Temporal RDF

Claudio Gutierrez[1], Carlos Hurtado[1], and Alejandro Vaisman[2]

[1] Department of Computer Science
Universidad de Chile
{cgutierr,churtado}@dcc.uchile.cl
[2] Department of Computer Science
Universidad de Buenos Aires
avaisman@cs.toronto.edu

1 Introduction

The *Resource Description Framework (RDF)* [16] 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* very much 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). This vocabulary deals with inheritance of classes and properties, as well as typing, among other features [4]. RDFS allows to write ontologies, i.e., descriptions of the concepts and relationships that can exist for a community of people and software agents, enabling knowledge sharing and reuse among them.

Although some studies exist about addressing changes in an ontology [17], little attention has deserved the problem of representing, updating and querying temporal information in RDF. Of course, time is present in almost any web and e-business 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 of applying temporal database concepts to RDF to allow metadata navigation across time.

Consider an RDF graph describing information about a university, as of its creation time, Figure 1 (left). Students were classified as technical, graduate or undergraduate, and the only graduate programs offered were at the level of 'Master' studies (like MBA or MSc); 'Professional Diploma' was the only program offered at the technical level. As the university evolved, the Ph.D program was created. Figure 1 (right) illustrates the new situation. Notice the dynamics of this example: students (e.g., John) can enroll in one program (e.g., Undergraduate), then shift to another one (e.g., Master), and so on. The figures show that the impact of disregarding the time dimension is twofold: on the one hand, when a change occurs, a new metadata document must be created (and the current

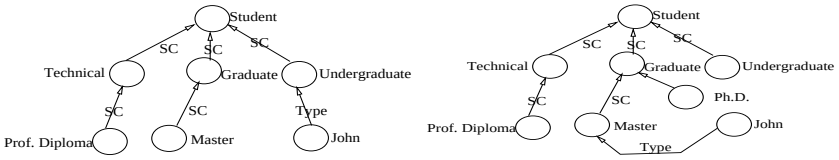

Fig. 1. (left) Initial RDF graph. (Right) The RDF graph after some changes.

document dropped). On the other hand, queries asking for past states of the metadata cannot be supported. For instance, we cannot ask for the programs offered when the university was created; also we cannot track the programs taken by John in different times as student.

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 main issues that arise when extending RDF with temporal information.

Versioning vs. Time Labeling There are two mechanisms for adding the time dimension to non-temporal RDF graphs: labeling and versioning . The former consists in 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. A variation of this strategy may consist in keeping the initial graph, and store only the changes, by means, for example, of edit scripts. We believe that for RDF data, labeling is better than versioning. On the one hand, labeling –as opposed to versioning– preserves the spiritsed to versioning– preserves the spirit of the distributed and extensible nature of RDF. On the other hand, for scenarios where changes are frequent and only affect a few elements of the document, labeling works better than versioning. In this situation, 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. Moreover, a version management approach accounts only for *transaction* time, while our approach is apt for handling *valid* time too.

Timestamp vs. Snapshot Semantics The labeling approach for temporal database representation considers the *timestamp* and the *snapshot* models. The former represents a temporal database as a function from a temporal domain to a database state. The latter associates a time instant to an element in the database (*e.g.* a tuple). Of course, both models are equivalent, but the snapshot representation appears to be not suitable for queries of the form: "all time instants where Φ holds in the database". As a second temporal issue, at least two temporal dimensions can be considered: *valid* time is the time when data is valid is the modeled

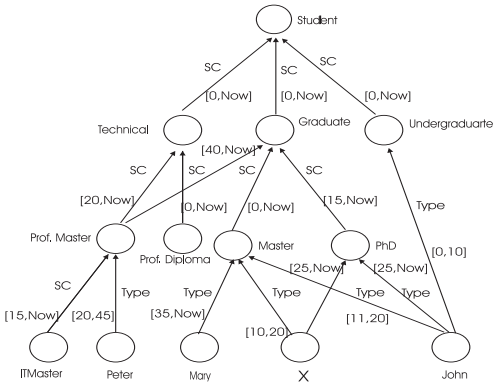

Fig. 2. A temporal RDF graph that accounts for the evolution of the university ontology and its students. For example, John was undergraduate until time $t = 10$ and then appears as master student from time $t = 11$ to $t = 20$, and since time $t = 25$ up to now is a Ph.D. student.

world; *transaction* time is the time when data is actually stored in the database. The snapshot representation captures transaction time, while timestamping is mostly used when representing valid time.

