to RDF and OWL. In particular, we devise a notion of temporal RDF and OWL graphs, which correspond to concrete temporal databases in the framework by Chomicki. Furthermore, we provide the semantics to temporal graphs by mapping them into first-order theories; the models of these theories correspond to abstract temporal databases in the framework by Chomicki.

In Section 4 we will need the ability to refer to predecessors and successors of time instants, so we use a discrete model of time. Furthermore, in this paper we do not consider implementation issues such as mapping time instants to calendar points, or selecting the appropriate granularity of the time line: any implementation of a time line that is isomorphic with the set of integers can be used.

Thus, to simplify the definitions, we take the set of all *time instants* $\mathcal{TI}$ to be the set of all integers, and we assume $\mathcal{TI}$ to be disjoint with $\mathcal{UBL}$. We use *many-sorted* first-order logic with a *temporal sort* t interpreted over $\mathcal{TI}$; we sometimes write t^t to stress that a term t is of sort t. The following example demonstrates the benefits of using many-sorted first-order logic.

Example 1. Let $\psi = \{\forall x : x \approx c\} \cup \Gamma_\approx$ be the theory that corresponds with the first-order encoding of the (nontemporal) OWL 2 DL ontology containing axiom $\top \sqsubseteq \{c\}$. Theory ψ requires all objects in a domain set to be equal. Thus, if we assume the interpretation domain set to contain $\mathcal{TI}$, then ψ becomes unsatisfiable, which is clearly undesirable. In contrast, ψ is satisfiable in our many-sorted setting: since x is not of sort t, the universal quantifier in formula $\forall x : x \approx c$ does not range over time instants. $\diamondsuit$

We integrate temporal instants into first-order logic as follows.

- We allow each time instant $t \in \mathcal{TI}$ to be used in first-order formulae as a constant of sort t; each such time instant is interpreted in each interpretation as t.
- We introduce the binary predicate $\leq$ (with both arguments of sort t), which is interpreted in each interpretation as the standard ordering on integers. We often abbreviate $t_1 \leq t \wedge t \leq t_2$ as $t_1 \leq t \leq t_2$.
- We introduce unary functions $+1$ and -1 (with the argument and the function value of sort t); usually, we write $+1(t)$ and $-1(t)$ as $t+1$ and $t-1$, respectively. These functions are interpreted in each interpretation as the successor and predecessor functions on integers.

We will often need to describe temporal intervals with unbounded start- and/or end-points. To this end, we introduce special constants $-\infty$ and $+\infty$ (not occurring in $\mathcal{UBL} \cup \mathcal{TI}$), which we allow to occur only in atoms of the following forms, for t^t a term of sort t:

$$-\infty \leq t^\mathsf{t} \qquad -\infty \leq +\infty \qquad t^\mathsf{t} \leq +\infty$$

All such atoms should be understood as syntactic abbreviations for true. Thus, $-\infty$ and $+\infty$ are not constants in the sense of first-order logic: they are not mapped to objects in an interpretation domain, but are used as syntactic abbreviations. Constants $-\infty$ and $+\infty$ allow us to simplify the notation for unbounded time intervals; for example, to state that the interval described by the formula $t_1 \leq x^\mathsf{t} \leq t_2$ has no lower bound, we can replace t_1 with $-\infty$, which makes the formula equivalent to $x^\mathsf{t} \leq t_2$. In addition to the set $\mathcal{TI}$ of all temporal instants, we use the following sets in the rest of this paper:

$$\begin{aligned}
\mathcal{TI}^\pm &= \mathcal{TI} \cup \{-\infty, +\infty\} \\
\mathcal{TI}^- &= \mathcal{TI} \cup \{-\infty\} \\
\mathcal{TI}^+ &= \mathcal{TI} \cup \{+\infty\}
\end{aligned}$$

We use $\leq$ and $<$ as the orderings on $\mathcal{TI}^\pm$ defined in the obvious way. Note that symbol $\leq$ thus ambiguously denotes both an ordering and a predicate; however, the appropriate meaning of the symbol should be clear from the context. We use analogous conventions for $+1$ and -1.

We next define the notion of temporal RDF graphs.

Definition 2. *A* temporal triple *is an expression of the form $\langle s,p,o\rangle[t]$ or $\langle s,p,o\rangle[t_1,t_2]$ such that $s,p,o \in \mathcal{UBL}$, $t \in \mathcal{TI}$, $t_1 \in \mathcal{TI}^-$, and $t_2 \in \mathcal{TI}^+$. A* temporal graph *G is a finite set of temporal triples.*

In order to store temporal triples in Web documents, a suitable encoding of temporal triples is needed. One possibility is to extend the RDF syntax and explicitly capture temporal information. Another possibility is to develop an encoding similar to the one used in [22] which encodes temporal graphs into nontemporal graphs by reifying temporal triples. In this paper, we do not discuss such implementation details, but instead focus on the conceptual aspects of the syntax, semantics, and query answering.

We next define the semantics of temporal triples by mapping them into formulae of many-sorted first-order logic. Towards this goal, we associate facts with validity time instants via the temporal arguments approach [40]. More concretely, for each n-ary predicate P, we introduce an $n+1$-ary predicate $\hat{P}$ in which the last argument is of sort t; then, $\hat{P}(u_1,\ldots,u_n,t^{\mathsf{t}})$ encodes the fact that $P(u_1,\ldots,u_n)$ is true at time instant t^{t}. As explained in Section 2, $\approx$ is an ordinary predicate with an explicit axiomatization, so $\hat{\approx}$ is a well-defined ternary predicate that should be understood as time-varying equality; the rationale behind such a definition is discussed in more detail in Example 12 at the end of this section.

For a (nontemporal) first-order formula φ and a term t^{t} of sort t, we define $\varphi\langle t^{\mathsf{t}}\rangle$ as the formula obtained from φ by replacing each atom $P(u_1,\ldots,u_n)$ with $\hat{P}(u_1,\ldots,u_n,t^{\mathsf{t}})$. Intuitively, $\varphi\langle t^{\mathsf{t}}\rangle$ states that the condition expressed by φ holds at the time instant corresponding to t^{t}.

Our definitions use a special unary predicate O. This predicate should be understood as being "internal" to our semantics—that is, it should not occur in any input. Furthermore, we assume that the axiomatization of equality $\Gamma_{\approx}$ *does not* contain the replacement rules for O. Intuitively, predicate O will be axiomatized so that it "contains" all elements of $\mathsf{ul}_X(G) \cup \mathsf{b}_X(G)$ that occur in G. As a consequence of such a definition, whenever a temporal graph G_1 X-entails a temporal graph G_2, all blank nodes in G_2 are satisfied by objects occurring in $\mathsf{usl}_X(G_1)$. We discuss the rationale behind this in Example 9.

We are now ready to define the mapping of temporal graphs into many-sorted first-order logic.

Definition 3. *Let X be an entailment relation from Section 2 other than DL, and let Γ_X be the first-order theory that characterizes X. Then, for G a temporal graph,*

- $\mathsf{ul}_X(G)$, $\mathsf{b}_X(G)$, *and* $\mathsf{ti}^{\pm}_X(G)$ *are the subsets of* $\mathcal{U}\cup\mathcal{L}$, $\mathcal{B}$, *and* $\mathcal{TI}^{\pm}$, *respectively, that occur in G;*
- $\pi_X(G)$, $\omega_X(G)$, *and* $\nu_X(G)$ *are defined as shown in Table 2;*
- $\xi_X(G)$ *is obtained from* $\nu_X(G)$ *by skolemizing the existential quantifiers in* $\exists\mathsf{b}_X(G)$*;*
- $\mathsf{usl}_X(G)$ *is* $\mathsf{ul}_X(G)$ *extended with the URI references used for the skolemization of* $\nu_X(G)$*; and*
- *the* X-skolemization *of G is the result of replacing in G each blank node from* $\mathsf{b}_X(G)$ *with the corresponding URI reference from* $\mathsf{usl}_X(G)$.

Graph G is X-satisfiable *if* $\nu_X(G)$ *is satisfiable. A temporal graph* G_1 X-entails *a temporal graph* G_2*, written* $G_1 \models_X G_2$*, if* $\nu_X(G_1) \models \nu_X(G_2)$.

Example 4. Let G_1 be the temporal graph containing temporal triples (8)–(11).

$$\langle :\!LHR, :\!flightTo, :\!MUC\rangle[50,120] \tag{8}$$
$$\langle :\!LHR, :\!flightTo, :\!MUC\rangle[100,150] \tag{9}$$
$$\langle :\!Munich, :\!hosts, :\!Oktoberfest\rangle[80,180] \tag{10}$$
$$\langle :\!hosts, rdfs\!:\!subPropertyOf, :\!hasEvent\rangle[130,300] \tag{11}$$

Triples in (8) and (9) 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 (10) states that Munich hosts Oktoberfest. Finally, triple (11) states that, if x hosts y, then x has y as an event; that this statement is not universally true might be due to the fact that events are relevant only during holiday seasons. Now let G_2 be the temporal graph containing temporal triple (12).

$$\langle :\!Munich, :\!hasEvent, :\!Oktoberfest\rangle[130,180] \tag{12}$$

One can readily verify that, according to Definition 3, G_1 *RDFS*-entails G_2. ♢

OWL 2 Direct Entailment is not characterized by a fixed set of first-order implications, so we next present separate definitions that are applicable to this entailment relation. We start by defining the notion of temporal OWL 2 DL axioms. The following definition allows us to attach validity time to arbitrary axioms and not just facts, which provides us with a flexible language that can represent, for example, class hierarchies that change over time.

Definition 5. *A* temporal OWL 2 DL axiom *is an expression of the form* $\alpha[t]$ *or* $\alpha[t_1,t_2]$*, where α is an OWL 2 DL axiom,* $t\in\mathcal{TI}$, $t_1\in\mathcal{TI}^-$*, and* $t_2\in\mathcal{TI}^+$*. A* temporal OWL 2 DL ontology *is a finite set of temporal OWL 2 DL axioms. Temporal axioms are mapped into many-sorted first-order formulae as follows, where $\theta(\alpha)$ is the translation of an OWL 2 DL axiom α into first-order logic as discussed in Section 2:*

$$\begin{aligned}\theta(\alpha[t]) &= \theta(\alpha)\langle t\rangle\\ \theta(\alpha[t_1,t_2]) &= \forall x^{\mathsf{t}} : (t_1 \le x^{\mathsf{t}} \le t_2) \rightarrow \theta(\alpha)\langle x^{\mathsf{t}}\rangle\end{aligned}$$

Our ultimate goal is to apply the OWL 2 Direct Entailment to temporal graphs. As in the case of nontemporal graphs, the first step is to translate a temporal graph into temporal OWL 2 DL axioms, and the following definition shows how to do this. Please refer to Section 2.2 for a brief overview of the transformation for nontemporal RDF and an explanation of the notion of lead triples.

Definition 6. *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 [37] modified as follows:*

Table 2: Mapping Temporal Graphs Into Logic

$$\begin{aligned}
\chi(\langle s,p,o\rangle[t]) &= \hat{T}(s,p,o,t)\\
\chi(\langle s,p,o\rangle[t_1,t_2]) &= \forall x^{\mathsf{t}} : (t_1 \le x^{\mathsf{t}} \le t_2) \to \hat{T}(s,p,o,x^{\mathsf{t}})\\
\pi_X(G) &= \bigwedge_{T\in G} \chi(T)\\
\omega_X(G) &= \bigwedge_{\_:x\in \mathsf{b}_X(G)} O(\_:x)\\
\nu_X(G) &= \{\exists \mathsf{b}_X(G) : \omega_X(G) \wedge \pi_X(G)\} \cup \{O(u) \mid u \in \mathsf{ul}_X(G)\} \cup \{\forall x^{\mathsf{t}} : \varphi\langle x^{\mathsf{t}}\rangle \mid \varphi \in \Gamma_X\}
\end{aligned}$$

1. *Each triple $\langle s,p,o\rangle$ in [37, Tables 3–8 and 10–15] is replaced with $\langle s,p,o\rangle[-\infty,+\infty]$.*
2. *Each triple pattern T in [37, Tables 16 and 17] with a lead triple $\langle s_L,p_L,o_L\rangle$ producing an axiom α is replaced with the following triple patterns T_1 and T_2.*

   Triple pattern T_1 is obtained from T as follows:
   (a) *each triple $\langle s,p,o\rangle$ other than $\langle s_L,p_L,o_L\rangle$ is replaced with temporal triple $\langle s,p,o\rangle[-\infty,+\infty]$;*
   (b) *$\langle s_L,p_L,o_L\rangle$ is replaced with $\langle s_L,p_L,o_L\rangle[t]$; and*
   (c) *T_1 produces the temporal axiom $\alpha[t]$.*

   Triple pattern T_2 is obtained from T as follows:
   (d) *each triple $\langle s,p,o\rangle$ other than $\langle s_L,p_L,o_L\rangle$ is replaced with temporal triple $\langle s,p,o\rangle[-\infty,+\infty]$;*
   (e) *$\langle s_L,p_L,o_L\rangle$ is replaced with $\langle s_L,p_L,o_L\rangle[t_1,t_2]$; and*
   (f) *T_2 produces the temporal axiom $\alpha[t_1,t_2]$.*

The following example discusses the intuition behind Definition 6.

Example 7. As explained in Section 2.2, the RDF encoding of each (nontemporal) OWL 2 DL axiom contains a single lead triple, which "represents" the axiom in an RDF graph. Thus, axiom $A \sqsubseteq \exists R.B$ is encoded using triples (13)–(16), with (13) being the lead triple.

$$\langle A, \mathit{rdfs{:}subClassOf}, \_{:}x\rangle \tag{13}$$
$$\langle \_{:}x, \mathit{rdf{:}type}, \mathit{owl{:}Restriction}\rangle \tag{14}$$
$$\langle \_{:}x, \mathit{owl{:}onProperty}, R\rangle \tag{15}$$
$$\langle \_{:}x, \mathit{owl{:}someValuesFrom}, B\rangle \tag{16}$$

By Definition 5, one can associate validity time with axioms, but not with parts of axioms (such as $\exists R.B$). Therefore, Condition 1 of Definition 6 requires triples that encode parts of axioms, such as triples (14)–(16), to be annotated with $[-\infty,+\infty]$. Condition 2 of Definition 6 essentially says that, to annotate an OWL 2 DL axiom with validity time, one should attach the validity time to the axiom's lead triple and annotate all other triples with $[-\infty,+\infty]$. Thus, each triple pattern from [37, Tables 15 and 17] that recognizes a nontemporal OWL 2 DL axiom is transformed into two triple patterns: the one defined in conditions 2(a)–2(c) recognizes temporal axioms that hold at a single time instant t, and the one defined in conditions 2(e)–2(f) recognizes temporal axioms that hold between time instants t_1 and t_2. Consequently, temporal axiom $(A \sqsubseteq \exists R.B)[5,8]$ should be encoded using temporal RDF triples as follows, which allows the triple patterns from Definition 6 to reconstruct the original axiom.

$$\langle A, \mathit{rdfs{:}subClassOf}, \_{:}x\rangle[5,8] \tag{17}$$
$$\langle \_{:}x, \mathit{rdf{:}type}, \mathit{owl{:}Restriction}\rangle[-\infty,+\infty] \tag{18}$$
$$\langle \_{:}x, \mathit{owl{:}onProperty}, R\rangle[-\infty,+\infty] \tag{19}$$
$$\langle \_{:}x, \mathit{owl{:}someValuesFrom}, B\rangle[-\infty,+\infty] \tag{20}$$

Apart from the modifications outlined in Definition 6, the process of extracting $\mathcal{O}(G)$ from G is the same as in [37]. If the transformation fails, G does not encode a temporal OWL 2 DL ontology, and so G cannot be used with temporal OWL 2 Direct Entailment. ◇

We are finally ready to define the semantics of temporal graphs under OWL 2 Direct Entailment. Remember that $\Gamma_{\approx}$ is the axiomatization of the equality predicate $\approx$.

