POLITECNICO DI MILANO

An OWL Ontology of Time and Events

FACOLTA DI INGEGNERIA

T2B INGEGNERIA INFORMATICA

Supervisor: Marco Colombetti
Student: Sun Jiaqi
Matricola: 737155

A.Y. 2011-2011

Statement of Authorship

Except where reference is made in the text of this thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a dissertation presented by me for another degree or diploma.

No other person's work has been used without due acknowledgement in the main text of the thesis.

This thesis has not been submitted for the award of any other degree or diploma in any other tertiary institution.

Name: Sun Jiaqi

Date: 2011/06/09

List of figures

List of Tables

Index

Abstract:

With fast development, Internet has become one of the most popular resources for information retrieval. Nevertheless, the categories of information on the Internet are so disparate that user constantly find it time consuming to retrieve precisely what they actually need. To solve the problem, the Semantic Web is emerging.

Ontology as the most important foundation of the Semantic Web, is used to describe the knowledge of a specific domain. The resources online can be defined explicitly with the use of ontology. Via the description of ontology, user can not only understand and access the resources on the internet, but also make computers and other terminals access or integrate the resources automatically.

Yet, the design and implementation of ontology is a time-consuming and monotonous work. As a result, to develop ontology automatically has becomes a critical issue.

In this study, a temporal ontology and an event ontology was designed and implemented with OWL, and E-commerce application was discussed as an example, by extending the interval temporal logic implemented by James Allen, July 1994 and the integration of existing event ontology on W3C. Experiments regarding different scenarios have been conducted and explained in later chapters.

Keyword: Ontology, Semantic Web, OWL, Temporal, Event.

Sommario

Con il suo rapido sviluppo, Internet è diventata una delle risorse più importanti per il recupero delle informazioni. Tuttavia, le categorie di informazioni su Internet sono così diverse che l'utente perde molto tempo nel recuperare esattamente ciò di cui ha effettivamente bisogno. Per risolvere il problema sta emergendo il Semantic Web.

L'ontologia, su cui si basa il Web Semantico, è usata per descrivere la conoscenza di un dominio specifico. Le risorse online possono essere definite esplicitamente attraverso l'ontologia. Attraverso la descrizione dell'ontologia, l'utente non solo può comprendere ed accedere alle risorse su Internet, ma anche rendere computer e altri terminali in grado di accedere alle risorse o di integrarle automaticamente.

Tuttavia, la progettazione e la realizzazione di ontologie è un lavoro lungo e monotono. Per questo automatizzare lo sviluppo di ontologie è diventato un problema cruciale.

In questo studio, una Ontologia temporale e una Ontologia dell'evento sono stati progettato e implementato con OWL, estendendo la "interval temporal logic" implementata da James Allen, luglio 1994, e l'integrazione con l'ontologia degli eventi presenti in W3C. Esperimenti riguardanti diversi scenari sono stati condotti e spiegati nei capitoli successivi.

Parole chiavi: Ontologia, Semantic Web, OWL, temporale, eventi.

Acknowledgement

I would like to express sincere thanks to my supervisor, Prof. Marco Colombetti, who helped me very much throughout this thesis. He gave me valuable advices on the thesis and provided a motivating atmosphere. Through his support and perseverance, I worked diligently and seriously on this thesis.

I would like to thank my friends, classmates and colleagues for their concern, support, encouragement and the stimulating research discussions. Lastly, I would like to thank my parents, without their spiritual support this thesis it not possible.

Chapter 1 Introduction

1.1 Goals

One of the most widely used functions of the Internet is the search engine that redirects the user to their interested web pages, with the introduce of Web 2.0, social elements have been added to the modern internet, for instance, sharing, subscribing, news feeds, publishing and various activities are supported by diverse websites such as YouTube, Face book, Flickr and many others. The Semantic Web network, which is also known as Web3.0 has embedded various information beneath the digital data, in this way, both the user and the computer itself could understand the semantics of a specific webpage.

However, the potential of the Semantic Web lays in automatic task performing base on the user preference and settings, which is expected to improve greatly the searching efficiency and precision comparing with manual searching. This feature of the Semantic Web could be fully applied by various professions from doctors, engineers, scientists, to musicians, designers, artists or anyone that needs to work with digital data on the web.

The goal of Semantic Web is to improve the automation and intelligence level of the internet, while Semantic Web technology involves a number of correlated technologies, among which theory research and application research are involved. The objective of this thesis is to design and implement an integrated ontology for event management application. This integrated ontology is composed of temporal ontology and event ontology that are able to handle time and event management. With the help of ontologies, business handlers could have their events and meets arranged in an automatic manner, the system is supposed to represent time conflicts and interrelations between diverse agents to the users.

1.2 Motivation

With rapid development of the Internet and its associated technology, web-based business or E-commerce has become more and more popular. However, the traditional "one-size-fits-all" approach in web-based business does not provide an effective outcome for business handlers. Most of the adaptive E-commerce systems that have been developed to provide personalized business management for business handlers according to the individual's need and preferences are developed in the closed world, which means that the business knowledge is arranged in a fixed format with advanced knowledge of the audience. However, this means that the E-commerce model reusability is limited. In the distributed knowledge database on the Internet, the business repositories can be vastly distributed across heterogeneous systems. In order to overcome this distributed business repositories concern, researches have been focused on the Semantic Web technology in which the business content can be reusable and interoperable in heterogeneous systems.

1.3 My Personal Contribution

My work behind this thesis is to implement a temporal ontology that could be applied to various scenarios. One application could be the E-commerce system with the help of temporal ontology and event ontology, the user of the system, for instance, a project manager could register conferences, create events or organize product delivery according to deadlines, while at the same time, the system itself is able to handle temporal conditions and constraints such as deadlines, time conflicts, or sequences of events and present the outcome to the user. In addition to this, the system could understand the relations between interactive events and find the relations of diverse agents through their roles in different sub-events, therefore the consistency of the overall event management will be preserved. This level of intelligence in the system could help multiple users to coordinate better their schedules so as to avoid overlapping of events, missing of deadlines or mismatching of event partners.

1.4 Structure of the thesis

The purpose of developing the temporal ontology and event ontology is to illustrate how OWL ontologies can help with web information management. In this work, first we will review the background and current situation of semantic web technology, then we try to investigate related problems and review current solutions. We will then explore how to make use of the temporal and event ontology for event management. An example of E-commerce will be discussed in the later chapters.

The structure of the thesis is organized as follows. Chapter 2 reviews background knowledge and related work. It introduces the current research situation on Semantic Web and OWL ontology. Chapter 3 discusses James Allen's temporal logic which will be used as the base of my temporal ontology. Chapter 4 and 5 describes the implementation of temporal ontology and event ontology separately, with detailed explanation about classes, relations, properties and assertions inserted, together with the help of various examples and graphs base on different scenarios. Chapter 6 uses E-commerce as an example for illustration how ontologies can help in real applications. Chapter 7 concludes the thesis and discusses future work that could be performed to further improve the ontology.

Chapter 2 Background and Review

This Chapter presents the background knowledge of my thesis focusing on introducing Semantic Web and ontology techniques, which are means for realizing intelligent web information retrievals and web applications such as E-commerce system. Literature review is as well covered in the chapter for reference purpose. Section 2.1 aims in explaining the research background regarding Semantic Web and ontology applications, both in theory and applications. Since the objective of this thesis is to develop a temporal ontology and an event ontology in order to study how they can be combined together for E-commerce application, section 2.2 is presented to explain how this could be achieved, in terms how Semantic Web could help with E-commerce application and how ontology can enhance the features of Semantic Web. Section2.2.7 is a brief introduction of the current available ontology editing tools, focusing on Protégé 4.1 which was used as the implementation tool of the ontologies in this thesis.

2.1 Research Background

2.1.1 Theory Research Background

The current theory research concerning Semantic Web technology mostly involves the development of ontology, Semantic Web language and the development of trust and proof.

The development of ontology includes ontology management, ontology adaption and ontology standardization, which are separately discussed in the following section:

- **Ontology Management**: the main goal of ontology is the sharing and recycled of knowledge, therefore an ontology system is expected to be open to storage, organization, symbolization and versioning. Open storage and

organization studies how the ontology stores and organizes the information for an easy access of information and its management. The Symbolization labels each ontology with a unique identification tag. Since most of the ontologies evolve through time, a version management structure is needed to maintain the consistency of the ontologies through different versions.

- **Ontology Adaption**: ontologies evolve through time, how to expand and update the current ontology has become an important issue. This includes searching, editing and induction in the ontology.

- **Ontology Standardization**: Integration and interaction are expected to be carried out on different ontologies, as a consequence, standardization is needed to realize this.

The goal of the Semantic Web is to realize interaction on the semantic level, in order to achieve this, formal languages used in the Semantic Web should follow a unified standard. XML and RDF are still under research, and the formal language used for ontology description is currently under focus of the research.

As an open distributed system, Semantic Web allows updates from anyone in the internet, which might cause to conflicts. Therefore one needs to verify the proof of source information and its tractability. Unfortunately there is no standard yet for proof and trust, a lot of work still needs to be done to integrate them in the Semantic Web system.

2.1.2 Application Research Background

The majority of the current applications of Semantic Web research are focusing on web-services, agent based distributed computing, semantic based search engine and semantic based digital library.

- Web-service is a series of standards and developing standards designed and specified by W3C, it is used for cross-platform program to provide interaction. Semantic web technology can improve the automatic level of

user tracing, selecting, integration and monitoring activities on Web-services.

- Agent based computing could benefit from semantic-web that uses ontologies to describe various online resources, therefore, the online knowledge will be represented in a structured, logical and semantic way, this will also change the ways users search, obtain and use online knowledge resources. On one side, Semantic Web is a distributed knowledge based network, agent can read and induct knowledge under the guidance of ontology; on the other side, Semantic Web is an integration of web-services that are based on ontologies, and this has become a dynamic media for user interactions.

- Another new application is the search engine base on Semantic Web technology. Most of the current search engine are key-word based, therefore the accuracy are decreased by the usage of synonyms and polysemies. Even though some researchers were trying to solve this problem by introducing new algorithms, still the performance is limited at text-access level. The invention of Semantic Web technology could better handle these problems stated above.

- In recent years, a huge quantity of various forms of digital media data have been found on the internet, the traditional multimedia search algorithm is based on the low level characteristics of multimedia data which made the online search of multimedia data more difficult. Semantic based search could utilize the high-level characteristics of multimedia resources and makes fully usage out of it. Semantic based digital library is one of the most important resources in Semantic Web.

2.2 Review on Semantic Web and Ontology

2.2.1 What is Semantic Web

Semantic Web is one of the hottest research and development topic in the Internet community. Its goal is to make the World Wide Web easier understandable by machines. The Semantic Web aims at providing semantic-based access to the Internet, and retrieve information from texts in addition to being used in many applications to explicitly declare the knowledge embedded in them . Semantic Web provides a common framework that allows data to be shared and recycled across applications, enterprises and community boundaries.

The World Wide Web contains enormous data and information. However, the large amount of information on the web is often scattered and unrelated. Internet users often find it difficult to locate their targets when there are numerous relevant information resources published on the Web. The diverse resources of information also inhibit the recycle of web data. The vision of the Semantic Web is to enable machines to interpret and process information in the World Wide Web in order to better use of the web content. The term Semantic Web encompasses efforts to build a new architecture that supports content with formal semantics. This enables automated agents to reason about the Web content and produces an intelligent response to unforeseeable situations.

In 2009, Tim Berners-Lee invented the hypertext system for the Internet which makes it possible to exchange and share information through the network, which has already greatly improved the development speed of internet. Nowadays, Internet has become the main communication tool, users can browse for information they concern and distribute their own information through the internet. However, with the rapid growth of the internet, the limitation of the current technology has been exposed. The core of current internet technology is hypertext system, which uses the URI for marking information online and realizes fast information retrieval. It does not describe or understand the meaning of the information though, but barely retrieve the information according

to the URL, while what the user really concern is the meaning of information, which refers to the meaning within the lines of texts and images. Due to the limitation of current internet technology, the automation and intelligence level of information processing on the internet is rather low, the high processing power of the computers have not yet been fully utilized.

Many Internet technology researchers are looking for new technologies to change this situation, among which, Semantic Web technology has received a high attention. Semantic web is also known as the next generation network that adds semantic meanings by expanding the current internet, and makes the computer able to interact with humans. In other words, various sources in Semantic Web are not only connected through tags, but also interrelate with each other in their semantic meanings in order to improve the computer capability to understand and process information automatically. Since the computer does not really "understand" the meaning, Semantic Web developers have to represent the information in an effective way and make a unified standard.

In the world XML conference 2000, Tim Berners-Lee has made the speech entitled "Semantic Web" to explain the concept of Semantic Web, and he also proposed the structure of the Semantic Web. With the purpose to help with the development of this new technology, W3C has formed a group to work on this topic. Currently, Semantic Web as a focus of information technology, it has brought great attention from researchers, governments, organizations and commercial departments, and it is expected to undergo a prompt growth in the coming years.

Semantic web has the following characteristics that differentiate it from web without semantic:

- Everything that could be identified(people, time, event, objects, things, etc) are involved in the network.
- Each entity has a uniform resource identifier (URI).
- Information is incomplete(the web is open, and so does Semantic Web, any information retrieved by search engine are merely part of all the relevant information online).
- Network is evolving.

- Minimum Design
    - Simplify the simple things, make complex things simple.
    - Start from simple applications, orient at the complex future applications.
    - It's not necessary to standardize everything.
    - Result is greater than the sum of all parts.

2.2.2 How could Semantic Web Help with E-Commerce

Many studies on web-based learning have been conducted recently, and a number of problems have been identified, which need to be studied in order to reach better results. The main problems faced by web-based business event management are [2]:

1. Few business resources can be recycled, thus it is effort and time-consuming in constructing new business models.
2. For any material to be recycled, it is necessary to know its location because it is not easy to run precise searches in the existing Web.
3. Recycle of information and interoperability between systems is restricted because there is a lack of semantic description of knowledge domain.
4. Recycle of adaptive techniques is limited because these techniques are strongly linked to the knowledge domain.

The Semantic Web promotes reusing and sharing of information through the use of ontologies. Thus, it is a promising research dimension to take the intelligent web-based education on new dimensions. The key property of the Semantic Web architecture seems to be powerful enough to satisfy the E-commerce requirement, which delivers fast, just-in-time and relevant learning. Semantically annotated business event materials make is easy to mix and match materials into a new business model. Semantic query and navigation through materials enables the business handlers to find useful material according to his preferences.

2.2.3 Review of Ontology

"An ontology is an explicit specification of a conceptualization" according to Thomas R. Gruber [4]. Currently, there have been great research interests in modeling the real world in order to build an application capable of handling real life applications. Since the real world is so complex that it is both time and effort consuming to represent each detail within, only a part of the real world or a domain is considered while developing an application. A domain is restricted to a subject or an area of real world knowledge, such as industry, mechanicals, commercial, physics, medicine, etc. Diverse methodologies were developed to summarize the domain knowledge. Ontology was one of the ways to represent domain concepts and their relationships. Ontology has become a hot topic in agent development society because of its capability in agent communication.

The concept of ontology was first brought up in philosophy, which refers to a systematic and objective explanation and specification. In computer science filed, ontology is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It could be seen as an unified understanding toward a specific domain, which has three layers of meanings: first, an ontology is about a specific domain; second, ontology is about the knowledge in this domain, the knowledge are represented by describing the relationships between concepts; last but not the least, ontology is a representation, by using the same ontology, people can communicate with each other while working on the same domain, which helps to avoid communication disorders by different understandings of concepts.

Currently, ontology has been used as a formal specification for a clear conceptualization in agent-based computing, distributed computer, expert system and information system. The user of ontology could make the following hypothesis possible:

- Providing background knowledge for re-writing or improving of query sentences
- Accessing the system with natural language

- Managing video or audio files
- Integrating and retrieving distributed, heterogeneous resources

Generally it is difficult to realize knowledge sharing due to the different methods of knowledge representation and encoding, developers have to start from beginning when construction a domain knowledge. However, thanks to the flexible, scalable and maintainable characteristics of ontology, the sharing between two domain knowledge (even heterogeneous) is made possible.

The main components of a practical classification system are sets of knowledge representations in terms of words and terms in a specific domain, relations and properties of each entity in this domain are identified by the ontology as well. Why is ontology needed to classify a specific domain? The answer is that, the result provided by the ontology could help the machines, in many cases, the computers, to understand better the knowledge structure, thus domain knowledge could be recycled and shared in other future applications.

The basic components of a practical classification system are listed below, which are also the components of the temporal ontology and event ontology explained in later chapters:

Class, also known as concepts, sets, collections, type of objects, or kinds of things

- Class name (*E.g. Interval, Event, Agent, Place, DateTimeDescription*)
- Definition, documentation
- Class type

Property, Slot, Role

- Name (*E.g. intervalBefore, hasSubEvent, Agent_In, hasDateTimeDescription…*)
- Type
- Constraints, Cardinality
- Domains/Facets

Rules are statements that describe the logical inferences that can be concluded from an assertion in a particular form.

