Introducing Time into RDF

Claudio Gutierrez, Carlos A. Hurtado, and Alejandro Vaisman

Abstract—The Resource Description Framework (RDF) is a metadata model and language recommended by the W3C. This paper presents a framework to incorporate temporal reasoning into RDF, yielding temporal RDF graphs. We present a semantics for these kinds of graphs which includes the notion of temporal entailment and a syntax to incorporate this framework into standard RDF graphs, using the RDF vocabulary plus temporal labels. We give a characterization of temporal entailment in terms of RDF entailment and show that the former does not yield extra asymptotic complexity with respect to nontemporal RDF graphs. We also discuss temporal RDF graphs with anonymous timestamps, providing a theoretical framework for the study of temporal anonymity. Finally, we sketch a temporal query language for RDF, along with complexity results for query evaluation that show that the time dimension preserves the tractability of answers.

Index Terms—Data models, query languages, temporal databases.

1 INTRODUCTION

The Resource Description Framework (RDF) [21] is a metadata model and language recommended by the W3C for building an infrastructure of machine-readable semantics for the data on the Web, a long-term vision known as Semantic Web. In the RDF model, the universe to be modeled is a set of resources, essentially anything that can have a universal resource identifier, URI. The language to describe them is a set of properties, technically, binary predicates. Descriptions are statements in the subject-predicate-object structure. Both subject and object can be anonymous objects, known as blank nodes. In addition, the RDF specification includes a built-in vocabulary with a normative semantics (RDFS) [5]. This vocabulary deals with inheritance of classes and properties, as well as typing, among other features that allow the descriptions of concepts and relationships that can exist for a community of people and software agents, enabling knowledge sharing and reuse. The RDF specification can be seen as a graph where each subject-predicate-object triple is represented as a node-edge-node structure.

Although some studies exist about addressing changes in an ontology [19] or the need for temporal annotations on Web documents [26], to the best of our knowledge, the first formal approach to the problem of modeling and querying temporal information in RDF was [16]. In this paper, we develop that framework in its entire generality. Time is present in almost any Web application. Indeed, as pointed out by Abiteboul [1], the modeling of time is one of the key primitives needed in a query language for Web and semistructured data. Thus, there is a clear need for applying temporal database concepts to RDF to allow metadata navigation across time. The need for querying the history of metadata descriptions may arise in different applications, such as accessing different versions of an ontology, retrieving past info about Web sites, distributing updates of logs (e.g., CVS), and querying metadata about resources that are temporal in nature (e.g., stocks and news). We will motivate temporal RDF with an example, which will be used throughout the paper, that refers to the application of RDF data to the description of Web services.

Web services are software applications that interact using Web standards. Although Web services technology is rapidly gaining popularity, it still requires more human involvement than a user may want. Avoiding this will imply the ability of automatically discovering or invoking Web services. Semantic Web technology has been proposed for helping to solve this problem by means of ontologies of services that are used for representing a service profile (a mechanism for describing services offered by a Web site). These ontologies can be used by service-seeking agents. Nevertheless, ignoring the changes that can occur throughout the life cycle of the Web service may lead to several problems that will be discussed in the paper.

We will start with a Web service ontology introduced by Antoniou and Van Harmel [3], and then we will show how this initial ontology may pass through different states. Fig. 1 shows an RDF representation of an ontology for a Web service, denoted Sport News, offered by a sports network (ESPN). The Web site delivers up-to-date articles about sports. As input, the service receives a sports category and the customer’s credit card number; it returns the requested articles.

Let us suppose that, at a certain point in time (we will denote this time instant “2”), ESPN sold the rights on Sport News to another sports network, Fox Sports; thus, beginning at time “3,” the Web service is offered by the latter network. The new owners decided (at time instant “4”) to add a new service: They will deliver videos of the best plays of the week for all sport events covered by the network. Thus, several changes must be performed over the previous RDF graph, summarized as: 1) the name, phone, and Web page of the service provider must be replaced and 2) the new service must be added to the graph. This implies the addition of the following triples: (play of the week, type,
offered service), (play of the week, provided by, Fox Sports), (play of the week, output, video), (play of the week, input, customer card), and (video, type, parameter). The state of the graph after all these changes is depicted in Fig. 2, where we have shown in bold the changes with respect to the initial state.

Fig. 1 and Fig. 2 demonstrate that the impact of disregarding the time dimension is twofold: On the one hand, when a change occurs, a new document must be created (and the current document dropped). On the other hand, queries asking for past states of the metadata cannot be supported. For instance, we cannot ask for the services offered by ESPN at a certain point in time. Moreover, the ontology itself may change (for instance, new properties may be required to describe Web services, or another one may cease to be needed). In this paper, we will provide an in-depth discussion on temporal issues for RDF specifications.

1.1 Problem Statement: Introducing Time into RDF

Generally speaking, a temporal database is a repository of temporal information. Although temporal databases were initially studied for adding the time dimension to relational databases, as new data models emerged, temporal extensions to these models were also proposed (see Section 1.2). We next discuss the main issues that arise when extending RDF with temporal information.

1.1.1 Versioning versus Time Labeling

There are two mechanisms for adding the time dimension to nontemporal RDF graphs: labeling and versioning.
The former consists of labeling the elements subject to changes (i.e., triples). The latter is based on maintaining a snapshot of each state of the graph. For instance, each time a triple changes, a new version of the RDF graph is created, and the past state is stored somewhere. Although both models are equivalent, versioning appears to be not suitable for queries of the form: “all time instants where /C8 holds in the database.” There are at least two temporal dimensions to consider when dealing with temporal databases: valid and transaction times. Valid time is the time when data is valid in the modeled world; transaction time is the time when data is actually stored in the database. The versioning approach captures transaction time, while labeling is mostly used when representing valid time. The approach we present in this paper supports both time dimensions.

In summary, we believe that, for RDF data, labeling is better than versioning because 1) it preserves the spirit of the distributed and extensible nature of RDF, and 2) in scenarios where changes are frequent and only affecting a few elements of the document, creating a new physical version of the graph each time an update occurs may lead to large overheads when processing temporal queries that span multiple versions.