Definition 8. *Let G be a graph that encodes a temporal OWL 2 DL ontology $\mathcal{O}(G)$. Then,*

- *$\mathsf{ul}_{DL}(G)$, $\mathsf{b}_{DL}(G)$, and $\mathsf{ti}^{\pm}_{DL}(G)$ are the subsets of $\mathcal{U}\cup\mathcal{L}$, $\mathcal{B}$, and $\mathcal{TI}^{\pm}$, respectively, that occur in $\mathcal{O}(G)$;*
- *$\pi_{DL}(G)$, $\omega_{DL}(G)$, and $\nu_{DL}(G)$ are defined as follows:*

$$\begin{aligned}
\pi_{DL}(G) &= \bigwedge_{A\in\mathcal{O}(G)} \theta(A)\\
\omega_{DL}(G) &= \bigwedge_{\_:x\in\mathsf{b}_{DL}(G)} O(\_:x)\\
\nu_{DL}(G) &= \{\exists \mathsf{b}_{DL}(G) : \omega_{DL}(G)\wedge\pi_{DL}(G)\} \cup\\
&\quad \{O(u) \mid u \in \mathsf{ul}_{DL}(G)\} \cup\\
&\quad \{\forall x^{\mathsf{t}} : \varphi\langle x^{\mathsf{t}}\rangle \mid \varphi\in\Gamma_{\approx}\}
\end{aligned}$$

- *$\xi_{DL}(G)$ is obtained from $\nu_{DL}(G)$ by skolemizing the existential quantifiers in $\exists\mathsf{b}_{DL}(G)$;*
- *$\mathsf{usl}_{DL}(G)$ is $\mathsf{ul}_{DL}(G)$ extended with the URI references used for the skolemization of $\nu_{DL}(G)$; and*

- *the DL-*skolemization *of G is the result of replacing in G each blank node from $\mathsf{b}_{DL}(G)$ with the corresponding URI reference from $\mathsf{usl}_{DL}(G)$.*

*Graph G is DL-*satisfiable *if $\nu_{DL}(G)$ is satisfiable. For G_1 and G_2 temporal graphs that encode temporal OWL 2 DL ontologies, G_1 DL-*entails *G_2, written $G_1 \models_{DL} G_2$, if $\nu_{DL}(G_1) \models \nu_{DL}(G_2)$.*

The following example explains the rationale behind the predicate O in Definitions 3 and 8.

Example 9. Let G_1, G_2, and G_3 be temporal graphs that encode the following temporal OWL 2 DL ontologies $\mathcal{O}(G_1)$, $\mathcal{O}(G_2)$, and $\mathcal{O}(G_3)$.

$$\mathcal{O}(G_1) = \{\ \exists p.\top(s)[-\infty, +\infty]\ \} \tag{21}$$
$$\mathcal{O}(G_2) = \{\ p(s, \_{:}y)[-\infty, +\infty]\ \} \tag{22}$$
$$\mathcal{O}(G_3) = \{\ p(s, \_{:}y)[1]\ \} \tag{23}$$

Let us for the moment assume that our definitions do not include the atoms with the O predicate. Then, G_1, G_2, and G_3 would be translated into the following theories; to simplify the presentation, we abbreviate with Δ the set of formulae obtained from $\Gamma_{\approx}$.

$$\nu'_{DL}(G_1) = \{\ [\forall x^{\mathsf{t}} : \exists z : \hat{p}(s, z, x^{\mathsf{t}})]\ \} \cup \Delta \tag{24}$$
$$\nu'_{DL}(G_2) = \{\ [\exists \_{:}y : \forall x^{\mathsf{t}} : \hat{p}(s, \_{:}y, x^{\mathsf{t}})]\ \} \cup \Delta \tag{25}$$
$$\nu'_{DL}(G_3) = \{\ [\exists \_{:}y : \hat{p}(s, \_{:}y, 1)]\ \} \cup \Delta \tag{26}$$

Note that $\exists z$ comes after $\forall x^{\mathsf{t}}$ in (24), so z can refer to different objects at different time instants—that is, variable z is *not rigid*. In contrast, $\exists \_{:}y$ comes before $\forall x^{\mathsf{t}}$ in (25), so the object that $\_{:}y$ refers to does not depend on x^{t}—that is, blank node $\_{:}y$ is *rigid* as it refers to the same object at all time instants. Finally, the rigidity of $\_{:}y$ in (26) is irrelevant, since (26) does not quantify over time instants. Based on these observations, we have $\nu'_{DL}(G_1) \not\models \nu'_{DL}(G_2)$ and $\nu'_{DL}(G_1) \models \nu'_{DL}(G_3)$, which can be counterintuitive. Note also that $\nu'_{DL}(G_2) \models \nu'_{DL}(G_1)$, which is intuitive: $\_{:}y$ in (25) is rigid, so it can satisfy the nonrigid existential quantifier $\exists z$ in (24).

In our approach, $\mathcal{O}(G_1)$, $\mathcal{O}(G_2)$, and $\mathcal{O}(G_3)$ are translated into the following first-order theories:

$$\nu_{DL}(G_1) = \{\ [\forall x^{\mathsf{t}} : \exists z : \hat{p}(s, z, x^{\mathsf{t}})],\ \ O(s)\ \} \cup \Delta \tag{27}$$
$$\begin{aligned}\nu_{DL}(G_2) = \\ \{\ [\exists \_{:}y : O(\_{:}y) \wedge \forall x^{\mathsf{t}} : \hat{p}(s, \_{:}y, x^{\mathsf{t}})],\ \ O(s)\ \} \cup \Delta\end{aligned} \tag{28}$$
$$\begin{aligned}\nu_{DL}(G_3) = \\ \{\ [\exists \_{:}y : O(\_{:}y) \wedge \hat{p}(s, \_{:}y, 1)],\ \ O(s)\ \} \cup \Delta\end{aligned} \tag{29}$$

As before, blank node $\_{:}y$ is rigid in (28) and (29). Atoms $O(s)$ and $O(\_{:}y)$ in (27)–(29) thus essentially "enumerate" all rigid objects. Consider now checking whether G_1 *DL*-entails G_2 or G_3. Atoms $O(\_{:}y)$ in (28) and (29) ensure that $\exists \_{:}y$ can be satisfied only by rigid objects, and not by nonrigid objects implied by $\exists z$ in (27). Consequently, we have $\nu_{DL}(G_1) \not\models \nu_{DL}(G_2)$ and $\nu_{DL}(G_1) \not\models \nu_{DL}(G_3)$, as one might expect. Furthermore, note that the O predicate does not preclude the conclusion $\nu_{DL}(G_2) \models \nu_{DL}(G_1)$.

To summarize, blank nodes can be understood in our framework as unnamed constants that denote the same object at all time instants. This is in line with all proposals for validity time on the Semantic Web known to us, including the one by Gutierrez et al. [22]. $\diamondsuit$

The rigidity of blank nodes allows us to eliminate them from entailment problems as shown next.

Theorem 10. *Let X be an entailment relation, let G be a temporal graph, let $\_{:}x$ be a blank node, and let ψ be a formula with free variable $\_{:}x$. Then,*

$$\xi_X(G) \models \exists \_{:}x : O(\_{:}x) \wedge \psi$$

holds if and only if an element $u \in \mathsf{usl}_X(G)$ exists such that $\xi_X(G) \models \sigma(\psi)$ for mapping $\sigma = \{\_{:}x \mapsto u\}$.

Proof. ($\Leftarrow$) Assume that $\xi_X(G) \models \sigma(\psi)$ holds for some $\sigma = \{\_{:}x \mapsto u\}$. Since $u \in \mathsf{usl}_X(G)$, a formula of the form $O(u) \wedge \varphi$ (with φ possibly empty) is contained in $\xi_X(G)$, so $\xi_X(G) \models \exists \_{:}x : O(\_{:}x) \wedge \psi$ clearly holds.

($\Rightarrow$) Assume that $\xi_X(G) \models \exists \_{:}x : O(\_{:}x) \wedge \psi$ holds. Let I be an arbitrary first-order model of $\xi_X(G)$, and let I' be the interpretation defined as follows:

- $\triangle^{I'} = \triangle^{I}$;
- $X^{I'} = X^{I}$ for each symbol X different from the special predicate O; and
- $O^{I'} = \{s^I \mid s \in \mathsf{usl}_X(G)\}$.

The predicate O occurs in $\xi_X(G)$ only in formulae of the form $O(u) \wedge \varphi$ (with φ possibly empty) for $u \in \mathsf{usl}_X(G_1)$. Therefore, we have $O^{I'} \subseteq O^{I}$ and $I' \models \xi_X(G)$; but then, $\xi_X(G) \models \exists \_{:}x : O(\_{:}x) \wedge \psi$ implies $I' \models \exists \_{:}x : O(\_{:}x) \wedge \psi$. Thus, an element $u \in \mathsf{usl}_X(G)$ exists such that $I' \models \sigma(\psi)$ for $\sigma = \{\_{:}x \mapsto u\}$; but then, $O^{I'} \subseteq O^{I}$ implies $I \models \sigma(\psi)$ as well, which proves our claim. $\square$

The following theorem provides us with basic building blocks for checking temporal entailment; we use this result in Section 5.1 to obtain a concrete algorithm.

Theorem 11. *Let X be an entailment relation, and let G_1 and G_2 be temporal graphs. Then, $G_1 \models_X G_2$ if and only if both of the following two conditions hold:*

- *a mapping $\sigma : \mathsf{b}_X(G_2) \to \mathsf{usl}_X(G_1)$ exists such that $\xi_X(G_1) \models \pi_X(\sigma(G_2))$, and*
- *for each $u \in \mathsf{ul}_X(G_2)$, either $u \in \mathsf{ul}_X(G_1)$ or $\mathsf{ul}_X(G_2)$ contains a literal semantically equivalent with u.*

Proof. By Definitions 3 and 8, $G_1 \models_X G_2$ if and only if $\nu_X(G_1) \models \nu_X(G_2)$. Since $\xi_X(G_1)$ is the skolemization of $\nu_X(G_1)$, by the properties of skolemization from Section 2.1, the latter condition is equivalent to $\xi_X(G_1) \models \nu_X(G_2)$. By Definitions 3 and 8, each formula ψ in $\nu_X(G_2)$ is of one of the following three types:

- $\exists \mathsf{b}_X(G_2) : \omega_X(G_2) \wedge \pi_X(G_2)$,
- $O(u)$ with $u \in \mathsf{ul}_X(G_2)$, and
- a formula of the form $\forall x^{\mathsf{t}} : \varphi\langle x^{\mathsf{t}}\rangle$ where φ is contained in Γ_X or $\Gamma_{\approx}$.

Since semantic entailment is distributive over conjunction, $\xi_X(G_1) \models \nu_X(G_2)$ if and only if $\xi_X(G_1) \models \psi$ for each formula ψ mentioned above. We have the following cases:

- Assume $\psi = \exists \mathsf{b}_X(G_2) : \omega_X(G_2) \wedge \pi_X(G_2)$. By Theorem 10, then $\xi_X(G_1) \models \psi$ if and only if a mapping $\sigma : \mathsf{b}_X(G_2) \to \mathsf{usl}_X(G_1)$ exists such that the entailment $\xi_X(G_1) \models \sigma(\pi_X(G_2))$ holds. By the definition of π_X, we have $\sigma(\pi_X(G_2)) = \pi_X(\sigma(G_2))$, so the last condition is equivalent to $\xi_X(G_1) \models \pi_X(\sigma(G_2))$.
- Assume $\psi = O(u)$. If we have $u \in \mathsf{ul}_X(G_1)$, then $\psi \in \xi_X(G_1)$, so $\xi_X(G_1) \models \psi$ clearly holds. In contrast, assume that $u \notin \mathsf{ul}_X(G_1)$; since O occurs in $\xi_X(G_1)$ only in atoms of the form $O(v)$ (note that $\xi_X(G_1)$ does not even contain the replacement axioms for O), then $\xi_X(G_1) \models \psi$ if and only if $\xi_X(G_1)$ contains an atom of the form $O(v)$ where v is a literal semantically equivalent with u.
- Assume $\psi = \forall x^{\mathsf{t}} : \varphi\langle x^{\mathsf{t}}\rangle$. But then, $\psi \in \xi_X(G_1)$, so $\xi_X(G_1) \models \psi$ always holds.

The claim of this theorem follows immediately from the above arguments. □

We finish this section with a brief discussion of why the equality predicate $\approx$ is treated in our framework as an ordinary predicate with an explicit axiomatization.

Example 12. Let G be a temporal graph such that $\mathcal{O}(G)$ contains the following temporal OWL 2 DL axioms.

$$(a \approx b)[1] \quad C(a)[1,2] \quad \neg C(b)[3] \quad D(a)[3] \tag{30}$$

Then $\pi_{DL}(G)$ is defined as follows.

$$\begin{aligned}\pi_{DL}(G) = \hat{\approx}(a,b,1) \wedge \neg\hat{C}(b,3) \wedge \hat{D}(a,3) \wedge \\ [\forall x^{\mathsf{t}} : (1 \leq x^{\mathsf{t}} \leq 2) \to \hat{C}(a,x^{\mathsf{t}})]\end{aligned} \tag{31}$$

Furthermore, $\nu_{DL}(G)$ contains $\pi_{DL}(G)$, $O(a)$, $O(b)$, and the following formulae obtained from $\Gamma_{\approx}$.

$$\forall x^{\mathsf{t}}, y : \hat{\approx}(y,y,x^{\mathsf{t}}) \tag{32}$$

$$\forall x^{\mathsf{t}}, y_1, y_2 : \hat{\approx}(y_1,y_2,x^{\mathsf{t}}) \to \hat{\approx}(y_2,y_1,x^{\mathsf{t}}) \tag{33}$$

$$\begin{aligned}&\forall x^{\mathsf{t}}, y_1, y_2, y_3 : \\ &\quad \hat{\approx}(y_1,y_2,x^{\mathsf{t}}) \wedge \hat{\approx}(y_2,y_3,x^{\mathsf{t}}) \to \hat{\approx}(y_1,y_3,x^{\mathsf{t}})\end{aligned} \tag{34}$$

$$\forall x^{\mathsf{t}}, y_1, y_2 : \hat{C}(y_1,x^{\mathsf{t}}) \wedge \hat{\approx}(y_1,y_2,x^{\mathsf{t}}) \to \hat{C}(y_2,x^{\mathsf{t}}) \tag{35}$$

$$\forall x^{\mathsf{t}}, y_1, y_2 : \hat{D}(y_1,x^{\mathsf{t}}) \wedge \hat{\approx}(y_1,y_2,x^{\mathsf{t}}) \to \hat{D}(y_2,x^{\mathsf{t}}) \tag{36}$$

First, note that, in each model $I = (\triangle^I, \cdot^I)$ of $\nu_{DL}(G)$, the domain set $\triangle^I$ is rigid—that is, it is the same at all time instants. Second, note that the interpretation of constants in each such I is also rigid: a^I and b^I are fixed domain elements that do not depend on the time instant. Third, the interpretation of $\approx$ is not rigid: constants a and b are equal in each model of $\nu_{DL}(G)$ at time instant 1, but a model of $\nu_{DL}(G)$ exists in which a and b are not equal at time instants other than 1. Thus, $\nu_{DL}(G)$ is satisfiable even though $\mathcal{O}(G)$ contains axiom $\neg C(b)[3]$; furthermore, one can see that

$$\nu_{DL}(G) \models \hat{C}(b,1) \tag{37}$$

$$\nu_{DL}(G) \models \exists x : \hat{C}(x,1) \wedge \hat{D}(x,3) \tag{38}$$

$$\nu_{DL}(G) \not\models \hat{C}(b,2) \tag{39}$$

all hold in $\nu_{DL}(G)$. ♢

4. Querying Temporal Graphs

To design a temporal query language, we must first identify the types of questions that the language should support. The language of first-order logic readily reveals the following natural types of questions, where B is a BGP, G is a temporal graph, and t, t_1, and t_2 are time instants:

Q1. Is B true in G at instant t?

