REPRESENTING COMPLEX TEMPORAL PHENOMENA FOR THE SEMANTIC WEB AND NATURAL LANGUAGE

by

Feng Pan

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

December 2007

Dedication

To my wife, parents and sister…

Acknowledgments

First of all, I would like to extend my heartfelt thanks to my advisor, Jerry Hobbs, for his continuous support and inspiring guidance during my graduate career. His encouragement greatly helped me bolster my confidence and also maintain a high motivation through the inevitable ups and downs of my PhD study. Without him I would not have accomplished what I have right now. I am also very grateful to other members of my thesis committee --Hans Chalupsky, Kevin Knight, Paul Rosenbloom, and Daniel O'Leary -- for their valuable comments and constructive criticism on my thesis proposal and draft.

During my graduate life at the ISD division of USC Information Sciences Institute (ISI), I have been very lucky to have many great mentors and colleagues. I would like to thank Craig Knoblock, Jim Blythe, Jeff Rickel (who has passed away, and we all miss him so much!), who have supervised me with different projects at ISI; I have learned from them not only broader knowledge of AI, but more importantly different ways of doing research. I have also learned a lot from our KR/NLP group meetings; special thanks to the ICT members of the meeting, Andrew Gordon, Wenji Mao, Reid Swanson, Erin Tavano, for their helpful feedbacks to my research and inspiring discussions. Many thanks to Rutu Mulkar-Mehta, who has collaborated with me under Jerry's supervision during the last two years, for her valuable contributions to the research. I am also greatly indebted to José Luis Ambite, Rahul Bhagat, Michael Bloodgood, Jim Blythe, Donghui Feng, Ulf Hermjakob, Shou-de Lin, Tom

Russ, Liang-Chih Yu for their very helpful feedbacks to my qualifying and defense presentations. There are many other fellow ISDers I would like to thank for their help and advice: Yigal Arens, Erika Barragan-Nunez, Harris Chi Ho Chiu, Tim Chklovski, Berthe Choueiry (visiting professor), Hal Daume III, Steven DeNeefe, Lei Ding, Alexander Fraser, Yolanda Gil, Eduard Hovy, Soo-Min Kim, Jihie Kim, Namhee Kwon, Brent Lance, Kary Lau, Chin-Yew Lin, Daniel Marcu, Jonathan May, Nick Mote, Alma Nava, Patrick Pantel, Karen Pressler, Lei Qu, Nadeesha Ranasinghe, Nishit Rathod, Deepak Ravichandran, Wei-Min Shen, Mei Si, Nathan Sobo, Radu Soricut, Snehal Thakkar, Quamrul Tipu, Ning Wang, Shumin Wu, Liang Zhou, Jingbo Zhu.

Finally, this thesis is dedicated to my family. I am so grateful to my parents and sister for their enormous support, endless love, and constant encouragement throughout my life. Special thanks to my parents-in-law and brother-in-law for their continuous support. My deepest love and gratitude to my lovely wife for her unconditional love, for always being there for me through the good times and the bad, and for making my life so colorful and joyful!

Table of Contents

List of Tables

List of Figures

Abstract

As an essential dimension of our information space, *time* plays a very important role in every aspect of our lives. A specification of temporal information is necessarily required for a large group of applications, including the Semantic Web and natural language. In response to this need, we have developed a rich *ontology of temporal concepts*, OWL-Time (formerly DAML-Time), for describing the temporal content of Web pages and the temporal properties of Web services. Since most of the information on the Web is in natural language, it can also be used for temporal reasoning and to increase the temporal awareness for different natural language applications. The ontology is represented in first-order logic (FOL) and the OWL Web Ontology Language.

The ontology covers a very rich set of temporal concepts. It extends Hobbs (2002)'s work with more complex temporal phenomena, such as temporal aggregates, temporal arithmetic mixing months and days, and vague event durations. We have also created axioms that map subsets of the problems that can be represented by the ontology in FOL to temporal constraint-based formalisms for more efficient temporal reasoning. The temporal aggregate part of the ontology is rich enough to handle both complex multiple-layered and conditional temporal aggregates. A systematic way of mapping recurrence sets in iCalendar (iCal) to temporal aggregates in OWL-Time was developed to give it access to the full ontology of time for temporal reasoning. A set of rules for temporal arithmetic mixing months and days were developed with consideration of different desired arithmetic properties, such as commutativity and associativity.

Since missing explicit and exact durations is one of the most common sources of incomplete information for temporal reasoning in natural language applications, we have constructed an annotated corpus to extract the implicit and vague event durations from text. We generated annotation guidelines, categorized the event classes to reduce gross discrepancies in inter-annotator judgments, used normal distributions to model event duration annotations that are intervals on a scale and to measure their inter-annotator agreement. Machine learning techniques were then applied to the annotated data and produced coarse-grained event duration information automatically, considerably outperforming a baseline and approaching human performance. The methods used here should be applicable to other kinds of vague but substantive information.

Chapter 1

Introduction

As an essential dimension of our information space, *time* plays a very important role in every aspect of our lives. A specification of temporal information is necessarily required for a large group of applications. This thesis builds on Hobbs (2002)'s time ontology work and extends it with complex temporal phenomena that has little study before. The goal is to provide an ontology of temporal concepts that is rich and expressive enough for the Semantic Web and natural language applications. As shown in Section 1.3, the ontology is actually useful for applications in a great variety of domains.

1.1 Time for the Semantic Web

The vision of the Semantic Web is to create a "meaningful" environment for programs and software agents to roam from page to page to carry out sophisticated tasks for users (Berners-Lee et al., 2001), and the specification of temporal information is necessarily required for bringing the Semantic Web into reality. For example, some program or software agent does a Web search for its user to find a place to buy a book needed before next Tuesday. In order to decide whether or not to use an online bookstore that promises delivery within five business days, the specification of temporal information is needed to represent durations in its service description (i.e., “delivery within five business days”), represent calendar dates and temporal relations for user’s constraints and preferences (i.e., “needed before next Tuesday”), do temporal arithmetic on calendar dates and durations (i.e., adding five business days to the current date, assuming shipping out immediately), and compare calendar dates to see whether it can arrive before next Tuesday.

Temporal information is everywhere on the Web, for example, in Web services, such as temporal availability of services (e.g., “an advertised service is available from 01/01/2004 to 01/15/2005” (Dumas et al. 2001)), temporal constraints on the user’s preferences (e.g., “I would prefer driving over flying if the driving time to my destination is less than three hours.” (McIlraith et al. 2001); “I would like to receive this book by next Monday.”), and so on.

In ubiquitous and pervasive computing, a time ontology is crucial for modeling and reasoning about the time dimension of the context (Bulcao Neto and Pimentel, 2005). In a smart meeting room application, for example, it can help to reason about the temporal orders among different events in a meeting room, and to model time intervals that may contain repeating time intervals or instants, which will be useful for representing recurrent events such as weekly meetings and classes (Chen et al., 2004). For a use case on scheduling:

> Suppose someone has <u>weekly</u> telecons <u>on Mondays at 2pm EST in Spring 2007</u>. You would like to make an appointment with him for <u>10am PST on 03/05/2007</u>, and expect the meeting to last <u>45 minutes</u>. Will there be an <u>overlap</u>?

the time ontology needs to be able to specify all the facts about the meetings that will allow a temporal reasoner to determine whether there is a conflict or overlap (see Appendix A. for the representation and proof steps for this use case).

In response to this need, such an ontology of temporal concepts, OWL-Time (formerly DAML-Time) (Hobbs and Pan, 2004, 2006), has been developed for describing the temporal content of Web pages and the temporal properties of Web services. It has been 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 easily available to a large group of Web developers and users. As (Allen, 1984; Hayes, 1994), OWL-Time is mainly concerned with the actual structure of time, instead of how facts persist through time, or how states of knowledge are sensitive to the changes that time can produce.

The ontology is represented in both first-order logic (FOL) and the OWL Web Ontology Language (McGuinness and Harmelen, 2003). The reason for choosing FOL to represent our ontology is that it is a flexible and well-understood approach to the representation of knowledge that satisfies many of the requirements raised for a meaning representation language. Specifically, it provides a sound computational basis for the verifiability, inference, and expressiveness requirements (Jurafsky and Martin, 2000). Considering the undecidability and high computational complexity of FOL theorem proving, sets of rules are developed to map subsets of the problems that can be represented by the time ontology in FOL to temporal constraint-based formalisms for more efficient temporal reasoning (see Section 7 for details).

Endorsed by W3C, the OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. As a vocabulary extension of RDF, OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. (McGuinness and Harmelen, 2004)

There have been many proposals for representing rules/axioms for the Semantic Web (e.g., SWRL (Horrocks, et al., 2004), RuleML (Boley, et al., 2001), SWSL (Grosof, et al., 2005)). SWRL (A Semantic Web Rule Language Combining OWL and RuleML), for example, has even proposed an extension of the language to FOL (Patel-Schneider, 2005). Unfortunately, none of these rule languages has become the standard for the Semantic Web (i.e., endorsed by W3C). Thus, in this thesis we only represent our ontology in FOL and OWL. When a rule language is endorsed by W3C, we should be able to straightforwardly translate our FOL axioms to that rule language.

1.2 Time for Natural Language

Since most of the information on the Web is in natural language, OWL-Time is also an ontology for natural language and can be used for temporal reasoning and to increase the temporal awareness for different natural language applications, such as question answering, information retrieval, and summarization. For example, in summarizing a story in terms of a timeline, a system may have to extract and chronologically order events in which a particular person participated. In answering a question as to a person's current occupation, a system may have to selectively determine which of several occupations reported for that person is the most recently reported one. (Mani et al., 2004)

A question answering system may have in its knowledge base the following sentences about Bill Clinton:

> On February 1, 2005, he (Bill Clinton) was picked by UN Secretary-General Kofi Annan to head the United Nations earthquake and tsunami relief and reconstruction effort. Five days later, he and Bush (George H. W. Bush) both appeared on the Super Bowl XXXIX pre-game show on Fox in support of their bipartisan effort to raise money for relief of the disaster through the USA Freedom Corps, an action which Bush described as "transcending politics." Thirteen days later, they both traveled to the affected areas to see how the relief efforts are going.

In order to answer questions like "when did Clinton and Bush both travel to the affected areas in 2005?", the specification of temporal information is needed to represent the calendar dates (e.g., "February 1, 2005") and durations (e.g., "thirteen days"), and do temporal reasoning on the date and the durations to get the answer date by applying temporal arithmetic rules (for this example, simply add 5 days and then 13 days to February 1, and get the answer date of February 19, 2005).

In general, given (possibly *incomplete* or *uncertain*) information about the temporal relations holding between events or facts in the represented domain, a system needs to be able to answer temporal queries about other implicit (entailed) relations; for example, queries about the possibility or the necessity that two particular future events will temporally overlap or about the shortest temporal distance separating two events that have occurred (Gerevini, 1997).

In order to achieve this, besides the representation of usual temporal concepts (e.g., temporal relations, dates and times, and durations), extraction of vague or typical event durations is also necessarily required due to the fact that missing durations is one of the most common sources of incomplete information for temporal reasoning, since very often explicit duration information (e.g., "a five-day meeting", "I have lived here for three years") is missing in natural language texts.

FOL representation and reasoning for natural language has already been applied successfully to question answering (Moldovan et al., 2002) at LCC[1], where questions and answers are first translated into FOL forms. A resolution-based module then proves that the question logically follows from the answer using a set of axioms that are automatically extracted from the WordNet (Miller, 1990) glosses.

For temporal information processing and reasoning in natural language text using OWL-Time, we can first use LFToolkit[2] to translate original temporal expressions in natural language to a "surface" FOL form where the predicate names are taken directly from the expressions. Then domain mapping rules are used to map the "surface" FOL form to the "deep" or domain FOL forms where the predicate

[1] http://www.languagecomputer.com/
[2] http://www.isi.edu/~nrathod/wne/LFToolkit/

names are from the OWL-Time vocabulary. For example, a natural language expression

“*four consecutive Mondays*”

is first translated into its “surface” FOL form

$plur(d,s) \wedge card(s, 4) \wedge consecutive(s, Monday1)$

The following mapping rule is then used

$(\forall\, s, p)\; consecutive(s, p) \equiv (\exists\, s_0)\; everyp(s, s_0, p)$

to translate this “surface” FOL form to the OWL-Time domain FOL form

$(everyp(s, s_0, Monday1) \wedge card(s, 4))$[3]

Finally, with all the propositions in the OWL-Time vocabulary, inferences can be made using OWL-Time inference axioms by any FOL theorem provers, e.g., OTTER (Kalman, 2001). Mapping rules have also been developed to map subsets of the problems that can be represented by the time ontology in FOL to different temporal constraint-based formalisms for more efficient temporal reasoning (see Section 7 for details).

OWL-Time can also be used to support and interpret useful annotation of temporal information in natural language text (Hobbs and Pustejovsky, 2003), e.g., TimeML (Pustejovsky et al., 2002), to provide a temporal grounding for texts that will facilitate reasoning over events, their orderings, and their relative granularity.

[3] See Section 4.3 for more details on this example.

1.3 Applications of OWL-Time: the Current Usage

OWL-Time can be applied to a very wide range of applications. Whenever you need a program or software agent to process and/or reason about temporal information, such a time ontology would be needed. In fact, OWL-Time has already been used in applications from all sorts of different fields besides the Semantic Web and natural language, from ubiquitous and pervasive computing, information integration, video event representation, to geographic information science, and so on. OWL-Time has already been used in/for

- OWL-S, an OWL-based Web service ontology which 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 (OWL-S Coalition, 2004). In the current version (i.e., OWL-S 1.1 release [4]), OWL-Time is used in its upper ontology for services in process.owl[5] and its Congo.com (a fictitious B2C site) example[6] (Pan and Hobbs, 2004). (Dobson and Sommerville, 2006) also uses OWL-Time in a Quality of Service (QoS) ontology to capture the semantics of availability.

- RSCDF as temporal and contextual extensions of Resource Description Framework (RDF) for the Semantic Web (Kaykova et al., 2005). A Semantic Web-based agent communication framework with a case study of meeting

[4] http://www.daml.org/services/owl-s/1.1/
[5] http://www.daml.org/services/owl-s/1.1/Process.owl
[6] http://www.daml.org/services/owl-s/1.1/examples.html

scheduling (Hritcu and Buraga, 2005). A reusable ontology for fluents in OWL, i.e., dealing with the problem of representing relationships that change over time in OWL (Welty and Fikes, 2006).

- Question Answering systems to handle questions that ask about temporal information. OWL-Time is used as one of the resources for identifying expected answer types, as well as supplying rules (axioms) for temporal inferences (Harabagiu and Bejan, 2005).

- Information integration for scientific data on the Grid to map temporal attributes from different sources (Tuchinda et al., 2004). A workday calendar model for workflows in business service grid environment (Liu et al., 2006).

- Video event representation for the development of a formal language for describing an ontology of video events (Nevatia et al., 2004). Temporal content modeling of video data for semantic video retrieval (QasemiZadeh et al., 2006).

- Ubiquitous and pervasive computing and its use for building a smart meeting room system for its standard ontology SOUPA and the broker-centric agent architecture CoBrA (Chen et al., 2003, 2004). A domain-independent semantic model for context-aware computing (Bulcao Neto and Pimentel, 2005). Event representation for building a smart hospital (Wu et al., 2005). The DoNet ontology for the applications to household appliances and buildings (Attard and Montebello, 2006).

- Data integration to resolve temporal semantic conflicts between data sources and receivers due to data semantics changes over time, e.g., historical stock price database, (Zhu et al., 2004).

- Improving portlet (i.e., applications within a portal in much the same way as servlets within a Web server) interoperability by defining events for the portal ontology (Díaz et al., 2005).

- Semantic Tuple Spaces (sTuples) to provide temporal concepts for creating ontologies in sTuples, a infrastructure for pervasive computing, where Tuple Space can be viewed as a logically shared memory, where producers add tuples to the space, while consumers read or extract tuples from the space using a search template (Khushraj et al., 2004).

- Ontology architecture for semantic Geo services for the Olympic Games 2008 in Beijing as its time ontology component (Weissenberg and Gartmann, 2003). Geographic Information Science (GIScience) to provide a basis for representing geographic information, e.g., "time regions" and "space-time regions" (Agarwal, 2005).

1.4 Major Contributions

The thesis has the following major contributions:

1. A rich and expressive ontology of temporal concepts in FOL and OWL Web Ontology Language that covers not only the basic concepts, such as temporal relations, durations, clock and calendar information, but also the more advanced ones, such as temporal aggregates, temporal arithmetic mixing months and days, and vague event durations.
2. Axioms to map subsets of the problems that can be represented by the ontology in FOL to temporal constraint-based formalisms for more efficient temporal reasoning.
3. The temporal aggregates part of the ontology is rich enough to represent complex multiple-layered and conditional temporal aggregates.
4. The first attempt to develop a set of rules for doing temporal arithmetic mixing months and days based on the "history-dependent intuition" with different desired arithmetic properties.
5. Use normal distributions to model vague judgments that are intervals on a scale, which can extend from time to other kinds of vague but substantive information.
6. The first effort to automatically extract vague and implicit event duration information in natural language text, and shows that the learning performance is very promising.

1.5 Thesis Outline

The rest of the thesis is organized as follows:

- Chapter 2 reviews related work on time ontologies and temporal reasoning formalisms.
- Chapter 3 gives an overview of the basics of the time ontology, including the representation of topological temporal relations, durations, clock and calendar information.
- Chapter 4 describes the representation of temporal aggregates (i.e., collections/aggregates of temporal entities), including complex multiple-layered and conditional temporal aggregates.
- Chapter 5 presents why and how we developed temporal arithmetic rules with mixing months and days, given a list of desired arithmetic properties.
- Chapter 6 presents our work on modeling and learning vague event durations from natural language texts.
- Chapter 7 describes a set of mapping axioms we developed for mapping subsets of the problems that can be represented by the ontology in FOL to different temporal constraint-based formalisms for more efficient reasoning.
- Chapter 8 concludes the thesis and discusses future research directions.

Chapter 2

Related Work

2.1 Ontology of Temporal Concepts

There already exist ontologies of temporal concepts from both commercial (e.g., time ontology in Cyc (Lenat, 1995)) and academic sources (e.g., a reusable time ontology (Zhou and Fikes, 2002); time ontology in SUMO (Suggested Upper Merged Ontology) (Niles and Pease, 2001); a catalog of temporal theories (Hayes, 1995)).

Both Cyc and SUMO are large upper ontologies encoding common sense knowledge, and hence more focus on abstract and philosophical concepts that are general enough to address a broad range of domain areas. Each domain (including time) is not intended to be complete, and they don't aim at Web or natural language applications either. Moreover, OWL-Time is designed to be cleanly separated from other ontologies one might build, where the interface between them will involve the minimum of elements, and the only ontologies assumed are arithmetic and set theory. As defined in a large upper ontology, the time ontology in Cyc and SUMO, however, is less separated from other domains in the upper ontology (e.g., event ontology), and makes more assumptions about higher level concepts and their properties. As a commercial institution, Cyc has only released a part of its entire ontology to the public, and it has not been subject to extensive peer review either.

SUMO is written in SUO-KIF (Pease, 2004) which was derived from KIF (Genesereth, 1992) to support the definition of SUMO. Special translation or a theorem prover would be needed in order to interact with ontologies written in other languages. The time ontology in SUMO is relatively less expressive. For example, it cannot express multiple-layered or conditional temporal aggregates.

(Zhou and Fikes, 2002) proposed a reusable time ontology written in KIF for large-scale knowledge system applications. Our treatments on topological temporal relations are pretty similar, for example, we both treat instant and interval as independent primitives. However, we have different treatment for many other domains (e.g., calendar and clock information and measures of durations), and OWL-Time includes a richer set of axioms and concepts and also broader coverage of the domains (e.g., time zone, temporal aggregates, and temporal arithmetic).

As a survey of several structures that time can be taken to have, a catalog of temporal theories (Hayes, 1994) gives a coherent overview of different theories and synthesizes them as much as possible. Like OWL-Time, what it's concerned with is the actual structure of time, instead of how facts persist through time, or how states of knowledge are sensitive to the changes that time can produce, which is the major concern for the research on temporal logic as described next.

2.2 Temporal Ontology for Natural Language

According to Mark Steedman (1997, 2002), in general there are two approaches to temporal ontology for natural language. The first one is the descriptive approach which is usually attributed to Vendler (1967) and has then been refined (Dowty 1979, 1982; Verkuyl, 1989) and further extended (Moens and Steedman, 1988; Smith, 1991). The work in this framework most concerns the descriptive properties of tense and aspect in natural language, for example, how to disambiguate the four aspectual categories (i.e., achievements, activities, accomplishments, and states), which is relevant to our work on extracting vague event durations from news articles (see Section 6 for details).

The logical and computational approach (McDermott, 1982; van Benthem, 1983, 1995; Allen, 1984; Galton, 1984; Allen and Ferguson, 1997) is the other approach that tries to formalize this quite complex ontology. OWL-Time falls into this category with the focus on representing and reasoning about temporal concepts and expressions, instead of tense and aspect in natural language.

Besides FOL, or first-order predicate calculus (FOPC), which is very popular in the logical and computational approach, special temporal operators were used in other temporal logic, for example, tense logic (Prior, 1957), which, as other modal logics, has a richer expressive power in its domain (e.g., time domain), but it needs special theorem proving systems (maybe inefficient or not practical). Most of the tense logics can be translated into FOL, and usually they are translated into FOL for (efficient/practical) theorem proving or automated deduction.

2.3 Other Temporal Representation and Reasoning Formalisms

Constraint-based formalisms for temporal reasoning have been developed for representing and reasoning, ranging from less expressive problems (e.g., Simple Temporal Problem (STP) (Dechter et al., 1991)) to more expressive ones (e.g., Disjunctive Temporal Problem (DTP)). Efficient algorithms have been developed for solving STPs and DTPs to get exact solutions (Tsamardinos and Pollack, 2003) or approximate solutions (Moffitt and Pollack, 2005). Planning and scheduling tasks are the applications they are most concerned with.

Although DTP is more expressive than STP and the Temporal Constraint Satisfaction Problem (TCSP), it can only represent duration constraints (plus simple relations as before/after/at given time points, which can be converted to their representation), and there are no calendar or clock concepts involved. It cannot represent temporal aggregates or any non-temporal properties either.

TCSP-based framework has also been applied to temporal reasoning for natural language (Han and Lavie, 2004), where not only the durations but also the calendar and granularity are modeled as a two level TCSP – the higher level models a TCSP with different temporal objects/variables, and within each temporal variable is a model of calendars, i.e., a CSP describing constraints among different temporal units. However, more advance temporal concepts like temporal aggregates (e.g., recurrence expressions) cannot be represented in their current framework (Han et al., 2006). A special-purpose reasoner, which is a conventional TCSP solver with the aid of the calendar model, has to be used to draw inferences on given temporal expressions.

A formal framework for the definition and representation of time granularities was proposed in (Bettini et al., 2000) especially for temporal database applications. This framework has also been extended to TCSP (Bettini et al., 2002) for more general temporal reasoning tasks.

Chapter 3

OWL-Time Basics

The material in this section is taken from (Hobbs, 2002; Hobbs and Pan, 2004). The full definitions of OWL-Time can be found in (Hobbs and Pan, 2004). Here we only present those parts that are essential for our later treatment of the complex temporal phenomena (i.e., temporal aggregates, temporal arithmetic mixing months and dates, and vague event durations), including a vocabulary for expressing facts about topological relations among instants and intervals (Section 3.1), together with information about durations (Section 3.3), and about clock and calendar information (Section 3.4). The abstract characterization of the concepts and relations are expressed in both FOL and OWL. A time zone resource in OWL developed for the entire world will also be described.

The notation of set theory is assumed for the ontology. Sets and elements of sets will be ordinary individuals, and relations such as "member" will be relations between such individuals. In particular, we will use the relation "member" between an element of a set and the set. We will use the notation "{x}" for the singleton set containing the element *x*. We will use the function "union" to refer to the union operation between two sets. The function "card" will map a set into its cardinality.

In an extension of the time ontology, it is also allowed for temporal predicates to apply directly to events (Pan and Hobbs, 2004), should the user wish, but here we restrict our treatment to temporal entities.

