Time in OWL-S

Feng Pan and Jerry R. Hobbs

University of Southern California / Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
{pan, hobbs}@isi.edu

Abstract
To provide support for describing temporal properties for OWL-S (formerly DAML-S), in this paper we first introduce an “entry” sub-ontology of time, which is much simpler than the full ontology of DAML-Time1 (Hobbs 2002) and provides the basic temporal concepts and relations that most simple applications would need, i.e., a vocabulary for expressing facts about topological relations among instants, intervals, and events, together with information about durations, and about dates and times. Then we demonstrate in detail, using the Congo.com and Bravo Air examples, how this entry sub-ontology of time can be used to support OWL-S, including use cases for defining input parameters and (conditional) output parameters. Segments of OWL encoding of the definitions are shown.

Introduction

OWL-S (formerly DAML-S) is an OWL-based Web service ontology that supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form (The DAML Services Coalition 2003). The latest 0.9 draft release is expected to be the last one built on DAML+OIL (Connolly et al. 2001), and the later releases will be based on OWL (Dean et al. 2003).

Temporal information is so common that it’s hard to find a real world Web service without it. For example, whenever you place an online order, the order date is always part of your order. When you reserve a car at a car rental site, you have to specify the dates you need it. In response to this need, in conjunction with OWL-S, a temporal ontology, DAML-Time (Hobbs 2002), has been developed for describing the temporal content of Web pages and the temporal properties of Web services. Its development is being informed by temporal ontologies developed at a number of sites and is intended to capture the essential features of all of them and make them and their associated resources easily available to a large group of Web developers and users.

An Entry Sub-Ontology of Time

DAML-Time is an abstract ontology of time intended to be a complete specification of a theory of time as required for Semantic Web (Berners-Lee et al. 2001) applications. Included in it is a rich collection of axioms that tightly constrain the interpretation of the predicates and functions of the theory. But for most simple applications this is far more than is required, and its complexity constitutes a barrier to the use of the ontology. The purpose of this entry sub-ontology of time is to provide quick access to the essential vocabulary in OWL for the basic temporal concepts and relations. The OWL encoding of the entire entry sub-ontology can be found online².

This entry sub-ontology of time covers topological relations among instants and intervals and instant-like and interval-like events (called instant events and interval events), such as “before” and “overlaps”. It includes measures for durations so that we can say a meeting will last 1 hour and 30 minutes, and it also includes clock and calendar terms so that we can say a meeting starts at 3:00pm PST on Monday, October 20, 2003.

A simple use case³ in OWL is presented for the sub-ontology: “Suppose someone has a telecon scheduled for 6:00pm EST on November 5, 2003. You would like to make an appointment with him for 2:00pm PST on the

---

1 DAML-Time Homepage:
http://www.cs.rochester.edu/~ferguson/daml/

2 http://www.isi.edu/~pan/damltime/time-entry-documentation.txt

3 http://www.isi.edu/~pan/damltime/time-entry.owl

4 http://www.isi.edu/~pan/damltime/time-entry-case1.owl
same day, and expect the meeting to last 45 minutes. Will there be an overlap?” In this use case we specify the facts about the telecon and the meeting using our ontology in OWL that will allow a temporal reasoner to determine whether there is a conflict. We have been in contact with several sites about linking our ontology with their temporal reasoners.

**Topological Temporal Relations**

The most basic temporal concepts in the sub-ontology are Instant, Interval, Instant Event, and Interval Event. Instants are, intuitively, point-like in that they have no interior points, and intervals are, intuitively, things with extent. Instant events are events that are instantaneous, such as the occurrence of a car accident or the arrival of a package, and interval events are events that span some time interval, for example, a meeting from 2pm to 3pm.

Besides these four basic temporal concepts, there are five other more general temporal concepts/classes: Temporal Thing, Temporal Entity, Instant Thing, Interval Thing, and Event. The subclass hierarchy of these temporal concepts/classes is shown in the Figure 1. For example, Instant Thing has two subclasses: Instant and Instant Event.

