Time in RDF

G. Falquet

CUI Université de Genève

April 8, 2024

Time in RDF

Some approaches to introduce time in RDF

a) Theoretical works

- Extending RDF and OWL with validity time (Gutierrez et al., 2007) (Motik, 2012)
- Temporal description logics (Lutz et Wolter, 2008)

b) Practical approaches

- Reification
- Named graphs
- 4D fluents
- n-ary relation patterns

Time

Two temporal dimensions

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

- Labeling is mostly used when representing valid time.

Introducing time into RDF[1]

Reference:

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

Temporal Domain

- Point-based temporal domain
- Encode time-points in intervals when possible (clarity)
- Time as a discrete, linearly ordered domain,
  - as usual in virtually all temporal database applications.
- $[a, b]$ with $a \leq b$, denotes the closed interval from a to b, i.e $[a, a+1, a+2, \ldots, b]$.

Temporal Graph Definition

- A temporal triple is an RDF triple (a, b, c) with a temporal label t (a natural number).
  - Notation (a,b,c)[t].
  - $(a, b, c)[t_1, t_2]$ denotes $\{(a, b, c)[t] \mid t_1 \leq t \leq t_2\}$
- A temporal graph is a set of temporal triples.

Operations

$G\mid_t$, the slice at t, $\{(a,b,c)[u] \in G \mid u = t\}$ all temporal triples with temporal label t.

$u(G)$, the underlying graph of G, is the set $\{(a,b,c) \mid (a,b,c)[t] \in G \text{ for some } t\}$

$G(t)$, the snapshot of G at t, $G(t) \stackrel{def}{=} u(G\mid_t)$

Temporal entailment

For ground temporal RDF graphs (no blank nodes)

$G_1 \models_{time} G_2$
if and only if $G_1(t) \models G_2(t)$ for each t

For arbitrary temporal RDF graphs

$G_1 \models_{time} G_2$
if and only if for every ground instance $\mu_1(G_1)$ there exists a ground instance $\mu_2(G_2)$ such that $\mu_1 G_1 \models_{time} \mu_2 G_2$.

Example

(X, Y are blank nodes)
$G_1 = \{(a, b, X)[3], (a, b, X)[4]\}$
$G_2 = \{(a, b, Y)[3], (a, b, X)[4]\}$

Entailments

$G_1 \models_{time} G_2$
$G_2 \not\models_{time} G_1$ (e.g. take $\mu_2(Y) = d, \mu_2(X) = e$, and $d \neq e$

Problem with blank nodes

$G_1(t) \models G_2(t)$ for each t does not imply $G_1 \models_{time} G_2$

- A blank node represents the same (unnamed) resource throughout the time range, rather than a sequence of different resources.
- Different from the classical setting.
- Temporal marks are not only a relation among fixed objects, but also among time-varying objects, the blank nodes.

Time arithmetics

- 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, subClassOf, c)[2; 3]; (c, subClassOf, d)[2],
  - then we should be able to derive (a, subClassOf, d)[2],
  - but not (a, subClassOf, d)[3]

Anonymous time

- To study temporal graphs that contain triples of the form $(a, b, c)[X]$, where X is an anonymous timestamp,
- State that the triple (a, b, c) is valid in some unknown time.
- ! The set of anonymous timestamps and blank nodes are disjoint;

- anonymous timestamps can be used to state that a set of triples occurred at the same time, even though their valid time is unknown.
  - $(a, b, c)[T], (a, b', c')[T]$
- A standard RDF graph can be made temporal by means of anonymous timestamps and, thus, modeled as temporal graphs.

Entailment

$$G_1 \models_{timegen} G_2$$

if and only if for each t-ground graph $\mu_1(G_1)$ (all the anonymous times have been replaced by time values), there is a t-ground graph $\mu_2(G_2)$ such that

$$\mu_1(G_1) \models_{time} \mu_2(G_2)$$

Examples

.

$$\{(a; subClassOf; b)[T_1]; (b; subClassOf; c)[T_1]\}$$

$$\models_{timegen} \{(a; subClassOf; c)[T_2]\}$$

But it is not the case that

$$\{(a; subClassOf; b)[T_2]; (b; subClassOf; c)[t_1]\}$$

$$\models_{timegen} \{(a; subClassOf; c)[T_2]\}$$

Query language

"Find the service providers who have offered a Web service between time instants 0 and 2, and return them qualified by early providers."

