


12-2007

Combining Geospatial and Temporal Ontologies

Kripa Joshi

Follow this and additional works at: <http://digitalcommons.library.umaine.edu/etd>

 Part of the [Databases and Information Systems Commons](#), and the [Geographic Information Sciences Commons](#)

Recommended Citation

Joshi, Kripa, "Combining Geospatial and Temporal Ontologies" (2007). *Electronic Theses and Dissertations*. 559.
<http://digitalcommons.library.umaine.edu/etd/559>

This Open-Access Thesis is brought to you for free and open access by DigitalCommons@UMaine. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of DigitalCommons@UMaine.

COMBINING GEOSPATIAL AND TEMPORAL ONTOLOGIES

By

Kripa Joshi

B. E., Institute of Engineering, Tribhuvan University, Nepal 2004

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

(in Spatial Information Science and Engineering)

The Graduate School

The University of Maine

December, 2007

Advisory Committee:

Kathleen Stewart Hornsby, Assistant Professor, Department of Geography,

University of Iowa, Advisor

Kate Beard-Tisdale, Professor of Spatial Information Science and Engineering

Max J. Egenhofer, Professor of Spatial Information Science and Engineering

© 2007 Kripa Joshi

All Rights Reserved

COMBINING GEOSPATIAL AND TEMPORAL ONTOLOGIES

By Kripa Joshi

Thesis Advisor: Dr. Kathleen Stewart Hornsby

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Master of Science
(in Spatial Information Science and Engineering)
December, 2007

Publicly available ontologies are growing in number at present. These ontologies describe entities in a domain and the relations among these entities. This thesis describes a method to automatically combine a pair of orthogonal ontologies using cross products. A geospatial ontology and a temporal ontology are combined in this work. Computing the cross product of the geospatial and the temporal ontologies gives a complete set of pair-wise combination of terms from the two ontologies. This method offers researchers the benefit of using ontologies that are already existing and available rather than building new ontologies for areas outside their scope of expertise. The resulting framework describes a geospatial domain over all possible temporal *granularities* or levels, allowing one domain to be understood from the perspective of another domain. Further queries on the framework help a user to make higher order inferences about a domain.

In this work, Protégé, an open source ontology editor and a knowledge base tool, is used to model ontologies. Protégé supports the creation, visualization and manipulation of ontologies in various formats including XML (Extensible Markup Language). Use of

standard and extensible languages like XML allows sharing of data across different information systems, and thus supports reuse of these ontologies. Both the geospatial ontology and the temporal ontology are represented in Protégé.

This thesis demonstrates the usefulness of this integrated spatio-temporal framework for reasoning about geospatial domains. SQL queries can be applied to the cross product to return to the user different kinds of information about their domain. For example, a geospatial term *Library* can be combined with all terms from the temporal ontology to consider *Library* over all possible kinds of times, including those that might have been overlooked during previous analyses. Visualizations of cross product spaces using Graphviz provides a means for displaying the geospatial-temporal terms as well as the different relations that link these terms. This visualization step also highlights the structure of the cross product for users. In order to generate a more tractable cross product for analysis purposes, methods for filtering terms from the cross product are also introduced. Filtering results in a more focused understanding of the spatio-temporal framework.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards my advisor Kathleen Stewart Hornsby for her support, guidance, and encouragement without which this thesis would have been impossible. I would also like to take this opportunity to thank the advising committee members Kate Beard and Max Egenhofer for their cooperation and encouragement.

I would like to express my appreciation to my professors and friends in the Department of Spatial Information Science and Engineering for their help and support during my graduate studies and their genuine interest in the success of my work. Special thanks to my professors for their valuable insight during the many presentations of my work. I am grateful to my friends in the department especially Kraig King, Daniel Bowe, Maria Vasardani, Hae-Kyong Kang, and Susan Elston for their help, feedback and ideas. My friends in Nepal and USA deserve a special mention for their encouragement and suggestions. I would also like to thank the staff in our department for their help throughout my studies.

I would also like to acknowledge the support of the National Geospatial-Intelligence Agency (grant number HM1582-05-1-2039) for funding this research.

Finally I would like to thank my parents and my brother for their love and support, and encouragement in my endeavors.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER	
1 INTRODUCTION	1
1.1 Research Motivation and Hypothesis	5
1.2 Scope of Thesis	7
1.3 Research Approach	8
1.4 Major Contributions.....	10
1.5 Intended Audience	11
1.6 Organization of the Remaining Chapters.....	11
2 MODELING WITH ONTOLOGIES	13
2.1 Defining Ontologies.....	13
2.2 Modeling Ontologies using Protégé.....	16
2.3 Ontology-Ontology Integration	17
2.3.1 Merging.....	18
2.3.2 Aligning	21
2.3.3 Mapping	22
2.3.4 Combining.....	23
2.4 Summary	26

3	COMBINING A GEOSPATIAL ONTOLOGY WITH A TEMPORAL ONTOLOGY	28
3.1	The Geospatial Ontology	29
3.2	The Temporal Ontology.....	34
3.3	Representing the Ontologies in Protégé.....	37
3.3.1	Understanding OWL Ontologies	37
3.3.2	The Protégé-OWL Editor.....	41
3.4	Computing the Cross Product	45
3.4.1	Parsing Terms from the Ontologies	48
3.4.2	Importing into an RDBMS.....	49
3.4.3	Applying SQL Operations to Compute the Cross Product	51
3.5	Summary	57
4	QUERYING THE CROSS PRODUCT.....	58
4.1	Queries Highlighting a Single Term	59
4.2	Queries Highlighting a Specific Combination of Terms	63
4.3	Recursive Queries	65
4.4	Summary	72
5	VISUALIZING THE CROSS PRODUCT.....	74
5.1	Visualizing the Cross Product using Graphviz	75
5.1.1	About Graphviz.....	75
5.1.2	Building a .dot file	76
5.1.3	Creating an Image File.....	80
5.2	Visualizing Refinements of Cross Product.....	83
5.3	Summary	87

6	FILTERING CROSS PRODUCTS	89
6.1	Filtering Nodes of Coarse Granularity.....	90
6.2	Filtering Nodes of Specific Granularity.....	92
6.3	Composition of Relations	93
6.3.1	Filtering Terms with No Parent Term but Two or More Child Terms	94
6.3.2	Filtering Terms with One or More Parent Term but No Child Term	95
6.3.3	Filtering Terms with One or More Parent as well as Child Terms.....	96
6.3.4	Composition Rules.....	97
6.4	Summary	100
7	CONCLUSIONS AND FUTURE WORK	102
7.1	Summary of the Thesis	102
7.2	Major Results.....	104
7.3	Future Work.....	106
	BIBLIOGRAPHY	108
	BIOGRAPHY OF THE AUTHOR	113

LIST OF TABLES

Table 6.1	Composition rules for ontological relations	99
-----------	---	----

LIST OF FIGURES

Figure 1.1	Schematic diagram showing the steps involved in the implementation of combination of two ontologies using cross products.....	9
Figure 2.1	Classes and relations of an ontology.....	15
Figure 2.2	Merging ontologies	20
Figure 2.3	Combining domain generalization graphs	24
Figure 3.1	Geospatial ontology.....	33
Figure 3.2	Temporal ontology.....	35
Figure 3.3	Subclass Explorer frame in OWLClasses tab in Protégé-OWL editor.	42
Figure 3.4	Class Editor frame in OWLClasses tab in Protégé-OWL editor.....	43
Figure 3.5	Create Restriction window in Protégé-OWL editor.....	44
Figure 3.6	Using Protégé-OWL editor to export an ontology in XML.....	45
Figure 3.7	Computing a cross product	46
Figure 3.8	Relations with extracted attribute values.....	50
Figure 3.9	Relations showing lists of classes	51
Figure 3.10	Sample tuples from relation obtained by the equi-join of <i>GeospatialXTemporalC</i> and <i>TemporalClasses</i>	54
Figure 3.11	Sample tuples from <i>CrossProduct</i> relation.....	56
Figure 4.1	Query result depicting <i>Library</i> combined with all temporal terms.....	60
Figure 4.2	Query result depicting all geospatial terms combined with <i>Morning</i>	62
Figure 4.3	Part of query result showing only <i>componentOf</i> relations.....	64

Figure 4.4	Query result displaying tuples where either of the two pairs of terms linked by a <i>Relation</i> in the <i>CrossProduct</i> is (<i>Building</i> , <i>Morning</i>).....	65
Figure 4.5	Query result displaying <i>Library</i> during <i>DayTime</i>	66
Figure 4.6	Query result displaying <i>LandArea</i> and all its subclasses at <i>Weekend</i> ..	67
Figure 4.7	Query result displaying all geospatial terms at <i>DayTime</i> and all its subclasses.....	69
Figure 4.8	Query result displaying <i>LandArea</i> and all its subclasses at all times.....	70
Figure 4.9	Query result displaying <i>LandArea</i> and all its subclasses at <i>DayTime</i> and all its subclasses.....	71
Figure 5.1	Undirected graph.....	77
Figure 5.2	Directed graph.....	78
Figure 5.3	User interface for <i>dot</i> layout engine in Graphviz.....	81
Figure 5.4	Displaying some of the nodes (geospatial-temporal pairs) and edges (relations) in a cross product.....	82
Figure 5.5	Visual representation of entire cross product using Graphviz. Icons represent the pairs of geospatial-temporal terms.....	82
Figure 5.6	Part of the cross product displayed using Graphviz.....	83
Figure 5.7	Graphical representation of <i>Library</i> with all temporal terms.....	85
Figure 5.8	Graphical representation of <i>Building_Morning</i> along with all geospatial-temporal terms that are directly related.....	86

Figure 5.9	Graphical representation of <i>Library</i> during <i>DayTime</i> (<i>DayTime</i> along with all its subclasses).....	86
Figure 6.1	Graphviz visualization of cross product.....	94
Figure 6.2	Part of the cross product showing the geospatial-temporal term <i>BookStore_AcademicSemester</i> and the relations linking it to other terms in the cross product.....	97
Figure 6.3	Part of the cross product after filtering <i>BookStore_AcademicSemester</i> ..	98
Figure 6.4	Applying composition rules in Table 6.1 to remove the discontinuity in the cross product due to filtering of the term <i>BookStore_AcademicSemester</i>	100

Chapter 1

INTRODUCTION

Ontologies describe the entities in a domain and the relations among these entities (Genesereth and Nilsson, 1987). In general, ontologies play an important role for knowledge representation, database design, information retrieval, and the semantic web, where they are used as information engineering tools, for taxonomic reasoning and for first order logical inference. This thesis describes an *automated* method to combine a pair of ontologies from different domains and develop tools for querying and retrieving information about a geospatial domain or a temporal domain. Combining ontologies allows experts to take an existing ontology from areas, perhaps outside their scope of expertise, and combine it with an ontology from their own field in order to understand one domain from the perspective of another.

Numerous public ontologies exist that can be shared freely among different users. Some of the publicly available ontologies include SUMO (<http://www.ontologyportal.org/>), OpenCyc (<http://www.opencyc.org/>), and NASA's SWEET ontology (<http://sweet.jpl.nasa.gov/ontology/>). Suggested Upper Merged Ontology (SUMO) combine terms from *upper ontologies* and *domain ontologies* together. An upper ontology describes very general categories of entities that are common to all domains. They are limited to entities that are meta, generic, abstract and philosophical, and therefore are general enough to address a broad range of domain areas

(<http://suo.ieee.org/>). OpenCyc, which contains hundreds of thousands of terms along with millions of assertions relating the terms to each other, also forms an upper ontology. NASA's Semantic Web for Earth and Environmental Terminology (SWEET) is another project that provides a common scientific framework for various earth science initiatives. In this research, a geospatial and a temporal ontology, both drawn from SUMO, are combined to create a spatio-temporal perspective of a geospatial domain.

Ontologies have been developed for numerous domains and many are publicly available and are reusable. Some examples of domain ontologies are the Dublin Core ontology for documents and publishing (<http://dublincore.org/>), the Gene Ontology for genomics (<http://www.geneontology.org/>), a transportation ontology from SUMO, and a biosphere ontology from NASA's SWEET ontology. The transportation ontology, for example, has classes *Ship*, *CargoShip* and *PetroleumTankerShip* represented in the ontology (<http://www.ontologyportal.org/>). *Ship* refers to the class of large *WaterVehicle* used for travel on oceans, seas, or large lakes and is the parent class of *CargoShip*. *CargoShip* is the subclass of *Ship* that transports goods in exchange for payment. *CargoShip* includes ships that carry all kinds of cargo, including oil and bulk products as well as packaged, palletized, or containerized goods. *CargoShip* subsumes class *PetroleumTankerShip*. The biosphere ontology has entities such as *Canopy*, *Vegetation* and *Plant* (<http://sweet.jpl.nasa.gov/ontology/>). *Canopy* is one variety of *Vegetation* consisting mainly of the tallest layer of trees in the forest. *Plant* refers to a major group of living things and serves as the more general *superclass* of *Vegetation*.

Ontologies not only capture categories of entities recognized for a domain, but at the same time describe relations that link classes including taxonomic relations such as

isA, mereological relations including *componentOf*, and topological relations, for example, *containedIn* (Genesereth and Nilsson, 1987). An *isA* relationship defines a hierarchy over the classes of entities and provides the basis for the inheritance of properties (Brachman, 1983). This affords ontologies a *multi-granular* aspect where entities in a domain are modeled over different levels of detail. The *componentOf* relation defines a part-whole relation between a component class and its integral class (Winston *et al.*, 1987). The *containedIn* relation describes classes of objects that are spatially enclosed within another object (Egenhofer, 1993).

Two or more ontologies, either from the same domain or from different domains, can be combined together. For example, ontologies from the *same domain* can be *merged* at a common class to form a new ontology that is more complete and extensive (Corbett, 2003; Klein, 2001). It is also possible to *align* ontologies that refer to the same domain in order to assure communication among different users and applications. Aligning ontologies occurs when semantically-related entities from different ontologies need to be identified. As a result of alignment two or more ontologies are brought into mutual agreement, making them consistent and coherent (Klein, 2001; Duckham and Worboys, 2007). Alignment of ontologies supports semantic interoperability (Hughes and Ashpole, 2004). Ontologies from *different domains* can also be combined to generate a new framework that incorporates aspects of each domain.

The ontologies, in this thesis, are represented using Protégé, an open source ontology editor (<http://protege.stanford.edu/>). The Protégé platform supports two main ways of modeling ontologies, one of them being the Protégé-OWL editor. The Protégé-OWL editor, used in this work, enables users to build ontologies in W3C's Web

Ontology Language (OWL). OWL is based on the Extensible Markup Language (XML). Languages including OWL and XML make it possible for information contained in ontologies to be processed and interpreted by machines.

In this thesis, we combine a geospatial ontology with a temporal ontology by computing the cross product of the terms from both these ontologies. The geospatial ontology used in this thesis describes entities in a university campus including *Library*, *Dormitory*, *StudentUnion*, and *SportsFacility*. Terms from a temporal domain can be similarly captured in a temporal ontology. The temporal ontology contains general temporal classes such as *DayTime*, *Week*, and *Noon*, as well as those temporal terms that are relevant to an academic setting, for example, *AcademicYear* and *ClassDay*. Both the geospatial ontology and the temporal ontology used in this thesis contain classes that are drawn from SUMO. Combining these ontologies gives us a temporal perspective of a geospatial domain, that is, it describes all of the geospatial entities in a domain at all times. For example, a university campus can be understood from the perspective of students who rush to their classes in the *morning*, and administrative and other staff who find themselves busy at the start of an *academic year*. Faculty members are occupied during *class days* as well as during an *exam week*. *Academic buildings* and *bookstores* are usually empty in the *evenings*, while the *library* and the *gymnasium* are crowded throughout the *day time*.

Combining ontologies based on computing the cross product is beneficial for many domains. This method of combining ontologies finds use, for example, in the field of molecular biology. Hill *et al.* (2002) describes a method to expand the Gene Ontology by combining concepts from two orthogonal vocabularies to generate a larger, more

specific vocabulary. In a biological context, orthogonal vocabularies are those vocabularies whose terms are unrelated (Hill *et al.*, 2002). An ontology of generic developmental process terms and an ontology of anatomical terms are combined to generate a species-specific developmental process vocabulary. For example, it is possible to describe mouse heart development based on the combination of existing anatomical and developmental process vocabularies. In the same way, a *HumanActivity* ontology, based on NASA's SWEET ontology, can be combined with an *EarthRealm* ontology, also from NASA's SWEET ontology, such that all types of human activities are considered over all entities in the earth realm. Human activities such as *industrialization* can be represented with respect to the *terrestrial ecosystem*, *marine ecosystem* and *atmosphere layer*. Similarly, *contamination*, a human activity, can be combined with *underground water* or *surface water*. The result of this combination of ontologies allows, in this case, an environmentalist to exhaustively consider all types of human activities that might have an effect on any type of earth realm.

1.1 Research Motivation and Hypothesis

The goal of this thesis is to develop methods for automatically combining a geospatial ontology with a temporal ontology. The main advantage of this approach is the ability to use available ontologies to combine knowledge about two different domains and compute a spatio-temporal reasoning framework that provides a perspective of a geospatial domain over all possible times. Providing a method to combine ontologies makes it possible for domain experts to focus on their area of expertise. Instead of developing a

completely new vocabulary for a domain that may be unfamiliar an already existing ontology can be automatically combined with another domain ontology to obtain a new framework.

The hypothesis to be tested in this thesis is, *A multi-granular, unified framework results from taking the cross product of a pair of orthogonal ontologies.*

As an outcome of this research, we demonstrate how such a multi-granular and unified reasoning framework is created by combining ontologies. In this thesis, the combination of ontologies is computed using a relational database approach. SQL operations, designed for the retrieval and management of data in relational database management systems, database schema creation and modification, and database object access control management, provides a straightforward method to compute the cross product of terms from a pair of ontologies.

This thesis illustrates the usefulness of this cross product for reasoning about geospatial domains. SQL queries can be applied to the cross product to return to the user different kinds of information about their domain. For example, SQL queries can be written to return information about a *SportsFacility* entity from the campus ontology at all possible times, or a *Library* can be observed at particular times (e.g., *DayTime* or *NightTime*). The cross product of the geospatial and the temporal ontologies gives a *complete set of pair-wise combinations of terms* from the two ontologies. Once the cross product has been completed, filtering terms from the cross product allows us to get a more focused understanding of the framework, and makes it easier to analyze and understand. Filtering eliminates terms that are not required for a particular purpose, such

as terms that are of coarse granularity (and therefore are most general) or those that contain a specific combination of geospatial-temporal terms.

1.2 Scope of Thesis

A geospatial ontology and a temporal ontology are used as the two base ontologies in this thesis. These ontologies have been previously developed by domain experts and are drawn from SUMO (<http://www.ontologyportal.com>). As part of this work, a check is performed to determine whether there are any classes common to both ontologies. In the case where no classes are common (i.e., the ontologies are orthogonal), the cross product will produce unique combinations of terms. In the case where no classes are common, the cross product will produce unique combinations of terms. Combination of classes from orthogonal ontologies provide a robust representation of relevant terms and an opportunity for evaluation of hypothetical combinations (Hill *et al.*, 2002). If any classes should be common to the pair of ontologies, however, some combinations like *Building_Building* or *Morning_Morning*, in the cross product may be implausible. Since we are combining a geospatial ontology with a temporal ontology, we expect no classes in common and testing confirms that this is indeed the case. The focus of this thesis is to build a new framework from the combination of ontologies from geospatial and temporal domains. The resulting framework allows users to understand one domain from the perspective of another. In this thesis, a geospatial ontology is explored over a range of temporal perspectives. Multiple inheritance is not supported for the base ontologies. Each child class has only one parent class. Only pair-wise combinations of ontologies are

considered. Combining more than two ontologies using cross products is possible, but the focus here has been on developing methods for pair-wise combinations only. Future work could consider more complex combinations involving more ontologies.

1.3 Research Approach

There are three main steps involved in combining a pair of ontologies using cross products. These steps are: parsing the OWL ontologies for extracting values of the classes, subclasses and relations; importing the values of classes and relations into a relational database; and using SQL operations to compute the cross product (Figure 1.1). The base ontologies, geospatial and temporal, are modeled as OWL files using Protégé. These files are parsed such that the values of the terms (e.g., *Library*, *StudentUnion*, *Road*) and the relations (e.g., *isA*, *componentOf*, *containedIn*) in the ontologies are extracted. Parsing the OWL ontologies requires an understanding of the structure of the OWL file that involves use of different tags to describe classes, subclasses and relations. The ontology parser has been implemented in Visual C#.NET. Once the parsing is completed, the values are then imported into a relational database management system. A relational database approach has been followed in this thesis for combining two domain ontologies in order to take advantage of the search tool a relational database provides in the form of SQL operations. The combination of the pair of ontologies uses a sequence of SQL operations, including, Cartesian products, joins and unions, to obtain the cross product of the geospatial and temporal ontologies in the form of a relation.

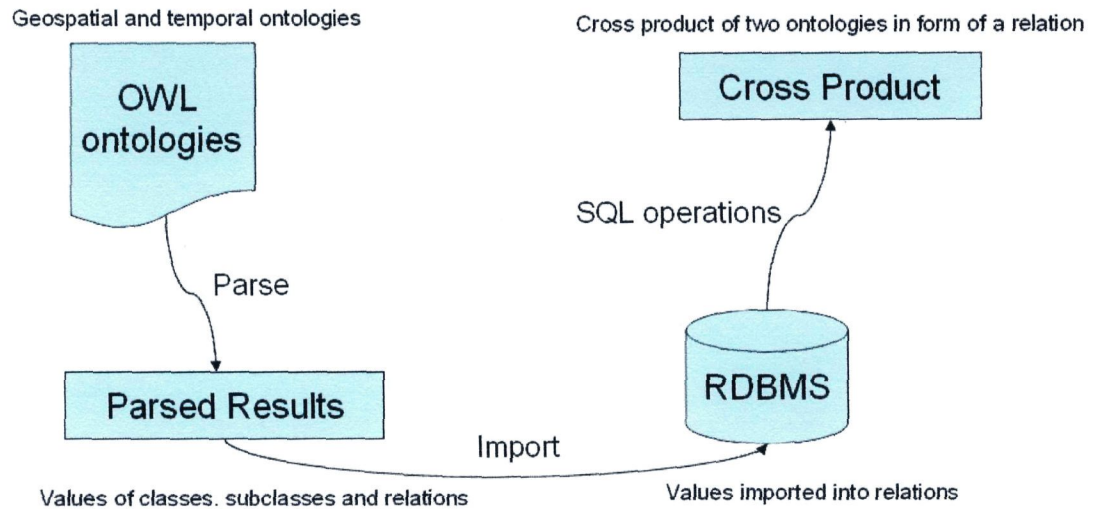


Figure 1.1 Schematic diagram showing the steps involved in the implementation of combination of two ontologies using cross products

By applying SQL queries, the cross product can be analyzed from different perspectives. For example, a single geospatial entity can be analyzed over a range of temporal perspectives, for example, an *academic building* over all times. Another possibility is exploring all geospatial entities at a particular time. For example, *all* geospatial terms can be observed at *night time*. It is also possible to investigate the cross product according to the type of relation that exists between any two classes, for example, obtain all pairs of terms linked by an *isA* relation. More complex queries allow users to find terms that are related to a specific geospatial and temporal term, for instance, *library at morning*. Recursive queries are also possible and the implementation of these types of queries requires the use of a programming language (e.g., Visual C#.NET). For example, a geospatial term *sports facility* can be observed at all times of the day such that *day time* involves a set of times including *morning, afternoon, mid-afternoon, and early morning*,

among others. *Day time* has child classes *morning* and *afternoon*, and in the same way *morning* is the parent of *early morning* and *afternoon* subsumes *mid-afternoon*.

