The University of Maine
DigitalCommons@UMaine

Electronic Theses and Dissertations | Fogler Library

12-2002

Modeling Multiple Granularities of Spatial Objects

Chitra Ramalingam

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

 Part of the Databases and Information Systems Commons, and the Theory and Algorithms Commons

Recommended Citation
Ramalingam, Chitra, "Modeling Multiple Granularities of Spatial Objects" (2002). *Electronic Theses and Dissertations*. 584.
http://digitalcommons.library.umaine.edu/etd/584

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.

MODELING MULTIPLE GRANULARITIES OF SPATIAL OBJECTS

By

Chitra Ramalingam

B.E. Datta Meghe College of Engineering, Bombay - India, 1998

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, 2002

Advisory Committee:

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

Kathleen Hornsby, Assistant Research Professor, National Center for Geographic Information and Analysis, Co-Advisor

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

Robert D. Franzosa, Professor of Mathematics

MODELING MULTIPLE GRANULARITIES OF SPATIAL OBJECTS

By Chitra Ramalingam

Thesis Co-Advisors: Dr. Max J. Egenhofer
Dr. Kathleen 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, 2002

People conceptualize objects in an information space over different levels of details or *granularities* and shift among these granularities as necessary for the task at hand. Shifting among granularities is fundamental for understanding and reasoning about an information space. In general, shifting to a coarser granularity can improve one's understanding of a complex information space, whereas shifting to a more detailed granularity reveals information that is otherwise unknown. To arrive at a coarser granularity, objects must be generalized. There are multiple ways to perform generalization. Several generalization methods have been adopted from the abstraction processes that are intuitively carried out by people. Although, people seem to be able to carry out abstractions and generalize objects with ease, formalizing these generalization and shifts between them in an information system, such as geographic information system, still offers many challenges. A set of rules capturing multiple granularities of

objects and the use of these granularities for enhanced reasoning and browsing is yet to be well researched.

This thesis pursues an approach for arriving at multiple granularities of spatial objects based on the concept of coarsening. Coarsening refers to the process of transforming a representation of objects into a less detailed representation. The focus of this thesis is to develop a set of *coarsening operators* that are based on the objects' attributes, attribute values and relations with other objects, such as containment, connectivity, and nearness, for arriving at coarser or amalgamated objects. As a result, a set of four coarsening operators- *group*, *compose*, *coexist*, and *filter* are defined.

A framework, called a *granularity graph*, is presented for modeling the application of coarsening operators iteratively to form amalgamated objects. A granularity graph can be used to browse through objects at different granularities, to retrieve objects that are at different granularities, and to examine how the granularities are related to each other. There can occur long sequences of operators between objects in the graph, which need to be simplified. Compositions of coarsening operators are derived to collapse or simplify the chain of operators. The semantics associated with objects amalgamations enable to determine correct results of the compositions of coarsening operators. The composition of operators enables to determine all the possible ways for arriving at a coarser granularity of objects from a set of objects. Capturing these different ways facilitates enhanced reasoning of how objects at multiple granularities are related to each other.

ACKNOWLEDGMENTS

This thesis would have been impossible without the support and guidance of many people. I take this opportunity to extend my sincere gratitude to all.

First, I gratefully acknowledge my advisors, Dr. Max Egenhofer and Dr. Kathleen Hornsby, whose enthusiasm and guidance was crucial for the progress of my thesis. Their generous cooperation and encouragement helped me accomplish the successful completion of this thesis.

I would like to specially thank Dr. Bob Franzosa for his willingness to discussions, which helped me in clarifying many doubts. I thank him and my other committee member, Dr. Kate Beard, for their support and guidance.

Thanks to all my colleagues in the SIE department, especially Ramaswamy Hariharan, Jim Farrugia, and Hariharan Gowrishankar for their many stimulating discussions. This research would not have been possible without the huge personal and academic support from Ram and I thank him for sharing every moment of this thesis with me.

My sincere appreciation and thanks to Dr. Alfred Leick for allowing me to work in his lab during my prototype implementation.

I would also like to thank my professors in India, Dr. Parvatham Venkatachalam and Dr. Krishna Mohan, whose encouragement and guidance enabled me to pursue the graduate study at SIE.

Thanks to my all friends at the University who made my study here a memorable one. This list is long, but I would like to mention Maya Panangadan, Isolde Schlaisich, Tam Thanh Huynh, Sirisha Pochareddy, Anuket Bhaduri, and Sharad Chakravarthy.

Funding for this thesis from National Imagery and Mapping Agency under grant number NMA201-00-1-2009 is specially acknowledged.

Finally, I thank all my family members, my parents, sisters, and bother-in-law for their love and support through all the years and encouraging me in all my endeavors.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ii

LIST OF TABLES viii

LIST OF FIGURES x

Chapters

1. INTRODUCTION 1

    1.1 Background of Thesis 4

    1.2 Motivation for Research 7

    1.3 Key Research Questions 9

    1.4 Goal and Hypothesis 10

    1.5 Scope of Thesis 11

    1.6 Major Results 13

    1.7 Organization of the Thesis 14

2. MODELING MULTIPLE GRANULARITIES 16

    2.1 Methods for Modeling Object Granularities 18

        2.1.1 Concept hierarchies 18

        2.1.2 Domain generalization graphs 20

        2.1.3 Ontologies 21

    2.2 Object-Oriented Approaches for Changing Granularity 22

        2.2.1 Object-oriented abstraction methods 22

        2.2.2 Object-orientation in GIS 25

    2.3 Map-Based Approaches for Changing Granularity 27

2.4 Visualization-Based Approaches 30

2.5 Summary 32

3. DERIVING OBJECT GRANULARITIES THROUGH COARSENING 33

3.1 Coarsening 34

3.2 Filter 37

3.2.1 Formalization of the operator 38

3.2.2 Example 38

3.3 Group 38

3.3.1 Formalization of the operator 40

3.3.2 Example 40

3.4 Compose 41

3.4.1 Formalization of the operator 44

3.4.2 Example 45

3.5 Coexist 45

3.5.1 Formalization of the operator 46

3.5.2 Example 46

3.6 Summary 47

4. GRANULARITY GRAPHS 49

4.1 Rationale for a Granularity Graph 49

4.2 Elements of a Granularity Graph 50

4.3 Constructing a Granularity Graph 52

4.3.1 Creating object granularities 54

4.3.2 Matching object granularities 55

4.3.3 Case study: Acadia Park 57
4.4 Browsing a Granularity Graph 62
4.4.1 Unary operations 63
4.4.2 Binary operations 64
4.4.3 Mapping of graph operations onto object granularities 65
4.5 Summary 70
5. COMPOSITION OF COARSENING OPERATORS 71
5.1 Definitions for Composing Coarsening Operators 73
5.1.1 Compositions with filter 75
5.1.2 Compositions with group and coexist 76
5.1.3 Compositions with compose 78
5.2 Inferences from Compositions 85
5.3 Application of Compositions 89
5.4 Summary 95
6. PROTOTYPE FOR CONSTRUCTING AND BROWSING A GRANULARITY GRAPH 97
6.1 Prototype Design and Specification 97
6.1.1 Objects 98
6.1.2 Coarsening operators 100
6.1.3 Granularity graph 101
6.2 The User Interface 102
6.2.1 Creating a new granularity graph 103
6.2.2 Applying coarsening operators 103

6.2.3 Browsing object granularities 104

6.3 Illustration of the Prototype 104

6.4 Summary 110

7. CONCLUSIONS AND FUTURE WORK 112

7.1 Summary 113

7.2 Conclusions 114

7.3 Future Work 116

BIBLIOGRAPHY 118

BIOGRAPHY OF THE AUTHOR 125

LIST OF TABLES

Table 3.1 Common attributes of objects: length and speed limit.....41

Table 3.2 Coarsening operators and the corresponding instance-class pairs to which the operators can be applied.....48

Table 4.1 Information space of the Acadia National Park modeling the different *Trails*.....54

Table 4.2 Similar attribute values of objects for the attribute *type_of_trail*: ♦ denotes *strenuous* trails, while ⊔ denotes *easy trails*.....55

Table 4.3 Relations among objects *Mount_desert_island*, *Bar_harbor*, and *Acadia_park*.....56

Table 4.4 Information space for Acadia National Park.....58

Table 4.5 Axioms partly describing the granularity graph for Acadia National Park.....62

Table 4.6 Unary browsing operations on graph.....64

Table 4.7 Binary browsing operations on the graph.....65

Table 5.1 Coarsening operators and their corresponding instance-class pairs.....74

Table 5.2 Compositions of the *compose* operators with each other. ~ represents undetermined compositions.....81

Table 5.3 Compositions of coarsening operators over classes. ~ signifies undetermined compositions.....86

Table 5.4 Compositions of coarsening operators over instances. ~ signifies undetermined compositions.....86

Table 5.5 Percentage of valid compositions over instances and classes.....87

Table 5.6 Compositions of the detailed *compose* operators with each other. ~ represents undetermined compositions. 88

Table 5.7 Applying compositions of operators to arrive at a shorter path. 90

Table 5.8 Multiple ways to arrive at *Acadia Park* from *Beehive Trails*. 92

Table 5.9 Simplification of a sequence of operators using the associative property of compositions. 92

Table 5.10 Simplifying sequence of operators for relating object granularities. 93

Table 5.11 Two different paths connecting *Sand Beach* to *Tourist Attractions* yields analogous simplifications. 95

LIST OF FIGURES

Figure 1.1 (a) Different datasets for the same geographic space. (b) Overlaying different datasets in a GIS leads to a refined granularity. 4

Figure 1.2 Map-based generalization: (a) snapshot view of map objects, generalization of data by (b) selection process and (c) aggregation. 5

Figure 2.1 Concept hierarchy for the *street* attribute. 20

Figure 2.2 Single and multi-path DGG for the *street* attribute. 21

Figure 2.3 Describing (a) city as a class and (b) instances of city, Bangor and Portland. 23

Figure 2.4 (a) Original map and (b) generalized map using aggregation operation and (c) amalgamation operation. 30

Figure 3.1 (a) Filtering objects x_3 and x_4 from the set $U = \{x_1, x_2, \ldots x_n\}$ and (b) amalgamation of objects x_1, x_2, and x_3 into y_1 34

Figure 3.2 (a) Filtering objects x_3 and x_4 from $U = \{x_2, x_3, x_4, x_5\}$, (b) trivial *filter* operation selecting all the objects in U, and (c) trivial *filter* operation on x_5 and amalgamation of x_2, x_3, and x_4 into y_3. 37

Figure 3.3 Grouping *Lake* and *Pond* into *Water-body* based on their common attributes, depth, volume, and type. 39

Figure 4.1 A granularity graph, $U = \{x_1, x_2, \ldots, x_{10}\}$. 51

Figure 4.2 Objects exhibiting a spatial containment relation. 56

Figure 4.3 Matching object granularity: A *compose* operator is used to connect the existing objects granularities *Acadia_park*, *Bar_harbor* and *Mount_desert_island* 57

Figure 4.4 A granularity graph of the Acadia Park consisting of two levels. 59

Figure 4.5 Filtering objects *Ponds*, *Campground*, and *Sand_beach* from the set *U* to result in a set *S* = { *Sand_beach*, *Campground*, *Trails*, *Ponds*}. 60

Figure 4.6 Granularity graph for Acadia National Park. 61

Figure 4.7 Result of the browse operation, *getDescendant(X, Park_loop_road)*. 67

Figure 4.8 Granularities retrieved by applying the binary browse operation *getAncestor* to *Cadillac_mountain*. 68

Figure 4.9 Browse operation *getAncestorDescendant(X, Trails, Acadia Park)* yields *Park_loop_road*, which is both an ancestor to *Trails* and descendant to *Acadia Park*.....69

Figure 5.1 Composition of operators *R* and *S* yield $T = R \otimes S$ from *A* to *C*. 72

Figure 5.2 Granularity graph for Acadia National Park. 73

Figure 5.3 Composition of *coexist* with *filter* yields a *coexist* operator. 76

Figure 5.4 Amalgamation of objects applying a *coexist* followed by a *group*: ■ - common attribute values and || - common attributes of objects. 77

Figure 5.5 Composition of *compose[Contained]* with *compose[Near]* yields an undetermined composition. 79

Figure 5.6 Composition of *compose[Contained]* with a *group* over instances yields a *compose[Contained]*. 82

Figure 5.7 Composition *compose[Contained]* ⊗ *group* over classes yields an undetermined result.....83

Figure 5.8 Simplifying the path from *Beehive Trails* to *Acadia Park*. 90

Figure 5.9 Granularity Graph with a selected path. 91

Figure 5.10 Simplified granularity graph applying compositions.....91

Figure 5.11 Deriving a path from *Beehive Trail* to *Acadia Park* with a *group* operator. 93

Figure 6.1 The prototype architecture: user-interface and graph builder.....98

Figure 6.2 Structure of an object, *GObject*. 99

Figure 6.3 Class *GraphBuilder*. 100

Figure 6.4 Classes for implementing the graph structure.....101

Figure 6.5 User interface of the prototype. 102

Figure 6.6 Selecting three *trails* for amalgamation from the set of objects. 105

Figure 6.7 Creating an amalgamated object *Easy_Trails* for the selected objects by applying the *group* operator.....106

Figure 6.8 Step-wise building of a granularity graph. (a) Creating an amalgamated object *Easy Trails* by the *group* operator and (b) adding amalgamated object *Forest Trails* by applying the *compose* operator. 107

Figure 6.9 Granularity graph for the Acadia Trails.....108

Figure 6.10 Result of the browse operation *getChlidOp* on *Trail_Routes* based on the *group* operator.....109

Figure 6.11 Result of the browse operation *getDescendants* on *Hike Trail*.....110

Chapter 1

INTRODUCTION

Spatio-temporal knowledge representation often requires changing from one level of detail to another so that users can carry out a desired task (Buttenfield and Delotto 1989; Guptill 1990; Hornsby and Egenhofer 1999). Location-based querying, monitoring of hazard zones, and planning a transport system, for example, need to be examined over different levels of details. Geographic information systems (GISs) typically support changes in the level of detail from the perspective of changes to geometric properties of map objects. Changing the map scale or emphasizing essential map objects while suppressing the unimportant are some functions that render different levels of detail. Data can be represented at different levels of detail, each suited for a particular purpose. In a GIS, for example, it is required to display data at certain levels of detail while guiding a person through a maze of streets or to assist military personnel across an area of landmines. In this thesis, we refer to the level of detail as *granularity* (Hornsby and Egenhofer 2002). Incorporating multiple granularities of spatio-temporal data and enabling translations among different granularities have been identified as important in information systems (Buttenfield and Delotto 1989). In order to deal successfully with several emerging spatio-temporal applications, such as multiple representations of spatio-

temporal data, GISs must support methods for modeling data at different granularities and enable shifts among them.

The most common approach to simplifying levels of detail of information is by the process of selection and generalization. In everyday life, the amount of information people encounter is vast and much more detailed than they can cognize. To deal with the complexities, people typically consider only things that are relevant to their tasks and abstract away the unnecessary details (Hobbs 1990; Bederson and Hollan 1994; Timpf and Frank 1998). Several methods for generalizing map objects have been adopted based on abstraction processes that are intuitively carried out by people. Abstraction factors out the commonalities in the description of several concepts in an information space into the description of a more general concept (Timpf 1999). For example, the different kinds of buildings on a university campus, such as academic buildings, administrative buildings, dormitories, gymnasium, and arts centers, can be abstracted to a simpler and more general concept of campus buildings. It is typically sufficient to refer to the various types of buildings as campus buildings when describing the university campus to a friend, whereas more specific details are required when directing a student to a particular building. In this process of understanding and reasoning about an information space, people intuitively perform shifts among the different granularities and draw on an information space that is at a required level of detail, according to their task.