Q2. Is B true in G at all instants between t_1 and t_2?

Q3. Is B true in G at some instant between t_1 and t_2?

By allowing t, t_1, and/or t_2 to be variables, we obtain non-Boolean questions that not only check the truth of B in G, but also retrieve values from G.

Such questions can be easily encoded using first-order formulae. Furthermore, the answer to a query Q in a temporal graph G w.r.t. an entailment relation X can be defined to contain each mapping μ of the free variables of Q to $\mathcal{UBL} \cup \mathcal{TI}^{\pm}$ such that $G \models_X \mu(Q)$. Such an approach, however, exhibits several important drawbacks, as the following example demonstrates.

Example 13. Let G_1 be the temporal graph shown in (40) and let Q be the query shown in (41).

$$G_1 = \{\ \langle a,b,c\rangle[5,12],\ \ \langle a,b,c\rangle[9,+\infty]\ \} \tag{40}$$

$$Q(x_1,x_2) = \forall x : x_1 \leq x \leq x_2 \to \langle a,b,c\rangle[x] \tag{41}$$

Evaluating Q on G_1 would not be a problem if x_1 and x_2 were concrete time instants. Note, however, that Q *retrieves* x_1 and x_2, and that it does not ask for *maximal* x_1 and x_2. Thus, without any restrictions, the answer to Q on G_1 is infinite since it contains each mapping μ such that $5 \leq \mu(x_1) \leq \mu(x_2) \leq +\infty$.

To overcome this problem, one might restrict all mappings in an answer to Q to refer only to time instants occurring in G. This, however, also has undesirable consequences. First, such answers can contain redundant mappings. For example, mapping $\mu_1 = \{x_1 \mapsto 5, x_2 \mapsto +\infty\}$ is the "most general mapping" in the answer to Q on

G_1, 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, let G_2 be the temporal graph shown in (42).

$$G_2 = \{ \ \langle a, b, c\rangle[5, 10], \ \ \langle a, b, c\rangle[7, +\infty] \ \} \tag{42}$$

Note that G_2 is equivalent with G_1 under Simple Entailment; however, mapping μ_2 is not contained in the answer to Q on G_2, and mapping $\mu_3 = \{x_1 \mapsto 7, x_2 \mapsto 10\}$ is not contained in the answer to Q on G_1. Third, computing redundant answers can be costly: an answer to Q 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 an answer can be exponential in n.

As a possible remedy, one might try to identify the "most general" mappings. Note, however, that a mapping refers to time instants rather than intervals; therefore, we were unable to devise a generality criterion that would be backed by a clear semantic justification. ♢

We deal with these problems in two stages. First, we introduce primitives that support questions of types Q1–Q3, and of types Q4–Q5 listed below. We thus introduce the notion of maximality into our query language.

Q4. Is $[t_1, t_2]$ the maximal interval such that B is true in G for each time instant in the interval?

Q5. Is t the smallest/largest time instant at which B is true in G?

We define our notion of answers w.r.t. $\mathcal{TI}^{\pm}$, which makes the answers independent from the syntactic form of temporal graphs. Second, we define a syntactic notion of *safety*, which guarantees that only questions of type Q4 and Q5 can bind variables in a temporal query, which then ensures finiteness of query answers.

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 [41], and they can be integrated into our query language via built-in expressions. For example, we can easily devise a built-in expression that takes two pairs of interval end-points and that evaluates to true if and only if 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 14. *A* temporal group pattern *(TGP) is an expression defined inductively as shown in Table 3, where B is a BGP, P_1 and P_2 are TGPs, R is a built-in expression, $t_1 \in \mathcal{TI}^- \cup \mathcal{V}$, $t_2 \in \mathcal{TI}^+ \cup \mathcal{V}$, and $t_3 \in \mathcal{TI} \cup \mathcal{V}$. TGPs from the first two lines of the table are called* basic.

Table 3: Definition of Temporal Group Patterns

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		

For B a variable-free BGP and X an entailment relation, $\omega_X(B)$ is the formula defined as follows, where O is the special predicate from Section 3:

$$\omega_X(B) = \bigwedge_{\_:x \in \mathsf{b}_X(B)} O(\_:x)$$

We redefine a mapping *as a partial function from $\mathcal{V}$ to $\mathcal{UBL} \cup \mathcal{TI}^{\pm}$—that is, $\mu : \mathcal{V} \to \mathcal{UBL} \cup \mathcal{TI}^{\pm}$. Let X be an entailment relation and G a temporal graph. We define $\mathsf{ad}_X(G) = \mathsf{ad}_X(G')$, where G' is the nontemporal graph obtained by replacing each temporal triple in G of the form $\langle s, p, o\rangle[u]$ or $\langle s, p, o\rangle[u_1, u_2]$ with $\langle s, p, o\rangle$. The* answer *to a basic temporal group pattern P in G w.r.t. X is the set $[\![P]\!]_G^X$ that contains each mapping μ such that*

- $\mathsf{dom}(\mu) = \mathsf{var}(P)$,
- $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup \mathcal{TI}^{\pm}$*, and*
- *$S = \mu(P)$ is a well-formed variable-free TGP that satisfies condition $\delta_X(S, G)$ in Table 4.*

Answers to TGPs that are not basic are defined as shown in Table 1.

Definition 14 should intuitively be understood as follows. Let P be a TGP, let G be a temporal graph, and let X be an entailment relation. Then, the answer $[\![P]\!]_G^X$ to P on G w.r.t. X contains each mapping μ that maps all variables in P into an element of $\mathsf{ad}_X(G) \cup \mathcal{TI}^{\pm}$; thus, each $\mu(x)$ is either an element of the answer domain of G w.r.t. X, a temporal instant, $-\infty$, or $+\infty$. Mapping μ must be such that $\mu(P)$ is a well-formed variable-free TGP. For example, for $P = \langle a, b, c\rangle$ at x, $\mu_1 = \{x \mapsto d\}$, and $\mu_2 = \{x \mapsto -\infty\}$, we have $\mu_1(P) = \langle a, b, c\rangle$ at d and $\mu_2(P) = \langle a, b, c\rangle$ at $-\infty$, which are not well-formed TGPs, so μ_1 and μ_2 cannot be contained in an answer to P. Finally, the variable-free TGP $S = \mu(P)$ must satisfy the condition $\delta_X(S, G)$ from Table 4 that defines the semantics of basic TGPs. This condition essentially make

- B at t_3 of type Q1,
- B during $[t_1, t_2]$ of type Q2,
- B occurs $[t_1, t_2]$ of type Q3,
- B maxint $[t_1, t_2]$ of type Q4, and
- B mintime t_3 and B maxtime t_3 of type Q5.

Table 4: Evaluating Basic Temporal Group Patterns

S	$\delta_X(S, G)$
B at t_3	$\xi_X(G) \models \exists \mathsf{b}_X(B) : \omega_X(B) \wedge \pi_X(B)\langle t_3 \rangle$
B during $[t_1, t_2]$	$\xi_X(G) \models \exists \mathsf{b}_X(B) : \omega_X(B) \wedge \forall x^{\mathsf{t}} : [t_1 \leq x^{\mathsf{t}} \leq t_2] \rightarrow \pi_X(B)\langle x^{\mathsf{t}} \rangle$
B occurs $[t_1, t_2]$	$\xi_X(G) \models \exists \mathsf{b}_X(B) : \omega_X(B) \wedge \exists x^{\mathsf{t}} : [t_1 \leq x^{\mathsf{t}} \leq t_2 \wedge \pi_X(B)\langle x^{\mathsf{t}} \rangle]$
B maxint $[t_1, t_2]$	a mapping $\sigma : \mathsf{b}_X(B) \rightarrow \mathsf{usl}_X(G)$ exists such that $\xi_X(G) \models \forall x^{\mathsf{t}} : [t_1 \leq x^{\mathsf{t}} \leq t_2] \rightarrow \pi_X(\sigma(B))\langle x^{\mathsf{t}} \rangle$, 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 $\mathsf{mintime}$ t_3	a mapping $\sigma : \mathsf{b}_X(B) \rightarrow \mathsf{usl}_X(G)$ exists such that $\xi_X(G) \models \pi_X(\sigma(B))\langle t_3 \rangle$ and $\xi_X(G) \not\models \exists x^{\mathsf{t}} : [x^{\mathsf{t}} \leq t_3 - 1 \wedge \pi_X(\sigma(B))\langle x^{\mathsf{t}} \rangle]$
B $\mathsf{maxtime}$ t_3	a mapping $\sigma : \mathsf{b}_X(B) \rightarrow \mathsf{usl}_X(G)$ exists such that $\xi_X(G) \models \pi_X(\sigma(B))\langle t_3 \rangle$ and $\xi_X(G) \not\models \exists x^{\mathsf{t}} : [t_3 + 1 \leq x^{\mathsf{t}} \wedge \pi_X(\sigma(B))\langle x^{\mathsf{t}} \rangle]$

Note that t_3 cannot be $-\infty$ or $+\infty$: since these constants do not represent concrete time instants, asking whether B holds at $-\infty$ or $+\infty$ does not make sense. Similarly, t_1 cannot be $+\infty$ and t_2 cannot be $-\infty$ in order to ensure that all temporal intervals in TGPs are well formed.

Conditions for S of types Q4 and Q5 involve several positive and negative entailment checks, and the mapping σ ensures that the blank nodes in S are interpreted in all checks in the same way. This affects the notion of answers as shown in the following example.

Example 15. Let G be a temporal graph and let P be a TGP defined as follows:

$$G = \{\langle a, b, c\rangle[2, 5],\ \langle a, b, d\rangle[3, 8]\} \tag{43}$$

$$P = \langle a, b, \_{:}y\rangle\ \mathsf{maxint}\ [x_1, x_2] \tag{44}$$

Blank node $\_{:}y$ in P can be satisfied in G by either c or d, so the answer to P on G w.r.t. Simple Entailment contains $\mu_1 = \{x_1 \mapsto 2, x_2 \mapsto 5\}$ and $\mu_2 = \{x_1 \mapsto 3, x_2 \mapsto 8\}$. This is consistent with the rigid interpretation of blank nodes introduced in in Section 3. ♢

One might argue that the conditions for S of type Q4 and Q5 are not elegant because the interpretation of the blank nodes in S is not captured using a first-order formula. As a possible remedy, one might try to rewrite the condition for $S = B$ maxint $[t_1, t_2]$ as

- $\xi_X(G) \models \exists \mathsf{b}_X(B)\ \forall x^{\mathsf{t}} : [t_1 \leq x^{\mathsf{t}} \leq t_2] \rightarrow \pi_X(B)\langle x^{\mathsf{t}} \rangle$, and
- $t_1 = -\infty$ or $\xi_X(G) \not\models \exists \mathsf{b}_X(B)\ \pi_X(B)\langle t_1 - 1 \rangle$, and
- $t_2 = +\infty$ or $\xi_X(G) \not\models \exists \mathsf{b}_X(B)\ \pi_X(B)\langle t_2 + 1 \rangle$.

Such a definition, however, would have counterintuitive consequences, as shown by the following example.

Example 16. Let G, P, μ_1, and μ_2 be defined as in Example 15. With the alternative condition for P outlined above, the answer to P on G w.r.t. simple entailment does not contain μ_1, since it is not the case that $\xi_{simple}(G) \not\models \exists \_{:}y : O(\_{:}y) \wedge \hat{T}(a, b, \_{:}y, 6)$; one can analogously see that the answer does not contain μ_2 either. Intuitively, this is because the alternative condition for P does not ensure that the blank nodes in P are interpreted in the same way in all entailment checks. ♢

Conditions for S of types Q4 and Q5 can be formulated in an autoepistemic extension of first-order logic with a modal operator $\mathbf{K}$, such as the one used in the definition of the EQL-Lite query language [42]. A detailed discussion of such an alternative formulation is out of scope of this paper; we merely observe that the condition for $S = B$ maxint $[t_1, t_2]$ (with $t_1 \neq -\infty$ and $t_2 \neq +\infty$) could be expressed as

$$\begin{aligned} \xi_X(G) \models \exists \mathsf{b}_X(B) : \omega_X(B)\ \wedge & \\ \mathbf{K}\,[\forall x^{\mathsf{t}} : (t_1 \leq x^{\mathsf{t}} \leq t_2) \rightarrow \pi_X(B)\langle x^{\mathsf{t}} \rangle]\ \wedge & \\ \neg\,\mathbf{K}\,\pi_X(B)\langle t_1 - 1 \rangle\ \wedge & \\ \neg\,\mathbf{K}\,\pi_X(B)\langle t_2 + 1 \rangle & \end{aligned}$$

and that conditions for S of type Q5 could be expressed in an analogous way.

We next present several temporal graph patterns and their answers.

Example 17. Let G be the temporal graph containing triples (8)–(11) from Section 3. Table 5 shows several TGPs that could be used in our running example.

TGP (45) returns all airports x and all maximal intervals $[y, z]$ during Oktoberfest in which there is a flight from x to Munich airport. The answer to (45) on G contains only the mapping $\{x \mapsto {:}LHR, y \mapsto 80, z \mapsto 150\}$.

TGP (46) retrieves the duration $[x, y]$ of Oktoberfest and all events z in London that have at least one time point in common with $[x, y]$. If occurs were changed to during, the TGP would retrieve all events z in London whose duration is contained in $[x, y]$.

TGP (47) retrieves the first time instant at which Munich hosts Oktoberfest. The answer to (47) on G contains only the mapping $\{x \mapsto 80\}$.

TGP (48) returns all prices x for a particular room during an event y in Munich within interval $[50, 100]$. $\diamondsuit$

According to Definition 14, $\mathsf{ad}_X(G)$ does not contain $\mathsf{ti}^{\pm}_X(G)$, so the time instants used in temporal triples do not affect the answer domain of a temporal graph. This allows us to prove the following property.

Proposition 18. *Let X be an entailment relation, let G_1 and G_2 be temporal graphs, and let G'_1 and G'_2 be the X-skolemizations of G_1 and G_2, respectively. If $G'_1 \models_X G'_2$, $G'_2 \models_X G'_1$, and $\mathsf{ad}_X(G_1) = \mathsf{ad}_X(G_2)$, then for each temporal group pattern P we have $[\![P]\!]^X_{G_1} = [\![P]\!]^X_{G_2}$.*

Proof. Due to $G'_1 \models_X G'_2$ and $G'_2 \models_X G'_1$, we have that formulae $\nu_X(G'_1)$ and $\nu_X(G'_2)$ are equivalent; but then, so are $\xi_X(G_1)$ and $\nu_X(G_2)$. Based on this observation, we prove our claim by induction on the structure of P. The inductive steps are straightforward, so we consider only the base cases where P is a basic TGP.