(E.g. ***intervalMeets(?i, ?m), intervalMeets(?j, ?m) -> sameending(?i, ?j*** *is a rule for describing the statement "if two intervals both meet the same interval m, then they are said to have the same ending instant. " One real example with the calendar would be : both the month interval Dec 2009 and day interval 31 Dec 2009 meets Year 2010, then we could say that Dec 2009 and 31 Dec 2009 have the same ending instant, which is obvious to human but not machines.)*

Attributes are aspects, properties, features, characteristics, or parameters that objects (and classes) can have.

2.2.4 The use of Ontology in Semantic Web

Ontologies are specifications of the conceptualization and corresponding vocabulary used to describe a domain (Gruber, 1993), they are the solutions to the limitation of RDFS. They are appropriate for describing heterogeneous, distributed and semi-structured information resources online. By defining shared and universal domain theories, ontologies help both people and machines to communicate quickly. As a result, it is important that any semantic is based on a clearly specified ontology. By doing this, user and developer of the Semantic Web can reach a shared understanding by exchanging ontologies that provide the vocabulary needed for communication.

Usually ontologies are consisted by the following components:

- Definitions of concepts related for the domain
- Relations and axioms about the concepts and relationships
- Several representation languages and system [7].

The most updated language, OWL is developed to unified different ontology languages. It is a representation language for expressing web resources and supporting interpretation over those resources. OWL provides a rich set of construct for developing ontologies and to markup ontologies so that it can be machine readable and understandable [8]. In this thesis, the temporal ontology and event ontology were implemented with OWL.

2.2.5 The application of ontology

By representing the knowledge in a domain, ontology helped to unify the terms and concepts used and improved the distribution and recycle of knowledge. In the book "Ontologies: Principles, methods and applications" by Uschold [10], he summarized the usage of ontology, which is communication, inter-operability and systems engineering.

- **Communication**: Ontology provides common terms for human-to-human or organization-to-organization communications, by using the same vocabulary to decrease different understandings toward the same concepts, in order to avoid the communication barriers caused by different habits of personal expression.

- **Inter-operability**: The ontology does the translation and mapping work in between of different modeling algorithms, formats, languages and software tools, in order to realize the inter-operability and integration.

- **Systems engineering**: Ontology could provide the systems engineering with formal specification, re-usability, and reliability. It helps to identify the needs and standards of systems; the obtained ontology could be shared or recycled in other engineering projects; non-formal ontology can help the designer check the design of the system, while formal ontology makes the automatic and consistent checking possible, and this increase the reliability of the software.

2.2.6 Representation of ontology – OWL DL

According to the formality degree of ontology representations, they could be divided into the following four categories:

1. Non-formal: arbitrary description by natural language
2. Descriptions with strict and structured natural language

3. Formal: representation by formal language
4. Strict-Formal: use consistent and complete axiom system for representation

Different representation methods were brought up by researchers, the following sections is a briefing of the most commonly used methods, which are already applied in many existing systems.

Since RDF has limited expressive power, in a sense that it is limited to binary predicates, one proposed solution to this is the Web Ontology Language (OWL). OWL extends RDF/RDFS to make it easier to express semantics. In Feb 2004, OWL has become a W3C recommendation.

Before the W3C Web Ontology Working Group defined OWL, a number of researchers joined together to define a markup language called DAML+OIL [25]. The DAML+OIL project proposed a standardized and broadly accepted ontology language for the Semantic Web. W3c found that it was necessary to propose a language to further extend the expressivity of RDF and RDFS. This led to DAML+OIL being superseded by OWL.

OWL is an ontology language for the web, it defines Web ontologies and their associated knowledge bases. It is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related [28]. The basic elements contained in an OWL ontology are sets of definitions of classes, object/datatype properties, relations between classes, class and property instances, and assertions which are inserted as knowledge base.

The OWL ontology works base on the open world assumption, and it has a reasonable trade-off between expressiveness and scalability. It includes the following features [27]:

- Fragment of first-order predicate logic
- Decidable
- Known complexity classes
- Reasonably efficient for real KBs

OWL allows resources to be described in a machine-accessible way. OWL is built upon RDF and RDFS. In OWL, instances are defined by using RDF

descriptions and most RDFS modeling primitives are reserved in OWL. OWL supports machine reasoning by using predicate logic and description logic [26]. There are three different sub-languages in OWL: OWL Lite, OWL DL and OWL Full. OWL Full is the superset of OWL Lite and OWL DL. OWL DL is based on description logic and its subset OWL Lite is based on the less expressive logic. Each of these sublanguages is a syntactic extension of its simpler predecessor. The following graph shows the relations of OWL Lite, OWL DL and OWL FULL:

OWL DL is used in this thesis, it supports users' basic need for hierarchy classification and simple constraints. OWL DL maximizes expressiveness while retaining computational completeness. OWL Full maximizes expressiveness but it may increase the computational complexity. Therefore, in this thesis OWL DL is chose for the E-commerce example.

OWL DL ontologies talk about worlds that contain individuals, classes (also noted as concepts) and properties (also known as roles). In the E-commerce example in this thesis, individuals may include specific agents such as ***Jack*** and ***Elwood***, or specific events such as ***Buy01*** and ***Sell01*** (as sub-events), classes (or concepts) may refer to ***interactiveEvent*** or ***simpleEvent***, which are classes defined in the example, properties in this example may refer to ***intervalBefore*** as temporal property or ***isAgent_In(agent_x, event_y)*** as an event property.

OWL ontologies are generally RDF documents, therefore the root element of a OWL ontology is an rdf:RDF element which also specifies a number of namespace[30].

The following OWL ontology example is extracted from the event ontology implemented in the thesis, ***Transaction01*** is an instance of class ***InteractiveEvent***, the following format shows how the instance of a class is initiated, and also, how it inherits the properties of its class.

Class membership : InteractiveEvent(Transaction01)

<InteractiveEvent rdf:about= "Transaction01"/>

rule version: -> InteractiveEvent(Transaction01)

The following is another example in the event ontology, which is for showing the scenario where ***Jack*** is an instance of class ***Agent,*** and he participates in event ***Sell01*** which is an instance of ***Event*** class, the relation between the two instances is noted as "***IsAgent_In***".

property membership : isAgent_In(Jack, Sell01)

<rdf: Description rdf: about= "Jack">

<isAgent_In rdf: resource = "Sell01" />

</rdf: Description>

rule version: -> isAgent_In(Jack, Sell01)

Information about how classes and properties relate in general could be represented as the following example, in which it is stated that class ***InteractiveEvent*** is a subclass of ***Event***:

subclass: InteractiveEvent ∈Event

<owl: Class rdf:about = "InteractiveEvent">

<rdfs:subClassOf rdf: resource = "Event" />

</owl: Class>

Rule version: InteractiveEvent(e) -> Event(e)

Information about how properties correlate in general could be represented as the following example, in which it is stated that property ***hasEventStarter*** is a subProperty of "***hasAgent***" . With the rule ***hasEventStarter(event, agent) -> hasAgent(event, agent)*** , it can be concluded that, if a certain agent is the event starter of a specific event, then he/she must be an agent participating in this event:

subProperty: hasEventStarter(event ,agent) ⇒ hasAgent(event ,agent)

<owl: objectProperty rdf: about = "hasEventStarter">

<rdfs: SubPropertyOf rdf: resource = "hasAgent" />

</owl: ObjectProperty>

Rule version: hasEventStarter(event, agent) -> hasAgent(event, agent)

OWL can build new classes from class, property and individual names, in the example in this thesis, ***Event*** is the union of two classes called ***SimpleEvent*** and ***InteractiveEvent***, represented as:

union: SimpleEvent InteractiveEvent

<owl: unionOf rdf:parseType = "Event">

<owl: Class rdf: about = "SimpleEvent" />

<owl: Class rdf: about = "InteractiveEvent"/>

</owl: unionOf>

OWL could also build intersections of two classes, in the example of event ontology, the intersection of two classes ***InteractiveEvent*** and ***Event*** is ***InteractiveEvent*** itself, since it is also a subclass of ***Event***.

intersection: Event InteractiveEvent

<owl: intersectionOf rdf: parseType = "InteractiveEvent" >

<owl: Class rdf: about = "Event" />

<owl: Class rdf: about = "InteractiveEvent"/>

</owl: intersectionOf>

OWL is able to represent disjoint relation between two classes, in the temporal ontology in this thesis, *Instant* and *Interval* are two disjoint classes belonging to *TemporalEntity* class, it is represented as:

```
<owl:Class rdf:about="#Interval">
  <rdfs:subClassOf rdf:resource="#ProperInterval"/>
    <owl:disjointWith rdf:resource="#Instant"/>
</owl:Class>
```

OWL DL is also used to infer the existence of a chain property, there are several chain properties defined in the temporal ontology example, the following shows the syntax of one of the chain properties that describes the relation between intervalMeet and intervalBefore:

```
intervalMeet o intervalMeet => intervalBefore
rule version: intervalMeet(j, k) ^ intervalMeet(k, l) -> intervalBefore(j, l)
<rdf: Description rdf:about = “intervalBefore”>
<owl: propertyChainAxiom rdf:parseType = “properInterval”>
   <owl: ObjectProperty rdf:about = “intervalMeet” />
   <owl: ObjectProperty rdf:about = “intervalMeet” />
   </owl: propertyChainAxiom>
</rdf: Description>
```

In OWL DL, a property can be the inverse property of another, for instance, *intervalBefore* is the inverse property of *intervalAfter* in the temporal ontology, it is noted as:

```
intervalBefore ≡ intervalAfter  −
Rule version:  intervalBefore (j, k) -> intervalAfter(k, j);
                intervalMeet(m, n) -> intervalMetBy(n,m)
```

Given a datatype map and a vocabulary over datatype map, an interpretation of OWL 2 for datatype map and vocabulary is a 9-tuple with the following structure [34]:

1. Object Domain
2. Data Domain
3. Class Interpretation Function
4. Object Property Interpretation Function (assigned to each object property)
5. Data Property Interpretation Function (assigns to each data property)
6. Individual Interpretation Function
7. Datatype Interpretation Function
8. Literal Interpretation Function
9. Facet Interpretation Function

With OWL one can also construct special class, three more common special classes are stated below as examples, however more special classes could be constructed.

- **Top class : ⊤**

  Top class refer to the class that contains all individuals of the domain, in this example, Thing is the top class.

  owl: Thing

- **Bottom class: ⊥**

  Bottom class is the “empty” class that contains no individuals.

  owl: Nothing

- **Universal property : U**

  This is the property that links every individual to every individual

  owl: topObjectProperty

In the temporal ontology designed in this thesis, the Top class is Thing, and it is the super class of all the classes including ***TemporalEntity***, ***DateTimeDescription*** and ***DurationDescription***, while ***topObjectProperty*** is the property that links all the instances in the temporal and event ontology.

We could illustrate the use of OWL on an example ontology "A business Event has an Agent as its participator; business Event is disjoint with Agent; SimpleEvent is an Event that is not InteractiveEvent; isSubEvent is a transitive property; isSubEvent is inverse of hasSubEvent". This example can be expressed in the syntax below:

$$Event \sqsubseteq \exists hasAgent.Agent$$
$$Event \cap Agent \equiv \bot$$
$$SimpleEvent \equiv Event \cap \neg InteractiveEvent$$
$$Tr(isSubEvent)$$
$$isSubEvent \equiv hasSubEvent^-$$

2.2.7 Ontology Management System and Editing Tools

Protégé is an ontology tool to help developers to construct domain ontology, such as defining concepts, predicates, properties and agent actions [11]. Due to its well defined API structure, providing library access, supporting third party plug-in and large user communities, Protégé is chosen as my ontology editor environment. With appropriate plug-in, the tool can generate Java class to support the definition of ontology. However, protégé does not keep track of ontology change.

Apart from protégé, there are many other management tools for developing ontology, such as OntoWeb, which provides a service description language DAML-S and a DASD (DAML agents for Service Discovery) to bind web service with client agents.

Semantic Network Ontology Base (SNOBASE) is another framework that provides a programming interface to access the ontology [19]. Its functionalities include loading ontologies from files, loading ontologies via the internet and locally creating, modifying, querying, and storing ontologies. However, there is no direct linking between ontology and an application.

In addition to this, Chimaera is a software system that supports users in creating and maintaining distributed ontologies on the web [20]. Two major functions it supports are merging multiple ontologies together and diagnosing individual or multiple ontologies. It is a knowledge management tool to build ontology for experts in certain domains. Again, it is not directly linked to any implementation services using the ontology.

OntoManager tries to formalize the ontologies and transform them into a more expressive representation, using RDF, DAML+OIL, OWL. It provides an interactive tool to semi-automate the detection and the resolution of various ontological mismatches in a workbench environment [21]. The following figure shows the conceptual architecture of the ontology management system according to the MAPE model.

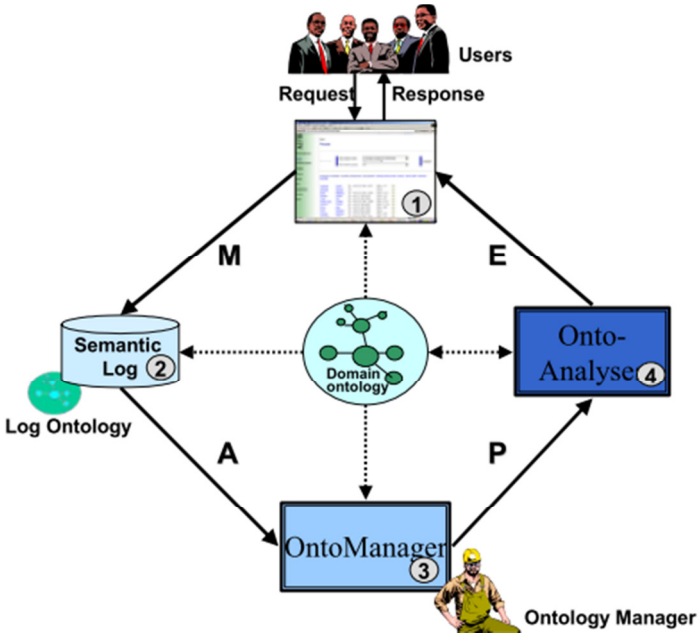

Figure 1 Conceptual Architecture of the ontology management system

In short, most of the current ontology management systems are mainly for ontology storage, knowledge management and Semantic Web. The links of ontology with agent development is not addressed.

Currently, most of the ontology editors only provide basic functions to construct ontologies. The agent construction and more comprehensive ontology construction functionalities, such as undo function, are not available in these editors. An ontology editor is suggested to have the following features:

- Supporting evolutionary changes
- Enabling users to resolve a change
- Providing control over evolution changes
- Declaring ways to undo previous change effects
- Managing ontology change history
- Providing support for continuous ontology improvement
- Finding inconsistencies and reasons for easy ontology management

Protégé allows user to define generic class and class hierarchy, properties and property value restrictions, relationships between classes and properties of these relationships. Users can create basic elements as classes in Protégé and create subclasses that inherit these basic element classes. For example, the class instant and interval are subclasses of TemporalEntity class.

Chapter 3 James Allen's Temporal Logic

Since the temporal ontology in this thesis is an extension base on James Allen's temporal logic, this chapter is dedicated in explaining and reporting the whole Allen's temporal logic and explaining how it can be extended to suit the event management example.

In July 1994, James Allen and George Ferguson from the University of Rochester has published "Actions and Events in Interval Temporal Logic" which aims in "present a representation of events and actions based on interval temporal logic that is significantly more expressive and more natural than most previous AI approaches"[5].

The representation of action and time has to support the following requirements:

- **Prediction**: Predict the possible consequence or outcome according to a predefined description of scenarios.

  Example: By extending James Allen's logic, in the event ontology implemented in the thesis, given an initial description that two agents Jack and Elwood participate separately in event sell and event buy, and these two simple events are sub_events of the same transaction, it can be predicted that Jack and Elwood are going to interact each other, and these two events are (likely) going to happen at the same geographical point.

- **Planning**: Plan the procedure to achieve the goal base on the already defined knowledge base and the target goal.

- **Explanation**: Given a set of observations about the world, find the best explanation of the data. When the observation are another agent's actions and the explanation desired is the agent's plan, and the problem is called plan recognition [29].

In the temporal logic proposed by James Allen, the top class for representing time is called *TemporalThing*, which has *TemporalEntity* as a subclass, and TemporalEntity has only two subclasses *Instant* and *Interval*. It is noted as:

: Instant
a owl:Class;
:Interval
a owl:Class;
:TemporalEntity
a owl:Class;
rdfs: subClassOf: TemporalThing;
owl: equivalentClass
[a owl:Class;
owl: unionOf (:Instant :Interval)
] .

Intervals are time periods in between of two instant points, usually intervals have with positive time length, time interval with zero length is still considered to be an interval since it has two instants that overlap with each other. Instants are individual points on the time line which by themselves do not have any length assigned.

An interval is associated to two instants with functions called *hasBeginning* and *hasEnd*. *Inside* is relation for representing a instant locates inside a certain time interval. The specification of these functions are as follows:

:hasBeginning
a owl: ObjectProperty;
rdfs: domain: TemporalEntity;
rdfs: range : Instant .

:hasEnd
a owl: ObjectProperty;
rdfs: domain: TemporalEntity;
rdfs: range : Instant .