Time Points vs. 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. Figure 2 shows a temporal RDF graph for the university example above.[3] The edges in the graph are labeled with their interval of validity. For example, the interval [0,Now] says that the triple (technical,`sc`,student) is valid from the document's creation time to the current time. There is a blank node saying that there was one resource of type 'Master' in the interval [0,20], and another one in the interval [25,Now]. Also, Figure 2 shows that John was first an undergrad student, then a master student, and, after some time, a Ph.D student. We will see that blank nodes introduce many interesting problems is temporal RDF.

Vocabulary for Temporal Labeling Temporal labeling can be implemented within the RDF specification, making use of reification plus some simple additional vocabulary, as Figure 3 shows. As we adopted the point-based, discrete and linearly ordered temporal domain, the left and right hand sides of Figure 3 are

[3] Note that the graph(ical) representation of an RDF graph is not the most faithful to convey the the idea of a triple (not only the edge) being labeled by a temporal element. Technically in the picture the temporal element should be attached to the whole subgraph $a \xrightarrow{b} c$, not only to the edge.

equivalent. We will use both representations indistinctly. Moreover, we define constructs that allow moving between intervals and time instants as follows: the instants depicted in Figure 3 (left) can be encoded in an interval as shown in Figure 3 (right). Both alternatives will be used in the query language.

Fig. 3. (left) Point-based labeling. (right)Interval-based labeling.

Temporal Entailment An RDF graph can be regarded as knowledge base from which new knowledge, i.e., other graphs, may be entailed. As an example, Figure 4 shows a simple example of a temporal entailment. In the graph of the left hand side of Figure 4, a snapshot at $t = 2$ entails the graph in the right hand side of the same figure, since the latter is a subgraph of the former. A problem that arises when defining entailment in the temporal setting, refers to the impact of blank nodes in the entailment of RDF graphs. 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. Blank nodes impose some constraints to this approach. Each of the three snapshots of Figure 5 (right) entails the corresponding snapshots of Figure 5 (left). However, the temporal graph of Figure 5 (left) cannot be entailed by the temporal graph of Figure 5 (right). The former has more information than the latter. Indeed, the graph of Figure 5 (left) states that there is an anonymous object called X, which is in the triple (a, b, X) at times 3 and 4, which is not the case for the other graph.

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 [22, 3]. 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 five years ago as a W3C recommendation [16]. Formal work in RDF includes the study of formal aspects of RDF data and query languages [12, 23], considering RDF features like the entailment, presence

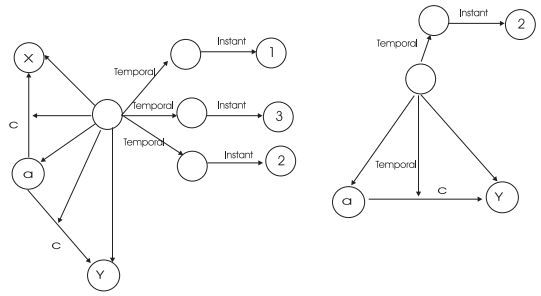

Fig. 4. Temporal entailment

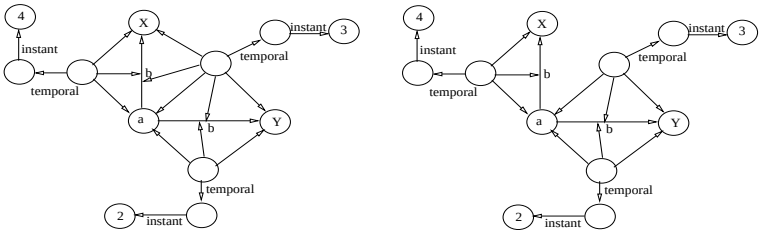

Fig. 5. Two temporal RDF graphs.