Let $P = B$ at t_3; let μ be an arbitrary mapping such that $\mu(t_3) \in \mathcal{TI}$. Then,

$$\xi_X(G_1) \models \exists \mathsf{b}_X(\mu(B)) : \omega_X(\mu(B)) \wedge \pi_X(\mu(B))\langle \mu(t_3)\rangle$$

if and only if

$$\xi_X(G_2) \models \exists \mathsf{b}_X(\mu(B)) : \omega_X(\mu(B)) \wedge \pi_X(\mu(B))\langle \mu(t_3)\rangle.$$

Thus, $\mathsf{ad}_X(G_1) = \mathsf{ad}_X(G_2)$ implies $[\![P]\!]^X_{G_1} = [\![P]\!]^X_{G_2}$. The proof is analogous for all other types of basic TGPs. $\square$

Example 19. Let G_1 and G_2 be as specified in Example 13. Although G_1 and G_2 contain different time instants, we have $\mathsf{ad}_X(G_1) = \mathsf{ad}_X(G_2)$. Thus, since the X-skolemizations of G_1 and G_2 X-entail each other, for each TGP P we have $[\![P]\!]^X_{G_1} = [\![P]\!]^X_{G_2}$. $\diamondsuit$

Note that Proposition 18 requires the X-skolemizations of G_1 and G_2 to X-entail each other, rather than G_1 and G_2 themselves. This is not a side-effect of our temporal framework: as the following example shows, this is already the case in nontemporal SPARQL.

Example 20. Let G_1 and G_2 be the nontemporal graphs defined in (49) and (50), respectively; furthermore, let P be the BGP (51); finally, let X be Simple Entailment.

$$G_1 = \{\langle a, b, \_{:}x\rangle, \langle c, d, \_{:}y\rangle\} \tag{49}$$
$$G_2 = \{\langle a, b, \_{:}y\rangle, \langle c, d, \_{:}x\rangle\} \tag{50}$$
$$P = \{\langle a, b, z\rangle\} \tag{51}$$

The X-skolemizations of G_1 and G_2 are as follows, where u_1 and u_2 are used to skolemize $\_{:}x$ and $\_{:}y$, respectively.

$$G'_1 = \{\langle a, b, u_1\rangle, \langle c, d, u_2\rangle\} \tag{52}$$
$$G'_2 = \{\langle a, b, u_2\rangle, \langle c, d, u_1\rangle\} \tag{53}$$

Clearly, $\mathsf{ad}_X(G_1) = \mathsf{ad}_X(G_2)$. Furthermore, we clearly have $G_1 \models_X G_2$ and $G_2 \models_X G_1$; however, we also have $G'_1 \not\models_X G'_2$ and $G'_2 \not\models_X G'_1$. In other words, G_1 and G_2 X-entail each other, but their X-skolemizations do not. Since P is evaluated in G_1 and G_2 w.r.t. their X-skolemizations, the answers to P on G_1 and G_2 differ:

$$[\![P]\!]^X_{G_1} = \{\ \{z \mapsto u_1\}\ \} \tag{54}$$
$$[\![P]\!]^X_{G_2} = \{\ \{z \mapsto u_2\}\ \} \tag{55}$$

This example can be straightforwardly extended to temporal graphs and queries, which requires us to restrict the applicability of Proposition 18. Apart from these issues inherited from nontemporal RDF and SPARQL, however, Proposition 18 shows that the addition of validity time does not introduce further problems. $\diamondsuit$

Since the answers are defined w.r.t. the entire set $\mathcal{TI}^{\pm}$, basic TGPs can have infinite answers, as discussed at the beginning of this section. We next define a notion of safe TGPs. In Section 5.2 we then present an algorithm that computes a finite answer for each safe TGP.

Definition 21. *Let P be a temporal group pattern. Then, $\mathsf{tvar}(P)$ and $\mathsf{bind}(P)$ are sets of variables, and $\mathsf{safe}(P)$ is a Boolean value defined as shown in Table 6. Pattern P is* safe *if and only if $\mathsf{safe}(P) = \mathsf{true}$.*

Intuitively, $x \in \mathsf{bind}(P)$ ensures that $\mu(x) \in \mathsf{ti}^{\pm}_X(G)$ for each mapping $\mu \in [\![P]\!]^X_G$. Thus, B at t_3, B during $[t_1, t_2]$, and B occurs $[t_1, t_2]$ are safe if and only if t_1, t_2, and t_3 are not variables: B can be true at potentially infinitely many time instants, 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 is true. For $P = P_1$ union P_2, a mapping in an answer to P is produced either by P_1 or P_2, so both P_1 and P_2 must be safe for P to be safe. For $P = P_1$ opt P_2, the definition of safety assumes that P_1 is evaluated "before" P_2: to evaluate P, one first evaluates P_1, and then for each mapping μ in the answer, one evaluates $\mu(P_2)$, so P is safe if and only if P_1 and $\mu(P_2)$ are safe. For $P = P_1$ filter R, all time instants in an answer to P are produced by P_1, so P is safe if and only if P_1 is safe. Finally, for $P = P_1$ and P_2, each time instant in an answer to P is produced by P_1 and P_2, so P is safe if and only if the value of each variable in the answer is produced by either P_1 or P_2. Note that this allows P to be safe even if neither P_1 nor P_2 is safe.

5. Algorithms for Temporal Graphs

In this section we turn our attention to the computational problems involving temporal graphs. In particular, in Section 5.1 we show how to solve certain basic types of entailments involving temporal graphs; then, in Section 5.2 we use these results to obtain an algorithm for computing answers to safe TGPs.

Table 5: Example Temporal Group Patterns

$$\{\langle x, {:}\mathit{flightTo}, {:}\mathit{MUC}\rangle,\ \langle {:}\mathit{Munich}, {:}\mathit{hosts}, {:}\mathit{Oktoberfest}\rangle\}\ \mathsf{maxint}\ [y, z] \tag{45}$$

$$\{\langle {:}\mathit{Munich}, {:}\mathit{hosts}, {:}\mathit{Oktoberfest}\rangle\}\ \mathsf{maxint}\ [x, y]\ \ \mathsf{and}\ \ \{\langle {:}\mathit{London}, {:}\mathit{hosts}, z\rangle\}\ \mathsf{occurs}\ [x, y] \tag{46}$$

$$\{\langle {:}\mathit{Munich}, {:}\mathit{hosts}, {:}\mathit{Oktoberfest}\rangle\}\ \mathsf{mintime}\ x \tag{47}$$

$$\{\langle {:}\mathit{Room123}, {:}\mathit{hasPrice}, x\rangle,\ \langle {:}\mathit{Munich}, {:}\mathit{hosts}, y\rangle\}\ \mathsf{occurs}\ [50, 100] \tag{48}$$

Table 6: The Definition of Functions tvar, bind, and safe

P	$\mathsf{tvar}(P)$	$\mathsf{bind}(P)$	$\mathsf{safe}(P)$
B at t_3	$\{t_3\} \cap \mathcal{V}$	$\emptyset$	$\mathsf{tvar}(P) = \emptyset$
B during $[t_1, t_2]$	$\{t_1, t_2\} \cap \mathcal{V}$	$\emptyset$	$\mathsf{tvar}(P) = \emptyset$
B occurs $[t_1, t_2]$	$\{t_1, t_2\} \cap \mathcal{V}$	$\emptyset$	$\mathsf{tvar}(P) = \emptyset$
B maxint $[t_1, t_2]$	$\{t_1, t_2\} \cap \mathcal{V}$	$\mathsf{tvar}(P)$	true
B mintime t_3	$\{t_3\} \cap \mathcal{V}$	$\mathsf{tvar}(P)$	true
B maxtime t_3	$\{t_3\} \cap \mathcal{V}$	$\mathsf{tvar}(P)$	true
P_1 and P_2	$\mathsf{tvar}(P_1) \cup \mathsf{tvar}(P_2)$	$\mathsf{bind}(P_1) \cup \mathsf{bind}(P_2)$	$\mathsf{tvar}(P) = \mathsf{bind}(P)$
P_1 union P_2	$\mathsf{tvar}(P_1) \cup \mathsf{tvar}(P_2)$	$\mathsf{bind}(P_1) \cap \mathsf{bind}(P_2)$	$\mathsf{safe}(P_1) \wedge \mathsf{safe}(P_2)$
P_1 opt P_2	$\mathsf{tvar}(P_1) \cup \mathsf{tvar}(P_2)$	$\mathsf{bind}(P_1)$	$\mathsf{safe}(P_1) \wedge \mathsf{safe}(\mu_{\mathsf{bind}(P_1)}(P_2))$
P_1 filter R	$\mathsf{tvar}(P_1)$	$\mathsf{bind}(P_1)$	$\mathsf{safe}(P_1) \wedge (\mathsf{var}(R) \subseteq \mathsf{var}(P_1))$

Note: mapping $\mu_{\mathsf{bind}(P_1)}$ is defined by setting $\mu_{\mathsf{bind}(P_1)}(x) = 0$ for each $x \in \mathsf{bind}(P_1)$.

5.1. Checking Temporal Entailments

In this section we show how to reduce several basic types of temporal entailment to nontemporal entailment. We first show how, given an entailment relation X and a temporal graph G, one can extract a nontemporal graph $\Xi_X(G, t)$ that describes the consequences of G at a time instant $t \in \mathcal{TI}$. As in all definitions presented thus far, we must treat the case of OWL 2 Direct Entailment separately; hence, we proceed as follows.

- If $X \neq DL$, then $\Xi_X(G, t)$ contains each nontemporal triple α for which G contains a temporal triple of the form $\alpha[t]$ or $\alpha[t_1, t_2]$ with $t_1 \leq t \leq t_2$.
- For $X = DL$, let $\mathcal{O}(G, t)$ be the nontemporal OWL 2 DL ontology that contains each OWL 2 DL axiom α for which $\mathcal{O}(G)$ contains a temporal axiom of the form $\alpha[t]$ or $\alpha[t_1, t_2]$ with $t_1 \leq t \leq t_2$; then, $\Xi_X(G, t)$ is the nontemporal graph that encodes $\mathcal{O}(G, t)$.

Let N be an arbitrary subset of $\mathcal{TI}^{\pm}$. A pair (t_1, t_2) with $t_1, t_2 \in \mathcal{TI}^{\pm}$ is *consecutive* w.r.t. N if $t_1 \leq t_2$ and no $t \in N$ exists with $t_1 < t < t_2$. The following property follows straightforwardly from the definition of $\Xi_X(G, t)$.

Proposition 22. *Let X be an entailment relation, let G be a temporal graph, and let t be an arbitrary time instant such that $t \in \mathcal{TI} \setminus \mathsf{ti}_X^{\pm}(G)$. For each time instant $t' \in \mathcal{TI}$ such that (t, t') or (t', t) is consecutive w.r.t. $\mathsf{ti}_X^{\pm}(G)$, we have $\Xi_X(G, t) \subseteq \Xi_X(G, t')$; furthermore, if in addition we have $t' \notin \mathsf{ti}_X^{\pm}(G)$, then $\Xi_X(G, t) = \Xi_X(G, t')$ holds as well.*

We next identify "interesting" time instants—that is, time instants at which the consequences of G can change. Let $t_1 \in \mathcal{TI}^-$ and $t_2 \in \mathcal{TI}^+$; then, $\mathsf{ex}_X(G, t_1, t_2)$ is the smallest set satisfying the following conditions:

- for each $t \in \mathsf{ti}_X^{\pm}(G) \cap \mathcal{TI}$ such that $t_1 \leq t \leq t_2$, we have $t \in \mathsf{ex}_X(G, t_1, t_2)$;
- if $t_1 \neq -\infty$, then $t_1 \in \mathsf{ex}_X(G, t_1, t_2)$;
- if $t_2 \neq +\infty$, then $t_2 \in \mathsf{ex}_X(G, t_1, t_2)$; and
- if $t_1 = -\infty$ and $t_2 = +\infty$, then $0 \in \mathsf{ex}_X(G, t_1, t_2)$.

Furthermore, $\mathsf{all}_X(G, t_1, t_2)$ is the smallest set such that, for each $t \in \mathsf{ex}_X(G, t_1, t_2)$, we have

- $t \in \mathsf{all}_X(G, t_1, t_2)$,
- $t_1 < t$ implies $t - 1 \in \mathsf{all}_X(G, t_1, t_2)$, and
- $t < t_2$ implies $t + 1 \in \mathsf{all}_X(G, t_1, t_2)$.

With these definitions in place, we can obtain the desired reductions as shown in Table 7. The following theorem shows that the reductions are correct.

Theorem 23. *All claims in Table 7 are true.*

Table 7: Reducing Temporal to Nontemporal Entailment

1. G is X-satisfiable if and only if $\Xi_X(G,t)$ is X-satisfiable for each $t \in \mathsf{ex}_X(G,-\infty,+\infty)$.
2. $\xi_X(G) \models \pi_X(B)\langle t_3\rangle$ if and only if G is not X-satisfiable or $\Xi_X(G,t_3) \models_X B$.
3. $\xi_X(G) \models \forall x^{\mathsf{t}} : [t_1 \le x^{\mathsf{t}} \le t_2] \to \pi_X(B)\langle x^{\mathsf{t}}\rangle$ if and only if G is not X-satisfiable or $\Xi_X(G,t) \models_X B$ for each $t \in \mathsf{all}_X(G,t_1,t_2)$.
4. $\xi_X(G) \models \exists x^{\mathsf{t}} : [t_1 \le x^{\mathsf{t}} \le t_2 \wedge \pi_X(B)\langle x^{\mathsf{t}}\rangle]$ if and only if G is not X-satisfiable or $\Xi_X(G,t) \models_X B$ for some $t \in \mathsf{ex}_X(G,t_1,t_2)$.

Note: X is an arbitrary entailment relation; G is an arbitrary temporal graph; B is an arbitrary BGP with $\mathsf{var}(B) = \mathsf{b}_X(B) = \emptyset$; $t_1 \in \mathcal{TI}^-$; $t_2 \in \mathcal{TI}^+$; and $t_3 \in \mathcal{TI}$.

Proof. Note that, by Definitions 3 and 8, $\xi_X(G)$ contains precisely one formula of the form

$$\bigwedge O(u) \wedge \bigwedge_{i=1}^{m} \psi_i\langle t_i\rangle \wedge \bigwedge_{j=1}^{n} \forall x^{\mathsf{t}} : (t_j^1 \le x^{\mathsf{t}} \le t_j^2) \to \kappa_j\langle x^{\mathsf{t}}\rangle,$$

zero of more formulae of the form $O(u)$, and zero or more formulae of the form $\forall x^{\mathsf{t}} : \varphi_k\langle x^{\mathsf{t}}\rangle$. For an arbitrary time instant $t \in \mathcal{TI}$, let Υ_t be the set of formulae that contains

- $\psi_i\langle t_i\rangle$ for each $1 \le i \le m$ such that $t_i = t$,
- $\kappa_j\langle t\rangle$ for each $1 \le j \le n$ such that $t_j^1 \le t \le t_j^2$, and
- $\varphi_k\langle t\rangle$ for each k.

Furthermore, let Υ be the set of formulae that contains each $O(u)$ occurring in $\xi_X(G)$ and Υ_t for each $t \in \mathcal{TI}$. Clearly, $\xi_X(G) \models \Upsilon$ and $\Upsilon \models \xi_X(G)$. Furthermore, let I_t be a model of $\xi_X(\Xi_X(G,t))$; then, the interpretation I_t', defined below, is clearly a model of Υ_t:

$$\begin{aligned}
\triangle^{I_t'} &= \triangle^{I_t} \\
c^{I_t'} &= c^{I_t} \text{ for each constant } c \\
\hat{P}^{I_t'} &= \{\langle \iota_1,\ldots,\iota_n,t\rangle \mid \langle \iota_1,\ldots,\iota_n\rangle \in P^{I_t}\} \\
&\qquad\qquad \text{for each } n\text{-ary predicate } P
\end{aligned}$$

Finally, each model of Υ_t can be converted into a model of $\xi_X(\Xi_X(G,t))$ by an analogous transformation.