:inside
a owl: ObjectProperty;
rdfs: domain: Instant;
rdfs: range : TemporalEntity .

Therefore the above temporal relation could be represented as the following graph:

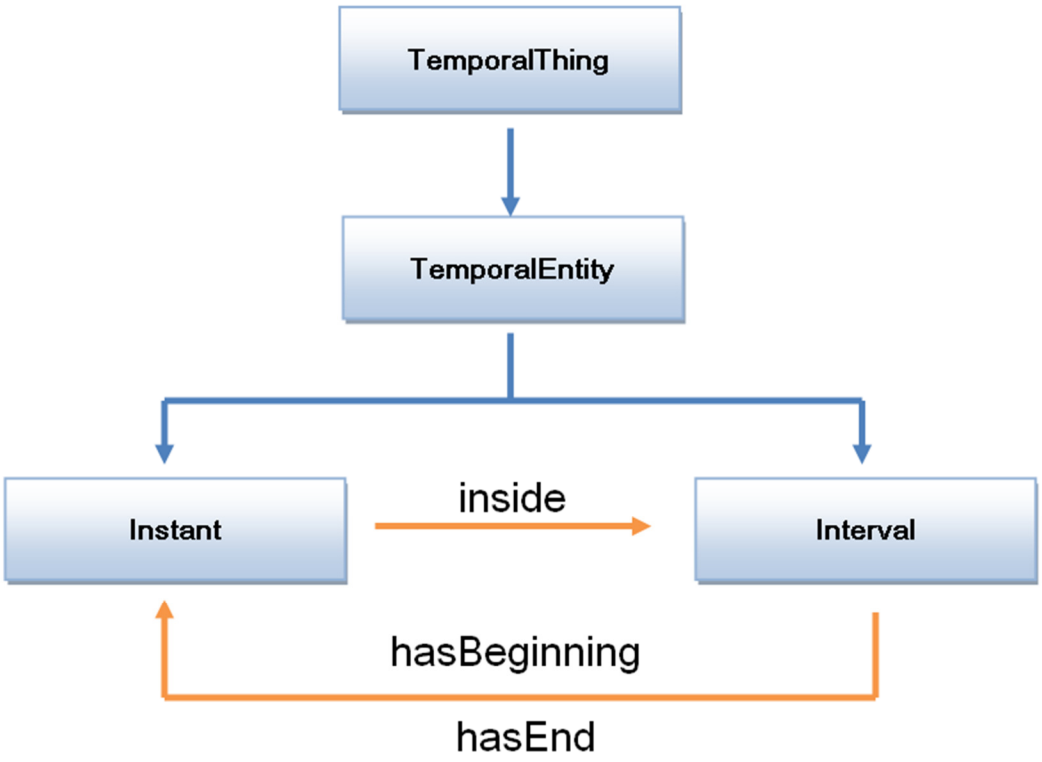

Figure 2 Temporal relation proposed in James Allen's temporal logic

A class called ***DurationDescription*** has been used in James Allen's temporal logic for assigning months, weeks, days, hours, minutes and seconds to ***TemporalThing***. It is useful for representing intervals with time arguments, for instance, an interval may have duration as 2 days , or 2 hour and 30 mins, or 1 sec. ***DurationDescription*** has a number of functions relating the durations to the values of the eight arguments (year, month, week, day, hour, minute, second), the syntax of the class is OWL is:

:DurationDescription

```
a          owl: Class;
rdfs :subClassOf
           [a        owl: Restriction ;
            owl: maxCardinality 1;
            owl: onProperty : seconds
           ];
```

```
rdfs :subClassOf
        [a      owl: Restriction ;
         owl: maxCardinality 1;
         owl: onProperty : minutes
        ];

rdfs :subClassOf
        [a      owl: Restriction ;
         owl: maxCardinality 1;
         owl: onProperty : hours
        ];

rdfs :subClassOf
        [a      owl: Restriction ;
          owl: maxCardinality 1;
          owl: onProperty : days
        ];

rdfs :subClassOf
        [a      owl: Restriction ;
         owl: maxCardinality 1;
         owl: onProperty : weeks
        ];

rdfs :subClassOf
        [a      owl: Restriction ;
         owl: maxCardinality 1;
         owl: onProperty : months
        ];

rdfs :subClassOf
        [a      owl: Restriction ;
         owl: maxCardinality 1;
         owl: onProperty : years
        ];
```

An interval can have multiple duration description, for example: 2 months, 3 days and 5 hours. But an interval can have only one duration. The interval above could be represented as :

```
:duration
    a       :DurationDescription ;
    : hours 5 ;
```

: days : 3
: months: 2 .

The *DurationDescription* class is assigned to *TemporalEntity* with relation *hasDurationDescription*:

:hasDurationDescription
a owl: ObjectProperty;
rdfs: domain: TemporalEntity;
rdfs: range: DurationDescription;

Another class called *DateTimeDescription* is used for specifying exact time to intervals that have positive time length, which is noted as *DateTimeInterval*. For example, an interval may have a description as "*Apr 20th*" or " *Friday 9.30 AM*". *DateTimeDescription* is defined in OWL as :

:DateTimeDescription
a owl: Class;
rdfs:subClassOf
[a owl: Restriction ;
owl: cardinality 1;
owl: onProperty :unitType
];
rdfs:subClassOf
[a owl: Restriction ;
owl: cardinality 1;
owl: onProperty :second
];
rdfs:subClassOf
[a owl: Restriction ;
owl: cardinality 1;
owl: onProperty :minute
];
rdfs:subClassOf
[a owl: Restriction ;
owl: cardinality 1;
owl: onProperty :hour
];
rdfs:subClassOf
[a owl: Restriction ;
owl: cardinality 1;
owl: onProperty :day

```
        ];
    rdfs:subClassOf
        [a      owl: Restriction ;
         owl: cardinality 1;
         owl: onProperty :dayofweek
        ];
    rdfs:subClassOf
        [a      owl: Restriction ;
         owl: cardinality 1;
         owl: onProperty :dayofyear
        ];
    rdfs:subClassOf
        [a      owl: Restriction ;
         owl: cardinality 1;
         owl: onProperty :month
        ];
    rdfs:subClassOf
        [a      owl: Restriction ;
         owl: cardinality 1;
         owl: onProperty :year
        ];
    rdfs:subClassOf
        [a      owl: Restriction ;
         owl: cardinality 1;
         owl: onProperty :timeZone
        ].
```

The function hasDateTimeDescription is used for assigning a ***DateTimeDescription*** to a ***DateTimeInterval***:

:hasDateTimeDescription

```
    a       owl:ObjectProperty
    rdfs: domain: DateTimeInterval;
    rdfs: range: DateTimeDescription .
```

In James Allen's temporal logic, certain basic axioms for representing temporal properties have been defined, they are represented in the graph below:

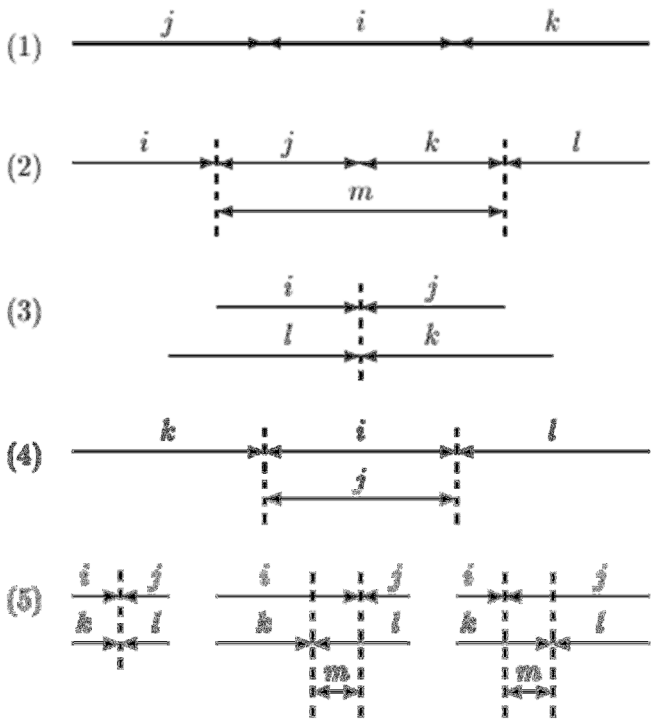

Figure 3 Procedure for Ontology Implementation

This graph(1) defines the relation Meet, which indicated that two periods m and n meet if and only if m precedes n, yet there is no time interval between m and n, and m does not overlap with n. the axiom is represented as:

$$\forall i \,.\, \exists j, k \,.\, Meets(j, i) \wedge Meets(i, k).$$

In this axiom, i,j,k are logical variables and represent time intervals. For any interval i that cannot be infinite, it has to meet and be meet by other intervals. In this case, interval i is meet by interval j, and it meets interval k, while both j and k are finite intervals. Note that, there is no beginning nor ending of time so the time line is infinite on both directions.

Graph(2) defines the concatenation of two intervals, for any two intervals that meet, there is another interval that is equal to the concatenation of them. The axiom is represented as:

$$\forall i, j, k, l \,.\, Meets(i, j) \wedge Meets(j, k) \wedge Meets(k, l) \supset \exists m \,.\, Meets(i, m) \wedge Meets(m, l).$$

In this case, for any interval i, j, k, l, if i meets j, j meets k, k meets l, there exists an interval m that is equal to j+k, here the plus sign refers to the concatenation of two intervals.

In graph (3), periods uniquely define an equivalent class of periods that meet them. The axiom is represented as follows:

$$\forall i, j, k, l \,.\, Meets(i, j) \wedge Meets(i, k) \wedge Meets(l, j) \supset Meets(l, k).$$

In particular, if i meets j and k, and l meets j, it is inferred that l also meets k.

Graph (4) indicates that equivalent classes uniquely define the periods.

$$\forall i, j, k, l \,.\, Meets(k, i) \wedge Meets(k, j) \wedge Meets(i, l) \wedge Meets(j, l) \supset i = j.$$

In particular, if two intervals are meet by the same interval, and they both meet another interval, they are considered to be equivalent. In this case, interval i and j are both meet by interval k, and both meet interval l, it is concluded that interval i and j are equivalent, since they have the same starting and ending points.

Finally, in graph (5), when two "meet" events occur, they either occur at the same time, or one has to precede the other.

$$\begin{aligned}\forall i, j, k, l \,.\, (Meets(i, j) \wedge Meets(k, l)) \supset \\ Meets(i, l) \otimes (\exists m \,.\, Meets(k, m) \wedge Meets(m, j)) \otimes \\ (\exists m \,.\, Meets(i, m) \vee Meets(m, l)).\end{aligned}$$

In the first sub graph of (5), i meets j and k meets l, these two "meet" occur at the same time, so they're said to be simultaneous. In the second sub graph, the "moment" when i meets j is after the "moment" when k meets l, the time difference between the two "moments" is represented as interval m, which has a positive length, and vice versa in the last sub graph.

In addition to the axioms stated above, many properties are intuitively assumed but not yet defined. For instance, no interval can meet itself, that is, "meet"

cannot be symmetric, in other words, finite circular models of time are not allowed.

To extend James Allen's temporal logic, more complete range of the intuitive relationships was defined. For example, the following axiom defines the relation "Before".

$$Before(i, j) \equiv \exists m \,.\, Meets(i, m) \wedge Meets(m, j).$$

One period is before another period if there exists another period that spans the time between them. In this case, there is an interval spam m that is met by interval i and meets interval j, therefore it is referred i is before j. As a relation, "before" is neither symmetric nor reflexive, but transitive, there means, if an interval i is before interval j, and j is before interval k, naturally i is before k. The inverse relation of Before is defined as "After", represented as After(j, i) given Before(i, j).

The following figure shows the extended relations. Inverse relations of each relation defined above are listed on the right side of the graph.

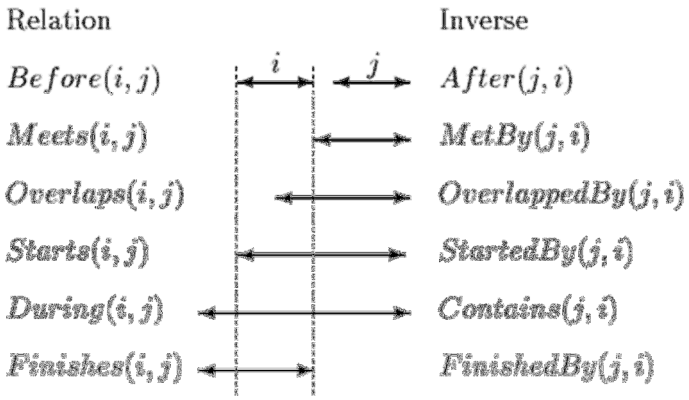

Figure 4 Relations and their inverse according to James Allen

The symbols used for representing each relation are listed below:

$$\begin{array}{llll} Meets(i,j) & i:j & & \\ Before(i,j) & i \prec j & Before(i,j) \vee Meets(i,j) & i \prec: j \\ During(i,j) & i \sqsubset j & During(i,j) \vee i = j & i \sqsubseteq j \end{array}$$

Finally, a relation called “disjoint” is defined as: two intervals are disjoint if they do not overlap in any way. It is written as $i \bowtie j$ and defined as $i \bowtie j \equiv i \prec: j \vee j \prec: i.$, that is, either interval i is before j or vice versa.

Chapter 4 Temporal ontology

This chapter aims in explaining the temporal ontology implemented base on James Allen's temporal logic. Ontology design and descriptions about the classes, properties and rules will be explained with examples in the following sections.

4.1 Tool used

In this thesis, protégé 4.1 Beta has been used for the implementation and extension of the ontology. The downloading address and configuration guide could be found at http://protege.stanford.edu/

Hermit OWL Reasoner is used for ontologies written using the Web Ontology Language (OWL). It can determine the consistency of an ontology, and identify subsumption relations between classes. The free downloading package and guide of Hermit can be found at http://www.hermit-reasoner.com/

Pellet Reasoner Plug-in is also used for Protégé 4 ,it is an environment for both commercial and academic users. Pellet 2 is available in protégé 4, including its unique capabilities: datatype reasoning, SWRL support, etc. The downloading package and installation guide could be found at http://clarkparsia.com/pellet/protege/

4.2 Temporal Ontology

A new ontology can be constructed from the ground or recycle the existing ontology. Ontology recycle can be divided into two categories: integration and merging. Integration means assembling, extending, specializing and adapting ontologies from different subjects or domains into a new ontology. The resulting ontology is often customized for a specific application. Merging combines different ontologies from the same domain or related domains into a common

ontology. Since organizations and industry in the same domain may construct their ontology similarly, merging can update the definition of concepts to the industry standard and attain consensus ontology. In this thesis, the ontology is composed of two parts, temporal ontology and event ontology. The temporal ontology is created from ground while the event ontology is extended and edited from an existing ontology libraries, afterwards these two ontologies are merged together for providing services for event management.