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 in the lines of traditional database query languages (e.g. SQL, OQL), others based on logic and rule languages. Good surveys are [15, 18].

To the best of our knowledge, there is still no formal study of temporality issues in RDF graphs and RDF query languages. The closest research done is in the area of Web and semistructured data. We end this section by surveying related work on temporal data models for Web and semistructured data.

Temporal database management has been extensively studied, including data models, mostly based on the relational model [21], and query languages [8], leading to the TSQL2 language [20]. Beyond the relational model, managing historical semistructured data was first proposed by Chawathe *et al* [7], 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* [9] 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 [10] 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* [5] 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 version

time. In a sequel of this work [6], they moved to a scheme where version management is performed by keeping references to the maximal unchanged subtree in the previous version sharing unchanged elements among versions. Gao *et al* [11] introduced τXQuery, an extension to XQuery supporting valid time while maintaining the data model unchanged. Finally, Mendelzon *et al* [19] 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.

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:

- A semantics for temporal RDF graphs in terms of the semantics of non-temporal RDF and RDFS graphs.
- A study of properties of temporal RDF graphs, such as normal forms, and the interplay between timestamp and snapshot semantics in temporal RDF graphs.
- A syntax to incorporate this framework into standard RDF graphs, which includes a vocabulary and rules. The syntax uses the standard RDF reification vocabulary plus temporal labels.
- A sound and complete inference system for temporal RDF graphs.
- Complexity bounds which show that entailment in temporal RDF graphs does not yield extra asymptotic time complexity with respect to standard RDF graphs.
- A sketch for a temporal query language for RDF. We show use cases, and complexity of query processing.

For the sake of space, we do not include proofs in this version of the paper.

2 RDF Preliminaries

In this section we present a streamlined formalization of the RDF model following the W3C documents [16, 14, 4], along the lines of [12].

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 $(v_1, v_2, v_3) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an *RDF triple*. In such a triple, v_1 is called 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 a graph. The *universe* of a graph G, universe(G), is the set of elements

of UBL that occur in the triples of G. The *vocabulary* of G is the set universe$(G) \cap (U \cup L)$. We will use letters $N, X, Y, \ldots$ to denote blank nodes, and $a, b, c, \ldots$ for URIs 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 $a \xrightarrow{b} c$. Note that the set of arc labels can have non-empty intersection with the set of node labels.

A *map* is a function μ : UBL $\rightarrow$ 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 μ is *consistent* with G if $\mu(G)$ is an RDF graph, *i.e.* , if s is the subject of a triple, then $\mu(s) \in UB$, 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 μ sends a blank node to an URI or a literal, or identifies two blank nodes of G. We will overload the meaning of map and speak of a *map* $\mu : G_1 \rightarrow G_2$ if there is a map μ such that $\mu(G_1)$ is a subgraph of G_2.

Two graphs G_1, G_2 are *isomorphic*, denoted $G_1 \cong G_2$, if there are maps μ_1, μ_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 [4], –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. It defines classes and properties that may be used for describing groups of related resources and relationships between resources.[4] *Classes* are sets of resources. Elements of a class are known as *instances* 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:Literal [`literal`], rdfs:Datatype [`datatype`], rdf:XMLLiteral [`xmlLit`], rdf:Property [`property`]. *Properties* are binary relations between subject resources and object resources. The built-in properties are: rdfs: range [`range`], rdfs:domain [`dom`], rdf:type [`type`], rdfs:subClassOf [`sc`], rdfs:subPropertyOf [`sp`].

In what follows will be important the *reification* vocabulary, which was designed to allow making statements about statements. It consists of rdf:Statement

[4] We omit in this paper vocabulary intended to describe lists, collections, some variations on these, as well as vocabulary to help document and describe other functionalities for which there is no normative semantics. The complete vocabulary can be consulted in [4].

[stat], rdf:subject [subj], rdf:predicate [pred], and rdf: object [obj]. The reification vocabulary has no standard semantics. Reification of a triple is not unique (can be different reifications of the same triple) and the reification of a triple does not follow from the triple itself.

