Temporal Aggregates for Web Services on the Semantic Web

Feng Pan
Information Sciences Institute, University of Southern California
pan@isi.edu

Abstract

In this paper we describe how we encode our temporal aggregates ontology in OWL for Web services on the Semantic Web. We also present one example to show how to use the ontology to represent temporal aggregates information.

1. Introduction

Temporal information is everywhere in Web services [1,2,6], such as temporal availability of services (e.g., "an advertised service is available from 01/01/2004 to 01/15/2005" [1]), temporal constraints on the user's preferences, and so on. Temporal aggregates are very useful for Web services, for example, "the customer service is available from 8am to 5pm EST every working day between 01/01/2004 to 01/15/2005 [1]".

In response to this need, in conjunction with OWL-S [5], a temporal ontology, OWL-Time [3,6] (formerly DAML-Time), has been developed for describing the temporal content of Web pages and the temporal properties of Web services, as required for Semantic Web applications. This paper more focuses on the OWL encodings of the temporal aggregate ontology in OWL-Time. The complete first-order logic (FOL) axiomatization of the ontology can be found in [7].

2. Temporal Aggregates Ontology

The predicate `everynthp` says that a temporal sequence s consists of every nth element of the temporal sequence s_0 for which property p is true (see [7] for its definition axiom): `everynthp(s,s₀,p,n)`

For example, `everynthp(s,s₀,Monday1,2)` defines a temporal sequence s of "every other Monday". The context temporal sequence s_0 is useful not only to constrain s in a particular segment of time, but also to express complex *multiple-layered* temporal aggregates.

The property p is not limited only to simple temporal properties. In theory it can be any temporal properties. For example, in *conditional* temporal aggregates "every 3rd rainy day that's not a holiday", p is the unary predicate representing "rainy day that's not a holiday".

In order to map easily between OWL-Time and iCalendar, we have introduced the predicate `byTulistRecurs` which is a special predicate for handling temporal aggregates that only involve temporal units (see [7] for its definition axiom): `byTulistRecurs (s,ls,s',tu,tu')`

It says that a temporal sequence s consists of a list (ls) of elements with temporal unit tu of the temporal sequence s' whose temporal unit is tu'. For example, `byTulistRecurs (s,{1, 5, 20},s',*Week*,*Year*)` defines a temporal sequence s of "every 1st, 5th and 20th weeks of a sequence (s') of years".

In order to encode the temporal aggregates ontology from first-order logic axioms to OWL which is based on description logic, we defined three classes: temporal sequence, temporal sequence member, and temporal aggregate description. The outline of the structure of the classes is shown as follows. See [4] for the complete OWL encodings of the ontology.

TemporalSeq
-- hasMember → TemporalThing
-- hasTemporalAggregateDescription
→ TemporalAggregateDescription

TemporalSeqMember
subClassOf: TemporalThing
-- isMemberOf → TemporalSeq (card = 1)
-- hasPosition → integer (card = 1)

TemporalAggregateDescription
-- hasStart → InstantThing
-- hasEnd → InstantThing
-- hasContextTemporalSeq → TemporalSeq
-- hasithTemporalUnit → positive integer (card >= 1)
-- hasTemporalUnit → TemporalUnit (card = 1)
-- hasContextTemporalUnit → TemporalUnit
-- hasPosition → integer
-- hasGap → positive integer
-- hasCount → positive integer

"card" means cardinality. Only the cardinality of the required properties is shown in the above outline. We first defined temporal sequence. Since we want to have a backward link pointing from the temporal sequence member to its associated sequence, a `TemporalSeq-Member` class is defined. It has a required pair of properties: `isMemberOf` and `hasPosition`, so that it can not only point back to the associated sequence but also locate itself in the sequence.

The most important class in the OWL encodings of the temporal aggregates ontology is the temporal aggregate description class. Analogous to the calendar-clock description class [6], it specifies the temporal aggregate description for temporal sequences, and it's associated with the temporal sequence class by `hasTemporalAggregateDescription` property.

The optional properties `hasStart` and `hasEnd` map from the temporal aggregate description to the instant thing, specifying the start and the end instants of a temporal sequence. The calendar and clock properties described in [6] can be used to specify the start and the end times or dates the instants are in.

The property `hasContextTemporalSeq` corresponds to s_0 in `everynthp(s,s₀,p,n)` and *s'* in `byTulist-Recurs(s,ls,s',tu,tu')`. When it's not present, context-free temporal aggregates (e.g. "every Monday") can be represented.

