Efficient Temporal Inference in RDF Stores

Ian Emmons and Matthew Hale
Raytheon BBN Technologies, Arlington, VA, USA
{iemmons,mhale}@bbn.com

***Abstract*—We propose ...**

***Index Terms*—Temporal Inference, Temporal Query, Temporal Index, RDF, OWL Time, SPARQL**

I. Introduction

Data with temporal aspects is common. In fact, one might go so far as to say that non-temporal data is relatively uncommon. Data encoded with the Resource Description Framework (RDF) is no exception. However, RDF and its query language, the SPARQL Protocol and RDF Query Language (SPARQL), have relatively few facilities with which to conveniently and efficiently handle temporal data:

- RDF literals of type `xsd:dateTime` and other time-related XML Schema Definition Language (XSD) data types, and
- SPARQL filter expressions to compare such literals.

Such rudimentary tools force us to write verbose and clumsy queries. Allen [1] correctly observed that time intervals are a more useful way to think about time, and this notion aligns conveniently with the prevalence of events in ontologies. Allen's Interval Algebra provides a convenient and concise expression of the various relationships between time intervals, allowing us to think of intervals as atomic entities, rather than as pairs of instants.

In this paper, we aim to present a well defined, concise, and flexible syntax with which to represent time intervals in RDF, together with an approach by which an RDF store and query processor may efficiently answer queries involving Allen's Interval Algebra. In addition, we will extend Allen's collection of temporal predicates to instants so that the inevitable occurrences of non-proper time intervals and instantaneous data may be handled through the same facility.

II. Conventions

We will use the following RDF prefixes throughout this document:

ot: <http://www.w3.org/2006/time#>
pt: <http://bbn.com/ontology/parliament/temporal#>
xsd: <http://www.w3.org/2001/XMLSchema#>

Here `ot:` is for the Web Ontology Language (OWL) Time ontology, `pt:` is for a new ontology we will introduce below, called "Parliament Time", and `xsd:` is for the standard XSD datatypes that are commonly used in RDF.

III. Related Work

A. Temporal Data Representation

Many RDF-based systems represent temporal facts with an ad hoc collection of properties that express important dates or times as `xsd:date` or `xsd:dateTime` literals. For instance, there might be birth and death properties for entities of type person, and start and end properties for a meeting.

Sometimes, rather than using `xsd:date` or `xsd:dateTime` literals, an ontology will incorporate the OWL Time ontology [2] to represent dates and times. This allows a fully structured representation, `DateTimeDescription`, in which the components of the date and time (year, month, etc.) are broken out as separate RDF facts. This approach can be helpful in some situations, but it is verbose and complicates some queries, because the query must compare all the components of the time individually.

Some ontologies formalize these sorts of concepts as events. The ontological structure of events is a complex topic, but here we are concerned only with the representation of the temporal extent of the events. In most such ontologies, the temporal extent is still represented in a simple way, as above, using something like start and end properties and `xsd:date` or `xsd:dateTime` literals or OWL Time's `DateTimeDescription`.

Older versions of Raytheon BBN Technologies' (BBN's) Parliament triple store recognized time intervals declared using the constructs of OWL Time. Many first-time users of this ontology are initially surprised to discover that its primary construct is not an instant in time, but rather the time interval. For instance, the phrase "ten o'clock" could be interpreted to mean the instant in time at the top of the hour, but in OWL Time it means the sixty-minute interval of time that starts at the top of the hour. OWL Time adopts consistently carries this point of view forward throughout. This means that it interprets an `xsd:dateTime` literal such as `2015-07-15T12:00:00Z` as a time interval of length one second beginning at noon on the indicated date.

Furthermore, although OWL Time does define the notion of an instant, it does not provide any way to specify one directly. Rather, it only allows us to say that a given instant is contained within an interval. By making that interval short, we can specify the instant more precisely, but we cannot directly specify the time of the instant. As a result of this cumbersome

arrangement, BBN's Parliament triple store specifyied an interval using the seven-statement OWL Time notation shown in Figure 1. (Note that this representation uses two of the Allen interval relations, "started by" and "meets".) This cumbersome expression of a time interval is expensive in both storage space and processing complexity.