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 t labeling a triple (a, b, c). For simplicity, without loss of generality, we will work with single intervals instead of temporal elements. In an RDF graph, given a triple (a, b, c), the temporal element t represents the time period when the triple was valid, *i.e.* the *valid time* of the triple. At this time we do not deal with *transaction time*, which can be addressed in an analogous way.

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 with a temporal label (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. A subgraph is a subset of the graph.*

For a temporal graph G, define the snapshot *at time t as the RDF graph*

$$G(t) = \{(a, b, c) \mid (a, b, c) : [t] \in G\}$$

The underlying RDF graph *of a temporal RDF graph G, denoted $u(G)$, is $\bigcup_t G(t)$, the* union *of the graphs $G(t)$.*

For an RDF graph, define G^t as the temporalization of all its triples by a temporal mark t, that is, $G^t = \{(a, b, c) : [t] \mid (a, b, c) \in G\}$.

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

Lemma 1. *Let G be an RDF graph, and G' be a temporal RDF graph. Then: (1) $G^t(t) = G$; (2) $(G'(t))^t \subseteq G'$, and (3) $G' = \bigcup_t (G'(t))^t$.*

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

- Recall we use 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 this issue 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 reification plus a small temporal vocabulary.
- Source of a temporal statement: 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. Define*

- *For ground temporal RDF graphs G_1, G_2 define $G_1 \models_t G_2$ as $G_1(t) \models G_2(t)$ for each t;*
- *For general graphs, $G_1 \models_t G_2$ iff there exist ground instances $\mu_1(G_1)$ and $\mu_2(G_2)$ such that $(\mu_1(G_1))(t) \models (\mu_2(G_2))(t)$ for each t.*

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

Proposition 1. *Let G_1, G_2 be temporal graphs. Then, $G_1 \models_t G_2$ implies $G_1(t) \models_t G_2(t)$ for all t, and the converse is true for ground graphs.*

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

- Existential variables (blank nodes) make 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 example in Figure 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 $(a, , c) : [2, 3], (c, , d) : [2]$, then we should be able to derive $(a, , d) : [2]$, but not $(a, , d) : [3]$.

In the rest of this section, we show that the notions of closure, lean graph, core –fundamental to define notions of normalization of this data– can be extended without difficulty to the temporal setting. (Compare discussion in [12]).

The *closure* of a temporal graph G, denoted $\mathrm{tcl}(G)$, is a maximal set of temporal triples G' over universe of G plus the RDF vocabulary such that G' contains G and is equivalent to it.

Proposition 2 (Entailment for Temporal graphs).
Let G, G_1, G_2 be temporal RDF graphs. Then

1. $\mathrm{tcl}(G) = \bigcup_t (\mathrm{cl}(G(t)))^t$;
2. $G_1 \models_t G_2$ *iff* $\mathrm{tcl}(G_1) \models_t G_2$.

A temporal graph G is *lean* iff there is no proper temporal subgraph G' of G such that $G \models_t G'$. The *core* of G is a lean subgraph of G equivalent to it.

The computational complexities of computing the core and testing whether a graph is lean, are asymptotically the same as the case of standard RDF graphs.

Proposition 3. *Let G, G' be graphs.*
1. *The problem of deciding if G' is the closure of G is DP-complete.*
2. *The problem of deciding if G' is the normal form of G is DP-complete.*
3. *The problem of deciding if G' is the reduction of G is DP-complete.*

For a temporal RDF graph G, as in the case of RDF graphs, we can define a notion of *normal form*, denoted by $\mathrm{nf}_t(G)$, as follows: $\mathrm{nf}_t(G) = \mathrm{core}_t(G)$ for a temporal closure (as in Definition 4) G' of G.

4 Syntax and Deductive System for Temporal Graphs

We present a deductive system for temporal RDF. It is based on a sound and complete set of rules given in [14], plus three rules capturing temporal issues.

4.1 RDF syntax of temporal triples

Definition 3 (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: $\mathrm{reif}(a, b, c, X)$: reification of the triple (a, b, c) with variable X, i.e. the set of triples $(X, \mathtt{subj}, a)$, $(X, \mathtt{pred}, b)$, $(X, \mathtt{obj}, c)$, $(X, \mathtt{type}, \mathtt{stat})$.