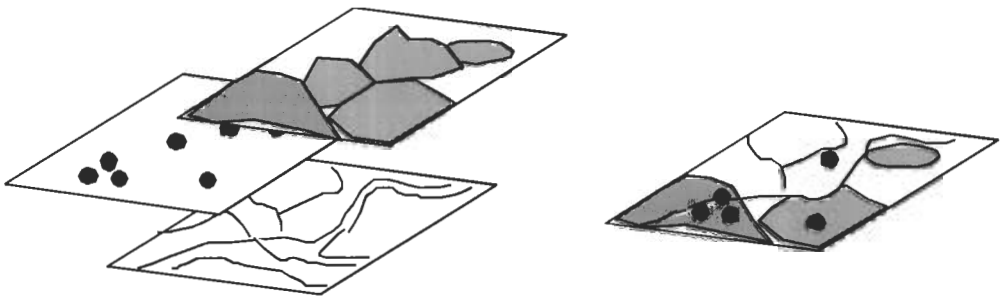
Shifting among granularities is fundamental for reasoning (Hobbs 1990) and key for any knowledge-based system. In general, shifting to a coarser granularity of entities can improve one's understanding of a complex information space. Conversely, shifting to a more detailed granularity can uncover information that otherwise is unknown (Hornsby

and Egenhofer 1999). It is, therefore, of primary importance to support multiple granularities and shifts among these granularities.

Although people carry out abstractions or shifts to more detailed granularities with ease, formalizing these shifts for integration into an information system and query languages still offers many challenges. Considerable amount of work has been done by the cartographic and GIS community to generate map-based generalizations, that is, translations of maps to higher or less precise scales (Buttenfield and McMaster 1991; Muller *et al.* 1995a). Multi-resolution map models have been proposed to offer increased capabilities for spatial reasoning and representation (Puppo and Dettori 1995; Stell and Worboys 1998). Techniques have also been developed to change granularities for enhanced visualization (Furnas 1986; Tanaka and Ichikawa 1988; Stone *et al.* 1994; Timpf and Frank 1998). To date, automated map generalization and visualization-based generalization have been based on a set of complex geometric and attribute manipulations. Object-oriented abstraction methods are other most commonly used approaches for arriving at generalizations (Smith and Smith 1977; Brodie *et al.* 1984) using the principles of inheritance and aggregation. However, deriving granularities is a subjective process and there can be multiple ways of arriving at a granularity. A set of rules capturing the complete set of possible granularities is yet to be developed. An approach for building the different granularities and enabling shifts among them is required. Other challenges are to find relations among entities at different granularities and to use them for enhanced reasoning based on multiple granularities.

1.1 Background of Thesis

A simple approach for modeling data at different granularities in a GIS is to store different datasets as layers for the same geographic space (Figure 1.1a). Each layer contains data specific for a task, such as town data, city data, and road network data, which can be integrated spatially (Figure 1.1b) resulting in a detailed granularity. Selecting and omitting the layers in the dataset allows shifting to more or fewer spatial details. In this approach, a fixed set of granularities is used, whereas it is often required to be able to derive different granularities according to the user's task at hand.

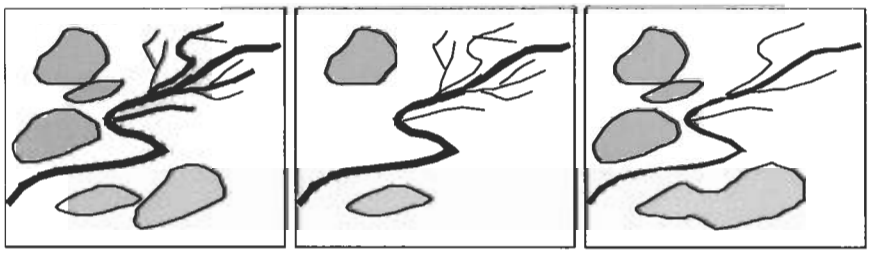


(a) (b)

Figure 1.1 (a) Different datasets for the same geographic space. (b) Overlaying different datasets in a GIS leads to a refined granularity.

Another approach to arrive at coarser granularities involves the selective omission of map features and the generalization of map features into more abstract forms (McMaster and Shea 1992). Map-based generalization or cartographic generalization has been a major subject of study, particularly from a geometric perspective. Cartographic generalization involves a reduction in the map content dependent on scale changes to maps and attribute data manipulation, transforming a detailed representation of the map (Figure 1.2a) into a less detailed representation (Buttenfield and McMaster 1991; Muller

et al. 1995a). An initial step in the generalization process involves the selection of map features, relevant to the task. Selection results in a less detailed granularity consisting of only the relevant map features (Figure 1.2b). Following the selection, such operations as smoothing, simplification, and aggregation (Figure 1.2c) are applied to map features based on geometric and attribute transformations to result in a new, generalized map representation (McMaster and Shea 1992). Similarly, automated map generalization, which employs agents to attain an acceptable level of detail (Lamy *et al.* 1999) and generalization of coverages using thematic information (Frank *et al.* 1997) are other methods to obtain map-based generalizations. Since map-based generalizations frequently address geometric manipulations, the computations can be complex and highly dependent on map features.



(a) (b) (c)

Figure 1.2 Map-based generalization: (a) snapshot view of map objects, generalization of data by (b) selection process and (c) aggregation.

To avoid the geometric complexities in map-based generalizations several techniques have been proposed to render enhanced visualization by manipulating the semantic properties of map elements. *Fisheye views* (Furnas 1986) provide a visualization of maps based on the concept of a panorama by implementing the local details in context with the global structure. Fisheye views are obtained by varying the semantic importance of map

elements. *Pad++* (Bederson and Hollan 1994) is a graphic zooming technique that renders generalizations based on semantic task-based filtering. A semantic panning and zooming technique from the perspective of filtering objects supports a step-by-step refinement of map elements through visual feedback from the user (Tanaka and Ichikawa 1988). The Per*spective Wall* (Mackinlay *et al.* 1991) and the *Magic Filter* (Stone *et al.* 1994) are other visualization-based generalizations, which also use filtering techniques in the generalization process. The visualization-based methods, however, do not incorporate any techniques to combine or amalgamate entities into more general concepts.

Several multi-resolution map models have been proposed to implement these filtering and abstraction mechanisms in order to offer increased support in spatial reasoning. Multi-resolution models facilitate storing the multi-granular representation of maps by using a hierarchical tree structure (Frank and Timpf 1994; Puppo and Dettori 1995). Such models provide flexibility to represent data with regard to scale and resolution. Stell and Worboys (1998) illustrate an approach to model multi-resolution spatial data by using granularity lattices and map spaces. They provide methods to shift between map spaces and a formal discussion for integrating semantically and geometrically heterogeneous spatial datasets.

For multi-resolution models to be effective, it was necessary to have a means to make appropriate transitions between different granularities of data (Hornsby and Egenhofer 1999; Stell and Worboys 1999). Further work on multi-resolution models has led to an investigation of the types of operations necessary to arrive at different granularities. Timpf (1999) provided a categorization of hierarchies for the abstraction process based on the generalization operations of filter, aggregation, and classification. Stell and

Worboys (1999) introduce two kinds of generalization operations, selection and amalgamation, enabling transitions to granularities with less detail. A few generalization operations have been identified, however, the operations do not model the different semantics associated with combining objects to result in coarser granularities. Also, there is a need for uniformity in modeling the operators. The definition of operations lacks a common set of criteria for comparison, which limits the use of the operators based on the multiple granularities. For example, application of the operators for determining the relation between the objects at different granularities or the different ways (i.e., shortest path or desired path) of arriving at an object granularity is not investigated. The process of abstracting to coarser granularities and the use of such models for enhanced reasoning and browsing based on multiple granularities is yet to be well researched.

1.2 Motivation for Research

We envision that objects are rich in structure and semantics. Exploiting an object's semantic attributes and relations with other objects can lead to a large number of possible ways to generalize data. Treating the geometry, scale, and attributes of the map features has resulted in a set of cartographic generalization operators (McMaster and Shea 1992), such as simplification, aggregation, smoothing, amalgamation, merging, collapse, refinement, exaggeration, enhancement, and displacement. Ormsby and Mackaness (1999) also discuss map generalizations based on the different phenomena of map features determined by geometry, semantic, and inter-object relationships. GIS, however, requires methods for addressing generalizations and granularity change operations beyond map-based generalization. Timpf (1999) and Stell and Worboys (1999) propose semantic generalization operations for multi-resolution models by considering spatial

data as objects. The semantic and task-based generalizations of map features have resulted in methods alternative to the complex geometrical computations for generalizing objects. Modeling multiple granularities, however, involve other challenges, such as determining the semantics associated with different granularities, enabling shifts among the different granularities for retrieving coarser and finer object granularities, and finding the relation between the different granularities.

This thesis develops an approach to model multiple granularities of objects based on the concept of *coarsening*. Coarsening refers to the process of transforming a representation of objects into a less detailed representation. Coarsening of objects is achieved by *filter* and *amalgamation*. Filter is the process of selecting a subset of objects from a set while omitting the other objects. The selected objects are considered as part of a coarser granularity. Amalgamation is the process of combining two or more objects to result in a single object at a coarser granularity (Stell and Worboys 1999). There are different semantics associated with object amalgamations that lead to a coarsening and in this thesis we identify the different kinds of object amalgamations.

An *object* is the representation of a physical entity, such as a building or a lake, or a fiat entity (Smith and Varzi 1997), such as a land-parcel or a university, in an information system. Objects are also distinguished as *classes* or as an *instance* of a class. For example, object *Building* is a class with attributes that correspond to buildings and object *Boardman Hall* is an instance of the class *Building*. Each object is defined by a (1) set of attributes, (2) attribute values, and (3) relations with respect to other objects. In this thesis, we address spatial relationships among the objects, namely *contained*, *connected*, and *near*. The different kinds of object amalgamations are defined based on the three

components of objects. As a result, we define four coarsening operators - *group*, *compose*, *coexist* and *filter* for combining and selecting objects to result in coarser granularities or amalgamated objects. Applying the coarsening operators iteratively to objects and their amalgamated objects leads to the creation of a framework consisting of multiple granularities of objects, called a *granularity graph*.

Granularity graphs represent a rich structure of objects at multiple granularities, connected by coarsening operators. The graphs provide a framework for shifting among different object granularities. It also supports browsing through granularities, enabling retrieval of objects at different granularities. A granularity graph can contain long sequences of operators connecting two object granularities and often it would be required to simplify the sequence of operators. For example, to determine a shorter path in the graph or how two granularities are connected to each other. Compositions of operators play a significant role in collapsing or simplifying the sequences of coarsening operators. In this thesis, we derive all possible compositions of the coarsening operators and determine valid compositions that can be used in simplifying the sequence of operators. The different applications of the composition of operators are also presented highlighting their use in enhanced reasoning based on the multiple granularities.

1.3 Key Research Questions

Systems that model multiple granularities need methods supporting translations among the different granularities and for arriving at a required granularity. When translating between two different granularities several questions arise with regard to the objects that are at finer or coarser granularities. To process such queries one needs a framework and

rules for modeling all object combinations that lead to coarse granularities. The development of such a framework is guided by the following research questions:

- Can two objects be amalgamated to result in a coarser granularity?
- What are the different ways in which an object can be combined with other objects?
- What are the semantics associated with the retrieval of coarser granularities of a set of objects?

A framework of multiple granularities consists of objects at different levels of details connected by coarsening operators. There can occur long sequences of operators connecting two objects in the framework that may require to be simplified. Composition of coarsening operators can be used to collapse or simplify the sequence of operators. Based on the valid compositions of the coarsening operators, we can answer challenging research questions, such as:

- Is it possible to derive valid compositions for all operators?
- Applying the compositions, can the number of operators be simplified or collapsed to a simpler sequence?
- Which of the operators result in most effective compositions or least effective compositions?

1.4 Goal and Hypothesis

The goal of this thesis is to model multiple granularities of objects and enabling shifts among them for retrieving finer or coarser granularities. The main focus of this approach is to distinguish the different semantics associated with combining objects for arriving at

multiple coarser granularities. A set of coarsening operators is developed to result in a framework of multiple granularities. Each operator models a distinct object amalgamation or selection and is defined based on the valid instances and classes of objects to which it can be applied. Another contribution of this work is the derivation of all possible compositions of the coarsening operators and their different useful applications based on multiple granularities.

Compositions of coarsening operators play an important role in simplifying the sequence of operators between two objects in a granularity graph. The compositions are used for determining a shorter path in the graph and the relations among the objects at different granularities in the graph. Composition of operators is significant for reasoning based on multiple granularities and enables one to obtain more complete semantics of objects at multiple granularities. Understanding the semantics of coarsening operators that connect the different granularities is important for determining valid compositions of operators. Hence, in this thesis we focus on the semantics of object amalgamations and their effect in deriving valid compositions of operators. The hypothesis of this thesis is:

> *Different semantics associated with the object amalgamations are needed to yield correct results of the compositions of coarsening operators.*

1.5 Scope of Thesis

This thesis builds on a framework of coarsening operators, consisting of filtering and amalgamations, to model multiple granularities of objects. We identify four ways in which objects can be combined or selected into a coarser granularity based on common attributes, common attribute values, and similar relations with other objects. The use of

geometry in the amalgamation process or the spatial resolution of objects is not a focus of this work. By common attributes and common attribute values of objects, we imply identical values of the attribute names and the attribute values of objects, respectively. The coarsening operators can also be extended to consider range of values of attributes instead of a particular value, though we do address range of values in this thesis. We consider only three spatial relations among objects, namely contained, connected, and near. Coarsening operators can be extended to include other spatial relations that capture different semantics of object amalgamations.

We develop a model for building multiple granularities of objects, support shifting among the different granularities and enable determining the relation between granularities. An object-oriented approach is used for arriving at granularities. It also includes spatial relations in order to capture rich semantics of object amalgamations. Similar to the other generalization techniques, the approach presented in this thesis enables to form multiple granularities of objects. Comparison between the different approaches is beyond the scope of this thesis.

Applying coarsening operators can result in an object that is a class, such as *Building* or an object that is an instance, such as *College of Engineering*. The operators, by their definitions, determine only a subset of the actual properties of the resulting amalgamated object. The operators do not assist in finding out the spatial nature or topology of the amalgamated object.

The granularity graph derived from using these operators does not necessarily represent all possible coarsenings that can exist among the granularities. Each level in the

granularity graph is derived using a particular sequence of coarsening operators. A different granularity graph may be obtained by changing the sequence of the operations.

We present a complete set of the compositions of coarsening operators based on the definitions on the coarsening operators. The result of a possible composition of operators can be valid, giving a single result, or undetermined, implying that there are more than one possible result of the composition. The undetermined composition requires more information from the user in order to determine the valid result of the composition.

This thesis does not present an exhaustive classification of amalgamations of objects. We restrict our amalgamation based on attributes, attribute values, and three spatial relations among objects. There can be other object amalgamation, such as evolution of objects or merging of objects that can be included in the set of operators by treating additional relations among objects. Our approach provides a conceptual modeling among granularities and does not deal with the multiple geometric representations of spatial objects. We suggest a framework for building multiple objects granularities and enable browsing through them. It does not yet support any query language for multiple granularities.

1.6 Major Results

This approach highlights the differences in the granularity of objects obtained from the different semantics of object combinations. A set of coarsening operators and granularity graphs are the outcome of modeling multiple granularities through coarsening. The granularity graph derived using the operators supports a multi-granular model of the objects. The graph structure enables mapping of the different graph operations for

browsing through the object granularities. The browsing operations enable the retrieval of objects at different granularities, such as objects that are at finer or coarser granularities, or objects that share a common coarser granularity in the graph. Several unary and binary browsing operations are reviewed.

Another major result from the coarsening operators is valid compositions of the coarsening operators. The composition of operators can be used to reduce the sequence of operators connecting two objects in the graph to a minimum. They enable determination of how two objects at different granularities are related to each other. Applying the compositions, it is also possible to determine the different ways of arriving at a granularity from a set of objects. The undetermined compositions prevent arriving at invalid results of compositions and prompt the user to find the valid composition by using more information.

1.7 Organization of the Thesis

The remainder of the thesis is organized into seven chapters.

Chapter 2 discusses the research and the literature that underlie this work: from the need to model granularity and granularity changes to the state-of-art of the different approaches to model multi-granular data.

Chapter 3 introduces the basis for arriving at coarser granularities of objects through filtering and amalgamation. Several different kinds of amalgamations and their semantics are presented. Four distinct coarsening operators are compiled to capture the semantics of the different amalgamations and filtering. Formal rules for implementing the operators are also specified.

Chapter 4 discusses the building of a granularity graph applying the coarsening operators. The element of a graph and a basic algorithm for constructing a granularity graph is presented. The several browsing operations for a graph and its mapping to browse objects at different granularities is discussed.

Chapter 5 is a study on the composition of the coarsening operators. We examine the sequences of operations and identify all the possible compositions of operators. Inferences from the compositions and their affect on the granularities are presented. Several applications of the compositions are also reviewed.