(Claim 1) If Υ is satisfiable, then Υ_t is clearly satisfiable for each $t \in \mathcal{TI}$, so $\Xi_X(G,t)$ is X-satisfiable for each $t \in \mathsf{ex}_X(G,-\infty,+\infty)$ as well. Conversely, assume that $\Xi_X(G,t)$ is X-satisfied in a model I_t for each time instant $t \in \mathsf{ex}_X(G,-\infty,+\infty)$, and consider an arbitrary time instant $t \in \mathcal{TI} \setminus \mathsf{ti}_X^{\pm}(G)$. Since $\mathsf{ex}_X(G,-\infty,+\infty)$ is not empty, a time instant $t_1 \in \mathsf{ex}_X(G,-\infty,+\infty)$ exists such that (t,t_1) or (t_1,t) is consecutive w.r.t. $\mathsf{ti}_X^{\pm}(G)$; then $\Xi_X(G,t) \subseteq \Xi_X(G,t_1)$ by Proposition 22, so by the monotonicity of first-order logic $\Xi_X(G,t)$ is X-satisfied in I_{t_1}. Thus, $\Xi_X(G,t)$ is X-satisfied in a model I_t for each $t \in \mathcal{TI}$. Without loss of generality, we can assume that all I_t have the same domain and that they interpret all constants in the same way; for example, we can take I_t to be Herbrand models with parameters [31]. Let I_t' be the models of Υ_t obtained from I_t using the transformation described above, and let I be the interpretation defined as follows:

$$\begin{aligned}
\triangle^{I} &= \triangle^{I_0'} \\
c^{I} &= c^{I_0'} \text{ for each constant } c \text{ occurring in } \Upsilon \\
O^{I} &= \{c^{I_0'} \mid c \text{ is a constant occurring in } \Upsilon\} \\
X^{I} &= \bigcup_{t \in \mathcal{TI}} X^{I_t'} \text{ for each predicate } X \ne O
\end{aligned}$$

Since each Υ_t refers only to the time instant t, it is clear that $I \models \Upsilon$, so $I \models \xi_X(G)$ as well.

(Claim 2) If $\xi_X(G)$ is unsatisfiable, then G is not X-satisfiable, so the claim holds. Assume now that $\xi_X(G)$ is satisfiable. Formula $\pi_X(B)\langle t_3\rangle$ contains only atoms of the form $\hat{P}(u_1,\ldots,u_n,t_3)$; thus, $\Upsilon \models \pi_X(B)\langle t_3\rangle$ if and only if $\Upsilon_{t_3} \models \pi_X(B)\langle t_3\rangle$. By the correspondences between the models of Υ_{t_3} and $\Xi_X(G,t_3)$ outlined above, we have that $\Upsilon_{t_3} \models \pi_X(B)\langle t_3\rangle$ if and only if $\Xi_X(G,t_3) \models_X B$, which proves our claim.

(Claim 3) The claim is equivalent to the claim that $\Upsilon \models \pi_X(B)\langle t\rangle$ for each t with $t_1 \le t \le t_2$ if and only if Υ is unsatisfiable or $\Upsilon_t \models \pi_X(B)\langle t\rangle$ for each $t \in \mathsf{all}_X(G,t_1,t_2)$. This is obvious if Υ is unsatisfiable, so assume that Υ is satisfiable. As in Claim 2, we can equivalently show that $\Xi_X(G,t) \models_X B$ for each t with $t_1 \le t \le t_2$ if and only if $\Xi_X(G,t) \models_X B$ for each $t \in \mathsf{all}_X(G,t_1,t_2)$. The ($\Rightarrow$) direction is obvious, so we focus on the ($\Leftarrow$) direction; consequently, assume that $\Xi_X(G,t) \models_X B$ for each time instant $t \in \mathsf{all}_X(G,t_1,t_2)$ and consider an arbitrary time instant $t' \in \mathcal{TI} \setminus \mathsf{all}_X(G,t_1,t_2)$ such that $t_1 \le t' \le t_2$. Now $\mathsf{all}_X(G,t_1,t_2)$ is not empty, so a time instant t'' with $t'' \in \mathsf{all}_X(G,t_1,t_2)$ and $t'' \notin \mathsf{ti}_X^{\pm}(G)$ exists such that (t',t'') or (t'',t') is consecutive w.r.t. $\mathsf{ti}_X^{\pm}(G)$; but then, by Proposition 22, $\Xi_X(G,t') = \Xi_X(G,t'')$, so $\Xi_X(G,t'') \models_X B$ implies $\Xi_X(G,t') \models_X B$. This holds for an arbitrary t' that satisfies the mentioned conditions, so our claim holds.

(Claim 4) The claim is equivalent to the claim that $\Upsilon \models \pi_X(B)\langle t\rangle$ for some t with $t_1 \le t \le t_2$ if and only if Υ is unsatisfiable or $\Upsilon_t \models \pi_X(B)\langle t\rangle$ for some $t \in \mathsf{ex}_X(G, t_1, t_2)$. This is obvious if Υ is unsatisfiable, so assume that Υ is satisfiable. As in Claim 2, we equivalently show that $\Xi_X(G, t) \models_X B$ for some t with $t_1 \leq t \leq t_2$ if and only if $\Xi_X(G, t) \models_X B$ for some $t \in \mathsf{ex}_X(G, t_1, t_2)$. The ($\Leftarrow$) direction is obvious, so we focus on the ($\Rightarrow$) direction; consequently, assume that $\Xi_X(G, t) \models_X B$ for some t with $t_1 \leq t \leq t_2$ and $t \notin \mathsf{ex}_X(G, t_1, t_2)$. Since $\mathsf{ex}_X(G, t_1, t_2)$ is not empty, a time instant $t' \in \mathsf{ex}_X(G, t_1, t_2)$ exists such that (t, t') or (t', t) is consecutive w.r.t. $\mathsf{ti}^{\pm}_X(G)$; but then, $\Xi_X(G, t) \subseteq \Xi_X(G, t')$ by Proposition 22; since first-order logic is monotonic, we have that $\Xi_X(G, t) \models_X B$ implies $\Xi_X(G, t') \models_X B$, which implies our claim. $\square$

The following example illustrates Theorem 23.

Example 24. Let G and B be as shown in (56) and (57), respectively, and let X be Simple Entailment. The set $\mathsf{ti}^{\pm}_X(G)$ is then shown in (58).

$$G = \{\langle a, b, c\rangle[2, 4],\ \langle a, b, c\rangle[8]\} \tag{56}$$

$$B = \langle a, b, c\rangle \tag{57}$$

$$\mathsf{ti}^{\pm}_X(G) = \{2, 4, 8\} \tag{58}$$

To check entailment

$$\xi_X(G) \models \forall x^{\mathsf{t}} : [3 \leq x^{\mathsf{t}} \leq 8] \rightarrow \pi_X(B)[x^{\mathsf{t}}] \tag{59}$$

we proceed as follows. First, we determine $\mathsf{ex}_X(G, 3, 8)$ as shown in (60) and $\mathsf{all}_X(G, 3, 8)$ as shown in (61), and we determine the relevant $\Xi_X(G, t)$ as shown in (62)–(64).

$$\mathsf{ex}_X(G, 3, 8) = \{2, 3, 4, 8\} \tag{60}$$

$$\mathsf{all}_X(G, 3, 8) = \{2, 3, 4, 5, 7, 8\} \tag{61}$$

$$\Xi_X(G, 5) = \Xi_X(G, 7) = \emptyset \tag{62}$$

$$\Xi_X(G, 2) = \Xi_X(G, 3) = \{\langle a, b, c\rangle\} \tag{63}$$

$$\Xi_X(G, 4) = \Xi_X(G, 8) = \{\langle a, b, c\rangle\} \tag{64}$$

Next, we check the satisfiability of G by checking whether $\Xi_X(G, t)$ is satisfiable for each $t \in \mathsf{ex}_X(G, 3, 8)$. This is the case, so we finally check whether $\Xi_X(G, t) \models_X B$ for each $t \in \mathsf{all}_X(G, 3, 8)$. The latter does not hold for $t = 5$ and $t = 7$, so (59) does not hold either. $\diamondsuit$

Note that Claims 1 and 4 in Table 7 involve only the time instants in $\mathsf{ex}_X(G, t_1, t_2)$, whereas Claim 3 involves the time instants in $\mathsf{all}_X(G, t_1, t_2)$. This can intuitively be explained by observing that, if $\Xi_X(G, t) \models_X B$ holds for some time instant t with $t_1 \leq t \leq t_2$, then Proposition 22 and the monotonicity of first-order logic imply that $\Xi_X(G, t') \models_X B$ holds as well for t' the "nearest" time instant to t such that $t' \in \mathsf{ex}_X(G, t_1, t_2)$; thus, we can focus our attention only to time instants between t_1 and t_2 that occur in G. In contrast, if we want $\Xi_X(G, t) \models_X B$ to hold for each time instant t with $t_1 \leq t \leq t_2$, then we need to ensure that $\Xi_X(G, t') \models_X B$ holds not only for t' explicitly occurring in G, but also for t' occurring "between" time instants in G.

Example 25. Let G be as in Example 24. Note that $\Xi_X(G, t) \models_X B$ for each $t \in \mathsf{ex}_X(G, 3, 8)$, but not for each $t \in \mathsf{all}_X(G, 3, 8)$. In other words, considering only the time instants in $\mathsf{ex}_X(G, 3, 8)$ would lead us to incorrectly conclude that (59) holds. $\diamondsuit$

Based on these results, whether $G_1 \models_X G_2$ holds can be checked using the following nondeterministic algorithm.

1. Guess a mapping $\sigma : \mathsf{b}_X(G_2) \rightarrow \mathsf{usl}_X(G_1)$.
2. Compute $\pi_X(\sigma(G_2))$; the result will be of the form

$$\bigwedge_{i=1}^{m} \psi_i\langle t_i\rangle \wedge \bigwedge_{j=1}^{n} \forall x^{\mathsf{t}} : (t_j^1 \leq x^{\mathsf{t}} \leq t_j^2) \rightarrow \kappa_j\langle x^{\mathsf{t}}\rangle.$$

3. Check $\xi_X(G_1) \models \psi_i\langle t_i\rangle$ for each $1 \leq i \leq m$.
4. Check $\xi_X(G_1) \models \forall x^{\mathsf{t}} : (t_j^1 \leq x^{\mathsf{t}} \leq t_j^2) \rightarrow \kappa_j\langle x^{\mathsf{t}}\rangle$ for each $1 \leq j \leq m$.
5. For each $u \in \mathsf{ul}_X(G_2)$, check whether $u \in \mathsf{ul}_X(G_1)$ or $\mathsf{ul}_X(G_1)$ contains a literal semantically equivalent with u.
6. Return true if all checks in Step 4 and 5 succeed.

The correctness of this algorithm follows straightforwardly from Theorem 11 and the fact that $\varphi \models \psi_1 \wedge \ldots \wedge \psi_k$ if and only if $\varphi \models \psi_i$ for each $1 \leq i \leq k$. Furthermore, all temporal entailment checks in Steps 3 and 4 can be solved using polynomially many nontemporal entailment checks as shown in Table 7. Since none of these entailments involve blank nodes, it is also straightforward to see that this algorithm is worst-case optimal for all entailment relations listed in Section 2.

5.2. Answering Temporal Queries

In Table 8 we define a recursive function that computes the answer for a safe temporal group pattern P. Roughly speaking, if P is a basic TGP, then $\mathsf{eval}_X(P, G)$ can be computed as follows. We enumerate all mappings potentially relevant to P. For each candidate mapping μ, we evaluate the condition from Table 4 by eliminating blank nodes as shown in Theorem 10, and then checking the remaining condition using Theorem 23; if all of these checks succeed, we include μ in the answer to P. Clearly, such a straightforward approach is unlikely to be suitable for practical use; however, it does show that the problem is solvable in principle. Furthermore, this general approach can be optimized; for example, in Section 6 we present optimizations that are applicable to deterministic entailment relations and that make our algorithm practicable.

The following theorem shows that, if P is safe, then $\mathsf{eval}_X(P, G)$ indeed computes $[\![P]\!]_G^X$, and the latter set is finite whenever $\mathsf{ad}_X(G)$ is finite.

Theorem 26. *Let G be a temporal graph, let X be an entailment relation, and let P be a safe temporal group pattern. Then $\mathsf{eval}_X(P, G) = [\![P]\!]_G^X$; furthermore, if $\mathsf{ad}_X(G)$ is finite, then $[\![P]\!]_G^X$ is finite as well.*

Table 8: Evaluation of Safe Temporal Group Patterns

$\mathsf{eval}_X(P, G)$ is the set of mappings that is inductively 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 $[t_1, t_2]$ or
$P = B$ maxint $[t_1, t_2]$ or
$P = B$ $\mathsf{mintime}$ t_3 or
$P = B$ $\mathsf{mintime}$ t_3 :
$\{\mu \mid \mathsf{dom}(\mu) = \mathsf{var}(P), \mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup \mathsf{ti}^{\pm}_X(G) \cup \{-\infty, +\infty\},$ and $\delta_X(\mu(P), G)$ holds$\}$

$P = P_1$ and P_2 :
$\{\mu_1 \cup \mu_2 \mid \mu_1 \in \mathsf{eval}_X(P_1, G), \mu_2 \in \mathsf{eval}_X(P_2, G),$ and μ_1 and μ_2 are compatible$\}$

$P = P_1$ union P_2 :
$\mathsf{eval}_X(P_1, G) \cup \mathsf{eval}_X(P_2, G)$

$P = P_1$ opt P_2 :
$\{\mu_1 \cup \mu_2 \mid \mu_1 \in \mathsf{eval}_X(P_1, G),$ and $\mu_2 = \emptyset$ if $\mathsf{eval}_X(\mu(P_2), G) = \emptyset$ or $\mu_2 \in \mathsf{eval}_X(\mu_1(P_2), G)\}$

$P = P_1$ filter R :
$\{\mu \in \mathsf{eval}_X(P_1, G) \mid \mu \models R\}$

Proof. Let $D = \mathsf{ti}^{\pm}_X(G) \cup \{-\infty, +\infty\}$. To prove this theorem, we show the following properties, where P is an arbitrary TGP in Points 1a and 1c, and P is a safe TGP in Points 1b, 2, and 3:

1. For each $\mu \in [\![P]\!]^X_G$, the following holds:
   (a) for each $x \in \mathsf{bind}(P)$, we have $x \in \mathsf{dom}(\mu)$ and $\mu(x) \in D$ (even if P is not safe);
   (b) $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$ (if P is safe); and
   (c) if $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$, then $\mu \in \mathsf{eval}_X(P, G)$ (even if P is not safe).
2. $\mathsf{eval}_X(P, G) \subseteq [\![P]\!]^X_G$ (if P is safe).
3. If $\mathsf{ad}_X(G)$ is finite, then $[\![P]\!]^X_G$ is finite as well (if P is safe).

Note that, if P is safe, then from Points 1b and 1c we can conclude $[\![P]\!]^X_G \subseteq \mathsf{eval}_X(P, G)$; combined with Point 2, we can conclude $[\![P]\!]^X_G = \mathsf{eval}_X(P, G)$. Thus, the claim of this theorem follows from Points 1–3.

For Point 3, note that the conditions in Table 8 ensure that $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$ for each $\mu \in \mathsf{eval}_X(P, G)$. The set $\mathsf{ti}^{\pm}_X(G)$ is always finite; thus, if $\mathsf{ad}_X(G)$ is finite, then $\mathsf{eval}_X(P, G)$ is finite as well; but then, $[\![P]\!]^X_G$ is finite by Points 1b, 1c, and 2.