![Figure1: Subclass hierarchy of temporal concepts](image)

Their axiom definitions are straightforward. For example, the axiom to define that Temporal Entity has only two subclasses, Instant and Interval, are:

\[
\text{TemporalEntity}(T) \leftrightarrow \text{Instant}(T) \lor \text{Interval}(T)^3
\]

The above axioms can be expressed in OWL as:

```xml
<owl:Class rdf:ID="TemporalEntity">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Instant"/>
    <owl:Class rdf:about="#Interval"/>
  </owl:unionOf>
</owl:Class>
```

Other temporal concepts and their relationships can be defined similarly.

**begins and ends are relations between instant things and temporal things, and the beginnings and ends of temporal entities, if they exist, are unique. In some approach to infinite intervals, a positively infinite interval has no end, and a negatively infinite interval has no beginning. Hence, we use the relations begins and ends in the ontology, rather than defining functions beginning-of and end-of, since the functions would not be total. The axioms that characterize begins, for example, are:**

\[
\text{begins}(t,T) \rightarrow \text{Instant}(t) \land \text{Temporal}(T)
\]

\[
\text{TemporalEntity}(T) \land \text{begins}(t1,T) \land \text{begins}(t2,T) \rightarrow t1 \neq t2
\]

These properties can be expressed in OWL as:

```xml
<owl:ObjectProperty rdf:ID="begins">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#TemporalThing"/>
  <rdfs:range rdf:resource="#InstantThing"/>
</owl:ObjectProperty>
```

**inside** is a relation between an instant thing and an interval thing, and it is not intended to include beginnings and ends of intervals.

We can define a proper-interval as one whose beginning and end are not identical:

\[
(\forall T)(\text{ProperInterval}(T) \leftrightarrow \text{Interval}(T) \land (\exists t1,t2)[\text{begins}(t1,T) \land \text{ends}(t2,T) \land t1 \neq t2])
\]

A half-infinite or infinite interval, by this definition, is proper. The ontology is silent about whether there are any intervals that are not proper intervals.

There is a before relation on temporal things, which gives directionality to time. If a temporal thing \(T1\) is before another temporal thing \(T2\), then the end of \(T1\) is before the beginning of \(T2\). Thus, before can be considered to be basic to instant things and derived for interval things:

\[
(\forall T1,T2)(\text{before}(T1,T2) \leftrightarrow (\exists t1,t2)[\text{ends}(t1,T1) \land \text{begins}(t2,T2) \land \text{before}(t1,t2)])
\]

The before relation is anti-reflexive, anti-symmetric and transitive.

**Interval Relations.** The relations between intervals defined in Allen's temporal interval calculus (Allen 1984)
can be defined in a relatively straightforward fashion in terms of before and identity on the beginning and end points. The standard interval calculus assumes all intervals are proper, and we do that here too, but we generalize proper intervals to proper interval things.

Axioms are defined for the interval relations: intEquals, intBefore, intMeets, intOverlaps, intStarts, intDuring, intFinishes, and their reverse interval relations: intAfter, intMetBy, intOverlappedBy, intStartedBy, intContains, intFinishedBy. For example, the definition of intEquals is:

\[
\forall T_1, T_2 \quad \text{intEquals}(T_1, T_2) \iff \exists T \quad \text{ProperIntervalThing}(T) \land \text{ProperIntervalThing}(T_1) \land \text{ProperIntervalThing}(T_2) \\
\land (\forall t_1 \quad \text{begins}(t_1, T_1) \iff \text{begins}(t_1, T_2)) \\
\land (\forall t_2 \quad \text{ends}(t_2, T_1) \iff \text{ends}(t_2, T_2)).
\]

**Duration Description**

The duration of an interval (or temporal sequence) can have many different descriptions. An interval can be 1 day 2 hours, or 26 hours, or 1560 minutes, and so on. It is useful to be able to talk about these descriptions in a convenient way as independent objects, and to talk about their equivalences. We do this first in terms of a predicate called "duration-of" that takes eight arguments, one for a temporal thing, and one each for years, months, weeks, days, hours, minutes, and seconds. Then we will define a specific kind of individual called a "duration description", together with a number of functions relating the duration description to the values of each of the eight arguments. Thereby we convert the 8-ary predicate "duration-of" into eight binary relations that are more convenient for description logic-based markup languages, such as OWL. Here is the definition of the duration description:

\[
\forall T, y, m, w, d, h, n, s \quad \text{duration-of}(T, y, m, w, d, h, n, s) \iff \exists d_1 \quad \text{DurationDescription}(d_1, T) \\
\land \text{years-of}(d_1) = y \\
\land \text{months-of}(d_1) = m \\
\land \text{days-of}(d_1) = d \\
\land \text{hours-of}(d_1) = h \\
\land \text{minutes-of}(d_1) = n \\
\land \text{seconds-of}(d_1) = s).
\]