In chapter 6, the implementation of the prototype for modeling multiple granularities is discussed. Following the design and specification of the prototype, an example is presented to demonstrate the construction of a granularity graph and browsing operations on the graph.

Chapter 7 concludes the thesis with a discussion of the major results. We also present the scope for carrying out further research work in this area.

Chapter 2

MODELING MULTIPLE GRANULARITIES

Different levels of details are useful for different tasks. For certain tasks, a coarser level of detail is needed, whereas other tasks may require a more detailed perspective of data. In this thesis, we refer to these levels of detail as granularity. Granularities of entities are fundamental for understanding and reasoning about the world (Hobbs 1990). People consider only certain relevant data according to their interest or tasks being performed and abstract the excessive details (Hornsby and Egenhofer 1999). In addition to simplifying the information space by selecting relevant data, combining or grouping data to coarser granularities enables better understanding of complex information space. Examples such as "I went to the Mall," "Stillwater is a good neighborhood," and "University parking is safe," are expressions that contain commonly used coarser approximations of an information space obtained by combining several features in the information space. Modeling the infinitely detailed real world into a finite system space, likewise, requires methods that translate the complexities of the world into simpler representations.

People intuitively draw on an information space that is at different levels of details and perform shifts among these granularities (Hobbs 1990). From a computational viewpoint, generalization methods are commonly used to generate simpler

representations by manipulating the attributes of entities in an information space. In a GIS, map-based generalization methods are developed as a set of complex rules based on the objects' geometries and spatial relations (Buttenfield and McMaster 1991; McMaster and Shea 1992; Weibel and Dutton 1999). Object-oriented concepts of classes and instances, and properties of classes, such as inheritance and aggregation, provide another perspective of modeling generalizations based on the semantic relations and attributes of objects (Hammer and McLeod 1981; Maier and Stein 1981; Stroustrup 1991). Research into the application of object-oriented concepts to GIS has proven to offer a good design environment well suited for a GIS by offering simple and better modeling methods (Oosterom and Bos 1989; Egenhofer and Frank 1992; Worboys 1994). It has been suggested (Schiel 1989; Kosters *et al.* 1996; Stell and Worboys 1999; Timpf 1999) that generalizations need not only be based on complex geometric calculations, but also be semantically possible.

Hobbs (1990) uses granularity to refer to the notion that the world is perceived at different grain sizes or granules. Local weather, for instance, is commonly given to the granularity of a city, whereas a person's driver's license is given to the granularity of a State. For temporal data, granularity is typically defined as calendar-dependent partitioning of a time line (Wiederhold *et al.* 1991; Dyreson and Snodgrass 1995). Different granularities of time exist, such as minutes, hours, and years. Granularities can be compared, where some granularities are finer or coarser with respect to other granularities.

The concept of granularity is also inherent in object-oriented programming (OOP) principles through abstraction. The provisions in OOP to support class inheritance, by

defining base class, derived class, and friend class, models the class structure as more generic or more detailed (Stroustrup 1991). The base class can be looked upon as a generalization of the several specialized derived classes. Similarly, in databases, a query result, such as a horizontal or vertical partition of the database, or a join operation among tables generate as results different granularities of the database.

Modeling multiple granularities conveys how the entities in an information space are related and thereby provide users with embedded knowledge of a system. The different granularities can be used for domain organization and for browsing through the granularities in order to obtain more detail or more general information.

This chapter provides an overview of granularity by reviewing the different methods for modeling multiple granularities. A detailed description of the different approaches for changing granularity, such as the object-based and map-based approaches, is also presented.

2.1 Methods for Modeling Object Granularities

Modeling multiple granularities provides a valuable framework for handling and integrating the different levels of details. The different levels of granularities in an information space can be examined in order to obtain more detail or more general information. In this section we describe the different methods for modeling multiple granularities of objects that are commonly adopted by computer science research.

2.1.1 Concept hierarchies

Concept hierarchies (CHs) define a sequence of mappings from a set of lower-level concepts to their higher-level correspondences resulting in a hierarchy of concepts

(Madria *et al.* 1998; Hilderman *et al.* 1999). CHs express the different granularities of objects based on domain values of attributes. For example, a set of objects {school, post office, restaurant} can be generalized into higher-level concept as a building. CH is defined on a set of attributes of objects. The most detailed concept corresponds to specific values of attributes, whereas the most general concepts are the all or the any description. A knowledge engineer or a domain expert constructs mappings of the different levels in a CH. Many different CHs can be constructed based on different viewpoints or user preferences; however, a common CH can be associated with an attribute.

CHs are constructed as part of the database definition phase for each of the attributes by defining a classification hierarchy based on the domain values of those attributes. A CH is represented as a tree, where the leaf node corresponds to the actual data and the intermediate nodes correspond to the more general concepts. The CH providing information for the street attribute in a road network database is as shown (Figure 2.1). The leaf nodes are the actual streets. At higher levels, a CH refers to streets by the corresponding street types (such as one-way street or a two-way street) or the division or area to which the streets belong, abstracting the individual streets. CHs present only one possible group of granularities to the user without evaluating the other possibilities. To facilitate the generation of other granularities of objects, new data structures such as the domain generalization graphs (Hilderman *et al.* 1999) are introduced.

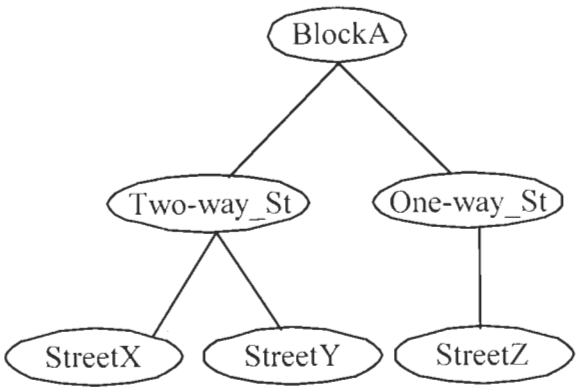
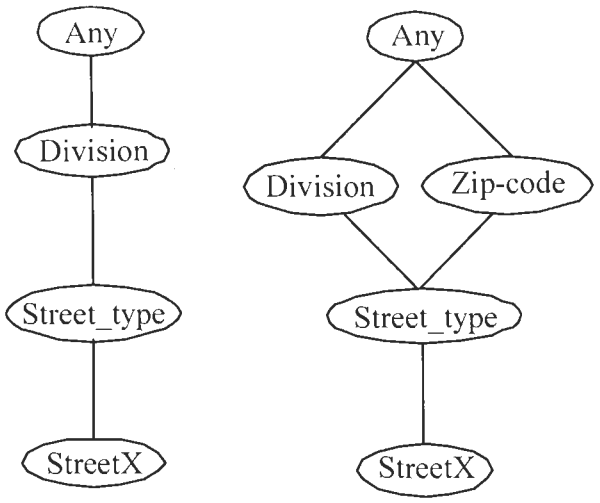


Figure 2.1 Concept hierarchy for the *street* attribute.

2.1.2 Domain generalization graphs

A domain generalization graph (DGG) (Read *et al.* 1992; Hilderman *et al.* 1999) supplements a concept hierarchy by defining a partial order that represents a set of generalization relations for the attribute. Different granularities of objects are possible, but, particularly for data mining, it is efficient and effective to limit the nodes in the hierarchies to those representing generalized encodings of the domain. The DGG models possible generalizations as a partial order rather than a strict hierarchy. A DGG is designed to include a single leaf node and a single root node. The node at each depth in a DGG is a general description of the nodes at the same depth in a corresponding CH. For example, the different nodes in a DGG for the street attribute would be the general concepts of the corresponding nodes in the CH (Figure 2.1), such as a particular street, street_type, and division respectively (Figure 2.2a). The edges in the graph denote the partial order relation between the nodes. When multiple CHs are associated with a single attribute, a multi-path DGG can be constructed for that attribute (Figure 2.2b). The generalizations modeled in a DGG are partial order relations among attributes. DGGs do

not model abstraction mechanisms such as aggregation or association, and do not make any distinction with respect to the semantic relations among objects.



(a) (b)

Figure 2.2 Single and multi-path DGG for the *street* attribute.

2.1.3 Ontologies

An ontology is defined as a specification of a conceptualization, that is, an ontology is a description (a formal specification of a program) of the concepts and relationships of entities that can exist in some domain (Guarino 1994). In the context of AI, an ontology refers to a formal model, constituted by a specific vocabulary used to describe a certain reality. It induces a set of assumptions regarding the intended meaning of the vocabulary words, and formal axioms that constrain the interpretation and well-formed use of these terms. By defining an ontology, some formal properties that account for distinctions among objects can be worked out, although complete definitions may not be given. Such formal properties result in a clearer taxonomy, clarifying the intended meaning of the concepts, reducing the inconsistencies, and producing a more reusable ontology. The relationships among the different entities in the domain, particularly the is-a relation, are

modeled at different levels in the ontology. Ontologies have been implemented as hierarchies and are formalized with constraints. The lower levels in the ontology hierarchy reveal more detail, giving the instance of a particular entity, the higher levels in the hierarchy are at a more general granularity.

The different levels in these methods are obtained by a change in the granularity. A change in granularity can occur by manipulating the geometric, spatial, and semantic attributes and relations of objects. In the following section we review the different approaches for changing granularity.

2.2 Object-Oriented Approaches for Changing Granularity

Object-oriented generalization (Smith and Smith 1977; Brodie *et al.* 1984) is targeted to exploit the semantics of objects' attributes and relations with other objects to result in coarser granularities of objects. The object model enables a separation of the complexities of the granularity used for visualization purposes, particularly dealing with geometry, maintaining geometric consistency of map features, and algorithms for processing map elements from the generalization process, limiting it to only the manipulation of semantics associated with objects. This section reviews the fundamental concepts of object-oriented models. Several implementations of the OO concepts in a GIS are also discussed.

2.2.1 Object-oriented abstraction methods

Object-oriented models decompose an information space into objects. Objects must be identifiable and describable. An object can be described by a set of attributes (such as name of a city or population), behavioral characteristics (such as a method for computing

the area of an object), and structural characteristics (such as part-whole relationships) (Winston *et al.* 1987). Each object has an identifier that uniquely defines an object. For example, a city can be described as an object with attributes name, population, and area (Figure 2.3a), and the name of the city can be considered as a unique identifier for city. Structural characteristics of an object can be specified as city X is part of a state Y or X connected to a city Z. Objects with similar behavior are organized into types. Thus, objects such as river, lake, and pond are of type water-bodies. At the implementation level the grouping of objects with corresponding attributes and methods is defined as a class. For example, Bangor and Portland can be grouped into a class city (Figure 2.3b). Giving specific values for the member attributes of a class is defined as an instance of a class. Thus, an object is an instance of a class. For example, city with name = Bangor and population = 32,000, is an instance of the class city.

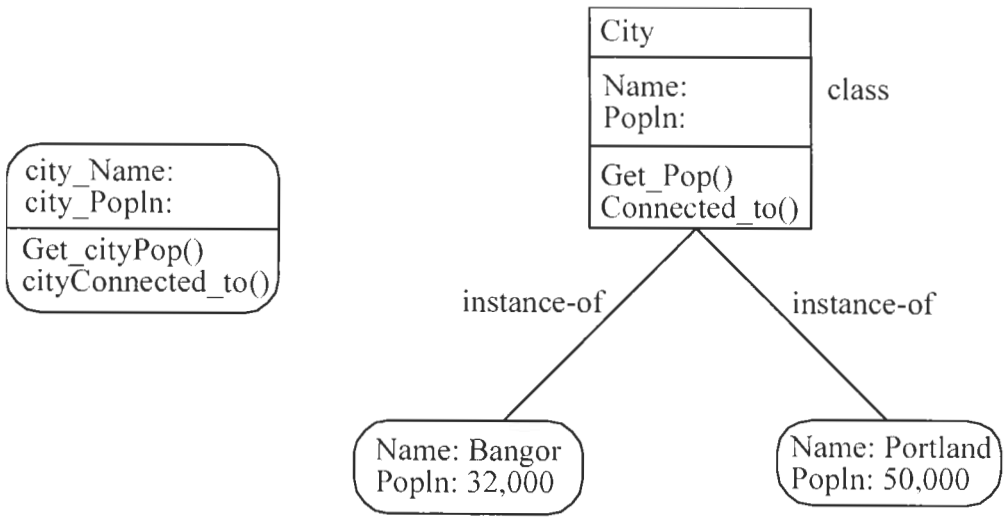


Figure 2.3 Describing (a) city as a class and (b) instances of city, Bangor and Portland.

An important object-oriented property is *inheritance*. In inheritance, generic features common to a group can be used to define a base class and then, new specialized classes, called derived classes can be created by modifying and adding existing features. For example, a generic class called *Shape* can be defined and specialized classes called *Rectangle* or *Circle* can be derived from shape. Inheritance consists of two operations, *generalization* and *specialization*. Abstracting common properties of several types into a generic type is called generalization. Generalization results in a higher-level class based on common properties of entities. For example, class *Shape* is a generalization of the different kinds of shapes, such as a *Circle*, *Rectangle*, *Triangle*, based on their common properties such as area, perimeter, etc. The reverse process of distinguishing the distinct types from a generic type according to specific roles is called specialization. Thus, *Point*, *Segment*, and *Polygon* can be considered as specializations of the class *Map_element*.

Another object-oriented method of abstraction is achieved by *aggregation*. An aggregate object is one, which contains other objects. For example, an *Airplane* class would contain *Engine*, *Wing*, *Tail*, and *Crew* as its component objects. Sometimes aggregation corresponds to physical containment (e.g., contained in the airplane). But sometimes it is more abstract (e.g., Club and Members). The condition to aggregate objects is to identify if there is a whole/part relationship between them. In linguistics, aggregate objects are called composite objects (Winston *et al.* 1987). Winston elaborated a study of the relations between the parts and wholes providing a taxonomy of part-whole relations.

Grouping of objects or *classification* is yet another method of abstraction. Classification is formed from a homogeneous set of objects, based on specific values of

common properties of objects. Rules for classification describe conditions on attributes of individual entities that must be satisfied so that the entities can be grouped into a more general entity. For example, *Urban_Area* is a classification of all the regions with population density of 0.6 and industrial growth ratio > 0.5.

Inheritance, aggregation, and generalization are abstraction mechanisms important for deriving and modeling multiple representations. These abstraction mechanisms have been implemented using the entity relation (ER) model as well as the object-oriented (OO) model (Ramakrishnan 1997). Hadzilacos and Tryfona (1997) extend the ER model to the Geo-ER model, which models the part-of relation for aggregation and member-of relation for grouping. These relations express the semantics of the geographic entities' position, spatial attributes and relations. For example, a spatial aggregation is defined if and only if the position of an object is the geometric union of the position of its geographic parts. Semantic networks used for modeling structural relationships among objects (Schiel 1989) also use generalization, aggregation and grouping as the semantic links and correspond to is-a, part-of, and element-of relationships, respectively. Similarly, abstraction hierarchies (Timpf 1999) use network graphs for data modeling, to develop conceptual schemata, and to demonstrate multiple granularities of objects.

2.2.2 Object-orientation in GIS

Geographic information systems are characterized by structurally complex information, specialized graphical requirements, and non-standard transactions (Oosterom and Bos 1989). The applicability of an object-oriented approach to GIS has been reviewed and promoted (Oosterom and Bos 1989; Egenhofer and Frank 1992; Worboys 1994). Applying object-oriented concepts, map features are modeled as special data structures

called reactive data structures to incorporate granularity information (Oosterom and Bos 1989). The level of detail is maintained in the data structure by a link to the parent object for each object in the system. The object model allows storing different representations of a map feature and also supports modeling semantic relations among objects, such as a part-whole relation, grouping two or more objects and representing them by a single graphical primitive.

Another approach that makes use of object-oriented principles is based on hierarchical structures that describe abstraction mechanisms, such as classification, generalization, and aggregation (Timpf 1999). A map is modeled as a complex object system composed of several elements and the elements of the map can be aggregated or generalized to more general objects. Timpf describes how the application of abstraction creates granularities of objects. Hierarchies are formed by factoring out the commonalities in the description of several objects into the description of a more general object.