1.1.2 Time Points versus Time Intervals
We will work with the point-based temporal domain for defining our data model and query language, but we will encode time-points in intervals when possible, for the sake of clarity. We will consider time as a discrete, linearly ordered domain, as usual in virtually all temporal database applications. An ordered pair \([a, b]\) of time points, with \(a \leq b\), denotes the closed interval from \(a\) to \(b\). Fig. 3 shows a temporal RDF graph for our running example. The arcs in the graph are labeled with their interval of validity. For the sake of readability, we have omitted all edge labels related to the RDFS ontology (i.e., type, subClass, domain, and range).¹

For example, the interval \([4, Now]\) means that the triple (plays of the week, provided by, Fox Sports) is valid from time instant “4” to the current time. For the sake of clarity, no temporal labels over an edge means that the triple is valid in the interval \([0, Now]\). Also, note that the figure includes the triples telling that ESPN provided the Sport News service in the interval \([0, 2]\), along with the network’s information. Thus, no information is lost, and past states can be reconstructed. An anonymous node was also created, indicating that some network we do not know yet (“X”) provided the service “Play of the week” in the interval \([2, 3]\). We will study the impact of blank nodes in a temporal setting later in the paper.

1.1.3 Vocabulary for Temporal Labeling
Temporal labeling can be implemented within the RDF specification, making use of a simple additional vocabulary, as Fig. 4 shows. As an example, the graph at the left-hand side of the figure represents the addition of temporal information to the triple (Fox Sports, Web page, www.foxsports.com). There is a blank node connected to the components of the triple, in a sort of “temporal reification” scheme (using the vocabulary tsubj, tpred, and tobj). The remainder of the graph are statements about the timestamps at which the triple was valid. As we adopted the point-based, discrete, and linearly ordered temporal domain, the left and right-hand sides of Fig. 4 are equivalent. We will use both representations indistinctly. Moreover, we allow moving between intervals and time instants as follows: The instants depicted in Fig. 4a can be encoded in an interval as shown in Fig. 4b. Both alternatives will be used in the query language.

¹ Note that the standard graph(ical) representation of an RDF graph is not the most faithful to convey the idea of statements (triples) being labeled by a temporal element. Technically, temporal labels should be attached to a whole subgraph \(u \leadsto v\), and not only to an arc.
1.1.4 Temporal Entailment
An RDF graph can be regarded as a knowledge base from which new knowledge, i.e., other graphs, may be entailed. Entailment in a temporal setting is slightly more involved in the RDF case than in the standard database case. In principle, one may be tempted to define the semantics as in temporal relational databases, i.e., defining the temporal database as the union of all of its snapshots. (A snapshot at time \( t \) of a temporal RDF graph \( G \) is the corresponding subgraph formed by triples labeled by an instant \( t \).) Blank nodes impose some constraints to this naive approach. For example, each of the snapshots of Fig. 5b entails the corresponding snapshots of Fig. 5a. However, the whole graph of Fig. 5a cannot be entailed by the graph of Fig. 5b. Indeed, the graph of Fig. 5a states the fact commented above, that there is an anonymous object \( X \) such that the triple \((\text{plays of the week}, \text{provided by}, X)\) is valid at times “2” and “3,” which is not the case for the other graph.

1.1.5 Temporal Query Language
Regarding query languages in temporal databases, basically two choices for defining the temporal domains exist: the point-based and the interval-based temporal domains, yielding different query languages [25], [4]. In the point-based approach, temporal variables in query languages refer to individual time instants, while in the interval-based domain, variables in the queries range over intervals, making queries more complicated and unnatural. Anyway, one can move easily between these two domains.

1.2 Related Work
The RDF model was introduced in 1998 by the World Wide Web Consortium (W3C) [21]. Formal work in RDF from a database point of view includes the study of formal aspects of RDF data and query languages [14], [15], [27], considering RDF features like the entailment, presence of blank nodes, reification, premises in queries, and the RDFS vocabulary with predefined semantics. Several languages for querying RDF data have been proposed and implemented. Some of them are in the lines of traditional database query languages (e.g., SQL and OQL), while others are based on logic and rule languages. Good surveys are [17], [20]. To the best of our knowledge, there is still no formal study of temporality issues in RDF graphs and RDF query languages.

Temporal database management has been extensively studied, including data models, mostly based on the relational model and query languages [24], leading to the TSQL2 language [23]. Beyond the relational model, managing historical semistructured data was first proposed by Chawathe et al. [9], who extended the Object Exchange Model (OEM) with the ability to represent updates and to keep track of them by means of “deltas.” Later, Dyreson et al. [12] allowed annotations on the edges of the database graph. In the XML world, Amagasa et al. [2] introduced a temporal data model based on XPath for the first time. Dyreson [11] proposed an extension of XPath with support for transaction time by means of the addition of several temporal axes for specifying temporal directions, focusing on document versioning over the Web in the absence of explicit time stamps. Chien et al. [10] proposed update and versioning schemes for XML through an edit-based schema in which the most current version of the document is maintained and reverse edit scripts allow moving backward in time.

Gao and Snodgrass [13] introduced \( \tau \)XQuery, an extension to XQuery supporting valid time while maintaining the data model unchanged. Rizzolo et al. [22] proposed a
temporal model for XML, a temporal extension to XPath, and a novel indexing strategy for temporal XML documents. Like in our approach, they use labeling and a point-based temporal domain and query language. Regarding temporal extensions to RDF, Visser et al. [26] proposed a temporal reasoning framework for the Semantic Web, which has been applied in BUSTER, an ontology-based prototype developed at the University of Bremen, supporting the so-called concept@location in time type of query. Also, Bry et al. [8], [7], [6] have stated the need of providing query languages and models for the Web with temporal reasoning capabilities.