It can be expressed in OWL as:

```xml
<owl:Class rdf:ID="DurationDescription">
  <owl:subClassOf>
    <owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#years" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
      </owl:Restriction>
    </owl:Restriction>
    <owl:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#months" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
      </owl:Restriction>
    </owl:Restriction>
    ...<owl:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#seconds" />
      </owl:Restriction>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>
```

The axiom saying that instant things have 0 duration is:

\[
\text{InstantThing}(t) \rightarrow \text{duration-of}(t, 0, 0, 0, 0, 0, 0, 0, 0).
\]

There are two ways to specify the duration description of a temporal thing. The relation `durationDescriptionOf` uses `DurationDescription` as its range, while `durationDescriptionDataType` uses the XML Schema datatype `duration` as its range:

```xml
<owl:ObjectProperty rdf:ID="durationDescriptionOf">
  <rdfs:domain rdf:resource="#TemporalThing" />
  <rdfs:range rdf:resource="#DurationDescription" />
</owl:ObjectProperty>
```

Using the XML Schema datatype `duration` is simpler and more standard. But it can’t specify weeks, which can be specified by using `DurationDescription`. Using `DurationDescription` also makes it easier to extract values from any field for the later use. The user has the freedom to choose either of these two properties/relations to specify a duration description for a temporal thing.

**Time Zones**

What hour of the day an instant is in is relative to the time zone. This is also true of minutes, since there are regions in the world, e.g., central Australia, where the hours are not aligned with GMT hours, but are, e.g., offset half an hour. Probably seconds are not relative to the time zone.

Days, weeks, months and years are also relative to the time zone, since, e.g., 2003 began in the Eastern Standard time zone three hours before it began in the Pacific Standard time zone. Thus, predications about all clock and calendar intervals except seconds are relative to a time zone.

We have been referring to time zones, but in fact it is more convenient to work in terms of what we might call the "time standard" that is used in a time zone. That is, it is better to work with the Pacific Standard Time (PST) as a legal entity than with the PST zone as a geographical region. A time standard is a way of computing the time, relative to a world-wide system of computing time. For each time standard, there is a zone, or geographical region, and a time of the year in which it is used for describing local times. Where and when a time standard is used have

6 [http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#duration](http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#duration)
to be axiomatized, and this involves interrelating a time ontology and a geographical ontology. These relations can be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.

If we were to conflate time zones (i.e., geographical regions) and time standards, it would likely result in problems in several situations. For example, the Eastern Standard zone and the Eastern Daylight zone are not identical, since most of Indiana is on Eastern Standard time all year. The state of Arizona and the Navajo Indian Reservation, two overlapping geopolitical regions, have different time standards during the daylight saving times -- one is Pacific and the other is Mountain.

Time standards that seem equivalent, like Eastern Standard and Central Daylight, should be thought of as separate entities. Whereas they function the same in the time ontology, they do not function the same in the ontology that articulates time and geography. For example, it would be false to say those parts of Indiana that would be quite complex. Only the entities like PST and EDT, the time standards, are part of the time ontology.
calendar-clock description in OWL for specifying both calendar and clock information for a calendar-clock interval.

A calendar-clock description has the following properties/fields: unitType, year, month, week, day, dayOfWeek, dayOfYear, hour, minute, second, and time zone. The property unitType specifies the temporal unit type of the calendar-clock description, and its domain is TemporalUnit:

```
<owl:Class rdf:ID="TemporalUnit">
  <owl:oneOf rdf:parseType="Collection">
    <TemporalUnit rdf:about="#unitSecond" />
    <TemporalUnit rdf:about="#unitMinute" />
    <TemporalUnit rdf:about="#unitHour" />
    <TemporalUnit rdf:about="#unitDay" />
    <TemporalUnit rdf:about="#unitWeek" />
    <TemporalUnit rdf:about="#unitMonth" />
  </owl:oneOf>
</owl:Class>
```

For example, the temporal unit type of 10:30 is minute (unitMinute), and the temporal unit type of March 20, 2003 is day (unitDay). The unit type is required. With a given temporal unit type, all the fields/properties for smaller units will be ignored. For instance, if the temporal unit type is day (unitDay), the values of the field/property hour, minute, and second, if present, will be ignored.