In order to implement an ontology that could be applied to a specific system, the following procedures should be followed:

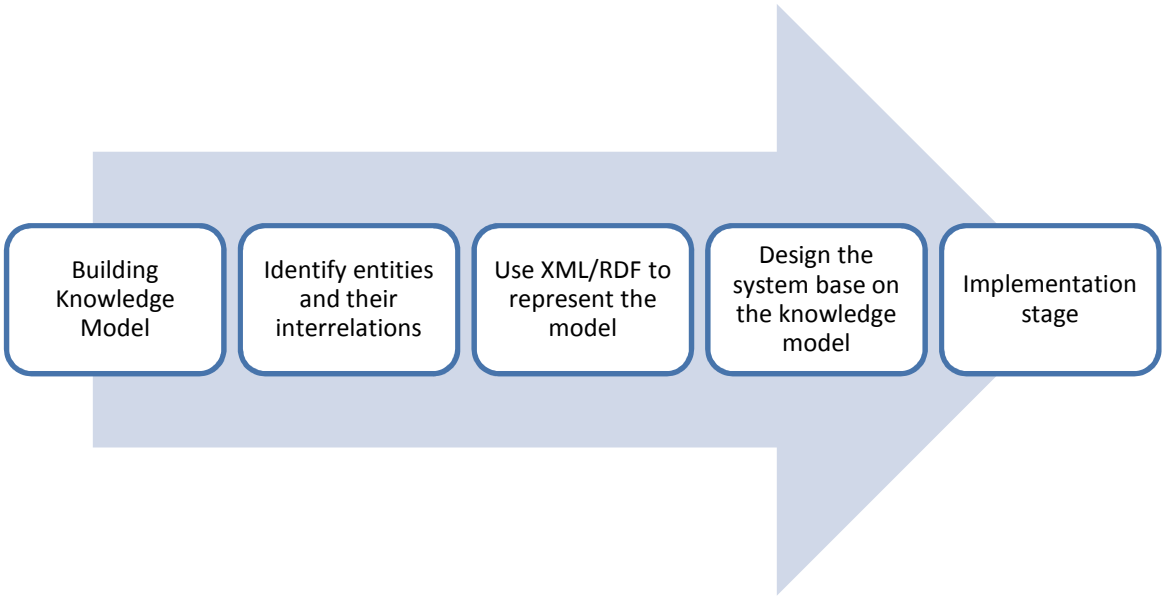

Figure 5 Procedure for Ontology Implementation

4.3 Classes in Temporal Ontology

This section states the classes created in the temporal ontology, which is represented in figure 7 below:

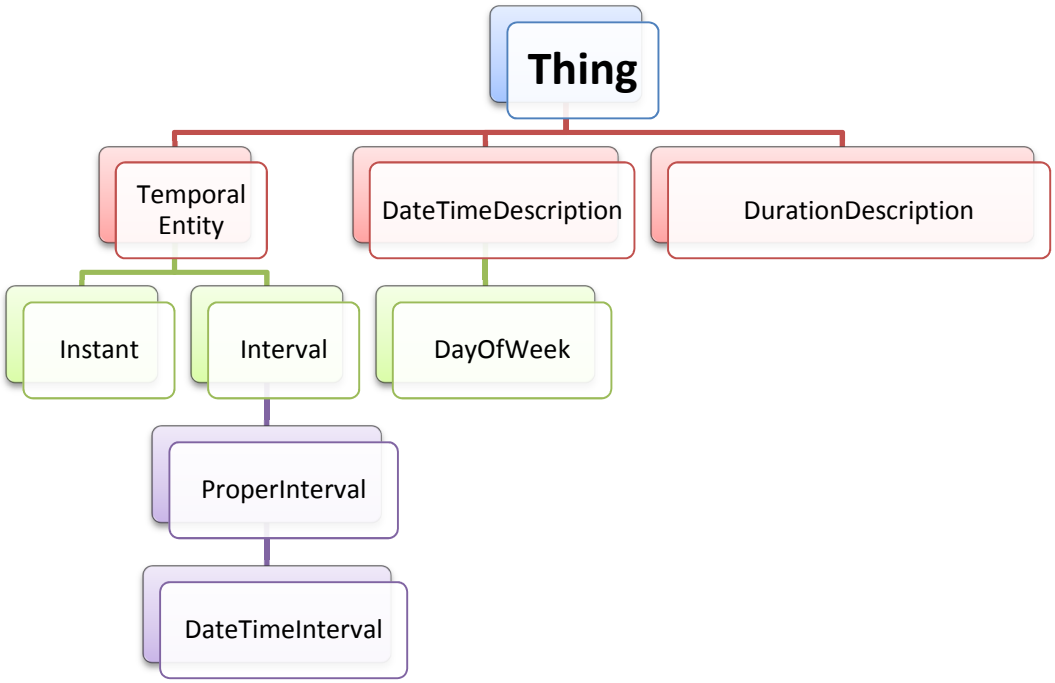

Figure 6 Temporal ontology classes

The descriptions of each class are explained below:

Thing

(Top class for everything)

- DateTimeDescription

  *(Class for representing date in the form of "**Jan2009**" or "**Apr2010**")*

  - Jan

    *(January as a class could have members as **Jan2009**, **Jan2010**…)*

  - Feb
  - Mar
  - Apr
  - May
  - Jun
  - Jul
  - Aug
  - Sep
  - Oct
  - Nov
  - Dec

- DayOfWeek
  (This class has members: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday, which are not subclasses but members)

- DurationDescription

        - Year
          *(Year as a class could have members as **2009**, **2010**…)*

- TemporalEntity
  (Both instant and interval are considered to be TemporalEntity, this is a class for all the temporal related objects)
    - Instant
      (Instant is a special interval that has time span equal to zero, it is considered to be a point in the time line.)
    - Interval
      (Interval is the distance between two different time instants, it has a positive time span, in this thesis, only positive time intervals are considered. Intervals are finite on both directions, every interval has a starting instant and an ending instant.)
        - ProperInterval
            - DateTimeInterval

- TemporalUnit
- TimeZone
  (This class has different time zones as members of the class. For each dateTimeDescription instant, a timezone information is assigned to avoid misunderstanding of event time.)

4.4 Object Properties of Temporal Ontology

This section states all the object properties in the temporal ontology :

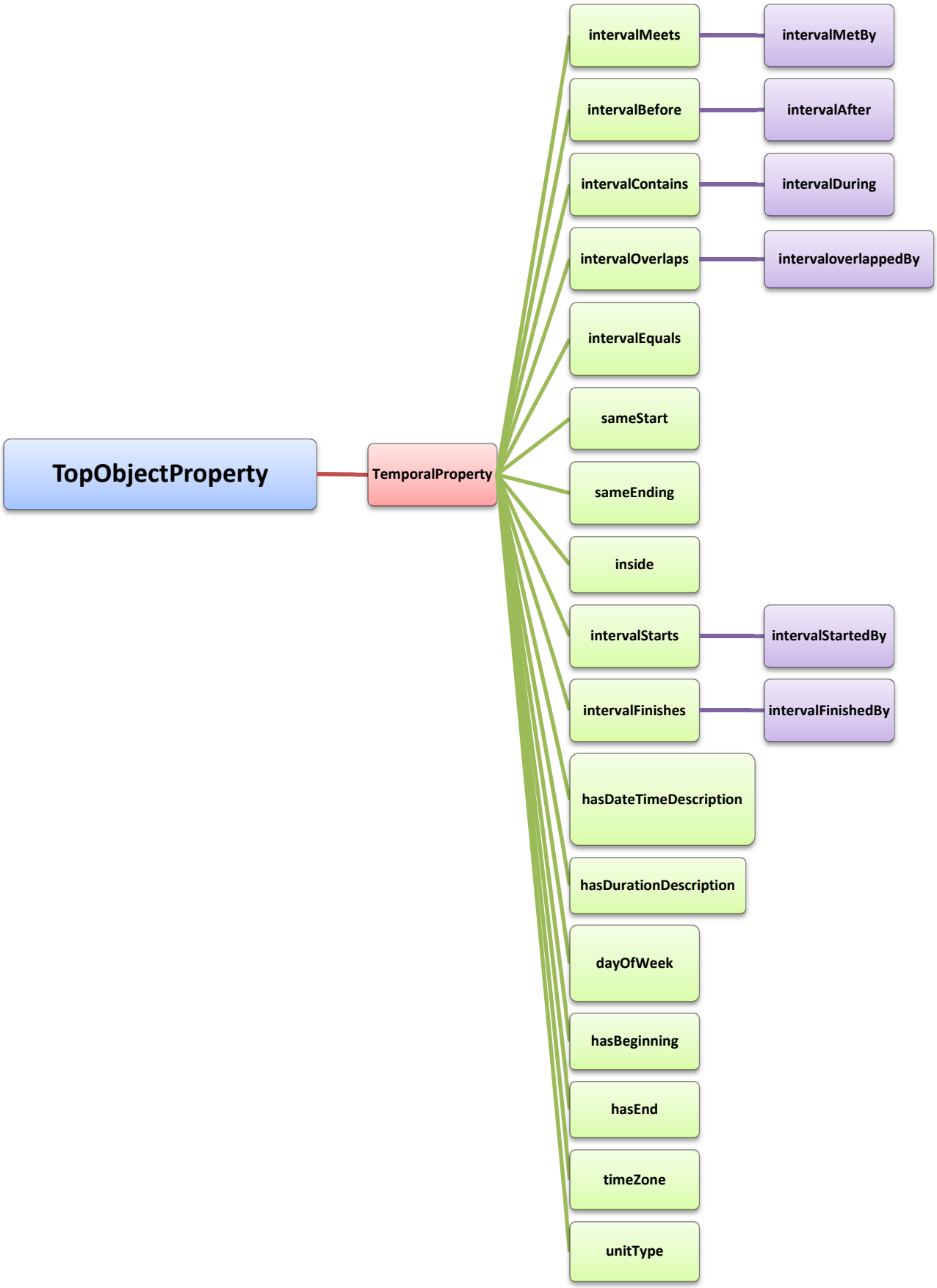

Figure 7 Temporal and event ontology properties

The properties are listed below:

topObjectProperty

intervalFinishes (an interval finishes another interval)

Domain	properInterval
Range	properInterval
Super Property	topObjectProperty
Inverse Property	intervalFinishedBy

Id (Each event can be associated with a unique ID for identification purpose.)

IdBuy

Domain	Buy
Range	Buy
Super Property	Id

IdSell

Domain	Sell
Range	Sell
Super Property	Id

hasBeginning

Domain	Temporal Entity
Range	Instant
Example	hasBeginning(interval_i, instant_j)

hasEnd

Domain	temporalEntity
Range	Instant
Example	hasEnding(interal_i, instant_k)

hasDateTimeDescription

Domain	DateTimeInterval
Range	DateTimeDescription
Property Chain	intervalDuring o hasDateTimeDescription => hasDateTimeDescription
Example	hasDateTimeDescription(interval_i, Dec2009)

hasDurationDescription

Domain	temporalEntity
Range	DurationDescription
Example	hasDurationDescription(interval_i, 2010)

inDateTime

Domain	Instant

Range	DateTimeDescription
Example	inDateTime(instant_j, 31Dec2009)

Inside

Domain	Instant
Range	Interval
Example	Inside(instant_j, interval _i)

intervalBefore

Domain	properInterval
Range	properInterval
Inverse Property	intervalAfter
Example	intervalBefore(interval_j, interval_i)
Property Chain	intervalMeets o intervalMeets => intervalBefore intervalMeets o intervalBefore => intervalBefore intervalBefore o intervalBefore => intervalBefore

intervalAfter

Domain	properInterval
Range	properInterval
Inverse Property	intervalBefore
Example	intervalAfter(interval_i, interval_j)
Property Chain	inverse (hasDeadline) o hasSubEvent o hasDeadline => intervalAfter

intervalContains

Domain	properInterval
Range	properInterval
Inverse Property	intervalDuring
Example	intervalContains(interval_i, interval_m)

intervalDuring

Domain	properInterval
Range	properInterval
Inverse Property	intervalContains
Property Chain	inverse (time) o inverse (hasSubEvent) o time => intervalDuring
Example	intervalDuring(interval_m, interval_i)

intervalEquals

Domain	properInterval
Range	properInterval

Example	intervalEquals(interval_i, interval_i)

intervalFinishes

Domain	properInterval
Range	properInterval
Inverse Property	intervalFinishedBy
Example	intervalFinishes(interval_2009, interval_Dec2009)

intervalFinishedBy

Domain	properInterval
Range	properInterval
Inverse Property	intervalFinishes
Example	intervalFinishedBy(interval_Dec2009, interval_2009)

intervalMeets

Domain	properInterval
Range	properInterval
Inverse property	intervalMetby
Example	intervalMeets(interval_i, interval_j)
Property Chain	intervalMeets o intervalStarts => intervalMeets intervalFinishes o intervalMeets => intervalMeets

intervalMetby

Domain	properInterval
Range	properInterval
Inverse property	intervalMeets
Example	intervalMetby(interval_j, interval_i)

intervalOverlaps

Domain	properInterval
Range	properInterval
Inverse property	intervalOverlappedBy
Example	intervalOverlaps (interval_Jan_to_Mar, Interval_Feb_to_Apr)

intervalOverlapppedBy

Domain	properInterval
Range	properInterval
Inverse property	intervalOverlaps
Example	intervalOverlapppedBy (Interval_Feb_to_Apr , interval_Jan_to_Mar)

intervalStarts

Domain	properInterval
Range	properInterval
Inverse property	intervalStartedBy
Example	intervalStarts(interval_Jan2009, interval_2009)

intervalStartedBy

Domain	properInterval
Range	properInterval
Inverse property	intervalStarts
Example	intervalStartedBy(interval_2009, interval_Jan2009)

Sameending

Domain	properInterval
Range	properInterval
Example	intervalSameEnding(interval_Dec2009, interval_31Dec2009)

Samestart

Domain	properInterval
Range	properInterval
Example	intervalSameStart(interval_Jan2009, interval_2009)

timeZone

Domain	DateTimeDescription
Range	TimeZone
Example	timeZone(Dec2009, UCT+1)

unitType

Domain	DateTimeDescription
Range	unitType
Example	unitType(Dec2009, Month)

Property of each temporal relation

Relation	Reflexive	Symmetric	Transitive
intervalMeet	No	No	No
intervalMetby	No	No	No
intervalBefore	No	No	Yes
intervalAfter	No	No	Yes
intervalEqual	Yes	Yes	Yes
intervalStarts	No(by assumption)	No	No
intervalEnds	No(by assumption)	No	No
intervalFinishes	No(by assumption)	No	Yes
intervalFinishedBy	No(by assumption)	No	Yes

intervalOverlaps	No(by assumption)	No	Yes
invervalOverlappedBy	No(by assumption)	No	Yes
sameStart	Yes(by assumption)	Yes	Yes
sameEnding	Yes(by assumption)	Yes	Yes
intervalDuring	No	No	Yes

Table 1 Property of each temporal relation

4.5 Entities of temporal Ontology

This section states the entities used in the ontology which are included in the reasoning examples covered in chapter 4 and 5.

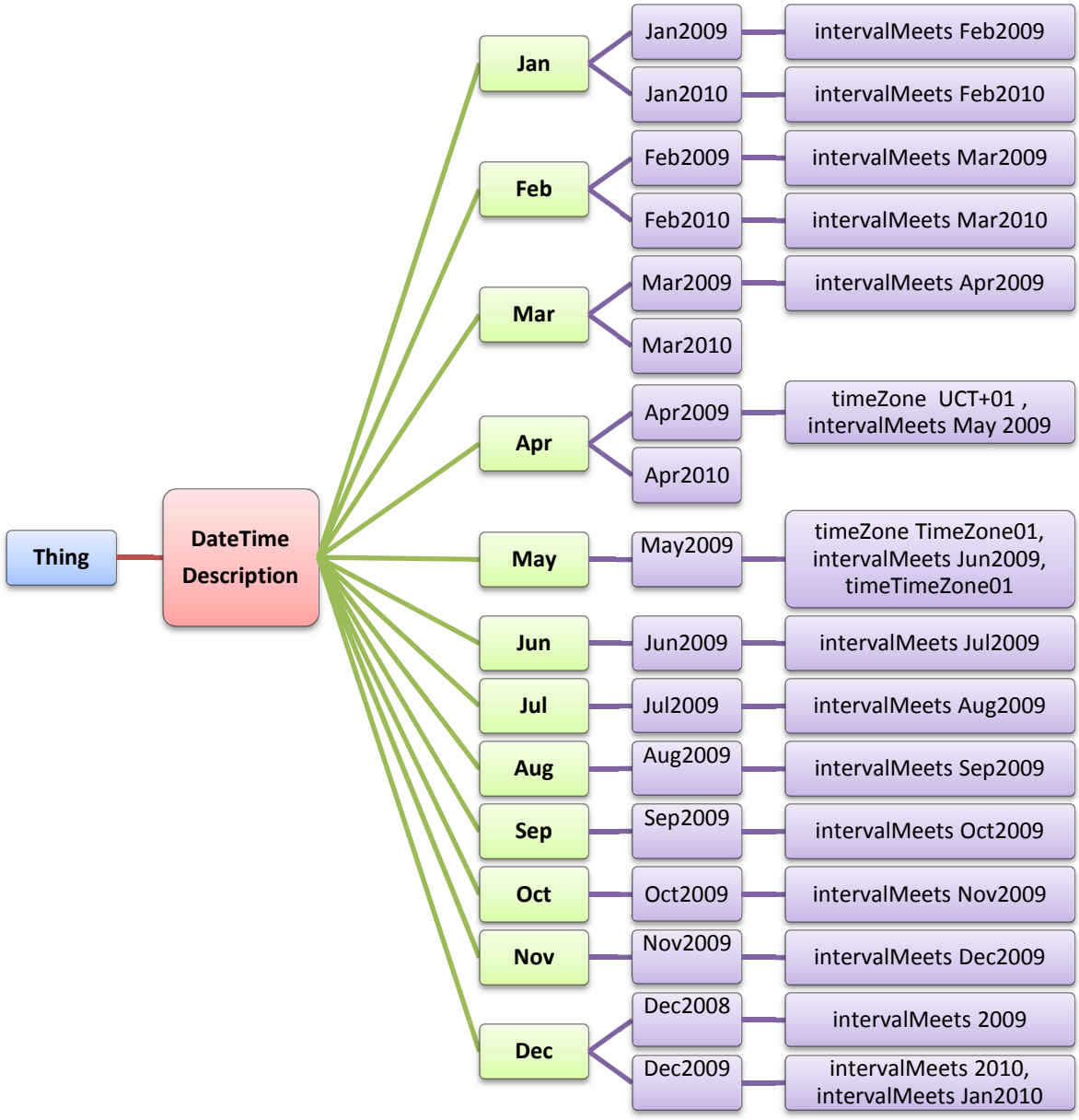

Figure 8 Object list of temporal ontology [1]