Definition 4 (Temporal triples and graphs). *Temporal triples are the following graphs using the temporal vocabulary.*

- $(a, b, c), \mathrm{reif}(a, b, c, X), (X, \mathtt{tpl}, Y), (Y, \mathtt{instant}, n)$ *where n is a natural number; we will summarize this as* $(a, b, c) : [X, Y, n]$;
- $(a, b, c), \mathrm{reif}(a, b, c, X), (X, \mathtt{tpl}, Y), (Y, \mathtt{interval}, Z), (Z, \mathtt{initial}, I), (Z, \mathtt{final}, F)$; *where I, F are natural number; we will summarize this as* $(a, b, c) : [X, Y, I, F]$;
- *A* temporal graph *will be defined as a* merge *of a set of temporal triples.*

Because RDF is extensible, nothing prevents the use of the blank nodes included in the definition as target or source of other properties beyond the temporal vocabulary. We want to have a definition of temporal triple independent of the blank nodes occurring in the proposed syntactic definition of temporal triples, e.g., we would like that $(a, b, c) : [X, Y, n]$ be essentially equivalent to $(a, b, c) : [n]$. Both previous issues are overcome in our syntax by adding certain rules, which regulate the temporal vocabulary.

4.2 Rules

The set of rules is arranged in four groups. Groups A, B, C, and D are intended to describe the classical RDFS semantics, and we follow the approach in [12]. We omit another group of rules that has to do with internal relationships of the RDF model itself and that we do not consider in this paper.

The novelty here is Group T (temporal rules), whose main objective is to be able to standardize the interval version and the instant version as well as help defining "absolute" temporal marks.

GROUP A (Existential) For a map $\mu : G' \to G$:

$$\frac{G}{G'} \tag{1}$$

GROUP B (Subproperty)

$$\frac{(a, \texttt{type}, \texttt{property})}{(a, \texttt{sp}, a)} \tag{2}$$

$$\frac{(a, \texttt{sp}, b) \quad (b, \texttt{sp}, c)}{(a, \texttt{sp}, c)} \tag{3}$$

$$\frac{(a, \texttt{sp}, b) \quad (x, a, y)}{(x, b, y)} \tag{4}$$

GROUP C (Subclass)

$$\frac{(a, \texttt{type}, \texttt{class})}{(a, \texttt{sc}, a)} \tag{5}$$

$$\frac{(a, \texttt{sc}, b) \quad (b, \texttt{sc}, c)}{(a, \texttt{sc}, c)} \tag{6}$$

$$\frac{(a, \texttt{sc}, b) \quad (x, \texttt{type}, a)}{(x, \texttt{type}, b)} \tag{7}$$

GROUP D (Typing)

$$\frac{(a, \texttt{dom}, c) \quad (x, a, y)}{(x, \texttt{type}, c)} \tag{8}$$

$$\frac{(a, \texttt{range}, d) \quad (x, a, y)}{(y, \texttt{type}, d)} \tag{9}$$

GROUP T (Temporal)

$$(i2t)\ \frac{\{(X, \texttt{tpl}, Y), (Y, \texttt{instant}, n) : n \in [t_1, t_2]\}}{(X, \texttt{tpl}, Y), (Y, \texttt{int}, Z), (Z, \texttt{initial}, t_1), (Z, \texttt{final}, t_2)} \tag{10}$$

$$(t2i)\ \frac{(X, \texttt{tpl}, Y), (Y, \texttt{int}, Z), (Z, \texttt{initial}, t_1), (Z, \texttt{final}, t_2)}{\{(X, \texttt{tpl}, Y), (Y, \texttt{instant}, n)}, \quad n \in [t_1, t_2] \tag{11}$$

$$(abs)\ \frac{(a,b,c) : [X_1, Y_1, n_1],\ (a,b,c) : [X_2, Y_2, n_2]}{(a,b,c) : [X_1, Y_1, n_1],\ (a,b,c) : [X_1, Y_1, n_2]} \tag{12}$$

Rules (i2t) (interval to instants) and (t2i) are needed to standardize the interval version and the instant version, by making them equivalent. Rule (abs) essentially says that marks (instants) can be collected in a single node. This permit to concentrate on temporal marks independent of other contexts in which the variables involving temporal vocabulary are immersed.