3.1 Topological Temporal Relations

There are two subclasses of TemporalEntity: Instant and Interval.

$$(\forall T)[TemporalEntity(T) \equiv Interval(T) \lor Instant(T)]$$

This can be expressed in OWL as:

```
<owl:Class rdf:ID="Instant">
  <rdfs:subClassOf rdf:resource="#TemporalEntity"/>
</owl:Class>

<owl:Class rdf:ID="Interval">
  <rdfs:subClassOf rdf:resource="#TemporalEntity"/>
</owl:Class>

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

Intervals are, intuitively, things with extent and instants are, intuitively, point-like in that they have no interior points.

The predicates "begins" and "ends" are relations between instants and temporal entities.

$$begins(t,T) \supset Instant(t) \land TemporalEntity(T)$$

$$ends(t,T) \supset Instant(t) \land TemporalEntity(T)$$

"begins", for example, can be specified in OWL as:

```
<owl:ObjectProperty rdf:ID="begins">
  <rdfs:domain rdf:resource="#TemporalEntity" />
  <rdfs:range  rdf:resource="#Instant" />
```

```
</owl:ObjectProperty>
```

The predicate "inside" is a relation between an instant and an interval.

$$inside(t,T) \supset Instant(t) \wedge Interval(T)$$

This concept of "inside" is not intended to include beginnings and ends of intervals.

It will be useful in characterizing clock and calendar terms to have a relation between instants and intervals that says that the instant is inside or the beginning of the interval.

$$(\forall\, t,T)[beginsOrIn(t,T) \equiv [begins(t,T) \vee inside(t,T)]]$$

A proper interval can be defined as one whose start and end are not identical.

$$(\forall\, T)ProperInterval(T)$$
$$\equiv Interval(T) \wedge (\forall\, t_1,t_2)[begins(t_1,T) \wedge ends(t_2,T) \supset t_1 \neq t_2]]$$

There is a "before" relation on temporal entities, which gives directionality to time. If temporal entity T_1 is before temporal entity T_2, then the end of T_1 is before the start of T_2. Thus, "before" can be considered to be basic to instants and derived for intervals.

$$(\forall\, T_1,T_2)[before(T_1,T_2)$$
$$\equiv (\exists\, t_1,t_2)[ends(t_1,T_1) \wedge begins(t_2,T_2) \wedge before(t_1,t_2)]]$$

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

$$before(T_1,T_2) \supset T_1 \neq T_2$$

$$before(T_1,T_2) \supset \neg before(T_2,T_1)$$

$$before(T_1,T_2) \wedge before(T_2,T_3) \supset before(T_1,T_3)$$

If an instant is inside a proper interval, then the beginning of the interval is before the instant, which is before the end of the interval. This is the principal property of inside.

$$inside(t,T) \wedge begins(t_1,T) \wedge ends(t_2,T) \wedge ProperInterval(T)$$
$$\supset before(t_1,t) \wedge before(t,t_2)$$

The relation "after" is defined in terms of "before".

$$after(T_1,T_2) \equiv before(T_2,T_1)$$

The relations between intervals defined in Allen's temporal interval calculus (Allen, 1984; Allen and Ferguson, 1997) can be defined in a straightforward fashion in terms of "before" and identity on the beginning and end points.

OWL-Time includes axioms defining 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 "intMeets" is:

$$intMeets(T_1,T_2)$$
$$\equiv [ProperInterval(T_1) \wedge ProperInterval(T_2)$$
$$\wedge (\exists\, t)[ends(t,T_1) \wedge begins(t,T_2)]]$$

It will be useful below to have a single predicate for intervals intersecting in at most an instant.

$$
\begin{aligned}
& nonoverlap(T_1, T_2) \\
& \quad \equiv [intBefore(T_1, T_2) \vee intAfter(T_1, T_2) \vee intMeets(T_1, T_2) \\
& \qquad \vee intMetBy(T_1, T_2)]
\end{aligned}
$$

This could have been defined as easily in terms of "before" relations on the beginnings and ends of the intervals.

"intMeets", for example, can be specified in OWL as:

```
<owl:ObjectProperty rdf:ID="intMeets">
  <rdfs:subPropertyOf rdf:resource="#nonoverlap" />
  <rdfs:domain rdf:resource="#ProperInterval" />
  <rdfs:range  rdf:resource="#ProperInterval" />
</owl:ObjectProperty>
```

3.2 Linking Time and Events

The time ontology links to other things in the world through four predicates: "atTime", "during", "holds", and "timeSpan". It is assumed that another ontology provides for the description of events, either a general ontology of event structure abstractly conceived, or specific, domain-dependent ontologies for specific domains.

The predicate "atTime" relates an event to an instant, and is intended to say that the event holds, obtains, or is taking place at that time.

$$atTime(e,t) \supset Instant(t)$$

The predicate "during" relates an event to an interval, and is intended to say that the event holds, obtains, or is taking place during that interval.

$$during(e,T) \supset Interval(T)$$

Whether a particular process is viewed as instantaneous or as occurring over an interval is a granularity decision that may vary according to the context of use, and is assumed to be provided by the event ontology.

The predicate "timeSpan" relates eventualities to instants or intervals (or temporal sequences of instants and intervals). For contiguous states and processes, it tells the entire instant or interval for which the state or process obtains or takes place.

$$timeSpan(T,e) \supset TemporalEntity(T) \vee tseq(T)$$[7]

$$timeSpan(T,e) \wedge Interval(T) \supset during(e,T)$$

$$timeSpan(t,e) \wedge Instant(t) \supset atTime(e,t)$$

Whether the event obtains at the start and end points of its time span is a matter for the event ontology to specify. The silence here on this issue is the reason "timeSpan" is not defined in terms of necessary and sufficient conditions.

In an extension of the time ontology, it is also allowed for temporal predicates to apply directly to events (Pan and Hobbs, 2004), should the user wish. Thus, "begins(t,e)" says that the instant *t* begins the interval that is the time span of event *e*.

3.3 Measuring Durations

3.3.1 Temporal Units

This development assumes that ordinary arithmetic is available.

[7] *tseq*(*T*): T is a temporal sequence. Please see Section 4 for details.

There are at least two approaches that can be taken toward measuring intervals. The first is to consider units of time as functions from Intervals to Reals. Owing to infinite intervals, the range must also include Infinity.

minutes: Intervals $\rightarrow$ Reals $\cup$ {Infinity}

minutes([5:14,5:17]) = 3

The other approach is to consider temporal units to constitute a set of entities, call it "TemporalUnits", and have a single function *duration* mapping "Intervals $\times$ TemporalUnits" into the Reals.

duration: Intervals $\times$ TemporalUnits $\rightarrow$ Reals $\cup$ {Infinity}

duration([5:14,5:17],**Minute**) = 3

The two approaches are interdefinable:

seconds(*T*) = *duration*(*T*,**Second**)

minutes(*T*) = *duration*(*T*,**Minute**)

hours(*T*) = *duration*(*T*,**Hour**)

days(*T*) = *duration*(*T*,**Day**)

weeks(*T*) = *duration*(*T*,**Week**)

months(*T*) = *duration*(*T*,**Month**)

years(*T*) = *duration*(*T*,**Year**)

Ordinarily, the first is more convenient for stating specific facts about particular units; the second is more convenient for stating general facts about all units.

The arithmetic relations among the various units are as follows:

$seconds(T) = 60*minutes(T)$

$minutes(T) = 60*hours(T)$

$hours(T) = 24*days(T)$

$days(T) = 7*weeks(T)$

$months(T) = 12*years(T)$

The relation between days and months (and, to a lesser extent, years) are specified later as part of the ontology of clock and calendar, below.

3.3.2 Concatenation and *Hath*

The multiplicative relations above don't tell the whole story of the relations among temporal units. Temporal units are *composed of* smaller temporal units. A larger temporal unit is a concatenation of smaller temporal units. A general relation of concatenation between an interval and a set of smaller intervals will be defined first. A predicate *Hath* that specifies the number of smaller unit intervals that concatenate to a larger interval will then be introduced.

Concatenation: A proper interval *x* is a concatenation of a set *S* of proper intervals if and only if *S* covers all of *x*, and all members of *S* are subintervals of *x* and are mutually disjoint. (The third conjunct on the right side of ≡ is because "beginsOrIn" only covers instants that begin or are inside *x*.)

$concatenation(x,S)$

$\equiv ProperInterval(x)$

$$\begin{aligned}
&\wedge\ (\forall\, z)[beginsOrIn(z,x) \supset (\exists\, y)[member(y,S)\ \wedge\ beginsOrIn(z,y)]]\\
&\wedge\ (\forall\, z)[ends(z,x) \supset (\exists\, y)[member(y,S)\ \wedge\ ends(z,y)]]\\
&\wedge\ (\forall\, y)[member(y,S)\\
&\qquad\qquad \supset [intStarts(y,x) \vee intDuring(y,x) \vee intFinishes(y,x)\\
&\qquad\qquad\qquad \vee intEquals(y,x)]]\\
&\wedge\ (\forall\, y_1,y_2)[member(y_1,S)\ \wedge\ member(y_2,S)\\
&\qquad\qquad \supset [y_1 = y_2\ \vee nonoverlap(y_1,y_2)]]
\end{aligned}$$

Hath: The basic predicate used here for expressing the composition of larger intervals out of smaller clock and calendar intervals is *Hath*, from statements like "30 days hath September" and "60 minutes hath an hour." Its structure is

$Hath(N,u,x)$

meaning "N proper intervals of duration one unit *u* hath the proper interval *x*." That is, if "$Hath(N,u,x)$" holds, then *x* is the concatenation of *N* unit intervals where the unit is *u*. For example, if *x* is some month of September then "*Hath*(30,**Day**,*x*)" would be true. *Hath* is defined as follows:

$$\begin{aligned}
&Hath(N,u,x)\\
&\qquad \equiv (\exists\, S)[card(S) = N\ \wedge\ (\forall\, z)[member(z,S) \supset duration(z,u) = 1]\\
&\qquad\qquad \wedge\ concatenation(x,S)]
\end{aligned}$$

That is, *x* is the concatenation of a set *S* of *N* proper intervals of duration one unit *u*.

The type constraints on its arguments can be proved as a theorem: *N* is an integer (assuming that is the constraint on the value of "card"), *u* is a temporal unit, and *x* is a proper interval:

$Hath(N,u,x) \supset integer(N) \wedge TemporalUnit(u) \wedge ProperInterval(x)$

This treatment of *concatenation* will work for scalar phenomena in general. This treatment of *Hath* will work for measurable quantities in general.

3.3.3 The Structure of Temporal Units

It is now possible to define predicates true of intervals that are 1 temporal unit long. For example, *week* is a predicate true of intervals whose duration is one week.

$second(T) \equiv seconds(T) = 1$

$minute(T) \equiv minutes(T) = 1$

$hour(T) \equiv hours(T) = 1$

$day(T) \equiv days(T) = 1$

$week(T) \equiv weeks(T) = 1$

$month(T) \equiv months(T) = 1$

$year(T) \equiv years(T) = 1$

It is now in a position to state the relations between successive temporal units.

$minute(T) \supset Hath(60,*Second*,T)$

$hour(T) \supset Hath(60,*Minute*,T)$

$day(T) \supset Hath(24,*Hour*,T)$

$week(T) \supset Hath(7,*Day*,T)$

$year(T) \supset Hath(12,*Month*,T)$

3.3.4 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 "durationOf" that takes eight arguments, one for a temporal entity, 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 relations relating the duration description to the values of each of the eight arguments. Thereby we convert the 8-ary predicate "durationOf" 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, as well as the binary relations like "yearsOf":

$$
\begin{aligned}
&(\forall\, T,y,m,w,d,h,n,s)[durationOf(T,y,m,w,d,h,n,s) \\
&\quad\equiv (\exists\, D)[durationDescriptionOf(D,T) \wedge DurationDescription(D) \\
&\qquad \wedge yearsOf(D, \mathrm{y}) \wedge monthsOf(D, m) \wedge weeksOf(D, w) \\
&\qquad \wedge daysOf(D, d) \wedge hoursOf(D, h) \wedge minutesOf(D, n) \\
&\qquad \wedge secondsOf(D, s)]]
\end{aligned}
$$

It can be expressed in OWL as:

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

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#seconds" />
         <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
         </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

There are two ways to specify the duration description of a temporal entity. The relation "durationDescriptionOf" uses "DurationDescription" as its range, while "durationDescriptionDataType" uses the XML Schema datatype "duration"[8] as its range:

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

<owl:DatatypeProperty rdf:ID="durationDescriptionDataType">
  <rdfs:domain rdf:resource="#TemporalEntity" />
  <rdfs:range  rdf:resource="&xsd;duration" />
</owl:DatatypeProperty>
```

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 entity.

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

3.4 Clock and Calendar

3.4.1 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 UTC (Coordinated Universal Time)[9] hours, but are, e.g., offset half an hour. Seconds are not relative to the time zone. Days, weeks, months and years are also relative to the time zone, since, e.g., 2005 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 might be called 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 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.

[9] http://aa.usno.navy.mil/faq/docs/UT.html

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 until recently was 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 shifted in April from Eastern Standard to Central Daylight time.

3.4.2 Time Zone Resource in OWL

A time zone resource[10] in OWL for not only the US but also the entire world has been developed, including three parts: the time ontology file[11], the US time zone instance file[12], and the world time zone instance file[13].

The time zone ontology links a preliminary geographic ontology with a time ontology. It defines the vocabulary about regions, political regions (countries, states, counties, reservations, and cities), time zones, daylight saving policies, and the

[10] http://www.isi.edu/~pan/timezonehomepage.html
[11] http://www.isi.edu/~pan/damltime/timezone-ont.owl
[12] http://www.isi.edu/~pan/damltime/timezone-us.owl
[13] http://www.isi.edu/~pan/damltime/timezone-world.owl

relationships between these concepts. Its instances also link to other existing data on the Web, such as US states instances developed by Terry Payne[14], FIPS 55 county instances[15], and ISO country instances[16].

It can handle all the usual time zone and daylight savings cases. For example, Los Angles uses PST, the time offset from UTC is -8 hours, and it observed daylight savings from April 3 to October 30 in 2005. But it handles unusual cases as well. For example, in Idaho the northern part is in the Pacific zone, the southern part in the Mountain. The city of West Wendover, Nevada is in the Mountain time zone, while the rest of Nevada is in the Pacific. For the details, see the documentation[17], which includes an outline of the ontology and the anticipated use.

3.4.3 Clock and Calendar Units

A day as a calendar interval begins at and includes midnight, and goes until, but does not include, the next midnight. This contrasts with a day as a duration which is any interval that is 24 hours in length.

Including the beginning but not the end of a calendar interval in the interval may strike some as arbitrary. But we get a cleaner treatment if, for example, all times of the form 12:xx am, including 12:00 am, are part of the same hour and day, and all times of the form 10:15:xx, including 10:15:00, are part of the same minute. Clock intervals are described with the predicate "clockInt":

[14] http://www.daml.ri.cmu.edu/ont/USRegionState.daml
[15] http://www.daml.org/2003/02/fips55/
[16] http://www.daml.org/2001/09/countries/iso
[17] http://www.isi.edu/~pan/damltime/time-zone documentation.txt

$clockInt(y,n,u,x)$

This expression says that y is the nth clock interval of type u in x. For example, the proposition "*clockInt*(10:03,3, **Minute**,[10:00,11:00])" holds. Here u is a member of the set of clock units, that is, one of *Second*, *Minute*, or *Hour*. The larger interval x may not line up exactly with clock intervals. In this case we take y to be the nth complete clock interval of type u in x. In addition, there is a calendar unit function with similar structure:

$calInt(y,n,u,x)$

This says that y is the nth calendar interval of type u in x. For example, the proposition "*calInt*(*12Mar2002*,12,**Day**,*Mar2002*)" holds. Here u is one of the calendar units *Day*, *Week*, *Month*, and *Year*.

A distinction is made above between clocks and calendars because they differ in how they number their unit intervals. The first minute of an hour is labeled with 0; for example, the first minute of the hour [10:00,11:00] is 10:00. The first day of a month is labeled 1; the first day of March is March 1. We number minutes for the number just completed; we number days for the day we are working on. Thus, if the larger unit has N smaller units, the argument n in "clockInt" runs from 0 to N-1; whereas, in "calInt", n runs from 1 to N. To state properties true of both clock and calendar intervals, we can use the predicate "calInt" and relate the two notions with the axiom

$calInt(y,n,u,x) \equiv clockInt(y,n\text{-}1,u,x)$

In "*calInt*(y,n,u,x)" and "*clockInt*(y,n,u,x)", y is not an arbitrary interval; it has to be a calendar-clock interval which is a subclass of a proper interval:

$$CalendarClockInterval(\mathrm{T}) \supset ProperInterval(T)$$

The names of months can be defined in terms of the predicate "calInt". For example, July is the seventh month of a year.

$$July(m,y) \equiv calInt(m,7,*Month*,y) \wedge (\exists\, n,t)\ [calInt(y,n,*Year*,t)]$$

The top-level time interval (for modern applications) is CE(z), which is the Common Era in time zone z. Thus, the year 2005 in the Eastern Standard Time Zone is the y such that "*calInt*(y, 2005, **Year**, *CE*(**EST**))".

A week is any seven consecutive days. A calendar week, by contrast, according to a commonly adopted convention, starts at midnight, Saturday night, and goes to the next midnight, Saturday night. That is, weeks start with Sunday. (By contrast, the ISO 8061 standard week starts with Monday, and this is also accommodated in the time ontology.) There are 52 weeks in a year, but there are not usually 52 calendar weeks in a year. Weeks are independent of months and years. However, we can still talk about the *n*th week in some larger period of time, e.g., the third week of the month or the fifth week of the semester. To say a time interval is a calendar-week, we say

$$calInt(y,n,*Week*,x)$$

As it happens, the n and x arguments will often be irrelevant when we only want to say that some period is a calendar week, and not say which.

The day of the week is a calendar interval of type *Day*. The *n*th day-of-the-week in a week is the *n*th day in that interval.

$$dayofweek(y,n,x) \equiv calInt(y,n,{*}Day{*},x) \ \wedge\ (\exists\ n_1,x_1)\ calInt(x,n_1,{*}Week{*}, x_1)$$

The days of the week have special names in English, such as Sunday, Monday, and so on. For example, Monday is defined as follows:

$$dayofweek(y,2,x) \equiv Monday(y,x)$$

This says that *y* is the Monday of week *x*. The ISO 8061 standard week is related to the traditional week as follows:

$$0 < n < 7 \supset [isodayofweek(y,n,x) \equiv dayofweek(\mathrm{y},n+1,x)]$$
$$isodayofweek(y,7,x) \equiv Sunday(y,x)$$

Holidays can also be specified in this ontology. To say that July 4 is a holiday one could write

$$(\forall\ d,m,y)[calInt(d,4,{*}Day{*},m) \wedge July(m,y) \supset holiday(d)]$$

Holidays like Easter can be defined in terms of this ontology coupled with an ontology of the phases of the moon.

Standard notation for date lists the year, month, day, and time zone. It is useful to define a predication for this.

$$\begin{aligned}&dateOf(t,y,m,d,z)\\ &\quad \equiv (\exists\ d_1,m_1,y_1,e)\ [beginsOrIn(t,d_1)\\ &\qquad \wedge\ calInt(d_1,d,{*}Day{*},m_1)\ \wedge\ calInt(m_1,m,{*}Month{*},y_1)\\ &\qquad \wedge\ calInt(y_1,y,{*}Year{*},e)\ \wedge\ CE(z) = e]\end{aligned}$$

Standard notation for times list the year, month, day, hour, minute, second, and time zone. It is useful to define a predication for this.

$$\begin{aligned} &timeOf(t,y,m,d,h,n,s,z) \\ &\qquad \equiv beginsOrIn(t,secFn(s,minFn(n,hrFn(h, \\ &\qquad\qquad daFn(d,monFn(m,yrFn(y,CE(z)))))))) \end{aligned}$$

Alternatively,

$$\begin{aligned} &timeOf(t,y,m,d,h,n,s,z) \\ &\qquad \equiv (\exists\, s_1,n_1,h_1,d_1,m_1,y_1,e)\, [beginsOrIn(t,s_1) \wedge sec(s_1,s,n_1) \\ &\qquad\qquad \wedge min(n_1,n,h_1) \wedge hr(h_1,h,d_1) \wedge da(d_1,d,m_1) \\ &\qquad\qquad \wedge mon(m_1,m,y_1) \wedge yr(y_1,y,e) \wedge CE(z) = e] \end{aligned}$$

For example, an instant *t* has the time

5:14:35pm PST, Wednesday, February 6, 2002

if the following properties hold for *t*:

$$\begin{aligned} &timeOf(t,2002,2,6,17,14,35,{*}PST{*}) \\ &\qquad \wedge (\exists\, w,x)[beginsOrIn(t,w) \wedge Wednesday(w,x)] \end{aligned}$$

The second line says that *t* is in the Wednesday *w* of some week *x*.

The relations among time zones can be expressed in terms of the "timeOf" predicate. Two examples follow:

$$timeOf(t,y,m,d,h,n,s,{*}EST{*}) \equiv timeOf(t,y,m,d,h,n,s,{*}CDT{*})$$

$$\begin{aligned} &timeOf(t,y,m,d,h,n,s,{*}GMT{*}) \wedge hours(T) = 8 \wedge ends(t,T) \\ &\qquad \wedge begins(t_1,T) \wedge timeOf(t_1,y_1,m_1,d_1,h_1,n,s,{*}GMT{*}) \\ &\qquad\qquad \supset timeOf(t,y_1,m_1,d_1,h_1,n,s,{*}PST{*}) \end{aligned}$$

3.4.4 Calendar-Clock Description

To express "*calInt*(*y*,*n*,*u*,*x*)" and "*clockInt*(*y*,*n*,*u*,*x*)" directly in OWL is inconvenient since *x* is itself a clock or calendar interval that requires description. So a calendar-clock description is defined 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" />
    <TemporalUnit rdf:about="#unitYear" />
  </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, a property, called "calendarClockDescriptionOf" with "CalendarClockDescription" as the range, for calendar-clock intervals is defined.

To express "*calInt*(*12Mar2002*,12,**Day**,*Mar2002*)", for example, using calendar-clock description, we need an instance of "CalendarClockDescription" that has values only for unitType ("unitDay"), year (2002), month (3), and day (12). "*clockInt*(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">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#unitType" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
        </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#year" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
        </owl:maxCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>

  ...

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#second" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
        </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#timeZone" />
        <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
        </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

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

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

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

With this “inCalendarClock” relation, we can say that an instant 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).

Two simpler relations, “calendarClockDescriptionDatatype” and “inCalendar-ClockDatatype” are also defined in OWL. 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”[18] as their ranges, while the above uses “CalendarClock-Description”:

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

<owl:DatatypeProperty rdf:ID="calendarClockDescriptionDataType">
  <rdfs:domain rdf:resource="#CalendarClockInterval" />
  <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 01/01/2005 can be expressed using both inCalendarClockDataType and inCalendarClock in OWL as:

```
<time:Instant  rdf:ID="instantExample">
    <time:inCalendarClock rdf:resource="instantExampleDescription" />
```

[18] http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#dateTime

```
      <time:inCalendarClockDataType rdf:datatype="&xsd;dateTime">
      2005-01-01T10:30:00-5:00</time:inCalendarClockDataType>
</time:Instant>

<time:CalendarClockDescription rdf:ID="instantExampleDescription">
      <time:unitType rdf:resource="&time;unitMinute" />
      <time:year rdf:datatype="&xsd;gYear">2005</time:year>
      <time:month rdf:datatype="&xsd;gMonth">1</time:month>
      <time:week rdf:datatype="&xsd;nonNegativeInteger">1</time:week>
      <time:day rdf:datatype="&xsd;gDay">1</time:day>
      <time:dayOfWeekField rdf:datatype="&xsd;nonNegativeInteger">6
      </time: dayOfWeekField>
      <time:dayOfYearField rdf:datatype="&xsd;nonNegativeInteger">1
      </time: dayOfYearField>
      <time:hour rdf:datatype="&xsd;nonNegativeInteger">10</time:hour>
      <time:minute rdf:datatype="&xsd;nonNegativeInteger">30</time:minute>
      <time:timeZone rdf:resource="&tz-us;EST" />
</time:CalendarClockDescription>
```

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 01/01/2005 is Saturday, 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 data[19]. 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.

[19] http://www.isi.edu/~pan/damltime/timezone-us.owl

Chapter 4

Temporal Aggregates

Temporal aggregates are collections/aggregates of temporal entities. Natural language texts involve many expressions of temporal aggregates, such as “every Tuesday”, “every 3rd Monday in 2001”, “4 consecutive Sundays”, “3 weekdays after today”, “the 4th of 6 days of voting”, and so on.

Such information is also very common on the Web, for example, in Web services you may have “the customer service is available from 8am to 5pm EST every working day between 01/01/2004 to 01/15/2005 (Dumas et al. 2001)”; “send me the closing price of IBM, every 7 days after it exceeds $100, as long as it remains above $100 (Motakis and Zaniolo 1997)”. Thus it is crucial to develop an ontology of temporal aggregates to represent these expressions.

In Section 4.1 we describe our general approach to temporal aggregates in FOL that is intended to handle any collection of temporal entities, regardless of how they are described. An important special case is temporal aggregates consisting of clock and calendar temporal entities, like calendar months. iCalendar (Dawson and Stenerson, 1998) is a popular framework for describing these, and in Section 4.2 we show how iCalendar can be embedded in OWL-Time. In Section 4.3 we present several examples of natural language expressions for temporal aggregates and how they would be represented in FOL. Some can be described in both iCalendar and OWL-Time, while others can only be represented in OWL-Time. We also demonstrate how we can translate from a natural language sentence to our representation. The temporal aggregates ontology represented in OWL is described in detail in Section 4.4. Two natural language examples will be taken from Section 4.3, and are expressed in OWL in Section 4.5. Suggestions on improving set expression (i.e., temporal aggregate) annotations in TimeML are proposed in Section 4.6. Section 4.7 reports the evaluation result on the coverage of the temporal aggregates ontology.

4.1 Temporal Aggregates in FOL

In this section, for convenience, we will make moderate use of second-order formulations, and quantify over predicate symbols. This could be eliminated with the use of an "apply" predicate and axiom schemas systematically relating predicate symbols to corresponding individuals, e.g., the axiom schema for unary predicates *p*,

$$(\forall\, x)[apply(*p*,x) \equiv p(x)]$$

4.1.1 Temporal Sequences

It will be convenient to have a relation "ibefore" that generalizes over several interval and instant relations, covering both "intBefore" and "intMeets" for proper intervals.

$$\begin{aligned}&(\forall\, T_1,T_2)[ibefore(T_1,T_2)\\&\quad\equiv [before(T_1,T_2) \vee\\&\qquad\qquad [ProperInterval(T_1) \wedge ProperInterval(T_2) \wedge intMeets(T_1,T_2)]]]\end{aligned}$$

It will also be useful to have a relation “iinside” that generalizes over all temporal entities and aggregates. A predicate “iinside-1” is defined first that generalizes over instants and intervals and covers “intStarts”, “intFinishes” and “intEquals” as well as “intDuring” for intervals. We break the definition into several cases.

$$
\begin{aligned}
&(\forall\ T_1,T_2)[\mathit{iinside\text{-}1}(T_1,T_2) \\
&\quad \equiv [T_1 = T_2 \\
&\qquad \vee\ [\mathit{Instant}(T_1) \wedge \mathit{ProperInterval}(T_2) \wedge \mathit{inside}(T_1,T_2)] \\
&\qquad \vee [(\exists\ t)\ \mathit{begins}(t,T_1) \wedge \mathit{ends}(t,T_1) \\
&\qquad\qquad \wedge \mathit{ProperInterval}(T_2) \wedge \mathit{inside}(t,T_2)] \\
&\qquad \vee [\mathit{ProperInterval}(T_1) \wedge \mathit{ProperInterval}(T_2) \\
&\qquad\qquad \wedge [\mathit{intStarts}(T_1,T_2) \vee \mathit{intDuring}(T_1,T_2) \\
&\qquad\qquad\qquad \vee \mathit{intFinishes}(T_1,T_2) \vee \mathit{intEquals}(T_1,T_2)]]]]
\end{aligned}
$$

The third disjunct in the definition is for the case of 0-length intervals, should they be allowed and distinct from the corresponding instants.

A temporal aggregate is first of all a set of temporal entities, but it has further structure. The relation “ibefore” imposes a natural order on some sets of temporal entities, and the predicate “tseq” is used to describe those sets.

$$
\begin{aligned}
&(\forall\ s)[\mathit{tseq}(s) \equiv (\forall\ t)[\mathit{member}(t,s) \supset \mathit{TemporalEntity}(t)] \\
&\qquad\qquad \wedge (\forall\ t_1,t_2)[\mathit{member}(t_1,s) \wedge \mathit{member}(t_2,s) \\
&\qquad\qquad\qquad \supset [t_1 = t_2 \vee \mathit{ibefore}(t_1,t_2) \vee \mathit{ibefore}(t_2, t_1)]]]
\end{aligned}
$$

That is, a temporal sequence is a set of temporal entities totally ordered by the “ibefore” relation. A temporal sequence has no overlapping temporal entities.

It will be useful to have the notion of a temporal sequence whose elements all have a property *p*.

$$(\forall\ s,p)[tseqp(s,p) \equiv tseq(s) \wedge (\forall\ t)[member(t,s) \supset p(t)]]$$

The same temporal aggregate can be broken up into a set of intervals in many different ways.

A minimal temporal sequence is one whose intervals are maximal, so that the number of intervals is minimal. We can view a week as a week or as 7 individual successive days; the first would be minimal. We can go from a non-minimal to a minimal temporal sequence by concatenating intervals that meet.

$$\begin{aligned}&(\forall\ s)[min\text{-}tseq(s)\\&\qquad \equiv (\forall\ t_1,t_2)[member(t_1,s) \wedge member(t_2,s)\\&\qquad\qquad \supset [t_1 = t_2 \vee (\exists\ t)[ibefore(t_1,t) \wedge ibefore(t,t_2) \wedge {\sim}member(t,s)]]]]\end{aligned}$$

That is, *s* is a minimal temporal sequence when any two distinct intervals in *s* have a temporal entity not in *s* between them.

A temporal sequence s_1 is a minimal equivalent temporal sequence to temporal sequence s_2 if s_1 is minimal and equivalent to s_2.

$$\begin{aligned}&(\forall\ s_1,s_2)[min\text{-}equiv\text{-}tseq(s_1,s_2)\\&\qquad \equiv min\text{-}tseq(s_1) \wedge tseq(s_2)\\&\qquad\qquad \wedge (\forall\ t,t_2)[TemporalEntity(t) \wedge iinside\text{-}1(t,t_2) \wedge member(t_2,s_2)\\&\qquad\qquad\qquad \supset (\exists\ t_1)[member(t_1,s_1) \wedge iinside(t,t_1)]]]\end{aligned}$$

4.1.2 Temporal Sequences and Their Elements

"iinside-1" can now be generalized to the predicate "iinside", which covers both temporal entities and temporal sequences. A temporal entity is "iinside" a temporal sequence if it is "iinside-1" one of the elements of its minimal equivalent temporal sequence.

$$
\begin{aligned}
&(\forall\, t,s)[\mathit{iinside}(t,s) \\
&\qquad \equiv [\mathit{TemporalEntity}(t) \wedge \mathit{TemporalEntity}(s) \wedge \mathit{iinside\text{-}1}(t,s)] \\
&\qquad\qquad \vee [\mathit{TemporalEntity}(t) \wedge \mathit{tseq}(s) \\
&\qquad\qquad\qquad \wedge (\exists\, s_1,t_1)[\mathit{min\text{-}equiv\text{-}tseq}(s_1,s) \wedge \mathit{member}(t_1,s_1) \\
&\qquad\qquad\qquad\qquad \wedge \mathit{iinside\text{-}1}(t,t_1)]]]
\end{aligned}
$$

A notion of "isubset" can be defined on the basis of "iinside".

$$
\begin{aligned}
&(\forall\, s,s_0)[\mathit{isubset}(s,s_0) \equiv [\mathit{tseq}(s) \wedge \mathit{tseq}(s_0) \wedge (\forall\, t)[\mathit{member}(t,s) \\
&\qquad \supset \mathit{iinside}(t,s_0)]]]
\end{aligned}
$$

That is, every element of temporal sequence s is inside some element of the minimal equivalent temporal sequence of s_0.