1.3 Contributions
In this paper, we present a framework to incorporate temporal reasoning into RDF, yielding temporal RDF graphs. In particular, we present the following contributions:

1. a semantics for temporal RDF graphs in terms of the semantics of nontemporal RDF and RDFS graphs,
2. a study of the properties of temporal RDF graphs and the interplay between timestamp and snapshot semantics in temporal RDF graphs,
3. a syntax to incorporate this framework into standard RDF graphs where; the syntax uses the standard RDF vocabulary plus temporal labels,
4. complexity bounds which show that entailment in temporal RDF graphs does not yield extra asymptotic time complexity with respect to standard RDF graphs,
5. a study of temporal RDF graphs with anonymous timestamps, i.e., graphs containing triples that we know are valid in some unknown time, and
6. a sketch of a temporal query language for RDF and complexity results for query evaluation.

An extended abstract with the ideas underlying this framework appeared in the Proceedings of the European Semantic Web Conference 2005 [16]. We have substantially developed, reformulated, and done a thorough revision of the material there. The new contributions include: 1) A new and more realistic running example based on a Web services ontology, for motivating our work. 2) Proofs for all theorems, propositions, and lemmas. Moreover, we also present new results; in particular, the theorem that decides whether temporal entailment for temporal RDF graphs is NP-complete. 3) A study of the problem of temporal RDF graphs with anonymous time (denoted general temporal RDF graphs), showing that deciding entailment for this class of graphs is also NP-complete.

In summary, we address in this paper the management of time in RDF, studying the inclusion of time for objects, anonymous objects, and statements, as well as developing the topic of anonymous time, an issue, that to the best of our knowledge, has not been addressed before.

1.4 Outline
The remainder of the paper is organized as follows: Section 2 presents preliminary notation related to RDF and RDFS from previous work [15]. Section 3 introduces temporal RDF graphs, studies their semantics and RDF syntax. Section 4 extends temporal graphs to handle anonymous time. Section 6 discusses the query language for temporal RDF. Finally, in Section 7, we conclude and outline some prospects for future work.

2 RDF PRELIMINARIES
In this section, we present a streamlined formalization of the RDF model following W3C documents [21], [18], [5] along the line of [15].

2.1 RDF Graphs
Assume there is an infinite set \( U \) (RDF URI references), an infinite set \( B = \{ N_j : j \in \mathbb{N} \} \) (Blank nodes), and an infinite set \( L \) (RDF literals). A triple \( (v_1, v_2, v_3) \in (U \cup B) \times U \times (U \cup B \cup U \cup L) \) is called an RDF triple. In such a triple, \( v_1 \) is the subject, \( v_2 \) the predicate, and \( v_3 \) the object. We often denote by UBL the union of the sets \( U, B, \) and \( L \).

An RDF graph (just graph from now on) is a set of RDF triples. A subgraph is a subset of an RDF graph. The universe of a graph \( G \), \( \text{universe}(G) \), is the set of elements of UBL that occur in the triples of \( G \). The vocabulary of \( G \) is the set \( \text{universe}(G) \cap (U \cup L) \), i.e., the nonblank elements of the universe. We will use letters \( N, X, Y, \ldots \) to denote blank node, and \( a, b, c, \ldots \) for URLs and literals. A graph is ground if it has no blank nodes. Graphically, we represent RDF graphs as follows: Each triple \( (a, b, c) \) is represented by the labeled graph \( a \rightarrow b \rightarrow c \). Note that the set of arc labels can have a nonempty intersection with the set of node labels.

A map is a function \( \mu : \text{UBL} \rightarrow \text{UBL} \) preserving URIs and literals, i.e., \( \mu(u) = u \) and \( \mu(l) = l \) for all \( u \in U \) and \( l \in L \). Given a graph \( G \), we define \( \mu(G) \) as the set of all \( (\mu(s), \mu(p), \mu(o)) \) such that \( (s, p, o) \in G \). A map \( \mu \) is consistent with \( G \) if \( \mu(G) \) is an RDF graph, i.e., if \( s \) is the subject of a triple, then \( \mu(s) \in U \cup B \), and if \( p \) is the predicate of a triple, then \( \mu(p) \in U \). In this case, we say that the graph \( \mu(G) \) is an instance of the graph \( G \). An instance of \( G \) is proper if \( \mu(G) \) has fewer blank nodes than \( G \). This means that either \( \mu \) sends a blank node to a URI or a literal, or identifies two blank nodes.

Two graphs \( G_1, G_2 \) are isomorphic, denoted \( G_1 \cong G_2 \), if there are maps \( \mu_1, \mu_2 \) such that \( \mu_1(G_1) = G_2 \) and \( \mu_2(G_2) = G_1 \).

We define two operations on graphs. The union of \( G_1, G_2 \), denoted \( G_1 \cup G_2 \), is the set theoretical union of their sets of triples. The merge of \( G_1, G_2 \), denoted \( G_1 + G_2 \), is the union \( G_1 \cup G_2' \), where \( G_2' \) is an isomorphic copy of \( G_2 \) whose set of blank nodes is disjoint with that of \( G_1 \). Note that \( G_1 + G_2 \) is unique up to isomorphism.

2.2 RDFS Vocabulary
There is a set of reserved words defined in the RDF vocabulary description language, RDF Schema [5]—just rdfs-vocabulary for us—that may be used to describe properties like attributes of resources (traditional attribute-value pairs), and also to represent relationships between resources. This vocabulary defines classes and properties that may be used for describing groups of related resources and relationships between resources.\(^2\) Classes are sets of resources. Elements of a class are known as instances.

\(^2\) We omit in this paper vocabulary for which there is no normative semantics, namely, those words intended to describe lists, collections, some variations on these, as well as vocabulary to help document and describe other functionalities. The complete vocabulary can be consulted in [5].
In this section, we define the notion of temporal RDF at a
3.1 Basic Definitions

In this paper, we extend RDF graphs by allowing temporal
of that class. To state that a resource is an instance of a class,
the property rdf:type may be used. The following are the
most important classes (in brackets, the name we will use in
this paper): rdfs:Resource [res], rdfs:Class [class], rdfs:Lit-
xmlLiteral, rdfs:Datatype [datatype], rdfs:XMLLiteral
[xmLiteral], and rdfs:Property [property]. Properties are
binary relations between subject resources and object
resources. The built-in properties are: rdfs: range [range],
rdfs:domain [domain], rdfs:type [type], rdfs: subClassOf
[subClass0f], and rdfs:subPropertyOf [subProperty0f].