Stell and Worboys (1999) present a method for coarsening using a multi-resolution model that distinguishes between selection of features and amalgamation of features. Each map feature is considered identifiable and describable, similar to objects. Selection refers to choosing the necessary features from the set and omitting the remaining features, and amalgamation refers to grouping features together such that some features become indistinguishable as a result. In general, a loss of detail between X and Y involves both selection and amalgamation operations. A type of generalization referred to as a simplification is presented as a selection followed by an amalgamation. This thesis

investigates the different semantics associated with object amalgamations that result in coarser granularities.

2.3 Map-Based Approaches for Changing Granularity

Multi-scale maps are commonly used to convey levels of detail of geographic space. Researchers investigated models to store multi-scale maps and developed algorithms to fetch the appropriate map representation that matched a task (Guptill 1990). Multi-scale maps posed several limitations, such as only a static set of representations was made available and expensive search routines were necessary to obtain a correct match. Further research suggested starting with a detailed map as an alternative to storing multiple representations (Beard 1990), and developing methods for changing granularity and translating among granularities. Deriving granularities was seen to be a useful alternative as it provided the flexibility with respect to the level of detail.

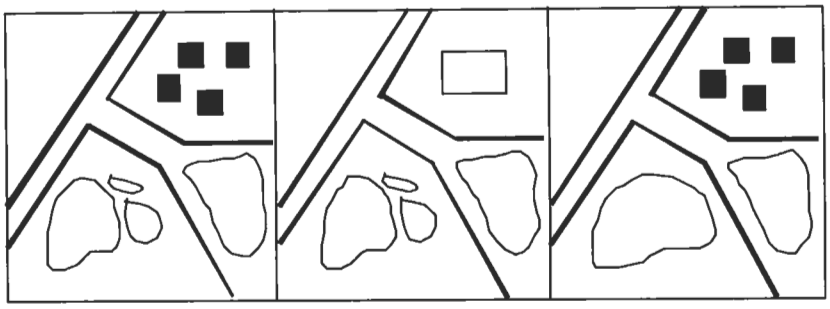
Cartographic generalization is aimed at generating visualizations and graphical symbolization of map features over multiple scales (Buttenfield and McMaster 1991; McMaster and Shea 1992; Muller *et al.* 1995a; Muller *et al.* 1995b). Cartographic generalizations reduce in scope the amount of data, scale, and graphical portrayal of map elements, to generate simple, clear, and easier-to-understand maps. McMaster and Shea (1992) present some guidelines for when and how to generalize. The map generalization process is necessary to generate maps for a specific purpose and an intended audience, with appropriate scale and clarity. It also requires methods for reducing the complexity, maintaining spatial and attribute accuracy, and for applying the rules in a consistent manner. The need for applying the methods arises when there is congestion or

complication of map features, or when there is a need to focus few map features. The generalization process involves the selection of map objects for representation followed by the manipulation of geometry and attributes of the geographic objects to generate a simplified representation of details (Weibel and Dutton 1999) appropriate to the scale and the purpose of the map.

Scale reduction automatically leads to an abstraction resulting only in the map features that have a fair resolution in the given scale. The remaining map features with very high resolution do not appear in the representation. Scale reduction alone, however, does not influence the generalization process. For the same scale, the map details required by a visitor to tour a wilderness park, for example, will be different from the ones targeted for the park ranger. Thus, the task at hand plays an important role in generalizing maps focusing on the information essential to the intended audience. Brassel and Weibel suggested a model to focus on the map features of interest by associating a measure of importance with each map feature (McMaster and Shea 1992). A phenomenological generalization method (Ormsby and Mackaness 1999) associates the degree of importance to the geometry of an object based on the object's semantics. For example, a rectangular geometry becomes more meaningful by adding whether it is a building or a tennis court. Stell and Worboys (1998) propose another formal approach for processing and reasoning about multiple granularities in spatial databases with regard to semantic and geometric precision. They organize a series of maps into a map space and define operations to shift among the map spaces. Frank and Timpf (1994) propose a multi-scale, hierarchical approach to cartographic generalization, where renderings of map objects are stored at different granularities. To obtain an output map, a top-down

selection of pre-generalized cartographic objects is performed until a sufficient granularity is reached.

Cartographic generalization is achieved through the application of a variety of generalization operators developed from the cartographic practices, each resulting in spatial abstraction. The initial step in any generalization operation is to identify the map objects of interest, called the selection process (Buttenfield and McMaster 1991) resulting in a granularity with fewer objects. Following the selection process, the selected map objects are subjected to the generalization process. Generalization operators may address the spatial component or the attributes of the map objects. A spatial transformation involves the manipulation of the object's geometry, focusing primarily on the locational aspects of data. Ten spatial transformations have been identified: simplification, smoothing, aggregation, amalgamation, merging, collapse, refinement, exaggeration, enhancement, and displacement (McMaster and Shea 1992). Each of the spatial transformations alters the data representation from a geographical or topological perspective. The aggregation operation, for instance, groups the point features in close proximity into a higher order class feature represented with a different symbolization (Figure 2.4b). Amalgamation performs a different grouping by joining contiguous polygonal features with similarities into a larger area feature (Figure 2.4c).



(a) (b) (c)

Figure 2.4 (a) Original map and (b) generalized map using aggregation operation and (c) amalgamation operation.

In addition to the spatial transformations, the other type of generalization operation is attribute transformation (Buttenfield and McMaster 1991) involving the process of classification / symbolization of map features based on the attributes of map objects.

2.4 Visualization-Based Approaches

Another approach for generalization focuses on the visualization of the map. Applying the generalization operations to map features locally can suffer from the drawback of losing context. For example, several levels of zoom-in operations, performed to understand the local details, are likely to move away from the original context of the map. To avoid this situation, it was necessary to model the local details in context with the global structure. Fisheye views (Furnas 1986) implement a strategy to provide a balance of the local detail and the global context. Fisheye views are generated by Degree of Interest (DOI) functions, which assign to each point in the structure a weight of interest that is a threshold to determine the contents of the display. A desired level of detail can be obtained by simply showing the n most interesting points as indicated by the DOI function. The perspective wall technique is an improvement over fisheye views

(Mackinlay *et al.* 1991), enabling users to see large linear information spaces by smoothly integrating detailed and contextual views.

Pad++, a framework for exploring visualization of graphical data with a zooming interface (Bederson and Hollan 1994), supports manipulation and navigation of multi-scale graphical objects. More details of an object are seen when zooming in. When zoomed out, however, a different representation of the object is viewed than of a scaled down version. The map details are rendered based on various semantic task-based considerations. Tanaka and Ichikawa (1988), describe a similar approach based on the degree of importance and semantic categories of map objects. Maps are derived through a step-by-step refinement of user preferences through visual feedback. The semantic zooming operation is used to control the display of map features according to the user's interest and the importance of categories, while a semantic panning operation specifies the different categories of relevant map elements. A more detailed map is obtained through semantic zoom-in. By semantically panning to a category, a thematic map is obtained that contains map elements belonging to a new category. Other kinds of operations, such as content zoom, intelligent zoom, and the filter operation, such as the magic filters (Stone *et al.* 1994), also support granularity change by improving visualization. Map-based approaches to granularity change can be intricate and detailed processes, involving geometric complexities in the generalization process. In their pursuit to provide alternate solutions, researchers have identified that generalizations can be non-algorithmic tasks (Muller *et al.* 1995a; Muller *et al.* 1995b).

2.5 Summary

Modeling multiple granularities and performing shifts among the granularities is fundamental to the process of reasoning about information space. In this chapter, we reviewed different methods to model objects at multiple granularities, such as concept hierarchies, domain generalization graphs, and ontological structures. There are different approaches for granularity change. This chapter described the object-oriented approaches, map-based, and visualization-based approaches for arriving at different granularities. The mapping community uses a set of cartographic generalization operations to generate different granularities of maps. These operations manipulate the geometric properties, spatial relations, and attributes of map objects spatial entities to result in different map granularity. Visualization-based approaches manipulate the semantic properties of map objects. These approaches involve complex geometric computations, are expensive to compute, and subjective in nature. Alternate non-computational generalizations for objects are discussed by using an object-based approach for granularity change, where the information space is treated as a set of objects, and the object's attributes and relationship among objects are manipulated to lead to different granularities.

In the next chapter, we present a novel approach for arriving at granularities of objects through the process of coarsening. The different semantics of objects that lead to a coarser granularity of objects are identified and a set of coarsening operators for arriving at coarser objects is presented.

Chapter 3

DERIVING OBJECT GRANULARITIES THROUGH COARSENING

This chapter identifies and presents different ways for arriving at multiple granularities of objects. In this thesis, we adopt object-based generalization operations to arrive at coarser granularities, namely *filter* and *amalgamation* (Section 2.2.2) (Stell and Worboys 1999). A reduction in detail is obtained by applying a filter to objects followed by amalgamating them. Compared to filtering, which mostly depends upon a user's interest or requirements, amalgamation can be complex as there are different ways in which objects can be combined to result in coarser granularities. This chapter distinguishes the different types of amalgamations by identifying the different semantics associated with amalgamations leading to coarser granularities. Filtering and the different kinds of amalgamations are described as operations that contribute to multiple granularities of objects through coarsening. Based on the filter and the kinds of amalgamations, a set of coarsening operators is presented to create coarser granularities of objects from a detailed granularity. The coarsening operators represent a method to achieve a richer set of simplifications.

3.1 Coarsening

Coarsening is defined as the process of transforming a granularity of objects in an information space into a less detailed granularity. It is a method for creating different object granularities. There can be more than one way to obtain coarser object granularities. In this thesis, coarsening is based on the two fundamental operations: filtering and amalgamation. We begin with a universal set of objects U that consists of all the objects in an information space. Coarsening is applied to objects in U such that the resulting granularity contains fewer objects.

Objects in U are filtered or amalgamated to arrive at a coarser object granularity. Filtering is applied to select a subset of objects from a set while omitting the other objects. Objects can be filtered from the set U such that the resulting coarser granularity contains fewer objects (Figure 3.1a). For example, applying filters to objects x_3 and x_4 from $U = \{x_1, x_2, .., x_n\}$ selects these objects from U resulting in a coarser granularity $\{x_3, x_4\}$. In a different way, objects in U can also be amalgamated resulting in a coarser granularity with fewer, coarse-grained objects.

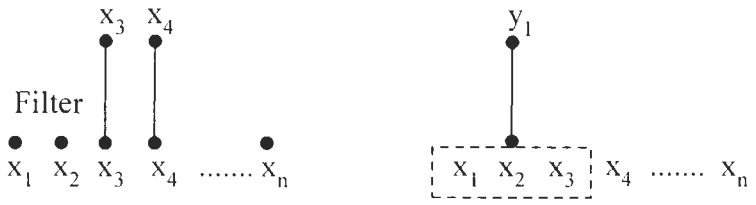


Figure 3.1 (a) Filtering objects x_3 and x_4 from the set $U = \{x_1, x_2, ...x_n\}$ and (b) amalgamation of objects x_1, x_2, and x_3 into y_1.

Amalgamation refers to the process of combining or grouping objects in order to arrive at a coarser granularity of objects. For instance, objects in U, x_1, x_2, and x_3 are amalgamated to a coarser granularity y_1 (Figure 3.1b).

Attributes of objects are used in deriving different granularities in generalization graphs (Hamilton and Randall 1999). However, attributes of objects only partially contribute to the generalization and do not capture all the possible semantics associated with grouping objects for arriving at a coarser granularity. Alternatively, there exist several hierarchical relationships among objects that model the semantics among objects (Schiel 1989), such as *Echo lake* is a type of *Lake* or *Echo lake* is part of *Acadia National Park*. Object-oriented approaches for generalization use the relations among objects to describe the different abstraction mechanisms, such as generalization, aggregation, and classification (Smith and Smith 1977; Brodie *et al.* 1984).

In this thesis, we investigate the different semantics associated with amalgamations of objects based on common attributes of objects, common attribute values, and similar relationships of objects with other objects. There are different ways to perform object amalgamation depending on an object's attributes and relations with other objects. We identify three kinds of object amalgamations: (1) grouping similar objects, (2) combining component objects to form a composite object, and (3) combining objects with similar values into a collection. The different kinds of amalgamations and filtering that lead to multiple granularities of objects are compiled as a set of coarsening operators. The term *coarsening operators* refers to a collection of operators that can be applied to a set of objects and result in coarser granularities of objects. The operators include *filter* and the

different kinds of amalgamations, namely *group*, *compose*, and *coexist*. Each operator is distinct and generates a unique coarse granularity of objects.

In our approach, an object in the universe U is defined by its attributes and relationships with other objects. Each attribute of an object belongs to a particular attribute type, such as integer, days of a week, or states in USA, and can only hold values that lie in the domain of their attribute type. For example, the town of *Orono* can be described as an object: Orono(population:int = 15,000; county:county-type = Penobscot County) consisting of the attributes *population* and *county* of type integer and county-type respectively and the attribute have values of 15,000 and Penobscot County. The relationship among objects is represented as *r(O1, O2)* where *r* is the relation and *O1* and *O2* are the objects connected by *r*. For example, *is-a*(Bangor, City) describes an is-a relation among the objects *Bangor* and *City*. A function *attr(Object)* is defined to list all the attributes of an object. Another function *attrVal(attr(Object))* lists the attribute values for all the attributes belonging to an object. For example, *attr(City)* = {population, county} and *attrVal(attr(City))* = {population : 15000, county : Penobscot County}. Objects are described as instances of classes or as classes. Instances of classes hold specific values for attributes, such as *City*(name = Bangor, population = 32000). Classes are more generic objects as *City*(name, population) and *Wasteland*(id, area, city). Classes have only attributes and do not have associated values. To arrive at coarser granularities of objects, we investigate the objects relationships between: instance-instance, instance-class, and class-class for the different kinds of amalgamations. The next section presents a detailed discussion on the coarsening operators with formalisms for each operator and examples.

3.2 Filter

Filtering objects of interest is a common method used to arrive at a coarser granularity of objects. Cartographic (McMaster and Shea 1992) and visualization-based generalization (Furnas 1986; Stone *et al.* 1994) approaches use a selection operation for choosing certain map features as being crucial for the abstraction process. Stell and Worboys (1999) also define a selection operation as part of the simplification process. Selecting objects from a set of objects creates a filter hierarchy (Timpf 1999). The select macro in database query languages returns records with specific attributes of interest from a database (Ramakrishnan 1997). In this thesis, a *filter* operator is used to describe selection of a subset of objects from a set while omitting the other objects (Figure 3.2a). Filtering objects contributes to coarsening by resulting in a less-detailed granularity consisting of fewer objects. Objects are filtered depending upon the user's requirements, such as attributes or specific values of objects.

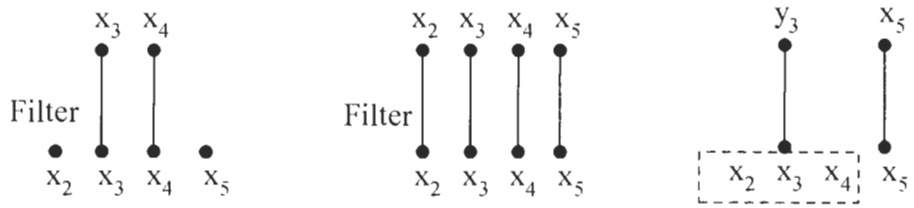


Figure 3.2 (a) Filtering objects x_3 and x_4 from $U = \{x_2, x_3, x_4, x_5\}$, (b) trivial *filter* operation selecting all the objects in U, and (c) trivial *filter* operation on x_5 and amalgamation of x_2, x_3, and x_4 into y_3.

A trivial case of *filter* is defined as applying the *filter* operator to select all the objects in a set (Figure 3.2b). Thus, a trivial *filter* does not lead to a coarsening by itself and it

must be applied only in combination with an amalgamation operation (Figure 3.2c) for resulting in a coarser granularity.

3.2.1 Formalization of the operator

If the universal set U = {all objects} defines the set of all objects in an information space, then a *filter* operator can be defined as a function $F: \wp(U) \rightarrow U$ such that,

$$F(x) = \{x: x \in U\} \tag{3.1}$$

3.2.2 Example

Consider the different metro lines present in the Boston subway network, U = *{RedT, BlueT, GreenT, OrangeT}*. The individual objects are described as:

- *OrangeT*(length:float = 18miles, no. of stops:int = 19, time:int = 33min)
- *BlueT*(length:float = 9.5miles, no. of stops:int = 12, time:int = 23min)
- *RedT*(length:float = 33miles, no. of stops:int = 22, time:int = 46min)
- *GreenT*(length:float = 31miles, no. of stops:int = 17, time:int = 42min)

The metro lines can be filtered based on their attributes or attribute values that satisfy a user's interest or requirement. For instance, the shortest metro line *{BlueT}* can be selected by applying a *filter* operator based on its attribute length. Also, a *filter* operator can be used to select metro lines having 15 or more stations resulting in a view of the metro lines that is at a coarser granularity *{OrangeT, RedT, GreenT}*.