A relation of "idisjoint" between two temporal sequences can also be defined.

$$
\begin{aligned}
&(\forall\, s_1,s_2)[\mathit{idisjoint}(s_1,s_2) \\
&\qquad \equiv [\mathit{tseq}(s_1) \wedge \mathit{tseq}(s_2) \\
&\qquad\qquad \wedge \sim(\exists\, t, t_1, t_2)[\mathit{member}(t_1,s_1) \wedge \mathit{member}(t_2,s_2) \\
&\qquad\qquad\qquad \wedge \mathit{iinside}(t,t_1) \wedge \mathit{iinside}(t,t_2)]]]
\end{aligned}
$$

That is, temporal sequences s_1 and s_2 are disjoint if there is no overlap between the elements of one and the elements of the other.

The last temporal entity in a temporal sequence is the one with any of the others "ibefore" it.

$$(\forall\, t,s)[last(t,s) \equiv [tseq(s) \wedge member(t,s)$$
$$\wedge\ (\forall\, t_1)[member(t_1,s) \supset [t_1 = t \vee ibefore(t_1,t)]]]]$$

More generally, the *n*th element of temporal sequence can be defined.

$$(\forall\, t,s)[nth(t,n,s)$$
$$\equiv [tseq(s) \wedge member(t,s) \wedge natnum(n)$$
$$\wedge\ (\exists\, s_1)[(\forall\, t_1)[member(t_1,s_1)$$
$$\equiv [member(t_1,s) \wedge ibefore(t_1,t)]] \wedge card(s_1) = n\text{-}1]]]$$

That is, the *n*th element of a temporal sequence has *n*-1 elements before it.

4.1.3 *Everynthp*

The predicate "ngap" will enable us to define "everynthp" below. Essentially, the idea is a temporal sequence *s* containing every *n*th element of s_0 for which *p* is true. The predicate "ngap" holds between two elements of *s* and says that there are *n*-1 elements between them that are in s_0 and not in *s* for which *p* is true.

$$(\forall\, t_1,t_2,s,s_0,p,n)\ [ngap(t_1,t_2,s,s_0,p,n)$$
$$\equiv [member(t_1,s) \wedge member(t_2,s) \wedge tseqp(s,p)$$
$$\wedge\ tseq(s_0) \wedge isubset(s,s_0) \wedge natnum(n)$$
$$\wedge\ (\exists\, s_1)[card(s_1) = n\text{-}1 \wedge idisjoint(s,s_1)$$
$$\wedge\ (\forall\, t)[member(t,s_1)$$
$$\equiv [iinside(t,s_0) \wedge p(t) \wedge ibefore(t_1,t) \wedge ibefore(t,t_2)]]]]]$$

The predicate "everynthp" says that a temporal sequence *s* consists of every *n*th element of the temporal sequence s_0 for which property *p* is true. It will be useful in describing temporal aggregates like "every third Monday in 2001", where s is the desired temporal sequence("every third Monday in 2001"), s_0 is the context temporal sequence ("in 2001"), n is 3 ("third"), and p is Monday.

$$
\begin{aligned}
&(\forall\, s,s_0,p,n)[everynthp(s,s_0,p,n)\\
&\qquad \equiv [tseqp(s,p) \wedge\ tseq(s_0) \wedge\ natnum(n)\\
&\qquad\qquad \wedge\ (\exists\, t_1)[nth(t_1,1,s) \wedge\ \sim(\exists\, t][iinside(t,s_0) \wedge\ ngap(t,t_1,s,s_0,p,n)]]\\
&\qquad\qquad \wedge\ (\exists\, t_2)[last(t_2,s) \wedge\ \sim(\exists\, t][iinside(t,s_0) \wedge\ ngap(t_2,t,s,s_0,p,n)]]\\
&\qquad\qquad \wedge\ (\forall\, t_1)[last(t_1,s) \vee (\exists\, t_2)\ ngap(t_1,t_2,s,s_0,p,n)]]]
\end{aligned}
$$

That is, the first element in s has no p element n elements before it in s_0, the last element in s has no p element n elements after it, and every element but the last has a p element n elements after it.

The variable for the temporal sequence s_0 is, in a sense, a context parameter. When we say "every other Monday", we are unlikely to mean every other Monday in the history of the Universe. The parameter s_0 constrains us to some particular segment of time. (Of course, that segment could in principle be the entire time line.)

The definition of "everyp" is simpler:

$$
\begin{aligned}
&(\forall s,s_0,p)[everyp(s,s_0,p)\\
&\qquad \equiv (\forall\, t)[member(t,s) \equiv [iinside(t,s_0) \wedge\ p(t)]]]
\end{aligned}
$$

It is a theorem that every p is equivalent to every first p.

$$(\forall s,s_0,p)[everyp(s,s_0,p) \equiv everynthp(s,s_0,p,1)]$$

"every-other-p" could be defined similarly.

More examples will be shown in Section 4.3 to illustrate how to use our ontology to represent different kinds of temporal aggregates expressions.

4.2 Embedding iCalendar Recurrence Sets in OWL-Time

Internet Calendaring and Scheduling Core Object Specification (iCalendar) is a widely supported standard for personal data interchange. It provides the definition of a common format for openly exchanging calendaring and scheduling information across the Internet. The recurrence set in iCalendar is the complete set of recurrence instances for a calendar component. iCalendar recurrence sets are a subset of OWL-Time temporal sequences. For example, "everyday in January, for 3 years" can be expressed as[20]

```
DTSTART;TZID=US-Eastern:19980101T090000
RRULE:FREQ=YEARLY;UNTIL=20000131T090000Z;
BYMONTH=1;BYDAY=SU,MO,TU,WE,TH,FR,SA
```

or

```
RRULE:FREQ=DAILY;UNTIL=20000131T090000Z;BYMONTH=1
```

DTSTART specifies the start time, and RRULE specifies the recurrence rule. The properties of the recurrence rule (e.g., FREQ, UNTIL) are discussed in this section. More examples will be shown in Section 4.3.

A systematic way has been developed to map a recurrence set in iCalendar to a temporal sequence in OWL-Time. Here only an illustrative subset of recurrence sets will be demonstrated.

As shown in the next section with natural language examples, there are cases that can be expressed in OWL-Time more accurately than in iCalendar, and there are

[20] The example is taken from iCalendar RFC 2445 page 119.

also cases that iCalendar can't express, but OWL-Time can. Moreover, embedding recurrence sets in OWL-Time gives access to the full ontology of time for temporal reasoning.

4.2.1 Reify Recurrence Rules

In iCalendar, a recurrence set *s* is defined by a recurrence rule *r* which can be expressed as "*RecurrenceSetRule*(*s*,*r*)". First, we list the properties of a recurrence rule *r*:

$$(\forall\ r)\ [rule(r) \supset (\exists\ tu,g)\ [freq(r) = tu \wedge\ TemporalUnit(tu) \wedge\ gap(g,r)]]$$

"freq(r)" corresponds to the FREQ property of recurrence rules. Since it is a required property, it is defined as a function mapping from a recurrence rule to a temporal unit, *tu*. For example, FREQ = WEEKLY will have a return value of temporal unit *Week*.

"gap(g,r)" corresponds to the INTERVAL property of recurrence rules. For example, "every 3rd month" will have an INTERVAL property with a value of 3. If it is missing, we assume it is given a default value of 1 during the translation process from iCalendar to OWL-Time.

It is required in iCalendar that either UNTIL or COUNT property may appear in a recurrence rule, but they must not occur in the same rule.

$$\begin{aligned}
&(\forall\ r)\ [rule(r) \\
&\qquad \supset (\exists\ n)\ [count(n,r) \wedge\ natnum(n)] \\
&\qquad\qquad \vee (\exists\ t,y,mo,d,h,mi,s,z)\ [until(t,r) \wedge\ timeOf(t,y,mo,d,h,mi,s,z)]]
\end{aligned}$$

"count(n,r)" corresponds to the COUNT property of recurrence rules. "until(t,r)" corresponds to the UNTIL property of recurrence rules.

Then, we specify the optional properties of a recurrence rule as in:

$$(\forall\ r, ls)\ [rule(r) \wedge bysecond(ls, r)$$
$$\supset (\forall\ s)\ [member(s, ls) \supset integer(s) \wedge 0 \leq s \leq 59]]$$

This corresponds to the BYSECOND property of recurrence rules. BYMINUTE, BYHOUR, and so on are defined similarly.

4.2.2 Map Recurrence Sets

Now we can map from the recurrence set generated by a recurrence rule *r* with either COUNT or UNTIL to a temporal sequence *s*:

$$(\forall\ r,n,s,g)\ [RecurrenceSetRule(s,r) \wedge count(n,r) \wedge gap(g,r)$$
$$\supset (\exists\ s_0)\ [card(s)=n \wedge everynthp(s,s_0,map2p(freq(r)),g)]]$$

$$(\forall\ r,t,s,g)\ [RecurrenceSetRule(s,r) \wedge until(t,r) \wedge gap(g,r)$$
$$\supset (\exists\ t',s_0)\ [last(t',s_0) \wedge ends(t,t') \wedge everynthp(s,s_0,map2p(freq(r)),g)]]$$

"map2p" is a function that maps from temporal units to their corresponding unary predicate names. For example,

$$map2p(*year*) = year1,$$

where $(\forall\ y)\ [year1(y) \equiv (\exists\ n,x)\ [calInt(y,n,*Year*,x)]]$

For example, the recurrence rule for expressing "daily for 10 occurrences" is:[21]

RRULE:FREQ=DAILY;COUNT=10

[21] The example is taken from iCalendar RFC 2445 page 118.

which can be expressed as:

$$RecurrenceSetRule(s,r) \wedge count(10,r) \wedge gap(1,r)$$

This can be translated to:

$$(\exists\, s_0)\ [card(s)=10 \wedge everynthp(s,s_0,map2p(freq(r)),1)]$$
$$\supset (\exists\, s_0)\ [card(s)=10 \wedge everynthp(s,s_0,map2p(*day*),1)]$$
$$\supset (\exists\, s_0)\ [card(s)=10 \wedge everynthp(s,s_0,day1,1)]$$
$$\supset (\exists\, s_0)\ [card(s)=10 \wedge everyp(s,s_0,day1)]$$

where $(\forall\, d)\ [day1(d) \equiv (\exists\, n,x)\ [calInt(y,n,*Day*,x)]]$

iCalendar can express very complicated recurrence sets. For example, "The 1st and 2nd hours of the 4th and 5th months of every year" will have a recurrence rule in which BYHOUR = 1, 2, BYMONTH = 4, 5, and FREQ = YEARLY.

To formalize this we need recursion through the sequence of temporal units (with the predicate "everyithtempunit") and recursion through the list of integers associated with each temporal unit (with the predicate "byTulistRecurs"). We do the latter first.

$$(\forall\, s,ls,r,tu,g)\ [RecurrenceSetRule(s,r) \wedge bytempunit\,(ls,r,tu) \wedge gap(g,r)$$
$$\supset (\exists\, s',s_0)\ [everynthp(s',s_0,map2p(freq(r)),g)$$
$$\wedge\ byTulistRecurs(s,ls,s',tu,freq(r))]]$$

$$(\forall\, s,ls,s',tu,tu')\ [byTulistRecurs\,(s,ls,s',tu,tu') \wedge card(ls) > 1$$
$$\supset (\exists\, s_1,ls_1,i,s_i,i)\ [ls = union(\{i\},ls_1) \wedge \sim member(i,ls_1)$$
$$\wedge\ everyithtempunit(s_i,s',i,tu,tu') \wedge s = union(s_1, s_i)$$
$$\wedge\ byTulistRecurs(s_1,ls_1,s',tu,tu')]]$$

Base case of the recursion:

$(\forall\ s,s',i,tu,tu')\ [byTulistRecurs(s,\{i\},s',tu,tu')$

$\supset everyithtempunit(s,s',i,tu,tu')]$

"bytempunit" is a generalized relation from "bysecond", "byminute", and so on. It is a relation among a recurrence rule *r*, a temporal unit *tu*, and a list of values associated with BYxxx with that temporal unit. For example,

$(\forall\ ls,r)\ [bytempunit(ls,r,*Second*) \equiv bysecond(ls,r)]$

"byTulistRecurs" is a relation among two temporal sequences (*s*, *s'*), their temporal units (*tu*, *tu'*), and a list of values associated with *tu*.

"everyithtempunit" is an important relation among two temporal sequences (*s*, *s'*), their temporal units (*tu*, *tu'*), and an integer value *i*. For example, "*everyithtempunit*(*s*,*s*',2,**Month**,**Year**)" specifies that the members of temporal sequence *s* are every 2nd month (i.e., February) of the members of temporal sequence *s*' (with temporal unit of *Year*); in English, it means "every other February" with an implicit context, i.e., a sequence of years.

It's possible that the temporal units of FREQ and BYxxx are not successive. For example, "the 1st two hours in every month", which can be expressed in iCalendar with FREQ = MONTHLY, BYHOUR = 1, 2. The temporal units of FREQ (*Month*) and BYHOUR (*Hour*) are not successive, and what the example actually says is "the 1st two hours *of the 1st day* of every month". Our axiom needs to be able to handle this case, as well as the normal case where the temporal units are successive. The predicate "everyithtempunit" is defined recursively:

$(\forall\ s_1,s_2,i,tu_1,tu_2)\ [everyithtempunit\ (s_1,s_2,i,tu_1,tu_2)$

$\wedge$ *tulevel*(tu_2) > *tulevel*(tu_1)+1 $\wedge$ $tu_1 \neq$ **Monthday**

$\wedge$ $tu_1 \neq$ **Yearday** $\wedge$ $tu_1 \neq$ **Week**

$\supset (\exists\, s')$ [*everyithtempunit*(s_1,s',i,tu_1,*level2tu*(*tulevel*(tu_2)-1))

$\wedge$ *everyithtempunit*(s',s_2,1,*level2tu*(*tulevel*(tu_2)-1),tu_2)]]

Base case:

$(\forall\, s_1,s_2,i,tu_1,tu_2)$ [*everyithtempunit* (s_1,s_2,i,tu_1,tu_2)

$\wedge$ *tulevel*(tu_2) = *tulevel*(tu_1)+1 $\wedge$ $tu_1 \neq$ **Monthday**

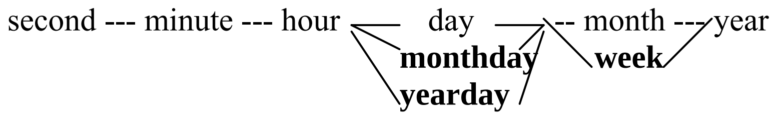
$\wedge$ $tu_1 \neq$ **Yearday** $\wedge$ $tu_1 \neq$ **Week**

$\supset (\forall\, t_1)$ [*member*(t_1,$s1$) $\supset (\exists\, t_2)$ [*member*(t_2,s_2)

$\wedge$ *iinside-1*(t_1,t_2) $\wedge$ *calclockInt*(t_1,i,tu_1,t_2)]]]

where *calclockInt*(y,n,u,x) $\equiv$ *calInt*(y,n,u,x) $\vee$ *clockInt*(y,n,u,x)

"tulevel" is a function mapping from a temporal unit to its level value according to the hierarchy below. For example, "tulevel(*second*) = 1, tulevel(*month*) = 5". "level2tu" is an inverse function of "tulevel". It maps from the level value to the associated temporal unit. For example, "level2tu(1) = *second*, level2tu(5) = *month*". This hierarchy is consistent with the temporal unit ordering in iCalendar.



The temporal units in bold, i.e., *Monthday*, *Yearday*, and *Week*, need to be axiomatized sepecially.

- When *Monthday* and *Month* are together,:

everyithtempunit(s_1, s_2, i, **Monthday**, **Month**)

$\supset (\forall\ t_1)\ [member(t_1,s_1) \supset (\exists\ t_2)\ [member(t_2,s_2) \wedge iinside\text{-}1(t_1,t_2)$
$\wedge\ calclockInt(t_1,i,*Monthday*,t_2)]]]$

- When *Yearday* and *Year* are together, force them to the base case, i.e., skip *Month*:

$everyithtempunit(s_1,\ s_2,\ i,\ *Yearday*,\ *Year*)$
$\supset (\forall\ t_1)\ [member(t_1,s_1) \supset (\exists\ t_2)\ [member(t_2,s_2) \wedge iinside\text{-}1(t_1,t_2)$
$\wedge\ calclockInt(t_1,i,*Yearday*,t_2)]]]$

- When *Week* and *Year* are together:

$everyithtempunit(s_1,\ s_2,\ i,\ *Week*,\ *Year*)$
$\supset (\forall\ t_1)\ [member(t_1,s_1) \supset (\exists\ t_2)\ [member(t_2,s_2) \wedge iinside\text{-}1(t_1,t_2)$
$\wedge\ calclockInt(t_1,i,*Week*,t_2)]]]$

For example, the recurrence rule for the expression "monthly on the 2nd and 15th of the month for 10 occurrences" is[22]:

RRULE:FREQ=MONTHLY;COUNT=10;BYMONTHDAY=2,15

which can be expressed as:

$RecurrenceSetRule(s,r) \wedge\ count(10,r) \wedge\ bymonthday(\{2,15\},r)$

This can be translated to:

$(\exists\ s_0)\ [card(s)=10 \wedge\ byTulistRecurs(s,\{2,15\},s_0,*Monthday*,*Month*)]$
$\supset (\exists\ s_0,s_1,s_2)\ [\text{everyithtempunit}(s_1,s_0,2,\text{*Monthday*},\text{*Month*})$
$\wedge\ everyithtempunit(s_2,s_0,15,*Monthday*,*Month*)$
$\wedge\ card(s)=10 \wedge\ s = union(s_1,s_2)]$

[22] The example is taken from iCalendar RFC 2445 page 121.

iCalendar allows one to specify lists of dates/times as well, with the attributes RDATE and EXDATE. These are translated into simple temporal sequences. A recurrence set is generated by generating recurrence sets specified by the RRULE and/or RDATE attributes and by the EXRULE and/or EXDATE attributes, subtracting the latter from the former, and constraining the result by the DTSTART attribute.

4.3 Natural Language Examples in FOL

In the previous section, we've shown how iCalendar statements can be systematically mapped to OWL-Time predicates; in this section we demonstrate how these predicates in OWL-Time can be used to express natural language expressions.

We will first take an example from iCalendar, and show how to express it in OWL-Time. Then we will show an example that OWL-Time can do better than iCalendar. After showing an example that iCalendar can't handle, but OWL-Time can, we will show more natural language expressions that can be represented using OWL-Time predicates.

1) "*Every other week on Monday, Wednesday and Friday until December 24, 1997, but starting on Tuesday, September 2, 1997.*"[23]

```
DTSTART;TZID=US-Eastern:19970902T090000
RRULE:FREQ=WEEKLY;INTERVAL=2;UNTIL=19971224T000000Z;
WKST=SU;BYDAY=MO,WE,FR
```

[23] The example is taken from iCalendar RFC 2445 page 120.

iCalendar uses “FREQ=WEEKLY;INTERVAL=2” to represent “every other week”, and uses “BYDAY=MO, WE,FR” to get “every Monday, Wednesday, and Friday”. WKST specifies the start of the week, which is Sunday in this example.

This example can be expressed in OWL-Time as a temporal sequence *s* for which the following is true:

$$\begin{aligned}&(\exists\, s,s',T,t_1,t_2)\ [\mathit{everynthp}(s',\{T\},\mathit{Week1},2)\\ &\quad\wedge\ \mathit{byTulistRecurs}(s,\{1,3,5\},s',{*}\mathit{Day}{*},{*}\mathit{Week}{*})\\ &\quad\wedge\ \mathit{begins}(t_1,T)\wedge \mathit{ends}(t_2,T)\wedge\ \mathit{dateOf}(t_1,1997,9,2)\\ &\quad\wedge\ \mathit{dateOf}(t_2,1997,12,24)]\\ &\text{where } (\forall\, w)\ [\mathit{Week1}(w)\equiv(\exists\, n,x)\ [\mathit{calInt}(w,n,{*}\mathit{Week}{*},x)]]\end{aligned}$$

s is the desired temporal sequence representing “every Monday, Wednesday, and Friday” of *s’* which is a temporal sequence of “every other week from 09/02/1997 to 12/24/1997”.

2) “*Every 3rd Monday in 2001.*”

This looks like a simple example, but iCalendar can’t express it exactly. Though iCalendar can express “the 3rd Monday” using BYDAY=3MO, it can’t express “every 3rd Monday”. In order to express this phrase in iCalendar, we have to paraphrase it first to: “Every 3rd week on Monday in 2001”.

However, the paraphrased one is not exactly the same as the original, since it depends on what the first week is. In iCalendar, the first week is defined as the week that “contains at least four days in that calendar year”. Based on this definition, however, if the first week starts from a Tuesday, the first 3rd Monday will be different from the first Monday of the 3rd week!

In order to express year 2001, in iCalendar it has to specify the start date (01/01/2001) using DTSTART and the end date (12/31/2001) using UNTIL. Here is how to express "Every 3rd week on Monday in 2001" in iCalendar:

```
DTSTART;TZID=US-Eastern:20010101T000000
RRULE:FREQ=WEEKLY;INTERVAL=3;UNTIL=20011231T000000Z;
WKST=SU;BYDAY=MO
```

In OWL-Time, this example can be expressed very straightforwardly as a temporal sequence *s* for which the following is true:

$$(\exists\, y,z)\ [yr(y,2001,CE(z)) \wedge everynthp(s,\{y\},Monday1,3)]$$

$$\text{where } (\forall\, d)\ [Monday1(d) \equiv (\exists\, w)\ [Monday(d,w)]]$$

3) "*Every Monday that's a holiday.*"

There is no way in iCalendar to express "conditional temporal aggregates" or any temporal aggregates that are not in a form of standard temporal units, such as "holidays", "voting dates", "days with classes", "months starting with a Monday", and so on.

OWL-Time, however, can express any kind of temporal aggregates, using the argument p in "$everynthp(s,s_0,p,n)$" or "$everyp(s,s_0,p)$". All the conditions and non-temporal-unit concepts can be captured using this p.

For example, we can express "Every Monday that's a holiday" in OWL-Time as a temporal sequence *s* for which the following is true:

$$(\exists\ s,s_0)\ [everyp(s,s_0,HolidayMonday)]$$

$$\text{where}\ \ (\forall\ d)\ [HolidayMonday(d) \equiv (\exists\ w)\ [Monday(d,w)] \wedge\ holiday(d)]$$

4) More natural language expressions represented in OWL-Time are as follows, where *s* is the set corresponding to the noun phrase:

- "*The past three Mondays.*"

$$(\exists\ s,s_0,t)\ [everyp(s,s_0,Monday1) \wedge card(s) = 3 \wedge\ last(t,s_0) \wedge\ ends(nowfn(D),t)]$$

In our treatment of temporal deictics, the function "nowfn" maps a document *d* into the instant or interval viewed as "now" from the point of view of that document, and *D* is the document this phrase occurs in.

- "*Every other Monday in every 4th month in every year.*"

$$(\exists\ s,s_1,s_2,s_0)\ [everyp(s_1,s_0,Year1) \wedge\ everynthp(s_2,s_1,Month1,4) \wedge\ everynthp(s,s_2,Monday1,2)]$$

$$\text{where}\ \ (\forall\ y)\ [Year1(y) \equiv (\exists\ n,x)\ [calInt(y,n,{*}Year{*},x)]]$$

$$(\forall\ m)\ [Month1(m) \equiv (\exists\ n,x)\ [calInt(m,n,{*}Month{*},x)]]$$

- "*Four consecutive Mondays.*"

$$(\exists\ s,s_0)\ [everyp(s,s_0,Monday1)\ \wedge\ card(s) = 4]$$

In order to translate from natural language sentences to our representation, we first run a semantic parser to get a "surface" logical form from a given sentence. Then we use some additional rules to map from the "surface" logical form to our domain ontology. For example, given the above input sentence, the "surface" logical form would be:

$$(\exists\ s,d)\ [plur(d,s) \wedge\ Monday1(d) \wedge\ consecutive(s, Monday1) \wedge\ card(s) = 4]$$

"plur" takes as its arguments both a set and a representative member of the set. "consecutive" takes as its arguments both a set and the property of the set, meaning that the members of the set are not only consecutive but also share a certain property. "card" is a function that returns the cardinality/size of the set.

Then we map the above "surface" logical form to our ontology predicates by applying the following rule:

$(\forall\ s,p)\ consecutive(s,p) \equiv (\exists\ s_0)\ everyp(s,s_0,p)$

4.4 Temporal Aggregates in OWL

4.4.1 Temporal Sequences and Their Members

In order to encode the temporal aggregates ontology in OWL, we first defined a temporal sequence. It has only one optional property "hasMemeber" which maps from a temporal sequence to any temporal entity. A temporal sequence can have no (empty sequence) or many members:

```
<owl:Class rdf:ID="TemporalSeq">
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty rdf:resource="#hasMember" />
               <owl:minCardinality
                     rdf:datatype="&xsd;nonNegativeInteger">0
               </owl:minCardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasMember">
   <rdfs:domain rdf:resource="#TemporalSeq" />
   <rdfs:range  rdf:resource="#TemporalEntity" />
</owl:ObjectProperty>
```

Since we also want to have a backward link pointing from the temporal sequence member to its associated sequence, a "TemporalSeqMember" class is defined. It's a subclass of “TemporalEntity”, and 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:

```
<owl:Class rdf:ID=“TemporalSeqMember”>
  <rdfs:subClassOf rdf:resource=“#TemporalEntity”/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=“#isMemberOf” />
          <owl:cardinality rdf:datatype=“&xsd;nonNegativeInteger”>1
             </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=“#hasPosition” />
          <owl:cardinality rdf:datatype=“&xsd;nonNegativeInteger”>1
             </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID=“isMemberOf”>
  <rdfs:domain rdf:resource=“#TemporalSeqMember” />
  <rdfs:range  rdf:resource=“#TemporalSeq” />
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID=“hasPosition”>
  <rdfs:range  rdf:resource=“&xsd;integer” />
</owl:DatatypeProperty>
```

Since “hasPosition” is also used for other classes, as will see later, it only has a range of integers.

For a temporal sequence member that is associated with multiple sequences, multiple instances of “TemporalSeqMember” must be defined. The reason for defining it in this way is that for a given temporal sequence member instance, it will only have one pair of “isMemberOf” and “hasPosition” values, so that it’s not confusing which “hasPosition” value should be paired with which “isMemberOf” value. Moreover, different temporal sequences may apply different attributes to their members.

4.4.2 Temporal Aggregate Description

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, it specifies the temporal aggregate description for temporal sequences, and it's associated with the temporal sequence class by "hasTemporalAggregate-Description" property.

The temporal aggregate description has the following fields/properties: "hasStart", "hasEnd", "hasContextTemporalSeq", "hasithTemporalUnit", "hasTemporalUnit", "hasContextTemporalUnit", "hasPosition", "hasGap", and "hasCount":

```
<owl:Class rdf:ID="TemporalAggregateDescription">
  <rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource="#hasStart" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasCount" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasTemporalAggregateDescription">
  <rdfs:domain rdf:resource="#TemporalSeq" />
  <rdfs:range  rdf:resource="#TemporalAggregateDescription" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasStart">
  <rdfs:domain rdf:resource="#TemporalAggregateDescription" />
  <rdfs:range  rdf:resource="#Instant" />
</owl:ObjectProperty>
```

The optional properties "hasStart" and "hasEnd" map from the temporal aggregate description to an instant, specifying the start and the end instants of a temporal sequence. The calendar and clock properties described in Section 3 can then be used to specify the start and the end times or dates the instants are in.

```
<owl:ObjectProperty rdf:ID="hasContextTemporalSeq">
  <rdfs:domain rdf:resource="#TemporalAggregateDescription" />
  <rdfs:range  rdf:resource="#TemporalSeq" />
</owl:ObjectProperty>
```

The optional property "hasContextTemporalSeq" maps from the temporal aggregate description to the temporal sequence, specifying the context (super) temporal sequence of a given (sub) temporal sequence.