2.3 Entailment

There is a notion of entailment between RDF graphs (see
[18]), which will be denoted by $\models$. For our purposes, it is
sufficient to have a working characterization of this notion.
A closure of a graph $G$ is a maximal set of triples $G'$ over
universe($G$) plus the RDFS vocabulary such that $G \subseteq G'$
and $G \models G'$. For our purposes, it is

Theorem 1 (see [18], [15]). \(G_1 \models G_2\) if and only if there is a map
from \(G_1\) to a closure of \(G_2\).

Lemma 1. \(G_1 \models G_2\) if and only if, for each ground instance
\(\mu_1(G_1)\), there is a ground instance \(\mu_2(G_2)\) such that
\(\mu_1(G_1) \models \mu_2(G_2)\).

Proof. Let \(\mu_1(G_1)\) be a ground instance. If \(G_1 \models G_2\), then
there is a map \(\phi : G_2 \rightarrow \text{cl}(G_1)\) (by cl, we denote any
closure). Also, we have a map \(\mu_1 : \text{cl}(G_1) \rightarrow \text{cl}(G_1)\)

Define \(\mu_2 : G_2 \rightarrow \text{cl}(\text{cl}(G_1))\) defined on the blank nodes
by \(\mu_2(X) = \mu_1(\phi(X))\). Then, clearly \(\mu_2(G_2) \subseteq \mu_1(\text{cl}(G_1))\),
that is, \(\mu_1(G_1) \models \mu_2(G_2)\).

Conversely, consider the map \(\mu_1\), sending blank
to fresh constants \(c_X\). Then, by hypothesis, there is
\(\mu_2\) with \(\mu_2(G_2)\) ground and \(\mu_1(G_1) \models \mu_2(G_2)\), i.e.,
\(\mu_2(G_2) \subseteq \mu_1(\text{cl}(G_1))\). Consider the map \(\nu : G_2 \rightarrow \text{cl}(G_1)\)
defined as: \(\nu(Y) = X\) if \(\nu_2(X) = c_X\), else \(\nu(X) = \nu_2(X)\).

This map proves that \(G_1 \models G_2\). \(\square\)

3 TEMPORAL RDF GRAPHS

In this paper, we extend RDF graphs by allowing temporal
elements to label triples. A temporal label is a temporal
element labeling a triple \((a, b, c)\). It represents the time
period when the triple was valid in the real world. Without
loss of generality, we will assume that temporal elements are
intervals.

3.1 Basic Definitions

In this section, we define the notion of temporal RDF at a
conceptual level.

Definition 1 (Temporal Graph).

1. A temporal triple is an RDF triple \((a, b, c)\) with a
temporal label \(t\) (a natural number). We will use the
notation \((a, b, c)[t]\). The expression \((a, b, c)[t_1, t_2]\) is a
notation for \(\{(a, b, c)[t] \mid t_1 \leq t \leq t_2\}\).
2. A temporal graph is a set of temporal triples.

There are certain operations on temporal graphs which
are useful for transforming them into standard RDF graphs
and vice versa, i.e., moving between both worlds. Given a
temporal graph \(G\) and a time \(t\), define the slice of \(G\) at \(t\),
denoted \(G|_t\), as the subgraph of \(G\) consisting of all temporal
triples of \(G\) with temporal label \(t\). We introduce also an
operator taking temporal graphs and returning standard
RDF graphs, the underlying graph of \(G\), denoted \(u(G)\), and
defined as \(u(G) = \{(a, b, c)[t] \mid \text{for some } t\}\).

Conversely, for an RDF graph \(H\) and a time \(t\), define \(H^t\) as the
temporalization of all its triples by a temporal mark \(t\), that
is, \(H^t = \{(a, b, c)[t] \mid (a, b, c) \in H\}\).

There is a particularly important operation, given a
temporal graph \(G\) and a time \(t\): the snapshot of \(G\) at \(t\), which
is defined as the graph \(G(t) = u(G)|_t\).

Usually, for a temporal graph \(G\), we will apply the same
notions used for standard RDF graphs, for example, we will
say "\(G\) is ground" meaning that \(u(G)\) is ground, write \(\mu(G)\)
for \(\{(\mu(a), \mu(b), \mu(c))[t] : (a, b, c)[t] \in G\}\), and so on.

The above definitions give the following elementary
consequences about the relationship between RDF graphs
and temporal RDF graphs:

Lemma 2. Let \(G\) be a temporal RDF graph, and \(H\) be a standard
RDF graph. Then,

1. \((G|_t)|_t = G|_t\) and, if \(G \subseteq G'\), then \(G|_t \subseteq G'|_t\).
2. \(G = \bigcup G|_t\) and \(u(G) = \bigcup G(t)\), and
3. \(H^t = H\) and \((G(t))^t = G|_t\).

Several issues on the definition of temporal RDF graph
are in order:

- Recall that we are using a temporal model where an
interval \([a, b]\) is of the form \([a, a + 1, \ldots, b]\) for a given
unit of time that we will assume to be universal in
this paper. The natural way to approach the issue
about the granularity of time is to specify, together
with the temporal mark, the unit of time it
represents. All the results given here extend without
difficulties to this setting.

- Temporal triples do not belong to the RDF syntax. In
the next section, we introduce an RDF-complying
syntax for temporal triples, using a small temporal
vocabulary.

- Due to the extensible nature of the RDF model, it is
possible to include the source of a temporal statement
(i.e., who is the author of the temporal statement),
and other properties that apply. Although our model
(see next section) allows this, we will not study the
semantic consequences of this extra information in
this paper, but rather stay in the classic setting of
temporal models.

3.2 Semantics

In what follows, we present the semantics for the notion
of entailment for temporal graphs based on the corresponding
notion for RDF graphs.

Definition 2 (Temporal Entailment). Let \(G_1, G_2\) be RDF
temporal graphs.