The Geospatial SPARQL (GeoSPARQL) standard [3], although it is concerned with locations and regions rather than time instants and intervals, offers an interesting alternative. GeoSPARQL makes no attempt to break out the structure of regions in RDF. Instead, it defines literal datatypes that represent regions as strings that contain latitude-longitude pairs and other numeric parameters, as appropriate for the particular type of region. This means that an individual vertex of a polygon, for example, is not directly accessible in the RDF. To manipulate a vertex, one would have to retrieve the entire polygon and then parse the literal. In practice, however, this works out very well because GeoSPARQL enables queries that operate at a higher level, with the qualitative spatial reasoning of the Region Connection Calculus (RCC) predicates. Meanwhile, the minutia of the coordinates is handled under the covers by code.

B. Temporal Query constructs

When temporal data is encoded using `xsd:date` or `xsd:dateTime` literals, queries must compare times using inequalities in filter expressions.

- SPARQL filters

- Oracle's `xsd:dateTime` index

- Older Parliament index

- GeoSPARQL

IV. Background

In a typical RDF store, one expects to find the following functionality:

- Storage of RDF-encoded data,
- A SPARQL query processor, and
- An inference engine.

Such a facility is useful in a great many contexts, but there are some important types of queries that cannot be answered efficiently using a standard query processor and inference engine. One example is temporal reasoning. To understand why, suppose that a number of events are stored, each with a start and end time, and then a query is issued for all events that occur within given time interval. In order to answer such a query, a typical RDF store must do one of two things. Either it must retrieve every stored event and compare it to the given interval at query time, or it must materialize every possible interval that could be contained within each event when that event is stored. The first approach is very slow for a large triple store, and the second requires materializing an effectively infinite number of intervals.

Another similar example is geospatial reasoning. In this case, SPARQL was extended by the GeoSPARQL standard to enable efficient geospatial reasoning in a standard way that fits very nicely into RDF and related standards. In this paper, we propose a similar mechanism for temporal reasoning.

and temporal reasoning [4, 5], which to implement via forward chaining would effectively require an infinite number of statements to be materialized. To support these, Parliament indexes certain triples that are spatial or temporal in nature at insert time, and then consults the indexes at query time to determine possible matches. This can be viewed as a hybrid implementation of inference, in which the indexing is a partial forward chaining action, and the consultation of the indexes is a partial backwards chaining.

In the past, Parliament's temporal indexing was based entirely on an OWL ontology called OWL Time [2] and the relationships that it references, taken from Allen's Interval Algebra [1]. While this approach was functional, it did require a cumbersome, seven-statement representation of a time interval. Since that time, BBN has implemented support for the GeoSPARQL standard [3] in Parliament, and in so doing has discovered the simplicity of typed literals for such situations. The goal of this document is to specify a new implementation of temporal indexing for Parliament based on this approach.

V. Introduction

Parliament™ is a general-purpose, standards-compliant triple store [6] designed expressly for the Semantic Web, which BBN makes available on an open-source basis.[1] Like most triple stores, Parliament services queries made via the SPARQL [7] and supports a modest but useful level of inference via forward chaining. However, in BBN's consulting work, we have found that there are some important inferences that cannot be achieved using a forward chaining engine. Two examples supported by Parliament are geospatial and temporal reasoning [4, 5], which to implement via forward chaining would effectively require an infinite number of statements to be materialized. To support these, Parliament indexes certain triples that are spatial or temporal in nature at insert time, and then consults the indexes at query time to determine possible matches. This can be viewed as a hybrid implementation of inference, in which the indexing is a partial forward chaining action, and the consultation of the indexes is a partial backwards chaining.

In the past, Parliament's temporal indexing was based entirely on an OWL ontology called OWL Time [2] and the relationships that it references, taken from Allen's Interval Algebra [1]. While this approach was functional, it did require a cumbersome, seven-statement representation of a time

[1] http://parliament.semwebcentral.org/

```
:interval1 a ot:ProperInterval ;
   ot:intervalStartedBy [
      a ot:DateTimeInterval ;
      ot:xsdDateTime "2013-12-10T11:00:00Z"^^xsd:dateTime
   ] ;
   ot:intervalMeets [
      a ot:DateTimeInterval ;
      ot:xsdDateTime "2013-12-10T15:00:00Z"^^xsd:dateTime
   ] .
```

Fig. 1. OWL Time Interval Representation

interval. Since that time, BBN has implemented support for the GeoSPARQL standard [3] in Parliament, and in so doing has discovered the simplicity of typed literals for such situations. The goal of this document is to specify a new implementation of temporal indexing for Parliament based on this approach.