It corresponds to s_0 in "*everynthp*(s,s_0,p,n)" and *s'* in "*byTulistRecurs* (*s*,*ls*,*s*',*tu*,*tu*')". When it's not present, context-free temporal aggregates (e.g., "every Monday") can be represented.

```
<owl:DatatypeProperty rdf:ID="hasithTemporalUnit">
  <rdfs:domain rdf:resource="#TemporalAggregateDescription" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger" />
</owl:DatatypeProperty>
```

The required property "hasithTemporalUnit" maps from the temporal aggregate description to positive integers, specifying the *i*th temporal unit elements in the temporal sequence.

It 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" (such examples will be illustrated in detail in the next section).

```
<owl:ObjectProperty rdf:ID="hasTemporalUnit">
  <rdfs:domain rdf:resource="#TemporalAggregateDescription" />
  <rdfs:range  rdf:resource="#TemporalUnit" />
</owl:ObjectProperty>
```

The required properties “hasTemporalUnit” and the optional “hasContextTemporalUnit” map from the temporal aggregate description to the temporal unit. They 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 context temporal unit is associated with the context temporal sequence property. Thus if the context temporal sequence is not present, so is the context temporal unit, but not vice versa, since it’s possible that the temporal unit of the context temporal sequence is unknown or not relevant.

```
<owl:DatatypeProperty rdf:ID=“hasPosition”>
  <rdfs:range  rdf:resource=“&xsd;integer” />
</owl:DatatypeProperty>
```

The optional property “hasPosition” is a shared property with the “TemporalSeqMember” class. It specifies the position of the element in the temporal sequence. For example, “the first two Tuesdays in every May” would have “hasPosition” value of 2. It’s also possible to have negative positions. For example, “the last Thursday in every November” would have “hasPosition” value of -1.

If this property value is not present, all the positions will be included in the temporal sequence. For example, “the Thursdays in every November” includes all the Thursdays in every November, while “the last Thursday in every November” only includes the last Thursday in every November.

```
<owl:DatatypeProperty rdf:ID=“hasGap”>
  <rdfs:domain rdf:resource=“#TemporalAggregateDescription” />
  <rdfs:range  rdf:resource=“&xsd;positiveInteger” />
</owl:DatatypeProperty>
```

The optional property “hasGap” maps from the temporal aggregate description to positive integers, specifying 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,s_0,p,n)”. For example, “every 3rd Monday” would have “hasGap” value of 3.

```
<owl:DatatypeProperty rdf:ID="hasCount">
  <rdfs:domain rdf:resource="#TemporalAggregateDescription" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger" />
</owl:DatatypeProperty>
```

The optional property “hasCount” maps from the temporal aggregate description to positive integers, specifying the cardinality or the size of the temporal sequence. For example, “four consecutive Sundays” would have “hasCount” value of 4.

4.5 Natural Language Examples in OWL

In this section, we will take two natural language examples from Section 4.3, and express them in OWL. One example is a complex multiple-layered temporal aggregate, and the other is a conditional temporal aggregate. Both FOL and OWL representations will be shown.

- *Every other week on Monday, Wednesday and Friday until December 24, 1997, but starting on Tuesday, September 2, 1997.*[24]

FOL:
$(\exists$ s,s’,T,t_1,t_2) [everynthp(s’,{T},Week1,2) $\wedge$ byTulistRecurs(s,{1, 3, 5},s’,*Day*,*Week*)

[24] This example is taken from iCalendar RFC 2445 page 120.

$\wedge$ begins(t_1,T) $\wedge$ ends(t_2,T) $\wedge$ dateOf(t_1,1997,9,2) $\wedge$ dateOf(t_2,1997,12,24)]
where ($\forall$ w) [Week1(w) $\equiv$ ($\exists$ n,x) [calInt(w,n,*Week*,x)]]

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

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

<time-entry:TemporalAggregateDescription rdf:ID="everyOtherWeek">
<time-entry:hasTemporalUnit rdf:resource="&time-entry;unitWeek" />
<time-entry:hasGap rdf:datatype="&xsd;positiveInteger">2</time-entry:hasGap>
</time-entry:TemporalAggregateDescription>

<time-entry:TemporalAggregateDescription rdf:ID="MWFeveryOtherWeek">
  <time-entry:hasStart rdf:resource="#tseqStart" />
  <time-entry:hasEnd rdf:resource="#tseqUntil" />
<time-entry:hasContextTemporalSeq rdf:resource="#tseq-everyOtherWeek" />
<time-entry:hasithTemporalUnit rdf:datatype="&xsd;positiveInteger">1
</time-entry:hasithTemporalUnit>
<time-entry:hasithTemporalUnit rdf:datatype="&xsd;positiveInteger">3
</time-entry:hasithTemporalUnit>
<time-entry:hasithTemporalUnit rdf:datatype="&xsd;positiveInteger">5
</time-entry:hasithTemporalUnit>
<time-entry:hasTemporalUnit rdf:resource="&time-entry;unitDay" />
<time-entry:hasContextTemporalUnit rdf:resource="&time-entry;unitWeek" />
</time-entry:TemporalAggregateDescription>

<time-entry:Instant  rdf:ID="tseqStart">
<time-entry:inCalendarClock rdf:resource="#tseqStartDescription" />
</time-entry:Instant>

<time-entry:Instant  rdf:ID="tseqUntil">
<time-entry:inCalendarClockDataType rdf:datatype="&xsd;dateTime">1997-12-24
</time-entry:inCalendarClockDataType>
</time-entry:Instant>

<time-entry:CalendarClockDescription rdf:ID="tseqStartDescription">
  <time-entry:unitType rdf:resource="&time-entry;unitDay" />
<time-entry:year rdf:datatype="&xsd;gYear">1997</time-entry:year>
<time-entry:month rdf:datatype="&xsd;gMonth">9</time-entry:month>
  <time-entry:day rdf:datatype="&xsd;gDay">2</time-entry:day>
<time-entry:dayOfWeekField rdf:datatype="&xsd;nonNegativeInteger">2
</time-entry:dayOfWeekField>
</time-entry:CalendarClockDescription>
```

The FOL axiom defines *s* as the set corresponding to the given temporal aggregate. The first part of the axiom defines *s'* as the set corresponding to "every other week", and it serves as the context temporal sequence for the desired temporal sequence *s*. Predicates "begins" and "ends" are used to represent the start and the end times of the given temporal aggregate.

Besides what is shown in the first example, the OWL encodings for this one show how "hasithTemporalUnit" is used to represent a list of temporal elements in the temporal sequence (i.e., "on Monday, Wednesday, and Friday"), and how "hasStart" and "hasEnd" are used and combined with the calendar and clock representations to represent the start and end dates of the given temporal aggregate.

This example also shows the tradeoffs of using XSD dateTime and the "CalendarClockDescription" class defined in OWL-Time, as mentioned in Section 3. In this example, the end date is represented using XSD dateTime, while the start date is represented using the "CalendarClockDescription" class. As we can see, XSD dateTime is simpler, but there's some information it cannot represent, for example, the start date is Tuesday, and this is the reason why "CalendarClockDescription" class (with "dayOfWeekField" property) is used for the start date. In fact, "CalendarClockDescription" class can also represent other information that XSD dateTime cannot, such as "week" and "day of year". Moreover, each field of the class is separate so that it's easier to extract the values of some fields for the later use and easier to reason about.

- *Every Monday that's a holiday.*

FOL:
  $(\exists\ s,s_0)$ [everyp(s,s_0,HolidayMonday)]
  where $(\forall\ d)$ [HolidayMonday(d) $\equiv$ $(\exists\ w)$ [Monday(d,w)] $\wedge$ holiday(d)[25]]

OWL:
```
  <EveryHolidayMonday rdf:ID="tseq" />

  <owl:Class rdf:ID="EveryHolidayMonday">
```

[25] It says *d* is a Holiday.

```
    <rdfs:subClassOf rdf:resource="&time-entry;TemporalSeq"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasMember" />
    <owl:allValuesFrom rdf:resource="#EveryHolidayMondayMember" />
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:ID="EveryHolidayMondayMember">
  <rdfs:subClassOf  rdf:resource="&time-entry;TemporalSeqMember"/>
    <rdfs:subClassOf rdf:resource="&time-entry;Holiday"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#isMemberOf" />
    <owl:allValuesFrom rdf:resource="#EveryMonday" />
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:ID="EveryMonday">
    <rdfs:subClassOf rdf:resource="&time-entry;TemporalSeq"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasTemporalAggregateDescription" />
    <owl:hasValue rdf:resource="#everyMonday" />
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <time-entry:TemporalAggregateDescription rdf:ID="everyMonday">
  <time-entry:hasithTemporalUnit rdf:datatype="&xsd;positiveInteger">1
  </time-entry:hasithTemporalUnit>
  <time-entry:hasTemporalUnit rdf:resource="&time-entry;unitDay" />
  </time-entry:TemporalAggregateDescription>
```

This example shows an important advantage of using our temporal aggregates ontology. It can represent *conditional temporal aggregates* which is hard or impossible in some other representations. For example, there is no way in iCalendar to express such conditional temporal aggregates or any temporal aggregates that are not in a form of standard temporal units, such as "holidays", "voting dates", "days with classes", "months starting with a Monday", and so on.

As we can see, the FOL axiom is much simpler than the corresponding OWL encodings. It defines *s* as the set corresponding to the given temporal aggregate, and a new predicate (i.e., "HolidayMonday") for this conditional temporal aggregate.

The OWL encodings show how this kind of conditional temporal aggregates can be defined in our representation in OWL.

The most important class in this example is the “EveryHolidayMonday-Member” class which defines a class for the members of the desired temporal sequence class (i.e., “EveryHolidayMonday”). This class is both a temporal sequence member and a holiday, and its associated temporal sequence class must be “every Monday” (i.e., “EveryMonday” class).

The desired temporal sequence is an instance of the “EveryHolidayMonday” class whose members are only from the “EveryHolidayMondayMember” class.

4.6 Improving Set Expression Annotations in TimeML

4.6.1 TimeML and Its Set Expression Annotations

TimeML (Pustejovsky et al., 2002) is a rich specification language for event and temporal expressions in natural language text. Unlike most previous attempts at event and temporal specification, TimeML separates the representation of event and temporal expressions from the anchoring or ordering dependencies that may exist in a given text.

TimeML includes four major data structures: EVENT, TIMEX3, SIGNAL, AND LINK. EVENT is a cover term for situations that happen or occur, and also those predicates describing states or circumstances in which something obtains or holds true. TIMEX3, which extends TIMEX2 (Ferro, 2001), is used to mark up explicit temporal expressions, such as time, dates, and durations. SIGNAL is used to annotate sections of text, typically function words that indicate how temporal objects are related to each other (e.g., “when”, “during”, “before”). The set of LINK tags encode various relations that exist between the temporal elements of a document, including three subtypes: TLINK (temporal links), SLINK (subordination links), and ALINK (aspectual links).

In TimeML temporal aggregate expressions are tagged with type “SET” (i.e., set expressions), and attributes “quant” and “freq” are used to specify sets that denote quantified times in a TIMEX3. “quant” is generally a literal from the text that quantifies over the expression, and “freq” contains an integer value and a time granularity to represent any frequency contained in the set, just as a period of time is represented in a duration. Here are some examples taken from the TimeML specification:

<TIMEX3 tid=“t3” type=“SET” value=“P1M” freq=“2X”>
twice a month
</TIMEX3>

<TIMEX3 tid=“t4” type=“SET” value=“P1M” quant=“EVERY” freq=“3D”>
three days every month
</TIMEX3>

<TIMEX3 tid=“t5” type=“SET” value=“P1D quant=“EVERY”>
daily

```
</TIMEX3>
```

```
<TIMEX3 tid="t1" type="SET" value="P1W" freq="2X">
```
twice a week
```
</TIMEX3>
```

```
<TIMEX3 tid="t1" type="SET" value="P1W" quant="EACH" freq="3d">
```
3 days each week
```
</TIMEX3>
```

From the above examples, we can see that there are some problems with the current TimeML set expressions in the TIMEX3 tag:

(1) It is not scalable to multiple-layered temporal aggregates. It seems impossible to express multiple layer (more than two layers) temporal aggregates. For example, "every Monday of every other month in every 3rd year".

(2) It is hard to annotate arbitrary "every *n*th", because in theory "quant" needs to have infinitely many literals, such as "EVERY_OTHER", "EVERY_THIRD", and so on.

4.6.2 Suggestions for Improvements

Here are some suggestions to improve the set expression annotations:

(1) Instead of annotating one entire temporal aggregate phrase at a time, we can break them into multiple embedded primitive (single layer) temporal aggregates.

For example, “every Monday of every other month in every 3rd year” can be broken (bracketed) into three primitive temporal aggregates:

[every Monday] of [every other month] in [every 3rd year].

In order to remember the embedding relations between the primitive temporal aggregates, some reference/anchoring attribute such as “anchorTimeID” can be used.

(2) To express “every *n*th” in a more scalable way, a new attribute (called “gap”) used to express the “*n*th” can replace the attribute “quant”. If the “gap” attribute is not present, the default value of 1 (i.e., “every/each”) will be applied. For example, “every Monday” would have a gap value of 1 (default value, the attribute is not necessary to be shown in the tag), “every other Monday” would have a gap value of 2, “every 3rd Monday” would have a gap value of 3, and so on.

With these two suggested modifications, the phrase “every Monday of every other month in every 3rd year” can then be annotated as follows:

<TIMEX3 tid=“t1” type=“SET”
value=“XXXX-WXX-1” temporalFunction=“true” **anchorTimeID=“t2”**>
every Monday
</TIMEX3>
of
<TIMEX3 tid=“t2” type=“SET” value=“P1M” **gap=2**
temporalFunction=“true”
anchorTimeID=“t3”>
every other month
</TIMEX3>
in

<TIMEX3 tid="t3" type="SET" value="P1Y" **gap=3**>
every 3rd year.
</TIMEX3>

(3) The "value" and "freq" attributes looks very confusing in set expressions. Based on the examples shown at the beginning of the section, we can see that the roles of the "value" and "freq" attributes are as follows

The "freq" attribute represents the *inner layer* temporal entities that repeat. For example, "twice" in "twice a month"; "three days" in "three days every month". If it's a single layer temporal aggregate, it simply represents the only temporal entity that repeats. For example, "October" in "every October" and "2 days" in "every 2 days".

The "value" attribute, on the other hand, represents the *outer layer* temporal entities that repeat, if it's a two layer temporal aggregates. For example, "a month" (monthly) in "twice a month"; "each week" (weekly) in "3 days each week".

The problem with this structure is that it's hard to scale up to more than two layer temporal aggregates. With the suggestion (1), we can actually solve this problem by breaking multiple layer temporal aggregates to embedded primitive (single layer) temporal aggregates. This suggestion would also eliminate the usage of the "value" attribute (i.e., for outer layer temporal aggregates) for set expressions, since all the temporal aggregates that need to be annotated would be primitive (single layer) ones.

Since the "value" attribute is used elsewhere, in order to minimize the number of total attributes, we could actually remove the "freq" attribute and use the "value" attribute in the way that the current `freq` attribute is used, i.e., representing the only temporal entity that repeats. Please note, in this way the “value” attribute will have to be able to take any CDATA values.

By adopting this suggestion, the two layer temporal aggregate examples above will be changed to something as follows:

- *twice a month*

<TIMEX3 tid=“t1” type=“SET” **value=“2X”** temporalFunction=“true”
anchorTimeID=“t2”>
twice
</TIMEX3>
<TIMEX3 tid=“t2” type=“SET” value=“P1M”>
a month
</TIMEX3>

- *three days every month*

<TIMEX3 tid=“t3” type=“SET” **value=“P3D”** temporalFunction=“true”
anchorTimeID=“t4”>
three days
</TIMEX3>
<TIMEX3 tid=“t4” type=“SET” value=“P1M” >
every month
</TIMEX3>

4.7 Evaluation

We evaluate the coverage of the temporal aggregates ontology against the temporal aggregates expressions extracted from the ACE (Automatic Content Extraction) corpus[26] (Doddington et. al., 2004). There are a total of 219 temporal aggregate expressions extracted from the corpus, many of which were found difficult to annotate by TIMEX2 (Ferro, 2001), an annotation schema for temporal expressions that provides the basis for the TIMEX3 tag element of TimeML. In fact, in the previous section we proposed a different annotation schema to improve set (i.e., temporal aggregate) expression annotations in TimeML.

Among all those temporal aggregate expressions, 11 expressions cannot be fully represented by the current temporal aggregate ontology, so the coverage is 208 / 219 = **95.0%**. Table 4.1 lists all those difficult temporal aggregate expressions associated with their frequencies in the corpus. We can see that the difficulties are mainly due to the terms in the expressions that convey vagueness or uncertainty, such as "almost", "several", "nearly", and "recent". Although an underlying theory of vagueness is required to fully solve this problem, we can still represent the expressions if we can give them ad-hoc definitions, for example, "several" as 2 to 5. Section 6 describes our work on modeling and extracting vague event durations from text, as well as how to represent and reason about them.

[26] Thanks to Laurie Gerber for collecting and sharing the temporal aggregate data from the corpus.

Temporal Aggregate Expressions	**Frequency**
almost every night	3
almost weekly	1
every *several* hundred thousand years	1
nearly every day	2
recent days	1
recent years	1
several summers	1
the last *several* nights	1

Table 4.1 Difficult Temporal Aggregate Expressions from the ACE corpus

Chapter 5

Temporal Arithmetic Mixing Months and Days

Temporal arithmetic involves the computation of adding durations to dates/times, subtracting durations from dates/times, and computing durations between date/time pairs. Such arithmetic (sometimes is also referred as date arithmetic when only dates are of interest) is very useful in many different domains, such as artificial intelligence (Bettini et al., 2002), databases (Chandra and Segev, 1994), time series management systems (Dreyer et al., 1994), agents (Mallya et al., 2004), and Web services (Pan and Hobbs, 2004).

As long as we stay within the year-month system or the day-hour-minute-second system, temporal arithmetic is just arithmetic and requires only a few simple axioms or rules to encode. However, when we mix months and days, problems arise. For example, consider that January 31, 2006 plus 2 months equals March 31, 2006. But if we add the months one at a time, we may get a different result: January 31, 2006 plus one month is February 28, 2006, but February 28, 2006 plus one month would seem to be March 28, 2006. If we want to avoid results like this, we need, in some sense, to keep track of the history of the computation.

This *motivating example* is based on the documentation for Java methods add() and roll() of the class Calendar[27], where they explained the behavior of the methods using the above example and claimed that, “as most users will intuitively expect”, the final date after adding the second month should be March 31, 2006, not March 28, 2006, and this is what you will get when you use the temporal arithmetic methods in Java. (In what follows, we will refer to this kind of intuition as the “*history-dependent intuition*”.)

However, different people may have different intuitions regarding how the computation should work. Some people may believe adding one month to February 28 should always be March 28, no matter where February 28 was originally computed from. (In what follows, we will refer to this kind of intuition as the “*history-independent intuition*”.) This intuition results in much simpler arithmetic rules, where when adding durations to dates, simply add each of their fields (temporal units) respectively, and handle overflow when needed. In fact, this approach is what XML Schema is currently using in their algorithm for adding “durations” to “dateTimes”[28] (Biron and Malhotra, 2004). But they didn’t consider at all the alternative (maybe even more popular) intuition (the “history-dependent intuition”) we described in the above motivating example.

[27] http://java.sun.com/j2se/1.4.2/docs/api/java/util/Calendar.html
[28] http://www.w3.org/TR/xmlschema-2/#adding-durations-to-dateTimes

Biron and Malhotra (2004) also warn that there are cases where the properties of *commutativity* and *associativity* cannot hold using the algorithm in XML Schema, and give one example as follows:

(2000-03-30 + P1D) + P1M = 2000-03-31 + P1M = 2000-04-30

(2000-03-30 + P1M) + P1D = 2000-04-30 + P1D = 2000-05-01

This example shows that adding one month and one day to March 30, 2000 in different orders using their algorithm would give different destination dates.

Many real-world applications, on the other hand, use a fixed number of days for each month (e.g., car rental companies usually count 28 days as one month), but they also cannot avoid the problem raised when mixing months and days. For example, Stonebraker (1990) pointed out that the yield calculation on financial bonds uses a calendar that has 30 days in every month for date arithmetic, but 365 days in the year for the actual yield calculation, which inevitably causes inconsistency.

To avoid this problem, Malhotra et. al. (2005) proposed, for XQuery and XPath, deriving two new (totally ordered) subtypes, “yearMonthDuration” (in the year-month system) and “dayTimeDuration” (in the day-hour-minute-second system), from the (partially ordered) datatype “duration”. But in this case, temporal arithmetic can only be carried out in the two separate systems, and months and days cannot be computed together.

Both the “history-independent intuition” and the approach with a fixed number of days for each month can be encoded straightforwardly with a few simple axioms or rules. However, to our knowledge, there has been no serious effort for doing temporal reasoning based on the "history-dependent intuition", which may be the most popular intuition, but the most difficult to model.

In this chapter, we present our work on creating a complete set of rules that can handle temporal arithmetic mixing months and days for the "history-dependent intuition" (including the motivating example), proposing a notion (called "days lost" DL) to concisely keep track of the history of the computation and explain possible inconsistencies in terms of different desired arithmetic properties.

In Section 5.1 we describe these desired properties for temporal arithmetic computation. The notion of "days lost" is introduced in Section 5.2, and in Section 5.3 we present the temporal arithmetic rules with examples.

Since the problem arises only when months and days are mixed, in this paper we only consider dates with months and days, and all other fields of a date/time (e.g., year, hour) will be omitted in the examples and rules. First, we introduce some notation and assumptions for later examples and rules.

Notation:

- $(m, d)^{DL}$ denotes a *date* with a month "m", a day "d", and days lost "DL" (see Section 5.2 for the details on "days lost"). For example, $(2, 28)^{3}$ means February 28th with 3 days lost. DL is omitted when DL = 0.

- [m, d] denotes a *duration* with "m" months and "d" days. For example, [3, 2] means 3 months and 2 days.

- #(m) denotes the number of days in a month "m". For example #(3) = 31, since there are 31 days in March.

Assumptions:

- The year is 2007, so the number of days in February is 28.

- Dates and durations are in canonical forms, i.e.,

  For dates: $1 \leq m \leq 12$

  $1 \leq d \leq$ last day of the current month, i.e., #(m)

  For durations: $1 \leq m < 12$

  $1 \leq d <$ last day of the destination month

5.1 Desired Properties for Temporal Arithmetic Computation

Before creating the rules for temporal arithmetic computation satisfying the "history-dependent intuition", a list of desired properties which we hope can hold during the computation are generated first, and these properties were also used to guide the creation of the rules. The desired properties are as follows, with the most desired ones at the top.

(1) Motivating example for the "history-dependent intuition":

$$(1, 31) + [1, 0] + [1, 0] = (2, 28) + [1, 0] = (3, 31)$$

(2) Addition-Simplicity property:

$$(m_1, d_1) + [m_2, d_2] = (m_1+m_2, d_1+d_2), \text{ if } d_1+d_2 \leq \#(m_1+m_2)$$

(3) Subtraction-Simplicity property:

$$(m_1, d_1) – [m_2, d_2] = (m_1-m_2, d_1-d_2), \text{ if } m_1 > m_2, d_1 > d_2, d_1-d_2 \leq \#(m_1-m_2).$$

(4) Subtraction property:

$$(m_1, d_1) + [m_2, d_2] = (m_3, d_3) \iff (m_3, d_3) - [m_2, d_2] = (m_1, d_1)$$

$$\iff (m_3, d_3) - (m_1, d_1) = [m_2, d_2]$$

(5) Commutativity property:

$$(m_1, d_1) + [m_2, d_2] + [m_3, d_3] = (m_1, d_1) + [m_3, d_3] + [m_2, d_2]$$

(6) Associativity property:

$$\{(m_1, d_1) + [m_2, d_2]\} + [m_3, d_3] = (m_1, d_1) + \{[m_3, d_3] + [m_2, d_2]\}$$

Property (1) is actually an instance of associativity, because

$$\{(1, 31) + [1, 0]\} + [1, 0] = (2, 28) + [1, 0] = (3, 31)$$

$$(1, 31) + \{[1, 0] + [1, 0]\} = (1, 31) + [2, 0] = (3, 31)$$

5.2 Meaning of “Day Lost (DL)”

In order to keep track of the history of the temporal arithmetic computation for the “history-dependent intuition”, we introduce a notion, called “days lost (DL)”. Its meaning is as follows:

(1) If the day of the month is the end of the month (e.g., $(2, 28)^3$), the days lost (“DL”) means the number of days lost in the current month. For example, adding 1 month to January 31st results in February 28th, and 3 days are “lost” in the computation. Thus we use $(2, 28)^3$ to “remember” that 3 days were lost in February.

(2) If the day of the month is not the end of the month (e.g., $(3, 2)^3$), the days lost (“DL”) is just a record of the number of days lost in the past. For example, adding 1 month and 2 days to January 31st results in March 2nd, where 3 days were “lost” in February.

In fact, “days lost (DL)” not only is used to keep track of the history of the computation, but also plays a crucial role in explaining the inconsistency of the computation results with respect to different desired properties (e.g., commutativity and associativity). For example, it can be used to explain the inconsistency described in the introduction regarding the temporal arithmetic algorithm in XML Schema (Biron and Malhotra, 2004):

(2000-03-30 + P1D) + P1M = 2000-03-31 + P1M = 2000-04-30

(2000-03-30 + P1M) + P1D = 2000-04-30 + P1D = 2000-05-01

The different results are actually due to the "one day lost" in the first computation when adding 1 month (P1M) to 2000-03-31, since there are only 30 days (not 31 days) in April. We will show in the next section how our rules can be used to deal with such inconsistency.

It can be argued that February 28th is ambiguous, since it can mean either the 28th day of February, or the last day of February. The notion of "day lost (DL)" can actually be used to capture all the different ambiguities. For example, $(2, 28)^3$ means the last day of February, whereas (2, 28) means the 28th day of February.

5.3 Temporal Arithmetic Rules

In this section, we describe a complete set of rules we created for temporal arithmetic satisfying the "history-dependent intuition", including adding durations to dates, subtracting durations from dates, and computing duration between two dates. Each rule is presented with one representative example, and additional examples are shown to demonstrate how to use multiple rules together in the computation. All the rules can be straightforwardly translated into FOL axioms, so that they can be incorporated into OWL-Time to give them access to the full ontology of time for temporal reasoning. One FOL translation of a temporal arithmetic rule is shown in Section 5.4.

5.3.1 Adding Durations to Dates

There are three sets of rules for adding durations to dates: *add-decompose*, *add-month*, and *add-day*. In the following table, the rules are shown in the left column with index numbers (e.g., (1.1)), and one representative example is shown for each line of the rules in the right column for the demonstration and justification purpose.

Temporal Arithmetic Rule	**Example**
(1) Add-decompose:	
$(m_1, d_1)^{DL} + [m_2, d_2]$	$(2, 28)^2 + [1, 2]$
(1.1) $= (m_1, d_1)^{DL} + [m_2, 0] + [0, d_2]$	$= (2, 28)^2 + [1, 0] + [0, 2]$
(2) Add-month: // $N = \#(m_1+m_2)$	
$(m_1, d_1)^{DL} + [m_2, 0]$	
$= \text{if } (d_1 = \#(m_1))$	
(2.1) (m_1+m_2, d_1+DL), $d_1+DL \le N$	$(2, 28)^2 + [2, 0] = (4, 30)$
(2.2) $(m_1+m_2, N)^{d_1+DL-N}$, $d_1+DL > N$	$(3, 31) + [1, 0] = (4, 30)^1$
else (i.e., $d_1 < \#(m_1)$)	
(2.3) (m_1+m_2, d_1), $d_1 \le N$	$(3, 2)^3 + [1, 0] = (4, 2)$
(2.4) $(m_1+m_2, N)^{d_1-N}$, $d_1 > N$	$(1, 30) + [1, 0] = (2, 28)^2$
(3) Add-day: // $N = \#(m_1)$	
$(m_1, d_1)^{DL} + [0, d_2]$	
$= \text{if } (d_1 = \#(m_1))$	
(3.1) $(m_1+1, d_2)^{DL}$	$(2, 28)^1 + [0, 1] = (3, 1)^1$
else (i.e., $d_1 < \#(m_1)$)	
(3.2) $(m_1, d_1+d_2)^{DL}$, $d_1+d_2 < N$	$(3, 2)^3 + [0, 20] = (3, 22)^3$
(3.3) (m_1, d_1+d_2), $d_1+d_2 = N$	$(3, 2)^3 + [0, 29] = (3, 31)$
(3.4) (m_1+1, d_1+d_2-N), $d_1+d_2 > N$	$(2, 20) + [0, 10] = (3, 2)$

Table 5.1 Rules for Adding Durations to Dates

For any given duration, before it adds to a date, the *add-decompose* rule is applied first to separate the months and days by decomposing the duration into two sub-durations with only months and days respectively, and then months are first added to the date using *add-month* rules, then the days are added using *add-day* rules.

Two more examples of adding durations to dates are also shown below. The first one is from the motivating example, and the second one demonstrates the complete procedure (3 steps) for adding durations to dates, when durations have both months and days (the rule applied for each step of the computation is also shown with a corresponding rule index number):

Ex.1. (1, 31) + [1, 0]
= $(2, 28)^3$ // add-month (2.2)

$(2, 28)^3$ + [1, 0]
= (3, 31) // add-month (2.1)

Ex.2. (1, 29) + [1, 2]
= (1, 29) + [1, 0] + [0, 2] // add-decompose (1.1)
= $(2, 28)^1$ + [0, 2] // add-month (2.4)
= $(3, 2)^1$ // add-day (3.1)

5.3.2 Subtracting Durations from Dates

There are three sets of rules for subtracting durations from dates: *sub-decompose*, *sub-day*, and *sub-month*:

Temporal Arithmetic Rule	Example
(1) Sub-decompose	
$(m_1, d_1)^{DL}$ - $[m_2, d_2]$	$(2, 28)^2$ - [1, 2]
(1.1) = $(m_1, d_1)^{DL}$ - $[0, d_2]$ - $[m_2, 0]$	= $(2, 28)^2$ - [0, 2] - [1, 0]
(2) Sub-day	
$(m1, d_1)^{DL}$ - $[0, d_2]$	
(2.1) = $(m_1, d_1-d_2)^{DL}$, $d_1 > d_2$	$(4, 30)^1 – [0, 15] = (4, 15)^1$
(2.2) $(m_1-1, d_1+\#(m_1-1)-d_2)^{DL}$, $d_1 \leq d_2$	$(3, 2)^3 – [0, 10] = (2, 20)^3$
(3) Sub-month // $N = \#(m_1-m_2)$	
$(m_1, d_1)^{DL}$ - $[m_2, 0]$	
= if $(d_1 = \#(m_1))$	
(3.1) (m_1-m_2, d_1+DL), $d_1+DL \leq N$	$(2, 28)^2 – [1, 0] = (1, 30)$
(3.2) $(m_1-m_2, N)^{d_1+DL-N}$, $d_1+ DL > N$	$(4, 30)^1 – [2, 0] = (2, 28)^3$
else (i.e., $d_1 < \#(m_1)$)	
(3.3) $(m_1-m_2, N)^{d_1-N}$, $d_1 > N$	$(3, 30)^2 – [1, 0] = (2, 28)^2$
(3.4) $(m_1-m_2, d1)$, $d_1 \leq N$	$(5, 16) – [1, 0] = (4, 16)$

Table 5.2 Rules for Subtracting Durations from Dates

For any given duration, before it subtracts from a date, the *sub-decompose* rule is applied first to separate the months and days by decomposing the duration into two sub-durations with only days and months respectively, and then days are subtracted first from the date using *sub-day* rules, then the months are subtracted using *sub-month* rules.

Two more examples of subtracting durations from dates are also shown below. The first one is from the motivating example, and the second one demonstrates the complete procedure (3 steps) for subtracting durations from dates, when durations have both months and days:

Ex.3. $(3, 31) - [1, 0]$
$= (2, 28)^3$ // sub-month (3.2)

$(2, 28)^3 - [1, 0]$
$= (1, 31)$ // sub-month (3.1)

Ex.1. and Ex.3. show that the temporal arithmetic rules work for the *motivating example* with the "history-dependent intuition" (desired property 1), and the first part of the "*subtraction property*" (desired property 2) also holds for the motivating example, because

$(1, 31) + [1, 0] = (2, 28)^3 \Leftrightarrow (2, 28)^3 - [1, 0] = (1, 31)$

$(2, 28)^3 + [1, 0] = (3, 31) \Leftrightarrow (3, 31) - [1, 0] = (2, 28)^3$

$(1, 31) + [1, 0] + [1, 0] = (3, 31) \Leftrightarrow (3, 31) - [1, 0] - [1, 0] = (1, 31)$

Ex.4. $(3, 2)^1 - [1, 2]$
$= (3, 2)^1 - [0, 2] - [1, 0]$ // sub-decompose (1.1)
$= (2, 28)^1 - [1, 0]$ // sub-day (2.2)
$= (1, 29)$ // sub-month (3.1)

Ex.2. and Ex.4. also show that the first part of the "*subtraction property*" (desired property 2) holds for this example whose duration has both months and days, because

$$(1, 29) + [1, 2] = (3, 2)^{1} \Leftrightarrow (3, 2)^{1} - [1, 2] = (1, 29)$$

5.3.3 Computing Duration between Two Dates

There is only one set of rules for computing duration between two dates:

Temporal Arithmetic Rule	**Example**
(1) Sub-between-dates	
$(m_2, d_2)^{DL_2} - (m_1, d_1)^{DL_1}$	
$= \text{if } (d_1 = \#(m_1))$	
(1.1) $[m_2-m_1, d_2-(d_1+DL_1)]$, $d_2 >= d_1+DL_1$	$(5, 31) - (2, 28)^2 = [3, 1]$
(1.2) $[m_2-m_1-1, (d_2+DL_2)+\#(m_2-1)-(d_1+DL_1)]$,	$(3, 2)^3 - (1, 31) = [1, 2]$
$d_2 < d_1+DL_1$, $d_2 < \#(m_2)$	
(1.3) $[m_2-m_1, 0]$, $d_2 < d_1+DL_1$, $d_2 = \#(m_2)$	$(4, 30)^1 - (2, 28)^3 = [2, 0]$
else (i.e., $d_1 < \#(m_1)$)	
(1.4) $[m_2-m_1, d2-d_1]$, $d_2 >= d_1$	$(3, 20)^2 - (2, 10) = [1, 10]$
(1.5) $[m_2-m_1-1, d2]$, $d_2 < d_1$, $d_2 < \#(m_2)$, $\#(m_2-1) \le d_1$	$(3, 2) - (1, 28) = [1, 2]$
(1.6) $[m_2-m_1-1, d_2+\#(m_2-1)-d_1]$,	$(3, 2) - (1, 20) = [1, 10]$
$d_2 < d_1$, $d_2 < \#(m_2)$, $\#(m_2-1) > d_1$	
(1.7) $[m_2-m_1, 0]$, $d_2 < d_1$, $d_2 = \#(m_2)$	$(2, 28)^1 - (1, 29) = [1, 0]$

Table 5.3 Rules for Computing Duration between Dates

Two more examples of computing duration between two dates are also shown below. The first one is from the motivating example, and the second one demonstrates the procedure for computing durations between two dates, when durations have both months and days:

Ex.5. $(3, 31) - (2, 28)^3 = [1, 0]$ // sub-between-dates (1.1)

$(2, 28)^3 - (1, 31) = [1, 0]$ // sub-between-dates (1.3)

$(3, 31) - (1, 31) = [2, 0]$ // sub-between-dates (1.1)

Ex.1., Ex.3., and Ex.5. show the complete "*subtraction property*" (desired property 2) holds for the motivating example, because

$$(1, 31) + [1, 0] = (2, 28)^3 \iff (2, 28)^3 - [1, 0] = (1, 31)$$
$$\iff (2, 28)^3 - (1, 31) = [1, 0]$$
$$(2, 28)^3 + [1, 0] = (3, 31) \iff (3, 31) - [1, 0] = (2, 28)^3$$
$$\iff (3, 31) - (2, 28)^3 = [1, 0]$$

They also show the "*associativity property*" holds for the motivating example, because

$$\{(1, 31) + [1, 0]\} + [1, 0] = (3, 31) \iff (1, 31) + \{[1, 0] + [1, 0]\}$$
$$= (1, 31) + [2, 0] = (3, 31)$$

Ex.6. $(3, 2)^1 - (1, 29) = [1, 2]$ // sub-between-dates (1.5)

Ex.2., Ex.4., and Ex.6. also show that the complete "*subtraction property*" (desired property 2) holds for this example, because

$$(1, 29) + [1, 2] = (3, 2)^1 \Leftrightarrow (3, 2)^1 - [1, 2] = (1, 29)$$
$$\Leftrightarrow (3, 2)^1 - (1, 29) = [1, 2]$$

Ex.7. $(1, 31) + [1, 2] + [2, 0]$

$= (1, 31) + [1, 0] + [0, 2] + [2, 0]$ // add-decompose (1.1)

$= (2, 28)^3 + [0, 2] + [2, 0]$ // add-month (2.2)

$= (3, 2)^3 + [2, 0]$ // add-day (3.1)

$= (5, 2)$ // add-month (3.4)

$(1, 31) + [2, 0] + [1, 2]$

$= (3, 31) + [1, 2]$ // add-month (2.1)

$= (3, 31) + [1, 0] + [0, 2]$ // add-decompose (1.1)

$= (4, 30)^1 + [0, 2]$ // add-month (2.2)

$= (5, 2)^1$ // add-day (3.1)

Ex.7. shows that the "*commutativity property*" hold for this example, if the supplemental information (days lost "DL") is ignored.

Ex.8. In the introduction, we showed the following example from XML Schema (Biron and Malhotra, 2004) that demonstrates the case where the properties of commutativity and associativity cannot hold using their temporal arithmetic algorithm:

(2000-03-30 + P1D) + P1M = 2000-03-31 + P1M = 2000-04-30

(2000-03-30 + P1M) + P1D = 2000-04-30 + P1D = 2000-05-01

Can our rules handle this problem? The computation is as follows.