1. For ground temporal RDF graphs \(G_1, G_2\), define
\(G_1 \models G_2\) if and only if \(G_1(t) \models G_2(t)\) for each
2. For arbitrary temporal RDF graphs, define \(G_1 \models G_2\) if and only if, for every ground instance \(\mu_1(G_1)\),
there exists a ground instance \(\mu_2(G_2)\) such that
\(\mu_1(G_1) \models \mu_2(G_2)\).
As an example, let $G_1$ be the temporal graph $\{(a, b, X)[3], (a, b, Y)[4]\}$, and let $G_2$ be the temporal graph $\{(a, b, Y)[3], (a, b, X)[4]\}$. Notice that $G_1$ and $G_2$ are the temporal graphs that represent the RDF graphs of Fig. 5a and Fig. 5b, respectively. We have that $G_1 \models G_2$. However, it is not the case that $G_2 \models G_1$; just consider the ground graph $\mu_2(G_2)$, where $\mu_2(Y) = d$, $\mu_2(X) = e$, and $d \neq e$.

Note that the definition for ground graphs resembles classical temporal definitions:

**Proposition 1.** Let $G_1, G_2$ be temporal graphs. Then, $G_1 \models G_2$ implies that, for each time $t$, the entailment holds for each slice, i.e., $G_1|_{t} \models G_2|_{t}$ for all $t$. The converse is true for ground graphs.

**Proof.** Just note that, from item 1 of the previous lemma, it follows $(G_1|_{t})(t) = G(t)$. □

In fact, the problems for general graphs are introduced by blank nodes. For example, $G_1(t) \models G_2(t)$ for all $t$ does not imply $G_1 \models G_2$ (see Fig. 5). We have the following issues:

- A blank node represents the same (unnamed) resource throughout the time range, rather than a sequence of different resources. This makes the behavior of temporal marks in Temporal RDF different from the classical setting. Temporal marks here—contrary to temporal XML, for example—are not only a relation among fixed objects, but also among time-varying objects, the blank nodes. See an example in Fig. 5.

- The notion of entailment for temporal RDF needs a basic arithmetic of intervals in order to combine the notion of temporality and deductive properties. For example, if we have $(a, s, c, d)[2], (c, s, a, d)[2]$, then we should be able to derive $(a, s, c, d)[2]$, but not $(a, s, c, d)[3]$. We can show that the decision problem of entailment for temporal graphs is NP-complete, thus maintaining the complexity of the nonontemporal case.

**Theorem 2.** Given two temporal graphs $G_1, G_2$, the problem of deciding if $G_1 \models G_2$ is NP-complete.

**Proof.** The problem is NP-hard because one can code a standard RDF instance of the problem using the fact that, for RDF graphs $G, H, G \models H$ if and only if $G' \models H'$.

Let us check the membership in NP for graph graphs first. Clearly, there are no more than $\max\{|G_1|, |G_2|\}$ different temporal elements. Hence, a witness for $G_1 \models G_2$ is the set $\{w_t\}$, where each $w_t$ is a witness for the entailment for $t$ (recall Definition 2, item 2), known to be in NP.

For the general case, just observe that, considering a set of constants disjoint from universe($G_1$) $\cup$ universe($G_2$) and defining $\mu_1$ a 1-1 map that sends the blank nodes to those constants, one avoids the checking for every ground $\mu_1(G_1)$. □

Another form of viewing the complexity of temporal entailment, which additionally gives a procedure to compute it, is obtained by using the notion of closure as in the nonontemporal case. For computational purposes, we will show that it is enough to compute a subset of the closure which we will call the slice closure.

**Definition 3.** Let $G$ be a temporal graph.

1. The closure $G$, denoted $\text{tlc}(G)$, is a maximal set of temporal triples $G'$ over universe($G$) plus the RDF vocabulary such that $G$ contains $G'$ and is equivalent to it, that is, $G \equiv G'$ and $G \models G'$.

2. The slice closure of $G$, denoted $\text{scl}(G)$, is a temporal graph defined by the expression $\bigcup\{\text{cl}(G(t))\}$, where $\text{cl}(G(t))$ is any closure of the RDF graph $G(t)$.

As in the nonontemporal case, there may exist several different closures and several different slice closures for a graph. The computation of the slice closure reduces to computing a nontemporal closure for each of the snapshot graphs (we refer the reader to [15] for further details on the closure of nontemporal graphs). The following example shows that the closure and slice closure of a temporal graph are not necessarily the same. Let $G$ be the following graph:

\[
\{(a, b, c)[t1], (a, b, c)[t2], (a, b, Y)[t2]\}.
\]

We have that $\text{tlc}(G)$ is the graph

\[
\{(a, b, c)[t1], (a, b, Y)[t1], (a, b, c)[t2], (a, b, Y)[t2]\}
\]

and $\text{scl}(G)$ is the graph

\[
\{(a, b, c)[t1], (a, b, c)[t2], (a, b, Y)[t2]\}.
\]

**Proposition 2.** Let $G, G_1, G_2$ be temporal RDF graphs. Then,

1. $G \subseteq \text{scl}(G)$ and $G \models \text{scl}(G)$.
2. $\text{scl}(G) \subseteq \text{tlc}(G)$ for some closure $\text{tlc}(G)$.
3. $\text{tlc}(G)$ is polynomial in the size of $G$.

**Proof.**

1. From the identity $G = \bigcup_i G_i = \bigcup_i G(t)_i^1$, it follows that $G \subseteq \bigcup_i \text{cl}(G(t)_i)^1$. Now, let $\mu(G)$ be a ground graph, then $\mu(\text{scl}(G))$ is also a ground graph because $G$ and $\text{scl}(G)$ have the same blank nodes. Now, it can be easily verified that, for each $t$, $\mu(G(t)) \models \mu(\text{cl}(G(t)))$. Then, from Proposition 1, it follows that $\mu(\bigcup_i \text{cl}(G(t))^1) \models \mu(\bigcup_i (\text{scl}(G))^1)$, that is, $\mu(G) \models \text{scl}(G)$.

2. The proposition follows from $G \models \text{scl}(G)$ and the fact that $\text{scl}(G)$ contains only triples over universe($G$).