VI. Understanding the OWL Time Approach

OWL Time is commonly used to express temporal concepts in RDF. Many first-time users of this ontology are initially surprised to discover that its primary construct is not an instant in time, but rather the time interval. Upon reflection, however, most people agree that this is appropriate, because most expressions of time really are intervals. For instance, the phrase "ten o'clock" could be interpreted to mean the instant in time at the top of the hour, but much more commonly it means the sixty-minute interval of time that starts at the top of the hour.

OWL Time adopts consistently carries this point of view forward throughout the ontology. This means that it interprets a date-time literal such as

```
2015-07-15T12:00:00Z
```

as a time interval of length one second beginning at noon on the indicated date. (As is often the case in RDF, we express a specific time using the syntax of the XSD data type `xsd:dateTime`.)

Furthermore, although OWL Time does define the notion of an instant, it does not provide any way to specify one directly. Rather, it only allows us to say that a given instant is contained within an interval. By making that interval short, we can specify the instant more precisely, but we cannot directly specify the time of the instant. As a result of this cumbersome arrangement, we at BBN have taken to specifying an interval using the seven-statement OWL Time notation shown in Figure 1. Note that this representation uses two of the Allen interval relations [1], "started by" and "meets". All thirteen of Allen's relations are shown graphically in Table I.

This cumbersome expression of a time interval is expensive in both storage space and processing complexity. In the remainder of this document we will explore an alternative representation.

VII. Goals of the New Implementation

We aim to extend SPARQL only through the two extension points defined by that standard. See Section 18.7, "Extending SPARQL Basic Graph Matching" and Section 17.6, "Extensible Value Testing" for details [7].

We also want to ensure that queries can simultaneously use this query extension with GeoSPARQL.

VIII. Temporal Data Format

The proper way to express intervals and instances using the Parliament temporal index is to use the `pt:interval` and `pt:instant` predicates to relate a `ot:ProperInterval` resource to a corresponding interval or instance literal. The syntax of these literals is explained below.

The Parliament temporal index contains definitions for its own interval and instance literals, `pt:intervalLiteral` and `pt:TimeInstant`. `pt:intervalLiteral`s themselves are expressed in the form "<begin>, <end>" . Both <begin> and <end> follow the XSD datatype format for DateTime, and they are separated by a comma followed by a space. Instances are expressed as an individual DateTime literal. Remember that all of these literals must have `^^pt:intervalLiteral` or `^^pt:TimeInstant` appended to the literal for it to be recognized as an interval or instant. The example shown in Figure 2 applies all of the aforementioned information on how to express an interval using Parliament's temporal index. In this example, `:interval2` is the interval, inferred to be a `ot:ProperInterval` and assigned the predicate `pt:asInterval`. The predicate connects `:interval2` to a literal which represents a time interval according to the Parliament namespace.

IX. Temporal Queries

Queries for temporal data within a Parliament triple store utilize Allen's interval relations, which are listed with their pictorial definitions in Table I. These relations can be used in the `WHERE` clause of a query to dramatically simplify queries for temporal data. For example, this query:

```
select ?x where {
   ?x ot:intervalOverlaps :interval2
```

```
:interval2 pt:asInterval
   "2015-06-07T10:32:14Z,2015-06-08T04:05:12Z"^^pt:intervalLiteral .
```

Fig. 2. New Time Interval Representation

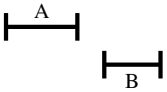
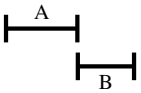
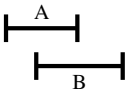
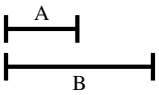
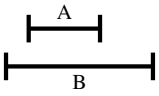
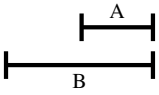
Relation(s)	Illustration
A ot:intervalEquals B	
A ot:intervalBefore B B ot:intervalAfter A	
A ot:intervalMeets B B ot:intervalMetBy A	
A ot:intervalOverlaps B B ot:intervalOverlappedBy A	
A ot:intervalStarts B B ot:intervalStartedBy A	
A ot:intervalDuring B B ot:intervalContains A	
A ot:intervalFinishes B B ot:intervalFinishedBy A	

TABLE I
ALLEN'S INTERVAL RELATIONS

```
}
```

returns all intervals that overlap with `:interval2` from Figure 2. Note that the above relations cannot be used with interval literals. As defined in the OWL Time namespace, these interval relation predicates function exclusively with `ProperInterval` resources, including those whose properties are defined by `pt:interval` or `pt:instant`.