```
(3, 30) + [0, 1] + [1, 0]
= (3, 31) + [1, 0]                    // add-day (3.3)
= (4, 30)^1                           // add-month (2.2)

(3, 30) + [1, 0] + [0, 1]
= (4, 30) + [0, 1]                    // add-month (2.3)
= (5, 1)                              // add-day (3.1)
```

The rules not only compute the results exactly based on the "history-dependent intuition", but also include supplemental information (days lost "DL") for explaining the reason why the commutativity property doesn't hold: there was "one day lost" in the first computation when one month is added to March 31.

In fact, the commutativity property would hold, if we specify that $(4, 30)^1$ "equals" to (5, 1). More generally, the commutativity and associativity properties hold with our temporal arithmetic rules, if we define the month-day equality "more softly" and take into account the effect of "days lost" (assume overflow is properly handled):

$$(m_1, d_1)^{DL_1} = (m_2, d_2)^{DL_2}, \text{ if } m_1 = m_2 \text{ AND } (d_1 = d_2 \textit{ OR } d_1 + DL_1 = d_2 + DL_2)$$

5.4 Translating Rules to FOL Axioms

All the above temporal arithmetic rules can be straightforwardly translated into FOL axioms in OWL-Time to give them access to the full ontology of time for temporal reasoning. For example, the rule "add-month (2.1)"

$$(m_1, d_1)^{DL} + [m_2, 0] = (m_1+m_2, d_1+DL),$$

$$\text{if } (d_1 = \#(m_1)) \text{ AND } d_1+DL \leq \#(m1+m2)$$

can be translated as:

$$\begin{aligned}
&(\forall\, T, t_1, t_3, m_1, m_2, m_3, d_1, d_3, DL, DL_3, n_1, n_3)\ \mathit{dateOf}(t_1, m_1, d_1, DL) \\
&\quad \wedge\ \mathit{durationOf}(T, m_2, 0)\ \wedge\ \mathit{begins}(t_1, T)\ \wedge\ \mathit{ends}(t_3, T) \\
&\quad \wedge\ \mathit{dateOf}(t_3, m_3, d_3, DL_3)\ \wedge\ \mathit{Hath}(n_1, {*Day*}, m_1) \\
&\quad \wedge\ \mathit{Hath}(n_3, {*Day*}, m_3)\ \wedge\ d_1 = n_1\ \wedge\ d_1+DL \leq n_3 \\
&\qquad \supset m_3 = m_1+m_2\ \wedge\ d_3 = d_1+DL\ \wedge\ DL_3 = 0
\end{aligned}$$

Chapter 6

Vague Event Durations

Consider the sentence from a news article:

George W. Bush <u>*met*</u> *with Vladimir Putin in Moscow.*

How long did the meeting last? Our first inclination is to say we have no idea. But in fact we do have some idea. We know the meeting lasted more than ten seconds and less than one year. As we guess narrower and narrower bounds, our chances of being correct go down. Just how accurately can we make duration judgments like this? How much agreement can we expect among people? Will it be possible to extract this kind of information from text automatically?

The uncertainty of temporal durations has been recognized as one of the most significant issues for temporal reasoning (Allen and Ferguson, 1994; Chittaro and Montanari, 2000). For example, we have to know how long a battery remains charged to decide when to replace it or to predict the effects of actions which refer to the battery charge as a precondition (Chittaro and Montanari, 2000).

As part of our commonsense knowledge, we can estimate roughly how long events of different types last and roughly how long situations of various sorts persist. For example, we know government policies typically last somewhere between one and ten years, while weather conditions fairly reliably persist between three hours and one day.

There is much temporal information in text that has hitherto been largely unexploited, implicitly encoded in the descriptions of events and relying on our knowledge of the range of usual durations of types of events. This chapter describes our work of an exploration into how this information can be captured automatically.

Missing durations is one of the most common sources of incomplete information for temporal reasoning in natural language applications, because very often explicit duration information (e.g., "a five-day meeting", "I have lived here for three years") is missing in natural language texts. Thus, this work can be very important in applications in which the time course of events is to be extracted from text. For example, whether two events overlap or are in sequence often depends very much on their durations. If a war started yesterday, we can be pretty sure it is still going on today. If a hurricane started last year, we can be sure it is over by now.

Although there has been much work on formalizing temporal information (Allen 1984; Moens and Steedman, 1988; Zhou and Fikes, 2002; Han and Lavie, 2004) and on temporal anchoring and event ordering in text (Hitzeman et al., 1995; Mani and Wilson, 2000; Filatova and Hovy, 2001; Boguraev and Ando, 2005; Mani et al., 2006), to our knowledge, there has been no serious published empirical effort to model and learn the vague and implicit duration information in natural language, such as the typical durations of events, and to perform reasoning over this information. (Cyc apparently has some fuzzy duration information, although it is not generally available; Rieger (1974) discusses the issue for less than a page; there has been work in fuzzy logic on representing and reasoning with imprecise durations (Godo and Vila, 1995; Fortemps, 1997), but these make no attempt to collect human judgments on such durations or to extract them automatically from text.)

Our goal is to be able to extract this event duration information from text automatically, and to that end we first annotated the events in news articles with bounds on their durations. For reliability, narrow bounds of duration are needed if we want to infer whether event *e* is happening at time *t*, while wide bounds of duration are needed to infer whether event *e* is *not* happening at time *t*.

The corpus that we have annotated currently contains all the 48 non-Wall-Street-Journal (non-WSJ) news articles (2132 event instances), as well as 10 WSJ articles (156 event instances), from the TimeBank corpus annotated in TimeML. The non-WSJ articles (mainly political and disaster news) include both print and broadcast news that are from a variety of news sources, such as ABC, AP, CNN and VOA. All the annotated data has already been integrated into the TimeBank corpus[29].

The events annotated in the TimeBank corpus are typically verbs, although event nominals, such as "crash" in "...killed by the crash", will also be annotated as events; they also cover a subset of the states in a document, including those that are either transient or explicitly marked as participating in a temporal relation.

This chapter is organized as follows. In Section 6.1 we first describe our annotation guidelines, including the annotation strategy and assumptions, and the

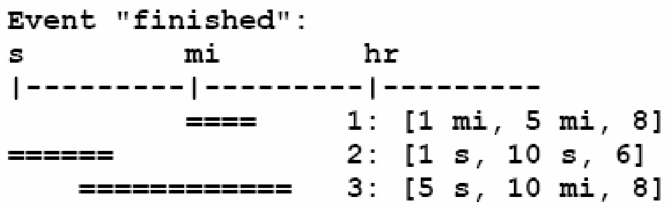
[29] The annotated data is available at http://www.isi.edu/~pan/EventDuration/annotations/

representative event classes we have categorized to minimize discrepant judgments between annotators. The method for measuring inter-annotator agreement when the judgments are intervals on a scale will be described in Section 6.2. In Section 6.3 we will show that machine learning techniques applied to the annotated data considerably outperform a baseline and approach human performance. We will discuss how to integrate our event duration annotations to TimeML in Section 4.

6.1 Annotation Guidelines and Events Classes

Every event to be annotated was already identified in the TimeBank corpus. Annotators are asked to provide lower and upper bounds on the duration of the event, and a judgment of level of confidence in those estimates on a scale from one to ten. An interface was built to facilitate the annotation. Graphical output is displayed to enable us to visualize quickly the level of agreement among different annotators for each event. For example, here is the output of the annotations (3 annotators) for the "finished" event (underlined) in the sentence

> *After the victim, Linda Sanders, 35, had finished her cleaning and was waiting for her clothes to dry, ...*