Filter(OrangeT, GreenT, RedT, BlueT) = {OrangeT, RedT, GreenT}

3.3 Group

Objects can be amalgamated from the recognition of similar object types, factoring out the commonalities in the individual objects into a more general object (Hammer and

McLeod 1981; Worboys 1994). For instance, one can intuitively say that a *vehicle* is an object and *car* is a subtype of vehicle. Similarity object type is commonly modeled as an *is-a* relationship among classes (Smith and Smith 1977; Brodie *et al.* 1984), such as car is-a type of vehicle. The rationale for modeling an is-a relation among objects is defined by the common attribute names of objects. For example, let us consider the objects, *Lake*(name, depth, volume, type, watershed area), *Pond*(area, depth, volume, type), and *Water-body*(type, depth, volume). Among their set of attributes, *Lake* and *Pond* have the attributes type, depth, and volume in common with the attributes in *Water-body* (Figure 3.3). Since *Lake* and *Pond* have additional attributes and *Water-body* has no other attributes than those of *Lake* and *Pond*, we can say that a *Lake* is-a *Water-body* and *Pond* is a *Water-body*. Objects with common attribute names are recognized as having the is-a relation and are amalgamated to a coarser object granularity by grouping.

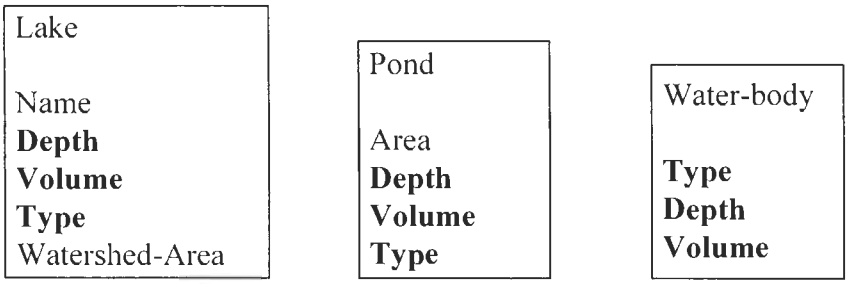


Figure 3.3 Grouping *Lake* and *Pond* into *Water-body* based on their common attributes, depth, volume, and type.

A *group* operator is defined as the amalgamation of similar objects into a common class. *Group* operator is expressed as a class-class amalgamation operation based on the is-a relation among classes. The resulting coarser object is a *superclass* (e.g., *Water-body*) and each of the combining objects is a *subclass*, (e.g., *Lake* and *Pond*). Similarity

in object types also occurs between instances and classes. Instances of a class are similar in behavior to the class but hold different attribute values for a class. For example, *Boardman Hall*(#floors:int =3, #rooms:int = 45, architect:string = "Mittal", type:bldtype = academic) and *Barrows Hall*(#floors:int =2, #rooms:int = 55, architect:string = "Raheja builders", type:bldtype = academic) are instances of a class *Building* based on the similar attributes, #floors, architect, and type. Instances belonging to a class can be grouped into a class based on the *instance-of* relation (Brodie *et al.* 1984). Thus, a *group* operator can be used to combine classes into a more abstract class or instances of a class into a class.

3.3.1 Formalization of the operator

If the universal set U = {all objects} defines the set of all objects in an information space, then the group operator can be expressed as a function $Gr: \wp(U) \rightarrow U$ such that for $S \subseteq U$,

$$Gr(S) = \{ y \in U: \forall x \in S, \exists \textit{is-a}(x,y) \text{ or } \exists \textit{instance-of}(x,y)\} \qquad (3.2)$$

- where $\textit{is-a}(x,y)$ if $attr(y) \subseteq attr(x)$ and
- $\textit{instance-of}(x,y)$ *if* $attr(y) \subseteq attr(x)$ and $\exists\, attrVal(attr(x))$

3.3.2 Example

Consider the following objects *Maple_Street*, *Elm_Street*, *Street*, *Highway*, *Road*, and *Boulevard*. The objects are described as:

- *instance-of*(Maple_Street, Street)
- *instance-of*(Elm_Street, Street)
- *Street*(name, id, length, speed limit)
- *Highway*(id, no. of lanes, distance, major connecting cities, length, speed limit)

- *Boulevard*(id, intersection, length, speed limit)
- *Road*(id, length, speed limit)

Objects	Attributes				
Highway	#lanes	length	speed limit	avg. traffic	cities...
Boulevard		length	speed limit	pavement width	
Street	name	length	speed limit		
Road		length	speed limit		

Table 3.1 Common attributes of objects: length and speed limit.

Based on the common attributes of objects (Table 3.1), it can be derived that *Street* has a subset of its attributes common with *Road* (i.e., length, speed limit) and the *Road* does not have any other attributes of its own. Hence, it can be derived that *is-a*(Street, Road). Likewise, *is-a*(Highway, Road) and *is-a*(Boulevard, Road). Applying the *group* operator, *Maple_Street* and *Elm_Street* are amalgamated into *Street* based on the instance-of relation among objects (Equation 3.2). And, *Street*, *Highway*, and *Boulevard* are amalgamated into a coarser object *Road* based on the is-a relation among objects (Equation 3.2).

$$Group\ (Street,\ Highway,\ Boulevard) = Road$$

$$Group\ (StreetX,\ StreetY) = Street$$

3.4 Compose

Objects can comprise of components objects. Component objects that combine to form a whole exhibit a part-of relation with the whole. For example, the individual buildings that comprise the university campus, are part-of the university campus. At times, it is sufficient to directly refer to the whole, abstracting the details of its parts or component objects. For example, the university campus, is adequate to describe the campus

boundary or the location and the details about the individual buildings can be abstracted. Component objects play a structural or a functional role to the whole (Winston *et al.* 1987). In this thesis, the foundation for addressing part-whole relationships among objects is based on spatial containment, spatial connectivity, or nearness of objects. These spatial relations capture distinct part-whole semantics among objects.

Objects can be spatially contained within another object. An object *A* contains *B* if *A* and *B* share the same interior region but do not have common or intersecting boundaries (Egenhofer 1993). For example, buildings in a university campus are contained within the campus. Objects that are contained in an object can be amalgamated into the whole based on their containment property. Spatial containment among objects is modeled as a relation *contained(A, B)*. The part-of relation among objects is defined based on a spatial containment relation.

Another basis for describing part-whole relationships among objects is through spatial connectivity among objects. Objects sharing a common boundary or a common point of contact are described as connected objects. Objects *A* and *B* are connected implies that there is a path from *A* to *B* and from *B* back to *A*. An object *A* connected to *B* is expressed by the relation *connected(A,B)*. Objects exhibiting the connected relation can be amalgamated into a whole, a coarser object representing the connected objects. For example, two minor roads *Exit51* and *Main Street* are connected to a major street *Stillwater Avenue* by the relations, *connected*(Exit 51, Stillwater Avenue) and *connected*(Main Street, Stillwater Avenue). The smaller roads can be amalgamated into the major road *Stillwater Avenue*. Objects *Exit 51* and *Main Street* are described as part-of *Stillwater Avenue* based on spatial connectivity. Thus, the result of the amalgamation

can be an object that is dominant in the connectivity among objects, e.g., *Stillwater Avenue*, or it could be a new object abstracting all the connecting objects, such as a *University Network* that consists of all the streets connected in the university.

Spatial proximity or nearness among objects is yet another way of signifying the part-whole relationships. Objects that are within a certain distance of each other can be combined into a whole based on the nearness among objects. For example, objects that are within a mile from the *Wilmington_Metro_Station* can be amalgamated into *Wilmington_Station_Area*. The different objects combining to form *Wilmington_Station_Area* are within a small radius of the station. Objects that are near each other can be modeled by the relation *near(A,B)* where object *A* is near object *B*. It might appear that the objects combining to form the *Wilmington_Station_Area* are more likely expressed as contained in the *Wilmington_Station_Area*. At a generic level, all part-whole relationships can be described using the *containment* relation among objects. With the relations, such as *connected* and *near*, we look at finer semantics of how the objects are contained to form the whole. Thus, it is true that the objects in our example are contained in the *Wilmington_Station_Area*, however, the *near* relation models the semantics of nearness among objects or the accessibility of objects from each other, which is not captured by the *contained* relation.

A *compose* operator is defined as the amalgamation of objects having a common part-of relation with another object. Part-whole relationships among objects are defined by the relations *contained(A,B)*, *connected(A,B)*, and *near(A,B)*. Among the three spatial relations, *contained* and *connected* are transitive, whereas *near* is not transitive. A *compose* operator identifies part-of relations among both instances of classes and among

classes (Schiel 1989). Since instances have specific values of attributes, the part-whole relation among instances is well defined. For example, *contained*(Civil_Engineering Department, Boardman_Hall) is a part-of relation among instances. There are also cases where the part-whole relation can exist among classes, such as *contained*(Building, University). However, classes are weakly connected to each other by the part-of relation since, not all instances of a class satisfy the part-whole relationship. For example, not every *Building* is part-of the *University campus*.

The class-instance pair for a *compose* is a special case of the operator. With multiple object granularities, objects that are classes to some objects may behave like an instance to another object. Consider for example, *Beehive Trail* and *Bowl Trail* as instances of a class *Trail*. There may also exist a relation *contained*(Trails, Acadia Park), that models all the *Trails* as contained in the *Acadia National Park*. The contained relation exists between the class *Trails* and an instance *Acadia National Park*. A contained relation modeling the parts of an instance, such as *Acadia National Park*, exhibits strong connectivity with the parts (i.e., one can say all the trails are part of the Acadia National park). Thus, compose operator from (c-i) is treated similar to compose operator from (i-i). Hence, *Trails*, which is a class for the individual trails, it is modeled as an instance for the *Acadia National Park*. The operator from class-instance enables modeling such dual roles of objects.

3.4.1 Formalization of the operator

If the universal set U = {all objects} in an information space, then a compose operator can be defined as $Co: \wp(U) \rightarrow U$ such that, for $S \subseteq U$

$$Co(S) = \{ y \in U: \forall x \in S, \exists part\text{-}of(x,y)\} \tag{3.3}$$

- where *part-of(x,y)* if $\exists$ *contained(x,y)* or
- $\exists$ *connected(x,y)* or
- $\exists$ *near(x, x')*

3.4.2 Example

Consider objects, *North_Ridge_Trail* and *Cadillac_Summit_Trail* with relations:

- *contained*(North_Ridge_Trail, Cadillac_Mountain) and
- *contained*(Cadillac_Summit_Trail, Cadillac_Mountain).

Based on their spatial containment relation, *North_Ridge_Trail* and *Cadillac_Summit_Trail* can be described as part-of *Cadillac_Mountain*. Thus, applying the compose operator *North_Ridge_Trail* and *Cadillac_Summit_Trail* can be amalgamated into *Cadillac_Mountain* (Equation 3.3).

Compose(North_Ridge_Trail, Cadillac_Summit_Trail) = Cadillac Mountain

3.5 Coexist

A different kind of amalgamation involves grouping objects that have a common purpose or objects in association with each other. Associativity among objects is described by the common attribute values of objects. Objects can have common attribute names with similar attribute values, such as *Brooke_Lake*(activity:watersport = motor boating, canoeing) and *Jordan_Pond*(activity:watersport = motor boat, canoeing, fishing). Such objects can be amalgamated based on similar attribute names and similar attribute values into a coarser object, a collection (e.g., *Acadia_Watersport*). Objects in a collection are referred to as member objects (Winston *et al.* 1987; Hornsby and Egenhofer 1998). The collection is defined by the coexistence of member objects with a common purpose and each member exhibits a *member-of* relation with the collection. Member objects do not

contribute to the structural or the functional definition of a collection and can be added to or removed from a collection.

A *coexist* operator is defined as the amalgamation of objects having a common member-of relation into a collection. The rationale for defining the member-of relation among objects is the common attribute values of objects. Hence, the member-of relation can only occur among instance-instance and instance-class. Similar to the part-whole relation among objects, a member-of relation is well defined among instances. For example, *member-of*(Building12, Theta_Kai_Fraternity). The member-of relation can also relate instances to a class. For example, *member-of*(Brown, Team) implies that *Brown* is a member of the class *Team*. The membership can also be described as each instance of the class *Team* has several instances of the class *Person*.

3.5.1 Formalization of the operator

If the universal set U = {all objects} in an information space, then a coexist operator can be defined as $Ce: \wp(U) \rightarrow U$ such that, for $S \subseteq U$

$$Ce(S) = \{ y \in U: \forall x \in S, \exists member\text{-}of(x,y)\} \tag{3.4}$$

- where $member\text{-}of(x,y)$ if $\bigcap_i attrVal(attr(x_i)) \neq \varnothing$

3.5.2 Example

Consider the different engineering departments, *Spatial*, *Civil*, *Mechanical*, *Electrical*, and *Chemical*.

- *Spatial*(Department:ProgramType = Engineering, Location:building = "Boadman Hall")
- *Civil*(Department:ProgramType = Engineering, #Labs:int = 3)
- *Mechanical*(Department:ProgramType = Engineering)

- *Electrical*(Major:ProgramType = Engineering, Location:building = "Barrows Hall")
- *Chemical*(Program:ProgramType = Engineering, Location:building = "Jenness Hall")

Based on the common attribute value 'Engineering', the objects can be combined into a whole, exhibiting a common purpose, i.e., departments contributing to engineering. Applying the *coexist* operator (Equation 3.4) the objects can be amalgamated to *College of Engineering*. Every object under *College of Engineering* must have an attribute department with a value 'engineering'.

Coexist(Spatial, Civil, Mechanical, Electrical, Chemical) = College of Engineering

3.6 Summary

This chapter introduces the operations of filtering and the different kinds of amalgamations to arrive at coarser granularities of objects. The semantics of the relations among objects are explored and different kinds of amalgamations are identified. The filtering and amalgamation operations are compiled as a set of coarsening operators. Coarsening operators include *filter*, *group*, *compose*, and *coexist*. The operators are used to arrive at different object granularities and are defined by the attributes and relations among objects. The table summarizes the different coarsening operators and their valid instance-class pairs to which they can be applied (Table 3.2).

Operators	Rationale	Relation	Instance-instance	Class-class	Instance-class	Class-instance
Filter	User-defined attribute		√	√	–	–
Group	Common attribute names	is-a	–	√	√	–
Coexist	Common attribute value	Member-of	√	–	√	–
Compose	Spatial containment, connectivity, nearness	part-of	√	√	√	√

Table 3.2 Coarsening operators and the corresponding instance-class pairs to which the operators can be applied.

The next chapter describes how applying the coarsening operators can lead to the generation of a granularity graph. A set of basic graph operations is also presented for querying and browsing object granularities.

Chapter 4

GRANULARITY GRAPHS

For multiple object granularities to be effective, we organize the objects and their higher-order granularities into an integrated framework and provide a means for making appropriate shifts among objects. Coarsening operators are applied to objects in order to arrive at different object granularities. Applying the operators recursively leads to an arrangement of object granularities, called a *granularity graph*. In this chapter, we introduce a granularity graph for modeling multiple object granularities. An algorithm is presented for constructing a granularity graph and illustrated with examples. Two aspects relating to modeling multiple granularities, namely matching objects granularities and assembling object granularities, are discussed. A review of the operations for browsing a granularity graph followed with the mapping of the browsing operations on multiple granularities, are also presented.

4.1 Rationale for a Granularity Graph

Shifting among granularities is fundamental for understanding and reasoning about an information space. In order to support appropriate shifts among object granularities it is required to examine the different object granularities in an integrated setting. There exist few approaches for modeling multiple object granularities, such as concept hierarchies

and domain generalization graphs (Hamilton *et al.* 1999). A concept hierarchy provides a natural way for expressing lower-level and higher-level concepts based on domain values of attributes. A domain generalization graph models relevant attribute based generalizations of objects as a partial order. Ontologies allow formal specification of the concepts and relationships of entities as a taxonomy particularly (Guarino and Welty 2000) using the class-subclass relation. These frameworks capture attribute-based generalizations and typically model inheritance among object granularities. These models, however, do not address *all* the relations among objects that contribute to object granularities. Also, methods for modeling object granularities based on both attributes and relations of objects are yet to be well explored.