X. The Parliament Time Implementation

The aim of Parliament's temporal index is to represent temporal data in a simple, intuitive, and holistic manner. In order to achieve this goal, several resources from the OWL Time ontology are used with resources.

Although the aforementioned Allen Time predicates enumerate all possible relations between two proper intervals, they fail to identify temporal relations that involve instants. For this reason, the Parliament temporal index must incorporate additional predicate functions in order to achieve an ideal level of functionality.

In addition to the Allen Time predicates, there are currently five unique predicates and two unique datatypes supported within a Parliament triple store for specifying temporal data. All but two of these additional resources originate from the OWL Time ontology. In order to denote a basic temporal inequality between any two temporal entities, queries against a Parliament knowledge base must use either `ot:before` or `ot:after`. These two properties are the respective super-properties of `ot:IntervalBefore` and `ot:IntervalAfter`, and in addition their domains and ranges are not constrained to resources of type `ot:ProperInterval` — they may be used to relate any two resources which are both temporal entities, including instants.

XI. Theoretical Analysis

A. Worst Case Analysis

B. Average Case Analysis

XII. Empirical Analysis

XIII. Conclusions

XIV. Glossary

BBN	Raytheon BBN Technologies , Inc. (pp. 1–3)
GeoSPARQL	Geospatial SPARQL (pp. 2, 3)
OWL	Web Ontology Language (pp. 1–4)
Parliament	Parliament™ is BBN's triple store, so named because "parliament" is the collective noun for a group of owls. A triple store is a specialized database tuned to the unique needs of the Semantic Web data representation. (pp. 1–4)
RCC	Region Connection Calculus (p. 2)
RDF	Resource Description Framework (pp. 1–3)
SPARQL	SPARQL Protocol and RDF Query Language . (If this seems confusing, it is because SPARQL is a recursive acronym.) (pp. 1–3)
XSD	XML Schema Definition Language (pp. 1, 3)

XV. References

[1] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983, ISSN: 0001-0782. DOI: 10.1145/182.358434. [Online]. Available: http://doi.acm.org/10.1145/182.358434 (cit. on pp. 1–3).

[2] J. R. Hobbs and F. Pan, "Time ontology in owl," World Wide Web Consortium (W3C), First Public Working Draft, Sep. 27, 2006. [Online]. Available: http://www.w3.org/TR/owl-time/ (cit. on pp. 1, 2).

[3] M. Perry and J. Herring, "Ogc geosparql - a geographic query language for rdf data," Open Geospatial Consortium, OGC Standard OGC 11-052r4, version 1.0, Sep. 10, 2012. [Online]. Available: http://www.opengeospatial.org/standards/geosparql (cit. on pp. 2, 3).

[4] R. Battle and D. Kolas, "Enabling the geospatial semantic web with parliament and geosparql," *Semantic Web*, vol. 3, no. 4, pp. 355–370, Oct. 2012. [Online]. Available: http://www.semantic-web-journal.net/sites/default/files/swj176_2.pdf (cit. on p. 2).

[5] D. Kolas, "Spatiotemporal indexing in parliament with jena," Raytheon BBN Technologies, Jena Users Workshop, 2010 Semantic Technology Conference, San Francisco, CA, Report, Jun. 23, 2010. [Online]. Available: http://asio.bbn.com/2010/06/semtech/Parliament_Spatiotemporal_Indexing.pdf (cit. on p. 2).

[6] D. Kolas, I. Emmons, and M. Dean, "Efficient Linked-List RDF Indexing in Parliament," in *Proceedings of the Fifth International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2009)*, (Washington, DC), ser. Lecture Notes in Computer Science, Springer, Oct. 2009, pp. 17–32. [Online]. Available: http://ceur-ws.org/Vol-517/ (cit. on p. 2).

[7] S. Harris, A. Seaborne, and E. Prud'hommeaux, "SPARQL 1.1 Query Language," W3C, Recommendation, version 1.1, Mar. 21, 2013. [Online]. Available: http://www.w3.org/TR/sparql11-query/ (cit. on pp. 2, 3).