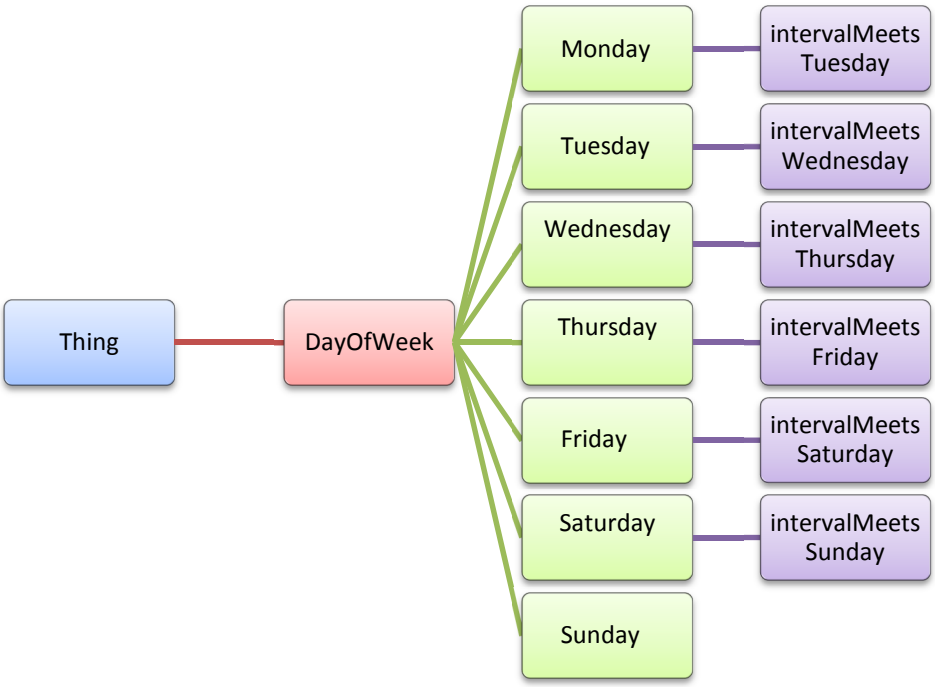

Figure 9 Object list of temporal ontology [2]

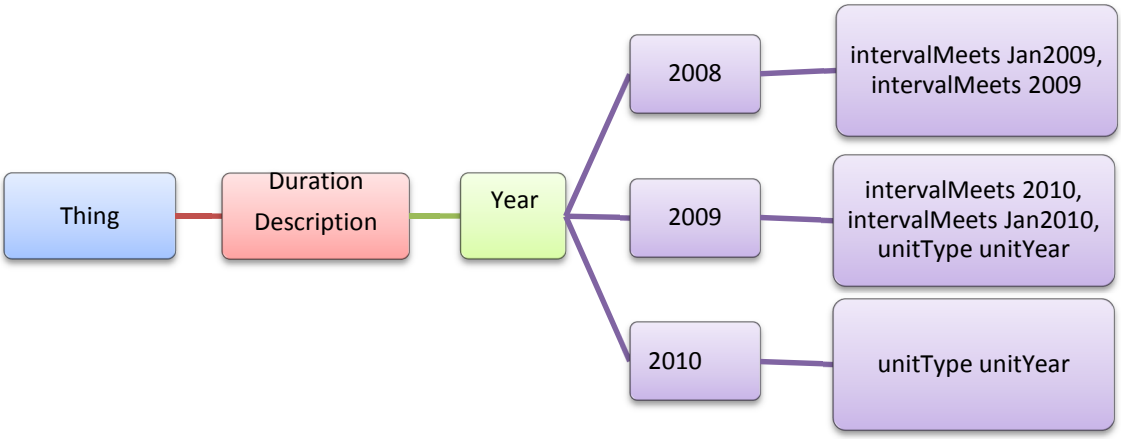

Figure 10 Object list of temporal ontology [3]

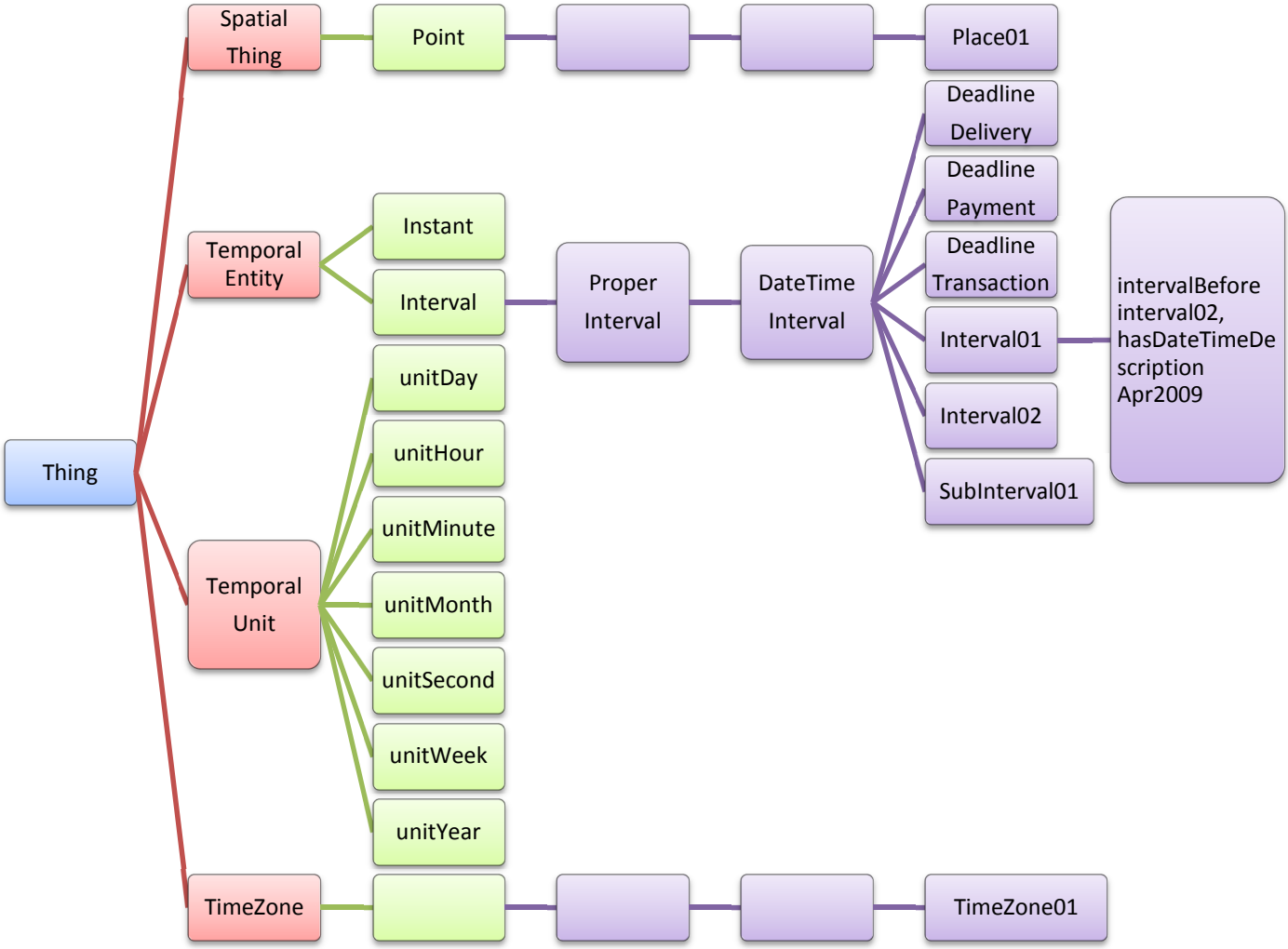

Figure 11 Object list of temporal ontology [4]

4.6 Combining Rules and Temporal Ontology

4.6.1 Why are Rules needed?

Generally, an OWL ontology contains a sequence of axioms and facts. Axioms may be also of different types, unfortunately, in some cases, OWL may not suffice for all applications, there are statements that cannot be expressed in OWL. Even though first-order logic is sufficient for formalizing many logic

descriptions, it has certain limitations on its expressiveness and the fragments of natural languages that it can describe. In addition to this, first order logic is also bad at handling default information which may lead to inconsistency.

An example showing the limitation of OWL is shown in the following graph which states , “interval i meets interval m, which is also meet by another interval j, it is inferred that interval i and j have the same ending, represented with the relation ***sameending***”.

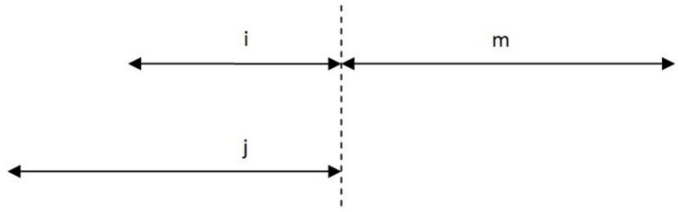

Figure 12 "sameEnding" [1]

In this case, first-order logic is not sufficient, rules as an alternative paradigm for modeling should be applied. Usually, a relatively informal “human readable” form similar to that used in many published works on rules are used, a rule has the form:

antecedent => consequent

Where both antecedent and consequent are conjunctions of atoms, a rule asserting that “all priests are male” would be written:

priest(x) => male(x) //if x is a priest, then it can be derived that x is male

Apart from the format stated above, there are rules an empty antecedent used for providing unconditional facts, which instead are preferred to be stated in OWL itself. While in some other cases, antecedent and consequent can be conjunctions of atoms. For instances, a rule asserting the fact that the friend of an enemy is an enemy would be written as:

Enemy(Tom, Jerry) ∧ Friend(Jerry , Bulldog) => Enemy (Tom, Bulldog)

Since rule languages are hardly compatible with each other, it is important to choose an adequate rule language, the possible criteria should be considered while choosing a rule language:

- Does it prove clear specification of syntax and semantics?
- Is it supported by software tools?
- Which expressive features are needed?
- Is the complexity of implementation acceptable for the situation?
- How is the performance?
- Is the rule compatible with other formats, e.g. OWL?
- Is it declarative or operational?
- …

In this thesis, SWRL rule is used for combining datalog and OWL, which is short for Semantic Web Rule Language. It is a combination of OWL DL, OWL Lite and Datalog, and has the full power of OWL DL.

4.6.2 Applying Rules in Temporal ontology

Some of the rules inserted for representing the assertions are as explained in the section below:

(1) intervalMeets(?i, ?m), intervalMeets(?j, ?m) -> sameending(?i, ?j)

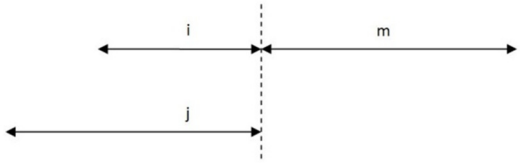

Figure 13 "sameEnding" [2]

Description: If an interval i meets interval m, while at the same time m is also meet by another interval j, it is inferred that i and j have the same ending instance, which is represented with the relation "sameending(x,y)".

(2) intervalMeets(?i, ?k), intervalMeets(?l, ?j), intervalMetBy(?j, ?i) -> intervalMeets(?l, ?k)

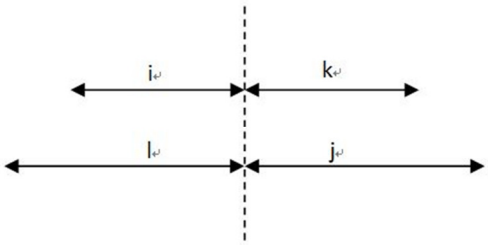

Figure 14 "intervalMeets" [1]

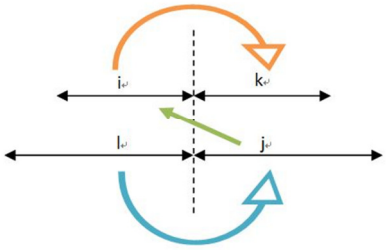

Figure 15 "intervalMeets" [2]

Description: If an interval i meets interval k, interval l meets interval j, while at the same time, i also meets j, (therefore j is meet by interval i), it is concluded that interval l also meets k.

(3) intervalMeets(?m, ?i), intervalMeets(?m, ?j) -> samestart(?i, ?j)

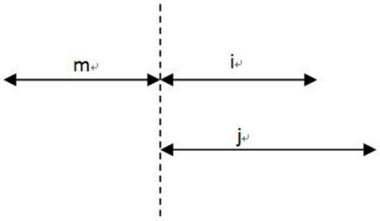

Figure 16 "samestart"

Description: If an interval m meets interval i and j, then i and j have the same starting instance, represented with relation “samestart(x,y)”.

(4) intervalBefore(?i, ?k), sameending(?k, ?j), samestart(?i, ?j) -> intervalStarts(?i, ?j)

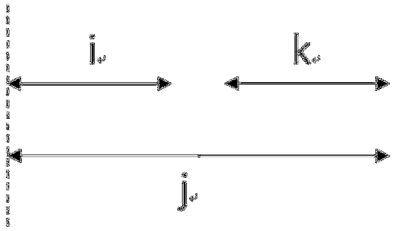

Figure 17 "intervalStarts"

Description: If an interval i is before interval k, and there is an interval j which has the same ending instance as k, and same starting instance as i, then i starts j, represented with relation “invervalStarts(x,y)”. Note that, it is pre-assumed that, if x starts y, then y has to finish later than i. otherwise we have to say that y starts x, with the relation “startedBy(x,y)”, which is the reserve relation of intervalStarts(x,y).

(5) intervalMeets(?i, ?l), intervalMeets(?j, ?l), intervalMeets(?k, ?i), intervalMeets(?k, ?j) -> intervalEquals(?i, ?j)

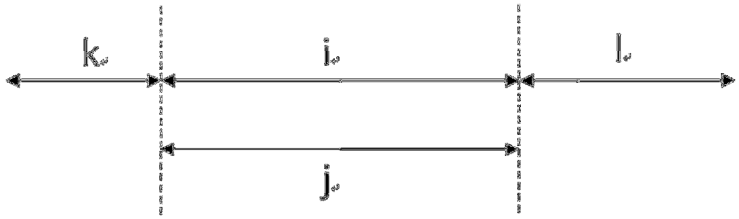

Figure 18 "intervalEquals"

Description: If an interval k meets interval i and j, and both i , j meet another interval l, then i is equivalent to j, represented as “intervalEquals(x,y)”.

(6) intervalMeets(?m, ?j), samestart(?i, ?j) -> intervalMeets(?m, ?i)

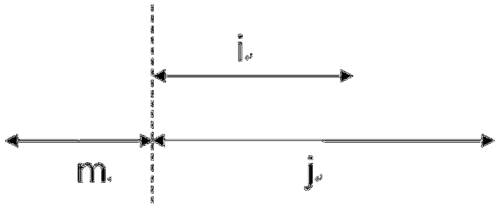

Figure 19 "intervalMeets"

Description: If an interval m meets interval j, while interval i has the same starting instance as j, then m also meets j.

(7) intervalMeets(?i1, ?i2), intervalMeets(?i10, ?i11), intervalMeets(?i11, ?i12), intervalMeets(?i2, ?i3), intervalMeets(?i3, ?i4), intervalMeets(?i4, ?i5), intervalMeets(?i5, ?i6), intervalMeets(?i6, ?i7), intervalMeets(?i7, ?i8), intervalMeets(?i8, ?i9), intervalMeets(?i9, ?i10), samestart(?i1, ?l), unitType(?i, unitMonth), unitType(?l, unitYear) -> intervalDuring(?i10, ?l), intervalDuring(?i11, ?l), intervalDuring(?i2, ?l), intervalDuring(?i3, ?l), intervalDuring(?i4, ?l), intervalDuring(?i5, ?l), intervalDuring(?i6, ?l), intervalDuring(?i7, ?l), intervalDuring(?i8, ?l), intervalDuring(?i9, ?l)

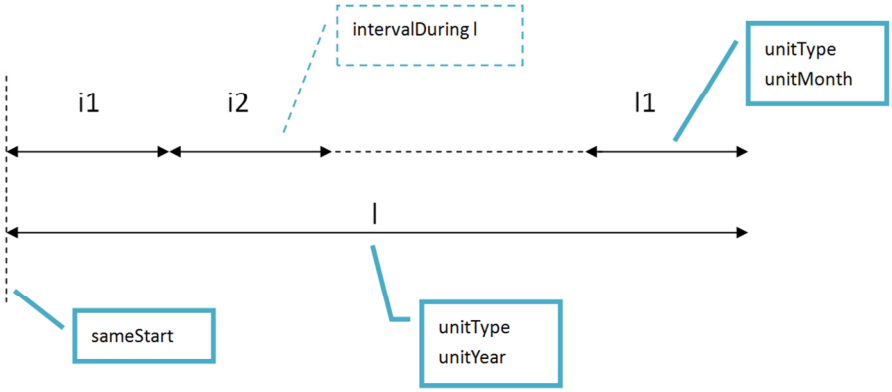

Figure 20 Month and year assertion [1]

This rules is a special assertion for the statement "every month of the year is 'intervalDuring' the year", in this assertion, i1…i12 are representing months while l is the year that has the same starting time as i1, it could be concluded that i1…i12 are during the year l.