In this thesis, we introduce a semantically rich framework for modeling multiple granularities of objects called a *granularity graph* (GG). A GG is an arrangement of objects connected by the coarsening operators. The different granularities can be used for domain organization and browsing support. GGs provide users with knowledge about the semantic relations among the entities in the system. A GG facilitates shifting among object granularities, leading to a more detailed or a coarser perspective on an object. GGs can also be exploited for potentially useful, valid patterns of object granularities and to establish relationships among object granularities.

4.2 Elements of a Granularity Graph

A granularity graph is defined by a graph structure $G = (N, E)$, where N and E are the disjoint finite sets of nodes and edges, respectively. Each node in the graph represents an object at a granularity. The edges in the graph are the coarsening operators (i.e., *filter,*

group, compose, coexist) used for arriving at a coarser granularity of objects. An edge connects two different objects and no two edges can connect to the same pair of objects.

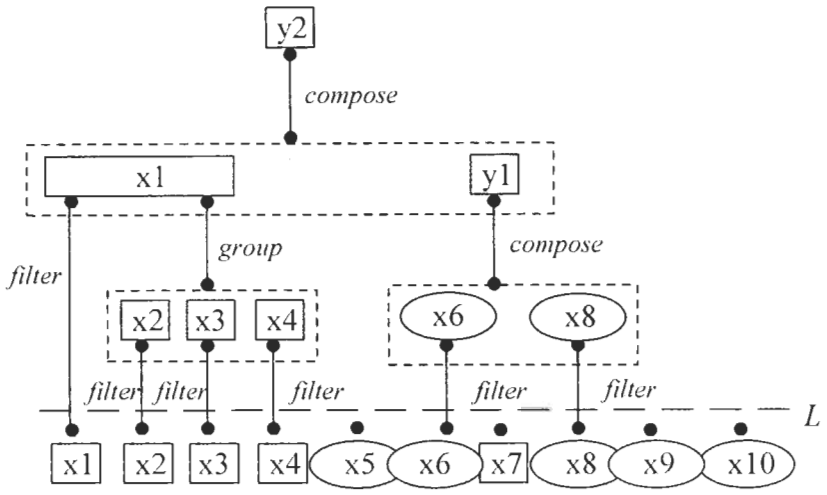


Figure 4.1 A granularity graph, $U = \{x_1, x_2, \ldots, x_{10}\}$.

Consider a granularity graph for an information space $U = \{x_1, x_2, \ldots, x_{10}\}$ (Figure 4.1). The objects or the nodes in the graph $\{x_1, \ldots, x_{10}\}$ are represented as a dot with a label. The edges are the lines connecting the nodes. The edges in the graph are labeled with the appropriate coarsening operators used for arriving at a coarser object granularity. Objects $\{x_1, \ldots, x_{10}\}$ below the line L represents all of the objects in U. Objects can also be described as an instance of a class or as a class. In a granularity graph, an instance of a class is represented as an ellipse (e.g., x_5, x_6, x_8) and a class is represented by a solid rectangle (e.g., x_1, x_2). A subset of relevant objects $S = \wp(U) = \{x_2, x_3, x_4, x_6, x_8\}$ is the leaf nodes in the granularity graph, which correspond to the fine-grained objects in U. Applying the coarsening operators, objects in S can be filtered or amalgamated resulting in coarser granularities. For example, objects $\{x_2, x_3, x_4\}$ are amalgamated to x_1 applying the *group* operator while objects $\{x_6, x_8\}$ are amalgamated to y_1 by the *compose* operator.

The object *x1* is also present in the set of objects in *U*. Hence, there is both a *filter* and *group* to *x1*. A *filter* operator is represented by an edge connecting the same object at two consecutive levels and a dotted rectangular box around the objects represents an amalgamation of objects. Objects $\{x_1, y_1\}$ are the intermediate nodes in the graph corresponding to coarser object granularities, and the root node y_2 is the coarsest object granularity for the objects in *U*.

4.3 Constructing a Granularity Graph

Constructing a granularity graph begins by defining an information space *U*. Details about objects, i.e., attributes and relations, that comprise the information space are specified as part of the definition. An object in *U* is described by a set of attributes with an optional attribute type and value, for example, *city (Name:string = Bangor, population:int = 32,000, state:USstate = Maine)*. Relationships are also expressed among two objects granularities, such as *contains(Boardman_Hall, Civil_engineering)*. Five object relationships are modeled in a granularity graph including: *is-a(A,B)*, *connected(A, B)*, *contains(A, B)*, *near(A, B)*, and *member-of(A,B)*. Based on objects' attributes and relations in *U*, coarsening operators are applied to objects. Coarsening operators can be recursively applied to the resulting object granularities leading to coarser granularities until the object can no longer be amalgamated. An algorithm (Algorithm 4.1) captures the process of building a granularity graph on a set of objects in *U*.

Algorithm 4.1 An algorithm for constructing a granularity graph

1. Select objects (i.e., the leaf nodes) from U, S= $\wp(U)$ applying the filter operator.

2. Initialize the set of resultant object granularities S′ = ∅.

3. For each operator (i.e., group, compose, coexist)

3.1. $\forall x \in S$

3.1.1. If x satisfies the necessary operator condition

mark x as red.

3.2. If the number of objects marked red > 1

3.2.1. Create a coarser object granularity y and store the relation parent(y, [objects marked red], operator).

3.2.2. ***if y ∉ U***

3.2.2.1. ***Add y to U.***

3.2.2.2. ***Add facts about y to U.***

3.2.3. Add the resultant object granularity y to S′.

3.2.4. Reset all marked objects.

4. If $\exists x \in S$ such that x is relevant to the user,

4.1. Select x applying the filter operator.

4.2. Add x to the set S'

5. If S′ = ∅

$\forall x \in U$ such that x can be amalgamated with $x \in S$, Add x to S.

else

S = S′, S′ = ∅

6. Repeat steps 3 to 5 until U = ∅.

There are two different settings in which the algorithm is used: (1) creating object granularities and (2) matching object granularities. The following two sections give an example for each.

4.3.1 Creating object granularities

An information space can contain objects at different levels of details, modeling the different perspectives of objects. Coarsening operators can be applied to objects resulting in amalgamated objects. The resultant object can already exist as one of the objects in the information space U or it can be a new object, which is not present in U. The process of creating object granularities combines objects to result in a new amalgamated object.

Object	Attributes	
	type_of_trail: type	distance: float (miles)
Beehive_trail	Strenuous	3.0
Bowl_trail	Easy	2.0
Jordan_pond_trail	Easy	5.0
Cadillac_mtn_trail	Strenuous	2.7
Great_head_trail	Easy	2.2

Table 4.1 Information space of the Acadia National Park modeling the different *Trails*.

Consider Acadia National Park as an information space U. The different trails in the Acadia park are presented as objects in U (Table 4.1). The trails have common attributes *type_of_trail* and *distance*. The domain for the attribute *type_of_trail* = *{strenuous, easy}*. A *coexist* operator amalgamates objects based on the common attribute values of objects. *Beehive_trail* and *Cadillac_mtn_trail* have a common attribute value '*strenuous*'

for the attribute *type_of_trail* (Table 4.2). Hence, by applying the *coexist* operator these trails are amalgamated to a coarser granularity *Strenuous_trails*. Similarly, *Bowl_trail*, *Jordan_pond_trail*, and *Great_head_trail* can be amalgamated into *Easy_trails* based on their common attribute value '*easy*' for the attribute *type_of_trail* (Table 4.2). The two resultant object granularities *Strenuous_trails* and *Easy_trails* do not exist in U and are newly derived by applying the *coexist* operator to objects. This process is referred as the creating object granularities. The derived objects can be added to U along with new attributes and relations of objects. Other coarsening operators are applied to objects in a similar way to derive new object granularities.

Object	Attribute	
	type: trailType	match
Beehive_trail	Strenuous	◆
Bowl_trail	Easy	□
Jordan_pond_trail	Easy	□
Cadillac_mtn_trail	Strenuous	◆
Great_head_trail	Easy	□

Table 4.2 Similar attribute values of objects for the attribute *type_of_trail*: ◆ denotes *strenuous* trails, while □ denotes *easy trails*.

4.3.2 Matching object granularities

An information space can contain perspectives based on fine-grained as well as coarser objects and the relationships among them. While combining objects to arrive at a coarser granularity it can be possible that the resulting granularity of object already exists as an object in the information space U. Thus, a matching process can be performed to compare

the resultant object with the existing objects in U. In this case, the coarsening operators play the role of establishing a connection among the granularities of objects in U.

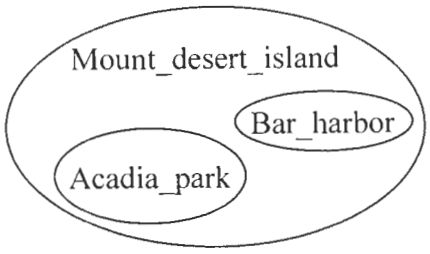


Figure 4.2 Objects exhibiting a spatial containment relation.

contained(Bar_harbor, Mount_desert_island)
contained(Acadia_park, Mount_desert_island)

Table 4.3 Relations among objects *Mount_desert_island*, *Bar_harbor*, and *Acadia_park*.

Consider the objects, *Bar_harbor* and *Acadia_park* to be contained in the *Mount_desert_island* (Figure 4.2). Based on spatial containment, a part-whole relation can be established among the objects (i.e., *Bar_harbor* part-of *Mount_desert_island* and *Acadia_park* part-of *Mount_desert_island*) (Table 4.3). The *compose* operator enables the amalgamation of objects by determining the part-of relation among the objects. Therefore, applying a *compose* operator, the objects, *Acadia_park* and B*ar_harbor* can be amalgamated to a coarser granularity *Mount_desert_island* (Figure 4.3). In this case, all three objects, *Bar_harbor*, *Acadia_park*, and *Mount_desert_island*, are objects that are described by relations in U and the *compose* operator plays a role in connecting the objects to each other. Similar to a *compose* operator, other coarsening operators can also be applied for matching object granularities from the information space.

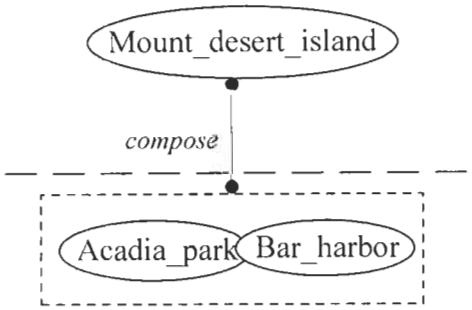


Figure 4.3 Matching object granularity: A *compose* operator is used to connect the existing objects granularities *Acadia_park*, *Bar_harbor* and *Mount_desert_island*.

4.3.3 Case study: Acadia Park

This section demonstrates how the algorithm works for constructing a granularity graph using Acadia National Park as a case study. The universal set U of all objects in Acadia Park and the relations among objects are described (Table 4.4).

1. From the set U, objects that are of interest to the user are selected as the leaf nodes of the granularity graph. These objects are typically instances of objects, described by a number of attributes and attribute values. In this example, we begin by selecting the leaf nodes $S=\wp(U)=\{Beehive\_trail,\ Bowl\_trail,\ Jordan\_pond\_trail,\ Great\_head\_trail,\ Cadillac\_mtn\_trail\}$ applying a *filter* operator. The *filter* operator selects these objects from the set U leading to a coarser granularity with fewer objects.

Beehive_trail (id:int = 01, type:trailType = strenuous, distance:float = 3.0 miles)
Bowl_trail (id:int = 05, type:trailType = easy, distance:float = 2.0 miles)
Jordan_pond_trail (id:int = 04, type:trailType = easy, distance:float = 5.0 miles)
Cadillac_mtn_trail (id:int = 02, type:trailType = strenuous, distance:float = 2.7 miles)
Great_head_trail (id:int = 07, type:trailType = easy, distance:float = 2.2 miles)
Ponds (depth:int, area:float)
Park_loop_road (distance:float = 22 miles)
Cadillac_mountain (elevation:int = 460 ft)

connected (Sand_beach, Park_loop_road)
connected (Campground, Park_loop_road)
connected (Trails, Park_loop_road)
connected (Ponds, Park_loop_road)
contained (Cadillac mountain, Mount_desert_island)
contained (Bar_harbor, Mount_desert_island)
contained (Acadia_park, Mount_desert_island)

Table 4.4 Information space for Acadia National Park.

2. Any of the coarsening operators can now be applied to the objects in *S*. In this scenario, objects in *S* have common attributes, such as *type_of_trail* and *distance* with respective attribute values (Table 4.4). Applying the group operator, which is based on common attributes, these objects can be combined into *Trails(type-of_trail:type, distance:float)*. Each object can be therefore expressed as an *instance-of* the class *Trail*. The *group* operator is applied to amalgamate objects in *S* to *Trails*, based on the instance-of relation among objects (Figure 4.4).

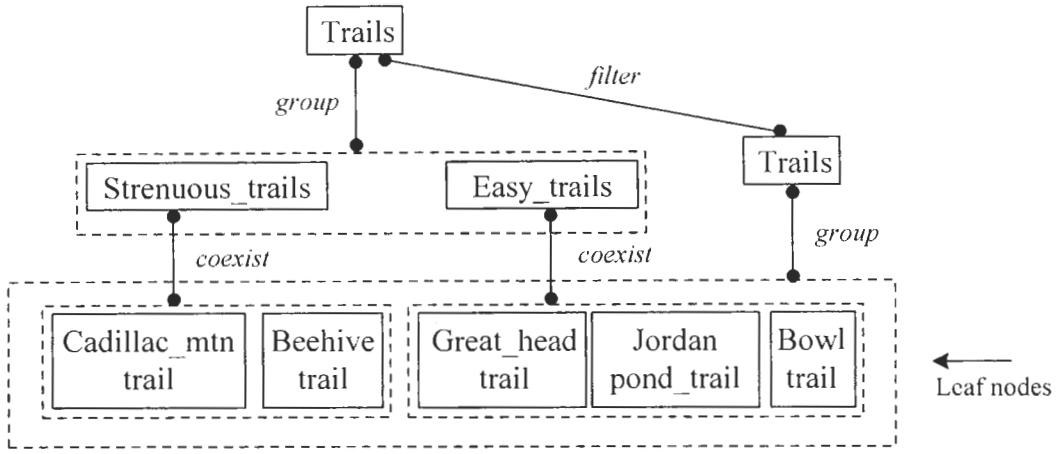


Figure 4.4 A granularity graph of the Acadia Park consisting of two levels.

The necessary condition for applying a *coexist* operator is that objects must have similar attribute values. Objects in S have common attribute values for the attribute *type_of_trail*. *Beehive_trail* and *Cadillac_mtn_trail* have a common attribute value *strenuous* trails and hence the two trails can be amalgamated to their common attribute value *Strenuous_trails* (Figure 4.4). Similarly, *Bowl_trail*, *Great_head_trail*, and *Jordan_pond_trail* can be amalgamated to their common attribute value *Easy_trails* (Figure 4.4). The resulting objects are newly derived object granularities and they are added to U. Attributes for the resultant objects, such as *Strenuous_trails(code:type = 's', id:int, distance:float)* and *Easy_trails(code:type = 'e', id:int, distance:float)* can also be included in U.

Objects in S do not satisfy the necessary conditions for the *compose* operator and hence cannot be further amalgamated. If desired, a *filter* operator can now be applied to select relevant objects. In this case, we assume that the objects do not satisfy user requirements and hence need not be filtered. Thus, after the first pass through the operators (i.e., steps 3 to 5), S = *{Strenuous_trails, Easy_trails, Trails}*.

3. The operators are recursively applied to objects in S. Applying a similar approach, objects *Strenuous_trails* and *Easy_trails* can be amalgamated into *Trails* by a *group* operator and at the end of the iteration $S = \{trails\}$ (Figure 4.4). The next pass through the steps 3 to 5 will result in $S' = \varnothing$. Thus, there are no objects that can be combined with *Trails* leading to more amalgamated objects. At this time, objects can be selected from the set of objects, *U*, and combined to create more amalgamated objects. For example, objects *Sand_beach*, *Campground*. and *Ponds* can be combined with *Trails* based on the spatial connectivity relation among the objects. Therefore, these objects are filtered and added to *S* such that $S = \{Sand\_beach, Campground, Trails, Ponds\}$ (Figure 4.5).