We prove Points 1 and 2 by induction on the structure of P. To simplify the presentation, we first consider Point 2. Assume that P is a basic TGP. Note that the ranges of the mappings in $\mathsf{eval}_X(P, G)$ are restricted to $\mathsf{ad}_X(G) \cup D$ (cf. Table 8), whereas in $[\![P]\!]^X_G$ they are restricted to $\mathsf{ad}_X(G) \cup \mathcal{TI}^{\pm}$ (cf. Table 4); apart from this difference, all remaining conditions are the same. Since $D \subseteq \mathcal{TI}^{\pm}$, Point 2 holds for P. Furthermore, assume that P is of the form P_1 and P_2. By induction assumption we have that $\mathsf{eval}_X(P_i, G) \subseteq [\![P_i]\!]^X_G$ for $i \in \{1, 2\}$, so then clearly $\mathsf{eval}_X(P, G) \subseteq [\![P]\!]^X_G$ as well. The cases when P is of the form P_1 union P_2 or P_1 filter R are analogous. Finally, assume that P is of the form P_1 opt P_2. Note that the following identities are true by the definition of $\bowtie$ and $\setminus$ for arbitrary P_1 and P_2.

$$[\![P_1]\!]^X_G \bowtie [\![P_2]\!]^X_G = \{\mu_1 \cup \mu_2 \mid \mu_1 \in [\![P_1]\!]^X_G \text{ and } \mu_2 \in [\![\mu_1(P_2)]\!]^X_G\}$$
$$[\![P_1]\!]^X_G \setminus [\![P_2]\!]^X_G = \{\mu_1 \mid \mu_1 \in [\![P_1]\!]^X_G \text{ and } [\![\mu_1(P_2)]\!]^X_G = \emptyset\}$$

The condition for P in Table 8 is thus equivalent to the one in Table 1. Assume now that P_1 and P_2 satisfy Points 1 and 2 and that P is safe. By Definition 21, P_1 is safe as well, so by Points 1 and 2 we have $\mathsf{eval}_X(P_1, G) = [\![P_1]\!]^X_G$. Furthermore, also by Definition 21, $\mu_1(P_2)$ is safe for each mapping $\mu_1 \in [\![P_1]\!]^X_G$, so by Points 1 and 2 we have that $\mathsf{eval}_X(\mu_1(P_2), G) = [\![\mu_1(P_2)]\!]^X_G$. Thus, Point 2 holds for P provided that P_1 and P_2 satisfy Point 1. We next show that Points 1a–1c hold for all possible forms of P.

Assume that $P = B$ at t_3 or $P = B$ during $[t_1, t_2]$ or $P = B$ occurs $[t_1, t_2]$ and consider an arbitrary mapping $\mu \in [\![P]\!]^X_G$. Since $\mathsf{bind}(P) = \emptyset$, Point 1a holds vacuously. Furthermore, if P is safe, then $\mathsf{tvar}(P) = \emptyset$, so t_1, t_2, and t_3 are not variables; thus, $\mathsf{dom}(\mu) = \mathsf{var}(B)$ and $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G)$, so Point 1b holds. Finally, if we have $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$, then the conditions in the definitions of $[\![P]\!]^X_G$ and $\mathsf{eval}_X(P, G)$ are equivalent for μ, so Point 1c holds.

Assume that $P = B$ maxint $[t_1, t_2]$ and consider an arbitrary mapping $\mu \in [\![P]\!]_G^X$. Let σ be a mapping for which $\delta_X(\mu(P), G)$ holds, and let $B' = \sigma(\mu(B))$. Assume that t_1 is a variable x_1 such that $\mu(x_1) \notin D$; by condition $\delta_X(\mu(P), G)$ then we have $\xi_X(G) \models \pi_X(B')\langle\mu(x_1)\rangle$ and $\xi_X(G) \not\models \pi_X(B')\langle\mu(x_1) - 1\rangle$. The latter condition implies that $\xi_X(G)$ is satisfiable, so by Claim 2 of Theorem 23 we have $\Xi_X(G, \mu(x_1)) \models_X B'$ and $\Xi_X(G, \mu(x_1) - 1) \not\models_X B'$. Furthermore, $\Xi_X(G, \mu(x_1)) \subseteq \Xi_X(G, \mu(x_1) - 1)$ by Proposition 22; moreover, first-order logic is monotonic, so we have $\Xi_X(G, \mu(x_1) - 1) \models_X B'$, which is a contradiction; thus $\mu(x_1) \in D$. In a completely analogous way we can show that, if t_2 is a variable x_2, then $\mu(x_2) \in D$. But then, $\mu \in \mathsf{eval}_X(P, G)$ and Points 1a–1c clearly hold.

Assume that $P = B$ $\mathsf{mintime}$ t_3 and consider an arbitrary mapping $\mu \in [\![P]\!]_G^X$. Let σ be a mapping for which $\delta_X(\mu(P), G)$ holds, and let $B' = \sigma(\mu(B))$. Assume that t_3 is a variable x_3 with $\mu(x_3) \notin D$; by condition $\delta_X(\mu(P), G)$ then $\xi_X(G) \models \pi_X(B')\langle\mu(x_3)\rangle$ and

$$\xi_X(G) \not\models \exists x^{\mathsf{t}} : [x^{\mathsf{t}} \leq \mu(x_3) - 1 \wedge \pi_X(B')\langle x_3\rangle];$$

by the latter condition, formula $\xi_X(G)$ is satisfiable and $\xi_X(G) \not\models \pi_X(B')\langle\mu(x_3) - 1\rangle$. By Theorem 23, we then have $\Xi_X(G, \mu(x_3)) \models_X B'$ and $\Xi_X(G, \mu(x_3) - 1) \not\models_X B'$. Furthermore, $\Xi_X(G, \mu(x_3)) \subseteq \Xi_X(G, \mu(x_3) - 1)$ by Proposition 22; moreover, first-order logic is monotonic, so we have $\Xi_X(G, \mu(x_3) - 1) \models_X B'$, which is a contradiction; thus $\mu(x_3) \in D$. But then, $\mu \in \mathsf{eval}_X(P, G)$ and Points 1a–1c clearly hold.

Assume that $P = B$ $\mathsf{maxtime}$ t_3. The proof is completely analogous to the previous case.

Assume that $P = P_1$ and P_2 and consider an arbitrary mapping $\mu \in [\![P]\!]_G^X$. Let μ_1 and μ_2 be the mappings compatible with μ such that $\mathsf{dom}(\mu_i) = \mathsf{var}(P_i)$ for each $i \in \{1, 2\}$; clearly, we have $\mu_i \in [\![P_i]\!]_G^X$.

- Consider an arbitrary variable $x \in \mathsf{bind}(P)$. By Table 6, $x \in \mathsf{bind}(P_1)$ or $x \in \mathsf{bind}(P_2)$. If $x \in \mathsf{bind}(P_1)$, then $x \in \mathsf{dom}(\mu_1)$ and $\mu_1(x) \in D$ by the induction assumption; but then, $x \in \mathsf{dom}(\mu)$ and $\mu(x) \in D$. By analogous reasoning for $x \in \mathsf{bind}(P_2)$, we see that Point 1a holds.

- If P is safe, then $\mathsf{tvar}(P) = \mathsf{bind}(P)$. Consider now an arbitrary variable $x \in \mathsf{var}(P)$: if $x \notin \mathsf{tvar}(P)$, then $\mu(x) \in \mathsf{ad}_X(G)$; otherwise, $x \in \mathsf{bind}(P)$ due to the safety of P, so $\mu(x) \in D$ by Point 1a. Consequently, $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$, and Point 1b holds.

- Assume that $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$; then clearly we have $\mathsf{rng}(\mu_i) \subseteq \mathsf{ad}_X(G) \cup D$ for each $i \in \{1, 2\}$. Since Point 1c holds for P_i and $\mu_i \in [\![P_i]\!]_G^X$ by the induction assumption, we have $\mu_i \in \mathsf{eval}_X(P_i, G)$. But then, $\mu \in \mathsf{eval}_X(P, G)$ by Table 8, so Point 1c holds.

Assume that $P = P_1$ union P_2 and consider an arbitrary mapping $\mu \in [\![P]\!]_G^X$. Clearly, we have $\mu \in [\![P_i]\!]_G^X$ for $i = 1$ or $i = 2$.

- Consider an arbitrary variable $x \in \mathsf{bind}(P)$. Then $x \in \mathsf{bind}(P_i)$ by Table 6, so $x \in \mathsf{dom}(\mu)$ and $\mu(x) \in D$ by the induction assumption, and Point 1a holds.

- If P is safe, then P_i is safe as well. But then, Point 1b is satisfied for P_i and μ by the induction assumption, we have $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$, so Point 1b holds.

- Assume that $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$. Since Point 1c holds for P_i and μ by the induction assumption, we have $\mu \in \mathsf{eval}_X(P_i, G)$. But then, $\mu \in \mathsf{eval}_X(P, G)$ as well, so Point 1c holds.

Assume that $P = P_1$ opt P_2 and consider an arbitrary mapping $\mu \in [\![P]\!]_G^X$. Let μ_1 be the mapping compatible with μ such that $\mathsf{dom}(\mu_1) = \mathsf{var}(P_1)$; clearly, $\mu_1 \in [\![P_1]\!]_G^X$. Furthermore, let μ_2 be the maximal mapping compatible with μ such that $\mathsf{dom}(\mu_2) \subseteq \mathsf{var}(\mu_1(P_2))$; clearly, $\mu_2 = \emptyset$ or $\mu_2 \in [\![\mu_1(P_2)]\!]_G^X$.

- Consider an arbitrary variable $x \in \mathsf{bind}(P)$. Then $x \in \mathsf{bind}(P_1)$ by Table 6, so we have $x \in \mathsf{dom}(\mu_1)$ and $\mu_1(x) \in D$ by the induction assumption; but then, $x \in \mathsf{dom}(\mu)$ and $\mu(x) \in D$, and Point 1a holds.

- Assume that P is safe. Then P_1 is safe as well, so Point 1b is satisfied for P_1 and μ_1 by the induction assumption, so we have $\mathsf{rng}(\mu_1) \subseteq \mathsf{ad}_X(G) \cup D$. Point 1b clearly holds if $\mu_2 = \emptyset$. Therefore, assume that $\mu_2 \neq \emptyset$ and let $P_2' = \mu_1(P)$; clearly, P_2' is safe, since the actual values that μ_1 assigns to the variables in $\mathsf{bind}(P_1)$ do not affect the definition of safety. Point 1b is satisfied for P_2' and $\mu_2 \in [\![\mu_1(P_2)]\!]_G^X$ by the induction assumption, so $\mathsf{rng}(\mu_2) \subseteq \mathsf{ad}_X(G) \cup D$. Since $\mu = \mu_1 \cup \mu_2$, we have $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$, so Point 1b holds.

- Assume that $\mathsf{rng}(\mu) \subseteq \mathsf{ad}_X(G) \cup D$. Then we have $\mathsf{rng}(\mu_i) \subseteq \mathsf{ad}_X(G) \cup D$ for each $i \in \{1, 2\}$. Point 1c holds for P_1 and μ_1, as well as for $\mu_1(P_2)$ and μ_2, so $\mu_1 \in \mathsf{eval}_X(P_1, G)$ and $\mu_2 \in \mathsf{eval}_X(\mu_1(P_2), G)$. But then, $\mu \in \mathsf{eval}_X(P, G)$ by Table 8, and Point 1c holds.

Assume that $P = P_1$ filter. Points 1a–1c hold for P_1 by the induction assumption; but then, by the definition of $\mathsf{eval}_X(P, G)$ they clearly hold for P as well. □

Intuitively, $\mathsf{eval}_X(P, G)$ evaluates P bottom-up, in a way similar to the one shown in Table 1; however, instead of matching the variables from P to the values in the infinite set $\mathsf{ad}_X(G) \cup \mathcal{TI}^{\pm}$, the values from the finite set $\mathsf{ad}_X(G) \cup \mathsf{ti}_X^{\pm}(G) \cup \{-\infty, +\infty\}$ suffice because the definition of safety ensures that each variable is "bound" by a safe basic TGP.

Note that the computation of $[\![P]\!]_G^X$ is not a decision (i.e., a yes/no) problem. Therefore, in order to determine the complexity of our query language, we must consider a decision version of the problem: given an entailment relation X, a temporal graph G, a safe temporal group pattern P, and a mapping μ, decide whether $\mu \in [\![P]\!]_G^X$.

A worst-case optimal procedure for deciding $\mu \in \llbracket P \rrbracket_G^X$ can be obtained by a slight modification of [30, Algorithm 1]: the only difference is in the treatment of basic TGPs, which can be handled as shown in Theorem 23. Thus, one can show that the complexity of deciding $\mu \in \llbracket P \rrbracket_G^X$ is either PSPACE or the complexity of the corresponding nontemporal entailment, whichever is higher. The proof of these claims is quite straightforward, so we refrain from going into further detail.

One might naïvely try to solve the above mentioned decision problem by computing $\mathsf{eval}_X(P, G)$ and then checking whether $\mu \in \mathsf{eval}_X(P, G)$. Function $\mathsf{eval}_X(P, G)$, however, can return exponentially many mappings (in $\mathsf{var}(P)$ and the number of URI references in G). Note that the latter is not because $\mathsf{eval}_X(P, G)$ is suboptimal: it is simply due to the fact that $\llbracket P \rrbracket_G^X$ can be exponential in size. Thus, the naïve decision procedure outlined above might use exponential space for storing intermediate results and would therefore be suboptimal. Function $\mathsf{eval}_X(P, G)$ is nevertheless useful since it shows that the answers to safe TGPs are finite: we would not be able to prove this by considering only the decision version of the problem. Furthermore, $\mathsf{eval}_X(P, G)$ provides us with a bottom-up algorithm that is quite similar to the algorithms used to evaluate relational algebra queries [43].

6. Optimized Query Answering

In order to compute the answer to a safe basic TGP P using the function $\mathsf{eval}_X(P, G)$ presented in Section 5.2, we can enumerate all candidate mappings and then decide for each mapping whether the mapping is contained in the answer. The latter problem can be solved using the reduction from Section 5.1 by checking one or more nontemporal entailments. Such an approach is very general and can handle arbitrary entailment relations; however, the resulting algorithm is unlikely to be practicable: the set of all candidate mappings is likely to be much larger than the answer to P, so the algorithm is likely to perform a lot of computation. In this section we optimize this general algorithm for specific entailment relations to make it more goal-directed.

In Section 6.1 we present a goal-oriented approach for evaluating TGPs under Simple Entailment. The resulting algorithm uses a preprocessing phase in order to normalize a temporal graph, after which temporal queries can be evaluated by simple matching in the normalization. This approach can be straightforwardly integrated into virtually all existing RDF management systems.

In Section 6.2 we extend our optimization to entailment relations that can be characterized by a set of deterministic rules, such as RDF(S) and OWL 2 RL/RDF Entailment. In particular, we show how to apply the rules to a temporal graph in a bottom-up fashion and thus materialize all relevant consequences. The resulting materialized graph can then be normalized, after which temporal queries can be answered using the optimization from Section 6.1.

In order to simplify our definitions, in the rest of this section we consider temporal triples of the form $\langle s, p, o\rangle[t]$ as syntactic abbreviations for $\langle s, p, o\rangle[t, t]$.

6.1. Simple Entailment

Simple Entailment is the basic entailment relation in which BGPs can be evaluated in nontemporal graphs by simple lookup. Such an approach provides the basis of virtually all practical RDF management systems, so it would be beneficial if similar approaches were applicable to TGPs and temporal graphs. As the following example demonstrates, however, a naïve application of graph matching would result in an incorrect algorithm.

Example 27. Let P be as in (65), and let G be the temporal graph that contains the temporal triples shown in (8)–(11) in Section 3; then, $\llbracket P \rrbracket_G^{simple}$ is shown in (66).

$$P = \{\langle :LHR, :flightTo, :MUC\rangle\} \ \mathsf{maxint}\ [x, y] \quad (65)$$

$$\llbracket P \rrbracket_G^{simple} = \{\{x \mapsto 50, y \mapsto 150\}\} \quad (66)$$

Note, however, that G does not contain temporal triple $\alpha = \langle :LHR, :flightTo, :MUC\rangle[50, 150]$, so $\llbracket P \rrbracket_G^{simple}$ cannot be computed by simple lookup. Temporal graph G is equivalent under Simple Entailment to the normalized temporal graph $\mathsf{nrm}(G)$ obtained from G by replacing (8) and (9) with α; furthermore, P can be evaluated in $\mathsf{nrm}(G)$ by simple lookup. ◊

Based on these observations, we develop our optimized approach for TGP evaluation.