(8) intervalMeets(?i1, ?i2), intervalMeets(?i10, ?i11), intervalMeets(?i11, ?i12), intervalMeets(?i2, ?i3), intervalMeets(?i3, ?i4), intervalMeets(?i4, ?i5), intervalMeets(?i5, ?i6), intervalMeets(?i6, ?i7), intervalMeets(?i7, ?i8), intervalMeets(?i8, ?i9), intervalMeets(?i9, ?i10), sameending(?i12, ?j), samestart(?i1, ?j) , unitType(?i, unitMonth)-> unitType(?j, unitYear), intervalFinishes(?i12, ?j), intervalStarts(?i1,?j)

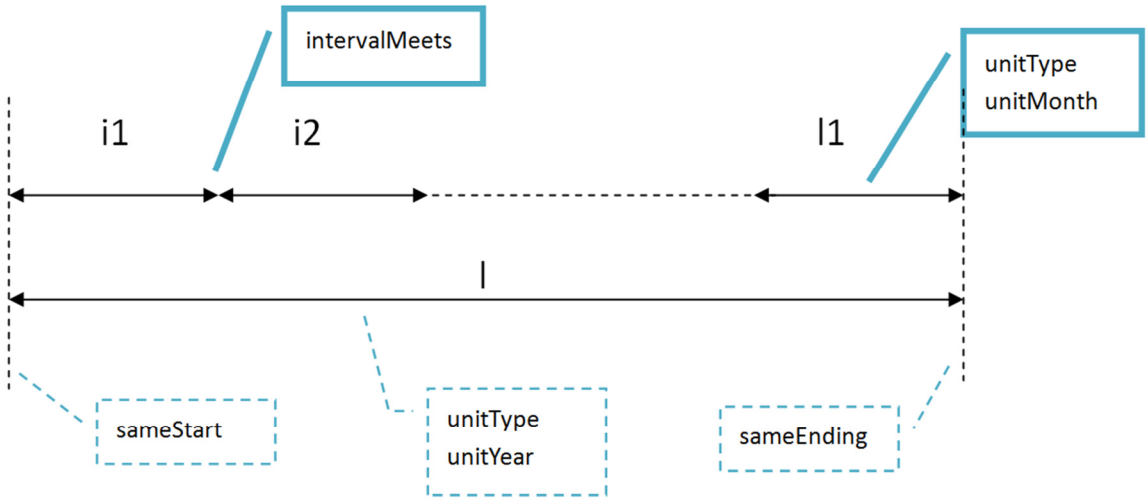

Figure 21 Month and year assertion [2]

This rule is inserted for the assertion "for twelve consequent months, if the last month has the same ending time as j, while j has the same starting time as the first month, it can be concluded that j is a year and it covers these twelve months".

4.7 Reasoning with Temporal ontology

This section shows examples of the deductive power of the ontology. Closed World Assumption has been applied.

Example 1: interval i and j are both DateTimeInterval, and i is "intervalDuring" j, j has DateTimeDescription as Jan2009 :

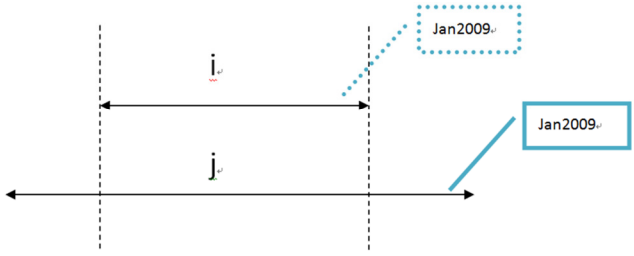

Figure 22 intervalDuring

According to the propertyChain

intervalDuring o hasDateTimeDescription => hasDateTimeDescription

it can be concluded that interval i has DateTimeDescription Jan2009.

Example 2: Interval i meets interval j, interval j meets interval k,

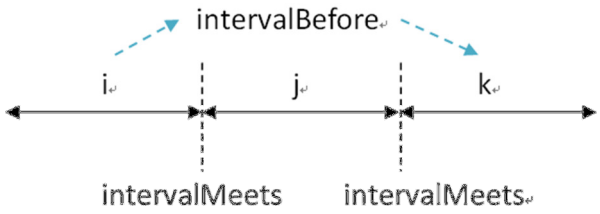

Figure 23 intervalBefore[1]

According to property chain

intervalMeets o intervalMeets => intervalBefore

It can be concluded that interval i is before interval k.

Example 3: Interval i is before interval j, interval j is before interval k:

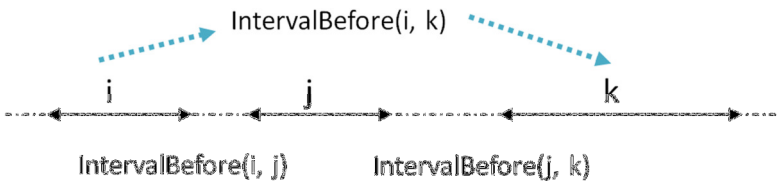

Figure 24 IntervalBefore[2]

According to property chain:

intervalBefore o intervalBefore => intervalBefore

It can be concluded that interval i is before interval k as well.

Example 4: An transaction process has deadline on July 2011, its sub_event "Place order" has the deadline on May 2011 :

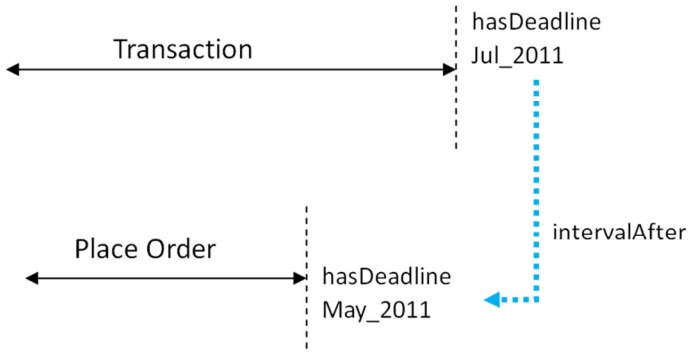

Figure 25 intervalAfter

According to the property chain:

inverse (hasDeadline) o hasSubEvent o hasDeadline => intervalAfter

It can be concluded that Jul_2011 is intervalAfter May_2011. (Even though this may seem obvious to human, the purpose of this property is to make machines able to maintain deadline consistency inside the event process.)

Example 5: In the same example as above, Place order is a sub_event of transaction, and they are separately assigned time intervals May 2011 and from Apr to Jul 2011

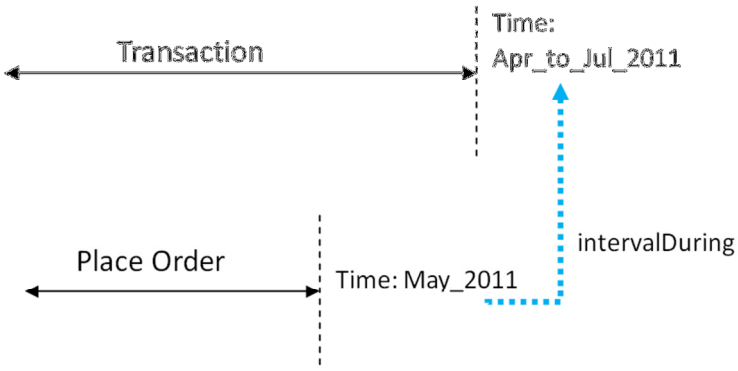

Figure 26 intervalDuring

According to the property chain:

inverse (hasDateTimeDescription) o inverse (hasSubEvent) o hasDateTimeDescription => intervalDuring

It can be concluded that May_2011 is intervalDuring Apr_to_Jul 2011.

More examples were included in the temporal ontology OWL File.

Chapter 5 Event ontology

This chapter explains the design of the event ontology which was built on base of the event ontology developed in the University of London , 2004. This ontology is focused on the notion of event and agents, the following figure shows the design of the event model [12]:

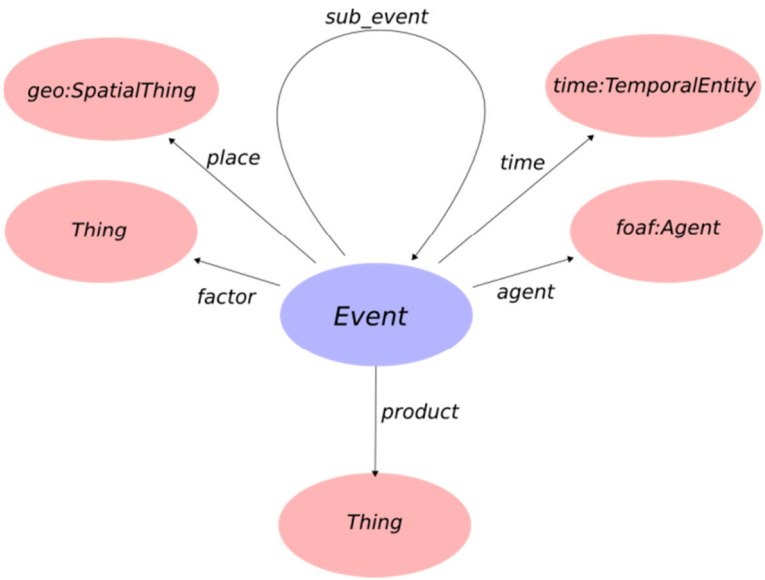

Figure 27 Event Ontology Design

From the above model it could be seen that, the event ontology deals with the notion of reified events as the main concept, and geographical, time, agent, factor and products as related classes.

5.1 Classes in Event Ontology

The following figure represents the class design of Event ontology:

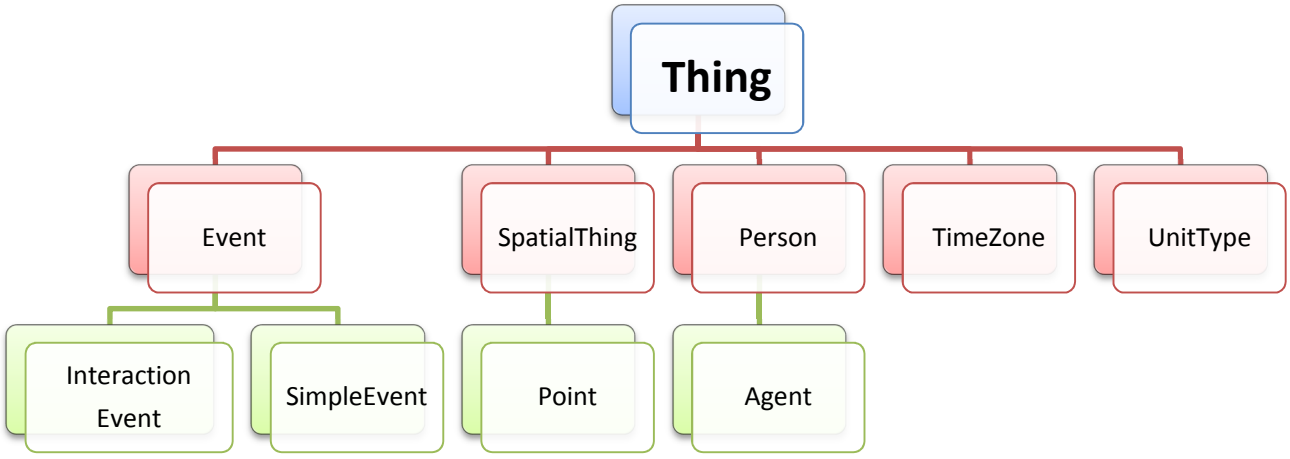

Figure 28 Event ontology classes

The descriptions of each class are explained below:

- **Person**
  Agent is a subclass of Person, which inherits all the properties from Person. Such as gender, nationality, etc.

- **Agent**
  *(Class for participants of events, for instance, a seller is considered to be one of the agents of a selling event. In the interactiveEvent example in the following section, **Jack** and **Elwood** are the two agents of this transaction event.)*

- **Factor**
  Factor is a class for everything that could be used as a cause of an event. Similarly for the class ***Product***.

- **Product**
  Product is a class for everything that are consequences of an event.

- **Event**
  Event is a classification of a space/time region, by a cognitive agent. An event may have one or many participating agents , either actively as event starter, or passively as event receiver, an event might also be associated to a factor and a product, which are ***Things***.
  - InteractiveEvent

(This class includes all the interactive events such as transactions, telephone conference, which usually have more than one agent in participation.)

- Transaction
  (For each transaction, there must be at least two participants, and they are both event starters and event receivers. E.g. Jack and Elwood are having a business transaction, which is considered to be an interactive event, both Jack and Elwood are event participants, even though one of them could be the buyer and the other one is the seller.)

▪ SimpleEvent
(SimpleEvent refers to events that are initiated by one agent, and has a single receiver)

- Buy
  (This event has buyer as the event starter, and seller as the event receiver.)
- Sell
  (This event has seller as the event starter, and buyer as the event receiver.)

○ **SpartialThing**

▪ Point
(Class for all the geographical points, for instance, an event may occur at Milan, then Milan is a member of the class Point.)

5.2 Object Properties of Event Ontology

This section states all the object properties in the temporal ontology:

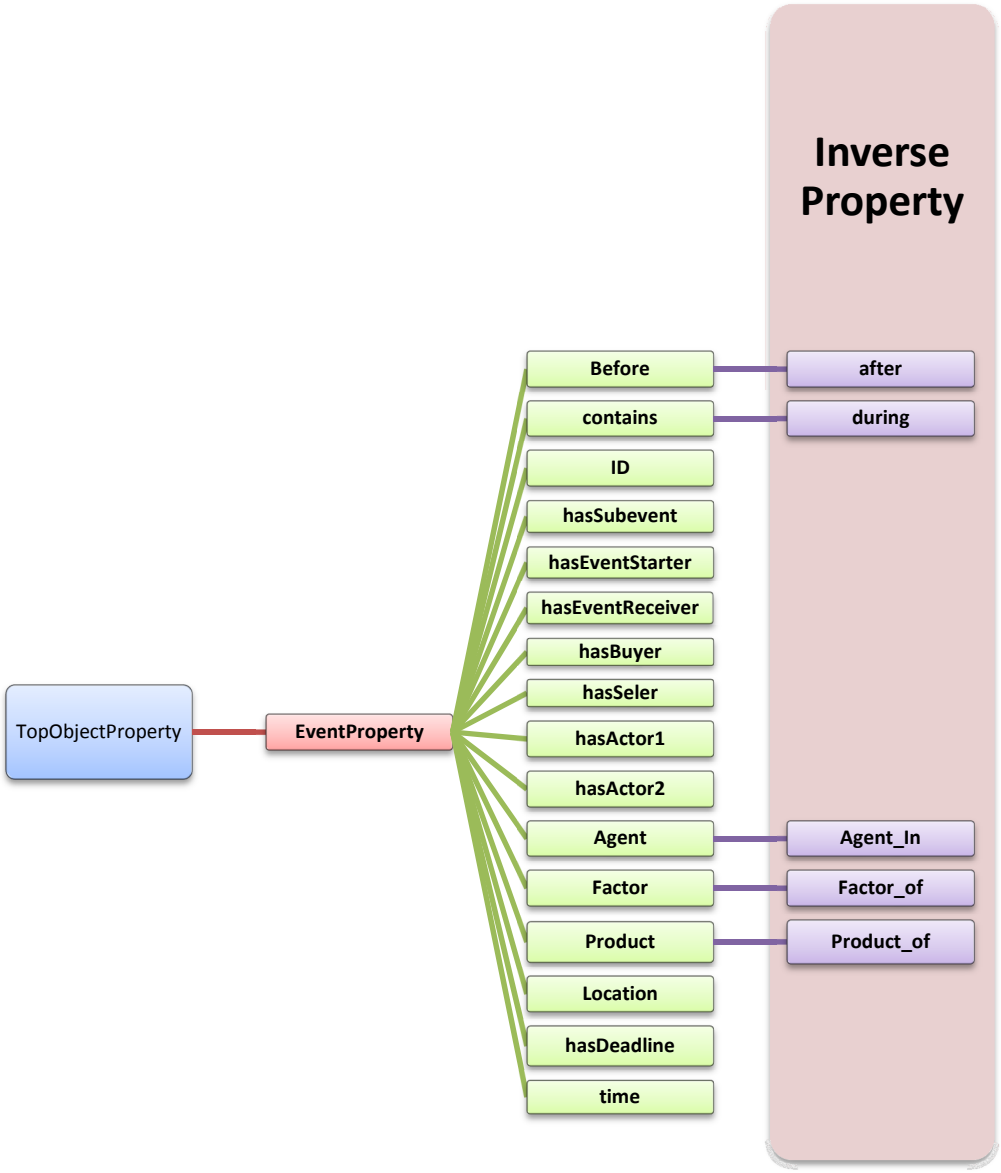

Figure 29 Object Property of Event Ontology

The properties are listed below:

topObjectProperty

Before (an event happens before another event)

Domain	Event
Range	Event
Super Property	topObjectProperty
Inverse property	After
Example	Before(Place_order, Payment)
Property Chain	Before o Before =>Before

After (an event happens after another event)

Domain	Event
Range	Event
Super Property	topObjectProperty
Inverse property	Before
Example	After(Payment, Place_Order)
Property Chain	After o After => After

Agent==has_Agent (an event has at least 1 agent as participator)

Domain	Event
Range	Agent
Equivalent Property	hasAgent
Inverse property	Agent_in
Example	Agent(Place_Order, Jack)
Property Chain	hasSubEvent o Id o hasEventStarter => hasAgent

Agent_in ==isAgentIn (an agent is the participator in an event)

Domain	Agent
Range	Event
Equivalent Property	isAgentIn
Inverse property	has_Agent / Agent
Example	Agent_in(Elwood, Product_Delivery)

location (an event has location at certain point)