In this thesis, we also explore methods to visualize the cross product relation in order to better understand parent-child relationships between any two terms in the cross product. The cross product is visualized in the form of a graph using the open source graph visualization software called Graphviz (<http://www.graphviz.org/>). Graphviz takes descriptions of graphs in simple text language and makes diagrams in different image formats such as GIF, JPEG, and TIFF. Each pair of terms in the cross product is denoted by a node in the Graphviz visualization. Each relationship that links any two pairs of ontological terms together is denoted by an edge. In the same manner, the results obtained after applying SQL queries to the cross product can also be visualized using Graphviz.

It is possible that not every geospatial-temporal term in the cross product is relevant for a domain and in this case, the cross product can be *filtered*. Filtering can be carried out on the cross product and refers to removal of terms from the cross product. The result is a reduced space that is more tractable for analysis and suits users needs better.

1.4 Major Contributions

In this thesis, we show how the class names or *terms* from a geospatial ontology can be extracted and combined with the terms from a temporal ontology using automated methods to capture *all possible combinations of terms* derived from the two base

ontologies. Specifically, cross products are computed to provide an integrated spatio-temporal reasoning framework that is multi-granular and that presents information about the domain over all the types of time represented in the temporal ontology. This new framework can be browsed, queried and visualized for more comprehensive analysis. The terms in the cross product are a combination of geospatial and temporal terms and can be used, for example, for making higher-order inferences about a domain, such as retrieving the geospatial term *CampusRoad* over all possible times. This thesis also presents a method to eliminate specific tuples from the cross product, which results in a more tractable spatio-temporal space.

1.5 Intended Audience

The intended audiences for this thesis are researchers and scientists working in the field of geographic information science, information systems and computer science, and also artificial intelligence. Researchers interested in semantic web applications will also benefit from the work in this thesis.

1.6 Organization of the Remaining Chapters

The rest of this thesis is structured as follows: Chapter 2 discusses related work on modeling ontologies and different approaches to combining ontologies. An example scenario based on combining an ontology for a university campus with a temporal

ontology is presented in this thesis to illustrate our approach for computing cross products of ontological terms.

Chapter 3 introduces the geospatial and temporal ontologies and their representations in OWL, using Protégé, followed by the approach for generating cross products using a relational database. Cross product computation is used to combine the geospatial ontology with the temporal ontology. This cross product computation involves parsing the values of classes and relations from the XML ontologies, importing the parsed results into a relational database and finally applying a sequence of SQL operations to obtain the cross product.

In the Chapter 4, different types of SQL queries that can be performed on the cross product are discussed. A SQL query that searches for a specific geospatial term, for example, *Library* over all times is one such example.

Visualization of the cross product results as well as the refinement using Graphviz, an open source graph visualization software, is presented in Chapter 5.

Chapter 6 discusses filtering the cross product. Two cases that call for filtering of terms from a cross product are presented in this chapter along with the various structural issues that visualization of the filtered cross product reveals. A theoretical framework for determining rules to solve the structural issues in the filtered cross product is discussed in this chapter.

Conclusions and future work are discussed in the final chapter of the thesis.

Chapter 2

MODELING WITH ONTOLOGIES

This thesis focuses on combining a pair of ontologies using cross products. The ontologies, representing different domains, are combined in order to analyze one of the domains, in our case, a geospatial domain, from the point of view of another, for instance, a temporal ontology. In this chapter, we discuss some of the fundamental aspects of ontologies that are especially relevant for this work, and the significance of ontologies to the field of GIScience. Related research that has examined the integration of one ontology with another ontology, where ontologies describe the same domain as well as different domains, is also discussed in this chapter.

2.1 Defining Ontologies

What we see in the world around us can be categorized into different groupings and these different classifications are represented in ontologies. Ontology has its roots in philosophy, but has found extensive application in numerous other diverse areas, including artificial intelligence, the semantic web, software engineering, biomedical informatics, and GIScience as a form of knowledge representation about the world or some parts of it (Agarwal, 2005). Given the widespread use of ontologies, there are numerous ways in which ontologies are defined and described. Gruber (1993) defines

ontology as “an explicit specification of a conceptualization.” Sowa (2000) defines ontology as a catalog of the types of things that are assumed to exist in a domain of interest *D* from the perspective of a person who uses a language *L* for the purpose of talking about *D*. An ontology can also be defined as the manifestation of a shared understanding of a domain that is agreed between a number of agents, where such agreement facilitates accurate and effective communications of meaning that in turn leads to other benefits such as inter-operability, reuse and sharing (Agarwal 2005). These definitions illustrate that ontologies capture categories of entities recognized for a domain as well as the relations that link the entities together in different ways. In an ontology, the entities are grouped into *classes* based on common *attributes* and these classes are linked by *relations* (Figure 2). Classes are abstract groups, sets or collections of entities, for example, car, or building. Subclasses are more specific versions of their superclass. For example, a residential building is a subclass of building and takes over or *inherits* attributes and functions of building class. Relations capture the ways classes are associated with one another, e.g., car *is_a* vehicle, a steering wheel is *part_of* a car. Relations are used in ontologies to determine the semantic content of the terms in the ontology (Neuhaus and Smith, 2007). Attributes describe properties, features or characteristics that a class of entities can have and share, and add richness to ontologies.

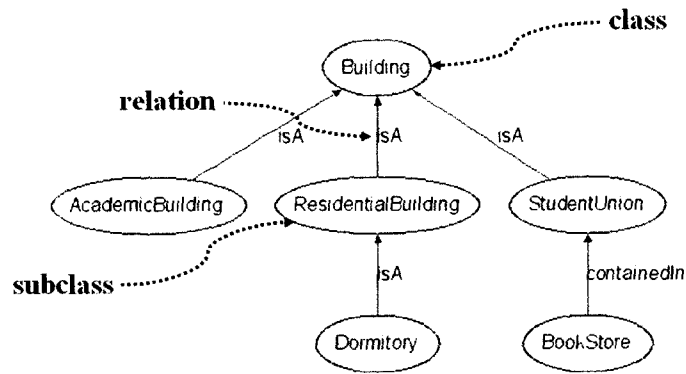


Figure 2.1 Classes and relations of an ontology

One of the challenges in building a geographic information system is being able to integrate geographic information of different kinds at various levels of detail (Fonseca *et al.*, 2002). The object-based data model and field-based data model are two of the most widely accepted data models describing the geographic world (Couclelis 1992; Goodchild 1992). The object model represents the world as a surface occupied by discrete, identifiable entities, with a geometrical representation and descriptive attributes, e.g., human-built features, such as roads and buildings. The field model views geographic reality as a set of spatial distributions over geographic space, for instance, climate, vegetation cover, and geology (Fonseca *et al.*, 2002; Fonseca *et al.*, 2006). Object models as well as field models are very generic conceptual models, without support for specific semantics for different types of spatial data (Fonseca *et al.*, 2002). This issue has led many researchers to consider the use of ontologies as a means of knowledge sharing among different user communities to improve interoperability among different geographic databases (Smith and Mark 1998; Fonseca and Egenhofer 1999). Ontologies find various applications in the field of GIScience and three main applications include:

- ontologies for knowledge generation,
- ontologies for domain specification, and
- ontologies for information system development (Agarwal, 2005).

The primary ontology initiatives in GIScience are aimed either at developing a comprehensive geospatial ontology or at modeling certain specialized tasks (Agarwal, 2005). The philosophical approach in GIScience, on the other hand, aims at finding an upper-level ontology for geo-spatial domains that can form a unifying framework for all concepts shared within the geographic community (Agarwal, 2005).

2.2 Modeling Ontologies using Protégé

For many public ontologies, and for the work in this thesis, Protégé is used to represent the ontologies. Protégé is a free, open-source ontology editor and knowledge-base framework. The Protégé platform supports two main ways of modeling ontologies via the Protégé-Frames and Protégé-OWL editors. In this research, the Protégé-OWL editor is used, which enables users to build ontologies for the semantic web. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. Protégé is supported by a strong community of developers and academic, government, and corporate users, who are using Protégé for knowledge solutions in areas as diverse as biomedicine, intelligence gathering, and corporate modeling (<http://protege.stanford.edu/>).

In this thesis, using the graphical user interface that Protégé provides, classes and relations in an ontology are entered. Protégé provides a tool that then outputs the

ontologies in XML format. Expressing ontologies in XML allows them to be processed and interpreted by machines, and thus are suitable forms of representation for retrieving information from the ontologies or integrating ontologies.

2.3 Ontology-Ontology Integration

Integrating ontologies is a central topic for extending and sharing knowledge and has been a subject of interest for researchers and scientists, particularly those interested in semantic web applications. Ontologies from the same domain, as well as different domains, can be integrated with each other. Typically multiple ontologies are independently developed for the same domain calling for the integrated use of ontologies. In some cases, ontologies from diverse domains must be coalesced to build a more comprehensive domain of interest. Integrating ontologies is important for interoperability, a key concern in geographic information science for sharing knowledge (Riedemann and Kuhn, 1999; Harvey *et al.*, 1999; Agarwal, 2005; Fonseca *et al.*, 2006). Interoperability refers to the ability to set up a correspondence between entities in one system to entities in the other to allow the transfer of data and models between different systems (Fonseca *et al.*, 2006).

Where we use the term *integration* in this thesis as a general term for the amalgamation of ontologies, Klein (2001) introduces a number of terms including *merging*, *aligning*, *mapping*, and *combining* among others. *Merging*, *aligning* and *mapping* concerns integration of ontologies from the same domain, while *combining* deals with ontologies from diverse domains.

2.3.1 Merging

Merging refers to creating a new ontology from two or more existing ontologies from the same domain based on a common class. The overlapping parts of the ontologies may be physical or virtual (Corbett, 2003; Klein, 2001). Merging falls under the broader concept of knowledge conjunction (Corbett 2003). The conjunction of ontologies can take two forms: merging the ontologies into one new ontology (this matches Klein's definition of merging), or placing links between the ontologies to indicate semantic identities while continuing to maintain two separate ontologies. Creating a new, merged ontology has the advantage of maintainability. Keeping the ontologies separate has the advantage that an owner of an ontology can "borrow" concepts from another ontology without the need of reorganizing one's own ontology (Corbett, 2003). Corbett (2003) describes an algorithm for merging two ontologies. Figure 2.2a shows a wildlife ontology and Figure 2.2b shows another ontology of the same domain. The algorithm begins with the user selecting a start node for the merge, which can be the top-most class or any other class in the ontology (e.g., *animal*). A check is then performed by comparing the name of each class of wildlife ontologies in Figure 2.2a and 2.2b to search for an exact match. If there is an exact match, then that class and everything subsumed by that type in both ontologies is copied into a new ontology (Figure 2.2).

Merging is an extension of the unification of conceptual graphs. The unification of two graphs contains neither more nor less information than the two graphs being unified. As described above, the merging of two ontologies begins with finding a common starting point on the two hierarchies (usually with the assistance of the user) and

then continuing outward from that point in a depth-first manner to find other matching points (Corbett, 2003 and 2004).

Several problems can arise when independently developed ontologies are used together. Mismatches of various types can occur. The first type of mismatch is a *language* or *meta-model level* mismatch, while the second level is the *ontology* or *model level* mismatch (Klein, 2001). A language level mismatch can also be termed a *non-semantic difference* describing the fact that the mismatch is a result of employing a different mechanism to define the classes and relations. Ontology level or semantic mismatch refers to the difference in the way a domain is modeled. Mismatches at the ontology level happen when two or more ontologies that describe (partly) overlapping domains are combined. These mismatches may occur when the ontologies are written in the same language, as well as when they use different languages. In addition, if the ontologies are not represented in the same language, a translation is often required.

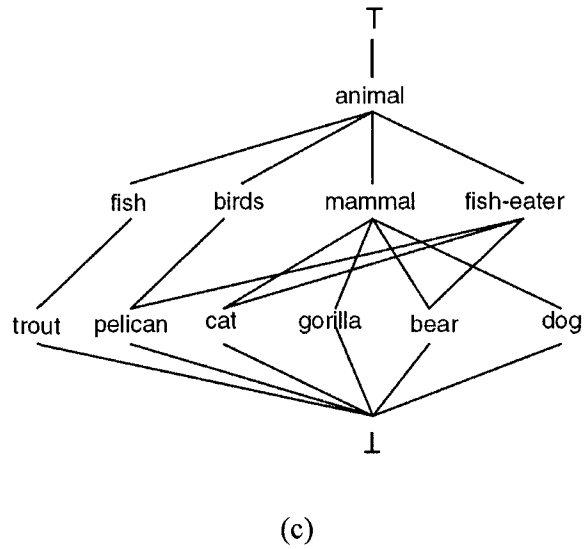
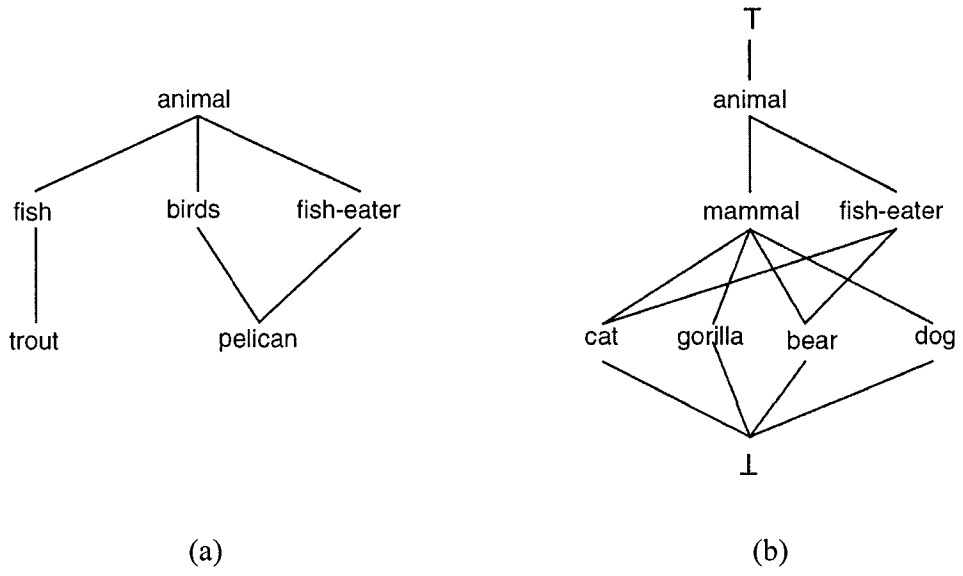


Figure 2.2 Merging ontologies. (a) An example of a wildlife ontology, (b) another ontology from the same domain, and (c) the merged ontology (from Corbett (2003))

2.3.2 Aligning

Aligning and mapping can be categorized together as cases of ontology matching. Ontology matching is commonly defined as a matter of dealing with semantic correspondences between terms in ontologies and thus refers to more specific activities such as mapping and aligning (Ceusters, 2006). Matching consists of dealing with *semantic correspondences* between the representational units (i.e., the single terms) of the individual ontologies. Ontology mapping is mostly concerned with the representation of correspondences between ontologies, while ontology alignment is concerned with the (semi-)automatic discovery of such correspondences (Brujin *et al.*, 2006).

Aligning brings two or more ontologies into mutual agreement, making them consistent and coherent (Klein, 2001; Duckham and Worboys, 2007). Ontology alignment involves the identification of semantically-related entities in different ontologies. The related entities can have an exact match, an approximate match, a null match, a superset match or a subset match (Cruz *et al.*, 2004). For example, if we have two ontologies for land use patterns, classes such as *industry*, *mining* and *manufacturing* in one ontology; these classes can be matched to *industrial sector*, *mining* and *mfg* in the second ontology (Cruz *et al.*, 2004). *Industry* and *manufacturing* have approximate matches in *industrial sector* and *mfg* while *mining* has an exact match.

Alignment refers to a *mapping* of entities and relations between two ontologies *A* and *B* that preserves the partial ordering by subtypes in both *A* and *B* (Sowa, 2000). If an alignment maps an entity or a relation *x* in ontology *A* to an entity or a relation *y* in ontology *B*, then *x* and *y* are said to be *equivalent*. The mapping may be *partial*: there

could be many entities in A or B that have no equivalents in the other ontology. Before two ontologies A and B can be aligned, it may be necessary to introduce new child classes or parent classes of entities in either A or B in order to provide suitable targets for alignment.

Aligning ontologies draws parallels with semantic database integration, where the schemas from different databases contain equivalent information but the attributes and the formats of the databases vary. Even though many databases contain database fields with equivalent information, both database field labels and formats of database entries may vary. In order to enable retrieval of data from several databases, the semantics of equivalent database fields have to be defined. One method of defining semantically equivalent attributes is maintaining meta-data (Kohler *et al.*, 2000).

2.3.3 Mapping

Mapping is similar to aligning in that it relates similar terms or relations collected by different information producers to each other by an equivalence relation (Klein, 2001; Zhou, 2003). Ontology mapping can be described where given two ontologies A and B , mapping one ontology to another means that for each entity in ontology A , a corresponding entity is searched for in B that has the same or similar semantics and vice versa (Ehrig and Sure, 2004). Kalfoglou (2003) describes ontology mapping as the task of relating the vocabulary of two ontologies in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected. Bouquet *et al.* (2004) defines ontology mapping in

similar vein as a formal expression that states the semantic relation between two entities belonging to different ontologies.

2.3.4 Combining

Combining refers to using two or more ontologies from *different* domains together such that the result can be used for a specific task (Corbett, 2003; Klein, 2001). Unlike other ontology-ontology integrations that result in a new ontology, combinations of two ontologies do not necessarily build a new ontology. While some methods for integrating ontologies, such as merging and aligning, take multiple ontologies from the same domain that have been developed independently and build a new ontology, in this thesis, ontologies from two *different* domains are combined to produce a framework that incorporates aspects of both domains. Combination of ontologies from different domains builds an extended framework that can be analyzed. Previous work on combining ontologies and domain generalization graphs has been done in the field of genomics and data mining, respectively.

In research on domain generalization graphs, methods have been employed to determine all possible generalizations that can exist within a concept hierarchy (Han *et al.* 1992, Han 1994, Pang *et al.*, 1996, Hamilton *et al.*, 1996). This work is motivated by the need to automate the knowledge discovery process for domains where more than one generalization is possible for a single attribute. As part of this earlier research, cross products of generalization paths were computed in order to generate the complete set of all possible combinations of generalizations of concepts represented in a domain

generalization graph (Hamilton *et al.* 1996; Hilderman, 1997a). Simple domain generalization graphs for attributes A and B are depicted in Figure 2.3a and Figure 2.3b respectively. Combining the nodes in the pair of domain generalization graphs produces the multi-attribute generalization graph (Figure 2.3c), which shows all possible levels of generalization for the domain associated with a set of attributes (Hamilton *et al.* 1996).

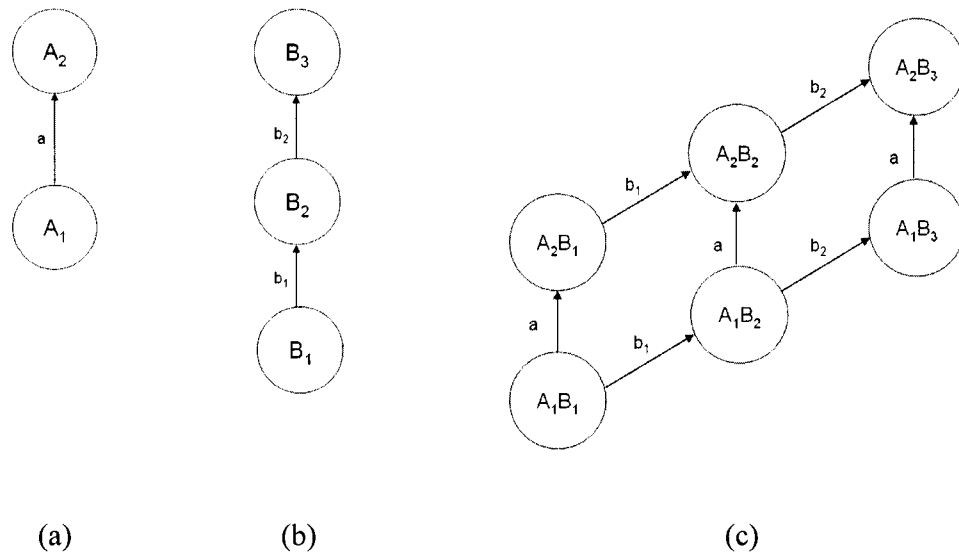


Figure 2.3 Combining domain generalization graphs (a) Domain generalization graph for attribute A , (b) domain generalization graph for attribute B , and (c) multi-attribute generalization graph (from Hamilton *et al.* 1996)

Computing a cross product involves taking a term from the first domain generalization graph (Figure 2.3a) and combining it with every term from the second (Figure 2.3b). The steps are repeated until all the terms from the first domain generalization graph are exhausted and a complete set of pair-wise combinations of each of the terms from the individual graphs have been created. For example, node A_1 from the first domain generalization graph (Figure 2.3a) is crossed with node B_1 to get A_1B_1

(Figure 2.3c). Node A_1 is subsequently combined with the remaining terms domain generalization graph B to return A_1B_2 and A_1B_3 respectively. This process is repeated for all other terms in the domain generalization graph for attribute A .