Since calendar-clock description is for describing calendar-clock intervals, we defined a property, called calendarClockDescriptionOf with CalendarClockDescription as the range, for calendar-clock intervals.

To express cal-int(12Mar2002,12,*Day*,Mar2002), for example, using calendar-clock description, we need an instance of CalendarClockDescription-Description that has values only for unitType (unitDay), year (2002), month (3), and day (12). clock-int(10:03,3,*Minute*,[10:00, 11:00)) can be expressed similarly.

CalendarClockDescription and calendarClockDescriptionOf are defined in OWL as:

```
<owl:Class rdf:ID="CalendarClockDescription">
  <owl:subClassOf>
    <owl:ObjectProperty rdf:resource="#unitType" />
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
  </owl:subClassOf>
  <owl:Restriction>
    <owl:oneOf>
      <owl:Restriction>
        <owl:oneOf>
          <owl:Restriction>
            <owl:oneOf>
              <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
          </owl:Restriction>
        </owl:oneOf>
      </owl:Restriction>
    </owl:oneOf>
  </owl:Restriction>
</owl:Class>
```

In order to specify that an instant thing is in a calendar-clock interval, an inCalendarClock property/relation is defined similarly to calendarClockDescriptionOf as follows:

```
<owl:ObjectProperty rdf:ID="inCalendarClock">
  <rdfs:domain rdf:resource="#InstantThing" />
  <rdfs:range rdf:resource="#CalendarClockDescription" />
</owl:ObjectProperty>
```

With this inCalendarClock relation, we can say that an instant thing is at a specific calendar-clock time. For example, the beginning of a meeting, which is an instant, is at 6:00pm which is actually in a calendar-clock interval of [6:00:00, 6:01:00).

We also defined in OWL two simpler relations, calendarClockDescriptionDatatype and inCalendarClockDatatype. Similar to durations, the only difference between these two relations and the above calendarClockDescriptionOf and inCalendarClock relations is their ranges: these two simpler relations use the XML Schema datatype dateTime as their ranges, while the above uses CalendarClockDescription:

```
<owl:DatatypeProperty rdf:ID="calendarClockDescriptionDatatype">
  <rdfs:domain rdf:resource="#IntervalThing" />
  <rdfs:range rdf:resource="&xsd;dateTime" />
</owl:DatatypeProperty>
```

To illustrate more clearly the difference between using CalendarClockDescription and using the XML datatype dateTime, let’s look at a concrete example: an instant, called “instantExample”, at 10:30am EST on 01/01/2003 can be expressed using both inCalendarClockData Type and inCalendarClock in OWL as:

```
<time:Instant rdf:ID="instantExample"><time:inCalendarClock rdf:resource="instantExampleDescription" />
  <time:maxCalendarDate rdf:resource="#dateTime" />
  <time:maxCalendarDate rdf:resource="&xsd;dateTime" />
  <time:Instant>
</time:Instant>
```

---

15 http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime
The namespace “time” points to the location of the current OWL encoding of our entry sub-ontology of time16. In this example Instant, a subclass of TemporalEntity, would be a better class to use than TemporalEntity to describe CreditCardExpirationDate, because the expiration date is actually an instant -- the midnight, of the day the credit card expires.

In the Bravo Air example, our time ontology can be used to describe the existing input parameters, DepartureDate and ArrivalDate. We will change this to the more appropriate DepartureTime and ArrivalTime. We can define DepartureTime in the profile of the Bravo Air example (i.e. BravoAirProfile.owl) as:

```
<profile:ParameterDescription rdf:ID="DepartureTime">
  <profile:input>  
    <profile:refersTo rdf:resource="&congoProcess;#departureDate_In"/>
  </profile:input>
</profile:ParameterDescription>
```

DepartureTime is defined as an Instant. With this definition, as we discussed in the previous calendar-clock description section, an instance of DepartureTime can has either an inCalendarClockDataType property/relation pointing to a specific value of XML Schema datatype dateTime, say 2003-01-01T10:30:00-5:00, or an inCalendarClock object-property/relation pointing to an instance of CalendarClockDescription class specifying a specific time, say 10:30am EST on 01/01/2003, Wednesday. It would be the user’s decision to define the time in either way based on the trade-offs discussed in last section.