Let G be a temporal graph, and let $A = \langle s, p, o\rangle[t_1, t_2]$ be a temporal triple in G. Then A *directly overlaps* with $\langle s, p, o\rangle[t_1', t_2'] \in G$ if $\max(t_1, t_1') \leq \min(t_2, t_2')$; *overlaps* is the transitive closure of "directly overlaps"; and G_A is the set that contains precisely all temporal triples in G that overlap with A. The normalization of A w.r.t. G is the temporal triple $\mathsf{nrm}_A(A)$ defined as follows:

$$\begin{aligned} \mathsf{nrm}_G(A) &= \langle s, p, o\rangle[t_1, t_2] \\ t_1 &= \min_{\langle s,p,o\rangle[t_1^i, t_2^i] \in G_A} t_1^i \\ t_2 &= \max_{\langle s,p,o\rangle[t_1^i, t_2^i] \in G_A} t_2^i \end{aligned}$$

Finally, the *normalization* of G is the temporal graph $\mathsf{nrm}(G)$ obtained from G by replacing each temporal triple $C \in G$ with $\mathsf{nrm}_G(C)$.

Temporal graph $\mathsf{nrm}_G(C)$ can be computed from G using Algorithm 1. Due to sorting, all temporal triples in G are grouped by s, p, and o, and within each group, the triples are ordered by t_1 and t_2. Thus, at each point in time during the execution of the for-loop, either s^c is undefined or $\langle s^c, p^c, o^c\rangle[t_1^c, t_2^c]$ is the normalization w.r.t. G of all triples of the form $\langle s^c, p^c, o^c\rangle[t_1, t_2]$ processed thus far. Consequently, the algorithm correctly computes $\mathsf{nrm}(G)$. In a system where G is stored in a relational database,

Algorithm 1 Computing $\mathsf{nrm}(G)$

```
Sort the triples $\langle s,p,o\rangle[t_1,t_2]$ in $G$ by $s$, $p$, and $o$, $t_1$, and $t_2$
$R := \emptyset$
$s^c := p^c := o^c := t_1^c := t_2^c := \mathit{undefined}$
for all $\langle s,p,o\rangle[t_1,t_2] \in G$ do
    if $s \neq s^c$ or $p \neq p^c$ or $o \neq o^c$ or $t_1 > t_2^c$ then
        if $s^c \neq \mathit{undefined}$ then
            $R := R \cup \{\langle s^c,p^c,o^c\rangle[t_1^c,t_2^c]\}$
        end if
        $s^c := s,\ p^c := p,\ o^c := o,\ t_1^c := t_1,\ t_2^c := t_2$
    end if
    $t_2^c = \max(t_2^c, t_2)$
end for
if $s^c \neq \mathit{undefined}$ then
    $R := R \cup \{\langle s^c,p^c,o^c\rangle[t_1^c,t_2^c]\}$
end if
return $R$
```

Algorithm 1 can be straightforwardly implemented using a stored procedure.

We next show how to answer TGPs under Simple Entailment using $\mathsf{nrm}(G)$.

Definition 28. *Let B be a BGP containing $\langle s_i,p_i,o_i\rangle$ for $1 \leq i \leq k$; and let $\vec{x} = \langle x_1,\ldots,x_k\rangle$, $\vec{y} = \langle y_1,\ldots,y_k\rangle$, z_1, and z_2 be distinct variables not occurring in B. Then, $\mathsf{M}_B(\vec{x},\vec{y},z_1,z_2)$ is the following conjunction:*

$$\begin{aligned}\mathsf{M}_B(\vec{x},\vec{y},z_1,z_2) \ \equiv\ & \left(\bigwedge_{i=1}^{k} \langle s_i,p_i,o_i\rangle[x_i,y_i]\right) \wedge \\ & z_1 = \max(x_1,\ldots,x_k) \wedge \\ & z_2 = \min(y_1,\ldots,y_k) \wedge \\ & z_1 \leq z_2\end{aligned}$$

For P a basic TGP, φ_P is the first-order formula defined as shown in Table 9.

Let G be a temporal graph. For P a safe basic temporal group pattern, $\mathsf{seval}(P,G)$ is the set of mappings obtained by evaluating φ_P in $\mathsf{nrm}(G)$, where the latter is treated in the obvious way as a relation in a relational database. For P a safe TGP of any other type, $\mathsf{seval}(P,G)$ is defined analogously as in Table 8.

Theorem 29. *For each temporal graph G and each safe temporal group pattern P, $\mathsf{seval}(P,G) = [\![P]\!]_G^{simple}$.*

Proof. Temporal graphs G and $\mathsf{nrm}(G)$ are obviously equivalent w.r.t. any entailment relation. Furthermore, by the definition of normalization, $\langle s,p,o\rangle[t_1,t_2] \in \mathsf{nrm}(G)$ if and only if $[\![\{\langle s,p,o\rangle\}\ \mathsf{maxint}\ [t_1,t_2]]\!]_G^{simple} \neq \emptyset$—that is, each temporal assertion in $\mathsf{nrm}(G)$ covers the maximal interval in which the assertion is valid.

Now let P be a safe TGP of the form B maxint $[t_1,t_2]$. Conjunction

$$\left(\bigwedge_{i=1}^{k} \langle s_i,p_i,o_i\rangle[x_i,y_i]\right)$$

is true in $\mathsf{nrm}(G)$ precisely for each mapping σ such that $\mathsf{dom}(\sigma) = \mathsf{var}(P) \cup \{\vec{x},\vec{y}\}$ and

$$\langle\sigma(s),\sigma(p),\sigma(o)\rangle[\sigma(t_1),\sigma(t_2)] \in \mathsf{nrm}(G),$$

so $\mathsf{M}_B(\vec{x},\vec{y},z_1,z_2)$ is true if and only if, in addition, interval (z_1,z_2) is the nonempty intersection of all intervals (x_i,y_i); consequently, φ_P is true in $\mathsf{nrm}(G)$ precisely for those mappings μ such that $\mu \in [\![P]\!]_G^{simple}$.

The proofs for the remaining forms of P are analogous and are omitted for the sake of brevity. □

Computing answers to safe basic TGPs can thus be reduced to the evaluation of first-order queries in the normalization of a temporal graph, and the latter can be implemented using SQL. Furthermore, all other types of TGPs can be handled by translating them into SQL, so Theorem 29 provides the foundation for a practical implementation of our framework in RDF systems.

We finish this section with an observation that, in SQL, $\mathsf{mintime}$ and $\mathsf{maxtime}$ can be handled using aggregate functions, rather than nested subqueries. We do not discuss this optimization any further because of the well-known problems involved in capturing the semantics of aggregate functions in first-order logic.

6.2. 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 (67).

$$A_1 \wedge \ldots \wedge A_n \rightarrow B \qquad (67)$$

Numerous RDF systems, such as such as Jena [3], Jena2 [4], Sesame [5], and Oracle [6], use a *materialization* approach to deal with such X: they initially compute and store all consequences w.r.t. Γ_X of a nontemporal graph. Any query can then be evaluated in the materialization by straightforward lookup.

We next show how to materialize the temporal closure $\mathsf{cls}_X(G)$ of a temporal graph G using the rules from Γ_X. After this is done, we can normalize $\mathsf{cls}_X(G)$ and use the algorithm from Section 6.1 in order to answer TGPs.

By examining the semantics of temporal entailment in Section 3, one might naïvely try to transform each rule of the form (67) into

$$A_1[x^{\mathsf{t}}] \wedge \ldots \wedge A_n[x^{\mathsf{t}}] \rightarrow B[x^{\mathsf{t}}] \qquad (68)$$

and then apply the transformed rules to G. Such an approach, however, will not work: most triples in a temporal graph are likely to be of the form $\langle s,p,o\rangle[t_1,t_2]$ rather than $\langle s,p,o\rangle[t]$, so a rule such as (68) will not match to most triples in a graph. We can, however, use a slight modification of this idea. In the rest of this section, we slightly abuse notation and allow a temporal graph G without blank nodes to be infinite; then, $\pi_X(G) = \bigcup_{A\in G} \pi_X(A)$.

Table 9: Evaluation of Safe TGPs under Simple Entailment

P	$\varphi_{\vec{v}}$
B at t_3	$\exists \vec{x}, \vec{y}, z_1, z_2 : \mathsf{M}_B(\vec{x}, \vec{y}, z_1, z_2) \wedge z_1 \leq t_3 \leq z_2$
B during $[t_1, t_2]$	$\exists \vec{x}, \vec{y}, z_1, z_2 : \mathsf{M}_B(\vec{x}, \vec{y}, z_1, z_2) \wedge z_1 \leq t_1 \leq t_2 \leq z_2$
B occurs $[t_1, t_2]$	$\exists \vec{x}, \vec{y}, z_1, z_2 : \mathsf{M}_B(\vec{x}, \vec{y}, z_1, z_2) \wedge \max(z_1, t_1) \leq \min(z_2, t_2)$
B maxint $[t_1, t_2]$	$\exists \vec{x}, \vec{y}, z_1, z_2 : \mathsf{M}_B(\vec{x}, \vec{y}, z_1, z_2) \wedge t_1 = z_1 \wedge t_2 = z_2$
B mintime t_3	$\exists \vec{x}, \vec{y}, z_1, z_2 : \mathsf{M}_B(\vec{x}, \vec{y}, z_1, z_2) \wedge t_3 = z_1 \wedge \forall \vec{x'}, \vec{y'}, w_1, w_2 : \mathsf{M}_B(\vec{x'}, \vec{y'}, w_1, w_2) \rightarrow z_1 \leq w_1$
B maxtime t_3	$\exists \vec{x}, \vec{y}, z_1, z_2 : \mathsf{M}_B(\vec{x}, \vec{y}, z_1, z_2) \wedge t_3 = z_2 \wedge \forall \vec{x'}, \vec{y'}, w_1, w_2 : \mathsf{M}_B(\vec{x'}, \vec{y'}, w_1, w_2) \rightarrow z_2 \geq w_2$

Definition 30. *For X and Γ_X as stated above, let Σ_X be the set containing the rule* (69) *for each rule* (67) *in Γ_X.*

$$\begin{array}{ll} A_1[x_1, y_1] \wedge & \\ \dots & \\ A_n[x_n, y_n] \wedge & \\ z_1 = \max(x_1, \dots, x_n) \wedge & \\ z_2 = \min(y_1, \dots, y_n) \wedge & \\ z_1 \leq z_2 & \rightarrow B[z_1, z_2] \end{array} \tag{69}$$

Let G be a temporal graph containing only triples of the form $\langle s, p, o\rangle[t_1, t_2]$.[3] *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 X-skolemization of G.*

Example 31. Let G be the temporal graph that contains the temporal triples shown in (8)–(11) in Section 3. By applying the approach from Definition 30 to G under RDFS entailment, one can see that (10) and (11) produce the following temporal triple:

$$\langle :\!Munich, :\!hasEvent, :\!Oktoberfest\rangle[130, 180] \tag{70}$$

This triple captures the RDFS consequences of G. $\diamondsuit$

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.

Theorem 32. *Let X and G be as stated in Definition 30, and let $G' = \mathsf{cls}_X(G)$. Then, for each temporal group pattern P, we have $[\![P]\!]_G^X = [\![P]\!]_{G'}^{simple}$.*

Proof. Let P be an arbitrary TGP, and let G^1 be the X-skolemization of G. Clearly $[\![P]\!]_G^X = [\![P]\!]_{G^1}^X$, so the claim of this theorem holds if and only if $[\![P]\!]_{G^1}^X = [\![P]\!]_{G'}^{simple}$.

Let G^2 be the temporal graph that contains $\langle s, p, o\rangle[t]$ for each $\langle s, p, o\rangle[t_1, t_2] \in G^1$ and each t with $t_1 \leq t \leq t_2$. Furthermore, let Ω_X be the set containing a rule of the form (68) for each rule of the form (67) in Γ_X; and let G^3 be the temporal graph obtained by applying exhaustively the rules in Ω_X to G^2. Since Ω_X is a deterministic set of rules, it can be understood as a datalog program; furthermore, G^3 can be seen as the least fixpoint of G^2 w.r.t. Ω_X. Thus, by [43], for each temporal triple A of the form $\langle s, p, o\rangle[t]$, we have $\pi_{simple}(G^3) \models \pi_{simple}(A)$ if and only if

$$\pi_X(G_1) \cup \{\forall x^{\mathsf{t}} : \varphi\langle x^{\mathsf{t}}\rangle \mid \varphi \in \Gamma_X\} \models \pi_{simple}(A),$$

so $[\![P]\!]_{G^1}^X = [\![P]\!]_{G^3}^{simple}$.

Let G^4 be the temporal graph that contains $\langle s, p, o\rangle[t]$ for each $\langle s, p, o\rangle[t_1, t_2] \in G'$ and each t with $t_1 \leq t \leq t_2$. It is straightforward to see that each derivation tree for a triple $\langle s, p, o\rangle[t] \in G^3$ can be transformed to a derivation tree of $\langle s, p, o\rangle[t_1, t_2] \in G'$ such that $t_1 \leq t \leq t_2$; similarly, one can easily see that each derivation tree for a triple $\langle s, p, o\rangle[t_1, t_2] \in G'$ can be transformed to a derivation tree of the triple $\langle s, p, o\rangle[t] \in G^3$ for each $t_1 \leq t \leq t_2$. Consequently, $G^3 = G^4$, so $\pi_{simple}(G^3)$ and $\pi_{simple}(G')$ are equivalent, and $[\![P]\!]_{G^3}^{simple} = [\![P]\!]_{G'}^{simple}$. $\square$

We next comment on a somewhat technical issue involved in Definition 30. For the sake of generality, our definitions allow Γ_X to be infinite so, given a finite temporal graph G, the closure $\mathsf{cls}_X(G)$ can be infinite as well. Thus, to correctly state our results, we need the ability to deal with infinite temporal graphs. Note, however, that blank nodes are encoded using existential quantifiers; therefore, to correctly capture the semantics of blank nodes in infinite temporal graphs, we might need to existentially quantify over infinite sets of formulae, which is not possible in standard first-order logic. In order to avoid such technical problems, Definition 30 simply skolemizes blank nodes in G before applying the rules. This does not affect the answers to TGPs since the definition of TGP semantics already involves skolemizing the blank nodes in G.

Rules of the form (69) are just plain datalog rules so, provided that Γ_X is finite, $\mathsf{cls}_X(G)$ can be computed using standard datalog evaluation techniques, such as variants of the seminaïve strategy [44]. These techniques are used in most materialization-based RDF systems to handle

[3]As explained earlier, triples of the form $\langle s, p, o\rangle[t]$ can be considered abbreviations for $\langle s, p, o\rangle[t, t]$.

nontemporal RDF, so extending the mentioned systems to handle temporal graphs should be straightforward.

7. Related Work

The management of validity time has been extensively studied in relational databases and artificial intelligence. Neither RDF nor OWL, however, supports validity time, and SPARQL does not provide the primitives for temporal querying of data. These deficiencies have been recognized in the Semantic Web community, and several proposals have emerged. In this section we present an overview of these related proposals and discuss the similarities and differences with our work.

Chomicki presented an extensive overview of the principles of the representation and querying of validity time in relational databases [39], and Vila did the same in the context of artificial intelligence [40]. The surveyed results, however, are quite general and do not handle the family of Semantic Web languages. In this paper we adapted the principles from [39, 40] to the specifics of RDF, OWL, and SPARQL, and we provided appropriate reasoning algorithms. In particular, we distinguish logical from physical temporal databases as in [39], and we use temporal arguments to associate time instants with facts as in [40].