To view all possible generalizations from these domain generalization graphs, data visualization techniques are used (Hilderman *et al.*, 1997b). These data visualization techniques are useful for a domain expert to quickly and efficiently analyze the generalizations from many different perspectives. These ideas are extended with the present research, showing how these concepts can be applied to spatio-temporal domains where temporal and geospatial ontologies are combined using cross products to result in an expanded reasoning framework.

In a similar manner in the Gene Ontology (GO) project (<http://www.geneontology.org/>), terms from two orthogonal biological vocabularies are combined with the goal to generate a larger, more specific and complete vocabulary that includes particular aspects from each of the two parent ontologies. In the GO project, a mouse heart anatomical ontology is combined with a developmental process ontology using cross products to describe all processes involved in the development of all anatomical parts of the heart (Hill *et al.*, 2002). The result describes all possible processes that can occur in combination with an anatomical structure, including some that may not have been discovered experimentally, or some that may not exist. For this research community, structured vocabulary development enhances the management of information in biological databases. A modified existing ontology or a completely new ontology can be combined with another ontology using cross products in a meaningful way as long as they are orthogonal. Combining ontologies is more advantageous than constructing an

ontology of, for example, heart development, using conventional methods of GO expansion, given the fact that domain experts can create new vocabularies in their own field without attempting to describe domains outside their area of expertise (Hill *et al.*, 2002).

In this thesis, terms from a pair of ontologies are combined using cross products. The cross product provides a framework that includes every possible combination of terms from the two base or input ontologies. The resulting framework is useful for making higher order inferences about a geographical domain.

2.4 Summary

This chapter discusses some of the key topics that underlie this research. Ontologies are introduced and varying definitions of ontologies are presented to illustrate the breadth of use of this construct. Ontologies and their significance in the field of GIScience are also discussed in this chapter. Previous work on the integration of two or more ontologies from the same domain as well as different domains is covered in this chapter. Integration of ontologies from the same area involves processes like *merging*, *aligning* and *mapping* while those that involve ontologies from diverse domains use *combining*. Computing the cross product is one of the ways to achieve such a combination and this method has been utilized in previous studies involving domain generalization graphs and biological ontologies. The foundation of this related work forms the basis for the work in this thesis. Existing ontologies from different domains, modeled in XML, are used as input and the values of classes and relations extracted. The extracted terms are then combined to

compute the cross product of terms. The steps involved in this process will be discussed in the next chapter.

Chapter 3

COMBINING A GEOSPATIAL ONTOLOGY WITH A TEMPORAL ONTOLOGY

In this thesis, *classes* or *terms* extracted from a pair of ontologies (a geospatial ontology and a temporal ontology) are automatically combined using cross products. The resulting framework is a fused spatio-temporal space that can be further analyzed to make higher-order inferences about the domain. In this chapter, the geospatial ontology and the temporal ontology are introduced and their representations in XML using Protégé is discussed. The ontologies in XML are processed such that the terms are extracted and imported into a relational database. Once the terms have been imported, SQL operations are applied to derive the cross product. These procedures are described in detail in this chapter.

Ontologies consist of different domain-based classes, with attributes that refer to the properties, features, or characteristics of a class that may be inherited by further subclasses. Each class is assumed to have no more than one superclass (i.e., multiple inheritance is not supported). Different relations link the classes together. The three main relations considered in this work are *isA*, *componentOf* and *containedIn*. The *isA* relation refers to cases of class inclusion, and is determined based on the similarities of one class member to other members of the class with respect to one or more intrinsic attributes, either physical or functional (Winston *et al.*, 1987). For example, a car *isA* vehicle. The

isA relationship defines a hierarchy over the classes of entities and provides the basis for the inheritance of properties (Brachman, 1983). A *componentOf* relation is a meronymic relation that captures the link between a component class and its integral class. Component classes typically bear a specific structural or functional relation to one another and to the wholes or integral classes to which they are related (Winston *et al.*, 1987). The *containedIn* relation describes cases of spatial inclusion. This relation describes classes of objects that are spatially enclosed within another object. If object *A* is contained within another object *B*, then the entire region occupied by *A* is also occupied by *B* (but not *vice versa*) nor do they possess any intersecting boundaries (Egenhofer, 1993). Ontologies often include additional types of relation that further refine the semantics they model. For example, a temporal ontology can have a *before* relation (e.g., *Sunset* occurs *before* *EarlyMorningHour*).

3.1 The Geospatial Ontology

In this thesis, a geospatial ontology is used as one of the example ontologies. The prototypical geospatial ontology consists of classes relevant to a university campus (Figure 3.1). This ontology is drawn from the Suggested Upper Merged Ontology (SUMO). SUMO is a candidate standard upper ontology for IEEE (<http://suo.ieee.org/index.html>) and the terms in SUMO have been mapped to the WordNet lexicon (<http://wordnet.princeton.edu/>). This mapping acts as a natural language index to the concepts in the ontology, allowing a user to retrieve all SUMO concepts that are related to natural language terms of interest and leading to easier data modeling with

the ontology. The mapping process also functions as a completeness check on SUMO (Niles and Pease, 2003). SUMO is written in SUO-KIF language. SUO-KIF is Standard Upper Ontology Knowledge Interchange Format, which is a language designed for use in the authoring and interchange of knowledge (Pease, 2004).

Out of a total of thirty-one geospatial classes, sixteen classes introduced in the geospatial ontology have been drawn from SUMO. This ontology also includes classes that have been added in order to more closely model a university campus. From the remaining classes, nine have been derived from WordNet, while the remaining six are based on classes of entities that can be found on the University of Maine campus. In fact, all the classes model entities that are commonly found on a university campus. The classes that do not exist in SUMO but are a part of WordNet are *ParkingSpace*, *Garden*, *Gymnasium*, *AthleticField*, *Track*, *BaseballDiamond*, *ServiceBuilding*, *DiningCommons*, *ConcertHall*, and *BookStore*. Classes *CampusObject*, *AcademicBuilding*, *DiningCommons*, *RecyclingCenter*, and *StudentServiceCenter* meanwhile have been drawn from a typical university campus. The *CampusObject* class has been introduced as an upper-level class for the geospatial ontology. All the classes that *CampusObject* subsumes exist within a university campus.

With the introduction of *CampusObject* class, a new relation *isA* has been introduced to link the class with *Object* class. *Artifact*, a class in SUMO is subsumed by *CorpuscularObject*, which in turn is subsumed by *SelfConnectedObject*, which is a subclass of *Object*. In the geospatial ontology, using the transitive property of the *isA* relation (subsumption relation), *Artifact* is presented as a subclass of *CampusObject* and hence also of *Object*. In the same manner, using the transitive property, *GeographicArea*

is depicted as a subclass of *CampusObject*. In SUMO, *GeographicArea* is subsumed by *Region* and *Region* is subsumed by *Object*, hence *CampusObject*. In SUMO, *Road* is subsumed by *LandTransitway*, which is then subsumed by *LandArea*. Using the transitive relation, *Road* is shown as a subclass of *LandArea*. Classes *BusStop* and *ParkingLot* are subclasses of *GeographicArea* and *StationaryArtifact* respectively, but both have been depicted as being subsumed by *LandArea*. *Garden*, a WordNet term, is a subclass of *CultivatedLandArea*, which in turn is subsumed by *LandArea*. Thus *Garden* holds an *isA* relation with *LandArea* in the geospatial ontology. *AthleticField* is subsumed by *LandArea*, but in the geospatial ontology, an *isA* relation holds between *AthleticField* and *SportsFacility*, given the fact that depicting *AthleticField* as a subclass of *SportsFacility* is more meaningful in a university campus domain. Similarly, *BaseballField* and *Track* are depicted to hold an *isA* and *containedIn* relation with *AthleticField* instead of subsumption relations with *SportsFacility* and *StationaryArtifact* respectively. *BaseballDiamond*, a WordNet term, is subsumed by *Region* but in the geospatial ontology, this class is presented as a *componentOf BaseballField*. *ConcertHall* and *Museum* are presented as *componentOf EntertainmentBuilding* instead of being subsumed by *Building*.

The most general class in the geospatial ontology is the *Object* class. *Object* refers to any entity that has a distinct existence, is tangible and can be perceived by the senses. *Object* class subsumes *CampusObject*, which is a class of *Objects* that is especially relevant for a university campus. *CampusObject* in turn subsumes *Artifact* and *GeographicArea*, where *Artifact* is a human-made *Object* that is associated with a university campus and *GeographicArea* refers to any three dimensional region of the

university campus that has definite boundaries. The classes that are subclasses of *Artifact* and *GeographicArea* are all assumed to be campus-related classes. *LandArea* is a subclass of *GeographicArea* class and refers to the class of land parcel objects occupying a definite region on the university campus.

StationaryArtifact class refers to any *Artifact* with a fixed spatial location on campus, and *Building* and *SportsFacility* are both subclasses of *StationaryArtifact*. *Building* subsumes classes *Library*, *AcademicBuilding*, *ResidentialBuilding*, *EntertainmentBuilding*, *ServiceBuilding*, *StudentUnion* and *DiningCommons*. Considering a campus domain, a *ResidentialBuilding* refers to buildings used for housing students and/or faculty and therefore subsumes *Dormitory* class. *EntertainmentBuilding* is a specialized type of *Building* used for conducting entertainment activities on campus. *EntertainmentBuilding* is an integral class for component classes, *Museum* and *ConcertHall* where a *componentOf* relations links *EntertainmentBuilding* to *Museum* and *ConcertHall*. *ServiceBuilding* is a class of buildings used for facilities management in the campus. *RecyclingCenter* is *containedIn* *ServiceBuilding*. *RecyclingCenter* refers to a class of buildings that involve recycling activities. A *StudentUnion* is a central gathering place for students. *StudentUnion* has numerous parts designated for different uses. *BookStore* and *StudentServiceCenter* share a *containedIn* relation with the class *StudentUnion*. Class *Building* subsumes *DiningCommons*, a type of *Building* specifically designed for serving meals to students and faculty.

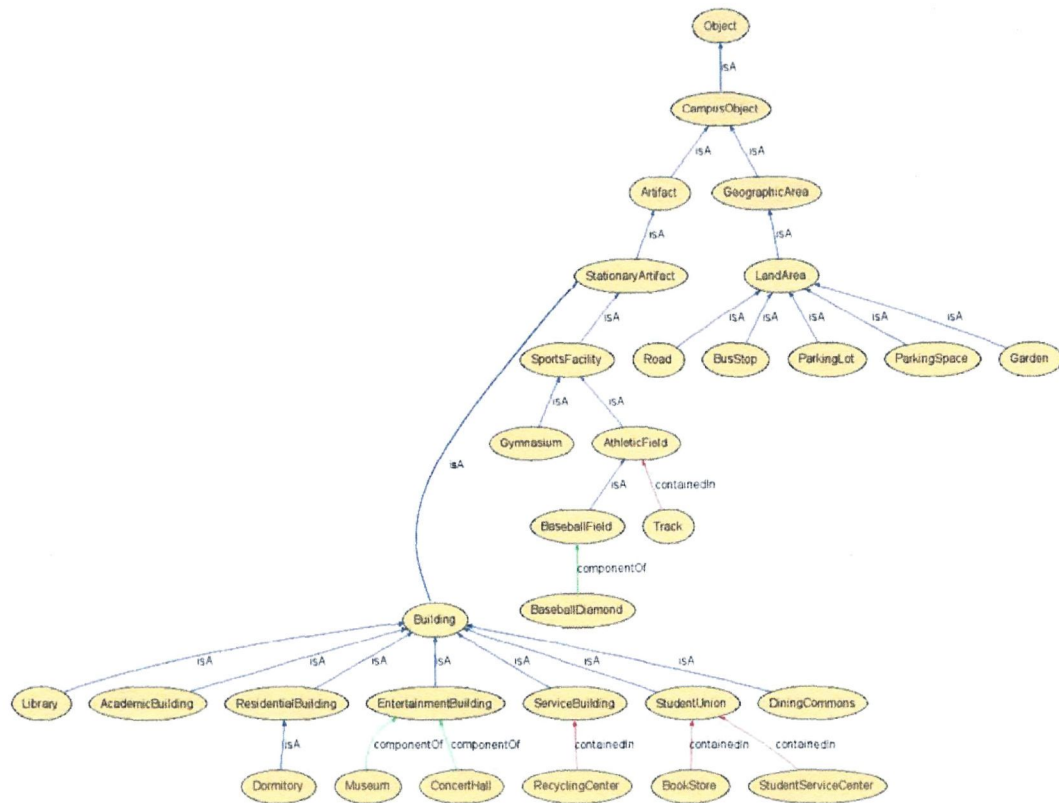


Figure 3.1 Geospatial ontology

Class *SportsFacility* subsumes campus-related classes *Gymnasium* and *AthleticField* where *Gymnasium* is a special kind of *SportsFacility* designed for physical training and contains courts or spaces for various indoor sporting activities. Class *AthleticField* is a *SportsFacility* prepared for the purpose of playing different sports. Class *Track* refers to a specialization of *AthleticField* used for different sporting events, such as races and shares a *containedIn* relation with class *AthleticField*. *BaseballField* refers to a *SportsFacility* designed specially for playing baseball and a *BaseballDiamond* is a *componentOf* *BaseballField*.

Finally, class *LandArea* subsumes campus classes *Road*, *BusStop*, *ParkingLot*, *ParkingSpace* and *Garden*. A class of *LandArea* that can be used for parking vehicles is a *ParkingSpace*. *ParkingLot* is a *LandArea* that has been leveled, paved and marked off for parking automobiles while *Garden* is a specific kind of *LandArea* where plants are cultivated.

3.2 The Temporal Ontology

The temporal ontology, which is the other ontology used to compute the cross product is also based on SUMO classes. The temporal ontology combines general temporal classes, e.g., *DayTime*, with temporal classes that are known to exist in an academic environment, e.g., class *ExamWeek* (Figure 3.2). Similar to the geospatial ontology, a total of seventeen classes have been added to the temporal ontology that is outside of SUMO. *Noon*, *Midnight*, *LunchTime*, *Break*, *AcademicYear*, *Weekday*, *Evening*, *LateNightHour*, *PublicHoliday*, *MidAfternoon*, *EarlyMorningHour*, and *SpringBreak* are the classes that have been drawn from WordNet. Classes *ExamWeek*, *CampusBreak*, *AcademicSemester*, *FallBreak* and *ClassDay* are terms outside of SUMO and WordNet that have been introduced into the temporal ontology given their relevance.

As with the geospatial ontology, a few modifications have been made in the relations existing between classes in SUMO in the temporal ontology. *Midnight* is a subclass of *TimePoint*, which in turn is subsumed by *TimePosition*. Using transitive property, *Midnight* is illustrated as a subclass of *TimePosition* class. Classes *Evening* and *LateNightHour*, subclasses of *TimeInterval* are presented as *componentOf* *NightTime*.

PublicHoliday, a WordNet term, subsumed by *Day* is shown as a subclass of *Holiday*. Similarly, *MidAfternoon* and *EarlyMorningHour* are illustrated as *componentOf* *NightTime* and *DayTime* respectively.

TimeMeasure is the most general class of the temporal ontology. *TimeMeasure* subsumes temporal class *TimePosition*. *TimePosition* subsumes classes *Noon*, *Midnight*, and *TimeInterval* where the class *TimeInterval* refers to a definite length of time marked off by two instants. Time intervals have an extent as well as a location on the universal timeline. This class subsumes classes *Year*, *Month*, *Day*, *DayTime*, *NightTime*, *Holiday*, *LunchTime*, *Weekend*, *Week*, and *Break*.

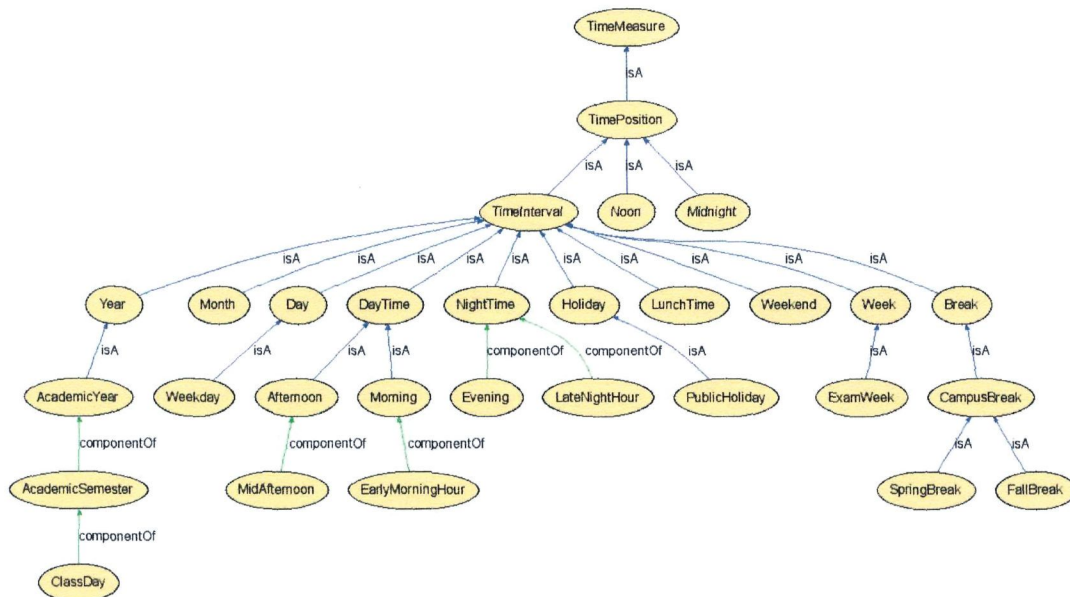


Figure 3.2 Temporal ontology

Class *Year* is a class of all calendar years. *Year* subsumes class *AcademicYear*, the *TimeInterval* during which the university is open and classes are in session.

AcademicSemester shares a *componentOf* relation with *AcademicYear*. *ClassDay* is a *componentOf* *AcademicSemester*. *Class Month* represents the set of all calendar months and *class Day* refers to all calendar days. *Day* subsumes *Weekday*. *WeekDay* is *Day* other than Sunday and/or Saturday. *Class DayTime* refers to those time intervals that begin at sunrise and end at sunset. *DayTime* subsumes *Afternoon*, that time during the day which extends from *Noon* to sunset. *Morning* begins at sunrise and ends at *Noon*. *MidAfternoon* is a *componentOf* *Afternoon*. *Class MidAfternoon* is defined as the period approximately halfway between noon and sunset. *EarlyMorningHour* is an hour early in the *Morning*. *NightTime* on the other hand refers to the time intervals that begin at sunset and end at sunrise. *Evening* is a *componentOf* *NightTime*. It is the latter part of the *Day*, from when the daylight starts to decrease until nightfall. *LateNightHour* is another *componentOf* *NightTime*. It is the latter part of night.

Class Holiday is a *TimeInterval* on or during which work is suspended by law or custom. *Holiday* in turn subsumes class *PublicHoliday*, which is the authorized by law and limits work or official government business. *LunchTime* is a subclass of *TimeInterval* class that defines a period set aside for eating a mid-day meal. *Weekend* class includes Saturdays and/or Sundays, which is dependant on the custom of a country. *Week*, similar to other *TimeInterval* subclasses, is a class of all calendar weeks. *ExamWeek* is a subclass of *Week* referring to the end of the *AcademicSemester* when classes finish and examinations are held. *Break* refers to a *TimeInterval* during which there is a temporary cessation of some activity. Considering an academic domain, class *Break* refers to intervals during which academic activities like teaching cease. *Break* class subsumes *CampusBreak* that subsumes classes *SpringBreak* and *FallBreak*.

3.3 Representing the Ontologies in Protégé

In this thesis, the ontologies have been modeled using Protégé, an open source ontology editor and a knowledge base framework (<http://protege.stanford.edu/>). Protégé supports the creation, visualization and manipulation of ontologies in various formats including the Resource Description Framework (RDF), the Web Ontology Language (OWL) and the Extensible Markup Language (XML). The Protégé-OWL editor enables users to build ontologies in OWL, for example, for the semantic web.

3.3.1 Understanding OWL Ontologies

The basic elements of an OWL ontology are classes, properties, instances and relationships. OWL is expressed using XML, allowing for an easy exchange of information irrespective of the platform. In this work, we are most interested in class names and relations and the representation of these elements in OWL is the focus of this section.

Every class in an OWL ontology is a subclass of `owl:Thing`. Classes specific to the temporal ontology or the geospatial ontology are defined in OWL by simply declaring a named class, where the value of `rdf:ID` indicates the name of the class. Thus, each class in the ontology is described with the tag `<owl:Class rdf:ID="ClassName"/>` (e.g., `<owl:Class rdf:ID="ParkingLot"/>`). Within the OWL document, any class can be referred to using `#ClassName` (e.g., `#ParkingLot`).

As each class in an ontology subsumes zero or more subclasses, `rdfs:subClassOf` is the fundamental taxonomic constructor for classes in OWL. This syntax relates a subclass to its more general superclass and is used to represent any *isA* relationships. In OWL, the subsumption hierarchy is denoted using the `<rdfs:subClassOf rdf:resource="#SuperClassName"/>` tag. The value of `rdf:resource` gives the name of the superclass. For example, *LandArea* is a superclass of *ParkingLot* and this relation is expressed by,

```
<owl:Class rdf:ID="ParkingLot">
  <rdfs:subClassOf rdf:resource="#LandArea"/>
</owl:Class>
```

Properties assert general facts about members of a class or specific facts about individuals. `ObjectProperty` relates one class with another, while `DatatypeProperty` describes attributes of a class. In this work, the tag `<owl:ObjectProperty rdf:ID="RelationName"/>` is used to describe the relations *componentOf* and *containedIn* where `ObjectProperty` is a type of `Property`. `ObjectProperty` links two classes and hence is a binary relation. Using the tag, and replacing `RelationName` with a particular relation, *componentOf* and *containedIn* relations are represented in the following manner,

```
<owl:ObjectProperty rdf:ID="componentOf"/>
<owl:ObjectProperty rdf:ID="containedIn"/>
```

The relation *componentOf* links all component classes with their integral whole class. For example, the following syntax is used in OWL to capture the *componentOf* relation between *BaseballDiamond* and *BaseballField*, where *BaseballDiamond* is a *componentOf* *BaseballField* expressed as,

```
<owl:Class rdf:ID="BaseballDiamond">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#componentOf"/>
      <owl:someValuesFrom
        rdf:resource="#BaseballField"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Using the Restriction tag `<owl:Restriction>` in specific contexts constrains the range of a property in addition to designating property characteristics. The `owl:onProperty` element indicates a restricted property. The `owl:someValuesFrom` restriction means that for all *BaseballDiamonds*, there is a *componentOf* link to at least one *BaseballField*.

For cases of spatial inclusion, the restricting property *containedIn* is described by the tag `<owl:onProperty rdf:resource="#containedIn"/>`. For example,

```

<owl:Class rdf:ID="Track">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#containedIn"/>
      <owl:someValuesFrom
        rdf:resource="#AthelticField"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

Classes in an ontology are described further by their attributes. The focus of this work is on the classes themselves and the relations that they share with other classes, but a class in a geospatial ontology, e.g., *Building* can have attributes *BuildingName* and *NumberOfFloors*. In OWL, attributes of a class are defined using a `DataTypeProperty` tag and are defined independently of the classes that they belong to. Classes are assigned as domains for the `DataTypeProperty`. Additional relational constraints (e.g., cardinality) can be added to the properties. For example,

```

<owl:DatatypeProperty rdf:ID="RoadLength">
  <rdfs:domain rdf:resource="#Road" />
</owl:DatatypeProperty>

```

In this way, the geospatial and temporal ontology classes, their attributes, as well as the relations that link the classes are systematically represented in XML format using Protégé.