The definition behaves well in the sense of the following lemma.

Lemma 2. *1.* $G \models_t \exists X \exists Y (a,b,c) : [X, Y, t_1, t_2]$ *if and only if* $G \models_t \exists X \exists Y \bigwedge_{t_1 \leq j \leq t_2} (a,b,c) : [X, Y, t_j]$

2. $G \models_t \exists X_1 \exists Y_1 (a,b,c) : [X_1, Y_1, t_1] \wedge \exists X_2 \exists Y_2 (a,b,c) : [X_2, Y_2, t_2]$ *if and only if* $G \models_t \exists X \exists Y ((a,b,c) : [X, Y, t_1] \wedge (a,b,c) : [X, Y, t_2])$

For a temporal RDF graph G, define G^* as the RDF graph $\{(a,b,c) : [X_t, Y_t, t] \mid (a,b,c) : [t] \in G\}$, where X_t, Y_t are free blank variables, different for each t. Conversely, for each RDF graph G with temporal vocabulary, define G_* as the temporal graph defined as $\{(a,b,c) : [t] \mid \exists X \exists Y\ (a,b,c) : [X, Y, t] \in G\}$.

Theorem 1. *1. Let* G_1, G_2 *be temporal RDF graphs. Then* $G_1 \models_t G_2$ *implies* $G_1^* \models G_2^*$.

2. Let G_1, G_2 *be RDF graphs with temporal vocabulary. Then* $G_1 \models G_2$ *implies* $(G_1)_* \models_t (G_2)_*$.

3. Let G *be a temporal RDF graph, and* G' *an RDF graph with temporal vocabulary. Then* $(G^*)_* = G$ *and* $G' \models (G'_*)^*$.

Now we can show that the syntax introduced captures the semantics of temporal RDF. The following deductive system based on the rules presented, is sound and complete for entailment of RDF graphs with rdfs vocabulary.

Definition 5. *Let* G *be a graph. For each rule* $r : \frac{A}{B}$ *above, define* $G \vdash_r G \cup \mu(B)$ *iff there is a map* $\mu : A \to G$. *Also define* $G \vdash_s G'$ *if and only if* G' *is a subgraph of* G.

Define $G \vdash G'$ *if there is a finite sequence of graphs* $G_1, \ldots, G_n$ *such that (1)* $G = G_1$; *(2)* $G' = G_n$; *and (3) for each* i, *either,* $G_i \vdash_r G_{i+1}$ *for some* r, *or* $G_i \vdash_s G_{i+1}$.

The following theorem shows that one can give a syntactic characterization over RDF graphs with temporal vocabulary for entailment of temporal RDF graphs:

Theorem 2. *For any pair of temporal RDF graphs* G_1, G_2*:*
$G_1 \models_t G_2$ *if and only if* $G_1^* \vdash G_2^*$

Note that that due to the examples presented in a previous Note, we cannot establish the theorem in its complete generality, namely, prove that if $G_1 \vdash G_2$ then $(G_1)_* \models_t (G_2)_*$.

The previous theorem permits to concentrate for the following sections in temporal RDF (instead of diving into syntactic issues).

5 Query language

In this section we present 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 Figure 2. Let us begin with a simple query: "Students taking Master courses between t_1 and t_2" (*i.e.* starting and completing their studies within this interval); the query can be expressed as:

$$(?X, \texttt{type}, \mathit{Student}) \leftarrow$$
$$(?X, \texttt{takes}, ?C) : [?T], (?C, \texttt{type}, \mathit{Master}) : [?T], t_1 \leq ?T, ?T \leq t_2.$$

This example query illustrates the need of a built-in arithmetic language in order to reason about time and intervals. Another important observation is that temporal queries may output non-temporal RDF graph, as the previous query does.