Gutierrez et al. presented a comprehensive framework for representing and querying validity time in RDF [22]. They defined a syntactic notion of temporal RDF graphs quite similar to the one presented in this paper. To obtain a notion of temporal entailment, they defined the *snapshot* of a temporal graph G at a time instant $t \in \mathcal{TI}$ as the nontemporal graph $G(t)$ that contains each $\langle s, p, o\rangle$ such that $\langle s, p, o\rangle[t] \in G$ or $\langle s, p, o\rangle[t_1, t_2] \in G$ and $t_1 \leq t \leq t_2$; then, for G_1 and G_2 temporal graphs, $G_1 \models G_2$ holds if and only if $G_1(t) \models G_2(t)$ for each $t \in \mathcal{TI}$. To obtain a practical characterization of temporal entailment, the authors defined the *slice closure* of a temporal graph G as the union of the closures of $G(t)$ for each $t \in \mathcal{TI}$, where the notion of a closure of a nontemporal graph is taken from [45]. Finally, the authors also developed an encoding of temporal graphs into nontemporal graphs that preserves temporal entailment, and they presented a basic temporal query language. This framework was extended in [23] with more general temporal constraints.

Although there are many similarities between our work and the framework presented in [22], our work differs in several important ways.

- The notion of temporal graph entailment from [22] is not applicable to OWL 2 Direct Entailment, and the notion of slice closures is applicable only to deterministic entailment relations without existential quantifiers, which excludes OWL 2 RDF-Based Entailment. In contrast, our work provides a unifying semantic and algorithmic framework for the management of validity time in the Semantic Web that is applicable to an arbitrary entailment relation, including OWL 2 Direct Entailment and OWL 2 RDF-Based Entailment. Note that our approach produces the same consequences as [22] for RDFS Entailment.
- Expressions of the form $\langle s, p, o\rangle[t_1, t_2]$ are considered in [22] to be syntactic abbreviations for temporal triples $\langle s, p, o\rangle[t]$ for each $t_1 \leq t \leq t_2$, rather than full-fledged parts of a temporal graph. Thus, t_1 and t_2 in such an expression cannot be $-\infty$ and $+\infty$, as this would result in an infinite temporal graph and would consequently invalidate key technical results, such as [22, Proposition 2]. Such a definition has another side-effect: each triple $\langle s, p, o\rangle[t_1, t_2]$ contributes $O(t_2 - t_1)$ to the size of a temporal graph (this is implicitly assumed in all technical results in [22]): since the slice closure of G is defined as the union of a closure of $G(t)$ for each $t \in \mathcal{TI}$, if G contains $\langle s, p, o\rangle[t_1, t_2]$, at least $t_2 - t_1 + 1$ time instants must be considered during the computation of a slice closure. Thus, the presence of large time intervals in a temporal graph can adversely affect the complexity of the employed algorithms. In contrast, temporal triples of the form $\langle s, p, o\rangle[t_1, t_2]$ are "first-class citizens" in our framework, they can be used to express facts with unbounded validity intervals, and the size of their time intervals does not affect the complexity of our reasoning algorithms.
- Third, the query language presented in [22] does not address the issues we discussed in Section 4.

Pugliese et al. [24] extended the approach by Gutierrez et al. in several ways. First, they allowed temporal triples to refer to unknown, rather than precisely named time points; second, they defined a temporal query language based on graph matching; and third, they presented a way for indexing temporal graphs. Since the approach by Pugliese et al. is based on [22], it does not handle nondeterministic entailment relations or entailment relations that involve existential quantification; furthermore, the presented query language does not address the problems discussed in Section 4.

Tappolet and Bernstein [25] discussed various technical issues involved in an implementation of the approach by [22] and presented a temporal extension of SPARQL. However, neither the syntax nor the semantics of the presented query language has been formally specified.

There are numerous proposals in the literature for extending description logics with validity time; Artale and Franconi [26] and Lutz et al. [27] presented comprehensive surveys of the relevant results. These works, however, typically focus on expressing and reasoning with complex temporal constraints. Such constraints usually allow for quantification over (potentially unknown) time instants; thus, the computational problems in temporal description logics are usually quite different from the ones in the Semantic Web, since the latter typically deal with efficient retrieval and management of known time instants.

Milea et al. presented a temporal extension of OWL [28]. Time instants are encoded as values of a particular concrete domain, and predefined concepts and properties are used to associate time intervals with facts. The authors, however, do not discuss an appropriate temporal query language.

In a somewhat independent line of research, several frameworks for annotating RDF triples were developed. One of the first such proposals was by Udrea et al. [46]; Straccia et al. extended this proposal in [29] with a more general deductive system and they presented in [47] a language for querying annotated RDF. These frameworks allow an RDF triple to be annotated by an element of a suitably defined *annotation domain*. The latter is assumed to exhibit some very general mathematical properties, such as being partially ordered or being a lattice, which is used to define the semantics of annotated RDF graphs and obtain an adequate deductive system. RDF annotation frameworks are thus very general and can capture a wide range of annotation information, such as fuzziness, provenance, or validity time. The latter can be captured by an annotation domain that contains time intervals; thus, the temporal domain is interval-based. This is in contrast to the work by Gutierrez et al. [22] and our work in which the temporal domain is point-based. Furthermore, unlike the work presented in this paper, none of the frameworks listed above are applicable to nondeterministic entailment relations, entailment relations that involve existential quantifiers, or OWL 2 Direct Entailment.

8. 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 ExperienceOn's backend 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 [44]. Datalog rules are modified as described in Section 6.2 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 as outlined in Section 6.1 and then evaluated using the query engine of PostgreSQL. The source code of the system is not open, and ExperienceOn 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 ExperienceOn 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 question is to see whether the general algorithm from Section 4 can be successfully used with entailment relations such as OWL 2 Direct Entailment. We believe this to be possible provided that the algorithm is adequately optimized.

References

[1] G. Klyne, J. J. Carroll, Resource Description Framework (RDF): Concepts and Abstract Syntax (February 10 2004).

[2] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Journal of Web Semantics: Science, Services and Agents on the World Wide Web 6 (4) (2008) 309–322.

[3] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson, Jena: Implementing the Semantic Web Recommendations, in: S. I. Feldman, M. Uretsky, M. Najork, C. E. Wills (Eds.), Proc. of the 13th Int. Conf. on World Wide Web (WWW 2004)—Alternate Track, ACM, New York, NY, USA, 2004, pp. 74–83.

[4] K. Wilkinson, C. Sayers, H. A. Kuno, D. Reynolds, Efficient RDF Storage and Retrieval in Jena2, in: I. F. Cruz, V. Kashyap, S. Decker, R. Eckstein (Eds.), Proc. of the 1st Int. Workshop on Semantic Web and Databases (SBWCB 2003), Berlin, Germany, 2003, pp. 131–150.

[5] J. Broekstra, A. Kampman, F. van Harmelen, Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, in: I. Horrocks, J. A. Hendler (Eds.), Proc. of the 1st Int. Semantic Web Conf. (ISWC 2002), Vol. 2342 of LNCS, Springer, Sardinia, Italy, 2002, pp. 54–68.

[6] E. I. Chong, S. Das, G. Eadon, J. Srinivasan, An Efficient SQL-based RDF Querying Scheme, in: K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, B. C. Ooi (Eds.), Proc. of the 31st Int. Conf. on Very Large Data Bases (VLDB 2005), Trondheim, Norway, 2005, pp. 1216–1227.

[7] D. Wood, P. Gearon, T. Adams, Kowari: A Platform for Semantic Web Storage and Analysis, in: XTech 2005 Conference, 2005.

[8] S. Harris, N. Lamb, N. Shadbol, 4store: The Design and Implementation of a Clustered RDF Store, in: Proc. of the 5th Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009), Washington DC, USA, 2009.

[9] A. Harth, S. Decker, Optimized Index Structures for Querying RDF from the Web, in: Proc. of the 3rd Latin American Web Congress (LA-Web 2005), Buenos Aires, Argentina, 2005, pp. 71–80.

[10] C. Weiss, P. Karras, A. Bernstein, Hexastore: Sextuple Indexing for Semantic Web Data Management, Proceedings of the VLDB Endowment 1 (1) (2008) 1008–1019.

[11] D. Beckett, The Design and Implementation of the Redland RDF Application Framework, in: Proc. of the 10th Int. World Wide Web Conf. (WWW 2001), ACM, 2001, pp. 449–456.

[12] D. Tsarkov, I. Horrocks, FaCT++ Description Logic Reasoner: System Description, in: Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006), Vol. 4130 of LNAI, Springer, Seattle, WA, USA, 2006, pp. 292–297.

[13] B. Parsia, E. Sirin, Pellet: An OWL-DL Reasoner, Poster at the 3rd Int. Semantic Web Conference (ISWC 2004) (November 7–11 2004).

[14] B. Motik, R. Shearer, I. Horrocks, Hypertableau Reasoning for Description Logics, Journal of Artificial Intelligence Research 36 (2009) 165–228.

[15] A. Sidhu, T. Dillon, E. Chang, B. S. Sidhu, Protein Ontology Development using OWL, in: Proc. of the OWL: Expreiences

and Directions Workshop (OWLED 2005), Vol. 188 of CEUR WS Proceedings, Galway, Ireland, 2005.

[16] C. Golbreich, S. Zhang, O. Bodenreider, The Foundational Model of Anatomy in OWL: Experience and Perspectives, Journal of Web Semantics 4 (3) (2006) 181–195.

[17] J. Goodwin, Experiences of using OWL at the Ordnance Survey, in: Proc. of the OWL: Expreiences and Directions Workshop (OWLED 2005), Vol. 188 of CEUR WS Proceedings, Galway, Ireland, 2005.

[18] S. Derriere, A. Richard, A. Preite-Martinez, An Ontology of Astronomical Object Types for the Virtual Observatory, in: Proc. of the 26th meeting of the IAU: Virtual Observatory in Action: New Science, New Technology, and Next Generation Facilities, Prague, Czech Republic, 2006, pp. 17–18.

[19] D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer, S. Katz, Reengineering Thesauri for New Applications: The AGROVOC Example, Journal of Digital Information 4 (4).

[20] L. Lacy, G. Aviles, K. Fraser, W. Gerber, A. Mulvehill, R. Gaskill, Experiences Using OWL in Military Applications, in: Proc. of the OWL: Expreiences and Directions Workshop (OWLED 2005), Vol. 188 of CEUR WS Proceedings, Galway, Ireland, 2005.

[21] R. T. Snodgrass, I. Ahn, Temporal Databases, IEEE Computer 19 (9) (1986) 35–42.

[22] C. Gutierrez, C. A. Hurtado, A. A. Vaisman, Introducing Time into RDF, IEEE Transactions on Knowledge and Data Engineering 19 (2) (2007) 207–218.

[23] C. A. Hurtado, A. A. Vaisman, Reasoning with Temporal Constraints in RDF, in: J. J. Alferes, J. Bailey, W. May, U. Schwertel (Eds.), Proc. of the 4th Int. Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2006), Vol. 4187 of LNCS, Springer, Budva, Montenegro, 2006, pp. 164–178.

[24] A. Pugliese, O. Udrea, V. S. Subrahmanian, Scaling RDF with Time, in: J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, X. Zhang (Eds.), Proc. of the 17th Int. Conf. on World Wide Web (WWW 2008), ACM, Beijing, China, 2008, pp. 605–614.

[25] J. Tappolet, A. Bernstein, Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL, in: L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, E. P. B. Simperl (Eds.), Proc. of the 6th European Semantic Web Conference (ESWC 2009), Vol. 5554 of LNCS, Springer, Heraklion, Greece, 2009, pp. 308–322.

[26] A. Artale, E. Franconi, A survey of temporal extensions of description logics, Annals of Mathematics and Artificial Intelligence 30 (1–4) (2000) 171–210.

[27] C. Lutz, F. Wolter, M. Zakharyaschev, Temporal Description Logics: A Survey, in: S. Demri, C. S. Jensen (Eds.), Proc. of the 15th Int. Symposium on Temporal Representation and Reasoning (TIME 2008), IEEE Computer Society, Monteéal, QC, Canada, 2008, pp. 3–14.

[28] V. Milea, F. Frasincar, U. Kaymak, Knowledge Engineering in a Temporal Semantic Web Context, in: D. Schwabe, F. Curbera, P. Dantzig (Eds.), Proc. of the 8th Int. Conf. on Web Engineering (ICWE 2008), IEEE, Yorktown Heights, NY, USA, 2008, pp. 65–74.

[29] U. Straccia, N. Lopes, G. Lukácsy, A. Polleres, A General Framework for Representing and Reasoning with Annotated Semantic Web Data, in: M. Fox, D. Poole (Eds.), Proc. of the 24th Nat. Conf. on Artificial Intelligence (AAAI 2010), AAAI Press, Atlanta, GA, USA, 2010, pp. 1437–1442.

[30] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of SPARQL, ACM Transactions on Database Systems 34 (3).

[31] M. Fitting, First-Order Logic and Automated Theorem Proving, 2nd Edition, Texts in Computer Science, Springer, 1996.

[32] H. B. Enderton, A Mathematical Introduction to Logic, 2nd Edition, Academic Press, 2001.

[33] P. Hayes, RDF Semantics, W3C Recommendation (February 10 2004).

[34] M. Schneider, OWL 2 Web Ontology Language: RDF-Based Semantics, W3C Recommendation (October 27 2009).

[35] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, OWL 2 Web Ontology Language: Profiles, W3C Recommendation (October 27 2009).

[36] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, 2nd Edition, Cambridge University Press, 2007.

[37] P. F. Patel-Schneider, B. Motik, OWL 2 Web Ontology Language: Mapping to RDF Graphs, W3C Recommendation (October 27 2009).

[38] B. Motik, P. F. Patel-Schneider, B. Cuenca Grau, OWL 2 Web Ontology Language: Direct Semantics, W3C Recommendation (October 27 2009).

[39] J. Chomicki, Temporal Query Languages: A Survey, in: D. M. Gabbay, H.-J. Ohlbach (Eds.), Proc. of the 1st Int. Conf. on Temporal Logic (ICTL '94), Vol. 827 of LNCS, Springer, Bonn, Germany, 1994, pp. 506–534.

[40] L. Vila, A Survey on Temporal Reasoning in Artificial Intelligence, AI Communications 7 (1) (1994) 4–28.

[41] J. F. Allen, Maintaining Knowledge about Temporal Intervals, Communications of the ACM 26 (11) (1983) 832–843.

[42] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, EQL-Lite: Effective First-Order Query Processing in Description Logics, in: M. M. Veloso (Ed.), Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJACI 2007), Hyderabad, India, 2007, pp. 274–279.

[43] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison Wesley, 1995.

[44] R. Ramakrishnan, D. Srivastava, S. Sudarshan, Rule Ordering in Bottom-Up Fixpoint Evaluation of Logic Programs, IEEE Transactions on Knowledge and Data Engineering 6 (4) (1994) 501–517.

[45] C. Gutiérrez, C. A. Hurtado, A. O. Mendelzon, Foundations of Semantic Web Databases, in: A. Deutsch (Ed.), Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2004), ACM, Paris, France, 2004, pp. 95–106.

[46] O. Udrea, D. Reforgiato Recupero, V. S. Subrahmanian, Annotated RDF, ACM Transaction on Computational Logic 11 (2).

[47] N. Lopes, A. Polleres, U. Straccia, A. Zimmermann, AnQL: SPARQLing Up Annotated RDFS, in: P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, B. Glimm (Eds.), Proc. of the 9th Int. Semantic Web Conf. (ISWC 2010), Vol. 6496 of LNCS, Springer, Shanghai, China, 2010, pp. 518–533.