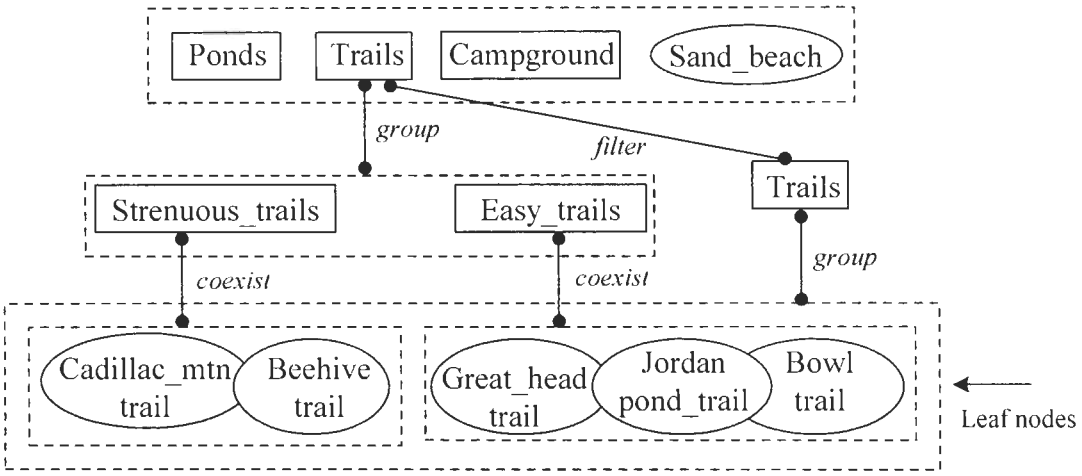


Figure 4.5 Filtering objects *Ponds*, *Campground*, and *Sand_beach* from the set *U* to result in a set $S = \{ Sand\_beach, Campground, Trails, Ponds\}$.

It is seen that there are two occurrences of the object *Trails* in the graph (Figure 4.4). In level 1, *Trails* is obtained by the process of creating an object granularity and the object is added to the set *U*. In level 2, *Trails* is already existing in *U* and hence, it is obtained by a matching process. Thus, though *Trails* is obtained differently in the two

levels, they are one and the same object. A *filter* operator is used to connect the two occurrences of *Trails*.

4. In a similar way, steps 3 through 5 from the algorithm are performed recursively to objects in S until there are no more objects in U that can be amalgamated with objects in S or until $U = \varnothing$. The granularity graph for the Acadia Park is as shown in Figure 4.6.

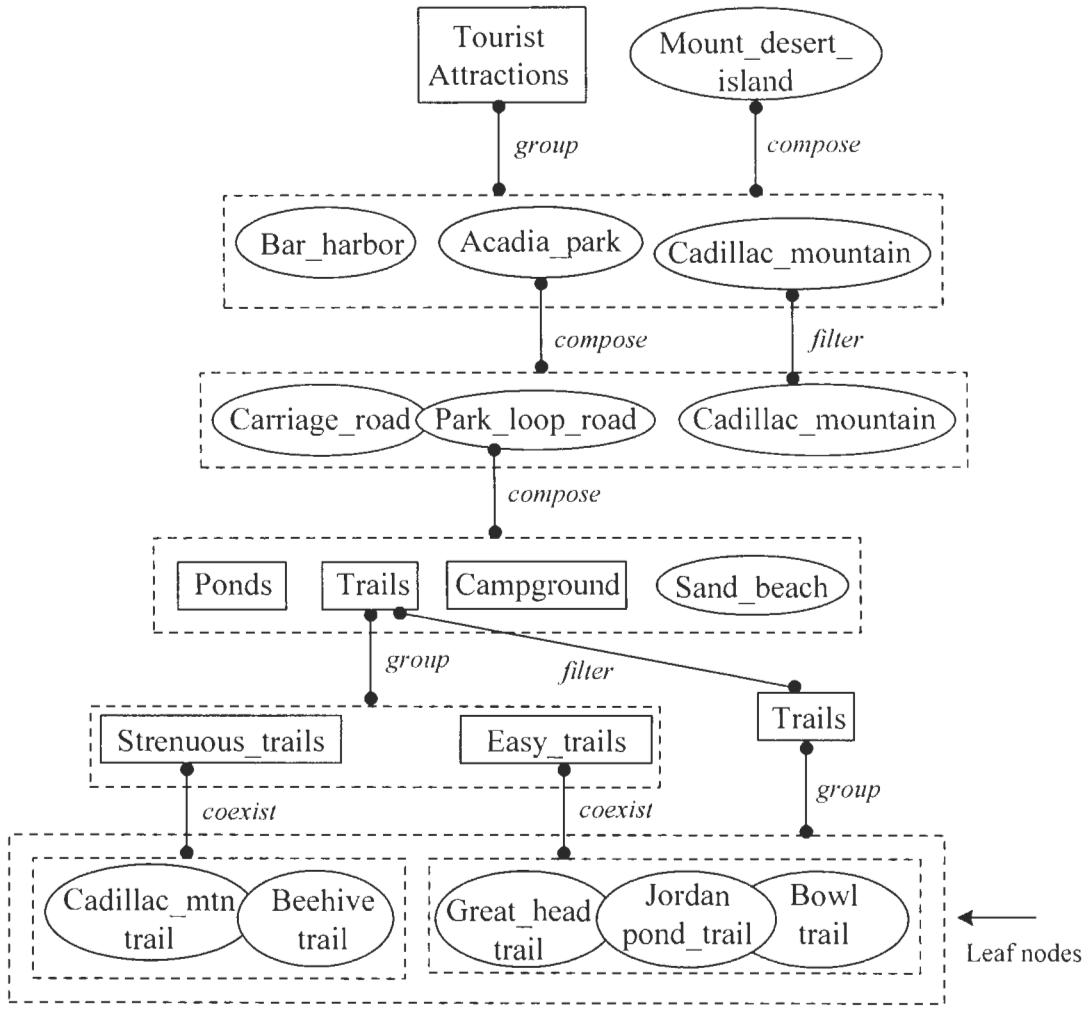


Figure 4.6 Granularity graph for Acadia National Park.

4.4 Browsing a Granularity Graph

The graph structure with connected nodes and edges facilitates translations among the different levels in a graph and supports browsing the object granularities in the graph. Browsing a granularity graph refers to making translations either to coarser granularities or more detailed object granularities in the graph. The browse operations on a graph are categorized as unary and binary operations. Simple browse operations on a graph are methods defined on a single node. Examples are methods for obtaining a parent or a child node of a particular node. These primitive browse operations can be extended for two nodes by applying set theoretic operations, such as union, intersection, symmetric difference, and set difference $\{\cup, \cap, \oplus, /\}$ to the simple operations. If X and Y are nodes in the graph and X is a parent of Y, then a relation *parent(X, Y)* can be derived from the graph. Further, if *op* is the coarsening operator connecting X and a list of objects Y, the parent relation can be expressed as *parent(X, Y, op)*. The parent relation can also be expressed with an optional operator o*p* parameter such as *parent(X, Y, _)*.

parent(Strenuous Trails, Cadillac_mtn Trail, coexist)
parent(Easy Trails, Great_head Trail, coexist)
parent(Trails, Strenuous Trails, group)
parent(Park_loop_road, Trails, compose)

Table 4.5 Axioms partly describing the granularity graph for Acadia National Park.

Following are some of the axioms (Table 4.5) that describe the granularity graph shown in Figure 4.6. Given the *parent(X, Y, op)* relation among the different nodes as axioms, the graph operations for browsing can be defined based on this relation. The operations are defined in predicate logic. Each operation is described as a predicate with its arguments. The arguments include both the operand and the result. A few predefined

operations are used in defining of the predicates for browsing. For example, *position(X, List, N)* – to find the position *N* of an element *X* in a list, *GetNumChildren(X)* to retrieve the number of children of an element *X*, *intersect(M,N,X)* – to find the intersection of two objects *M* and *N*, *union(M,N,X)* – to find the union of two objects *M* and *N*.

4.4.1 Unary operations

A unary browse operation has only one operand, the node that is operated on. A unary operation is applied to a node and retrieves as a result a node or a set of nodes. For a node *N*, the basic unary operations would be to retrieve the nodes above and below *N*, if any. The following are the different unary operations for a graph (Table 4.6).

Predicate	Description	Definition
getParent	Takes an operand *Y* and returns the parents(s) *X* of *Y*	getParent (*X*, *Y*) :- parent(*X*, *List*, _), member(*Y*, *List*).
getChild	Takes an operand *Y* and returns the *N*-th child *X* of *Y*	getChild (*X*, *Y*, *N*) :- parent (*Y*, *List*, _), position (*X*, *List*, *N*).
getChildrenOp	Takes an operand *Y* and returns a list of children List of *Y* connected by the operator *Op*	getChildrenOp (*List*, *Y*, *Op*) :- parent (*Y*, *List*, *Op*). getChildrenOp (*List*, *Y*, _) :- parent (*Y*, *List*, _).
getDescendant	Takes an operand *Y* and returns all the descendants *X* of *Y*	getDescendant (*X*, *Y*) :- getChildren (*X*, *Z*, _), For *I* = 1 to getNumChildren (*X*) getDescendant (position (*Z*, *X*, *I*), *Y*).
getAncestor	Takes an operand *Y* and returns all the ancestors *X* of *Y*	getAncestor (*X*, *Y*) :- parent(*X*, *List*, _), member(*Z*, *List*), getAncestor(*Z*, *Y*).

Table 4.6 Unary browsing operations on graph.

4.4.2 Binary operations

Binary operations take two operands and return a resultant node or list of nodes by operating on both the operands. Binary operations are extensions of the primitive unary operations and are defined using set theoretic operations, such as union, intersection, difference, symmetric difference. The primitive unary operations can be combined using the set operations leading to interesting binary operations (Table 4.7). In this section we introduce only the common binary operations based on the union and intersection of the primitive operations.

Predicate	Description	Definition
getCommonAncestor	Takes two operands *O1* and *O2* and returns the common ancestors *X* of *O1* and *O2*	getCommonAncestor (*X, O1,O2*):- getAncestor (*M, O1*), getAncestor (*N, O2*), intersect(*M, N, X*).
getCommonDescendant	Takes two operands *O1* and *O2* and returns the common descendants *X* of *O1* and *O2*	getCommonDescendant (*X, O1,O2*) :-getDescendant (*M,O1*), getDescendant (*N,O2*), intersect (*M, N, X*).
getAllAncestors	Takes two operands *O1* and *O2* and returns the ancestors *X* belonging to either *O1* or *O2* or both	getAllAncestors (*X, O1, O2*) :- getAncestor (*M, O1*), getAncestor (*N, O2*), union (*M, N, X*).
getAllDescendants	Takes two operands *O1* and *O2* and returns the descendants *X* belonging to either *O1* or *O2* or both	getAllDescendants (*X, O1, O2*) :- getDescendant (*M, O1*), getDescendant (*N, O2*), union (*M, N, X*).
getAncestorDescendant	Takes two operands *O1* and *O2* and retrieves the common relatives *X* that are an ancestor of *O1* and a descendant of *O2*	getAncestorDescendant (*X, O1, O2*) :- getAncestor (*M, O1*), getDescendant (*N, O2*), intersect (*M, N, X*).

Table 4.7 Binary browsing operations on the graph.

4.4.3 Mapping of graph operations onto object granularities

The unary and the binary browsing operations enable the retrieval of parent, child, ancestors and other connected nodes for a selected node or nodes respectively. These operations can be applied to translate up the graph to the root as well as shifting to lower levels in the graph as far as the leaf nodes. These translations can be well mapped to

operations for retrieving object granularities. The unary and binary graph operations can be mapped to two kinds of operations: refinement browsing operations that enables retrieval or shifts to finer object granularities and coarsening browsing operations that enables retrieval or shifts to more coarse granularities. The browsing operations on the graph and the corresponding mapping to the object granularities are presented.

Refinement operations - Graph operations leading to finer object granularities:

getChild (*X, Y, N*) getChildren (*List, Y, Op*)	Returns the fine-grained granularity ***X*** or granularities ***List*** of an object ***Y*** such that ***X*** and ***Y*** are connected to each other by a single coarsening operator.
getDescendant (*X, Y*)	The getDescendant method returns all the fine-grained granularities ***X*** of an object ***Y*** present in the granularity graph until the leaf nodes are reached.
getCommonDescendant (*X, O1, O2*)	The method retrieves the detailed granularity ***X*** common to the objects ***O1*** and ***O2***.
getAllDescendants (*X, O1, O2*)	Given two objects ***O1*** and ***O2***, the method retrieves the fine-grained object granularities connected to object ***O1*** or ***O2*** or both.

An example of applying the method *getDescendant(X, Y)* with *Y = Park_loop_road* is shown in Figure 4.7. The getDescendant method retrieves all the objects that are at a finer granularity to *Park_loop_road* in the granularity graph.

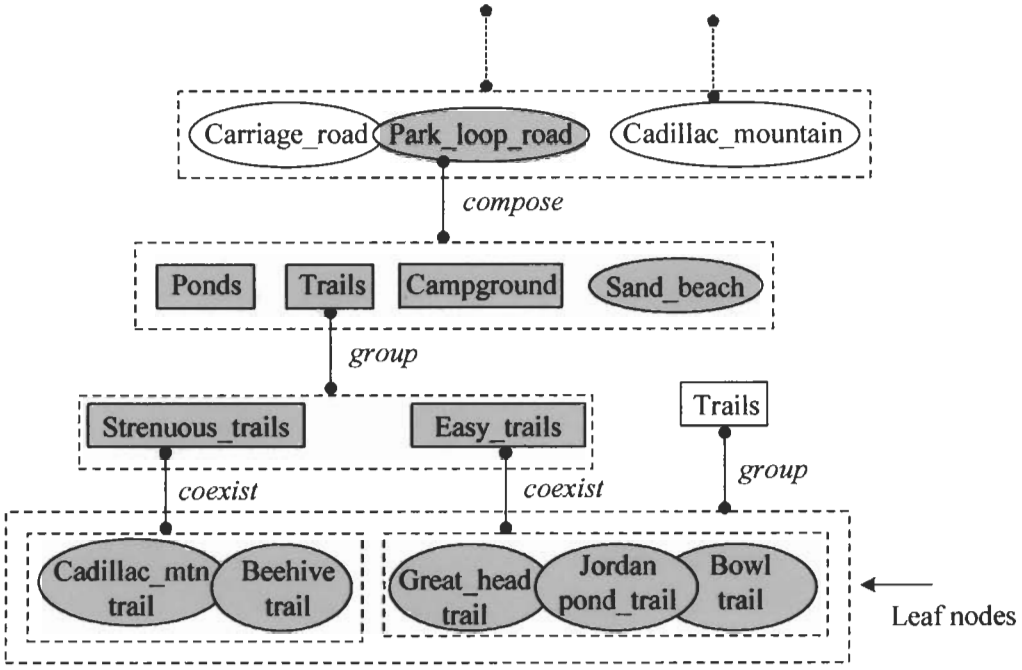


Figure 4.7 Result of the browse operation, *getDescendant(X, Park_loop_road)*.

Coarsening operations - Graph operations leading to coarser object granularities:

getParent(*X, Y*) Retrieves the multiple coarser granularities ***X*** of an object ***Y***, where ***a single coarsening operator connects X and Y***.

getAncestor (*X, Y*) This method retrieves all the coarser granularities ***X*** of an object ***Y*** until the root node in the graph (Figure 4.8).

getCommonAncestor (*X, O1, O2*) Given two objects ***O1*** and ***O2***, the method retrieves a coarser granularity ***X*** that is common to both ***O1*** and ***O2***.

getAllAncestors (*X, O1, O2*) Given two objects ***O1*** and ***O2***, the getAllAncestors method retrieves all of the coarser granularities belonging to either or both the objects.

An example of applying the *getAncestor* browse operation to *Cadillac_mountain*, retrieves all the objects that are at a coarser granularity to *Cadillac_mountain*, such as *Mount_desert_island* and *Tourist_Attractions* (Figure 4.8).