$$
\begin{aligned}
(?X, type, earlyProvider) \quad & \leftarrow (?X, type, serviceProvider)[?T], \\
& (?S, providedBy, ?X)[?T], 0 \leq ?T, ?T \leq 2
\end{aligned}
$$

asking for a snapshot of the graph at time 2, we have:

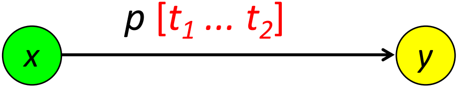
$$(?X; ?Y; ?Z) \leftarrow (?X; ?Y; ?Z)[2]$$

"Find the services providers, along with the Web services they have offered, and the time instants when this occurred."

$$(?X; hasprovided; ?Y)[?T] \leftarrow (?Y; providedby; ?X)[?T]$$

Practical approaches

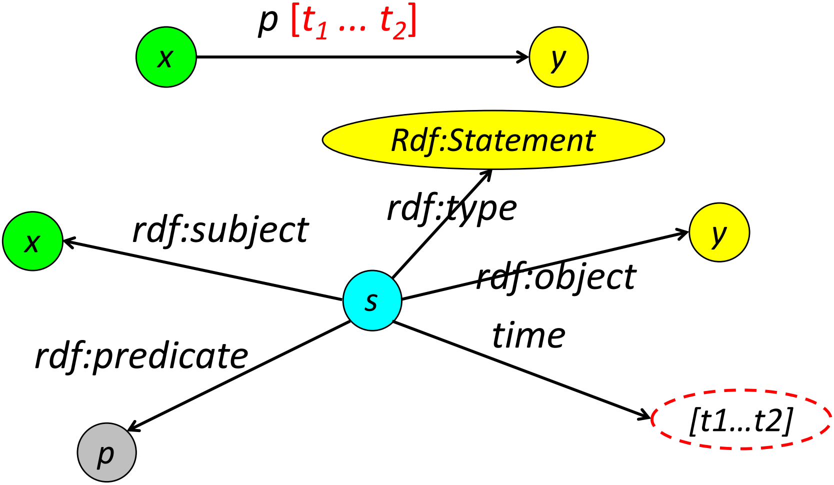
Problem: how to represent



How to implement entailment ?

Reification

transform relations into objects



Discussion

Representation

- complete
- replace 1 triple by $\geq$ 3 triples

Queries become much more complex

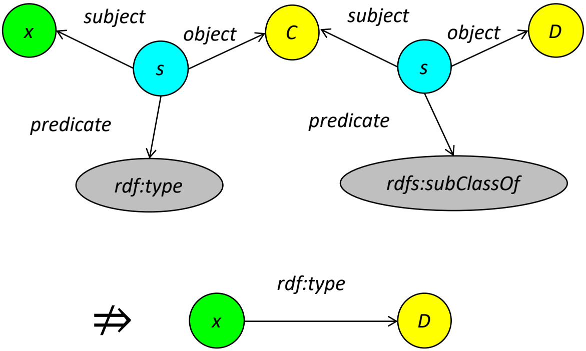
```
select ... where {?x ?p ?y}
```

$\Downarrow$

```
select ... where
{?s rdf:subject ?x; rdf:object ?y; rdf:predicate ?p}
```

Discussion

Reification is not taken into account in RDF, RDFS, OWL entailment
$\Rightarrow$ no reasoning on the temporal triples

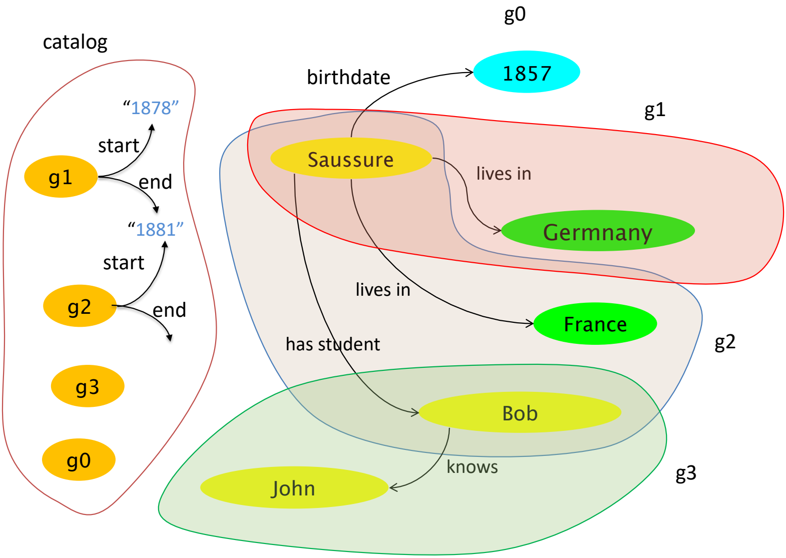


Named graph technique

- Each named graph has an associated time interval and contains triples that are valid for that time interval[2].
- An additional graph is used as a catalog of the named graphs and the interval associated with each.