3. The closure $\text{tlc}(G)$ adds to $G$ triples over the fixed vocabulary universe($G$); therefore, $\text{tlc}(G)$ cannot have more than $|\text{universe}(G)|^3$ triples. □

The next theorem shows that testing temporal entailment reduces to first computing the slice closure, and then finding mappings between two temporal graphs. The latter is similar to finding mappings between nonontemporal graphs [15].

**Theorem 3.** Let $G_1, G_2$ be temporal RDF graphs. Then, $G_1 \models G_2$ if and only if there is a map from $G_2$ to $\text{scl}(G_1)$.

**Proof.** (If) Let $\mu$ be the map, and let $\mu_1(G_1)$ be a ground instance. Let $\mu_2 = \mu \circ \mu_1$. Then, it is easily verified that $\mu_2$ is a ground instance of $G_2$ and that, for each $t$, $\mu_2(G_2(t)) \subseteq \mu_1(\text{cl}(G_1(t)))(t)$. Then, for each $t$,

\[
\mu_1(G_1(t)) \models \mu_2(G_2(t)).
\]
Then, from Proposition 1, it follows that \( \mu_1(G_1) \models \mu_2(G_2) \).

(OnlyIf) Consider the ground instance \( \mu_1(G_1) \) that maps each variable \( X \) to a different constant \( c_X \). Consider the ground instance \( \mu_2(G_2) \) such that \( \forall t : \mu_1(G_1)(t) \models \mu_2(G_2)(t) \). Thus, for each \( t_i \)

\[
\mu_2(G_2)(t_i) \subseteq \text{cl}(\mu_1(G_1)(t_i)).
\]

Thus,

\[
\mu_2(G_2) \subseteq \bigcup_t (\text{cl}(\mu_1(G_1)(t)))^2.
\]

Now, we rename in both mappings \( \mu_1 \) and \( \mu_2 \) the constants of the form \( c_X \) by \( X \) obtaining that \( \mu_2 \) is a mapping from \( G_2 \) to \( \text{sc}l(G_1) \). \( \square \)

Note. It can be shown—it escapes the scope of this paper—that the notions of lean graph and core—fundamentally to define notions of normalization of RDF data—can be extended without difficulty to this temporal setting. (Compare discussions in [15].)

3.3 Syntax for Temporal Graphs

In this section, we will show that the whole framework presented can be implemented using the standard syntax of RDF.

**Definition 4 (Temporal Vocabulary).** The temporal vocabulary is the following: temporal (abbreviated as tpl), instant, interval, initial, and final, all of type property, and now of type plain literal. The range of instant, initial, and final is the set of natural numbers.

We will use the following notation shortcuts: reif\((a,b,c,X)\): the set of triples \((X,\text{tsubj},a)\), \((X,\text{tpred},b)\), and \((X,\text{tojb},c)\), a kind of “temporal reification” of \((a,b,c)\).

**Definition 5 (Temporal Triples and Graphs).** Temporal triples are the following graphs using the temporal vocabulary.

- \((a,b,c),\text{reif}(a,b,c,X),(X,\text{tpl},Y), (Y,\text{instant},n)\) and where \( n \) is a natural number; we will summarize this as \((a,b,c)[X,Y,n]\);
- \((a,b,c),\text{reif}(a,b,c,X),(X,\text{tpl},Y), (Y,\text{interval},Z), (Z,\text{initial},I), (Z,\text{final},F)\), where \( I, F \) are natural numbers; we will summarize this as \((a,b,c)[X,Y,I,F]\).

A temporal graph (in this syntactic setting) will be defined as a merge of a set of temporal triples.

**Definition 6.** Let \( G \) be a temporal graph and let \( H \) be an RDF graph with temporal vocabulary. Define \( G^* \) as the RDF graph \( \{(a,b,c)[X,Y,t] | (a,b,c)[t] \in G\} \), where \( X_t, Y_t \) are fresh blank variables, different for each \( t \).

Conversely, define \( H^* \) as the temporal graph \( \{(a,b,c)[t] | \exists X \exists Y (a,b,c)[X,Y,t] \in H\} \).

In what follows, for simplicity, we will assume that all RDF graphs with temporal vocabulary do not use interval for each triple \((a,b,c)[X,Y,t]\), and the blanks \( X, Y \) are fresh, that is, there is no clash between blank nodes of the temporal description and others that occur in the graph.

Formally,

**Definition 7.** An RDF graph \( H \) with temporal vocabulary is normal if and only if \((H^*)^* \) is equivalent to \( H \).

**Definition 8.** Let \( H_1, H_2 \) be normal RDF graphs with temporal vocabulary. Then, we define entailment for RDF graphs with temporal vocabulary, denoted \( \models_T \), by: \( H_1 \models_T H_2 \) if and only if \((H_1)^*_r \models (H_2)^*_r\).

**Theorem 4.** Let \( H_1, H_2 \) be normal RDF graphs with temporal vocabulary. Then, \( H_1 \models_T H_2 \) if and only if \((\text{sc}l((H_1)_r))^* \models (H_2)^*_r\).

**Proof.** The statement follows from Theorem 3. By definition, \((H_1)_r \models (H_2)_r \) means \((H_1)_r \models (H_2)_r \), which, in turn, by Theorem 3, is equivalent to the existence of a map \( \mu \) from \((H_2)_r \) to \((H_1)_r \). It is not difficult to check that then there is also a map from \((H_2)^*_r \) to \((\text{sc}l((H_1)_r))^* \) (the same \( \mu \) extended in the obvious way to the fresh blanks occurring in the temporal descriptions). Finally, just recall that \((H_1)^*_r = H\). \( \square \)

There are two aspects which the notion of normal temporal graphs of Definition 7 does not cover. First, cases where there are clashes of blanks nodes occurring in the temporal part of the graph (the blanks occurring in \((a,b,c)[X,Y,n]\)). Second, the equivalence between the interval and the point version of the labels. Both issues can be treated in the general case by adding syntactic rules. We will not treat them in this paper.