For the query asking for a snapshot of the graph at t_1, we have:

$$(?X, ?Y, ?Z) \leftarrow (?X, ?Y, ?Y) : [t_1].$$

Now consider the query "Students taking Ph.D courses together, and the time instants when this occurred." For simplicity we expressed this as a point-based query. The translation of the result into intervals is straightforward.

$$(?\mathrm{X}, \mathit{together}, ?\mathrm{Y})[?T] \leftarrow (?\mathrm{X}, \texttt{type}, \mathit{Ph.D}) : [?T], (?\mathrm{Y}, \texttt{type}, \mathit{Ph.D}) : [?T].$$

Next, we give examples of queries that use temporal triples with intervals.

The query "Time intervals when the IT Master was offered" can be expressed as follows:

$$(X, \texttt{interval}, Y), (Y, \texttt{initial}, t_i), (Y, \texttt{final}, t_f) \leftarrow$$
$$(\mathit{ITMaster}, \texttt{sc}, \mathit{Prof.Master}) : [t_i, t_f].$$

Observe that the previous query returns a set of intervals. In order to retrieve maximal intervals we need a more subtle query, since their computation do not follows directly from the temporal rules. For the query "Compute the maximal interval when the triple (a, b, c) holds", we need aggregate operators `MAX` and `MIN`.

$$(a, b, c) : [?T_1, ?T_2] \leftarrow (a, b, c) : [?T_i, ?T_f], ?T_1 = \texttt{MIN}(?T_i), ?T_2 = \texttt{MAX}(?T_f)$$

For a query asking for "Students taking Ph.D courses after completing a master program" we have:

$$(?X, \texttt{type}, Ph.D) \leftarrow (X, \texttt{type}, Ph.D) : [?T], (?\text{X}, \texttt{type}, Master) : \|t_i, t_f\|, t_f <?T.$$

Here, the notation $(?\text{X}, \texttt{type}, Master) : \|t_i, t_f\|$ stands that t_i and t_f match with the maximal interval for the triple $(?\text{X}, \texttt{type}, Master)$, computed with the query given above.

Finally, consider the query "Students enrolled in the Professional Master program, exactly the time when the Professional Master program becomes recognized as a Graduate program."

$$(?\text{X}, \texttt{type}, Prof.Master) \leftarrow (?\text{X}, \texttt{type}, Prof.Master) : [?T], (Prof.Master, \texttt{sc}, Graduate) : \|?T, t_f\|.$$

5.2 Semantics and Complexity

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 $V_t \subset V$.

The query language we define is analogous to the one presented by Gutierrez et al. [13]. A query is a *temporal tableau*, which is a pair $(H, B \cup A)$, where H and B are a temporal RDF graphs with some UBLs replaced by variables in V, and with some Ts 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 *safe rule* from Datalog to prevent operations on infinite predicates. A rule is safe is all its variables are limited. A variable is limited if one of the following hold: 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 is the usual in these cases. Given a temporal tableau $(H, B \cup A)$ and a temporal RDF graph G, for each matching of the graph pattern B in 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 answer. 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 $q(D)$ non-empty? (2) Data complexity version: For a fixed query q, given a database D, is $q(D)$ non-empty?

Theorem 3. *The evaluation problem is NP-complete for the query complexity version, and polynomial for the data complexity version.*

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 [22] showed, a point-based temporal query language has the same properties than a First Order query language, in spite of the temporal variable.

6 Conclusions

We have proposed an RDF vocabulary to assert the times when triples are valid in RDF graphs. This allows an explicit treatment of time inside RDF. We have also offered a complete and sound inference procedure for temporal RDF graphs, and a query language for them. Our framework allows to browse, query, and reason across different versions of RDF graphs.

As future work, we consider the definition of a built-in arithmetic language to reason about intervals in queries. This language should include at least basic set theoretical operations with intervals. As an example, we may need to capture the extremes of a time interval when a triple holds, or the intersection interval when a set of triples are valid.

Other issue is to handle anonymous times. For example, we may want to say that a triple holds in sometime inside an interval, but do not know the exact valid time of the triple. Anonymous times may help in the specification of triples without temporal labels, which is a form to specify incomplete temporal information, i.e. for the case when one cannot deduce any positive temporal statement about RDF triples. Another issue is the output of the answer. Usually there are two kinds of answers: temporal ones, and plain ones. A unified semantic for these two classes of answers would allow closeness and full query composition in a temporal query language for RDF.