**Use Cases for (Conditional) Output Parameters**

In fact, there is much more that our time ontology can do to support OWL-S. In the current Congo.com and Bravo Air examples, the time ontology is not used for any output parameters. However, in the real world many service outputs are time-related. For example, in the Congo.com example we can add two outputs that are very common in real world book-selling sites: process time and delivery duration.

**Adding a ProcessTime output parameter.** ProcessTime is a conditional output parameter that specifies how long before the book will be ready for delivery, say, 24 hours, which depends on whether the book is in stock. In this use case, the process time is returned only if the book is in stock. It can be defined in the process model of the Congo.com example (i.e. CongoProcess.owl) as:

```
<profile:ParameterDescription rdf:ID="departureDate_In"/>
<profile:ParameterDescription rdf:ID="outboundDate_In"/>
```

We can see from this example that it’s much simpler to use the XML Schema datatype, dateTime. However, the advantage of using CalendarClockDescription is that it can express more information than dateTime, such as "week", "day of week" and "day of year", so in the above example, we can also know that 01/01/2003 is Wednesday, on the first day of the year, and in the first week of the year. The namespace “tz-us" points to our US time zone data16. Moreover, each field of CalendarClockDescription is separate so that it's easier to extract the value of some fields for the later use and easier to reason about.

**Time in OWL-S**

Congo.com and Bravo Air are the two examples being used in the latest OWL-S 0.9 draft release (The DAML Services Coalition 2003). Congo.com is a fictitious book-selling service site, and Bravo Air is a fictitious airline-ticketing service site. We use these two examples to demonstrate in detail how our entry sub-ontology of time can be used to support OWL-S, including use cases for defining input parameters and (conditional) output parameters. Segments of OWL encoding of the definitions are shown.

**Use Cases for Input Parameters**

In the profile of the Congo.com example (i.e. CongoProfile.owl), for example, our time ontology is currently used for describing the input parameter CreditCardExpirationDate:

```
<profile:ParameterDescription rdf:ID="CreditCardExpirationDate">
  <profile:input>  
    <profile:refersTo rdf:resource="&congoProcess;#creditCardExpirationDate"/>
  </profile:input>
</profile:ParameterDescription>
```

16 http://www.isi.edu/~pan/damltime/timezone-us.owl

17 http://www.isi.edu/~pan/damltime/time-entry.owl
ProcessTime is defined as an Interval, rather than a duration. As discussed previously, in our time ontology durations are properties of intervals. Thus to talk about a duration, i.e. a quantity of time, an interval must be defined first. This approach may look roundabout at first glance. However, the process time is not purely a quantity of time; it has a location on the time line. The beginning of the process time is the time the user places the order, and the end of the process time is the time the order is shipped out. An advantage of defining ProcessTime as an interval is that if the relationship among the order time, the shipping time, and the process time is known, any one of them (e.g. the shipping time) can be computed from the other two (e.g. the order time and the process time) by temporal arithmetic.

Adding a DeliveryDuration output parameter. Delivery- Duration is a conditional output parameter that specifies how long it will take for the customer to receive the book after it is shipped out, which depends on the delivery type the customer selects. As defined in the process model of the Congo.com example (i.e. CongoProcess.owl), the current delivery types are FedExOneDay, FedEx2-3day, UPS, and OrdinaryMail.

To add this output parameter may seem similar to the above ProcessTime example. However, since an instance of Condition is a logical formula that evaluates to true or false (see the comment with the definition of Condition), DeliveryType cannot be directly used as a condition to determine the delivery duration. Thus one property and one condition are defined for each delivery type.