Domain	Event
Range	SpatialThing
Example	Location(Place_Order, New York)

Contains (an event contains another event)

Domain	Event
Range	Event
Inverse property	During
Example	Contains(Transaction, Product_Delivery)
Property Chain	Contains o Contains => Contains

During (an event happens during the happening of another event)

Domain	Event
Range	Event

Inverse property	Contains
Example	During(Place_Order, Transaction)
Property Chain	During o During => During

dayOfWeek

Domain	DateTimeDescription
Range	dayOfWeek
Example	dayOfWeek(31Dec2009, Thursday)

factor == hasFactor

Domain	Event
Range	Factor
Inverse property	Factor_of
Example	Factor(Payment , Place_Order)

isFactorOf == factor_of

Domain	Factor
Range	event
Inverse property	factor

hasProduct== Product

Domain	Event
Range	Product
Equivalent Property	Product
Example	Product(Transaction, Payment)

isProductOf == product_of

Domain	Product
Range	event
Inverse property	product

hasActor1

Domain	interactiveEvent
Range	Agent
Example	hasActor1(Transaction, Jack)

hasActor2

Domain	interactionEvent
Range	Agent
Example	hasActor2(Transaction, Elwood)

hasAgent==agent

Domain	Event
Range	Agent

Inverse property	Agent_in
Property Chain	hasSubEvent o Id o hasEventStarter => hasAgent

hasBuyer

Domain	SimpleEvent
Range	Agent
Property Chain	hasSubEvent o IdBuy o hasEventStarter => hasBuyer hasBuyer=>hasAgent
Example	hasBuyer(Transaction, Jack)

hasSeller

Domain	Event
Range	Agent
Property Chain	hasSubEvent o IdSell o hasEventStarter => hasSeller hasSeller=>hasAgent
Example	hasSeller(Transaction, Elwood)

hasEventReceiver

Domain	Event
Range	Agent
Property Chain	IdSell o inverse (hasSubEvent) o hasSubEvent o IdBuy o hasEventStarter SubPropertyOf hasEventReceiver => hasEventReceiver IdBuy o inverse (hasSubEvent) o hasSubEvent o IdSell o hasEventStarter => hasEventReceiver
Example	hasEventReceiver(Place_Order, Seller_Elwood)

hasEventStarter

Domain	Event
Range	Agent
Example	hasEventStarter(Place_Order, Buyer_Jack)

hasSubEvent

Domain	Event
Range	Event
Example	hasSubEvent(Transaction, Sign_Contract)

hasDeadline

Domain	Event
Range	Interval
hasDeadline	hasDeadline (Payment , 2_Jul_2011)

isAgentIn==agent_in

Domain	Agent
Range	Event
Inverse property	Agent

place

Domain	Event
Range	SpatialThing
Inverse property	inverse (hasSubEvent) o place => place

Time

Domain	event
Range	temporalEntity
Example	Time(Transaction, Jul_2011)

Interact

Domain	Agent
Range	Agent
Inverse Property	Interact
Example	Interact(Jack, Elwood)

Property of each Event relation

Relation	Reflexive	Symmetric	Transitive
Before	No	No	Yes
After	No	No	Yes
Contains	No	No	Yes
During	No	No	Yes
hasAgent	No	No	No
hasSeller	No	No	No
hasBuyer	No	No	No
hasEventStarter	No	No	No
hasEventReceiver	No	No	No
Factor	No	No	No
Product	No	No	No
hasSubEvent	No	No	Yes
hasDeadline	No	No	No
Place	No	No	No
Time	No	No	No
Interact	No	Yes	No
DayofWeek	No	No	No

Table 2 Property of each temporal relation

5.3 Entities of Event Ontology

This section states the entities used in the ontology

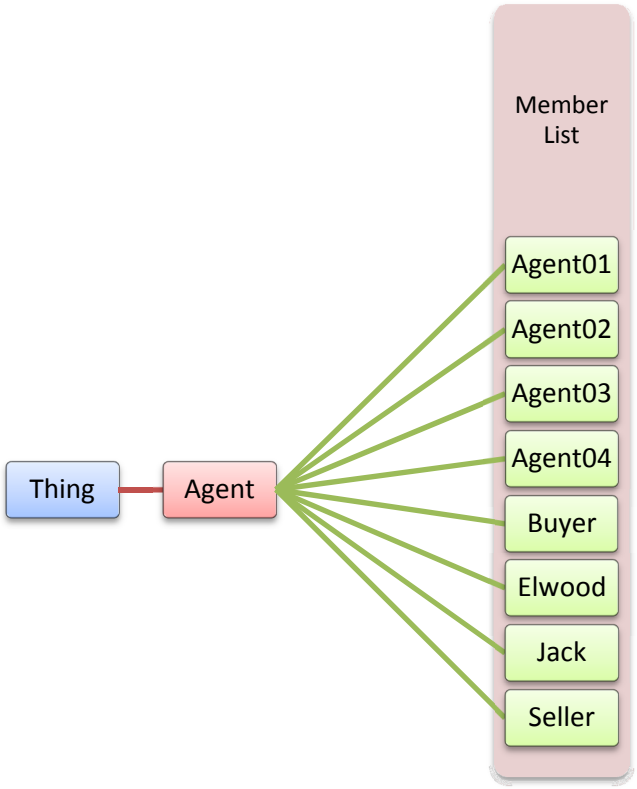

Figure 30 Object list of event ontology

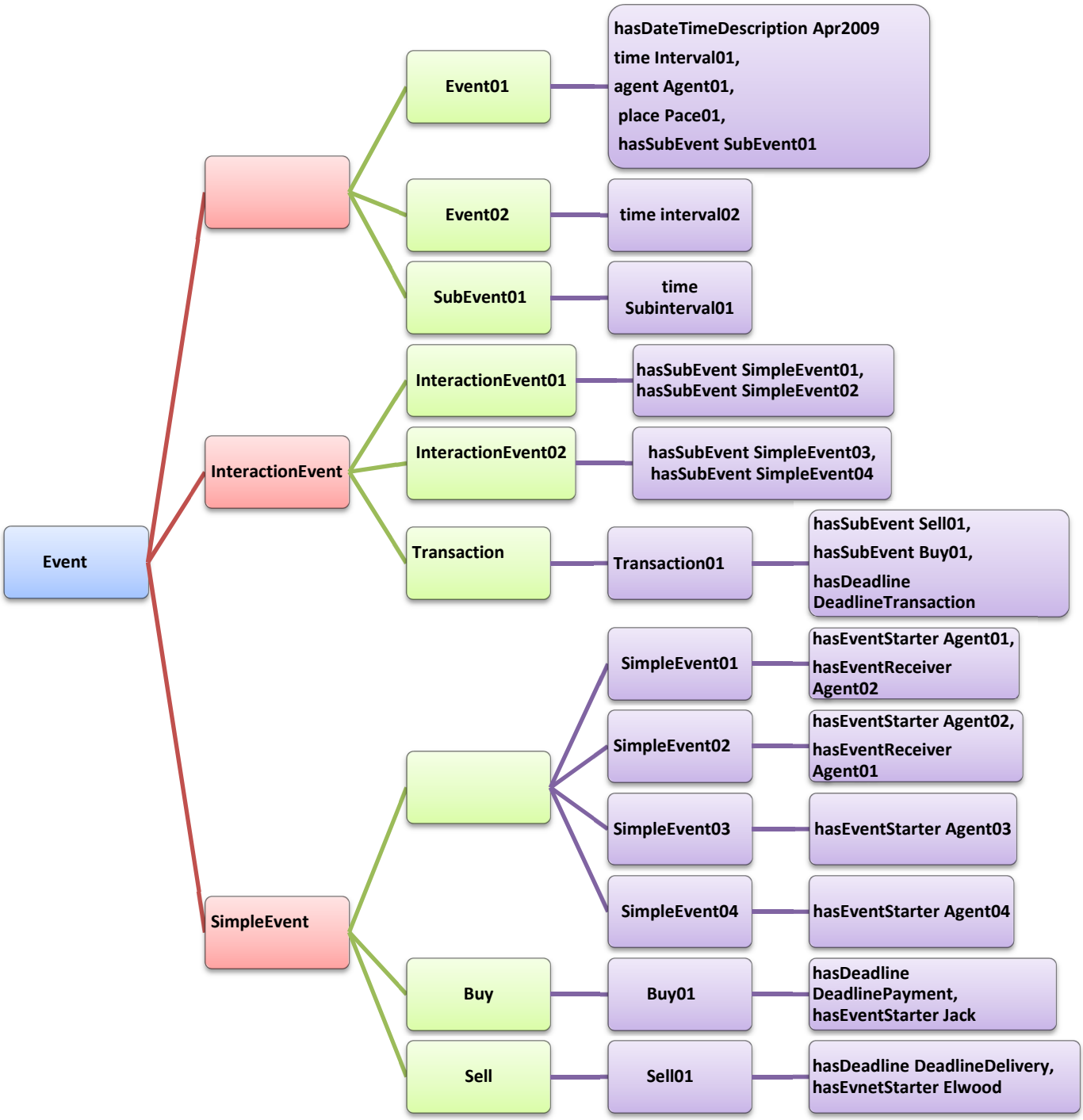

Figure 31 Object list of event ontology [2]

5.4 Applying Rules in Event Ontology

In addition to rules related with temporal assertions, the following rule was created for event assertion:

hasSubEvent(?interact, ?event1), hasSubEvent(?interact, ?event2), hasEventReceiver(?event1, ?actor2), hasEventReceiver(?event2, ?actor1), hasEventStarter(?event1, ?actor1), hasEventStarter(?event2, ?actor2) -> hasAgent(?interact, ?actor1), hasAgent(?interact, ?actor2), Interact(?actor1, ?actor2)

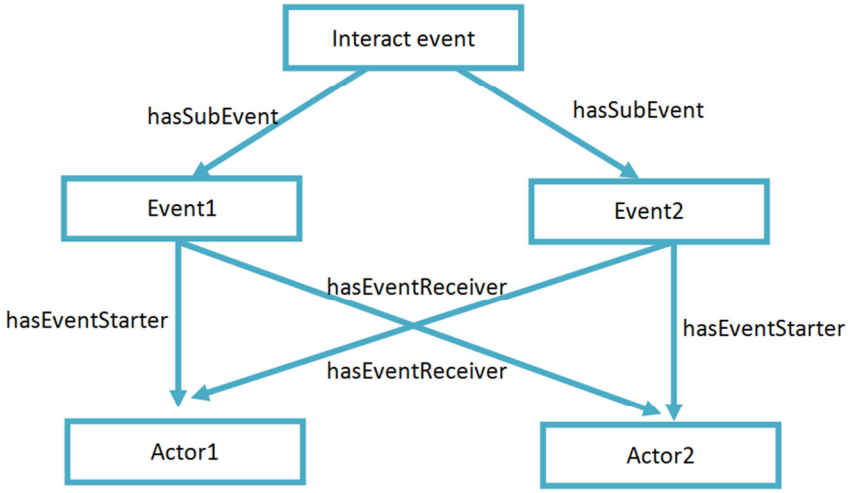

Figure 32 Event assertion

This rule assertion states that "if event 1 and event 2 are both sub-events of an interact event, event1 has event starter actor1 and receiver actor2, and vice versa for event2. It can be concluded that both actor1 and actor2 are agents of the interact event, and they have the relationship 'interact' with each other."

5.5 Reasoning with Event Ontology

Example 1: Transaction01 is a interactionEvent, Buy01 is its sub-event with Jack as the buyer.

	Class	Instance
Event	interactionEvent	Transaction01
SubEvent	Buy	Buy01

Agent	Buyer	Jack

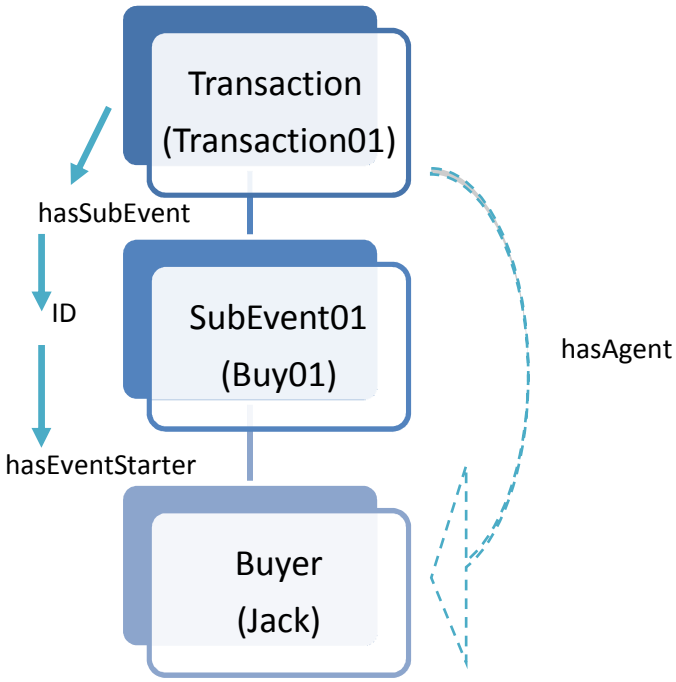

Figure 33 Example 1

According to the chain property:

hasSubEvent o Id o hasEventStarter => has Agent

It can be concluded that Transaction01 has Jack as an agent.

Example 2: Jack and Elwood are the agents in a transaction event , Elwood as the seller is selling some product to Jack, the deadline of buying and selling are separately *DeadlineDelivery* and *DeadlinePayment,* the following assertions have been inserted:

	Class	**Instance**
Event	interactionEvent	Transaction01
SubEvent	Buy	Buy01
SubEvent	Sell	Sell01
Agent	Buyer	Jack

Agent	Seller	Elwood

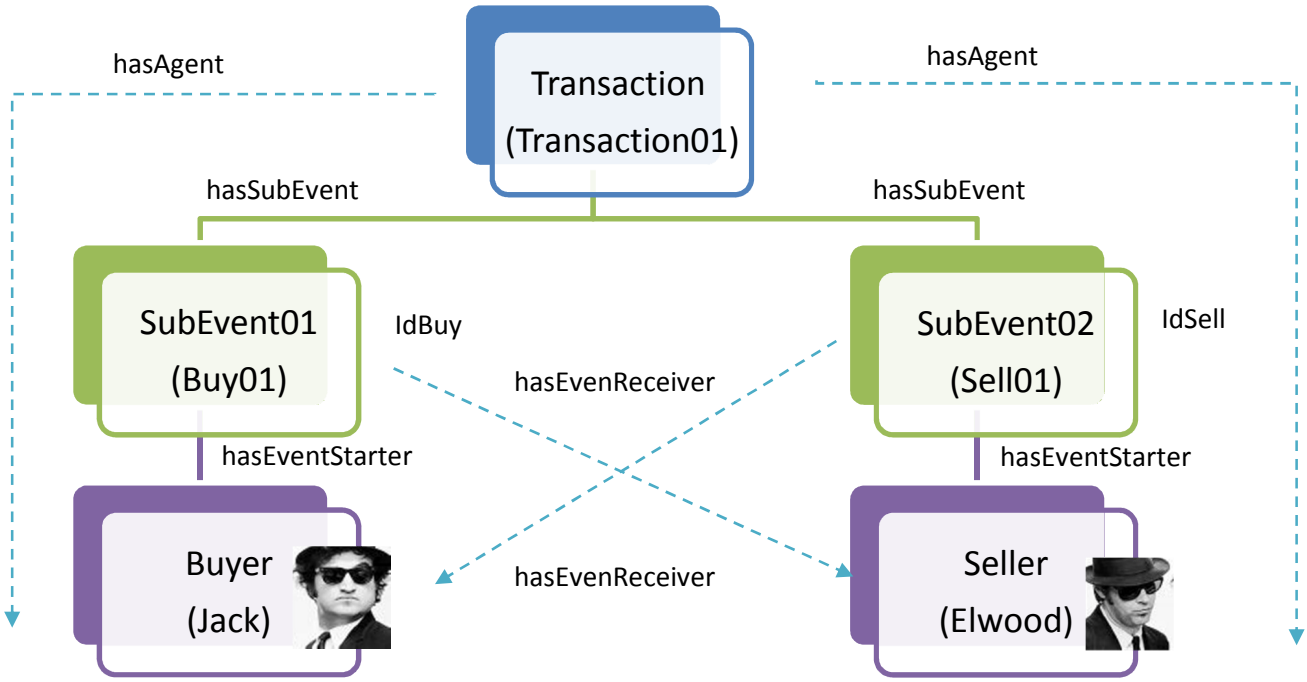

Figure 34 Example 2

According to the property chain:

IdSell o inverse (hasSubEvent) o hasSubEvent o IdBuy o hasEventStarter SubPropertyOf hasEventReceiver => hasEventReceiver

It could be concluded that Jack is the event receiver of SubEvent02, so Jack is buying from Elwood.

According to the property chain:

IdBuy o inverse (hasSubEvent) o hasSubEvent o IdSell o hasEventStarter => hasEventReceiver

It could be concluded that Elwood is the event receiver of SubEvent01, so Elwood is the seller for Jack.

According to the property chain:

hasSubEvent o IdSell o hasEventStarter => hasSeller

It could be concluded that Elwood is the seller of Transaction01.