```
Event "finished":
s         mi        hr
|---------|---------|---------
          ====     1: [1 mi, 5 mi, 8]
======             2: [1 s, 10 s, 6]
    ============   3: [5 s, 10 mi, 8]
```

This graph shows that the first annotator believes that the event lasts for minutes whereas the second annotator believes it could only last for several seconds. The third annotates the event to range from a few seconds to a few minutes. The confidence level of the annotators is generally subjective but as all three are higher than 5, it shows reasonable confidence. A logarithmic scale is used for the output (see Section 6.2.1 for details).

6.1.1 Annotation Instructions

Annotators are asked to make their judgments as intended readers of the article, using whatever world knowledge is relevant to an understanding of the article. They are asked to identify upper and lower bounds that would include 80% of the possible cases. For example, rainstorms of 10 seconds or of 40 days and 40 nights might occur, but they are clearly anomalous and should be excluded. There are two strategies for considering the range of possibilities:

1. Pick the most probable scenario, and annotate its upper and lower bounds.
2. Pick the set of probable scenarios, and annotate the bounds of their upper and lower bounds.

We deem the second to be the preferred strategy.

The judgments are to be made in context. First of all, information in the syntactic environment needs to be considered before annotating. For example, there is a difference in the duration of the watching events in the phrases "*watch a movie*" and "*watch a bird fly*".

Moreover, the events need to be annotated in light of the information provided by the entire article. This means annotators should read the entire article before starting to annotate. One may learn in the last paragraph, for example, that the demonstration event mentioned in the first paragraph lasted for three days, and that information should be used for annotation.

However, they should not use knowledge of the future when annotating a historical article. For example, an article from the fall of 1990 may talk about the coming war against Iraq. Today we know exactly how long that lasted. But annotators are asked to try to put themselves in the shoes of the 1990 readers of that article, and make their judgments accordingly. This is because we want people's estimates of typical durations of events, rather than the exact durations.

Annotation can be made easier and more consistent if *coreferential* and *near-coreferential* descriptions of events are identified initially. Annotators are asked to give the same duration ranges for such cases. For example, in the sentence "*during the <u>demonstration</u>, people <u>chanted</u> antigovernment slogans*", annotators should give the same durations for the "demonstration" and "chanted" events.

6.1.2 Analysis

When the articles were completely annotated by the three annotators, the results were analyzed and the differences were reconciled. Differences in annotation could be due to the differences in interpretations of the event; however, we found that the vast majority of radically different judgments can be categorized into a relatively small number of classes. Some of these correspond to aspectual features of events, which have been intensively investigated (e.g., Vendler, 1967; Dowty, 1979; Moens and Steedman, 1988; Passonneau, 1988; Giorgi and Pianesi, 1997; Madden and Zwaan, 2003; Smith, 2005).

We then developed guidelines to make annotators aware of these cases and to guide them in making the judgments (see the next section). There is a residual of gross discrepancies in annotators' judgments that result from differences of opinion, for example, about how long a government policy is typically in effect. But the number of these discrepancies was surprisingly small.

These guidelines were then used to annotate a test set. It was shown that the agreement in the test set was greater than the agreement obtained when annotations were performed without the guidelines. (See Section 6.2.3 for the experimental results.)

6.1.3 Event Classes

Action vs. State: Actions involve change, such as those described by words like "speaking", "gave", and "skyrocketed". States involve things staying the same, such as being dead, being dry, and being at peace. When we have an event in the passive tense, sometimes there is an ambiguity about whether the event is a state or an action. For example, in

Three people were <u>injured</u> in the attack.

does the word “injured” describe an action or a state? This matters because they will have different durations. The state begins with the action and lasts until the victim is healed. There are some general diagnostic tests to distinguish them (Vendler, 1967; Dowty, 1979); for example, action verbs are fine in the progressive form but stative verbs are usually odd when used in the progressive form. Another test can be applied to this specific case: Imagine someone says the sentence after the action had ended but the state was still persisting. Would they use the past or present tense? In the “injured” example, it is clear we would say “Three people *were* injured in the attack”, whereas we would say “Three people *are* injured *from* the attack.” Our annotation interface handles events of this type by allowing the annotators to specify which interpretation they are giving. If the annotator feels it’s too ambiguous to distinguish, annotations can be given for both interpretations.

Although “retired” usually indicates a state, in the following example

Farkas was ordered home and <u>retired</u>.

it looks more like an action by his company of retiring Farkas.

Aspectual Events: Some events are aspects of larger events, such as their start or finish. Although they may seem instantaneous, we believe they should be considered to happen across some interval, i.e., the first or last *sub-event* of the larger event. For example, in

After the victim, Linda Sanders, 35, had <u>finished</u> her cleaning and was waiting for her clothes to dry,…

the "finished" event should be considered as the last sub-event of the larger event (the "cleaning" event), since it actually involves opening the door of the washer, taking out the clothes, closing the door, and so on. All this takes time. This interpretation will also give us more information on typical durations than simply assuming such events are instantaneous.

In the following example

General Abacha's supporters <u>began</u> a two day rally in the capital.

the gathering of people marks the "beginning" of the rally, and it generally takes time for a crowd of people to get together to start a rally.

Reporting Events: These are everywhere in the news. They can be direct quotes, taking exactly as long as the sentence takes to read, or they can be summarizations of long press conferences. We need to distinguish different cases:

Quoted Report: This is when the reported content is quoted. The duration of the event should be the actual duration of the utterance of the quoted content. The time duration can be easily verified by saying the sentence out loud and timing it. For example, in

"It looks as though they panicked," a detective <u>said</u> of the robbers.

the saying probably took between 1 and 3 seconds; it's very unlikely it took more than 10 seconds.

Unquoted Report: When the reporting description occurs without quotes, the report could be as short as the duration of the actual utterance of the reported content (lower bound), and as long as the duration of a briefing or press conference (upper bound).

If the sentence is very short, then it's likely that it is one complete sentence from the speaker's remarks, and a short duration should be given; if it is a long, complex sentence, then it's more likely to be a summary of a long discussion or press conference, and a longer duration should be given. For example, consider

The police <u>said</u> it did not appear that anyone else was injured.

A Brooklyn woman who was watching her clothes dry in a laundromat was killed Thursday evening when two would-be robbers emptied their pistols into the store, the police <u>said</u>.

If the first sentence were quoted text, it would be very much the same. Hence the duration of the "said" event should be short. In the second sentence everything that the spokesperson (here the police) has said is compiled into a single sentence by the reporter, and it is unlikely that the spokesperson said only a single sentence with all this information. Thus, it is reasonable to give longer duration to this "said" event.

Multiple Events: Many occurrences of verbs and other event descriptors refer to multiple events, especially, but not exclusively, if the subject or object of the verb is plural. For example, in

Iraq has <u>destroyed</u> its long-range missiles.

both single (i.e., destroyed one missile) and aggregate (i.e., destroyed all missiles) events happened. This was a significant source in disagreements in our first round of annotation. Since both judgments provide useful information, our current annotation interface allows the annotator to specify the event as multiple, and give durations for both the single and aggregate events.

In the following example

Seventy-five million copies of the rifle have been *built* *since it entered* *production* *in February 1947.*

"built" and "production" are both multiple events. The annotators were asked to give durations for both the single (i.e., built/produce one rifle) and aggregate (i.e., built/produce seventy-five million copies of the rifle) events.

Events Involving Negation: Negated events didn't happen, so it may seem strange to specify their duration. But whenever negation is used, there is a certain class of events whose occurrence is being denied. Annotators should consider this class, and make a judgment about the likely duration of the events in it. In addition, there is the interval during which the nonoccurrence of the events holds. For example, in

He was willing to withdraw troops in exchange for guarantees that Israel would not be *attacked*.

there is the typical amount of time of "being attacked", i.e., the duration of a single attack, and a longer period of time of "*not* being attacked". The first is probably from seconds to minutes and the second from months to years. Similarly to multiple events, annotators are asked to give durations for both the event negated and the negation of that event.

Appearance Events. Verbs like "seem" and "appear" usually indicate appearance events. The duration of this kind of events depends on the duration of the validity or availability of the evidence that causes one to have some impression at the time. Such an event begins when enough evidence has accumulated for one to make that guess or judgment, and it ends when either the evidence is contradicted or certainty is achieved. For example, in

> *It appears that the destruction of this city in 2700 B.C. was related to the eruption of the volcano.*

the "appears" event lasts from when the archaeologist discovers enough evidence to make the conjecture until the time the conjecture is refuted or confirmed.

Positive Infinite Durations: These are states which continue essentially forever once they begin, for example,

> *He is dead.*

Here the state continues for an infinite amount of time, and we allow this as a possible annotation.

6.2 Inter-Annotator Agreement

Although the graphical output of the annotations enables us to visualize quickly the level of agreement among different annotators for each event, a quantitative measurement of the agreement is needed.

The kappa statistic (Krippendorff, 1980; Siegel and Castellan, 1988; Carletta, 1996; Eugenio and Glass, 2004), which factors out the agreement that is expected by chance, has become the de facto standard to assess inter-annotator agreement. It is computed as follows:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

P(*A*) is the observed agreement among the annotators, and *P*(*E*) is the expected agreement, which is the probability that the annotators agree by chance.

In order to compute the kappa statistic for our task, we have to compute *P*(*A*) and *P*(*E*) first. But those computations are not straightforward.

P(*A*): What should count as agreement among annotators for our task?

P(*E*): What is the probability that the annotators agree by chance for our task?

6.2.1 What Should Count as Agreement?

Determining what should count as agreement is not only important for assessing inter-annotator agreement, but is also crucial for later evaluation of machine learning experiments. For example, for a given event with a known gold standard duration range from 1 hour to 4 hours, if a machine learning program outputs a duration of 3 hours to 5 hours, how should we evaluate this result?

We first need to decide what scale is most appropriate. One possibility is just to convert all the temporal units to seconds. However, this way would not correctly capture our intuitions about the relative relations between duration ranges. For example, the difference between 1 second and 20 seconds is significant; while the difference between 1 year 1 second and 1 year 20 seconds is negligible. Consider the range from 1 year to 5 years and the range from 1 second to 5 seconds. The distance between 1 year and 5 years in seconds would be much larger than that between 1 second and 5 seconds, but intuitively, they represent the same level of uncertainty. In order to handle this problem, we use a logarithmic scale for our data. After first converting from temporal units to seconds, we then take the natural logarithms of these values. This logarithmic scale also conforms to the half orders of magnitude (HOM) (Hobbs, 2000; Hobbs and Kreinovich, 2001) which has been shown to have utility in several very different linguistic contexts.

In the literature on the kappa statistic, most authors address only category data (either in nominal scales or ordinal scales); some can handle more general data, such as data in interval scales or ratio scales (Krippendorff, 1980; Carletta, 1996). However, none of the techniques directly apply to our data, which is range duration from a lower bound to an upper bound.

In fact, what coders annotate for a given event is not just a range, but a *duration distribution* for the event, where the area between the lower bound and the upper bound covers about 80% of the entire distribution area. Since it's natural to assume the most likely duration for such distribution is its mean (average) duration, and the distribution flattens out toward the upper and lower bounds, we use the normal or Gaussian distribution to model our duration distributions.

In order to determine a normal distribution, we need to know two parameters: the mean and the standard deviation. For our duration distributions with given lower and upper bounds, the mean is the average of the bounds. Under the assumption that the area between lower and upper bounds covers 80% of the entire distribution area, the lower and upper bounds are each 1.28 standard deviations from the mean. Then the standard deviation can be computed using either the upper bound (X_{upper}) or the lower bound (X_{lower}) as follows:

$$\sigma = \frac{X_{upper} - \mu}{1.28} = \frac{X_{lower} - \mu}{-1.28}\text{, where } \mu = \frac{X_{upper} + X_{lower}}{2}$$

With this data model, the agreement between two annotations can be defined as the overlapping area between two normal distributions.[30] The agreement among many annotations is the average overlap of all the pairwise overlapping areas. For example, for a given event, suppose the two annotations are:

1) Lower: 10 minutes; upper: 30 minutes

2) Lower: 10 minutes; upper 2 hours

After converting to seconds and to the natural logarithmic scale, they become:

1) Lower: 6.39692; upper: 7.49554

2) Lower: 6.39692; upper: 8.88184

We then compute their means and standard deviations:

1) $\mu_1 = 6.94623$; $\sigma_1 = 0.42861$

2) $\mu_2 = 7.63938$; $\sigma_2 = 0.96945$

[30] This idea is due to Hoa Trang Dang.

The distributions and their overlap are then as in Figure 6.1. The overlap or agreement is 0.508706.

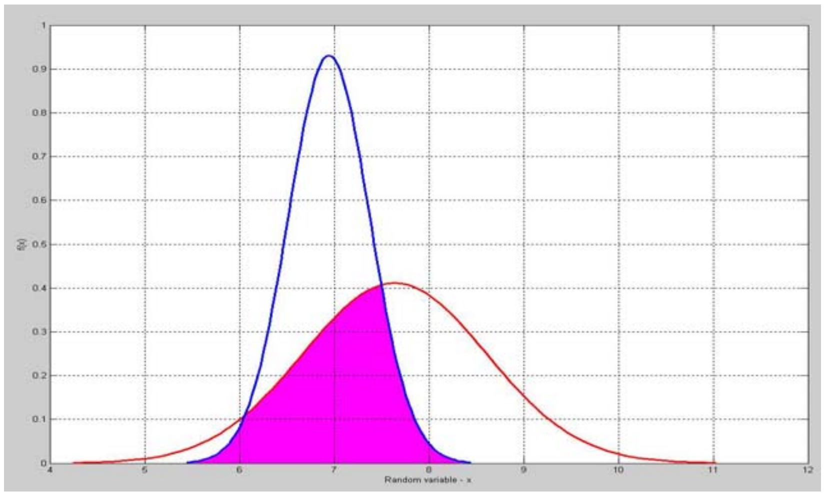


Figure 6.1 Example for Overlap of Judgments

6.2.2 Expected Agreement

What is the probability that the annotators agree by chance for our task? The first quick response to this question may be 0, if we consider all the possible durations from 1 second to 1000 years or even positive infinity.

However, not all the durations are equally possible. As in (Krippendorff, 1980; Siegel and Castellan, 1988), we assume there exists one global distribution for our task (i.e., the duration ranges for all the events), and "chance" annotations would be consistent with this distribution. Thus, the baseline will be an annotator who knows the global distribution and annotates in accordance with it, but does not read the specific article being annotated. Therefore, we must compute the global distribution of the durations, in particular, of their means and their widths. This will be of interest not only in determining expected agreement, but also in terms of what it says about the genre of news articles and about fuzzy judgments in general.

We first compute the distribution of the means of all the annotated durations. Its histogram is shown in Figure 6.2, where the horizontal axis represents the mean values in the natural logarithmic scale and the vertical axis represents the number of annotated durations with that mean.

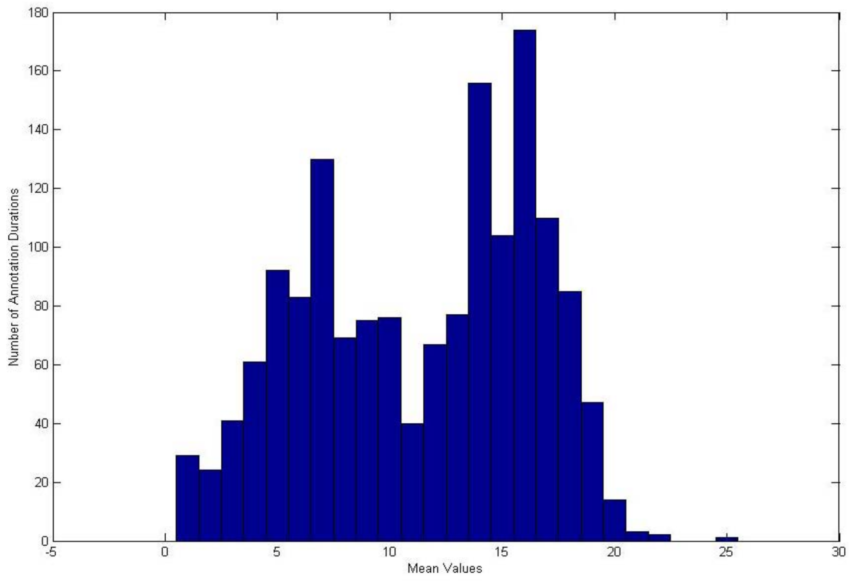


Figure 6.2 Distribution of means of annotation durations

There are two peaks in this distribution. One is from 5 to 7 in the natural logarithmic scale, which corresponds to about 1.5 minutes to 30 minutes. The other is from 14 to 17 in the natural logarithmic scale, which corresponds to about 8 days to 6 months. One could speculate that this bimodal distribution is because daily newspapers report short events that happened the day before and place them in the context of larger trends. The lowest point between the two peaks occurs at 11 which roughly corresponds to one day.

We also compute the distribution of the widths (i.e., $X_{upper} - X_{lower}$) of all the annotated durations, and its histogram is shown in Figure 6.3, where the horizontal axis represents the width in the natural logarithmic scale and the vertical axis represents the number of annotated durations with that width.

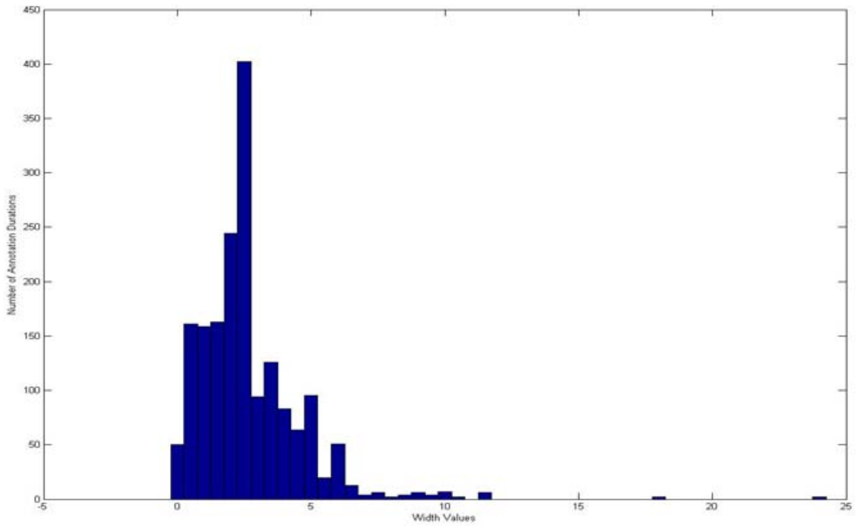


Figure 6.3 Distribution of widths of annotation durations

The peak of this distribution occurs at 2.5 in the natural logarithmic scale. This shows that for annotated durations, the most likely uncertainty factor from a mean (average) duration is 3.5:

$$\frac{X_{upper}}{\mu} = \frac{\mu}{X_{lower}} = e^{1.25} = 3.5 \text{, since}$$

$$\log(X_{upper}) - \log(\mu) = \log(\frac{X_{upper}}{\mu}) = 2.5/2 = 1.25$$

This is the half orders of magnitude factor that Hobbs and Kreinovich (2001) argue gives the optimal granularity; making something 3 to 4 times bigger changes the way we interact with it.

Since the global distribution is determined by the above mean and width distributions, we can then compute the expected agreement, i.e., the probability that the annotators agree by chance, where the chance is actually based on this global distribution. Two approaches were used to approximate this probability, both of which use a normal distribution to approximate the global distribution.

The first approach is to compute a fixed global normal distribution with the mean as the mean of the mean distribution and the standard deviation as the mean standard deviation (this can be straightforwardly computed from the width distribution). We then compute the expected agreement by averaging all the agreement scores (overlaps) between this fixed distribution and each of the annotated duration distributions.

The second approach is to generate 1000 normal distributions whose means are randomly generated from the mean distribution and standard deviations are randomly computed from the width distribution. We then compute the expected agreement by averaging all the agreement scores (overlaps) between these 1000 random distributions.

In a sense, both of these capture the way an annotator might annotate if he or she did not read the article but only guessed on the basis of the global distribution. As it turns out, the results of the two approaches of computing the expected agreement are very close; they differ by less than 0.01: $P(E)_1 = 0.1439$, $P(E)_2 = 0.1530$. We will use the results of the second approach as the baseline in the next section.

6.2.3 Inter-Annotator Agreement Experiments

In order to see how effective our guidelines are, we conducted experiments to compare the inter-annotator agreement *before* and *after* annotators read the guidelines.

The data for the evaluation was split into two sets. The first set contained 13 articles (521 events, 1563 annotated durations) which were all political and disaster news stories from ABC, APW, CNN, PRI, and VOA. The annotators annotated independently *before* reading the guidelines. The annotators were only given short instructions on what to annotate and one sample article with annotations. The second set (test set) contained 5 articles (125 events, 375 annotated durations) which were also political and disaster news stories from the same news sources. The annotators annotated independently *after* reading the guidelines.

The comparison is shown in Figure 6.4. Agreement is measured by the area of overlap in two distributions and is thus a number between 0 and 1. The graphs show the answer to the question "If we set the threshold for agreement at x, counting everything above x as agreement, what is the percentage y of inter-annotator agreement?" The horizontal axis represents the overlap thresholds, and the vertical axis represents the agreement percentage, i.e., the percentage of annotated durations that agree for given overlap thresholds. There are three lines in the graph. The top one with circles represents the after-guidelines agreement; the middle one with triangles represents the before-guidelines agreement; and the lowest one with squares represents the expected (baseline) agreement. This graph shows that, for example, if we define agreement to be a 10% overlap or better (an overlap threshold of 0.1), we can get 0.8 agreement after reading the guidelines, 0.72 agreement before reading the guidelines, and 0.36 expected agreement with only the knowledge of the global distribution. From this graph, we can see that our guidelines are indeed effective in improving the inter-annotator agreement.

Table 6.1 shows more detailed experimental results. For each overlap threshold, it shows the expected (baseline) agreement, the before-guidelines agreement, and the after-guidelines agreement, and also the kappa statistic computed from the after-guidelines agreement ($P(A)$) and the expected (baseline) agreement ($P(E)$).

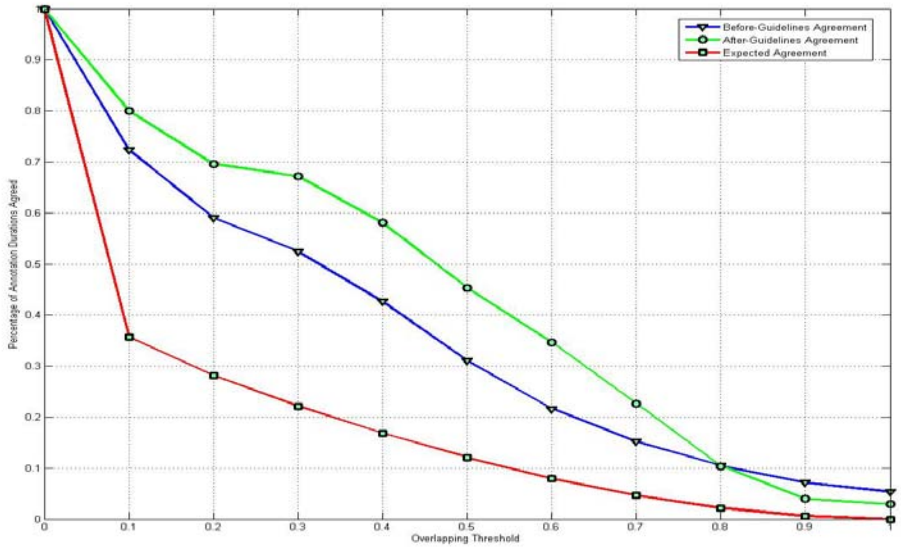


Figure 6.4 Inter-Annotator Agreements Comparison

Overlapping Threshold	**Expected Agreement**	**Before-G. Agreement**	**After-G. Agreement**	**Kappa (After-G. Agreement)**	**Factor**
0.1	0.36	0.72	0.80	0.69	28.50
0.2	0.28	0.59	0.70	0.58	13.46
0.3	0.22	0.52	0.67	0.58	8.17
0.4	0.17	0.43	0.58	0.49	5.47
0.5	0.12	0.31	0.45	0.38	3.86
0.6	0.08	0.22	0.35	0.29	2.86
0.7	0.05	0.15	0.23	0.19	2.23
0.8	0.02	0.10	0.10	0.08	1.65
0.9	0.01	0.07	0.04	0.03	1.28
1.0	0.00	0.05	0.03	0.03	1.00

Table 6.1 Inter-Annotator Agreement with Different Overlapping Thresholds

Moreover, Table 6.1 also shows a factor value that represents how far apart the means of two annotations can be in order to overlap with the given overlap threshold, assuming the width of the two annotations is the mean width, 2.6 on the natural logarithmic scale as computed from the width distribution shown in Figure 6.3.

For example, when the overlap threshold is 0.1, the factor value is 28.5, which means if one annotator guesses the mean duration for a given event is 1 minute, the other annotator will have 0.8 probability of guessing a mean duration from about 2 seconds (1 minutes / 28.5) to 28.5 minutes and their duration distributions will have at least 10% overlap. This also shows that if someone guesses 1 minute for a given event, it's not very likely (with a 0.2 probability) that the event will last more than 28.5 minutes or less than 2 seconds on average. Obviously, as Table 6.1 shows, if we want tighter bounds on the duration, our reliability will go down.

This factor is very useful for temporal reasoning tasks where we need to know whether given events have already ended or not.

6.3 Learning Event Durations

6.3.1 Features

In this section, we describe the lexical, syntactic, and semantic features that we considered in learning event durations.

6.3.1.1 Local Context

For a given event, the local context features include a window of n tokens to its left and n tokens to its right, as well as the event itself, for $n = \{0, 1, 2, 3\}$. The best n determined via cross validation turned out to be 0, i.e., the event itself with no local context. But we also present results for $n = 2$ in Section 6.3.2 to evaluate the utility of local context.

A token can be a word or a punctuation mark. Punctuation marks are not removed, because they can be indicative features for learning event durations. For example, the quotation mark is a good indication of quoted reporting events, and the duration of such events most likely lasts for seconds or minutes, depending on the length of the quoted content. However, there are also cases where quotation marks are used for other purposes, such as emphasis of quoted words and titles of artistic works.

For each token in the local context, including the event itself, three features are included: the original form of the token, its lemma (or root form), and its part-of-speech (POS) tag. The lemma of the token is extracted from parse trees generated by the CONTEX parser (Hermjakob and Mooney, 1997) which includes rich context information in parse trees, and the Brill tagger (Brill, 1992) is used for POS tagging.

The context window doesn't cross the boundaries of sentences. When there are not enough tokens on either side of the event within the window, "NULL" is used for the feature values.

The local context features extracted for the "signed" event in sentence (1) is shown in Table 6.2 (with a window size *n* = 2). The feature vector is [signed, sign, VBD, the, the, DT, plan, plan, NN, Friday, Friday, NNP, on, on, IN].

(1) *The two presidents on Friday <u>signed</u> the plan.*

Features	**Original**	**Lemma**	**POS**
Event	signed	sign	VBD
1token-after	the	the	DT
2token-after	plan	plan	NN
1token-before	Friday	Friday	NNP
2token-before	on	on	IN

Table 6.2 Local context features for the "signed" event in sentence (1) with n = 2

6.3.1.2 Syntactic Relations

The information in the event's syntactic environment is very important in deciding the durations of events. For example, there is a difference in the durations of the "watch" events in the phrases "*<u>watch</u> a movie*" and "*<u>watch</u> a bird fly*".

For a given event, both the head of its subject and the head of its object are extracted from the parse trees generated by the CONTEX parser. Similarly to the local context features, for both the subject head and the object head, their original form, lemma, and POS tags are extracted as features. When there is no subject or object for an event, “NULL” is used for the feature values.

For the “signed” event in sentence (1), the head of its subject is “presidents” and the head of its object is “plan”. The extracted syntactic relation features are shown in Table 6.3, and the feature vector is [presidents, president, NNS, plan, plan, NN].

Features	**Original**	**Lemma**	**POS**
Subject	presidents	president	NNS
Object	plan	plan	NN

Table 6.3 Syntactic relation features for the “signed” event in sentence (1)

6.3.1.3 WordNet Hypernyms

Events with the same hypernyms may have similar durations. For example, events “ask” and “talk” both have a direct WordNet hypernym of “communicate”, and most of the time they do have very similar durations in the corpus.

However, closely related events don’t always have the same direct hypernyms. For example, “see” has a direct hypernym of “perceive”, whereas “observe” needs two steps up through the hypernym hierarchy before reaching “perceive”. Such correlation between events may be lost if only the direct hypernyms of the words are extracted.

It is useful to extract the hypernyms not only for the event itself, but also for the subject and object of the event. For example, events related to a group of people or an organization usually last longer than those involving individuals, and the hypernyms can help distinguish such concepts. For example, “society” has a “group” hypernym (2 steps up in the hierarchy), and “school” has an “organization” hypernym (3 steps up). The direct hypernyms of nouns are not always general enough for such purpose, but a hypernym at too high a level can be too general to be useful. For our learning experiments, we extract the first 3 levels of hypernyms from WordNet.

Hypernyms are only extracted for the events and their subjects and objects, not for the local context words. For each level of hypernyms in the hierarchy, it’s possible to have more than one hypernym, for example, “see” has two direct hypernyms, “perceive” and “comprehend”. For a given word, it may also have more than one sense in WordNet. In such cases, as in (Gildea and Jurafsky, 2002), we only take the first sense of the word and the first hypernym listed for each level of the hierarchy. A word disambiguation module might improve the learning performance. But since the features we need are the hypernyms, not the word sense itself, even if the first word sense is not the correct one, its hypernyms can still be good enough in many cases. For example, in one news article, the word “controller” refers to an air traffic controller, which corresponds to the second sense in WordNet, but its first sense (business controller) has the same hypernym of “person” (3 levels up) as the second sense (direct hypernym). Since we take the first 3 levels of hypernyms, the correct hypernym is still extracted.

When there are less than 3 levels of hypernyms for a given word, its hypernym on the previous level is used. When there is no hypernym for a given word (e.g., "go"), the word itself will be used as its hypernyms. Since WordNet only provides hypernyms for nouns and verbs, "NULL" is used for the feature values for a word that is not a noun or a verb.

For the "signed" event in sentence (1), the extracted WordNet hypernym features for the event ("signed"), its subject ("presidents"), and its object ("plan") are shown in Table 6.4, and the feature vector is [write, communicate, interact, corporate_executive, executive, administrator, idea, content, cognition].

Feature	**1-hyper**	**2-hyper**	**3-hyper**
Event	write	communicate	interact
Subject	corporate executive	executive	administrator
Object	idea	content	cognition

Table 6.4 WordNet hypernym features in sentence (1)

6.3.2 Learning Binary Event Durations

The distribution of the means of the annotated durations in Figure 6.2 is bimodal, dividing the events into those that take less than a day and those that take more than a day. Thus, in our first machine learning experiment, we have tried to learn this *coarse-grained* event duration information as a binary classification task.

6.3.2.1 Inter-Annotator Agreement, Baseline, and Upper Bound

Before evaluating the performance of different learning algorithms, the inter-annotator agreement, the baseline and the upper bound for the learning task are assessed first.

Table 6.5 shows the inter-annotator agreement results among 3 annotators for binary event durations. The experiments were conducted on the same data sets as in Section 6.2.3. Two kappa values are reported with different ways of measuring expected agreement (P(E)), i.e., whether or not the annotators have prior knowledge of the global distribution of the task.

P(A)	**P(E)**	**Kappa**
0.877	0.528	0.740
	0.500	0.755

Table 6.5 Inter-Annotator Agreement for Binary Event Durations

The human agreement before reading the guidelines (0.877) is a good estimate of the *upper bound* performance for this binary classification task. The *baseline* for the learning task is always taking the most probable class. Since 59.0% of the total data is "long" events, the baseline performance is 59.0%**.**

6.3.2.2 Data

The original annotated data can be straightforwardly transformed for this binary classification task. For each event annotation, the most likely (mean) duration is calculated first by averaging (the logs of) its lower and upper bound durations. If its most likely (mean) duration is less than a day (about 11.4 in the natural logarithmic scale), it is assigned to the "short" event class, otherwise it is assigned to the "long" event class. (Note that these labels are strictly a convenience and not an analysis of the meanings of "short" and "long".)

We divide the total annotated non-WSJ data (2132 event instances) into two data sets: a training data set with 1705 event instances (about 80% of the total non-WSJ data) and a held-out test data set with 427 event instances (about 20% of the total non-WSJ data). The WSJ data (156 event instances) is kept for further test purposes (see Section 6.3.2.4).

6.3.2.3 Experimental Results (non-WSJ)

Learning Algorithms. Three supervised learning algorithms were evaluated for our binary classification task, namely, Support Vector Machines (SVM) (Vapnik, 1995), Naïve Bayes (NB) (Duda and Hart, 1973), and Decision Trees C4.5 (Quinlan, 1993). The Weka (Witten and Frank, 2005) machine learning package was used for the implementation of these learning algorithms. Linear kernel is used for SVM in our experiments.

Each event instance has a total of 18 feature values, as described in Section 6.3.1, for the event only condition, and 30 feature values for the local context condition when $n = 2$. For SVM and C4.5, all features are converted into binary features (6665 and 12502 features).

Results. 10-fold cross validation was used to train the learning models, which were then tested on the unseen held-out test set, and the performance (including the precision, recall, and F-score[31] for each class) of the three learning algorithms is shown in Table 6.6. The significant measure is overall precision, and this is shown for

[31] F-score is computed as the harmonic mean of the precision and recall: F = (2*Prec*Rec)/(Prec+Rec).

the three algorithms in Table 6.7, together with human agreement (the upper bound of the learning task) and the baseline.

We can see that among all three learning algorithms, SVM achieves the best F-score for each class and also the best overall precision (76.6%). Compared with the baseline (59.0%) and human agreement (87.7%), this level of performance is very encouraging, especially as the learning is from such limited training data.

Class	**Algor.**	**Prec.**	**Recall**	**F-Score**
Short	**SVM**	0.707	0.606	**0.653**
	NB	0.567	0.768	0.652
	C4.5	0.571	0.600	0.585
Long	**SVM**	0.793	0.857	**0.823**
	NB	0.834	0.665	0.740
	C4.5	0.765	0.743	0.754

Table 6.6 Test Performance of Three Algorithms

Algorithm	**Precision**
Baseline	59.0%
C4.5	69.1%
NB	70.3%
SVM	**76.6%**
Human Agreement	87.7%

Table 6.7 Overall Test Precision on non-WSJ Data

Feature Evaluation. The best performing learning algorithm, SVM, was then used to examine the utility of combinations of four different feature sets (i.e., event, local context, syntactic, and WordNet hypernym features). The detailed comparison is shown in Table 6.8.

Class	**Event Only (*n* = 0)**			**Event Only + Syntactic**			**Event + Syn + Hyper**		
	Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F
Short	0.742	0.465	0.571	0.758	0.587	0.662	0.707	0.606	0.653
Long	0.748	0.908	0.821	0.792	0.893	0.839	0.793	0.857	0.823
Overall Prec.	**74.7%**			**78.2%**			**76.6%**		
	Local Context (*n* = 2)			**Context + Syntactic**			**Context + Syn + Hyper**		
Short	0.672	0.568	0.615	0.710	0.600	0.650	0.707	0.606	0.653
Long	0.774	0.842	0.806	0.791	0.860	0.824	0.793	0.857	0.823
Overall Prec.	**74.2%**			**76.6%**			**76.6%**		

Table 6.8 Feature Evaluation with Different Feature Sets using SVM

We can see that most of the performance comes from event word or phrase itself. A significant improvement above that is due to the addition of information about the subject and object. Local context does not help and in fact may hurt, and hypernym information also does not seem to help[32]. It is of interest that the most important information is that from the predicate and arguments describing the event, as our linguistic intuitions would lead us to expect.

6.3.2.4 Test on WSJ Data

The last section describes the experimental results with the learned model trained and tested on data from the same genre, i.e., non-WSJ articles. In order to evaluate

[32] In the two cases using Hyper, the learning algorithm with and without local context gives identical results, probably because the other features dominate.

whether the learned model can perform well on data from different news genres, we tested it on the unseen WSJ data (156 event instances). The performance (including the precision, recall, and F-score for each class) is shown in Table 6.9. The precision (75.0%) is very close to the test performance on the non-WSJ data, and indicates the significant generalization capacity of the learned model.

Class	**Prec.**	**Rec.**	**F**
Short	0.692	0.610	0.649
Long	0.779	0.835	0.806
Overall Prec.	**75.0%**		

Table 6.9 Test Performance on WSJ data

6.3.3 Learning the Most Likely Temporal Unit

These encouraging results prompted us to try to learn more fine-grained event duration information, viz., the most likely temporal units of event durations (cf. (Rieger 1974)'s ORDERHOURS, ORDERDAYS).

For each original event annotation, we can obtain the most likely (mean) duration by averaging its lower and upper bound durations, and assigning it to one of seven classes (i.e., second, minute, hour, day, week, month, and year) based on the temporal unit of its most likely duration.

However, human agreement on this more fine-grained task is low (44.4%). This is understandable. An annotation of [30 minutes, 1 hour] and [35 minutes, 2 hours] will not match, even the area of their overlap is 52.72%[33].

Based on this observation, instead of evaluating the *exact* agreement between annotators, an "*approximate* agreement" is computed for the most likely temporal unit of events. In "approximate agreement", temporal units are considered to match if they are the same temporal unit or an adjacent one. For example, "second" and "minute" match, but "minute" and "day" do not.

We conducted an experiment for learning this multi-classification task. The same data sets as in the binary classification task were used. The only difference was that the class for each instance was now labeled with one of the seven temporal unit classes.

The baseline for this multi-classification task is always taking the temporal unit which with its two neighbors spans the greatest amount of data. Since the "week", "month", and "year" classes together take up largest portion (51.5%) of the data, the baseline is always taking the "month" class, where both "week" and "year" are also considered a match. Table 6.10 shows the inter-annotator agreement results for most likely temporal unit when using "approximate agreement". Human agreement (the upper bound) for this task increases from 44.4% to 79.8%.

[33] A natural logarithmic scale is used for the overlap/agreement computation, as shown in Section 3.1.

P(A)	P(E)	Kappa
0.798	0.151	0.762
	0.143	0.764

Table 6.10 Inter-Annotator Agreement for Most Likely Temporal Unit

10-fold cross validation was also used to train the learning models, which were then tested on the unseen held-out test set. The performance of the three algorithms is shown in Table 6.11. The best performing learning algorithm is again SVM with 67.9% test precision. Compared with the baseline (51.5%) and human agreement (79.8%), this again is a very promising result, especially for a multi-classification task with such limited training data. It is reasonable to expect that when more annotated data becomes available, the learning algorithm will achieve higher performance when learning this and more fine-grained event duration information.

Algorithm	Precision
Baseline	51.5%
C4.5	56.4%
NB	65.8%
SVM	**67.9%**
Human Agreement	79.8%

Table 6.11 Overall Test Precisions

Although the coarse-grained duration information may look too coarse to be useful, computers have no idea at all whether a meeting event takes seconds or centuries, so even coarse-grained estimates would give it a useful rough sense of how long each event may take. More fine-grained duration information is definitely more desirable for temporal reasoning tasks. But coarse-grained durations to a level of temporal units can already be very useful.

6.4 Integrating Vague Event Durations to TimeML

This section describes our work on integrating annotations of typical durations of events to TimeML, which can enrich the expressiveness of TimeML, and provides natural language applications that exploit TimeML with this additional implicit event duration information for their temporal information processing tasks.

6.4.1 Event Classes in TimeML

Our event duration annotations can be integrated into the EVENT tag. In TimeML each event belongs to one of the seven event classes, i.e., reporting, perception, aspectual, I-action, I-state, state, occurrence. TimeML annotation guidelines[34] give detailed description for each of the classes:

Reporting. This class describes the action of a person or an organization declaring something, narrating an event, informing about an event, etc (e.g., say, report, tell, explain, state).

Perception. This class includes events involving the physical perception of another event (e.g., see, watch, view, hear).

Aspectual. In languages such as English and French, there is a grammatical device of aspectual predication, which focuses on different facets of event history, i.e.,

[34]http://www.cs.brandeis.edu/~jamesp/arda/time/timeMLdocs/annguide12wp.pdf

initiation, reinitiation, termination, culmination, continuation (e.g., begin, stop, finish, continue).

I-Action. An I-Action is an Intensional Action. It introduces an event argument (which must be in the text explicitly) describing an action or situation from which we can infer something given its relation with the I-Action (e.g., attempt, try, promise).

I-State. This class of events are similar to the previous class. This class includes states that refer to alternative or possible worlds (e.g., believe, intend, want).

State. This class describes circumstances in which something obtains or holds true (e.g., on board, kidnapped, peace).

Occurrence. This class includes all the many other kinds of events describing something that happens or occurs in the world (e.g., die, crash, build, sell).

6.4.2 Integrating Event Duration Annotations

Our event duration annotations can be integrated into TimeML by adding two more attributes to the EVENT tag for the lower bound and upper bound duration annotations (e.g., the `lowerBoundDuration` and `upperBoundDuration` attributes).

To minimize changes of the existing TimeML specifications caused by the integration, we can try to share as much as possible our event classes as described in Section 2.2 with the existing ones in TimeML as described in Section 3.

We can see that four event classes are shared with very similar definitions, i.e., reporting, aspectual, state, and action/occurrence. For the other three event classes that only belong to TimeML (i.e., perception, I-action, I-state), the I-action and perception classes can be treated as special subclasses of the action/occurrence class, and the I-state class as a special subclass of the state class.

However, there are still three classes that only belong to the event duration annotations (i.e., multiple, negation, and positive infinite). The positive infinite class can be treated as a special subclass of the state class with a special duration annotation for positive infinity.

Each multiple event has two annotations. For example, for

*Iraq has <u>destroyed</u> its long-range **missiles**.*

there is the time it takes to destroy one missile and the duration of the interval in which all the individual events are situated – the time it takes to destroy all its missiles.

Since the single event is usually more likely to be encountered in multiple documents, and thus the duration of the single event is usually more likely to be shared and re-used, to simplify the specification, we can take only the duration annotation of the single events for the multiple event class, and the single event can be assigned with one of the seven TimeML event classes. For example, the "destroyed" event in the above example is assigned with the occurrence class in TimeBank.

The events involving negation can be simplified similarly. Since the event negated is usually more likely to be encountered in multiple documents, we can take only the duration annotation of the negated event for this class. For example, in

He was willing to withdraw troops in exchange for guarantees that Israel would ***not*** *be* <u>*attacked*</u>*.*

the event negated is the "being attacked" event and it is assigned with the occurrence class in TimeBank. Alternatively, TimeML could be extended to treat negations of events as states.

The format used for annotated durations is consistent with that for the value of the DURATION type in TimeML. For example, the sentence

The official ***said*** *these sites could only be* ***visited*** *by a special team of U.N. monitors and diplomats.*

can be marked up in TimeML as:

```
The official <EVENT eid="e63" class="REPORTING"> said </EVENT>
these sites <SIGNAL sid="s65" >could</SIGNAL> only be <EVENT
eid="e64" class="OCCURRENCE"> visited </EVENT> by a special
team of <ENAMEX TYPE="ORGANIZATION"> U.N. </ENAMEX> monitors
and diplomats.
```

If we annotate the "said" event with the duration annotation of [5 seconds, 5 minutes], and the "visited" event with [10 minutes, 1 day], the extended mark-up becomes:

```
The official <EVENT eid="e63" class="REPORTING" lowerBound-
Duration="PT5S" upperBoundDuration="PT5M"> said </EVENT> these
sites <SIGNAL sid="s65" >could</SIGNAL> only be <EVENT eid="e64"
class="OCCURRENCE" lowerBoundDuration="PT10M" upperBound-
Duration="P1D"> visited </EVENT> by a special team of <ENAMEX
TYPE="ORGANIZATION"> U.N. </ENAMEX> monitors and diplomats.
```

6.5 Representing and Reasoning about Vague Event Durations

Vague event durations can be represented in FOL with a simple predicate, for example, "vagueDuration" (where T is an interval, D_1 and D_1 are the descriptions of the lower and upper bound durations):

$$vagueDuration(T, D_1, D_2) \supset Interval(T) \wedge DurationDescription(D_1) \\ \wedge DurationDescription(D_2)$$

More axioms can be added to support the reasoning on the vague event durations. This can be translated into OWL very straightforwardly (XSD durations can be used in the place of duration descriptions):

```
<owl:ObjectProperty rdf:ID="vagueDurationOf">
  <rdfs:domain rdf:resource="#Interval" />
  <rdfs:range  rdf:resource="#VagueDuration" />
</owl:ObjectProperty>