References

1. S. Abiteboul. Querying Semi-Structured Data. *Proceedings of the 6th International Conference on Database Theory (ICDT'97)*. Delphi, Greece, 1997. Lecture Notes in Computer Science, Springer.
2. T. Amagasa, M. Yoshikawa, S. Uemura, *A Temporal Data Model for XML Documents*, Proceedings of DEXA Conference, 2000, 334-344.

3. M. Bölen98, R. Busatto, C.S. Jensen*Point- Versus Interval-based Temporal Data Models*, Proceedings of IEEE/ICDE, 1998.
4. Dan Brickley, R.V. Guha Eds., *RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft 23 January 2003*.
5. S. Chien, V. Tsotras, C. Zaniolo, *Version Management of XML Documents*, Proceedings of the Third International Workshop on the Web and Databases, 2000, 75-80.
6. S. Chien, V. Tsotras, C. Zaniolo, *Efficient Management of Multiversion Documents by Object Referencing*, Proceedings of the 27th International Conference on Very Large Data Bases,2002,Rome, Italy, 291-300.
7. S. Chawathe, S. Abiteboul, J. Widom, *Managing Historical Semistructured Data*,Theory and Practice of Object Systems, Vol 5(3), 1999,143-162.
8. J. Chomicki, *Temporal Query Languages: a Survey*, Proceedings of the 1st International Conference on Temporal Logic,LNAI 827, 1994, 506-534.
9. C.E. Dyreson, M.H. Bolen, C.S. Jensen, *Capturing and Querying Multiple Aspects of Semistructured Data*, Proceedings of the 25th VLDB Conference, 1999, 290-301.
10. C.E. Dyreson, *Observing Transaction-time Semantics with TTXPath*, Proceedings of WISE 2001, 2001,193-202.
11. C. Gao, R.Snodgrass, *Temporal Slicing in the Evaluation of XML Queries*, Proceedings of the 29th International Conference on Very Large Data Bases, 2003, 632-643, Berlin, Germany.
12. C. Gutierrez, C. Hurtado, A.O. Mendelzon,*Formal aspects of querying RDF databases*, SWDB 2003, 293-307
13. C. Gutierrez, C. Hurtado, A.O. Mendelzon,*Foundations of Semantic Web Databases*, 23rd. Symposium on Principles of Database Systems, PODS 2004, 95-106.
14. Patrick Hayes Ed., *RDF Semantics, W3C Working Draft, 1 October 2003*
15. P. Haase HAASE, J. Broekstra, A. Eberhart, R. Volz. *A comparison of RDF Query Languages*. International Semantic Web Conference, 2004.
16. O. Lassila, R. Swick Eds., *Resource description framework (RDF) model and syntax specification*, Working draft, W3C, 1998.
17. A. Maedche, B. Motik, L. Stojanovic, R. Studer, R. Volz*Establishing the semantic web 11: An infrastructure for searching, reusing, and evolving distributed ontologies*, Proceedings of the 12th. International Conference on World Wide Web,2003, 439-448.
18. A. Magkanaraki et al. *Ontology Storage and Querying*, Technical Report No. 308, April 2002, Foundation for Research and Technology Hellas, Institute of Computer Science, Information System Laboratory.
19. A.O. Mendelzon, F. Rizzolo, A. Vaisman, *Indexing Temporal XML*, Proceedings of the 30th International Conference on Very Large Data Bases, Toronto, Canada, 2004, 216-227.
20. R. Snodgrass, *The TSQL2 Temporal Query Language*, Kluwer Academic Publishers,1995.
21. A. Tansel, J. Clifford, S. Gadia Eds., *Temporal Databases: Theory, Design and Implementation*, Benjamin/Cummings, 1993.
22. D. Toman, *Point vs. Interval-based Query Languages for Temporal Databases*, 15 th. Symposium on Principles of Database Systems, PODS 1996, 58-67.
23. G. Yang, M. Kifer, *On the Semantics of Anonymous Identity and Reification* Proc. First International Conference on Ontologies, Databases and Applications of Semantics (ODBASE), 2002, 1047-1066.