4 ANONYMOUS TIME

All the framework presented until now follows the classical assumption that timestamps (i.e., temporal labels) are constants. In this section, we will show that this restriction is not necessary, and one can allow variables (anonymous timestamps). We study temporal graphs with anonymous timestamps, that is, graphs which contain triples of the form \((a,b,c) : [X]\), where \( X \) is an anonymous timestamp, stating that the triple \((a,b,c)\) is valid in some unknown time. We refer to temporal graphs with constant or anonymous timestamps as general temporal graphs.

Anonymous time may help in the specification of triples without temporal labels, which is a way of specifying incomplete temporal information. As an example, anonymous timestamps can be used to state that a set of triples occurred at the same time, even though their valid time is unknown. In addition, a standard RDF graph can be made temporal by means of anonymous timestamps and, thus, modeled as temporal graphs.

Fig. 6 shows an excerpt of our running example, assuming we do not know the exact instants when ESPN stopped offering Sport News and Fox Sports started offering this service. Temporal labels \([0,T]\) and \([T, \text{Now}]\), respectively, allow expressing the former situation.

4.1 Definitions and Semantics

Let \( T \) be the set of anonymous timestamps and \( N \) be the set of timestamps (natural numbers). The set of anonymous timestamps and blank nodes are disjoint; in fact, they belong to different frameworks: time labels and triples.
Let \( \text{Definition 10.} \) anonymous timestamps.

A general temporal graph is t-ground if it does not contain temporal triples \( \langle a, b, c \rangle \) and given a general temporal graph \( G \) and \( t \)-labels.

Indeed, the t-ground graph \( (a, b, c) \) does not entail any t-ground graph of \( (a, b, c) \).

Notice that Proposition 1 does not hold for general temporal graphs. For example, let \( G \) be the graph \( \{ (a, b, c)[t_1], (c, d, e)[t_1], (a, b, c)[T_2] \} \).

We have that \( G \models_{\text{gen}} \{ (c, d, e)[T_2] \} \); however, it is not the case that \( G(T_1) \models_{\text{gen}} \{ (c, d, e)[T_2] \} \).

Next, we show that entailment of general temporal graphs reduces to closure computation in a similar fashion to temporal graphs, that is, we present an extended version of Theorem 3 for general temporal graphs. First, we define the slice closure of a general temporal graph \( G \), \( \text{scl}(G) \), as \( \bigcup_{l \in (T \cup N)} \text{scl}(G(l)) \).

**Theorem 5.** Let \( G_1, G_2 \) be general temporal RDF graphs. Then, \( G_1 \models_{\text{gen}} G_2 \) if and only if there is a t-map \( v \) and a map \( \mu \) such that \( \mu(v(G_2)) \subseteq \text{scl}(G_1) \).

**Proof.** (If) Consider a t-ground graph \( \gamma_1(G_1) \) of \( G_1 \). Now, let \( \gamma_2 = v \circ \gamma_1 \). We have that \( \gamma_2(G_2) \) is a t-ground graph of \( G_2 \) (because it maps all the anonymous timestamps to timestamps). Also, it can be easily verified that \( \mu(\gamma_2(G_2)) \subseteq \gamma_1(\text{scl}(G_1)) \). From Theorem 3, it follows that \( \gamma_1(G_1) \models_{\text{gen}} \gamma_2(G_2) \).

Now, from Definition 12, we obtain \( G_1 \models_{\text{gen}} G_2 \). (Only If) Then, from Definition 12, it follows that there is a t-ground graph \( v(G_2) \) such that \( \rho(G_1) \models_{\text{gen}} v(G_2) \). Then, from Theorem 3, it follows that there is a mapping \( \mu \) such that \( \mu(v(G_2)) \subseteq \text{scl}(G_1) \).

From Theorem 5, it follows that the testing entailment of general temporal graphs reduces to the following steps: 1) compute the slice closure \( \text{scl}(G_1) \), and 2) find a pair of mappings \( v \) and \( \mu \) from \( G_2 \) to \( \text{scl}(G_1) \). Step 1 reduces to computing the RDF closure of all the snapshots of \( G' \). The complexity of step 2 is not different than finding a mapping between two standard RDF graphs.

**Theorem 6.** Given two general temporal graphs \( G_1, G_2 \), the problem of deciding if \( G_1 \models_{\text{gen}} G_2 \) is NP-complete.
The statement follows from Theorem 8 by a similar built-in predicates such as \(<; >;\) graph treatment of time. As an example, the extended temporal timestamps and temporal variables. By incorporating them complex predicates) to model richer time domains using extension of Theorem 4.

5.1 The Query Language by Example

We left for future work the incorporation of arithmetic built-in predicates such as \(<; >;=;\) etc. (or even more complex predicates) to model richer time domains using timestamps and temporal variables. By incorporating them into the temporal RDF framework, we may support a richer treatment of time. As an example, the extended temporal graph

\[(a, s, c, b)\[T_1], t_1 < T_1, T_1 < t_2\]

states that the triple \((a, s, c, b)\) holds in a particular time inside the interval whose limits are \(t_1\) and \(t_2\), but we do not know the exact valid time of the triple.

Furthermore, built-in predicates over anonymous time yield a further notion of entailment. An example may be able to test the entailment of the graph \((a, s, c)\[T_2]\) from the extended temporal graph

\[(a, s, c, b)\[T_1], (b, s, c)\[t_1, t_2], t_1 < T_1, T_1 < t_2\]

This additional expressiveness can be handled by interpreting the arithmetic predicates as constraints to the t-ground graphs of the extended temporal graphs.

5 Query Language

In this section, we present a query language for temporal RDF graphs, along with its semantics. We also present a brief study of the complexity of query processing.

5.1 The Query Language by Example

We will give the flavor of the query language using our running example, the database of Fig. 3. Let us begin with a simple query: “Find the service providers who have offered a Web service between time instants 0 and 2, and return them qualified by early providers.” This query can be expressed as:

\((?X, \text{type, early provider}) \leftarrow (?X, \text{type, service provider})[?T],
(?S, \text{provided by, } ?X)[?T], 0 \leq ?T, ?T \leq 2\).