3.3.2 The Protégé-OWL Editor

The OWLClasses tab in Protégé-OWL editor consists of two main frames, including Subclass Explorer and Class Editor. The Subclass Explorer frame provides a visual interface that allows users to add subclasses to existing classes, create sibling classes, delete existing classes (Figure 3.3). An empty ontology initially contains one class called `owl:Thing`. The `owl:Thing` class is the default, upper-most class that subsumes all other classes. Creating a new class involves selecting the desired class in the Asserted Hierarchy of the Subclass Explorer frame and using the Create Subclass button to create a new class as a subclass of the selected class. An appropriate name for the newly created class is input in the Class Editor frame (Figure 3.4). In a similar manner, a sibling class can be created using the Create Sibling Class button. Sibling classes have the same parent class. To delete any class, the Delete Selected Class(es) button is used. Deleting a selected class with subclasses removes the chosen class as well as all its children.

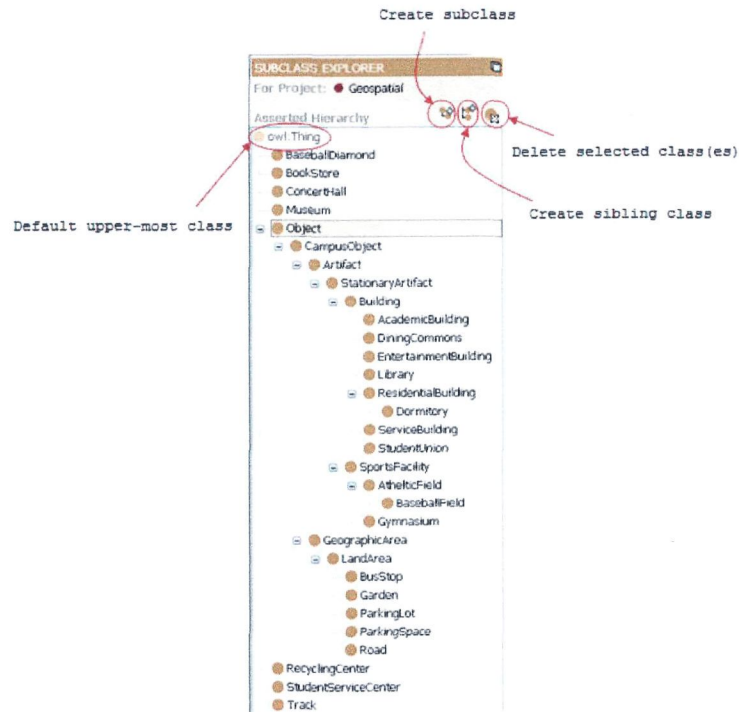


Figure 3.3 Subclass Explorer frame in OWLClasses tab in Protégé-OWL editor

Once a new subclass or a new sibling class is created, the Class Editor frame in the OWLClasses tab can be used to enter and edit class details (Figure 3.4). The name of the class as well as the restrictions can be input in this frame. In Protégé, *isA* relations are defined using subclasses. For example, *AcademicBuilding isA Building* is expressed by creating a subclass *AcademicBuilding* to *Building* class. In case of other relations, including *containedIn* and *componentOf*, each relation is declared as a property. Using the property value, *containedIn* and *componentOf* relations between different classes can be expressed. The Properties tab in the Protégé-OWL editor allows Properties to be added and deleted. The relations are input as new Object Properties in the Property

Browser while the Property Editor allows a selected Object Property to be renamed. To delete a selected Property, the Delete Properties button is used.

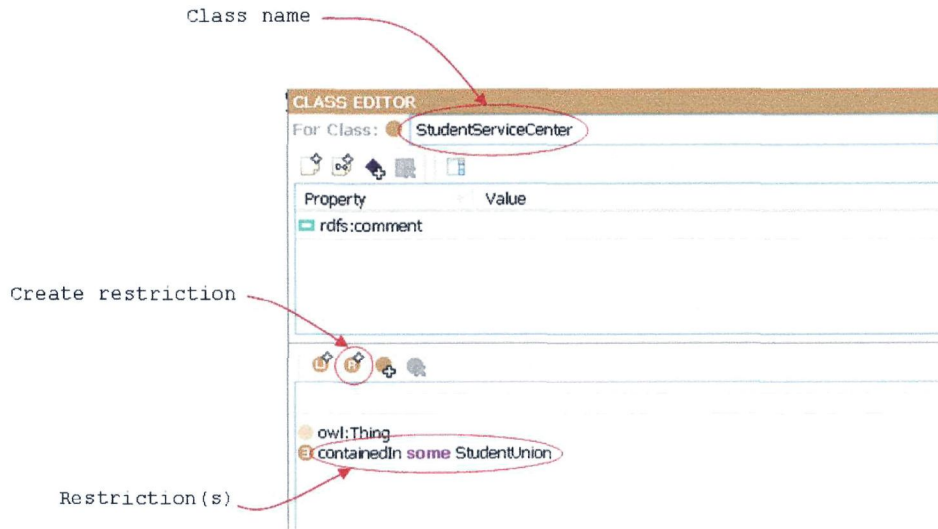


Figure 3.4 Class Editor frame in OWLClasses tab in Protégé-OWL editor

To create a new restriction for a selected class, e.g., *StudentServiceCenter*, **Create Restriction** button is used. In the **Create Restriction** window, the type of restricted property is selected, e.g., *containedIn*, along with the Restriction, e.g., *someValuesFrom*. Then the class with which the chosen class is linked is selected using the **Insert Class** option (Figure 3.5). The *someValuesFrom* restriction is composed of a quantifier and a filler. The quantifier used is an existential quantifier, which is read as at least one, or more. So for a set of individuals, an existential restriction specifies the existence of at least one relationship along a given property to an individual that is a member of a specific class.

Protégé-OWL editor also provides a tool to export the ontologies in XML format. After the classes and the relations have been defined, the ontology can be exported by using the Show RDF/XML source code under the Code tab in the Protégé-OWL editor (Figure 3.6). These OWL ontologies are used as inputs for the Parser tool. This parser has been developed using Visual C#.NET.

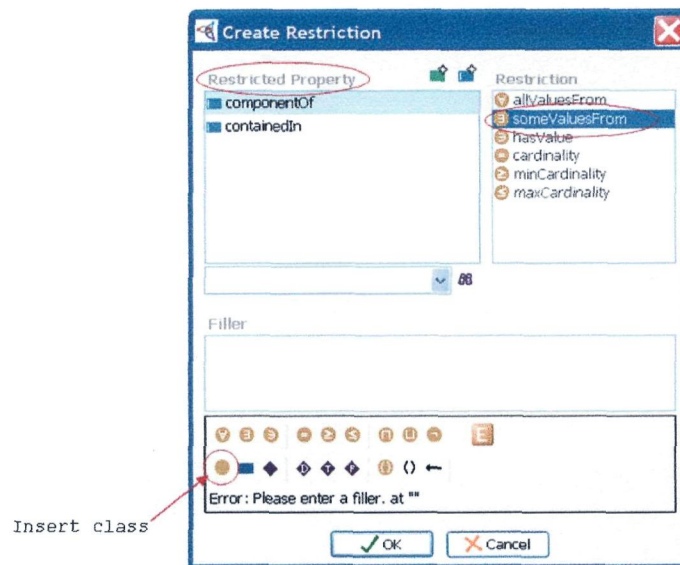


Figure 3.5 Create Restriction window in Protégé-OWL editor

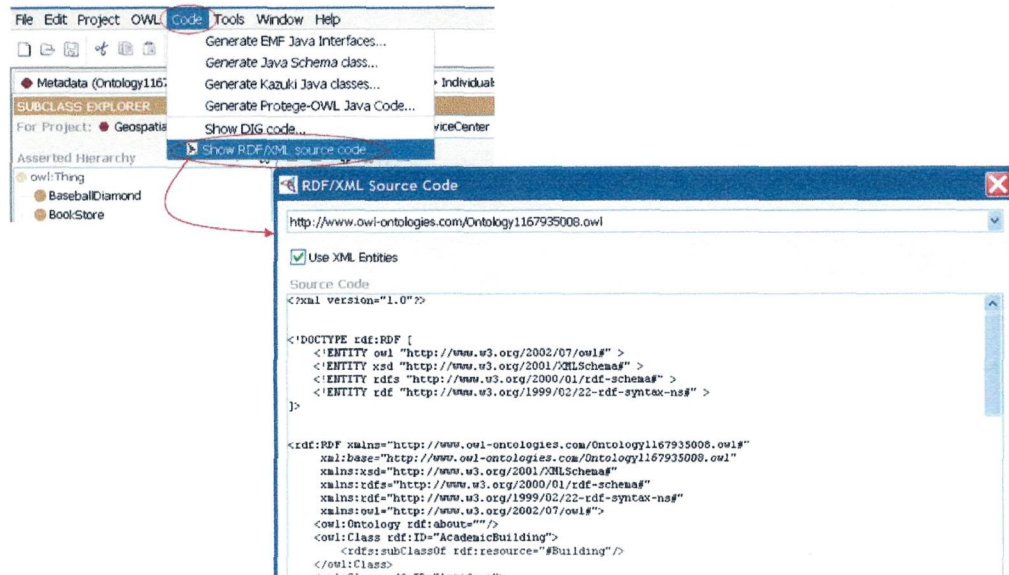


Figure 3.6 Using Protégé-OWL editor to export an ontology in XML

3.4 Computing the Cross Product

Combining a pair of ontologies such as a geospatial and temporal ontology using automated means allows a user to quickly build a more comprehensive description of a domain of interest. The resulting combination is useful for making higher-order inferences. For example, for a campus domain, the combination of a geospatial and temporal ontology affords reasoning about entities on a university campus over a range of possible times. The resulting reasoning framework provides support by showing all the possible times relating to campus entities (e.g., buildings) helping, for example, a facilities management administrator to be sure to consider entities in their domain over *all* possible times. Spatio-temporal aspects of ontologies have been discussed in Grenon and Smith (2004) where discussion highlights how spatial ontologies typically support

snapshot views of the world at successive instants of time and how ontologies can be extended to create spatiotemporal ontologies of change and process.

One method for combining the class names or *terms* from a pair of orthogonal ontologies is to compute the cross product. Computing a cross product involves taking a term from the first ontology (Figure 3.7a) and combining it with every term from the second ontology (Figure 3.7b). The steps are repeated until all the terms from the first ontology are exhausted and a complete set of pair-wise combinations of each of the terms from the individual ontologies have been created (Figure 3.6c). For example, the term *Library* from the geospatial ontology, when crossed with *WeekDay* from the temporal ontology, gives *Library_Weekday*. The term *Library* is subsequently combined with the remaining terms in the temporal ontology, *Day* and *TimeInterval*, to return *Library_Day* and *Library_TimeInterval* respectively. This process is repeated for all other terms in the geospatial ontology.

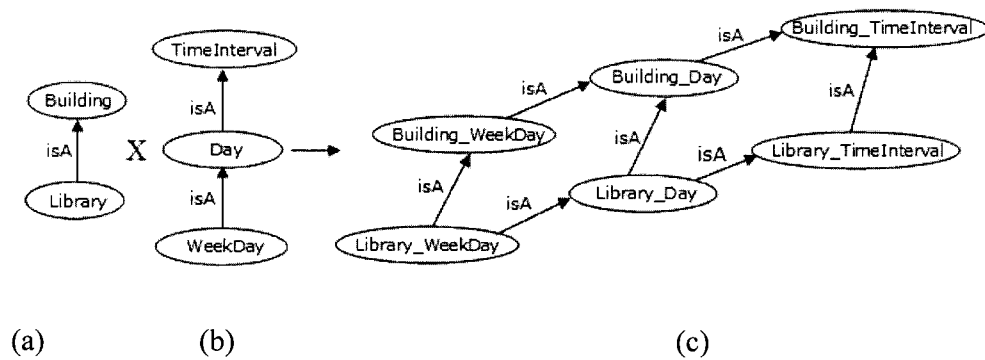


Figure 3.7 Computing cross product (a) Sample terms and relations from the campus geospatial ontology, (b) sample terms and relations from the temporal ontology, and (c) cross product of the terms and relations from the geospatial ontology and temporal ontology

Just as the terms in an ontology are linked by relations, the geospatial-temporal pairs of terms in the cross product are also connected by relations. Any grouping of *isA*, *componentOf* and *containedIn* relations can exist in either of the two ontologies and the cross product accounts for those. Each geospatial-temporal term is related to two other geospatial-temporal terms unless either one of the terms from the base ontology is a class that does not have any superclass (e.g., *Building_Weekday*). In that case, a geospatial-temporal term is linked to only one other term. In the case where either of the classes being crossed are not subclasses of any other class, this geospatial-temporal combination will not have a parent in the cross product. For example, *Building_TimeInterval* is not related to any parent term (Figure 3.7c). The type of relation that links any two geospatial-temporal terms in the cross product is determined from the corresponding relations in the base ontologies. For example, the geospatial-temporal combination *Library_Day* is linked to both *Library_TimeInterval* and *Building_Day* with *isA* relations, based on the relation that originally links *Day* with *TimeInterval* (*isA*) and *Library* with *Building* (*isA*).

The resulting cross product represents a more complete vocabulary that includes specific aspects from each of the two base ontologies (Hill *et al.*, 2002). These pair-wise combinations of terms from a geospatial ontology and a temporal ontology form a spatio-temporal granularity framework that captures a complete spatio-temporal perspective of a domain.

To compute the cross product of a pair of ontologies modeled using OWL, there are two principal steps:

- parsing terms from each ontology and importing these terms into an RDBMS,
- and
- applying SQL operations to derive the cross product of the two ontologies.

The following sections describe each of these steps in more detail.

3.4.1 Parsing Terms from the Ontologies

The first step in computing the cross product involves extracting terms from the pair of ontologies modeled in OWL. This is undertaken in order for the values of the terms and relations to be imported into a relational database management system (RDBMS). A relational database setting provides a straightforward means to perform the computation of cross products where structured query language (SQL) queries express each of the steps.

A parser, implemented on a Visual C#.NET platform, has been built to extract values from the OWL ontologies. This implementation uses an INSERT (SQL) operation that results in two relational tables, *GeospatialRel* (Figure 3.8a) and *TemporalRel* (Figure 3.8b). In order to populate the *GeospatialRel* and the *TemporalRel*, each `<owl:Class...></owl:Class>` block in the respective OWL ontologies is read to extract the values of superclass and subclass pairs as well as the relation linking the classes (e.g., *isA*, *componentOf* or *containedIn*).

3.4.2 Importing into an RDBMS

After the required values have been parsed from the pair of ontologies, a test is performed to ensure that the two ontologies do not have common terms, that is, same class name. In this thesis, two ontologies that do not have any class names in common are termed orthogonal ontologies. In the event that the pair of ontologies is not orthogonal, the next step of could involve determining whether or not the classes are semantically same or not (see for example, Rodriguez and Egenhofer, 2003), but this is outside the scope of this research. Further steps can involve, for example, renaming each of the classes with a prefix of the name of the base ontology for cases where the classes do not carry the same semantics.

Once the test has been completed and the common classes either renamed or removed, the values are written into an RDBMS as tuples in the relations *GeospatialRel* and *TemporalRel*. The schema for *GeospatialRel* and *TemporalRel* relations consists of three attributes each: *Child*, *Parent* and *Relation*. The *Parent* attribute refers to all superclass terms from each of the ontologies, and the *Child* attribute refers to subclasses of these superclass terms. The *Relation* attribute contains the name of the ontological relation that links superclasses and subclasses in the ontologies.

Child	Parent	Relation
AcademicBuilding	Building	isA
Artifact	CampusObject	isA
AthleticField	SportsFacility	isA
BaseballDiamond	BaseballField	componentOf
BaseballField	AthleticField	isA
BookStore	StudentUnion	containedIn
Building	StationaryArtifact	isA
BusStop	LandArea	isA
CampusObject	Object	isA
ConcertHall	EntertainmentBuilding	componentOf
DiningCommons	Building	isA
Dormitory	ResidentialBuilding	isA
EntertainmentBuilding	Building	isA
Garden	LandArea	isA
GeographicArea	CampusObject	isA
Gymnasium	SportsFacility	isA
LandArea	GeographicArea	isA
Library	Building	isA
Museum	EntertainmentBuilding	componentOf
ParkingLot	LandArea	isA
ParkingSpace	LandArea	isA
RecyclingCenter	ServiceBuilding	containedIn
ResidentialBuilding	Building	isA
Road	LandArea	isA
ServiceBuilding	Building	isA
SportsFacility	StationaryArtifact	isA
StationaryArtifact	Artifact	isA
StudentServiceCenter	StudentUnion	containedIn
StudentUnion	Building	isA
Track	AthleticField	containedIn

(a)

Child	Parent	Relation
AcademicSemester	AcademicYear	componentOf
AcademicYear	Year	isA
Afternoon	DayTime	isA
Break	TimeInterval	isA
CampusBreak	Break	isA
ClassDay	AcademicSemester	componentOf
Day	TimeInterval	isA
DayTime	TimeInterval	isA
EarlyMorningHour	Morning	componentOf
Evening	NightTime	componentOf
ExamWeek	Week	isA
FallBreak	CampusBreak	isA
Holiday	TimeInterval	isA
LateNightHour	NightTime	componentOf
LunchTime	TimeInterval	isA
MidAfternoon	Afternoon	componentOf
Midnight	TimePosition	isA
Month	TimeInterval	isA
Morning	DayTime	isA
NightTime	TimeInterval	isA
Noon	TimePosition	isA
PublicHoliday	Holiday	isA
SpringBreak	CampusBreak	isA
Thanksgiving	PublicHoliday	isA
TimeInterval	TimePosition	isA
TimePosition	TimeMeasure	isA
Week	TimeInterval	isA
Weekday	Day	isA
Weekend	TimeInterval	isA
Year	TimeInterval	isA

(b)

Figure 3.8 Relations with extracted attribute values (a) Tuples from *GeospatialRel* relation and (b) tuples from *TemporalRel* relation. Both relations have three attributes, *Child*, *Parent*, and *Relation*

The relations *GeospatialClasses* (Figure 3.9a) and *TemporalClasses* (Figure 3.9b) consist of a single attribute *Classes* that is based on all the terms in the respective geospatial and temporal ontologies as extracted from OWL. To construct these relations, the value of `rdf:ID` is read for each `<owl:Class...>` tag.