<owl:Class rdf:ID="VagueDuration">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#lowerDuration" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
        </owl:cardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#upperDuration" />
        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
        </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="lowerDuration">
  <rdfs:domain rdf:resource="#VagueDuration " />
  <rdfs:range  rdf:resource="#DurationDescription" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="upperDuration">
  <rdfs:domain rdf:resource="#VagueDuration " />
  <rdfs:range  rdf:resource="#DurationDescription" />
</owl:DatatypeProperty>
```

In terms of reasoning about this vague event durations, Section 7 describes sets of mapping axioms to map subsets of the problems that can be represented by the time ontology in FOL to temporal constraint-based formalisms for more efficient temporal reasoning. Since temporal constraint-based formalisms usually specify *a range of durations* as constraints, they are very suitable for temporal reasoning with vague durations.

If we want to capture the probabilistic properties of vague event durations that are modeled by the normal distribution, we would need to incorporate probability into FOL and OWL. However, results of (Halpern, 1990; Abadi and Halpern, 1994) show that in general it is impossible to have complete axiomatization (deductive system) of FOL of probability, though a complete axiomatization system can be obtained for some special cases. A probabilistic extension to OWL for uncertainty modeling is proposed in (Ding and Peng, 2004).

Chapter 7

More Efficient Reasoning with Temporal Constraint-Based Formalisms

Due to the undecidability and high computational complexity of FOL theorem proving, sets of mapping axioms have been developed in order to map subsets of the problems that can be represented in the time ontology in FOL to different temporal constraint-based formalisms for more efficient temporal reasoning. This will allow applications that require efficient temporal reasoning to use our ontology when the problem (or part of the problem) can be mapped to a temporal constraint-based formalism by the mapping axioms.

Temporal constraint-based formalisms usually specify *a range of durations* as constraints. Thus it is naturally a very suitable formalism for reasoning about vague event durations described in Section 6.

Section 7.1 describes different kinds of temporal constraint-based formalisms. The mapping axioms are discussed in Section 7.2 for each of the formalisms.

7.1 Temporal Constraint-Based Formalisms

There are two types of temporal constraints: qualitative and quantitative. Qualitative constraints specify temporal relations between temporal entities, and quantitative constraints place absolute or duration bounds to temporal entities.

Both interval algebra (Allen, 1983) and point algebra (Vilain and Kautz, 1986) can be represented and reasoned with qualitative temporal networks. While it's generally NP-complete to decide consistency for interval algebra constraint networks, polynomial time can be achieved for point algebra constraint networks.

Quantitative temporal constraint satisfaction problems (TCSPs) involve a set of variables $\{x_1, \ldots, x_n\}$ with continuous domains. Each variable represents a time point. A unary constraint restricts the domain of variable x_i to the given set of intervals, for example, $(a_1 \leq x_i \leq b_1) \vee \ldots \vee (a_n \leq x_i \leq b_n)$. A binary constraint restricts the duration of $(x_j - x_i)$ to the given set of intervals, for example, $(a_1 \leq x_j - x_i \leq b_1) \vee \ldots \vee (a_n \leq x_j - x_i \leq b_n)$. For example, Ex 7.1 can be formulated as a TCSP, as shown in Figure 7.1 (Dechter et al., 1991). It can answer queries like "what are the possible times at which Fred left home?". The general reasoning in TCSPs is NP-hard.

> [Ex 7.1] (Dechter et al., 1991) John travels to work either by car (30-40 minutes) or by bus (at least 60 minutes). Fred travels to work either by car (20-30 minutes) or in a carpool (40-50 minutes). Today John left home between 7:10 and 7:20 AM, and Fred arrived at work between 8:00 and 8:10

AM. We also know that John arrived at work 10-20 minutes after Fred left home.

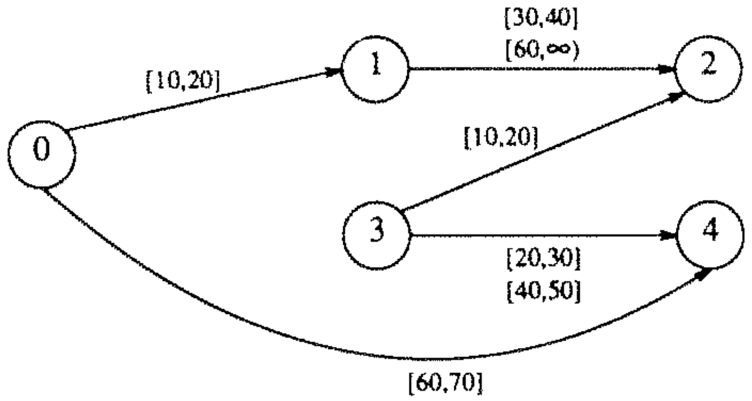


Figure 7.1 A TCSP Example

Polynomial solutions exist for a special kind of TCSPs -- Simple Temporal Problems (STPs) (Dechter et al., 1991). A STP is a TCSP where all the constraints specify a single interval, i.e., no disjunction is allowed. For example, Ex 7.1 can be simplified to a STP by removing alternatives (disjunctions) from the constraints, i.e., John only used a car and Fred only used a carpool.

Disjunctive Temporal Problems (DTPs) extends STPs by allowing disjunctive constraints and TCSPs by allowing constraints with arbitrary arity (e.g., constraints with three or more variables) (Stergiou and Koubarakis, 1998). For example, to represent the constraint "two intervals *A* and *B* cannot overlap", four variables are needed, i.e., A_S, A_E, B_S, B_E for the start and end times of the intervals, and the DTP constraint is: $(A_E - B_S) \leq 0 \vee (B_E - A_S) \leq 0$. A DTP can be viewed as a collection of alternative STPs. Efficient techniques have been proposed to solve DTPs with exact solutions (Tsamardinos and Pollack, 2003) or approximate solutions (Moffitt and

Pollack, 2005). Planning and scheduling tasks are the applications that they are most concerned with.

Although DTPs are more expressive than STPs and TCSPs, they can only represent duration constraints (plus simple relations as before/after/at given time points, which can be converted to their representation), and there are no calendar or clock concepts involved. It cannot represent temporal aggregates or any non-temporal properties either.

TCSP-based framework has also been applied to temporal reasoning for natural language (Han and Lavie, 2004), where not only the durations but also the calendar and granularities are modeled as a two-level constraint satisfaction problem: at the higher level is a TCSP relating different temporal objects (variables), and within each temporal variable is a model of the calendar – a unary TCSP describing constraints among different temporal units. However, it cannot represent arbitrary temporal aggregates in their current framework (Han et al., 2006). A special-purpose reasoner, which is a conventional TCSP solver with the aid of the calendar model, has to be used to draw inferences on given temporal expressions.

A formal framework for the definition and representation of time granularities was proposed in (Bettini et al., 2000) especially for temporal database applications. This framework has also been extended to TCSPs (Bettini et al., 2002) for more general temporal reasoning tasks. The framework extends STPs with multiple time granularities, for example, the difference between two variables is 5 to 10 business days and they must be in the same month. They also propose a sound and complete algorithm to solve the problem, and the algorithm is exponential in terms of the involved granularities, but polynomial in terms of the variables and constraints in the network.

Morris and Khatib (1998) have proposed a constraint-based formulation of reasoning problems with repeating events. The problem (e.g., a set or sequence of subintervals I_i) is characterized with five properties: number of subintervals, subinterval duration, gap duration (the duration between consecutive subintervals), period (the duration between consecutive subinterval start times), and extent (the duration from the start of the earliest subinterval to the end of the latest). Each property defines a type of quantitative constraint for a repeating event. Ex 7.2 formalizes the constraint on *I* that the number of subintervals is between 3 and 5, the duration of each subinterval is between 4 and 7 time units, the gap between subintervals is between 2 and 3 time units, and the extent is between 30 and 50 time units.

[Ex 7.2] (Morris and Khatib, 1998) The following is a set of constraints for a repeating event *I*:

- Number: $3 \leq n_I \leq 5$
- Sub-interval duration: $4 \leq d_i \leq 7$, $i = 1 \dots |I|$
- Gap duration: $2 \leq g_i \leq 3$, $i = 1 \dots |I| - 1$
- Extent: $30 \leq e_I \leq 50$

The complexity of different classes of the problem is discussed, and an efficient algorithm is proposed for the class of the problem with binary and convex constraints [35] (Morris and Khatib, 2000). Qualitative constraints have also been integrated into the formulation to represent temporal relations. However, the formulation can only handle one layer regular temporal aggregates (either with a fixed gap, or the gap distribution is known). It cannot represent multi-layered temporal aggregates or ones with an irregular gap (e.g., a weekly meeting on any business days).

7.2 Mapping Axioms to Temporal Constraint-Based Formalisms

The mapping axioms have been created for each of the different temporal constraint-based formalisms for more efficient temporal reasoning. As discussed in the previous section, some formalisms (e.g., STP) have polynomial solutions, some don't but may still have very efficient reasoning algorithms.

For each of the mapping axioms, when the conditions on the left hand side hold, the problem can be mapped to an equivalent one on the right hand side in the corresponding temporal constraint-based formalism.

- **STP, assuming "second" is the basic temporal unit (polynomial solutions exist):**

[35] A convex constraint means the range of possible values is a single interval.

➢ Vague duration between two instants:

$$Interval(T) \wedge begins(t_1, T) \wedge ends(t_2, T) \wedge \text{seconds}(T) = n \wedge n_1 \leq n \leq n_2$$[36]

$$\Rightarrow n_1 \leq (t_2 - t_1) \leq n_2$$

When the duration between two instants (i.e., t_1 and t_2) is n seconds and n is in a range of $[n_1, n_2]$, the problem can be mapped to a STP problem where the two instants are two temporal variables and their difference is in the range of $[n_1, n_2]$. Polynomial solutions exist for this problem.

- **TCSP, assuming "second" is the basic temporal unit:**

➢ Alternative vague durations between two instants:

$$Interval(T) \wedge begins(t_1, T) \wedge ends(t_2, T) \wedge \text{seconds}(T) = n$$
$$\wedge (n_1 \leq n \leq n_2 \vee n_3 \leq n \leq n_4)$$
$$\Rightarrow n_1 \leq (t_2 - t_1) \leq n_2 \vee n_3 \leq (t_2 - t_1) \leq n_4$$

TCSP is STP with disjunctions. When the duration between two instants (i.e., t_1 and t_2) is n seconds and n is in a range of $[n_1, n_2]$ or $[n_3, n_4]$, the problem can be mapped to a TCSP problem where the two instants are two temporal variables and their difference is in the range of $[n_1, n_2]$ or $[n_3, n_4]$.

[36] Prob($n_1 \leq n \leq n_2$) = 0.8 would be more accurate in terms of the vague event durations defined in Section 6. But here we want to keep it in the standard first-order logic without probability reasoning.

- **Bettini et al. (2002) with multiple granularities (polynomial solutions exist):**

➢ Vague duration between two instants with one granularity:

$$Interval(T) \wedge begins(t_1, T) \wedge ends(t_2, T) \wedge duration(T,G) = n$$
$$\wedge\ TemporalUnit(G) \wedge n_1 \le n \le n_2$$
$$\Rightarrow (t_2 - t_1) \models [n_1, n_2]\, G$$

This formalism is STP with one granularity. When the duration between two instants (i.e., t_1 and t_2) is n with a granularity (temporal unit) of G and n is in a range of $[n_1, n_2]$, the problem can be mapped to a formalism in (Bettini et al. 2002) with one granularity where the two instants are two temporal variables and their difference is in the range of $[n_1, n_2]$ with granularity (temporal unit) of G. Polynomial solutions exist for this problem.

The definition of the notation "$[m, n]\, G$" is (Bettini et al., 2002):

Let $m,n \in Z \cup \{-\infty,+\infty\}$ with $m \le n$ and G a granularity. Then $[m, n]\, G$, called a *temporal constraint with granularity* (TCG), is the binary relation on positive integers defined as follows: for positive integers t_1 and t_2, $(t_1, t_2) \models [m, n]\, G$ ((t_1, t_2) *satisfies* $[m, n]\, G$) if and only if (1) $\lceil t_1 \rceil^G$ and $\lceil t_2 \rceil^G$ are both defined, and (2) $m \le (\lceil t_2 \rceil^G - \lceil t_1 \rceil^G) \le n$, where the $\lceil x \rceil^G$ function returns the index of the granule of G with the bottom granularity.

For example, the pair (t_1, t_2) satisfies [0, 0] Day if t_1 and t_2 are within the same day; (t_1, t_2) satisfies [-1, 1] Hour if t_1 and t_2 are at most one hour apart; (t_1, t_2) satisfies [1, 1] Month if t_2 is in the next month with respect to t_1.

➢ Vague duration between two instants with multiple granularities:

$$Interval(T) \wedge begins(t_1, T) \wedge ends(t_2, T) \wedge durationOf(T,y,m,w,d,h,n,s)$$
$$\wedge\ y_1 \leq y \leq y_2 \wedge m_1 \leq m \leq m_2 \wedge w_1 \leq w \leq w_2 \wedge d_1 \leq d \leq d_2 \wedge h_1 \leq h \leq h_2$$
$$\wedge\ n_1 \leq n \leq n_2 \wedge s_1 \leq s \leq s_2$$
$$\Rightarrow (t_2 - t_1) \models [y_1, y_2] \text{ Year} \wedge (t_2 - t_1) \models [m_1, m_2] \text{ Month}$$
$$\wedge\ (t_2 - t_1) \models [w_1, w_2] \text{ Week} \wedge (t_2 - t_1) \models [d_1, d_2] \text{ Day}$$
$$\wedge\ (t_2 - t_1) \models [h_1, h_2] \text{ Hour} \wedge (t_2 - t_1) \models [n_1, n_2] \text{ Minute}$$
$$\wedge\ (t_2 - t_1) \models [s_1, s_2] \text{ Second}$$

This formalism is STP with multiple granularities. When the duration between two instants (i.e., t_1 and t_2) is with multiple granularities (i.e., y years in a range of $[y_1, y_2]$, m months in a range of $[m_1, m_2]$, w weeks in a range of $[w_1, w_2]$, d days in a range of $[d_1, d_2]$, h hours in a range of $[h_1, h_2]$, n seconds in a range of $[n_1, n_2]$, and s seconds in a range of $[s_1, s_2]$), the problem can be mapped to a formalism in (Bettini et al. 2002) with multiple granularities where the two instants are two temporal variables and their difference is in the corresponding range for each granularity. Polynomial solutions exist for this problem.

➢ Vague duration between two instants with gap granularities, e.g., weekday[37]

$$weekday(y,x) \equiv [Monday(y,x) \vee Tuesday(y,x) \vee Wednesday(y,x)$$
$$\vee\ Thursday(y,x) \vee Friday(y,x)]$$

[37] In (Bettini et al., 2002), weekdays are business days ("b-day")

The statement of "the duration between t_1 and t_2 is x business days" can be converted to an equivalent temporal aggregate statement: "x business days between t_1 and t_2". Thus, the mapping axiom is

$$\begin{aligned}&everyp(s, s_0, weekday1) \wedge card(s) = x \wedge x_1 \le x \le x_2 \wedge first(T_1,s) \wedge last(T_2,s)\\ &\wedge begins(t_1,T_1) \wedge ends(t_2,T_2) \wedge timeOf(t_1,y_1,m_1,d_1,h_1,n_1,s_1,z_1)\\ &\wedge timeOf(t_2,y_2,m_2,d_2,h_2,n_2,s_2,z_2)\\ &\qquad \Rightarrow (t_2 - t_1) \models [x_1, x_2] \text{ b-day}\end{aligned}$$

where $(\forall\, d)\ [weekday1(d) \equiv (\exists\, w)\ [weekday(d,w)]]$

This formalism is STP with gap granularities. When the duration between two instants (i.e., t_1 and t_2) is x with a gap granularity, e.g., weekday, and x is in a range of $[x_1, x_2]$, the problem can be mapped to a formalism in (Bettini et al. 2002) with gap granularities where the two instants are two temporal variables and their difference is in the range of $[x_1, x_2]$ with the gap granularity, e.g., weekday. Polynomial solutions exist for this problem.

- **Morris and Khatib (1998) for repeating events, assuming "second" is the basic temporal unit:**

  - n: number of subintervals, $n_1 \le n \le n_2$
  - d: subinterval duration, $d_1 \le d \le d_2$
  - g: gap duration, $g_1 \le g \le g_2$
  - p: period, $p_1 \le p \le p_2$
  - e: extent, $e_1 \le e \le e_2$

$tseq(s) \supset card(s) = \boldsymbol{n}$

$\supset (\exists\, n_1,n_2)\ n_1 \leq \boldsymbol{n} \leq n_2]$

$tseq(s) \supset [(\forall\, T,d)\ member(T,s) \wedge seconds(T) = \boldsymbol{d}$

$\supset (\exists\, d_1,d_2)\ d_1 \leq \boldsymbol{d} \leq d_2]$

$tseq(s) \supset [(\forall T, T_1,T_2,t_1,t_2,g)\ member(T_1,s) \wedge member(T_2,s) \wedge nth(T_1,n,s)$
$\wedge\ nth(T_2,n{+}1,s) \wedge ends(t_1,T_1) \wedge begins(t_2,T_2)$
$\wedge\ timeBetween(T,t_1,t_2) \wedge seconds(T) = \boldsymbol{g}$
$\supset (\exists\, g_1,g_2)\ g_1 \leq \boldsymbol{g} \leq g_2]$

$tseq(s) \supset [(\forall T,T_1,T_2,t_1,t_2,p)\ member(T_1,s) \wedge member(T_2,s) \wedge nth(T_1,n,s)$
$\wedge\ nth(T_2,n{+}1,s) \wedge begins(t_1,T_1) \wedge begins(t_2, T_2)$
$\wedge\ timeBetween(T,t_1,t_2) \wedge seconds(T) = \boldsymbol{p}$
$\supset (\exists\, p_1,p_2)\ p_1 \leq \boldsymbol{p} \leq p_2]$

$tseq(s) \supset [(\forall T,T_1,T_2,t_1,t_2,e)\ first(T_1,s) \wedge first(T_1,s) \wedge last(T_2,s) \wedge begins(t_1,T_1)$
$\wedge\ ends(t_2,T_2) \wedge timeBetween(T,t_1,t_2) \wedge seconds(T) = \boldsymbol{e}$
$\supset (\exists\, e_1,e_2)\ e_1 \leq \boldsymbol{e} \leq e_2]$

There are five properties to specify for a given problem with repeating events (i.e., temporal aggregates): number of subintervals (i.e., the cardinality/size of a temporal sequence) is n and it's in a range of $[n_1, n_2]$; subinterval duration (i.e., the duration of each member of a temporal sequence) is d seconds and d is in a range of $[d_1, d_2]$; gap duration (i.e., the duration between consecutive members of a temporal sequence) is g seconds and g is in a range of $[g_1, g_2]$; period (i.e., the duration between start times of consecutive members of a temporal sequence) is p seconds and p is in a range of [p_1,

p_2]; and extent (i.e., the duration from the start of the first member of a temporal sequence to the end of the last member) is e seconds and e is in a range of [e_1, e_2].

We will use the STP version of Ex 7.1 (i.e., remove disjunctions) to show the represention of the problem in OWL-Time, and illustrate how to apply the mapping axioms to map it to STP constraints. The STP version of Ex 7.1 is as follows:

"John travels to work by car (30-40 minutes). Fred travels to work in a carpool (40-50 minutes). Today John left home between 7:10 and 7:20 AM, and Fred arrived at work between 8:00 and 8:10 AM. We also know that John arrived at work 10-20 minutes after Fred left home."

The sentence-by-sentence representation of the problem in OWL-Time is:

"John travels to work by car (30-40 minutes)":

$Interval(T_1) \wedge begins(t_1, T_1) \wedge ends(t_2, T_1) \wedge duration(T_1, *Minute*) = n_1 \wedge 30 \leq n_1 \leq 40$

"Fred travels to work in a carpool (40-50 minutes)":

$Interval(T_2) \wedge begins(t_3, T_2) \wedge ends(t_4, T_2) \wedge duration(T_2, *Minute*) = n_2 \wedge 40 \leq n_2 \leq 50$

"Today John left home between 7:10 and 7:20 AM":

$timeOf(t_1, y, m, d, 7, n_3, 0) \wedge 10 \leq n_3 \leq 20$

"Fred arrived at work between 8:00 and 8:10 AM":

$timeOf(t_4, y, m, d, 8, n_4, 0) \wedge 0 \leq n_4 \leq 10$

"John arrived at work 10-20 minutes after Fred left home":

$Interval(T_5) \wedge begins(t_3, T_5) \wedge ends(t_2, T_5) \wedge duration(T_5, *Minute*) = n_5$

$\wedge\ 10 \leq n_5 \leq 20$

The following axioms can be straightforwardly mapped to STP constraints (assuming "minute" is the basic temporal unit):

$Interval(T_1) \wedge begins(t_1, T_1) \wedge ends(t_2, T_1) \wedge duration(T_1, *Minute*) = n_1 \wedge 30 \leq n_1 \leq 40$

$\Rightarrow Interval(T_1) \wedge begins(t_1, T_1) \wedge ends(t_2, T_1) \wedge minutes(T_1) = n_1 \wedge 30 \leq n_1 \leq 40$

$\Rightarrow 30 \leq (t_2 - t_1) \leq 40$

$Interval(T_2) \wedge begins(t_3, T_2) \wedge ends(t_4, T_2) \wedge duration(T_2, *Minute*) = n_2 \wedge 40 \leq n_2 \leq 50$

$\Rightarrow Interval(T_2) \wedge begins(t_3, T_2) \wedge ends(t_4, T_2) \wedge minutes(T_2) = n_2 \wedge 40 \leq n_2 \leq 50$

$\Rightarrow 30 \leq (t_4 - t_3) \leq 40$

$Interval(T_5) \wedge begins(t_3, T_5) \wedge ends(t_2, T_5) \wedge duration(T_5, *Minute*) = n_5 \wedge 10 \leq n_5 \leq 20$

$\Rightarrow Interval(T_5) \wedge begins(t_3, T_5) \wedge ends(t_2, T_5) \wedge minutes(T_5) = n_5 \wedge 10 \leq n_5 \leq 20$

$\Rightarrow 10 \leq (t_2 - t_3) \leq 20$

The following axioms can be mapped to STP constraints, if we add 7:00 AM (t_0), as in the original example:

$timeOf(t_1, y, m, d, 7, n_3, 0) \wedge 10 \leq n_3 \leq 20 \wedge timeOf(t_0, y, m, d, 7, 0, 0)$

$\Rightarrow Interval(T_3) \wedge begins(t_0, T_3) \wedge ends(t_1, T_1) \wedge duration(T_3, *Minute*) = n_3$

$\wedge\ 10 \leq n_3 \leq 20$

$\Rightarrow 10 \leq (t_1 - t_0) \leq 20$

$timeOf(t_4, y, m, d, 8, n_4, 0) \wedge 0 \leq n_4 \leq 10 \wedge timeOf(t_0, y, m, d, 7, 0, 0)$

$\Rightarrow Interval(T_4) \wedge begins(t_0, T_4) \wedge ends(t_4, T_4) \wedge duration(T_4, *Minute*) = n_4$

$\wedge\ 60 \leq n_4 \leq (60+10)$

$\Rightarrow 60 \leq (t_4 - t_0) \leq 70$

All the above constraints correspond to all the temporal constraints represented in the temporal constraint network in Figure 7.1 (STP version without disjunctions).

Chapter 8

Conclusion

In this dissertation, we have developed a rich ontology of temporal concepts represented in both FOL and OWL Web Ontology Language, which can be used for the Semantic Web, natural language, and many other applications from different domains. It covers a very complete set of temporal concepts by extending Hobbs (2002)'s work with more complex temporal phenomena, such as temporal aggregates, temporal arithmetic mixing months and days, and vague event durations. Mapping rules have also been developed to map subsets of the problems that can be represented by the ontology in FOL to different kinds of temporal constraint-based formalisms for more efficient temporal reasoning. A time zone resource is developed in OWL for the entire world. In the future, when a rule language for the Semantic Web is endorsed by W3C, the FOL axioms in the ontology can be straightforwardly translated to that rule language, and the Semantic Web applications can then benefit more from the ontology's deep temporal reasoning capacity.

The temporal aggregate part of the ontology is rich enough to handle both complex multiple-layered and conditional temporal aggregates in FOL and OWL, which is very useful for representing recurrent events. A systematic way of mapping recurrence sets in iCalendar to temporal sequences in OWL-Time is developed to gives it access to the full ontology of time for temporal reasoning. To the best of our knowledge, it is the first attempt to develop a set of rules for doing temporal arithmetic mixing months and days based on the "history-dependent intuition" with consideration of different desired arithmetic properties, such as the subtraction, commutativity and associativity properties.

In this thesis we have also addressed a problem – modeling and extracting vague event durations from text -- that has heretofore received very little attention in the field. It is information that can have a substantial impact on applications where the temporal placement of events is important, and the exact and explicit duration information is not available.

In fact, it is representative of a set of problems – making use of the vague information in text – that has largely eluded empirical approaches in the past. We have explicated the linguistic categories of the phenomena that give rise to grossly discrepant judgments among annotators, and give guidelines on resolving these discrepancies. We have also described a method for measuring inter-annotator agreement when the judgments are intervals on a scale; this should extend from time to other kinds of vague but substantive information. This is the first empirical effort to learn the implicit and vague event durations from natural language text, and we have shown that the learning performance is very impressive. The vague event duration annotations have also been integrated into the TimeBank corpus for more widely usage. As future work, we can try some unsupervised machine learning approaches (e.g., some information extraction methods) to extract the vague event durations from larger corpora, e.g., the Web. Then we can compare the distributions of event durations between the two approaches.

References

Abadi, M. and J. Y. Halpern. 1994. Decidability and expressiveness for first-order logics of probability. *Information and Computation*,112(1):1-36.

Agarwal, P. 2005. Ontological considerations in GIScience. *International Journal of Geographical Information Science*, vol. 19, pp. 501-535.

Allen, J. F. 1984. Towards a general theory of action and time. *Artificial Intelligence* 23, pp. 123-154.

Allen, J. F. and Ferguson, G. 1994. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531-579.

Allen, J. F. and Ferguson, G. 1997. Actions and events in interval temporal logic. In *Spatial and Temporal Reasoning*. O. Stock, ed., Kluwer, Dordrecht, Netherlands, 205-245.

Attard, M. and M. Montebello. 2006. DoNet: a semantic domotic framework, *Proceedings of the 15th international conference on World Wide Web (WWW)*, Edinburgh, Scotland.

van Benthem, J. 1983. The Logic of Time. *Kluwer Academic Publishers*.

van Benthem, J. 1995. Temporal Logic, in D. M. Gabbay, C. J. Hogger, and J. A. Robinson, *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 4, Oxford: Clarendon Press, pages 241-350.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web. *Scientific American*, 284(5):34–43.

Bettini, C., Jajodia, S., Wang, X. S. 2000. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*, Springer-Verlag.

Bettini, C., Wang, S. X. and Jajodia, S.. 2002. Solving multi-granularity temporal constraint networks, *Artificial Intelligence*, vol. 140, pp. 107-152.

Biron, P. V. and Malhotra, A. 2004. XML Schema Part 2: Datatypes Second Edition. *W3C* Recommendation. http://www.w3.org/TR/xmlschema-2/

Boguraev, B. and R. K. Ando. 2005. TimeML-Compliant Text Analysis for Temporal Reasoning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI).*

Boley, H.; Tabet, S.; and Wagner, G. 2001. Design rationale of RuleML: A markup language for semantic web rules. In *Proceedings of the International Semantic Web Working Symposium*.

Bry, F., F.-A. Rieß, and S. Spranger. 2005. CaTTS: Calendar Types and Constraints for Web Applications. In *Proceedings of the 14th international conference on World Wide Web (WWW).*

Bulcao Neto, R. F. and M. G. C. Pimentel. 2005. Toward a domain-independent semantic model for context-aware computing. In *Proceedings of the 3rd Latin American Web Congress (LA-WEB).*

Carletta, J. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Lingustics*, 22(2):249–254.

Chandra, R., Segev, A., and Stonebraker, M. 1994. Implementing Calendars and Temporal Rules in Next Generation Databases". In *Proceedings of the Tenth International Conference on Data Engineering*, pages 264–273, Houston, Texas.

Chen, H.; Finin, T.; and Joshi, A. 2003. An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 2003.

Chen, H.; Perich, F.; Chakraborty, D.; Finin, T.; and Joshi, A. 2004. Intelligent agents meet semantic web in a smart meeting room. In *Proceedings of the third International Joint Conference on Autonomous Agents and Multi Agent Systems*.

Chen, H., F. Perich, T. Finin, and A. Joshi. 2004. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *Proceedings of First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous).*

Chittaro, L. and A. Montanari. 2000. Temporal Representation and Reasoning in Artificial Intelligence: Issues and Approaches. *Annals of Mathematics and Artificial Intelligence*, vol. 28, no.1-4, pp. 47-106.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurements*, 20:37–46.

Dawson, F. Stenerson, D. 1998. Internet Calendaring and Scheduling Core Object Specification (iCalendar), RFC2445, *Internet Society*.

Dechter, R., I. Meiri and J. Pearl (1991). Temporal constraint networks. *Artificial Intelligence* 49: 61-95.

Díaz, O.; Iturrioz, J.; Irastorza, A. 2005. Improving portlet interoperability through deep annotation. In *Proceedings of the 14th international conference on World Wide Web*.

Ding, Z. and Y. Peng. 2004. A Probabilistic Extension to Ontology Language OWL. In *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-37)*.

Dobson, G. and Sommerville, I. 2006. Disambiguating Availability Specification through the use of OWL. In *Proceedings of the 2nd International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements (SOCCER)*.

Doddington, G., A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, R. Weischedel. 2004. The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation, In *Proceedings of LREC*, pp. 837-840.

Dowty, D. 1979. *Word Meaning and Montague Grammar*, Dordrecht, Reidel.

Dowty, D. 1982. Tenses, time-adverbs, and compositional semantic theory, *Linguistics and Philosophy*, 5, 23-55.

Dreyer, W, Dittrich, A. Koa, and Schmidt, D. 1994. Research Perspectives for Time Series Management Systems. *SIGMOD RECORD*, Vol. 23, No. 1, pp. 10-15.

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

Eugenio, B. D. and Glass, M. 2004. The Kappa statistic: a second look. *Computational Linguistics*, 30(1):95–101.

Ferro, L. 2001. Instruction Manual for the Annotation of Temporal Expressions. Mitre Technical Report MTR 01W0000046, *the MITRE Corporation*, McLean, Virginia.

Filatova, E. and E. Hovy. 2001. Assigning Time-Stamps to Event-Clauses. *Proceedings of ACL Workshop on Temporal and Spatial Reasoning*.

Fortemps, P. 1997. Jobshop Scheduling with Imprecise Durations: A Fuzzy Approach. *IEEE Transactions on Fuzzy Systems* Vol. 5 No. 4.

Freksa, C. 1992. Temporal Reasoning based on Semi-Intervals. *Artificial Intelligence*, Vol. 54:199-227.

Galton, A. 1984. *The Logic of Aspect*, Clarendon Press Oxford.

Genesereth, M. 1991. "Knowledge Interchange Format", In Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning, Allen, J., Fikes, R., Sandewall, E. (eds), Morgan Kaufman Publishers, pp 238-249.

Gildea, D. and D. Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245-288.

Godo, L. and L. Vila. 1995. Possibilistic Temporal Reasoning based on Fuzzy Temporal Constraints. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI).*

Grosof, B., M. Kifer, and D. L. Martin. Rules in the Semantic Web Services Language (SWSL): An overview for standardization directions. In *Proceedings of the W3C Workshop on Rule Languages for Interoperability*.

Halpern, J. Y. 1990. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350.

Han, B. and Lavie, A. 2004. A Framework for Resolution of Time in Natural Language. *TALIP Special Issue on Temporal Information Processing*, 2004.

Han, B., D. Gates, and L. Levin. 2006. Understanding Temporal Expressions in Emails. In *Proceedings of Human Language Technology conference - North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL).*

Harabagiu, S. and Bejan, C. 2005. Question Answering Based on Temporal Inference. In *Proceedings of the AAAI-2005 Workshop on Inference for Textual Question Answering*, Pittsburgh, PA, USA, 2005.

Haring, G., Juiz, C., Kurz, C., Puigjaner, R., and Zottl, J. 2004. Framework for the Performance Assessment of Architectural Options on Intelligent Distributed Applicatio. *Performance Metrics for Intelligent Systems (PerMIS '04)*, 2004.

Hayes, P. 1995. A Catalog of Temporal Theories. *Tech report UIUC-BI-AI-96-01*, University of Illinois 1995.

Hitzeman, J., M. Moens, and C. Grover. 1995. Algorithms for Analyzing the Temporal Structure of Discourse. In *Proceedings of EACL*. Dublin, Ireland.

Hobbs, J. R. 2000. Half Orders of Magnitude, *KR-2000 Workshop on Semantic Approximation, Granularity, and Vagueness*, Breckenridge, Colorado.

Hobbs, J. R. and Kreinovich, V. 2001. Optimal Choice of Granularity in Commonsense Estimation: Why Half Orders of Magnitude, In *Proceedings of Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, Vacouver, British Columbia, pp. 1343-1348.

Hobbs, J. R. 2002. Towards an Ontology for Time for the Semantic Web. In *Proceedings of LREC 2002 Workshop on Annotation Standards for Temporal Information in Natural Language*, pp. 28-35, Las Palmas, Spain.

Hobbs, J. R. and Pustejovsky, J. 2003. Annotating and reasoning about time and events. In *Proceedings of AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning,* Stanford, CA.

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

Hobbs, J. R. and Pan, F. 2006. Time Ontology in OWL. Ontology Engineering Patterns Task Force of the Semantic Web Best Practices and Deployment Working Group, *World Wide Web Consortium (W3C)*. (http://www.w3.org/TR/owl-time/)

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M. 2004. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C Member Submission*. http://www.w3.org/Submission/SWRL/

Hritcu, C. and Buraga, S. C. 2005. A reference implementation of ADF (Agent Developing Framework): semantic Web-based agent communication. In *Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC).*

Jurafsky, D. and Martin, J. 2000. *Speech and Language Processing*, Prentice Hall.

Kalman, J. A. 2001. *Automated Reasoning with Otter*, Rinton Press.

Kaykova, O., Khriyenko, O., Naumenko, A., Terziyan, V., Zharko, A. 2005. RSCDF: A Dynamic and Context-Sensitive Metadata Description Framework for Industrial Resources, In *Eastern-European Journal of Enterprise Technologies*, Vol. 3, No. 3.

Khushraj, D.; Lassila, O.; and Finin, T. 2004. Semantic Tuple Spaces, *Processing in Mobile Ad-Hoc Networks Proceedings of the International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, August 2004.

Krippendorf, K. 1980. *Content Analysis: An introduction to its methodology*. Sage Publications.

Lenat, D. 1995. "Cyc: A Large-Scale Investment in Knowledge Infrastructure," Communications of the ACM, 38, no. 11.

Liu, J., C. Zhou, M. Tang. 2006. Research on the Workday Time Management Model for Workflows in Business Service Grid Environment. In *Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops (GCCW).*

Madden, C. J. and Zwaan, R. A. 2003. How does verb aspect constrain event representations? *Memory & Cognition*, 31, 663-672.

Malhotra, A, Melton, J., and Walsh, N. 2005. XQuery 1.0 and XPath 2.0 Functions and Operators. *W3C Candidate Recommendation*. http://www.w3.org/TR/xpath-functions/

Mallya, A. U., Yolum, P., and Singh, M. P. 2004. Resolving commitments among autonomous agents. In F. Dignum, editor, Advances in Agent Communication, In *Proceedings of the International Workshop on Agent Communication Languages*, Germany.

Mani, I. and G. Wilson. 2000. Robust Temporal Processing of News. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mani, I., M. Verhagen, B. Wellner, C. M. Lee, and J. Pustejovsky. 2006. Machine Learning of Temporal Relations. In *Proceedings of COLING-ACL*.

McDermott, D. 1982. A temporal logic for reasoning about processes and actions, *Cognitive Science*, 6,101-155.

McGuinness, D. L. and Harmelen, F. v, 2004. OWL Web Ontology Language Overview. *World Wide Web Consortium (W3C)* Recommendation.

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

Medjahed, B., Bouguettaya, A. and Elmagarmid, A, 2003. Composing Web Services on the Semantic Web. *The Very Large Data Base Journal, Special Issue on the Semantic Web*,Springer Verlag, 12(4).

Miller, G. A. 1990. WordNet: an On-line Lexical Database. *International Journal of Lexicography* 3(4).

Moens, M. and Steedman, M. 1988. Temporal Ontology and Temporal Reference. *Computational Linguistics* 14(2): 15-28.

Moffitt, M. D. and Pollack, M. E. 2005. Applying Local Search to Disjunctive Temporal Problems, *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2005).*

Moldovan, D. and V. Rus. 2001. Logic Form Transformation of WordNet and its Applicability to Question Answering. In *Proceedings of ACL*.

Moldovan, D. and A. Novischi. 2002. Lexical Chains for Question Answering. In *Proceedings of COLING.*

Moldovan, D.; Harabagiu, S.; Girju, R.; Morarescu, P.; Lacatusu, F.; Novischi, A.; Badulescu, A.; Bolohan, O. 2002. LCC Tools for Question Answering. In *Proceedings of the TREC-2002 Conference*, NIST. Gaithersburg, MD.

Moldovan, D., C. Clark, and S. Harabagiu. 2005. Temporal Context Representation and Reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

Motakis, I. and C. Zaniolo, 1997. Temporal Aggregation in Active Databases Rules. In *International Conference on the Management of Data*, May.

Nevatia, R.; Hobbs, J. R.; Bolles, B. 2004. An Ontology for Video Event Representation. *IEEE Workshop on Event Detection and Recognition*, June 2004.

Niles, I. and Pease, A. 2001. Towards a standard upper ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001.

Qasemizadeh, B., Haghi, H. R., Kangavari, M. 2006. A Framework for Temporal Content Modeling of Video Data Using an Ontological Infrastructure. In *Proceedings of the Second International Conference on Semantics, Knowledge, and Grid (SKG).*

OWL-S Coalition, 2004. OWL-S 1.1 Release. (http://www.daml.org/services/owl-s/1.1/)

Pan, F. and Hobbs, J. R. 2003. A Time Zone Resource in OWL. Homepage at: http://www.isi.edu/~pan/timezonehomepage.html

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

Pan, F. 2005. A Temporal Aggregates Ontology in OWL for the Semantic Web. In *Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web*, Arlington, Virginia.

Pan, F. and Hobbs, J. R. 2005. Temporal Aggregates in OWL-Time. In *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, Clearwater Beach, Florida, pp. 560-565, AAAI Press.

Pan, F. 2005. Temporal Aggregates for Web Services on the Semantic Web. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, Orlando, Florida, pp. 831-832, IEEE Computer Society.

Pan, F., R. Mulkar, and J. R. Hobbs. 2006. Learning Event Durations from Event Descriptions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pp. 393-400, Sydney, Australia.

Pan, F., R. Mulkar, and J. R. Hobbs. 2006. Extending TimeML with Typical Durations of Events. In *Proceedings of Annotating and Reasoning about Time and Events workshop at the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pp. 38-45, Sydney, Australia.

Pan, F. and Hobbs, J. R. 2006. Temporal Arithmetic Mixing Months and Days. In *Proceedings of the 13th International Symposium on Temporal Representation and Reasoning (TIME)*, pp. 212-217, Budapest, Hungary, IEEE Computer Society.

Pan, F., R. Mulkar, and J. R. Hobbs. 2006. An Annotated Corpus of Typical Durations of Events. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pp. 77-82, Genoa, Italy.

Pan, F., R. Mulkar-Mehta, and J. R. Hobbs. 2007. Modeling and Learning Vague Event Durations for Temporal Reasoning. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI)*, Nectar Track, pp. 1659-1662, Vancouver, Canada.

Passonneau, R. J. 1988. A Computational Model of the Semantics of Tense and Aspect. *Computational Linguistics* 14:2.44-60.

Patel-Schneider, P. F. 2005. A Proposal for a SWRL Extension towards First-Order Logic, *W3C* member submission. http://www.w3.org/Submission/SWRL-FOL/

Pease, A. 2004. Standard Upper Ontology Knowledge Interchange Format. Available at http://cvs.sourceforge.net/viewcvs.py/*checkout*/sigmakee/sigma/suo-kif.pdf?rev=1.4

Prior, A. 1957. Time and Modality. *Oxford: Oxford University Press*.

Pustejovsky, J.; Gaizauskas, R.; Sauri, R.; Setzer, A.; Ingria, R. 2002. Annotation Guideline to TimeML 1.0, available at http://time2002.org.

Pustejovsky, J.; Hanks, P.; Saurí, R.; See, A; Gaizauskas, R.; Setzer, A.; Radev, D.; Sundheim, B.; Day, D.; Ferro, L.; and Lazo, M. 2003. The timebank corpus. In *Corpus Linguistics 2003*, Lancaster, U.K.

Rieger, C. J. 1974. Conceptual memory: A theory and computer program for processing and meaning content of natural language utterances. *Stanford AIM-233*.

Siegel, S. and Castellan, N. J. 1988. Jr. Nonparametric Statistics for the Behavioral Sciences. *McGraw-Hill*, second edition.

van der Sluijs, K., G. Houben , J. Broekstra , S. Casteleyn. 2006. Hera-S: web design using sesame, In *Proceedings of the 6th international conference on Web engineering*, Palo Alto, California, USA

Smith, C. 1991. *The Parameter of Aspect*, Kluwer Academic Press, Dordrecht.

Smith, C. 2005. Aspectual entities and tense in discourse. In P. Kempchinsky and R. Slabakova (eds.) *Aspectual Inquiries*. Kluwer, Dordrecht.

Steedman, M. 1997. Temporality. In J. van Benthem and A. ter Meulen, (eds.), *Handbook of Logic and Language*, Elsevier North Holland, 1997, 895-935.

Steedman, M. 2002. The Productions of Time, *Draft tutorial notes about temporal semantics*, Draft 4.1, July 2002.

Stonebraker, M. R. Extensibility. 1990. In M.R. Stonebraker, editor, *Readings in Database Systems*. Morgan Kaufman.

Toivonen, S.; Denker, G. 2004. The Impact of Context on the Trustworthiness of Communication: An Ontological Approach. *ISWC Workshop on Trust, Security, and Reputation on the Semantic Web*.

Tsamardinos, I. and Pollack, M. E. 2003. Efficient Solution Techniques for Disjunctive Temporal Reasoning Problems, *Artificial Intelligence*, 151(1-2):43-90.

Tuchinda, R.; Thakkar, S.; Gil, Y.; Deelman, E. 2004. Artemis: Integrating Scientific Data on the Grid, *Proceedings of Sixteenth Conference on Innovative Applications of Artificial Intelligence (IAAI)*, San Jose, California.

Vendler, Z. 1967. *Linguistics in Philosophy*, Ithaca, Cornell University Press.

Verkuyl, H. 1989. Aspectual classes and aspectual composition, *Linguistics and Philosophy*, 12, 39-94.

Weissenberg, N. and Gartmann, R. 2003. Ontology Architecture for Semantic GeoServices for Olympia 2008, In: Bernard, L., A. Sliwinski and C. Senkler (Eds). *Münsteraner GI-Tage, Münster*, 2003. IfGIprints 18. 267-283.

Welty, C. and R. Fikes. 2006. A reusable ontology for fluents in OWL. In *Proceedings of FOIS*. pp. 226-236. Baltimore.

Witten, I. H. and E. Frank. 2005. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco.

Wu, B., Z. Liu, R. George, and K. Shujaee. 2005. eWellness: Building a Smart Hospital by Leveraging RFID Networks. In *Proceedings of the 27th International Conference of IEEE Engineering in Medicine and Biology Society (EMBS2005).*

Zhou, Q. and Fikes, R. 2002. A Reusable Time Ontology. *Proceeding of the AAAI Workshop on Ontologies for the Semantic Web.*

Zhu, H.; Madnick, S.E.; Siegel, M.D. 2004. Effective Data Integration in the Presence of Temporal Semantic Conflicts, *Proceedings of 11th International Symposium on Temporal Representation and Reasoning (TIME 2004)*, pp109-114, Normandie, France.

Appendices

Appendix A. Representation and Proof Steps for the Scheduling Use Case

A use case on scheduling as shown on page 2:

> *Suppose someone has* *<u>weekly</u>* *telecons* *<u>on Mondays at 2pm EST in Spring 2007</u>*. *You would like to make an appointment with him for* *<u>10am PST on 03/05/2007</u>*, *and expect the meeting to last* *<u>45 minutes</u>*. *Will there be an* *<u>overlap</u>*?

The sentence-by-sentence representation of the use case in OWL-Time is as follows:

"weekly telecons on Mondays at 2pm EST in Spring 2007":

$$everyp(s,\{T\},Monday1) \wedge begins(t_1,T) \wedge dateOf(t_1,2007,1,8) \wedge ends(t_2,T) \wedge dateOf(t_2,2007,5,11) \wedge member(d,s) \wedge beginsOrIn(T_1,d) \wedge begins(t_3,T_1) \wedge timeOf(t_3,y,m,d,14,0,0,*EST*)$$

"10am PST on 03/05/2007":

$$timeOf(t_4,2007,3,5,10,0,0,*PST*)$$

"45 minutes":

$$ProperInterval(T_2) \wedge begins(t_4,T_2) \wedge duration(T_2,*Minute*) = 45$$

"overlap":

$nonoverlap(T_1,T_2)$[38]

$$\equiv [intBefore(T_1,T_2) \vee intAfter(T_1,T_2) \vee intMeets(T_1,T_2)$$

[38] The axiom is taken from (Hobbs and Pan, 2004).

$\vee\ intMetBy(T_1, T_2)]$

Where $(\forall\ d)\ [Monday1(d) \equiv (\exists\ w)\ [Monday(d,w)]]$

The proof steps of the problem:

$everyp(s,\{T\},Monday1) \wedge member(d,s)$

$\Rightarrow tseqp(s,Monday1) \wedge member(d, s)$

$\Rightarrow Monday1(d)$

$begins(t_1,T) \wedge dateOf(t_1,2007,1,8) \wedge ends(t_2,T) \wedge dateOf(t_2,2007,5,11)$

$\wedge\ timeOf(t_4,2007,3,5,10,0,0,*PST*)$

$\Rightarrow beginsOrIn(t_4,T)$

$timeOf(t_4,2007,3,5,10,0,0,*PST*)$

$\Rightarrow dateOf(t_4,2007,3,5) \wedge beginsOrIn(t_4, d_1) \wedge Monday1(d_1)$

$timeOf(t_4,2007,3,5,10,0,0,*PST*) \wedge ProperInterval(T_2) \wedge begins(t_4,T_2)$

$\wedge\ duration(T_2,*Minute*) = 45$

$\Rightarrow ends(t_5,T_2) \wedge timeOf(t_5,2007,3,5,13,45,0,*EST*)$

$ProperInterval(T_2) \wedge ends(t_5,T_2) \wedge timeOf(t_5,2007,3,5,13,45,0,*EST*)$

$\wedge\ timeOf(t_3,2007,3,5,14,0,0,*EST*) \wedge ProperInterval(T_3) \wedge begins(t_3,T_3)$

$\Rightarrow before(t_5,t_3)$

$\Rightarrow intBefore(T_2,T_3)$

$\Rightarrow nonoverlap(T_2,T_3)$

$\Rightarrow$ *"There will be no overlap!"*

Appendix B. Temporal Aggregates from the ACE Corpus

The following table includes all the 219 temporal aggregates extracted from the ACE corpus. Each of the temporal aggregate expressions along with their frequencies appeared in the corpus is shown in the table.

Expressions	Freq	Expressions	Freq
a-year	1	every year	11
365 days in the year	1	every year in Russia	1
370 days	1	everyday	1
Annual	2	five nights a week	1
Each November	1	five times a day	1
Every Sunday	1	four or five times a day	1
Every day	1	four consecutive Sundays	1
Every week that passes	1	hourly	2
Every year	1	hours daily	1
Monday	2	monthly	6
a day	4	nearly every day	2
a month	11	night after night	1
a week	9	once every three hundred thousand years	2
a year	11	once every decade	1
a-month	1	once every two	1
almost every night	3	per night	1
almost weekly	1	per week	1
an hour	3	practically every day	1
annual	30	recent days	1
annually	3	recent years	1
annuals	1	semiannual	1
daily	22	seven days a week	1
each July	1	several summers	1
each Thursday	1	six days a week	1
each day	3	the day	1
each of the last two seasons	2	the last several nights	1
each passing day	1	the past three summers	1
each year	8	the two half-days of deliberations	1

evening	1	three nights in a row	1
every day of the trip	1	three times a week	1
every year	1	three weeks	1
every December	1	twice a day	1
every day	5	twice a month	1
every good afternoon	1	twice a year	1
every hour	2	twice-monthly	1
every month	3	two weeks	1
every morning	2	weekends	1
every night	1	weekly	13
every several hundred thousand years	1	year after year	2
every six months	1	yearly	2

Here are the representations of some representative ones from the list above:

"*annual*" (similarly, "each year", "every day", "every December", "weekly", etc.)

$everyp(s,s_0,Year1)$, where $Year1(y) \equiv (\exists\, n,x)\ callInt(y,n,{*}Year{*},x)$

"*each passing day*" " (similarly, "Every week that passes", "each of the last two seasons", "the past three summers", etc.)

$everyp(s,s_0,Day1) \wedge last(t,s_0) \wedge ends(nowfn(D),t)$

where the function "nowfn" maps a document d into the instant or interval viewed as "now" from the point of view of that document, and D is the document this phrase occurs in.

"*four consecutive Sundays*" (similarly, "three nights in a row", "two weeks", etc.)

$everyp(s,s_0,Sunday1) \wedge card(s,4)$

"*every six months*"[39] (similarly, "once every three hundred thousand years", etc.)

$everyp(s,s_0,SixMonth)$, where $SixMonth(x) \equiv Interval(x) \wedge months(x) = 6$

[39] Alternatively, this (and other similar expressions) can be treated as a rate or frequency, which can be represented by the axioms provided by Hobbs at http://www.isi.edu/~hobbs/bgt-time.text