This example query illustrates the need of a built-in arithmetic language to reason about time and intervals. Another important observation is that temporal queries may output nontemporal RDF graphs, as it happens with the previous query. For the query asking for a snapshot of the graph at time 2, we have:


Now, consider the query “Find the services providers, along with the Web services they have offered, and the time instants when this occurred.” We express it as the following point-based query:

\((?X, \text{has provided, } ?Y)[?T] \leftarrow (?X, \text{provided by, } ?Y)[?T].

Next, we give examples of queries that use temporal triples with intervals. The previous query can be adapted as follows to capture time intervals:

\((?X, \text{has provided, } ?Y)[?T_1, ?T_2] \leftarrow
(?X, \text{provided by, } ?Y)[?T_1, ?T_2].

Observe that this query returns a set of intervals. In order to retrieve maximal intervals, we need a more subtle query since maximal intervals are not generated by the temporal rules given. For the query “Compute the maximal interval when the triple \((a, b, c)\) holds,” we need aggregate operators \(\text{MAX}\) and \(\text{MIN}\).

\((a, b, c) : {?T_1, ?T_2} \leftarrow (a, b, c)[?T_1, ?T_2],
?T_1 = \text{MIN}(?T_1), ?T_2 = \text{MAX}(?T_2).

For a query asking for “Service providers that have offered Web services for more than four consecutive periods (timestamps) and the maximal number of such consecutive periods,” we have:

\((?X, \text{interval, } t_f - t_i) \leftarrow (?X, \text{provided by, } ?X)
\|t_i, t_f\|, t_f - t_i > 4.

Here, the notation \(|t_i, t_f|\) stands for the fact that \(t_i\) and \(t_f\) match with the maximal interval for the corresponding triple computed with the query given above.

5.2 Semantics and Complexity

The temporal query language we present in this section extends the conjunctive fragment of RDF query languages formalized by Gutierrez et al. [15].

Let \(V\) be a set of variables (disjoint from UBLT). Individual variables will be denoted \(?X, ?Y, ?Z,\) etc. There is also a set of temporal variables \(V_t \subseteq V\).

A query is a temporal tableau, which is a pair \((H, B)\), where \(H\) and \(B\) are temporal RDF graphs with some elements of UBL replaced by variables in \(V\) and with some elements of \(T\) replaced with variables in \(V_t; B\) has no blank nodes and all the variables in \(H\) occur also in \(B\). The set \(A\) has the usual arithmetic built-in predicates such as \(<; >;=;\), over elements in \(V_t\) and \(T\).

We adopt the usual notion of \(\text{safe rule}\) from Datalog to prevent operations on infinite predicates. A rule is \(\text{safe}\) if all its variables are limited. A variable is \(\text{limited}\) if one of the following holds: A variable appears as an argument in a non-built-in predicate of the body; the variable \(X\) appears in a subgoal \(X = t\) (or \(t = X)\), where \(t\) is a constant in \(T\) or the variable \(X\) appears in a subgoal \(X = Y\) (or \(Y = X)\), where \(Y\) is limited.
The semantics are similar to the non-temporal case [14]. Given a temporal tableau \((H, B \cup A)\) and a temporal RDF graph \(G\), for each matching of the graph pattern \(B\) in the temporal closure of \(G\), pick up the values of the variables and check whether they satisfy the built-in predicates in \(A\). If this is the case, construct a pre-answer, which is the graph resulting by substituting the values of the variables in the head. Finally, the answer of the query is the union of all pre-answers.

We end this section by showing that the additional time dimension in our model does not play any relevant role in the complexity of query answering, that is, the query language preserves the tractability of answers. In order to do this, we consider the simpler problem of testing emptiness of the query answer set in the following forms:

1. Query complexity version: For a fixed database \(D\), given a query \(q\), is \(q(D)\) nonempty?
2. Data complexity version: For a fixed query \(q\), given a database \(D\), is \(q(D)\) nonempty?

**Theorem 8.** The evaluation problem is NP-complete for the query complexity version and polynomial for the data complexity version.

**Proof.** Query complexity version: Reduction of 3SAT to the problem of evaluating a conjunctive query over a database. Here, the time variables play the role of ordinary variables in conjunctive queries. Membership in NP follows immediately.

Data complexity version: This follows from the fact that the number of potential matchings of the body of \(q\) in a temporal graph \(tcl(G)\) is bounded by the number of subgraphs of \(tcl(G)\) of size \(|q|\). In addition, we have that \(tcl(G)\) is also polynomial on the size of \(G\).

The previous result shows that the temporal labeling over the triples does not introduce any complexity overhead. This is consistent with previous works in temporal databases. As Toman [25] showed, a point-based temporal head. This is consistent with previous works in temporal over the triples does not introduce any complexity over-...

Claudio Gutierrez received degrees in mathematics and in mathematical logic from Chilean universities and the PhD degree in computer science from Wesleyan University in 1999. He is an associate professor in the Computer Science Department at the Universidad de Chile. His research interests lie in the intersection of logic, databases, and the Semantic Web. In 2005, he received the Best Paper Award at the European Semantic Web Conference. Currently, he is an associate researcher at the Center for Web Research, where he works on Semantic Web databases.

Carlos A. Hurtado received the PhD degree in computer science from the University of Toronto in 2002. He is an assistant professor in the Computer Science Department at the Universidad de Chile, Chile, and an associate researcher at the Center for Web Research. His research areas include databases, the Semantic Web, Web data mining, OLAP, and data warehousing. He has served on the program committee of the International Conference on Very Large Databases (2004) and the International Conference on Database Theory (2007), among others.

Alejandro Vaisman is a civil engineer and computer scientist and holds a PhD degree in computer science from the Universidad de Buenos Aires, Argentina. He was a postdoctoral researcher at the Universidad de Chile, and a visiting researcher at the University of Hasselt, Belgium. He was also an invited lecturer at the Universidad Politecnica de Madrid, Spain. He has authored and coauthored several scientific papers presented in major database conferences. His research interests are in relational and deductive databases, OLAP and data warehousing, temporal databases, data mining, and Web-based information systems. He has worked in design and operation of database systems and has served as the vice-dean at the Universidad de Belgrano, Argentina, and vice-head of the Computer Science Department at the Universidad de Buenos Aires, Argentina. He is now a visiting researcher at the Universidad de Chile.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.