Classes
AcademicBuilding
Artifact
AthleticField
BaseballDiamond
BaseballField
BookStore
Building
BusStop
CampusObject
ConcertHall
DiningCommons
Dormitory
EntertainmentBuilding
Garden
GeographicArea
Gymnasium
LandArea
Library
Museum
Object
ParkingLot
ParkingSpace
RecyclingCenter
ResidentialBuilding
Road
ServiceBuilding
SportsFacility
StationaryArtifact
StudentServiceCenter
StudentUnion
Track

(a)

Classes
AcademicSemester
AcademicYear
Afternoon
Break
CampusBreak
ClassDay
Day
DayTime
EarlyMorningHour
Evening
ExamWeek
FallBreak
Holiday
LateNightHour
LunchTime
MidAfternoon
Midnight
Month
Morning
NightTime
Noon
PublicHoliday
SpringBreak
Thanksgiving
TimeInterval
TimeMeasure
TimePosition
Week
Weekday
Weekend
Year

(b)

Figure 3.9 Relations showing lists of classes (a) Tuples from *GeospatialClasses* relation and (b) tuples from *TemporalClasses* relation

3.4.3 Applying SQL Operations to Compute the Cross Product

Once the *GeospatialClasses*, *TemporalClasses*, *GeospatialRel*, and *TemporalRel* relations have been established, a series of SQL operations are carried out in order to compute the cross product of the two ontologies. The first step is a Cartesian product between *GeospatialRel* and *TemporalClasses* that returns *GeospatialXTemporalC* relation, that is, $\text{GeospatialRel} \times \text{TemporalClasses} \rightarrow \text{GeospatialXTemporalC}$. This new relation has four attributes, *Child*, *Parent*, *Relation*, and *Classes*. In SQL, this step is summarized as,


```

SELECT GeospatialRel.*, TemporalClasses.*
FROM GeospatialRel, TemporalClasses;

```

An equi-join operation is performed on *GeospatialXTemporalC* with *TemporalClasses*, resulting in *XTemporal* relation (Figure 3.10). The equi-join enforces the case that only those tuples where the temporal terms in both relations have identical values appear in the new relation *XTemporal*, that is,

$$\text{GeospatialXTemporalC} \underset{\text{Child1B=Parent1B}}{\bowtie} \text{TemporalClasses} \longrightarrow \text{XTemporal}$$

This step is expressed using SQL as,

```

SELECT GeospatialXTemporalC.Child AS Child1A,
TemporalClasses.Classes AS Child1B,
GeospatialXTemporalC.Relation, GeospatialXTemporalC.Parent
AS Parent1A, GeospatialXTemporalC.Classes AS Parent1B
FROM GeospatialXTemporalC, TemporalClasses
WHERE
TemporalClasses.Classes = GeospatialXTemporalC.Classes;

```

In a similar fashion, a Cartesian product between relations *TemporalRel* and *GeospatialClasses* is performed, resulting in *TemporalXGeospatialC*, that is, $\text{TemporalRel} \times \text{GeospatialClasses} \rightarrow \text{TemporalXGeospatialC}$. The schema of *TemporalXGeospatialC* is the same as for *GeospatialXTemporalC*, that is, four attributes,

Child, Parent, Relation, and Classes. An equi-join based on geospatial terms that are equal in both the relations is performed on *TemporalXGeospatialC* with *GeospatialClasses* to achieve *XGeospatial*, that is,

$$\text{TemporalXGeospatialC} \underset{\text{Child1A=Parent1A}}{\triangleright\triangleleft} \text{GeospatialClasses} \longrightarrow \text{XGeospatial}$$

As a final step, a Union operation is applied to *XTemporal* and *XGeospatial* relations to produce *CrossProduct* (Figure 3.11) that is, $\text{XTemporal} \cup \text{XGeospatial} \rightarrow \text{CrossProduct}$. Using SQL, this step can be summarized as,

```
SELECT XGeospatial.Child1A, XGeospatial.Child1B,
XGeospatial.Relation, XGeospatial.Parent1A,
XGeospatial.Parent1B
FROM XGeospatial
UNION
SELECT XTemporal.Child1A, XTemporal.Child1B,
XTemporal.Relation, XTemporal.Parent1A, XTemporal.Parent1B
FROM XTemporal;
```

Child1A	Child1B	Relation	Parent1A	Parent1B
AcademicBuilding	AcademicSemester	isA	Building	AcademicSemester
Artifact	AcademicSemester	isA	CampusObject	AcademicSemester
AthleticField	AcademicSemester	isA	SportsFacility	AcademicSemester
BaseballDiamond	AcademicSemester	componentOf	BaseballField	AcademicSemester
BaseballField	AcademicSemester	isA	AthleticField	AcademicSemester
BookStore	AcademicSemester	containedIn	StudentUnion	AcademicSemester
Building	AcademicSemester	isA	StationaryArtifact	AcademicSemester
BusStop	AcademicSemester	isA	LandArea	AcademicSemester
CampusObject	AcademicSemester	isA	Object	AcademicSemester
ConcertHall	AcademicSemester	componentOf	EntertainmentBuilding	AcademicSemester
DiningCommons	AcademicSemester	isA	Building	AcademicSemester
Dormitory	AcademicSemester	isA	ResidentialBuilding	AcademicSemester
EntertainmentBuilding	AcademicSemester	isA	Building	AcademicSemester
Garden	AcademicSemester	isA	LandArea	AcademicSemester
GeographicArea	AcademicSemester	isA	CampusObject	AcademicSemester
Gymnasium	AcademicSemester	isA	SportsFacility	AcademicSemester
LandArea	AcademicSemester	isA	GeographicArea	AcademicSemester
Library	AcademicSemester	isA	Building	AcademicSemester
Museum	AcademicSemester	componentOf	EntertainmentBuilding	AcademicSemester
ParkingLot	AcademicSemester	isA	LandArea	AcademicSemester
ParkingSpace	AcademicSemester	isA	LandArea	AcademicSemester
RecyclingCenter	AcademicSemester	containedIn	ServiceBuilding	AcademicSemester
ResidentialBuilding	AcademicSemester	isA	Building	AcademicSemester
Road	AcademicSemester	isA	LandArea	AcademicSemester
ServiceBuilding	AcademicSemester	isA	Building	AcademicSemester
SportsFacility	AcademicSemester	isA	StationaryArtifact	AcademicSemester
StationaryArtifact	AcademicSemester	isA	Artifact	AcademicSemester
StudentServiceCenter	AcademicSemester	containedIn	StudentUnion	AcademicSemester
StudentUnion	AcademicSemester	isA	Building	AcademicSemester
Track	AcademicSemester	containedIn	AthleticField	AcademicSemester
AcademicBuilding	AcademicYear	isA	Building	AcademicYear
Artifact	AcademicYear	isA	CampusObject	AcademicYear
AthleticField	AcademicYear	isA	SportsFacility	AcademicYear
BaseballDiamond	AcademicYear	componentOf	BaseballField	AcademicYear
BaseballField	AcademicYear	isA	AthleticField	AcademicYear
BookStore	AcademicYear	containedIn	StudentUnion	AcademicYear
Building	AcademicYear	isA	StationaryArtifact	AcademicYear
:	:	:	:	:

Figure 3.10 Sample tuples from relation obtained by the equi-join of *GeospatialXTemporalC* and *TemporalClasses*

CrossProduct has five attributes including two geospatial-temporal pairs of attributes, as well as a *Relation* attribute that is the corresponding link (*isA*, *componentOf*, *containedIn*) between the geospatial-temporal pairs. Each tuple in the relation can be regarded as a set of terms, for instance, $\{AcademicBuilding, AcademicSemester, isA, Building, AcademicSemester\}$, where *AcademicBuilding_AcademicSemester* form one geospatial-temporal pair, and *Building_*

AcademicSemester form another pair. In this case, these two pairs are related by an *isA* relation.

The complete set of tuples in the *CrossProduct* relation corresponds to all possible combinations of terms from the two input ontologies as well as the relations that link those combinations. In general, the size of the resulting cross product is determined by the number of terms in each of the two base ontologies. Assuming M to be the number of classes in the geospatial ontology and N , the number of classes in the temporal ontology, then the number of terms in the cross product is $M \times N$. For example, in this case, combining the campus geospatial ontology (31 terms) with the temporal ontology (30 terms) results in a cross product that has 930 geospatial-temporal terms.

Child1A	Child1B	Relation	Parent1A	Parent1B
AcademicBuilding	AcademicSemester	componentOf	AcademicBuilding	AcademicYear
AcademicBuilding	AcademicSemester	isA	Building	AcademicSemester
AcademicBuilding	AcademicYear	isA	AcademicBuilding	Year
AcademicBuilding	AcademicYear	isA	Building	AcademicYear
AcademicBuilding	Afternoon	isA	AcademicBuilding	DayTime
AcademicBuilding	Afternoon	isA	Building	Afternoon
AcademicBuilding	Break	isA	AcademicBuilding	TimeInterval
AcademicBuilding	Break	isA	Building	Break
AcademicBuilding	CampusBreak	isA	AcademicBuilding	Break
AcademicBuilding	CampusBreak	isA	Building	CampusBreak
AcademicBuilding	ClassDay	componentOf	AcademicBuilding	AcademicSemester
AcademicBuilding	ClassDay	isA	Building	ClassDay
AcademicBuilding	Day	isA	AcademicBuilding	TimeInterval
AcademicBuilding	Day	isA	Building	Day
AcademicBuilding	DayTime	isA	AcademicBuilding	TimeInterval
AcademicBuilding	DayTime	isA	Building	DayTime
AcademicBuilding	EarlyMorningHour	componentOf	AcademicBuilding	Morning
AcademicBuilding	EarlyMorningHour	isA	Building	EarlyMorningHour
AcademicBuilding	Evening	componentOf	AcademicBuilding	NightTime
AcademicBuilding	Evening	isA	Building	Evening
AcademicBuilding	ExamWeek	isA	AcademicBuilding	Week
AcademicBuilding	ExamWeek	isA	Building	ExamWeek
AcademicBuilding	FallBreak	isA	AcademicBuilding	CampusBreak
AcademicBuilding	FallBreak	isA	Building	FallBreak
AcademicBuilding	Holiday	isA	AcademicBuilding	TimeInterval
AcademicBuilding	Holiday	isA	Building	Holiday
AcademicBuilding	LateNightHour	componentOf	AcademicBuilding	NightTime
AcademicBuilding	LateNightHour	isA	Building	LateNightHour
AcademicBuilding	LunchTime	isA	AcademicBuilding	TimeInterval
AcademicBuilding	LunchTime	isA	Building	LunchTime
AcademicBuilding	MidAfternoon	componentOf	AcademicBuilding	Afternoon
AcademicBuilding	MidAfternoon	isA	Building	MidAfternoon
AcademicBuilding	Midnight	isA	AcademicBuilding	TimePosition
AcademicBuilding	Midnight	isA	Building	Midnight
AcademicBuilding	Month	isA	AcademicBuilding	TimeInterval
AcademicBuilding	Month	isA	Building	Month
AcademicBuilding	Morning	isA	AcademicBuilding	DayTime
AcademicBuilding	Morning	isA	Building	Morning
AcademicBuilding	NightTime	isA	AcademicBuilding	TimeInterval
AcademicBuilding	NightTime	isA	Building	NightTime
AcademicBuilding	Noon	isA	AcademicBuilding	TimePosition
AcademicBuilding	Noon	isA	Building	Noon
AcademicBuilding	PublicHoliday	isA	AcademicBuilding	Holiday
AcademicBuilding	PublicHoliday	isA	Building	PublicHoliday
AcademicBuilding	SpringBreak	isA	AcademicBuilding	CampusBreak
AcademicBuilding	SpringBreak	isA	Building	SpringBreak
AcademicBuilding	Thanksgiving	isA	AcademicBuilding	PublicHoliday
AcademicBuilding	Thanksgiving	isA	Building	Thanksgiving
AcademicBuilding	TimeInterval	isA	AcademicBuilding	TimePosition
AcademicBuilding	TimeInterval	isA	Building	TimeInterval
AcademicBuilding	TimeMeasure	isA	Building	TimeMeasure
AcademicBuilding	TimePosition	isA	AcademicBuilding	TimeMeasure
AcademicBuilding	TimePosition	isA	Building	TimePosition
:	:	:	:	:

Figure 3.11 Sample tuples from *CrossProduct* relation

3.5 Summary

This chapter introduces the two domain ontologies that are used in this research. Both the geospatial ontology and the temporal ontology are based on SUMO, a public upper ontology. We use an open-source ontology editor, Protégé, to model the ontologies in XML. Computing the cross product is achieved through three main steps, namely, parsing the terms from the ontologies, importing the terms into a relational database management system and applying SQL operations to compute the cross product. Before the ontologies are parsed, a test is performed to ensure that no common terms exist between the two ontologies. A Visual C#.NET platform has been used to implement the orthogonal nature of the ontologies as well as the parsing of the terms from the ontologies.

Once the cross product is computed, different SQL queries can be applied on the spatio-temporal framework. Using the SQL queries, we can select tuples that are relevant for a particular analysis, e.g., tuples that show the class *Library* at all times.

Chapter 4

QUERYING THE CROSS PRODUCT

In this work, pairs of ontologies are automatically combined using cross products. The comprehensive spatio-temporal reasoning space that results allows one to describe all geospatial domain entities over all temporal perspectives as well as perform additional, more focused reasoning.

In this chapter, we consider the different types of queries that are possible on a cross product relation. One type of SQL query searches for a particular value in either the terms from the geospatial ontology or the temporal ontology, or for a specific kind of relation linking any two pairs of terms. For example, search for a single term from the geospatial ontology, for instance, *Building*, and combine it with all terms in the temporal ontology, or find all geospatial-temporal pairs of terms connected with a *containedIn* relation. Next there are SQL queries that search for a specific combination of terms from the geospatial ontology with terms from the temporal ontology. For example, a query that explores combinations of a particular term from the geospatial ontology, for instance, *Library*, with a specific term from the temporal ontology, for example, *Morning*. Another type of SQL query looks for a combination of a particular term from one ontology with another term and *all its subclasses* from the other ontology.

4.1 Queries Highlighting a Single Term

The complete cross product of two ontologies presents all possible combinations of geospatial and temporal terms. One possible SQL query on the cross product searches for a specific geospatial term and returns all temporal combinations of that term, allowing a user to analyze a single geospatial term over a range of different times. For example, a query that finds a geospatial term *Library* and returns all *Library-temporal* pairs of terms from *CrossProduct*. This query is expressed as,

```
SELECT Child1A, Child1B FROM CrossProduct
WHERE Child1A = "Library" AND Parent1A = "Library";
```

Attribute *Child1A* and *Parent1A* refer to terms from the geospatial ontology and only those tuples that have the value *Library* for both *Child1A* and *Parent1A* appear in the query result (Figure 4.1). The result allows a user to exhaustively consider a geospatial entity, *Library*, all times, including those that might have been overlooked during previous analyses. This query result can be used by an IT Supervisor at the Library to determine IT Support Staff assignment by gaining an understanding of different times at which services need to be provided.

Child1A	Child1B
Library	AcademicSemester
Library	AcademicYear
Library	Afternoon
Library	Break
Library	CampusBreak
Library	ClassDay
Library	Day
Library	DayTime
Library	EarlyMorningHour
Library	Evening
Library	ExamWeek
Library	FallBreak
Library	Holiday
Library	LateNightHour
Library	LunchTime
Library	MidAfternoon
Library	Midnight
Library	Month
Library	Morning
Library	NightTime
Library	Noon
Library	PublicHoliday
Library	SpringBreak
Library	Thanksgiving
Library	TimeInterval
Library	TimePosition
Library	Week
Library	Weekday
Library	Weekend
Library	Year

Figure 4.1 Query result depicting *Library* combined with all temporal terms

With this query, we demonstrate that we support the hypothesis presented in Chapter One. The hypothesis states that *a multi-granular, unified framework results from taking the cross product of a pair of orthogonal ontologies*. The query result shown in Figure 4.1 displays a geospatial term *Library* combined with different terms from the temporal ontology. Each new term in this relation corresponds to a different granularity. For example, the term *Library_AcademicYear* is a different granularity from *Library_AcademicSemester*. The resulting framework combines information from two diverse areas into a single unified reasoning space.

Similarly, all geospatial terms can be combined with a single term from the temporal ontology to return, for example, all *geospatial-Morning* pairs of terms. This query that retrieves all geospatial terms at a particular time (e.g., *Morning*), is expressed as,

```
SELECT Child1A, Child1B FROM CrossProduct
WHERE Child1B = "Morning" AND Parent1B = "Morning";
```

The attributes, *Child1B* and *Child1A*, contain the temporal term *Morning* in combination with all geospatial terms (Figure 4.2). These queries allow a user to understand a geospatial domain at different temporal granularities, for instance, at *TimeInterval*, at *DayTime* or at *EarlyMorningHour*. The result of this query can be used by a Facilities Manager to understand what classes in a campus might need special kind of attention at a given time or for staffing purposes in general at a specific time. For example, assigning snow plows to clear snow from *Roads* and *ParkingLots* at *EarlyMorningHour* in winter.

Child1A	Child1B
AcademicBuilding	Morning
Artifact	Morning
AthleticField	Morning
BaseballDiamond	Morning
BaseballField	Morning
BookStore	Morning
Building	Morning
BusStop	Morning
CampusObject	Morning
ConcertHall	Morning
DiningCommons	Morning
Dormitory	Morning
EntertainmentBuilding	Morning
Garden	Morning
GeographicArea	Morning
Gymnasium	Morning
LandArea	Morning
Library	Morning
Museum	Morning
ParkingLot	Morning
ParkingSpace	Morning
RecyclingCenter	Morning
ResidentialBuilding	Morning
Road	Morning
ServiceBuilding	Morning
SportsFacility	Morning
StationaryArtifact	Morning
StudentServiceCenter	Morning
StudentUnion	Morning
Track	Morning

Figure 4.2 Query result depicting all geospatial terms combined with *Morning*

Another type of query returns all the tuples with a certain relation that links any two geospatial-temporal terms in the cross product. Applying this query, all geospatial-temporal terms that share a particular relation with another geospatial-temporal pair are returned, allowing for a focus on different relationships; an *isA* relation that defines a hierarchy, a *containedIn* relation that describes spatial inclusion, or a *componentOf* relation that describes a functional relationship. For example, a query that returns only those tuples that have a *componentOf* relation linking them is expressed as,

```
SELECT * FROM CrossProduct
WHERE Relation = "componentOf";
```

Only those tuples with geospatial-temporal pairs that are linked by a *componentOf* relation appear in the query result (Figure 4.3).

4.2 Queries Highlighting a Specific Combination of Terms

The previous queries search for a specific term from either one of the ontologies or for a specific kind of relation linking the geospatial-temporal pairs of terms. It is also possible to find all the terms related to a specific combination of the geospatial-temporal terms using SQL queries. So, for example, a query that returns tuples that correspond to the terms related to *Building* (a geospatial term) and *Morning* (a temporal term) is often useful (Figure 4.4). The relation resulting from the query is a list of tuples where either one of the geospatial-temporal terms, that is, either *Child1A* and *Child1B* or *Parent1A* and *Parent1B*, has the values *Building* and *Morning* respectively. These geospatial-temporal pairs are connected by any one of the three relationships, *isA*, *componentOf*, *containedIn*.

Child1A	Child1B	Relation	Parent1A	Parent1B
AcademicBuilding	AcademicSemester	componentOf	AcademicBuilding	AcademicYear
AcademicBuilding	ClassDay	componentOf	AcademicBuilding	AcademicSemester
AcademicBuilding	EarlyMorningHour	componentOf	AcademicBuilding	Morning
AcademicBuilding	Evening	componentOf	AcademicBuilding	NightTime
AcademicBuilding	LateNightHour	componentOf	AcademicBuilding	NightTime
AcademicBuilding	MidAfternoon	componentOf	AcademicBuilding	Afternoon
Artifact	AcademicSemester	componentOf	Artifact	AcademicYear
Artifact	ClassDay	componentOf	Artifact	AcademicSemester
Artifact	EarlyMorningHour	componentOf	Artifact	Morning
Artifact	Evening	componentOf	Artifact	NightTime
Artifact	LateNightHour	componentOf	Artifact	NightTime
Artifact	MidAfternoon	componentOf	Artifact	Afternoon
AthleticField	AcademicSemester	componentOf	AthleticField	AcademicYear
AthleticField	ClassDay	componentOf	AthleticField	AcademicSemester
AthleticField	EarlyMorningHour	componentOf	AthleticField	Morning
AthleticField	Evening	componentOf	AthleticField	NightTime
AthleticField	LateNightHour	componentOf	AthleticField	NightTime
AthleticField	MidAfternoon	componentOf	AthleticField	Afternoon
BaseballDiamond	AcademicSemester	componentOf	BaseballDiamond	AcademicYear
BaseballDiamond	AcademicSemester	componentOf	BaseballField	AcademicSemester
BaseballDiamond	AcademicYear	componentOf	BaseballField	AcademicYear
BaseballDiamond	Afternoon	componentOf	BaseballField	Afternoon
BaseballDiamond	Break	componentOf	BaseballField	Break
BaseballDiamond	CampusBreak	componentOf	BaseballField	CampusBreak
BaseballDiamond	ClassDay	componentOf	BaseballDiamond	AcademicSemester
BaseballDiamond	ClassDay	componentOf	BaseballField	ClassDay
BaseballDiamond	Day	componentOf	BaseballField	Day
BaseballDiamond	DayTime	componentOf	BaseballField	DayTime
BaseballDiamond	EarlyMorningHour	componentOf	BaseballDiamond	Morning
BaseballDiamond	EarlyMorningHour	componentOf	BaseballField	EarlyMorningHour
BaseballDiamond	Evening	componentOf	BaseballDiamond	NightTime
BaseballDiamond	Evening	componentOf	BaseballField	Evening
BaseballDiamond	ExamWeek	componentOf	BaseballField	ExamWeek
BaseballDiamond	FallBreak	componentOf	BaseballField	FallBreak
BaseballDiamond	Holiday	componentOf	BaseballField	Holiday
BaseballDiamond	LateNightHour	componentOf	BaseballDiamond	NightTime
BaseballDiamond	LateNightHour	componentOf	BaseballField	LateNightHour
:	:	:	:	:

Figure 4.3 Part of query result showing only *componentOf* relations

This particular query can be expressed as,

```
SELECT * FROM CrossProduct
WHERE (Child1A = "Building" and Child1B = "Morning")
OR (Parent1A = "Building" and Parent1B = "Morning");
```

The query result displays all the terms that are related to the *Building_Morning* pair. Parent terms as well as child terms are obtained using this query operation. The

resulting list of tuples can be used to understand finer impacts a given activity might have. For example, a fire drill may be planned for *Buildings* on campus in the *Morning*; using this query result it is possible to show the set of entities that are affected by this activity, including *StudentUnion*, *Library* and *ResidentialBuildings*, or observe *Library* at a more specific time (e.g., *EarlyMorningHour*) or more generic time (e.g., *DayTime*). In this result, there are ten tuples representing each relation along with the pair of geospatial-temporal terms that the relation links.

Child1A	Child1B	Relation	Parent1A	Parent1B
AcademicBuilding	Morning	isA	Building	Morning
Building	EarlyMorningHour	componentOf	Building	Morning
Building	Morning	isA	Building	DayTime
Building	Morning	isA	StationaryArtifact	Morning
DiningCommons	Morning	isA	Building	Morning
EntertainmentBuilding	Morning	isA	Building	Morning
Library	Morning	isA	Building	Morning
ResidentialBuilding	Morning	isA	Building	Morning
ServiceBuilding	Morning	isA	Building	Morning
StudentUnion	Morning	isA	Building	Morning

Figure 4.4 Query result displaying tuples where either of the two pairs of terms linked by a *Relation* in the *CrossProduct* is (*Building*,*Morning*)

4.3 Recursive Queries

More complex semantics can be captured through queries where a geospatial entity is modeled over a range or certain set of times. For example, it is possible to select a temporal class *DayTime* along with all its subclasses and then combine those temporal terms with a geospatial term such as *Library*. In this case, class *DayTime* and all its immediate subclasses *Morning* and *Afternoon*, as well as subclasses of *Morning* and *Afternoon*, which are *EarlyMorningHour* and *MidAfternoon* respectively, are combined with a single geospatial term *Library* (Figure 4.5). Analysis of this kind requires

recursion and a programming language (in this case, Visual C#.NET) is used to extend SQL and implement recursion.

```

Var X = "DayTime";

While (true) {

    SELECT * FROM CrossProduct

    WHERE Parent1B = X AND

    (Child1A = 'Library' AND Parent1A = 'Library');

    X = Child1B;

}

```

	Child1A	Child1B	Relation	Parent1A	Parent1B
	Library	Afternoon	isA	Library	DayTime
	Library	EarlyMorningHour	componentOf	Library	Morning
	Library	MidAfternoon	componentOf	Library	Afternoon
▶	Library	Morning	isA	Library	DayTime

Figure 4.5 Query result displaying *Library* during *DayTime*

This query result has four tuples that represents a geospatial term *Library* at *DayTime*, *Morning*, *Afternoon*, *MidAfternoon* and *EarlyMorningHour*. This query result can be used in the same way as the one that combines *Library* with all temporal terms, but in this case, the geospatial term is observed over a smaller range of selected temporal terms.