[2]Tappolet, J. and Bernstein, A., Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In: Proceedings of the European Semantic Web Conference, LNCS 5554, (2009) 308-22

Example


catalog
“1878”
start
g1
end
“1881”
start
g2
end
g3
g0
g0
birthdate
1857
g1
Saussure
lives in
Germnany
lives in
France
g2
has student
Bob
knows
John
g3

```
# retrieve validity interval
[start, end] = tripleToAdd.validity
if tripleToAdd containsOneOrMore blankNode
        # generate URI for blank-node
        tripleToAdd = convertBlankToURI(tripleToAdd)
endif
# if no named graph for the interval exists then make one
if !namedGraph[start, end].exists
  create new namedGraph[start, end]
  # add the new named graph to directory in default graph
  namedGraph.defaultGraph += namedGraph[start,end].temporalInfo
endif
namedGraph[start,end] +=  tripleToAdd
```

Listing 1. Pseudo Code for an insert operation into a temporal graph

```
<PREFIX time: http://www.w3.org/2006/time#>
<PREFIX foaf: http://xmlns.com/foaf/0.1/#>
SELECT  ?s2 ?e2 ?person WHERE{
        [?s1,?e1] ?einstein foaf:name "Albert Einstein" .
        [?s2,?e2] time:intervalOverlaps [?s1,?e1] .
        [?s2,?e2] ?person a foaf:Person .
}
```

Listing 4. τ-SPARQL: Interval relations

```
<PREFIX time: http://www.w3.org/2006/time#>
<PREFIX foaf: http://xmlns.com/foaf/0.1/#>
SELECT  ?s2 ?e2 ?person WHERE{
        GRAPH g1 { ?einstein foaf:name "Albert Einstein" .}
        ?g2 time:intervalOverlaps ?g1 .
        GRAPH g2 { ?person a foaf:Person }.
        ?g2 time:hasBeginning ?start .
        ?start time:inInteger ?s2 .
        ?g2 time:hasEnd ?end .
        ?end time:inInteger ?e2 .
}
```

Listing 6. Rewritten τ-SPARQL query

Discussion

- Reduces the number of triples required over some other approaches.
- Problem with blank nodes
  - B-nodes == existentially qualified variables
  - the scope of quantification is the graph
  - this approach puts statements about the same b-node for different time intervals into different named graphs
  - To avoid this problem
    - b-nodes are replaced by URIs thus allowing the use of off the shelf software for processing RDF.
- Named graphs are not part of description logic (not considered in OWL)

4D Fluents

The 4D approach[3] is not something that immediately appeals to common sense, as statements such as

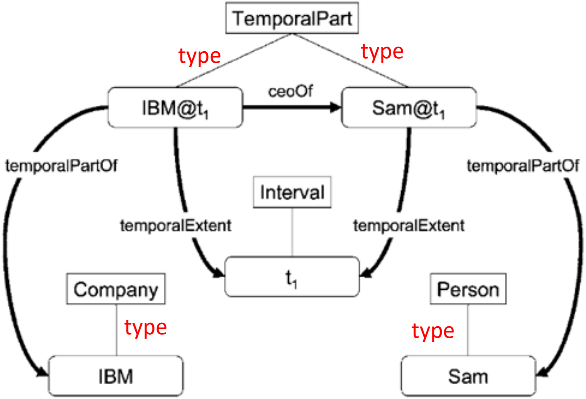
"Joe walked into the room"

must be represented as the logical equivalent of

"A temporal part of Joe walked into a temporal part of the room".

[3] C. Welty and R. Fikes. A Reusable Ontology for Fluents in OWL. In Formal Ontology in Information Systems B. Bennett and C. Fellbaum (Eds.) IOS Press, 2006.