DeliveryDuration is defined with two boundaries: one minDeliveryDuration and one maxDeliveryDuration. For example, an order with the FedEx2-3day delivery type takes 2 to 3 days, so its min delivery duration is 2 days, and its max delivery duration is 3 days. For the delivery duration of the order with FedExOneDay delivery type, the min and max delivery duration will both be 1 day. We can define DeliveryDuration in the process model of the Congo.com example (i.e. CongoProcess.owl) as:

Both minDeliveryDuration and maxDeliveryDuration are defined as properties of DeliveryDuration. For the same reason discussed for the process time example, both properties use Interval as their ranges. The cardinality of 1 for both properties in the definition of DeliveryDuration indicates that an instance of DeliveryDuration must have one and only one property value for minDeliveryDuration and maxDeliveryDuration respectively.

For example, in order to define delivery duration for FedEx2-3day, we have to first define a condition of FedEx2-3day being selected:

Then we define an output property, called deliverySelectFedEx2-3day that is conditional on FedEx2-3dayCondition defined above:

18 http://www.daml.org/services/owl-s/0.9/Process.owl
This definition says that deliverySelectFedEx2-3day is a conditional output, and if FedEx2-3dayCondition is true, an instance of FedEx2-3dayDuration class will be the output. However, FedEx2-3dayDuration is not defined yet.

In order to define it, we have to define its min delivery duration, i.e. 2 days, and max delivery duration, i.e. 3 days. Since the range of minDeliveryDuration and maxDeliveryDuration is Interval, intervals with specific durations need to be created first. For FedEx2-3dayDuration, we need to define Interval2Days and Interval3Days first as follows:

<owl:Class rdf:ID="Interval2Days">
  <!-- intervals with a duration of 2 days -->
  <rdfs:subClassOf rdf:resource="&time;#Interval"/>
  <owl:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&time;#durationDescriptionDataType"/>
      <owl:hasValue rdf:datatype="&xsd;duration">P2D</owl:hasValue/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Interval3Days">
  <!-- intervals with a duration of 3 days -->
  <rdfs:subClassOf rdf:resource="&time;#Interval"/>
  <owl:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&time;#durationDescriptionDataType"/>
      <owl:hasValue rdf:datatype="&xsd;duration">P3D</owl:hasValue/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>

These two definitions use durationDescriptionDataType, a relatively simpler duration property of Interval using the XML Schema datatype duration, meaning 2 days and 3 days.

Finally, FedEx2-3dayDuration restricts the value of minDeliveryDuration and maxDeliveryDuration to class Interval2Days and Interval3Days respectively as follows:

<owl:Class rdf:ID="#FedEx2-3dayDuration">
  <rdfs:subClassOf rdf:resource="#DeliveryDuration"/>
  <owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#minDeliveryDuration"/>
      <owl:allValuesFrom rdf:resource="#Interval2Days"/>
    </owl:Restriction>
  </owl:Restriction>
  <owl:Restriction>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#maxDeliveryDuration"/>
      <owl:allValuesFrom rdf:resource="#Interval3Days"/>
    </owl:Restriction>
  </owl:Restriction>
</owl:Class>

Properties to output delivery durations when the user selects other delivery types (FedExOneDay, UPS, and OrdinaryMail) can be defined similarly.

Conclusion and Future Work

In this paper, we have demonstrated how the entry sub-ontology of time can support OWL-S. We believe it should be able to help describe most of the temporal properties of real world services, since they usually only require basic topological relations, and information about durations, dates and times.

In order to test our time ontology, it needs to be linked to a temporal reasoner that reasons about the query from the user using our ontology, and produces the result to the query.

We are currently axiomatizing temporal arithmetic for the time ontology, for example, adding a duration (e.g. one month and two days) to a date/time (e.g. January 30, 2004). A treatment of temporal aggregates will also be extracted from full DAML-Time to the entry sub-ontology, so that it can express something like “every other Wednesday”.

Acknowledgements

This research was funded in part by the Defense Advanced Research Projects Agency (DARPA) as part of the DAML program under Air Force Research Laboratory contract F30602-00-C-0168, and in part with funds from the Advanced Research and Development Agency (ARDA). The authors have profited from discussions with Ken Barker, Mike Dean, George Ferguson, Pat Hayes, Jessica Jenkins, David Martin, Drew McDermott, Ian Niles, James Pustejovsky, Tom Russ, James Zais.

References


The DAML Services Coalition 2003. DAML-S (and OWL-S) 0.9 Draft Release. http://www.daml.org/services/daml-s/0.9/


http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#duration