It is also possible to analyze a geospatial term along with all its subclasses at a given time. First a geospatial term and all its subclasses are selected (e.g., *LandArea* and subclasses *Road*, *BusStop*, *ParkingLot*, *ParkingSpace* and *Garden*). These terms are then combined with a temporal term of interest (e.g., *Weekend*). The result of this query

returns a range of related geospatial entities at a particular time (Figure 4.6). This query draws parallels with the query, described in chapter 4.1, which shows the entire geospatial ontology at a specific time. This is useful in cases where only a selected set of geospatial entities need to be considered.

```

Var X = "LandArea";

While (true) {

    SELECT * FROM CrossProduct

    WHERE Parent1A = X AND

    (Child1B = 'Weekend' AND Parent1B = 'Weekend');

    X = Child1A;

}

```

Child1A	Child1B	Relation	Parent1A	Parent1B
BusStop	Weekend	isA	LandArea	Weekend
Garden	Weekend	isA	LandArea	Weekend
ParkingLot	Weekend	isA	LandArea	Weekend
ParkingSpace	Weekend	isA	LandArea	Weekend
Road	Weekend	isA	LandArea	Weekend

Figure 4.6 Query result displaying *LandArea* and all its subclasses at *Weekend*

It is also possible to extend the query that displays a geospatial term over a range of times such that *all* geospatial terms are viewed over a range of times, e.g., *DayTime* and its subclasses. The `WHERE` clause in the original query is truncated such that those tuples with any value for *Child1A* and *Parent1A* are returned. This query also requires recursion and is expressed as,


```
Var X = "DayTime";  
While (true) {  
    SELECT * FROM CrossProduct  
    WHERE Parent1B = X;  
    X = Child1B;  
}
```

The query that displays a set of geospatial terms at a particular time can also be modified such that the set of geospatial terms are viewed over *all* possible times (Figure 4.8).

```
Var X = "LandArea";  
While (true) {  
    SELECT * FROM CrossProduct  
    WHERE Parent1A = X;  
    X = Child1A;  
}
```

Child1A	Child1B	Relation	Parent1A	Parent1B
AcademicBuilding	Afternoon	isA	AcademicBuilding	DayTime
AcademicBuilding	Afternoon	isA	Building	Afternoon
AcademicBuilding	DayTime	isA	Building	DayTime
AcademicBuilding	EarlyMorningHour	componentOf	AcademicBuilding	Morning
AcademicBuilding	MidAfternoon	componentOf	AcademicBuilding	Afternoon
AcademicBuilding	Morning	isA	AcademicBuilding	DayTime
AcademicBuilding	Morning	isA	Building	Morning
Artifact	Afternoon	isA	Artifact	DayTime
Artifact	Afternoon	isA	CampusObject	Afternoon
Artifact	DayTime	isA	CampusObject	DayTime
Artifact	EarlyMorningHour	componentOf	Artifact	Morning
Artifact	MidAfternoon	componentOf	Artifact	Afternoon
Artifact	Morning	isA	Artifact	DayTime
Artifact	Morning	isA	CampusObject	Morning
AthleticField	Afternoon	isA	AthleticField	DayTime
AthleticField	Afternoon	isA	SportsFacility	Afternoon
AthleticField	DayTime	isA	SportsFacility	DayTime
AthleticField	EarlyMorningHour	componentOf	AthleticField	Morning
AthleticField	MidAfternoon	componentOf	AthleticField	Afternoon
AthleticField	Morning	isA	AthleticField	DayTime
AthleticField	Morning	isA	SportsFacility	Morning
BaseballDiamond	Afternoon	componentOf	BaseballField	Afternoon
BaseballDiamond	Afternoon	isA	BaseballDiamond	DayTime
BaseballDiamond	DayTime	componentOf	BaseballField	DayTime
BaseballDiamond	EarlyMorningHour	componentOf	BaseballDiamond	Morning
BaseballDiamond	MidAfternoon	componentOf	BaseballDiamond	Afternoon
BaseballDiamond	Morning	componentOf	BaseballField	Morning
BaseballDiamond	Morning	isA	BaseballDiamond	DayTime
BaseballField	Afternoon	isA	AthleticField	Afternoon
BaseballField	Afternoon	isA	BaseballField	DayTime
BaseballField	DayTime	isA	AthleticField	DayTime
BaseballField	EarlyMorningHour	componentOf	BaseballField	Morning
BaseballField	MidAfternoon	componentOf	BaseballField	Afternoon
BaseballField	Morning	isA	AthleticField	Morning
BaseballField	Morning	isA	BaseballField	DayTime
BookStore	Afternoon	containedIn	StudentUnion	Afternoon
BookStore	Afternoon	isA	BookStore	DayTime
:	:	:	:	:

Figure 4.7 Query result displaying all geospatial terms at *DayTime* and all its subclasses

Child1A	Child1B	Relation	Parent1A	Parent1B
BusStop	AcademicSemester	isA	LandArea	AcademicSemester
BusStop	AcademicYear	isA	LandArea	AcademicYear
BusStop	Afternoon	isA	LandArea	Afternoon
BusStop	Break	isA	LandArea	Break
BusStop	CampusBreak	isA	LandArea	CampusBreak
BusStop	ClassDay	isA	LandArea	ClassDay
BusStop	Day	isA	LandArea	Day
BusStop	DayTime	isA	LandArea	DayTime
BusStop	EarlyMorningHour	isA	LandArea	EarlyMorningHour
BusStop	Evening	isA	LandArea	Evening
BusStop	ExamWeek	isA	LandArea	ExamWeek
BusStop	FallBreak	isA	LandArea	FallBreak
BusStop	Holiday	isA	LandArea	Holiday
BusStop	LateNightHour	isA	LandArea	LateNightHour
BusStop	LunchTime	isA	LandArea	LunchTime
BusStop	MidAfternoon	isA	LandArea	MidAfternoon
BusStop	Midnight	isA	LandArea	Midnight
BusStop	Month	isA	LandArea	Month
BusStop	Morning	isA	LandArea	Morning
BusStop	NightTime	isA	LandArea	NightTime
BusStop	Noon	isA	LandArea	Noon
BusStop	PublicHoliday	isA	LandArea	PublicHoliday
BusStop	SpringBreak	isA	LandArea	SpringBreak
BusStop	Thanksgiving	isA	LandArea	Thanksgiving
BusStop	TimeInterval	isA	LandArea	TimeInterval
BusStop	TimeMeasure	isA	LandArea	TimeMeasure
BusStop	TimePosition	isA	LandArea	TimePosition
BusStop	Week	isA	LandArea	Week
BusStop	Weekday	isA	LandArea	Weekday
BusStop	Weekend	isA	LandArea	Weekend
BusStop	Year	isA	LandArea	Year
Garden	AcademicSemester	isA	LandArea	AcademicSemester
Garden	AcademicYear	isA	LandArea	AcademicYear
Garden	Afternoon	isA	LandArea	Afternoon
Garden	Break	isA	LandArea	Break
Garden	CampusBreak	isA	LandArea	CampusBreak
Garden	ClassDay	isA	LandArea	ClassDay
:	:	:	:	:

Figure 4.8 Query result displaying *LandArea* and all its subclasses at all times

Another extension of these recursive queries combines a set of geospatial terms with a set of temporal terms. Figure 4.9 shows tuples resulting from the query that combines *LandArea* and all its subclasses with *DayTime* and all its subclasses. This type of query combines a subset of terms from the geospatial ontology with a subset of term from the temporal ontology.

Child1A	Child1B	Relation	Parent1A	Parent1B
BusStop	Afternoon	isA	BusStop	DayTime
BusStop	Afternoon	isA	LandArea	Afternoon
BusStop	DayTime	isA	LandArea	DayTime
BusStop	EarlyMorningHour	componentOf	BusStop	Morning
BusStop	EarlyMorningHour	isA	LandArea	EarlyMorningHour
BusStop	MidAfternoon	componentOf	BusStop	Afternoon
BusStop	MidAfternoon	isA	LandArea	MidAfternoon
BusStop	Morning	isA	BusStop	DayTime
BusStop	Morning	isA	LandArea	Morning
Garden	Afternoon	isA	Garden	DayTime
Garden	Afternoon	isA	LandArea	Afternoon
Garden	DayTime	isA	LandArea	DayTime
Garden	EarlyMorningHour	componentOf	Garden	Morning
Garden	EarlyMorningHour	isA	LandArea	EarlyMorningHour
Garden	MidAfternoon	componentOf	Garden	Afternoon
Garden	MidAfternoon	isA	LandArea	MidAfternoon
Garden	Morning	isA	Garden	DayTime
Garden	Morning	isA	LandArea	Morning
LandArea	Afternoon	isA	LandArea	DayTime
LandArea	EarlyMorningHour	componentOf	LandArea	Morning
LandArea	MidAfternoon	componentOf	LandArea	Afternoon
LandArea	Morning	isA	LandArea	DayTime
ParkingLot	Afternoon	isA	LandArea	Afternoon
ParkingLot	Afternoon	isA	ParkingLot	DayTime
ParkingLot	DayTime	isA	LandArea	DayTime
ParkingLot	EarlyMorningHour	componentOf	ParkingLot	Morning
ParkingLot	EarlyMorningHour	isA	LandArea	EarlyMorningHour
ParkingLot	MidAfternoon	componentOf	ParkingLot	Afternoon
ParkingLot	MidAfternoon	isA	LandArea	MidAfternoon
ParkingLot	Morning	isA	LandArea	Morning
ParkingLot	Morning	isA	ParkingLot	DayTime
ParkingSpace	Afternoon	isA	LandArea	Afternoon
ParkingSpace	Afternoon	isA	ParkingSpace	DayTime
ParkingSpace	DayTime	isA	LandArea	DayTime
ParkingSpace	EarlyMorningHour	componentOf	ParkingSpace	Morning
ParkingSpace	EarlyMorningHour	isA	LandArea	EarlyMorningHour
ParkingSpace	MidAfternoon	componentOf	ParkingSpace	Afternoon

Figure 4.9 Query result displaying *LandArea* and all its subclasses at *DayTime* and all its subclasses

With this work we show that many SQL queries can result in smaller more tractable subsets of complete cross product, for cases where complete cross products are not useful, these cases provide a reasonable space for analysis.

This kind of complex recursive query is similar to computing a cross product of ontologies that are smaller subsets of the individual base ontologies and is useful if a user is interested in reduced base ontologies. It is expressed as,

```
Var X = "LandArea";  
Var Y = "DayTime";  
While (true) {  
    SELECT * FROM CrossProduct  
    WHERE Parent1A = X AND Parent1B = Y;  
    X = Child1A;  
    Y = Child1B;  
}
```

4.4 Summary

In this chapter, SQL queries that can be carried out on the tuples in the *CrossProduct* were discussed. The types of SQL queries were divided into three different categories: queries highlighting a single term from one ontology combined with all terms from the other ontology, queries highlighting a specific combination of terms from the pair of ontologies, and recursive queries. An example of the first type of query is one that finds a specific geospatial term and combines it with all terms from the temporal ontology. Similarly, an example of the second category of query is one that searches for a specific combination of a geospatial term and a temporal term, returning the particular combination as well as those geospatial-temporal pairs that are directly linked to it. A

recursive query involves searching for a particular geospatial term as well as all its subclasses and combining them with a specific term from the temporal ontology. All the SQL queries return a set of tuples as a result. The numbers of tuples in the query results are remarkably less than the tuples in the complete cross product. The reduced number of tuples reiterates the fact that the SQL queries produce a smaller, more tractable result that is especially useful for a particular purpose.

The cross product as well as the relations that result from the application of SQL queries on the cross product can be visualized in the form of graphs. Visualizing the cross product as a set of tuples, that is, in standard relational database form is certainly one way to present the info stored in cross product. However a graphic presentation of this information is also useful and in the next chapter we show how graph visualization software is applied for visualizing the cross product.

Chapter 5

VISUALIZING THE CROSS PRODUCT

The result of the cross product of terms from two ontologies is stored in a relational database as explained in Chapter Four. The major challenge with large databases is to extract meaning from the data they contain: to discover structure, find patterns, and derive causal relationships (Stolte *et al.*, 2002). Visualizing data, in the form of graphs, is one way of representing structural information, and is a promising technique for the analysis of data in the databases. Data visualization enables a domain expert to quickly and efficiently analyze the contents of a database from many different perspectives (Hilderman *et al.*, 1998). To make the visualization of data especially effective, there needs to be close integration of visual presentation and database queries (Stolte, 2003). In this chapter, the cross product is represented visually as a graph using graph visualization software tools. Graph visualization has been shown to be useful for software engineering, database and web design, networking, as well as for visual interfaces for many other domains. The benefit of this step for this work is that structure in data is more evident through visual means and hence visualization acts as a useful analysis tool.

5.1 Visualizing the Cross Product using Graphviz

In this thesis, open source graph visualization software, Graphviz, is utilized to visualize the information contained in the relational database. This process of visualization involves two main steps: building a *.dot* file, and creating an image file. In order to build a *.dot* file, the values of attributes in each tuple in the *CrossProduct* relation are read and written into a file using a graph description language, *DOT*, a plain text graph description language. This language offers a simple and user-friendly way of translating relational data into graphs. Once the *.dot* file has been created, Graphviz is used to read the file and render it visually. The Graphviz software platform consists of a set of tools that can generate and process *DOT* files.

5.1.1 About Graphviz

Graph Visualization Software (Graphviz) is a package of open source tools initiated and developed by AT&T Research Labs for drawing graphs specified in *DOT* language scripts. Graphviz is free software licensed under Common Public License. This graph visualization tool offers web and interactive graphical interfaces, and auxiliary tools, libraries, and language bindings so that its services and functions can be used through different programming language platforms. (<http://www.graphviz.org/>). Graphviz has versions developed for Macintosh as well as Windows operating systems. One of the main advantages of Graphviz is that it takes descriptions of graphs in a simple text language, and produces diagrams in several useful formats, such as images and SVG for

web pages, postscript for inclusion in PDF or other documents, or display in an interactive graph browser. Graphviz has many helpful options for specifying and modifying colors, fonts, tabular nodes, line styles, hyperlinks, and custom shapes to build different graphs such as ER diagrams, process diagrams, or network graphs.

Graphviz is built around a graph description language named *DOT*, and a set of tools that can generate and process *DOT* files:

- *dot*: *dot* is a command-line tool to lay out directed graphs into a variety of output formats. *dot* creates hierarchical or layered drawings of directed graphs. The layout algorithm aims edges in the same direction (top to bottom, or left to right) and then attempts to avoid crossings and reduce edge length.
- *neato*: *neato* is the counterpart of *dot* for undirected graphs
- *twopi*: *twopi* is for radial graph layouts
- *circo*: *circo* is for circular graph layouts
- *fdp*: *fdp* is another layout engine for undirected graphs
- *dotty*: *dotty* is a graphical user interface to visualize and edit graphs
- *lefty*: *lefty* is a programmable widget that displays *DOT* graphs and allows the user to perform actions on them with the mouse

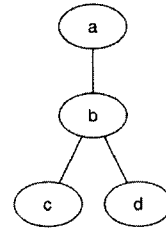
5.1.2 Building a .dot file

Creating a *.dot* file involves reading the values of each attribute from all the individual tuples in the *CrossProduct* relation. Once the values have been read, they are written into a text file ending with a *.dot* extension. Understanding the structure of *DOT* is a part of

the process of building a *.dot* file. The simplest kinds of graphs that *DOT* can be used to describe are undirected graphs. The syntax of an undirected graph is shown in Figure 5.1a while an example of an undirected graph is shown in Figure 5.1b.

```
graph graphname
{
    a - - b - - c;
    b - - d;
}
```

(a)



(b)

Figure 5.1 Undirected graph (a) Syntax for an undirected graph, and (b) drawing of undirected graph described in Figure 5.1a

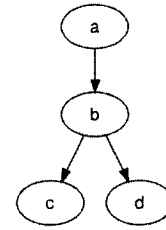
DOT can also describe directed graphs. Figure 5.2a and 5.2b show the syntax for a directed graph and a rendered directed graph respectively. The syntax for directed graph is similar to that for an undirected graph, except the digraph keyword is used to begin the graph, and an arrow (\rightarrow) is used to show relationships between nodes.

```

digraph graphname
{
    a -> b -> c;

    b -> d;
}

```



(a)

(b)

Figure 5.2 Directed graph (a) Syntax for a directed graph, and (b) drawing of directed graph described in Figure 5.2a

In this work, directed graphs are used to represent the geospatial-temporal pairs and the relations that connect these terms. Following the syntax presented by *DOT* for directed graphs, the value of the each attribute in a tuple is read and written into a file in the following format,

```

Child1A_Child1B -> Parent1A_Parent1B [label = Relation];

```

Each tuple in *CrossProduct* is read such that values of Child1A, Child1B, Relation, Parent1A and Parent1B are obtained and written to a file in the above format such that the file can be read by Graphviz. A part of the cross product showing geospatial-temporal pairs, *Object_TimeMeasure*, *Object_TimePosition*, *Object_TimeInterval*, and *Object_Noon* along with the relations that links those pairs expressed in DOT is shown as an example below,

```

digraph G {
rankdir=BT;

node [shape = plaintext, fontname = Helvetica, fontsize = 8];

edge [arrowsize = .5, fontname = Helvetica, fontsize = 8];

Object_TimeMeasure;

Object_TimePosition;

Object_TimePosition->Object_TimeMeasure[color = blue, label = isA];

Object_TimeInterval;

Object_TimeInterval->Object_TimePosition[color = blue, label = isA];

Object_Noon;

Object_Noon->Object_TimePosition[color = blue, label = isA];

. . . }

```

The keyword `digraph` describes graph G as a directed graph. `rankdir` describes the orientation of the directed graph. It takes different values including, *BT* and *LR* which means top to bottom orientation or left to right orientation respectively. `node[...]` and `edge[...]` describe properties of the nodes and edges respectively. Some of the attributes of a node are *shape* (e.g., `plaintext`, `ellipse`), *fontname* (e.g., `Helvetica`, `TimesNewRoman`), and *fontsize* (e.g., `8`, `10`). Similarly, a number for attributes of an edge can be described and they are *arrowsize*, *fontname*, and *fontsize*. A node is created when its name first appears in the file, e.g., `Object_TimeInterval`, while an edge is created when nodes are joined by the edge operator, e.g., `->`. Once the `.dot` file is created, the command-line tool `dot` provided by Graphviz reads it and renders it in the form of an image file e.g., JPEG.

5.1.3 Creating an Image File

The *DOT* language defines a graph, but does not provide facilities for rendering the graph. There are several programs that can be used to render, view, and manipulate graphs in *DOT* language. For this research, Graphviz is selected as the tool for rendering the cross product as a graph. Graphviz has a number of tools for rendering directed as well as undirected graphs and for our work we use the tool called *dot*. *dot* runs as a command line program, or with a compatible graphical interface (Figure 5.3). Using the graphical user interface provided by Graphviz, an input file (with a *.dot* extension) can be selected for rendering. Once a *.dot* file has been chosen, a suitable output file type is chosen (e.g., JPEG). To complete the output process, Do layout button is pressed. The image file is stored in the path specified in the Output file input box. After a successful rendering, a message is shown in the Output box (`dot said: Layout ended successfully.`). In case of syntax errors, an error message is displayed and the rendering process can be repeated after the errors have been corrected.

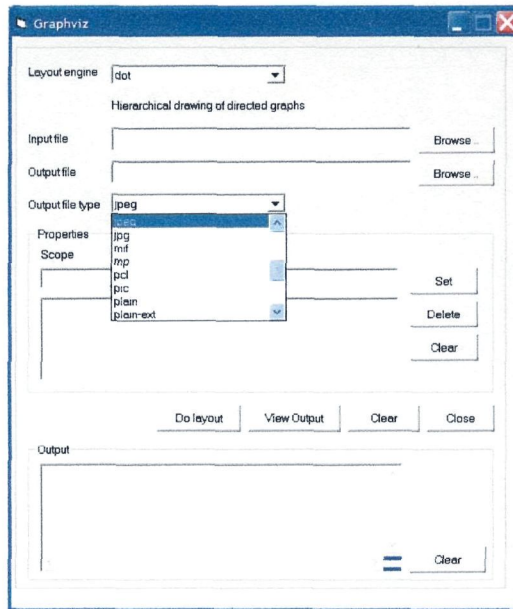


Figure 5.3 User interface for *dot* layout engine in Graphviz

Each pair of terms in the cross product corresponds to a node in the Graphviz visualization (Figure 5.4). Each relationship that links any two pairs of ontological terms together is denoted by an edge. For example, nodes *Object_TimeMeasure* and *Object_TimePosition* are linked by an *isA* relation.

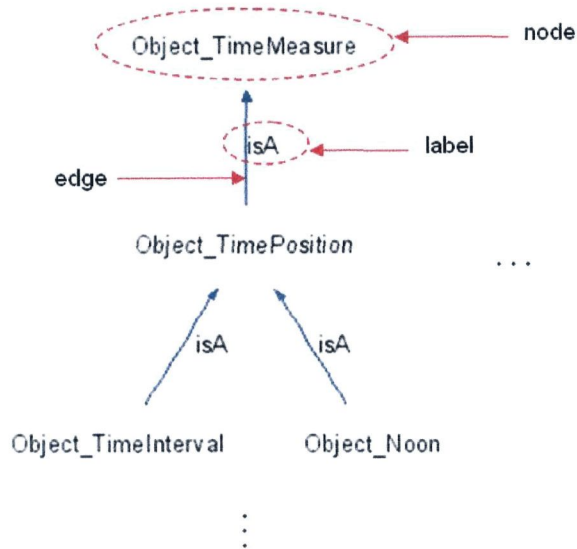


Figure 5.4 Displaying some of the nodes (geospatial-temporal pairs) and edges (relations) in a cross product

When visualizing a large cross product, the labeled nodes can be replaced with an icon (e.g., a filled square). In this way, the actual structure of the cross product is preserved for viewing by a user, although the class names are abstracted. Figure 5.5 shows the complete cross product computed from the geospatial and the temporal ontologies in Section 3.4. Given the size of this cross product, replacing the nodes with icons allows the entire cross product to be viewed at once.

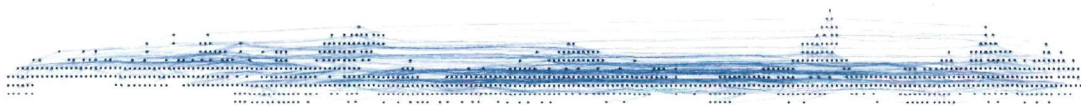


Figure 5.5 Visual representation of entire cross product using Graphviz. Icons represent the pairs of geospatial-temporal terms

Though the use of icons in lieu of actual terms is useful from a space perspective, Graphviz also allows us to view the entire cross product with labeled nodes and edges. Graphviz also offers tools for zooming, so smaller portions of the cross product (with labeled nodes and edges) can be enlarged and viewed through zooming (Figure 5.6).