According to the property chain:

hasSubEvent o IdBuy o hasEventStarter => hasBuyer

It could be concluded that Jack is the buyer of Transaction01.

According to the rule:

hasAgent(?event, ?agent1), hasAgent(?event, ?agent2) =>
interact(?agent1, ?agent2)

It could be concluded that Jack interacts with Elwood.

Example 3: An event Transaction01 has sub-event Sell01, DeadlineTransaction and DeadlineDelivery are the deadlines of these two events separately.

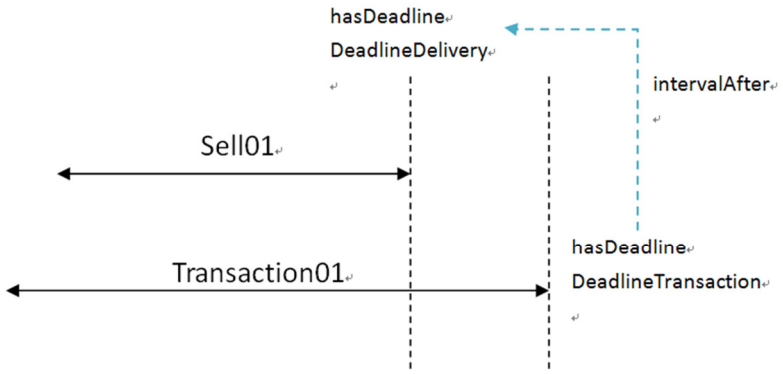

Figure 35 Example 3

According to the property chain:

inverse (hasDeadline) o hasSubEvent o hasDeadline => intervalAfter

it can be concluded that DeadlineTransaction is "intervalAfter" DeadlineDelivery. That is to say, the deadline of an event is later than the deadlines of its sub-events.

Example 4: Sell01 is a sub-event of Transaction01, Sell01 has time Time02 while Transaction01 is assigned to a time Time01:

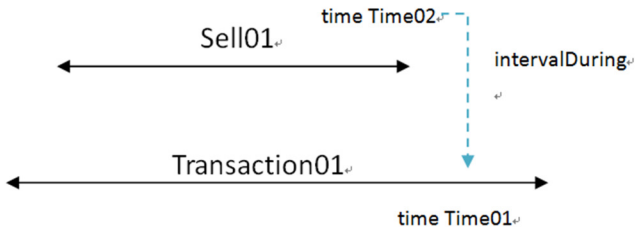

Figure 36 Example 4

According to the property chain:

inverse (time) o inverse (hasSubEvent) o time => intervalDuring

it can be concluded that Time02 is "intervalDuring" Time01, that is to say, the time of a sub-event is during the time of its parent event.

Example 5: Sell01 is the sub-event of Transaction01, whose location is in Milan:

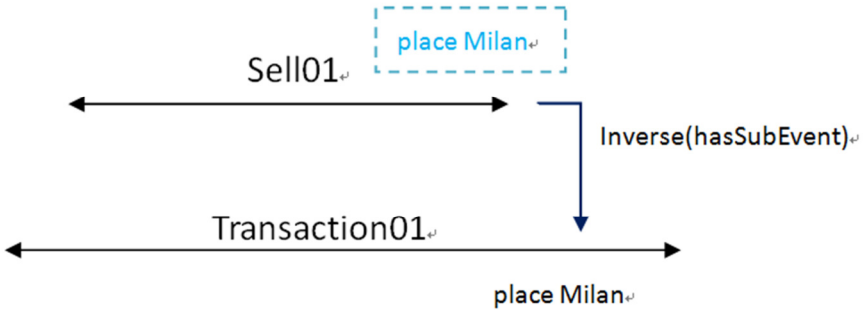

Figure 37 Example 5

According to the property chain:

inverse (hasSubEvent) o place => place

it is concluded that the sell event Sell01 also occurs in Milan because its parent event occurs in Milan.

Example 6: Considering a series transactions in real commerce scenario, Jack places an order, this event is the factor of the "process order" event that is conducted by Elwood, when Elwood finishes processing the order, the payment event took place as a product of "process order", see the following graph:

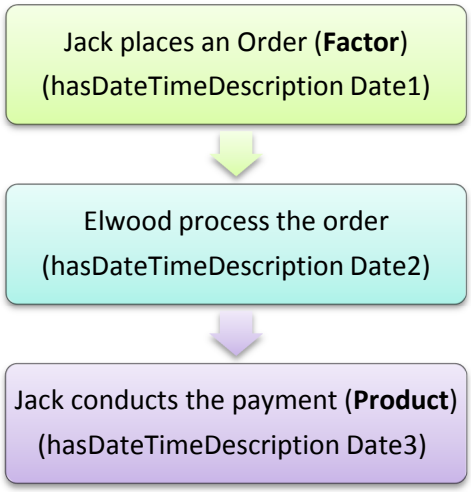

Figure 38 Example of Factor and Product

According to the property chain:

inverse(hadDateTimeDescription) o factor_of o product o hadDateTimeDescription => intervalBefore

inverse(hasDateTimeDescription) o factor_of o hasDateTimeDescription => intervalBefore

inverse(hasDateTimeDescription) o product o hasDateTimeDescription => intervalBefore

It can be concluded that, the time of the factor has to be in advance of the time of the product. The event flow has to follow the nature time flow, that is to say, a factor should always proceeds the consequence in time line.

More examples were included in the Event ontology OWL file.

Chapter 6 Temporal Ontology and Event Ontology for E-business

6.1 E-business Requests

With the rapid development of distance business and web technology, web-based business or E-commerce has now become an important branch in the business technology. E-commerce uses the World Wide Web as the communication medium and supporting technology to provide information and resources to the learner anytime at any location. It also enhances the business handler to manage events through different web-based business managing tools. In fact, the adaptability and the personalization feature of the E-commerce environment is one of the research areas that draw much attention in the E-commerce field. The next generation web technology, Semantic Web, provides a common framework that allows data to be shared and recycled across applications, and it appears to be a promising technology for implementing E-commerce systems.

According to Harvard Business Review [13], the reason for the failures of online business is "*...trying to engage with too many partners too fast is one of the main reasons that so many online market makers have foundered. The transactions they had viewed as simple and routing actually involved many subtle distinction in terminology and meaning*…" in addition to this, technical problems are also a critical issue for the failures of online business, according to Gartner research published on February 28, 2002 [14], "*..lack of technologies and products to dynamically mediate discrepancies in business semantics will limit the adoption of advanced Web services for large public communities whose participants have disparate business processes*". Therefore, a proper way of managing the e-Business transactions and efficient technical support are essential for the success of online business. In order to achieve this, close-world assumption is required in the implementation of the ontology system. The following section explains close-world assumption in details.

6.2 Close-World Assumption

The application of temporal ontology and event ontology on the E-commerce example was carried out base on the closed world assumption, which is the presumption that whatever is undefined or unknown is considered to be false. The opposite of the closed world assumption is the open world assumption which was the base of OWL DL.

In the close-world assumption, all relevant facts are contained in the knowledge base, therefore information provided is usually complete. However, the knowledge base itself is incomplete, it does not contain enough information to produce an answer to certain questions, and therefore a decision (or induction) has to be made without sufficient information provided. Usually, if one statement is not proved, it is assumed to be false. For instance, in James Allen's temporal logic, meet is a relation between intervals, it is not stated that one interval i cannot meet itself, however, by using close-world assumption, it is assumed that one interval (finite interval) cannot meet itself, since it has a positive time span, the starting instance and the ending instance must not be equivalent, therefore it does not meet the requirement of "meet" relation, therefore, it is also concluded that "meet" is not symmetric. The close-world assumption is designed to finesse and is adopted in default for a better solution.

In the E-commerce example, the knowledge base is incomplete but an optimal definite answer is expected to be derived from this incomplete information. For example, if a database contains the following table with the information of a few events and their corresponding agents:

Agent	Event
Elwood	Process Order 01
Johnson	Place Order 02
Elwood	Product Delivery 01
Jack	Payment 01

Table 3 Part of the Database Showing a Few Events

In the closed world assumption, the table is assumed to be complete, the information about the event "Place Order 01" is missing, however, according to the assumption that there could be only two agents participating in an interactive event (also known as a transaction in this example) and according to closed world assumption, it can be inferred that Jack is the agent for the event "Place Order 01", and the relation between Jack and Elwood is "Interact".

6.3 What can Temporal and Event Ontology do with E-Commerce

With the combination of temporal and event ontology, it is possible to do the following predictions and reasonings for E-commerce applications:

1. Predict time conflicts (if a time interval starts or overlaps another time interval)
2. Managing Deadlines
3. Check the consistency of different deadlines involved in a transaction series (if the deadlines are coherent with the event flow)
4. Check correct time flows of events (if factor proceeds product/consequences)
5. Managing human relations (see the interact relations)
6. Predict event places (base on its sub_event or parent event)

Chapter 7 Conclusion, Evaluation and Future Work

In this thesis, I would like to design and develop an integration of temporal ontology and event ontology, an example of E-commerce was describing using these ontologies.

The main objectives of the ontology implemented for this thesis is to design a reusable and adaptive framework for E-business system. Rule-based approach was used in the ontologies so that the knowledge base can be encoded as rules in a clear and logic-like way. This approach also allows event handlers to edit the adaptation rules and manage the adaptation behavior of the system according to different needs and change of scenarios.

An ontology-based approach was used to describe the temporal and event domain model. This approach provides an easy way to improve the content, create more events and temporal intervals and handle their relations accordingly.

There are many areas in which the temporal ontology and event ontology can be improved, both in terms of enhancements and extension of the current structure. The future work will be discussed in the last section of this chapter.

7.1 Overview

The main objective of the thesis is to create an adaptive and personalized temporal ontology and event ontology for E-commerce system using OWL and protégé. The motivation behind the development of this framework is to increase the usability and effectiveness of the temporal and event model. The framework that I have developed aim at helping business handler manages events and schedules.

.

7.2 Related work:

Semantic web technology has become a hot point of internet technology studies, until now, a great number of projects and applications are have been developed or under development.

Yahoo: Yahoo website is a world famous commercial website. It has adopted a light ontology, which contains only the hierarchy classification method. Yahoo provides key word search base on hierarchical navigation, but its searching scope is limited within the file. It could return files relevant with the keywords, without being able to process the content of the file according to ontology information.

COHSE system: COHSE is sponsored by EPSRC (The Engineering and Physical Sciences Research Consil) for the DIM (Distributed Information Management) project. The aim of COHSE was to investigate methods to improve significantly the quality, consistency and breadth of linking of online documents at retrieval time and authoring time. The emphasis of COHSE is on hypertext authoring, it focuses on the process of resource discovery and search engine technology. COHSE is used as a link between site navigation and linking.

SHOE System: SHOE system supports semantic based information retrieval by inserting marks in HTML. It provides limited induction support.

Ontobroker System: The objective of Ontobroker is to offer intelligent online knowledge access, it uses the ontology to mark and achieve web page files, and provide ontology based information retrieval service. Comparing with SHOE system, ontobroker provides support for logical framework, thus it could provide a more complex reasoning support for query result.

KAON System: KAON is a commercial application with an open-source ontology management infrastructure. It contains a series of tools, including ontology creation tools, management tools, and it provides a basis for ontology-based applications. KAON's focus is on ontology management and application of traditional technology and business application technology integration.

7.3 Future work:

There are many areas in which the temporal ontology and event ontology can be improved, both in terms of enhancements and extension of the current framework.

The temporal ontology was extended base on James Allen's temporal logic, however, in James Allen's logic, not only temporal logic was describe, but also the logic of events was discussed. In the future work, the logic of events could be taken into consideration and future extension could be conducted accordingly.

The event ontology could be further refined with more classes and axioms. More realistic constrains could be applied to the model, for instance, if an agent could follow two events at different geographical points simultaneously; to what extend should a sub_event inherit properties from its parent_event; etc. In addition, in this thesis, timeZone information was inserted but could be implemented further for comparing time interval relations, which is not supported yet at the current thesis.

The capability of the E-commerce system could be improved by introducing other relevant ontologies, such as FOAF (Friend of a Friend), bibliographic ontology, etc. With the introducing of new ontologies, the E-commerce system could handle human-relations and geographical relations.

In the reasoning of the ontologies with the E-commerce example, more complex scenarios could as well be designed for testing the reasoning ability of the ontologies. For instance, more rules for assertions and constrains concerning more complex scenarios could be inserted on the existing knowledge base.

Bibliography

[1] Klein,M., et al. 2002. Siguenza, Spain. *Ontology versioning and change detection on the web. In 13th International Conference on Knowledge Engineering and Knowledge Management* (EKAW02).

[2] G. Antoniou & F. Van Harmelen, The MIT Press, 2004, *A Semantic Web primer.*

[3] L. Aroyo & D. Dicheva, 2004, *The New Challenges for E-learning: The Educational Semantic Web, Educational Technology & Society*, 7(4), 59-69.

[4] Gruber, T.R., 1993, Kluwer Academic Publishers. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing, in Formal Ontology in Conceptual Analysis and Knowledge Representation*.

[5] Vladimir Kolovski, Strahil Jordanov and John Galletly , *2004, An Electronic Learning Assistant*, International Conference on Computer Systems and Technologies - CompSysTech'

[6] Berners-Lee, T. Semantic Web - XML2000. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

[7] Reka Marta Sabou, geboren te Turda, Roemeni¨e, "*Building Web Service Ontologies*" vrije universiteit

[8] Li Ding, Pranam Kolari, Zhongli Ding, Sasikanth Avancha, Tim Finin, Anupam Joshi , "*Using Ontologies in the Semantic Web: A Survey"*, Department of Computer Science and Electrical Engineering,University of Maryland Baltimore County

[9]Julio César Arpírez1, Asunción Gómez-Pérez1, *Reference Ontology and (ONTO)2Agent: The Ontology Yellow Pages*, Adolfo Lozano-Tello2 and Helena Sofia Andrade N. P. Pinto3

[10] Mike Uschold & Michael Gruninger, June 1996 *Ontologies: Principles, Methods and Applications* Volume 11 Number 2.

[11] Stanford Center for Biomedical Informatics Research ,2011 *What is Protégé OWL?* http://protege.stanford.edu/overview/protege-owl.html , Copyright © 2011

[12] Yves Raimond, Samer Abdallah, 25th October 2007 ,"*The Event Ontology"*

[13] Federico Chesani, "*Semantic Web*", http://www-lia.deis.unibo.it/phd/materials/courses/Introduction%20to%20Knowledge%20Representation%20and%20the%20Semantic%20Web/IntroToSW.pdf 16 Febbraio 2010

[14] Gartner research published on February 28, 2002

[15] Conceptual Graphs http://www.jfsowa.com/cg/

[16] Conceptual Graph Examples http://www.jfsowa.com/cg/cgexampw.htm

[17] Robin Cover, *"Ontology Interchange Language"* http://xml.coverpages.org/oil.html

[18] DAML ontology http://www.ai.sri.com/daml/ontologies/

[19] Abílio Fernandes ,Tópicos em Bancos de Dados , May 2004 S*nobase*

[20] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder, Copyright © 2000, *The Chimaera Ontology Environment* American Association for Artificial Intelligence

[21] Nenad Stojanovic , Jens Hartmann and Jorge Gonzalez "The OntoManager – a system for the usage-based ontology management" Institute AIFB, University of Karlsruhe

[22] Jerry R.Hobbs, Feng Pan, Time Ontology in OWL ,W3C Working Draft 27 September 2006, latest version: http://www.w3.org/TR/owl-time , University of Southern California / Information Sciences Institute

[23] Marc Moens *Temporal Ontology and Temporal Reference*, Centre for Cognitive Science, University of Edinburgh

[24] Shervin Daneshpajouh , *Temporal Ontology*

[25] Weng J Y, Cohen P, Herniou M.Camera *Calibration with distortion models and accuracy evaluation*[J].IEEE Trans PAMI, 1992; 14(8) :965~980

[26] Baader et al., 2002, Horrocks and Sattler

[27] Pascal Hitzler, Markus Krotzsch, Sebastian Rudolph KI 2009 Paderborn. *Knowledge Representation for the Semantic Web.*

[28] Ontology Web Language (OWL) Qualips –Quality platform for Open Source Software, Creative Common Attribution-Share Alike 3.0 License, IST-FP5-034753

[29] James F. Allen, George Ferguson, July 1994 *Actions and Events in Interval Temporal Logic*, the University of Rochester, Computer Science Department, Rochester, New York.

[30] Grigoris Antoniou and Frank van Harmelen, "Web Ontology Language: OWL" , department of Computer Science, University of Crete, Department of AI, Vrije Universiteit Amsterdam

[31] P.Brusilovsky, Adaptive Hypermedia, 2011, *User Modeling and User Adapted* Interaction, 11,87-110.

[32] V. Devedži, 2006, *Semantic web and education* (Springer Science+Business Media, LLC).

[33] A. Gómez-Pérez & O. Corcho, 2002, *Ontology Languages for the Semantic Web*, IEEE Intelligent Systems, 17(1), 54-60.

[34] Boris Motik, Peter F. Patel-Schneider, 27 Oct 2009, *OWL 2 Web Ontology Language Direct Semantics*