The property `hasithTemporalUnit` corresponds to *ls* in `byTulistRecurs(s,ls,s', tu,tu')`. Thus it's very possible to have many such property values for a given temporal sequence. For example, "every 3rd Monday, Tuesday, and Friday".

The property `hasTemporalUnit` and `hasContext-TemporalUnit` specify the temporal unit of the given temporal sequence and the context temporal sequence respectively. They correspond to *tu* and *tu'* in `byTulistRecurs(s,ls,s',tu,tu')`.

The property `hasPosition` specifies the position of the element in the temporal sequence. It's possible to have negative positions. For example, "the last Thursday in every November" would have `hasPosition` value of -1.

The property `hasGap` specifies the gap between the elements in the temporal sequence. If it's not present, the default value of 1 will be used, for example, as in "every Monday". It corresponds to *n* in `everynthp(s, s₀,p,n)`. The property `hasCount` specifies the cardinality or the size of the temporal sequence.

The following example shows how our ontology can be used to represent a two-layered temporal sequence (see [3,7] for the details about the FOL predicates). More complex multiple-layered and conditional temporal aggregates can be represented similarly.

Every other Monday in every 3rd month.

FOL:

$$(\exists\, s,s_1,s_2)\ [\mathrm{everynthp}(s_2,s_1,\mathrm{Month1},3) \wedge \mathrm{everynthp}(s,s_2,\mathrm{Monday1},2)]$$

$$\text{where}\ (\forall\, m)\ [\mathrm{Month1}(m) \equiv (\exists\, n,x)\ [\mathrm{calInt}(m,n,{*}\mathrm{Month}{*},x)]]$$
$$(\forall\, d)\ [\mathrm{Monday1}(d) \equiv (\exists\, w)\ [\mathrm{Monday}(d,w)]]$$

OWL:
```
<time-entry:TemporalSeq rdf:ID="tseq">
  <time-entry:hasTemporalAggregateDescription
          rdf:resource="#everyOtherMondayEvery3rdMonth" />
</time-entry:TemporalSeq>

<time-entry:TemporalSeq rdf:ID="tseq-every3rdMonth">
  <time-entry:hasTemporalAggregateDescription
          rdf:resource="#every3rdMonth" />
</time-entry:TemporalSeq>

<time-entry:TemporalAggregateDescription
                        rdf:ID="every3rdMonth">
  <time-entry:hasTemporalUnit
          rdf:resource="&time-entry;unitMonth" />
  <time-entry:hasGap rdf:datatype="&xsd;positiveInteger">3
</time-entry:TemporalAggregateDescription>

<time-entry:TemporalAggregateDescription
          rdf:ID="everyOtherMondayEvery3rdMonth">
  <time-entry:hasContextTemporalSeq
          rdf:resource="#tseq-every3rdMonth" />
  <time-entry:hasithTemporalUnit
          rdf:datatype="&xsd;positiveInteger">1
  </time-entry:hasithTemporalUnit>
  <time-entry:hasTemporalUnit
          rdf:resource="&time-entry;unitDay" />
  <time-entry:hasContextTemporalUnit
          rdf:resource="&time-entry;unitMonth" />
  <time-entry:hasGap rdf:datatype="&xsd;positiveInteger">2
  </time-entry:hasGap>
</time-entry:TemporalAggregateDescription>
```

3. References

[1] Dumas, M., J. O'Sullivan, M. Heravizadeh, D. Edmond, and A. Hofstede. Towards a semantic framework for service description. *In Proceedings of the IFIP Conference on Database Semantics*, Hong Kong, April 2001.

[2] McIlraith, S. A., Son, T. C. and Zeng, H. Semantic Web Services. *IEEE Intelligent Systems* 16(2):46–53, 2001.

[3] Hobbs, J. R. and Pan, F. An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Processing (TALIP)* Vol. 3, No. 1, pp. 66-85, 2004.

[4] OWL encodings of the temporal aggregates ontology in OWL-Time: http://www.isi.edu/~pan/damltime/TemporalAggregates.owl

[5] OWL-S Coalition. OWL-S 1.1 Release. (http://www.daml.org/services/owl-s/1.1/)

[6] Pan, F. and Hobbs, J. R. Time in OWL-S. *In Proceedings of the AAAI Spring Symposium on Semantic Web Services*, Stanford University, CA, 2004.

[7] Pan, F. and Hobbs, J. R. Temporal Aggregates in OWL-Time. *In Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, Clearwater Beach, Florida, 2005.