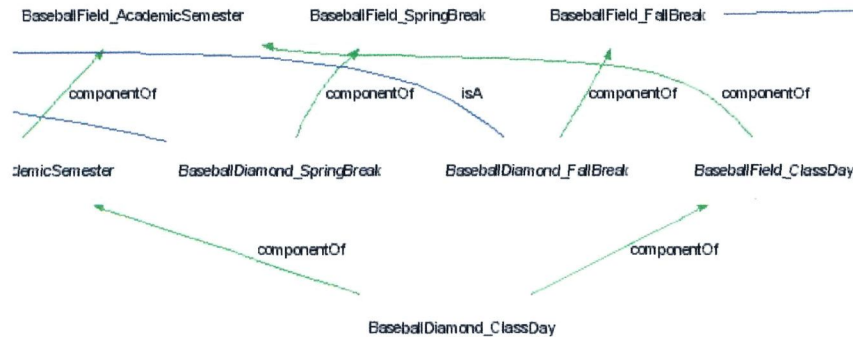


Figure 5.6 Part of the cross product displayed using Graphviz

5.2 Visualizing Refinements of Cross Product

Queries on the cross product, for example, the results of the queries discussed in Chapter Four can be visualized using Graphviz. The use of a graphical tool allows for a visualization and understanding of parent and child relationships between any two geospatial-temporal terms in the cross product. As an example, the visualization of the query that returns combinations of a single geospatial term *Library* over all temporal terms can be depicted in form of a graph (Figure 5.7). Each node in the graph is a combination of geospatial and temporal terms. The nodes are linked by labeled edges that represent the relation between geospatial-temporal terms. Visualizing the query result helps a user to look at *Library* from all possible temporal perspectives. The query result

in relational form contains the same data as the visual representation of the data does, but a graph adds a structural component to the data. Users can see the immediate geospatial-temporal terms that are connected pairs, for example, *Library_DayTime* has two children *Library_Morning* and *Library_Afternoon*, and is a child of *Library_TimeInterval*. In addition, it is possible to view all other nodes including the top-most node and all the leaf nodes. An IT Supervisor, for example, not only sees the combination of *Library* with every temporal class, but also understands how all geospatial-temporal pairs of terms in the query result are related to one another. This enables an IT Supervisor to analyze the staffing needs at different times in the *Library*.

Similarly, the visualization of the query that returns combinations of the geospatial term *Building* with temporal term *Morning* from the temporal ontology is shown in Figure 5.8. The visualization of this query shows all the terms that are related to the *Building_Morning* pair. For example, *Building_Morning* is a child of *Building_DayTime*, and a parent of *Building_EarlyMorningHour*. A *CrossProduct* relation gives us a list of tuples, but visualization adds structure to our understanding of the cross product. It is easier to comprehend all the parent nodes as well as the child nodes to which the geospatial-temporal term *Building_Morning* is linked. The visualization provides an understanding of what geospatial entities can be affected due to a given activity at a location and time.

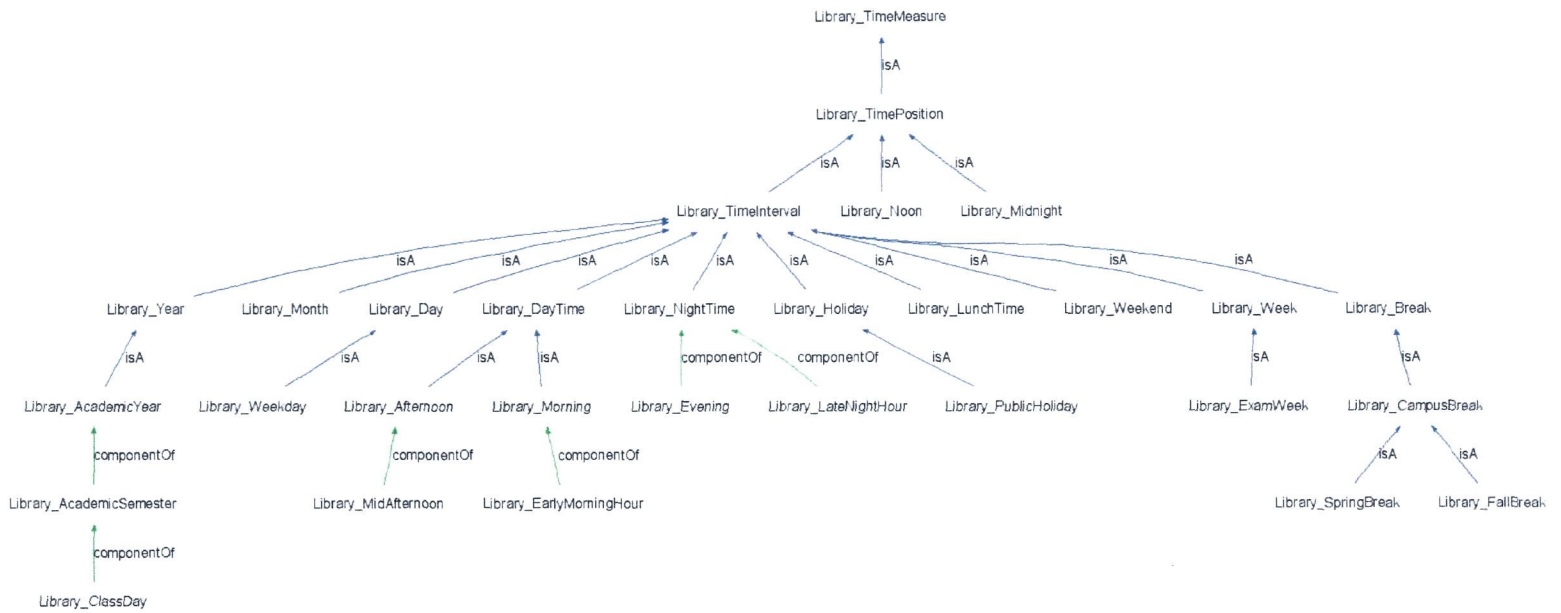


Figure 5.7 Graphical representation of *Library* with all temporal terms

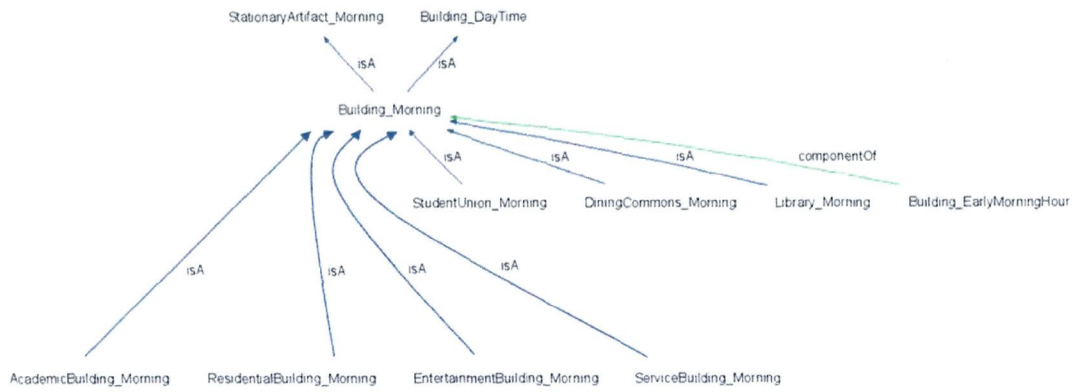


Figure 5.8 Graphical representation of *Building_Morning* along with all geospatial-temporal terms that are directly related

The results of recursive queries discussed in Chapter Four can also be visualized using Graphviz. The query result that combines *Library* with *DayTime* and all its subclasses, *Morning*, *Afternoon*, *EarlyMorningHour* and *MidAfternoon* is easily viewed and analyzed (Figure 5.9).

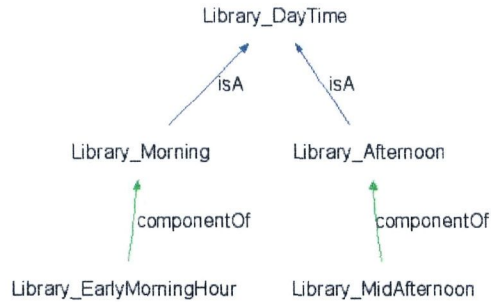


Figure 5.9 Graphical representation of *Library* during *DayTime* (*DayTime* along with all its subclasses)

5.3 Summary

Chapter 4 explained the process of combining a pair of ontologies to generate a cross product in the form of a relation. In this chapter, the steps involved in taking the cross product from a relational database platform to a visual medium are presented. The first step involves reading the values of attributes in the *CrossProduct* relation and writing the values to a text file. The values are written using DOT, which is a plain text graph description language. The syntax of DOT, whose understanding is essential to correctly represent directed graphs, is also discussed. Once the dot file is created, open source graph visualization software, Graphviz, is used to render the graphs and produce image outputs, for example, in JPEG format.

Using graphs to visualize the cross products provides an intuitive way to analyze the cross products as well as results of the SQL queries. The structure of the cross product is maintained, and at the same time, the graphs help users to find patterns and understand relationships that exist amongst the nodes in the cross product, such as parent-child relationships between any two geospatial-temporal pairs of terms. In addition, visualizing the cross products helps to highlight the role of the relations and the connectivity that exists among terms in the cross product.

Using SQL queries on the cross product returns selected tuples from the relation, which contains geospatial as well as temporal terms, both parent and child along with the ontological relations. These queries select tuples related to a particular term (geospatial or temporal or both). In the next chapter, a different operation, filtering, is discussed. Filtering removes tuples containing a specific term in order to eliminate combinations

that do not contain any attribute data or are of too coarse granularity and results in smaller cross product. In the following chapter, we show how filtering is implemented using different SQL operations and discuss some of the ramifications that result from filtering a cross product.

Chapter 6

FILTERING CROSS PRODUCTS

A cross product of the terms from a pair of ontologies gives us a complete, exhaustive set of combinations of terms from each ontology. In addition to the queries on the cross product as presented in Chapter 4, there are other operations that are possible on the cross product. For example, it is possible to remove a specific combination of geospatial-temporal terms (e.g., *BaseballField_Midnight*) or a term highlighting either one of the domains (e.g., *<geospatial term>_Weekend* or *BookStore_<temporal term>*) from the cross product. In this thesis, the process of removing terms from the cross product is called filtering. Filtering is necessary, for example, in cases where terms at a coarse granularity need to be eliminated, perhaps due to a lack of data. Filtering is also useful when terms related to a certain granularity, which is not relevant also need to be removed. For example, if we are interested in analyzing a campus domain at times when classes are in session, then it is useful to be able to remove those geospatial-temporal terms that deal with temporal terms, such as *Holiday* and *Break*. Filtering terms from a cross product results in a smaller set of tuples.

A number of cases may arise that need attention as a result of filtering single or multiple geospatial-temporal term(s) from the cross product. The resulting structure is dependent on the relations the geospatial-temporal term shares with other terms in the cross product. For instance, if the geospatial-temporal term that is filtered is related to

two or more child terms but no parent terms, the result will be multiple terms without a parent; requiring a new common parent to be assigned to the child terms, that is, insertion of new tuples in the cross product. Another example is the filtering of a geospatial-temporal term that is related to one or more parent terms and one or more child term(s). Filtering of such terms results in the removal of those tuples from the *CrossProduct* that have the particular geospatial-temporal term as child terms or parent terms.

Graphviz can be used to visualize the filtered cross product. A graphical representation of the filtered cross product in some cases will reveal *cuts* or discontinuities that occur when geospatial-temporal terms initially linked to each other by another term become disconnected by the removal of the central geospatial term.

6.1 Filtering Nodes of Coarse Granularity

In many cases, not *all* of the terms in the cross product relation are needed for analysis. In these cases, for example, terms corresponding to a granularity coarser than a selected threshold geospatial-temporal term can be filtered from the cross product. Lack of data for terms of coarse granularity, or the need to focus on a more specific part of the cross product are motivations for filtering. For example, filtering all geospatial-temporal terms coarser than *<geospatial term>_TimeInterval* is one example of removing coarse terms from the cross product. This type of filtering results in the elimination of terms representing multiple granularities, *<geospatial term>_TimeMeasure* and *<geospatial term>_TimePosition* (e.g., *Object_TimeMeasure*, *Library_TimePosition*, and *CampusObject_TimePosition*).

Filtering is implemented in a relational database setting through the application of different SQL operations, extended with the help of programming language. The above example can be expressed as,

```
Var X = "TimeInterval";  
Num = 1;  
Table = "CrossProduct";  
While(true) {  
    SELECT distinct Parent1B  
    FROM Table  
    WHERE (Child1B = X AND Child1B <> Parent1B);  
    X = Parent1B;  
  
    SELECT * INTO Temp[Num] FROM [Table]  
    WHERE NOT (Child1B = X OR Parent1B = X);  
  
    Table = Temp[Num];  
    Num = Num + 1;  
}
```

It can be observed from the extended SQL operations that the first query extracts the terms that are coarser than *TimeInterval*. For example, executing the query for the first time results in $X = TimePosition$. Once the parent term (i.e., coarser term) is obtained, the second SQL operation extracts a list of tuples that do not contain *TimePosition*. For example, tuples that have values of $Child1B = TimePosition$ or

Parent1B = *TimePosition* are eliminated. The queries are repeated until a term without any parent terms is reached (term *TimeMeasure* in our example).

6.2 Filtering Nodes of Specific Granularity

Another motivating factor for filtering is the need to remove specific geospatial-temporal terms from the cross product. Filtering a specific combination of geospatial-temporal terms, e.g., *BookStore_Midnight*, is undertaken when certain combinations existing in the cross product are not relevant, or do not contain any data. Taking the example of an IT Supervisor who makes use of *Library_<temporal term>* combinations to evaluate staffing needs for a *Library* at different times, terms like *Library_Holiday* are of no interest since the *Library* is closed on those days. This filter operation can be expressed using SQL as shown below.

```
SELECT * FROM CrossProduct
WHERE
NOT (Child1B = "Holiday" OR Parent1B = "Holiday")
AND Child1A = "Library" AND Parent1A = "Library";
```

Similarly, geospatial-temporal terms belonging to a particular granularity are filtered if they are not of any interest for a particular kind of analysis. For example, all terms related to a specific temporal granularity like *Midnight* or a particular geospatial granularity like *BookStore* can be removed if not found relevant, producing a smaller, more appropriate combined framework. For this case, the SQL statement is expressed

such that only those tuples that do not have the value *BookStore* for attributes *Child1A* and *Parent1A* are selected.

```
SELECT * FROM CrossProduct
WHERE
NOT (Child1A = "BookStore" OR Parent1A = "BookStore");
```

6.3 Composition of Relations

Filtering terms from a cross product arises in a number of different scenarios as introduced in Sections 6.1 and 6.2. Removing terms, in these cases, results in different structural issues for the cross product. In general, geospatial-temporal terms in a cross product can be categorized into three kinds:

- Terms related to no parent term but to two or more child terms (Figure 6.1a)
- Terms related to one or more parent terms but to no child term (Figure 6.1b)
- Terms related to one or more parent terms and to one or more child terms (Figure 6.1c)

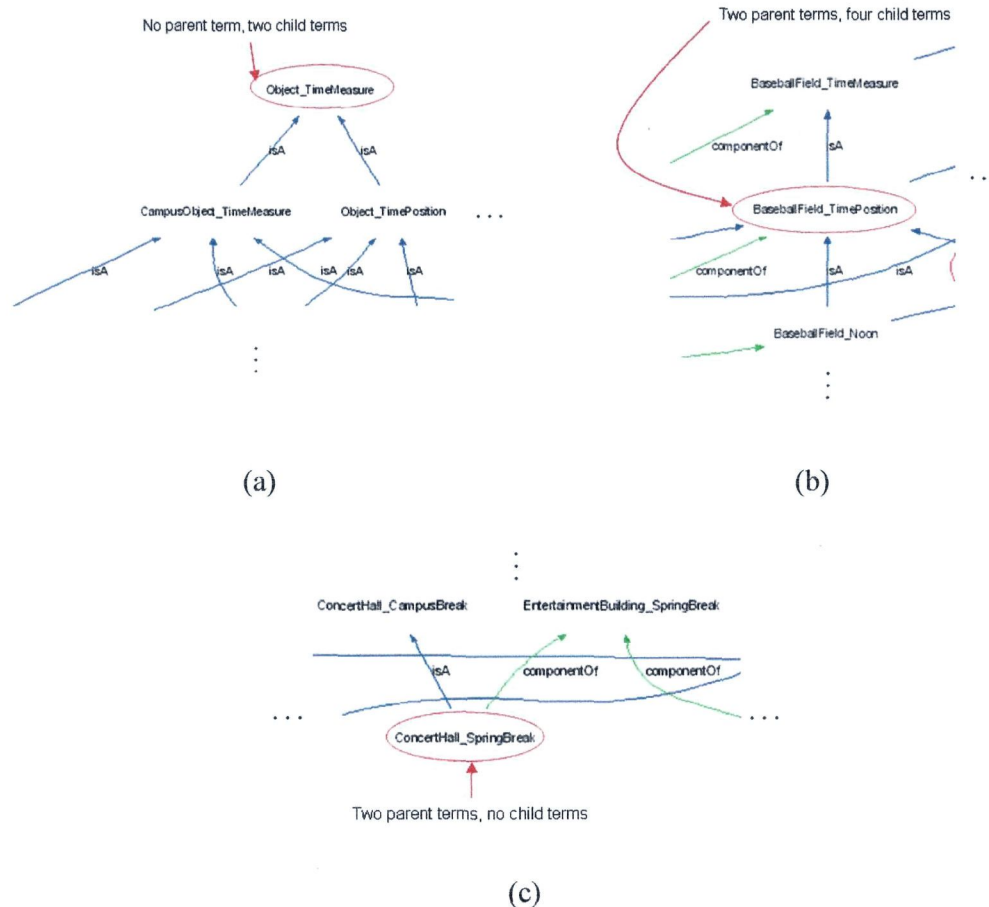


Figure 6.1 Graphviz visualization of cross product. Showing (a) terms related to no parent term but to two or more child terms, (b) terms related to one or more parent terms but to no child term, and (c) terms related to one or more parent terms and to one or more child terms

6.3.1 Filtering Terms with No Parent Term but Two or More Child Terms

Filtering those geospatial-temporal terms from the cross product that are related to no parent term, but related to two or more child terms may result in more than one geospatial-temporal term in the cross product that are related to no parent. Such cases might arise when filtering pairs of terms that are of coarse granularity or those that do not

generate interest in a given analysis. For example, removing a geospatial-temporal term, *Object_TimeMeasure* involves execution of the following SQL operation,

```
SELECT * FROM CrossProduct
WHERE
NOT((Child1A = "Object" AND Child1B = "TimeInterval")
OR (Parent1A = "Object" AND Parent1B = "TimeInterval"));
```

If the goal is to preserve the linkages as much as possible in the cross products then when filtering of this type gives us more than one geospatial-temporal term without any parent terms, it may be desired to insert new tuples such that those geospatial-temporal terms are linked to a new common parent term. For example, a new tuple added to the relation will have the following values for the attributes: *Child1A = CampusObject*, *Child1B = TimeMeasure*, *Relation = isA*, *Parent1B = ANY* and *Parent1A = ANY*.

6.3.2 Filtering Terms with One or More Parent Term but No Child Term

It is also possible to filter those geospatial-temporal terms that are related to one or more parent terms but no child term. This operation results in the removal of that particular combination of terms as well as the relation(s) linking the removed term to the parent class(es). This type of filtering is useful in cases where a particular geospatial-temporal term does not contain any attribute data. For example, removing the geospatial-temporal term *BaseballDiamond_ClassDay* from the cross product, involves execution of the following SQL operation,

```

SELECT * FROM CrossProduct
WHERE NOT
  ((Child1A = "BaseballDiamond" AND Child1B = "ClassDay")
OR
  (Parent1A = "BaseballDiamond" AND Parent1B = "ClassDay"));

```

Eliminating *BaseballDiamond_ClassDay* from the cross product results in the removal of tuples from the cross product.

6.3.3 Filtering Terms with One or More Parent as well as Child Terms

Removal of those geospatial-temporal terms that have one or more parent terms *and* one or more child terms results in the removal of tuples from *CrossProduct* that have the particular geospatial-temporal term as child terms or parent terms. Filtering this category of geospatial-temporal terms from the cross product also involves removal of relations that link the particular term to its parent terms and child terms. For example, filtering *BookStore_AcademicSemester* from the cross product results in the elimination of tuples where the relation *componentOf* links *BookStore_ClassDay* to *BookStore_AcademicSemester*, *containedIn* links *BookStore_AcademicSemester* to *StudentUnion_AcademicSemester*, and *componentOf* links *BookStore_AcademicSemester* to *BookStore_AcademicYear* (Figure 6.2). The SQL expression of this filter operation is,

```

SELECT * FROM CrossProduct
WHERE NOT
((Child1A="BookStore" AND Child1B="AcademicSemester")
OR
(Parent1A="BookStore" AND Parent1B="AcademicSemester"));

```

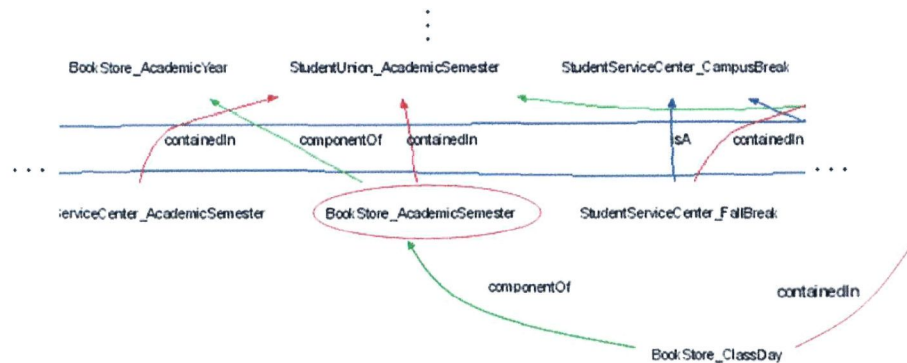


Figure 6.2 Part of the cross product showing the geospatial-temporal term *BookStore_AcademicSemester* and the relations linking it to other terms in the cross product

6.3.4 Composition Rules

Visualizing the filtered cross products exposes discontinuities in the resulting graph view. Using Graphviz for visualizing the tuples obtained after filtering *BookStore_AcademicSemester* from the cross product (example presented in Section 6.3.3), the path that existed through *BookStore_ClassDay*, *BookStore_AcademicSemester* and *BookStore_AcademicYear*, is no longer represented. In the cross product, *BookStore_AcademicSemester* is related to *BookStore_ClassDay* and *StudentUnion_AcademicSemester* (Figure 6.2). If this term is filtered, however, then this

linkage is lost and *BookStore_ClassDay* no longer is related to *BookStore_AcademicYear* (Figure 6.3). In order to maintain a relationship between two terms at different granularities, composition of relations is required. The parent term of the filtered term has to be related to the child term of the same geospatial-temporal term so that the relations between different terms from varying levels of granularity are not lost.