Example

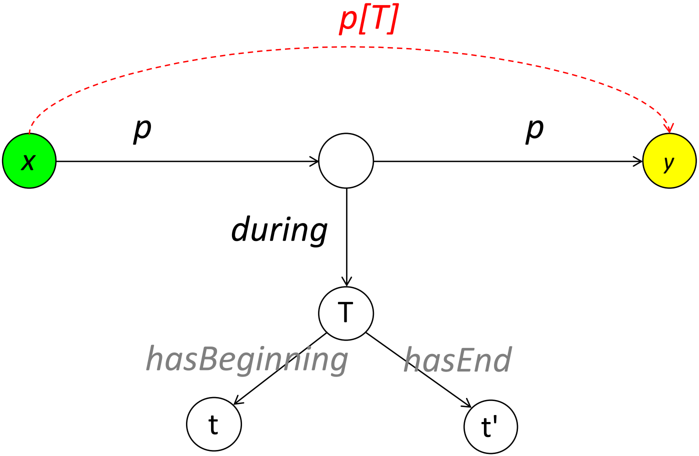


TemporalPart
type
type
IBM@t1
ceoOf
Sam@t1
temporalPartOf
temporalPartOf
Interval
temporalExtent
temporalExtent
t1
Company
type
Person
type
IBM
Sam

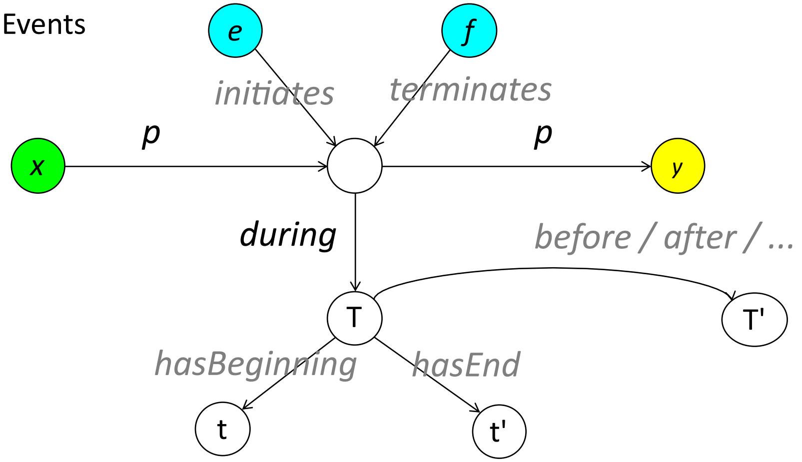
Discussion

- Entirely compatible with RDF and OWL
  - except for synchronic reasoning (impossible to restrict inference to triples that "occur at the same time"
- RDF/S reasoners can be used to infer new facts
- Proliferation of entities
  - each entity is represented by several slices

N-ary relation pattern for fluent relations



With events and temporal relations



Entailment

Non-temportal or basic temporal entailment can be expressed in OWL

$$X \equiv \exists livesIn.(\exists during.(\exists hasBeginning.\{1900\}) \sqcap \exists livesIn.EuropeanCity)$$

More complex inferences must be expressed in SPARQL (SPIN)

Example

Inference rule for a temporally functional property (has only one value at a given instant)

```
insert (?y1 owl:sameAs ?y2)
where {Q(?x, ?f1, ?y1, ?i1),
       Q(?x, ?f2, ?y2, ?i2),
      filter (overlaps(?i1, ?i2))}
```

where

```
Q(?X, ?F, ?Y, ?I)
```

represents

```
?F a FluentRelation. ?X prop ?F. ?F prop ?Y.
?F during ?I.
```

Inference that generate new nodes

Representing the rule:

> *If a manuscript M is a letter from A to B and the writing time of M is [t1 .. t2] then A knows B during [t1 .. end of considered period]*

```
INSERT {?A :knows
  [a :FluentRelation ;
   :during [a :TimeInterval ;
              :hasBeginning ?t1;
              :hasEnd :end-of-period];
   :knows ?B]}
 WHERE {?L a :Letter; :author ?A;
          :to ?B; writingDate ?t1}
```

$\Rightarrow$ creation of a new blank node

Discussion

Compatible with RDFs/OWL entailment, except for synchronic reasoning
Creation of new nodes when infering new temporal facts $\Rightarrow$ potentially infinite computations, unless

- no redundant fluent is created
- the created nodes are never used in a subjet or object position in a new fluent assertion