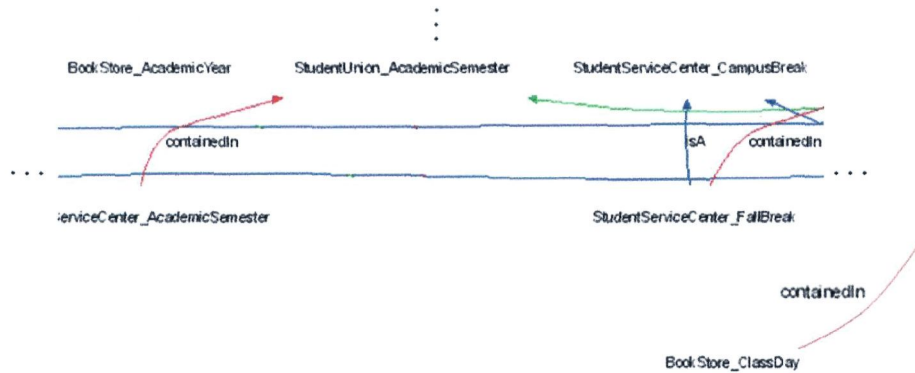


Figure 6.3 Part of the cross product after filtering *BookStore_AcademicSemester*

The composition of relations refers to determining which relation hold between the parent and child terms of the filtered geospatial-temporal term. This requires rules that establish which relation has precedence. It is understood that a hierarchical ordering exists among different types of relations (Winston *et al.*, 1987). The types of relations that exist in the domain ontologies used in this thesis are *isA*, *componentOf* and *containedIn*. An *isA* relation is a case of class inclusion (or taxonomic relation), *componentOf* an example of mereological inclusion, while *containedIn* is a case of spatial inclusion (or topological relation). The ordering among these different relations corresponds to:

class inclusion > merological inclusion > spatial inclusion (Winston *et al.*, 1987)
 that is, *isA* > *componentOf* > *containedIn*

Based on this hierarchical ordering of relations, we can determine the complete set of compositions, where composition operation is denoted by the symbol \otimes (Table 6.1):

Table 6.1 Composition rules for ontological relations

\otimes	<i>isA</i>	<i>componentOf</i>	<i>containedIn</i>
<i>isA</i>	<i>isA</i>	<i>componentOf</i>	<i>containedIn</i>
<i>componentOf</i>	<i>componentOf</i>	<i>componentOf</i>	<i>containedIn</i>
<i>containedIn</i>	<i>containedIn</i>	<i>containedIn</i>	<i>containedIn</i>

Using these composition rules, performing a composition of an *isA* relation with a *componentOf* relation results in a *componentOf* relation,

$$\textit{containedIn} \otimes \textit{componentOf} \rightarrow \textit{containedIn}$$

For example, relating the term *BookStore_ClassDay* with *StudentUnion_AcademicSemester* after *BookStore_AcademicSemester* is filtered, involves the composition of relations *componentOf* and *containedIn*. Using the composition rules in Table 6.1, *BookStore_ClassDay* is related to *StudentUnion_AcademicSemester* by a *containedIn* relation and the cut in the graph is

eliminated (Figure 6.4). If additional relations are introduced, then the ordering rules would need to be extended. Further research will be required to determine the hierarchy of the new relations introduced in the ontology, for example, if a new relation *before* is added, an ordering needs to be established that will resolve which relation will take precedence when composed with *before*. From the perspective of a relational database, filtering results in the elimination of tuples, while composition of relations results in the insertion of new tuples.

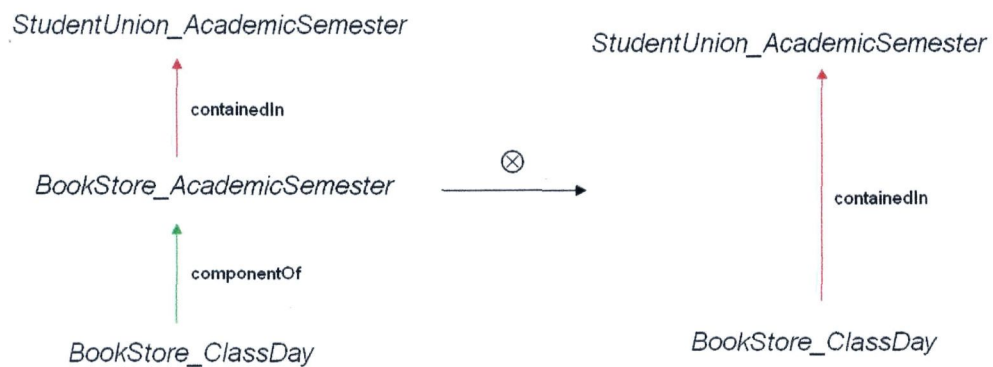


Figure 6.4 Applying composition rules in Table 6.1 to remove the discontinuity in the cross product due to filtering of the term *BookStore_AcademicSemester*

6.4 Summary

In this chapter, two different rationales for filtering terms from a cross product were discussed. The first case describes eliminating those terms from the cross product that are of coarser granularity than a chosen term. The second case discussed filtering a specific combination of geospatial-temporal terms from the cross product. This type of filtering is useful when some of the terms or a specific granularity is not relevant for an analysis.

Both of these cases result in a cross product that is more tractable for analysis and viewing.

When the cross product that results from filtering is viewed in Graphviz, various structural issues become apparent depending on the type of filtering that is applied to the framework. Removing a geospatial-temporal term that does not have a parent but has two or more children, results in multiple terms that do not have a parent class. Another case is where geospatial-temporal terms possess one or more parent but no children. When these types of terms are filtered from a cross product, the term and the relation linking that term to its parent are removed. Finally, geospatial-temporal terms that have one or more parents and one or more children can also be filtered. In this particular case, the graph view of the result shows a cut in the graph structure.

In this case, the composition of relations can be applied to maintain relations between terms. Composition of relations refers to rules based on the hierarchical ordering of the ontological relations. These composition rules can be used as the foundation for performing compositions of relations in the cross product.

In the next chapter, we present the conclusions of this thesis. Major findings, advantages, and ideas for future research are discussed.

Chapter 7

CONCLUSIONS AND FUTURE WORK

In this final chapter, the research presented in this thesis is summarized, the major contributions are highlighted, and some possible extensions of this research are presented as topics for future work.

7.1 Summary of the Thesis

In this thesis, an automated method for combining a pair of orthogonal ontologies is described. The ontologies used in this work are domain ontologies, one representing a geospatial domain (in this case, a university campus) and the other representing a temporal domain. Both of the ontologies used in this research have been derived from SUMO, which is one of the largest publicly available formal upper ontologies. The motivation behind this research is to use existing ontologies as a basis for generating a more comprehensive spatio-temporal framework that represents all possible combinations of both the geospatial and temporal terms. This new framework is useful, for example, for an intelligence analyst to explore geospatial terms for a particular domain over all possible temporal granularities. Such a combination promotes reuse of available ontologies and allows domain experts to focus on their area of expertise.

Protégé, an open source ontology editor and knowledge base framework, is used to model the geospatial and temporal ontologies. The ontologies are expressed using OWL, which is based on the general purpose mark-up language XML. Many of the publicly available ontologies are described using OWL, and can be processed and interpreted by machines and shared across different information systems.

In this work, cross products are used to combine the pair of ontologies. The cross product of the geospatial and temporal ontologies gives a complete set of pair-wise combinations of terms from the two ontologies. Combining the two ontologies involves three main steps: parsing the OWL ontologies to extract the values of the classes, subclasses and relations; importing the values of classes and relations into a relational database; and using SQL operations to compute the cross product. A programming language, Visual C#.NET is used to implement this process.

The cross product contains all possible geospatial-temporal terms as well as the relations (*isA*, *componentOf*, and *containedIn*) that link those terms. Various SQL queries can then be applied on the resulting cross product to analyze the combined framework from different perspectives. Queries can return a particular geospatial term over all temporal granularities as well as return a specific geospatial term over a selected range of temporal terms. More complex queries call for recursion and thus require an extension using Visual C#.NET to the query. The cross products, as well as the results of the SQL queries on the cross products, are visualized in form of a graph using open source graph visualization software, Graphviz. One advantage of using Graphviz is that this tool makes use of simple text language to describe the graphs. The visualization process involves reading the attribute values and writing them into a file, which is then read and rendered

by Graphviz. Use of such a graphical tool allows for visualization and understanding of parent-child relationships between any two geospatial-temporal terms in the cross product.

A method to eliminate tuples from the cross product that are irrelevant for a particular analysis or that contain terms belonging to coarse granularity is also described. This operation is termed *filtering* and results in a reduced, more tractable reasoning space. Since cross products can be very large, it is likely that not all terms in the cross product are necessary for analysis. For this reason, it is expected that the filtering operation will be an important tool for working with these frameworks.

7.2 Major Results

One of the major contributions of this thesis is the development of a tool that automatically combines two orthogonal ontologies based on computing the cross product. The result of this combination is a comprehensive spatio-temporal framework. Computing a cross product involves combining every term from the geospatial ontology with all terms from the temporal ontology, thus making it possible to represent geospatial domain at all temporal granularities. The resulting cross product is a multi-granular, unified framework that contains knowledge from both domains. It is an integrated reasoning framework that can be browsed and queried.

The second result of this thesis is the development of a method for determination of the types of queries possible on a cross product. This research categorizes possible queries into three different types. An example of such a query is one that combines a

specific geospatial term such as *Library* will all temporal terms to consider *Library* over all possible kinds of time. This query supports the hypothesis which states that *a multi-granular, unified framework results from taking the cross product of a pair of orthogonal ontologies*. One of the main advantages of using a relational database approach to combine ontologies is the ability to query the cross product. Based on these SQL queries, higher order inferences are drawn about a domain.

A third result of this thesis is the method to visualize the cross products. The steps involved in visualizing the cross products as well as results of SQL queries on the cross products are described. Visualizing the spatio-temporal framework enables us to understand any existing patterns in the database, and discover structure from the relations, which otherwise are unseen. Thus, visualization provides us with an intuitive method to analyze the cross products as well the results of SQL queries on the cross products.

The fourth and final result of this thesis is the method (relational database approach) it provides to filter tuples from the cross product. The integrated spatio-temporal framework is a complete and exhaustive combination of terms from both domain ontologies. Not all combinations in the cross product are useful for every analysis owing to the fact that some of the terms are of coarse granularity and some terms may not be associated with any data, while other terms are just not relevant for a given scenario. This thesis introduces filtering as a method to eliminate irrelevant terms from the cross product. The filtered cross products when visualized may reveal cuts or discontinuities in the cross product. This work also lays a theoretical foundation for determining rules that need to be followed to remove cuts in the cross product.

7.3 Future Work

Some of the possibilities for future work are discussed in this section. One possibility for future work is the automation of filtering. Filtering involves two main steps. The first step involves the execution of an SQL operation to eliminate tuples that display terms that are too coarse or irrelevant, while the second step requires composition of relations. Future work could focus on how to automate this process where the cuts in the cross product are automatically discovered. For example, when filtering a term from the cross product that has one or more parent terms as well as one or more child terms, the tuples that are related to that particular term are deleted and for each deleted tuples a flag can be set which triggers a new tuple to be inserted. These inserted tuples describe the new relation that holds between the child and parent of the filtered term.

In this research, combinations of pairs of ontologies are being considered. A second possibility for expansion of this work is developing methods for combining more than two ontologies. For example, combining terms from a vegetation ontology with a weather ontology and also a terrain ontology, will result in a framework that can be used by a forest ranger to assess the potential for forest fires. Considering three ontologies, *A*, *B*, and *C*, the implementation of this expansion could involve taking the first term of ontology *A* combining it with first terms of ontologies *B* and *C*, then taking the first term of ontology *A* and *B* again but combining it with the second term of ontology *C*. This process could be repeated until all terms from ontology *C* is exhausted. The steps need to be repeated until all possible combinations of terms from ontologies *A*, *B* and *C* are obtained.

As an alternative to the method of combining ontologies using relational database approach as described in this thesis, another method for combining information or data sources that could be explored in the future exploits multiple inheritance (Frank, 1988). In this case, a new class is created that inherits attributes from two parent classes. For example, Road_DayTime could be a new class that inherits properties of classes Road and DayTime, and represents a combination of terms from the pair of base ontologies.

As seen from the work done in this research, visualization of cross products as well as the result of SQL queries on a cross product, and filtered cross products are very intuitive and help users understand parent-child relationships that exist among the geospatial-temporal terms in the cross products as well as discover structure and find causal relationships. Extending work beyond the static Graphviz visualizations of the cross product could be another possibility for future work. A dynamic visualization of the cross product, that is, on-the-fly visualization while tuples are removed from the cross product, is an exciting prospect. Better understanding of the structure can be provided by selecting a tuple in the cross product and highlighting the corresponding geospatial-temporal terms in the graph view of the cross product.

The use of cross products to combine terms from a pair of ontologies, as described in this thesis, gives us a complete and exhaustive set of combinations. Having every possible combination of terms ascertains that no combinations are overlooked and provides a useful structure for performing analysis.

BIBLIOGRAPHY

- Agarwal, P. (2005). Ontological Considerations in GIScience. *International Journal of Geographical Information Science*, 19(5): 501-536.
- Bouquet, P., M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Krötzsch, L. Serafini, G. Stamou, Y. Sure, S. Tessaris (2004). Specification of a Common Framework for Characterizing Alignment. *KnowledgeWeb Deliverable D2.2.1*.
- Brachman, R. (1983). What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer*, 16(10):30-36.
- Brujin, J., M. Ehrig, C. Feier, F. Martin-Recuerda, F. Scharffe and M. Weiten (2006). Ontology Mediation, Merging and Aligning. *Semantic Web Technologies*, Wiley, UK.
- Ceusters W. (2006). Towards A Realism-Based Metric for Quality Assurance in Ontology Matching. in: B. Bennett, C. Fellbaum (Eds.) *Formal Ontology in Information Systems*, IOS Press, Amsterdam, pp. 321-332.
- Corbett, D. (2003). Comparing and Merging Ontologies: A Concept Type Hierarchy Approach. *Proceedings of the 14th International Symposium on Methodologies for Intelligent Systems*, Maebashi City, Japan, pp. 75-82.
- Corbett, D. (2004). Interoperability of Ontologies Using Conceptual Graph Theory. *Proceedings of Conceptual Structures at Work: 12th International Conference on Conceptual Structures, ICCS, 2004*, Huntsville, AL, pp. 375-387.
- Couclelis, H. (1992). People Manipulate Objects (but Cultivate Fields): Beyond the Raster-Vector Debate in GIS. in: A. U. Frank, I. Campari, and U. Formentini, (Eds.). *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*. LNCS, Vol. 639, Springer-Verlag, New York, pp. 65-77.
- Cruz, I., W. Sunna and A. Chaudhry (2004). Ontology Alignment for Real-World Applications. *Proceedings of The National Conference on Digital Government Research*, Seattle, WA, pp. 393-394.

- Duckham, M. and M. Worboys (2007). Automated geographic information fusion and ontology alignment. in: A. Belussi, B. Catania, E. Clementini, and E. Ferrari (Eds). *Spatial data on the Web: Modelling and Management*, Chapter 6, pp. 109-132.
- Egenhofer, M. J. (1993). A Model for Detailed Binary Topological Relationships. *Geomatica (Ottawa, ON), Canadian Institute of Geomatics*, 47(3&4):261-273.
- Ehrig, M. and Y. Sure (2004). Ontology Mapping – An Integrated Approach. *Proceedings of the First European Semantic Web Symposium, ESWS 2004*, Heraklion, Crete, Greece, pp. 76-91.
- Fonseca, F. and M. Egenhofer (1999). Ontology-Driven Geographic Information Systems. in: C. B. Medeiros (Ed.). *7th ACM Symposium on Advances in Geographic Information Systems*, Kansas City, MO, pp. 14-19.
- Fonseca, F., Egenhofer, M., Agouris, P., and Camara, G. (2002). Using Ontologies for Integrated Geographic Information Systems. *Transactions in GIS*, 6(3):231-257.
- Fonseca F., G. Camara, and A. Monteriro (2006). A Framework for Measuring the Interoperability of Geo-ontologies. *Spatial cognition and computation*, 6(4): 307-329.
- Frank, A. U. (1988). Multiple Inheritance and Genericity for the Integration of a Database Management System in an Object-Oriented Approach. *Lecture Notes in Computer Science*. Vol 334: 268-273.
- Genesereth, M. R., and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. San Mateo, CA, Morgan Kaufmann Publishers.
- Goodchild, M. (1992). Geographical Data Modeling. *Computers and Geosciences* 18(4):401-408.
- Grenon, P. and B. Smith (2004). SNAP and SPAN: Towards Dynamic Spatial Ontology. *Spatial Cognition and Computation*, 4(1): 69-104.

- Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199-220.
- Hamilton, H. J., R. J. Hilderman, and N. Cerone (1996). Attribute-Oriented Induction Using Domain Generalization Graphs. *Proceedings of the Eighth IEEE International Conference on Tools with Artificial Intelligence (ICTAI'96)*, Toulouse, France, pp. 246-253.
- Han, J., Y. Cai and N. Cerone (1992). Knowledge Discovery in Databases: An Attribute Oriented Approach. in: L. Yuan, editor. *Proceedings of the 18th International Conference on Very Large Databases*, Vancouver, Canada, pp. 547-559.
- Han, J. (1994). Towards Efficient Induction Mechanism in Database Systems. *Theoretical Computer Science*, 133(2): 361-385.
- Harvey, F., W. Kuhn, H. Pundt, Y. Bishr and C. Riedemann (1999). Semantic Interoperability: A Central Issue for Sharing Geographic Information. *Annals of Regional Science*, 33(2): 213-232.
- Hilderman, R. J., L. Li and H. J. Hamilton (1997a). Data Visualization in the DB-Discover System. *Proceedings of 9th International Conference on Tools with Artificial Intelligence*, Washington DC, pp. 474-477.
- Hilderman, R.J., H. J. Hamilton, R. J. Kowalchuk and N. Cerone (1997b). Parallel Knowledge Discovery Using Domain Generalization Graphs. in: J. Komorowski, and J. Zytchow (Eds.). *Proceedings of Principles of Data Mining and Knowledge Discovery*, Paris, France, pp. 25-35.
- Hill, D. P., J. A. Blake, J. E. Richardson and M. Ringwald (2002). Extension and Integration of the Gene Ontology (GO): Combining GO Vocabularies with External Vocabularies. *Genome Research*, 12(12):1982-1991.
- Hughes, T. C. and B. C. Ashpole (2004). The Semantics of Ontology Alignment. *Proceedings of Performance Metrics for Intelligent Systems*, Gaithersburg, MD.
- Kalfoglou, Y. and M. Schorlemmer (2003). Ontology Mapping: The State of the Art. *Knowledge Engineering Review*, 18(1):1-31.

- Klein, M. (2001). Combining and Relating Ontologies: An Analysis of Problems and Solutions. in: A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, and A. Uschold (Eds.). *Workshop on Ontologies and Information Sharing, International Joint Conferences on Artificial Intelligence*, Seattle, WA, pp. 53-62.
- Kohler, J., M. Lange, R. Hofstadt, S. Schulze-Kremer (2000). Logical and Semantic Database Integration. *IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, Arlington, VA, pp. 77-80.
- Neuhaus, F. and B. Smith (2007). Modeling Principles and Methodologies – Relations in Anatomical Ontologies. in: A. Burger, D. Davidson and R. Baldock (Eds.). *Anatomy Ontologies for Bioinformatics: Principles and Practice*, Springer Verlag (in press).
- Niles, I. and A. Pease (2003). Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. *Proceedings of the 2003 International Conference on Information and Knowledge Engineering*, Las Vegas, NV, pp. 412-416.
- Pang, W., R. J. Hilderman, H. J. Hamilton and S. D. Goodwin (1996). Data Mining with Concept Generalization Digraphs. in: FLAIRS-96, Key West, FL.
- Pease, A. (2004). Standard Upper Ontology Knowledge Interchange Format. Language manual, <http://ontolog.cim3.net/file/resource/reference/SIGMA-kee/suo-kif.pdf>.
- Riedemann, C., and W. Kuhn (1999). What are Sports Grounds? Or: Why Semantics Requires Interoperability. in: A. Vckovski, K. E. Brassel, and H. Schek (Eds.). *International Conference on Interoperating Geographic Information Systems*, Zurich, Switzerland, Lecture Notes in Computer Science, Vol. 1580, pp. 217-229.
- Rodríguez, A. and M. Egenhofer (2003). Determining Semantic Similarity Among Entity Classes from Different Ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15 (2): 442-456.
- Smith, B. and D. Mark (1998). Ontology and Geographic Kinds. in: *International Symposium on Spatial Data Handling*, Vancouver, Canada, pp. 308-320.

- Sowa, J. F. (2000). Knowledge Representation: Logical, Philosophical, and Computational Foundations, <http://www.jfsowa.com/ontology/index.htm>.
- Stolte, C., D. Tang, and P. Hanrahan (2002). Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics*, 10(1): 52-65.
- Stolte, C. (2003). Query, Analysis, and Visualization of Multidimensional Databases. *Ph.D. Dissertation*, Department of Computer Science, Stanford University, http://graphics.stanford.edu/papers/cstolte_thesis/thesis.pdf.
- Winston, M. E., R. Chaffin, and D. J. Herrmann (1987). A Taxonomy of Part-Whole Relations. *Cognitive Science*, 11(4):417-444.
- Zhou, N. (2003). A Study on Automatic Ontology Mapping of Categorical Information. *Proceedings of the 2003 Annual National Conference on Digital Government Research*, Boston, MA, pp. 1-4.

BIOGRAPHY OF THE AUTHOR

Kripa Joshi was born in Lalitpur, Nepal on April 06, 1981. She did her undergraduate degree (Bachelor of Engineering) in Computer Engineering at the Institute of Engineering, Tribhuvan University, Kathmandu, Nepal. After completing her B.E., she worked for a year as a Software Engineer at Yomari Inc., Pvt. Ltd., Nepal. She started her Masters of Science program in Spatial Information Science and Engineering at the Department of Spatial Information Science and Engineering, The University of Maine, Orono in January 2006. Kripa is a candidate for the Master of Science degree in Spatial Information Science and Engineering from The University of Maine in December, 2007.