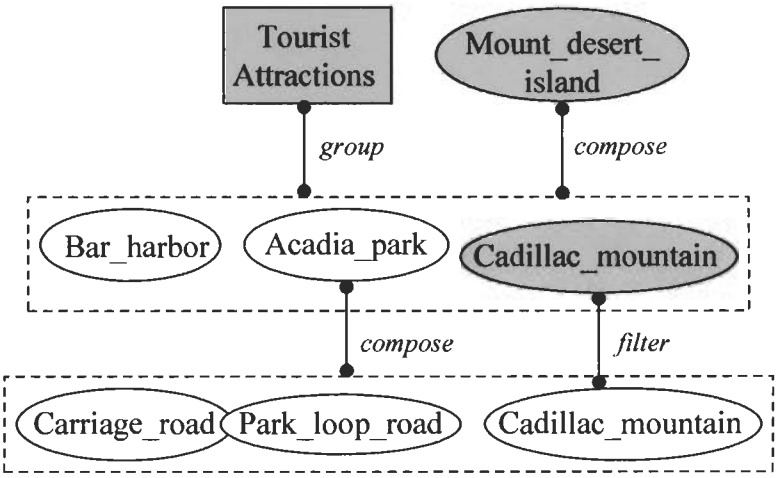


Figure 4.8 Granularities retrieved by applying the binary browse operation *getAncestor* to *Cadillac_mountain*.

A few of the binary graph operations are not strictly refinement or coarsening operations. These operations play a dual role by retrieving objects that are at a coarser granularity with respect to an object while at a finer granularity to another object. For example, applying the *getAncestorDescendant* operation to *Trails* and *Acadia Park*, retrieves the coarser granularity of *Trails*, i.e., *Park_loop_road* and *Acadia_park*, and the finer granularities to *Acadia Park*, i.e., *Park_loop_road*, *Ponds*, *Trails*, *Campgounds*, and

Sand_beach. The intersection of these two sub-operations, i.e., *Park_loop_road*, is the result of the browse operation (Figure 4.9).

getAncestorDescendant(*X, O1, O2*) The method retrieves an object ***X*** that has object ***O1*** at a finer granularity and object ***O2*** at a coarser granularity.

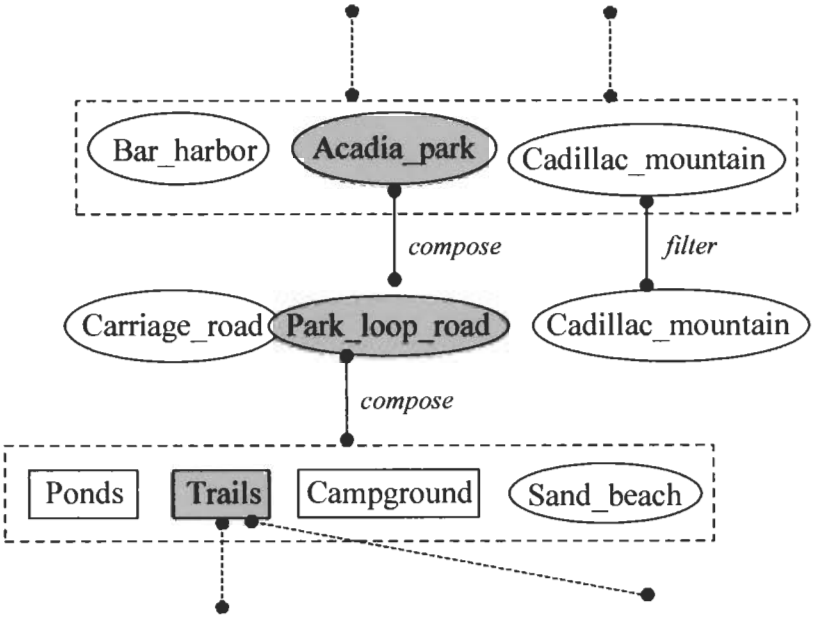


Figure 4.9 Browse operation *getAncestorDescendant(X, Trails, Acadia Park)* yields *Park_loop_road*, which is both an ancestor to *Trails* and descendant to *Acadia Park*.

In a similar way, other browse operations can be defined by combining the primitive unary and binary browse operations to retrieve a particular pattern of granularities or to retrieve object based on specific operators. Browsing operations can also be defined to retrieve all leaf nodes in the graph or all top nodes in the graph.

4.5 Summary

This chapter introduces a semantically rich framework for modeling multiple object granularities called a granularity graph. A granularity graph is constructed by applying the coarsening operators to objects in a recursive procedure. An algorithm describing the process for constructing a graph is presented. The coarsening operators can be used to derive object granularities as well as match object granularities with the existing granularity. The algorithm used for constructing a granularity graph is demonstrated by using Acadia National Park as a case study.

The granularity graph also provides a framework for performing translations among the different levels of granularities. We present a suite of graph operations for browsing an object's multiple granularities. Browsing a graph enables the retrieval of more details or less details from the graph.

In the next chapter we investigate the compositions of the coarsening operators and derive valid composition.

Chapter 5

COMPOSITION OF COARSENING OPERATORS

Granularity graphs provide a framework for objects at different levels of detail, related by coarsening operators. An information space consisting of a large number of objects or objects comprising an extensive structure may lead to a granularity graph of significant depth. With an increase in the number of levels in the graph, the chain of operators connecting objects at finer granularities to objects at a coarser granularity also increases and it would be required to simplify the sequence of operators. Compositions of coarsening operators can be used to collapse or simplify the sequence of operators. Simplifying the sequence of operators between two objects supports in deriving a shorter path in the granularity graph. It also provides for efficient retrieval of objects from the graph and for determining how two objects at different levels of detail are related in the graph. Compositions of operators can also be used in determining the multiple ways of arriving at a granularity. In this chapter we explore the composition of operators for the four coarsening operators used in constructing a granularity graph: *group*, *compose*, *coexist*, and *filter*.

Let A, B, and C be objects at different levels of detail in a granularity graph. Given a coarsening operator R from A to B and a coarsening operator S from B to C, the

composition $T = R \otimes S$ of the operators R and S (where R is performed first followed by S) yields an operator from A to C (Figure 5.1), which is defined by:

$$R \otimes S = \{(A,C): \text{ there exist an object } B \text{ for which } R(A,B) \text{ and } S(B,C)\} \qquad (5.1)$$

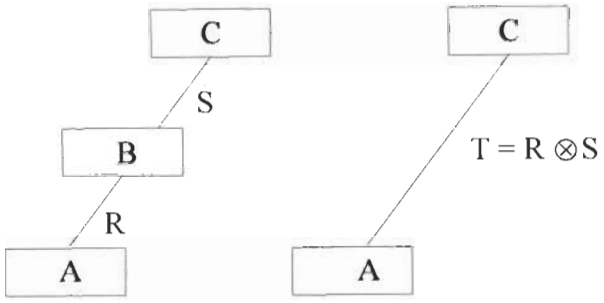


Figure 5.1 Composition of operators R and S yield $T = R \otimes S$ from A to C.

In order to work with the compositions we define a path in a granularity graph. Objects in a granularity graph are connected by a sequence of coarsening operators, which constitutes a path. Consider a granularity graph constructed for objects that are related to Acadia National Park consisting of six levels of objects at different granularities (Figure 5.2). A simple path in the graph is an alternating sequence of objects and operators, such that an operator R_i begins at an object A and ends at an object B. For example, a path from *Beehive Trail* to *Acadia Park* is {*Beehive Trail, group, Trails, filter, Trails, compose, Acadia Park*}. The number of operators in the path denotes length of the path. Thus, length of the path from *Beehive Trail* to *Acadia Park* is 3, consisting of a sequence of operators *group*, *filter*, and *compose*. One of the applications of the compositions is to reduce the length of a path in a granularity graph up to a minimum number. The applications are discussed in detail in Section 5.4.

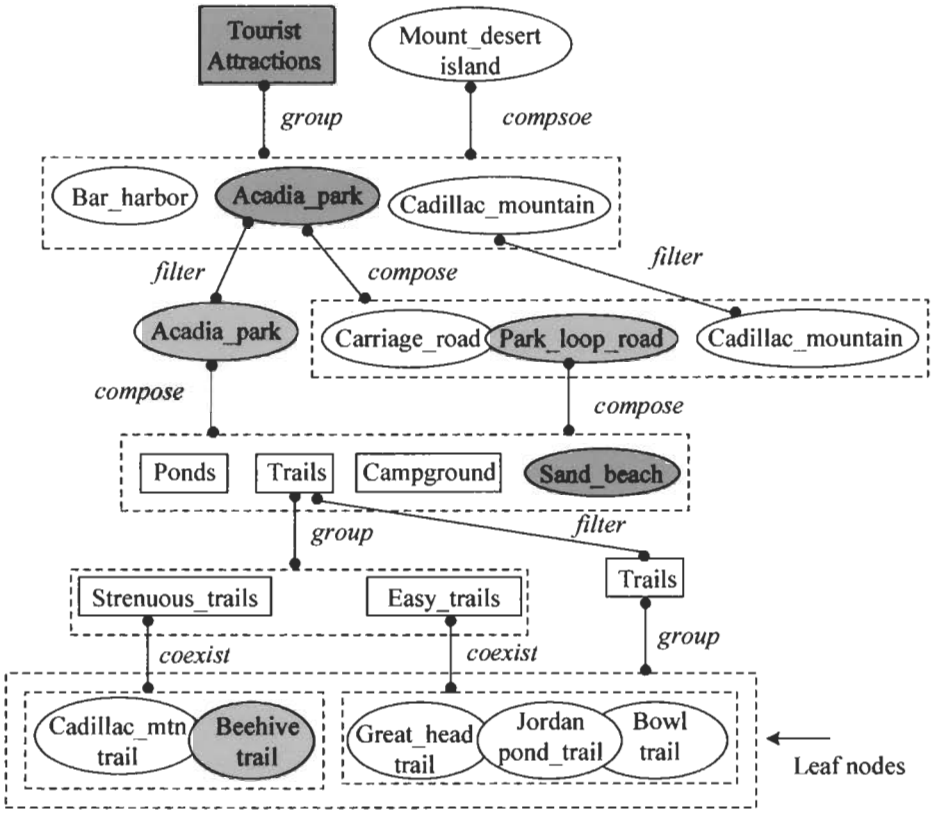


Figure 5.2 Granularity graph for Acadia National Park.

In the remainder of this chapter, we derive the exhaustive composition of all coarsening operators, determine valid compositions, and present several useful applications that will assist the development of efficient retrieval methods for multiple granularities.

5.1 Definitions for Composing Coarsening Operators

Different possible compositions of the coarsening operators, including *filter*, *group*, *compose*, and *coexist* are investigated. Compositions of operators are defined based on the structure of an object and operators that are applied. Among the four coarsening operators, the *compose* operator is complex, defined by a part-of relation among objects. The part-of relation is modeled based on the spatial relations among objects, such as

containment, connectivity, and nearness (Chapter 4). To analyze compositions correctly, we distinguish three kinds of compose operators based on its spatial relations (Section 3.4), that is, *compose[Contained]*, *compose[Connected]*, and *compose[Near]*. Including the three kinds of compose with the other coarsening operators, gives six coarsening operators in the context of composition of operators. For each coarsening operator the valid instance-class pair to which the operators can be applied are also recognized (Chapter 3).

	Instance-instance	Class-class	Instance-class	Class-instance
Filter	√	√	–	–
Group	–	√	√	–
Coexist	√	–	√	–
Compose[contained]	√	√	√	√
Compose[connected]	√	√	√	√
Compose[near]	√	√	√	√

Table 5.1 Coarsening operators and their corresponding instance-class pairs.

The compositions can be redefined as follows. If R is an operator given as a function $R: A \rightarrow B$ and S is another operator given as a function $S: B \rightarrow C$, then the compositions of operators in which the co-domain of R is the domain of S, are possible (Table 5.1). Based on instance-class pairs (i-i), (i-c), (c-c), (c-i) for each operator, there exists 18 valid pairs. Thus, there are a total of 18x18 = 324 compositions of operators. Each of the *filter*, *group*, and *coexist* operators have two valid instance-class pairs. Hence, they can be combined with other with other possible operators (Table 5.1) can lead to 3x(2x18)=108 compositions. The types of compose operators in composition amongst themselves can lead to 12x12 = 144 compositions and 12 x 6 =72 compositions with the other operators.

Summing these gives us 324 possible compositions of coarsening operators. For each composition, we evaluate cases of objects, as instances and as classes.

5.1.1 Compositions with filter

A *filter* operator is used to select a subset of objects from a set. The selected objects become available for amalgamation. While treating compositions with *filter*, we use the trivial *filter*. The trivial *filter* allows selecting all the objects in the set. Trivial *filter* acts as an identity operator. Applying a *filter* to an instance or a class results in the same instance or class, respectively. For example, applying a trivial *filter* operator to *Easy Trails* results in the same object *Easy Trails* (Figure 5.3). Consider a trivial *filter* operator from object A to B (i.e., (i-i) or (c-c)) and a coarsening operator R from B and C. Since objects A and B are connected by a *filter* operator, $A = B$. Thus, *filter* $\otimes$ R, the composition of *filter* and R, yields the operator R from A to C. Similarly, the composition of a coarsening operator R with *filter*, is equivalent to applying the operator R (i.e., if R is the operator from A to B and trivial filter is applied from B to C, then $B = C$). Thus, $R \otimes$ *filter* also yields R. For example, *coexist* $\otimes$ *filter*, composition of a *coexist* operator from *{Great Head Trail, Jordan Trail, Bowl Trial}* to *Easy Trails* and the *filter* operator from *Easy Trails* to *Easy Trails* is the same as a *coexist* operator from *{Great Head Trail, Jordan Trail, Bowl Trial}* to *Easy Trails* (Figure 5.3).

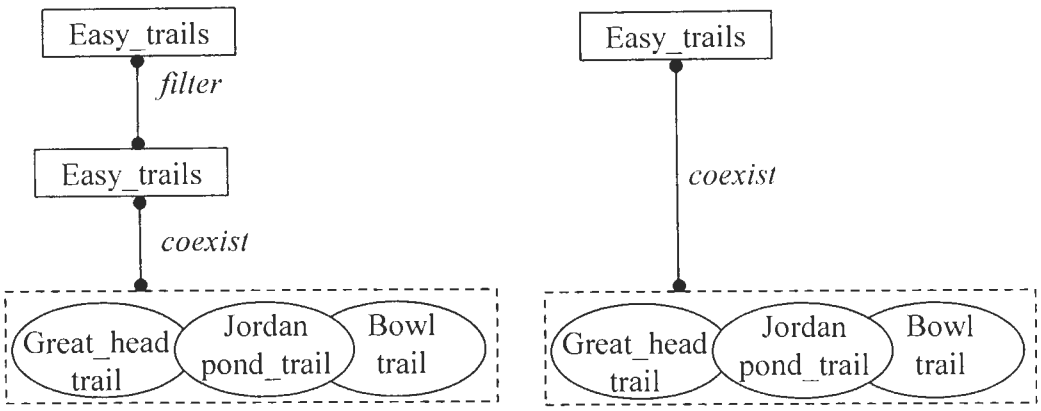


Figure 5.3 Composition of *coexist* with *filter* yields a *coexist* operator.

5.1.2 Compositions with group and coexist

A *group* operator captures an is-a relation among objects, while a *coexist* operator defines a member-of relation among objects. The is-a relation and member-of relations defining the operators are used for determining the compositions with *group* and *coexist*. *Group* operates from instance-class and from class-class (Table 5.1). Consider an is-a relation from object *A* to *B* (i.e., (i-c)) and from *B* to *C* (i.e., (c-c)). The is-a relation is transitive, that is, is-a(*A*,*B*) and is-a(*B*,*C*) implies is-a(*A*,*C*). Thus, *group* ⊗ *group*, the composition of a *group* operator with itself, yields *group*. The member-of relation defining a *coexist* operator is based on common attribute values of objects. All objects that combine by a *coexist* operator must have the same attribute value. *Coexist* can be applied from instance-instance or from instance-class. Consider a member-of relation from *A* to *B*, (i.e., (i-i) or (i-c)) and from B to C. The member-of relation among objects is also transitive, implying that *A* is a member-of *C*. Thus, the composition, *coexist* ⊗ *coexist* yields a *coexist*.

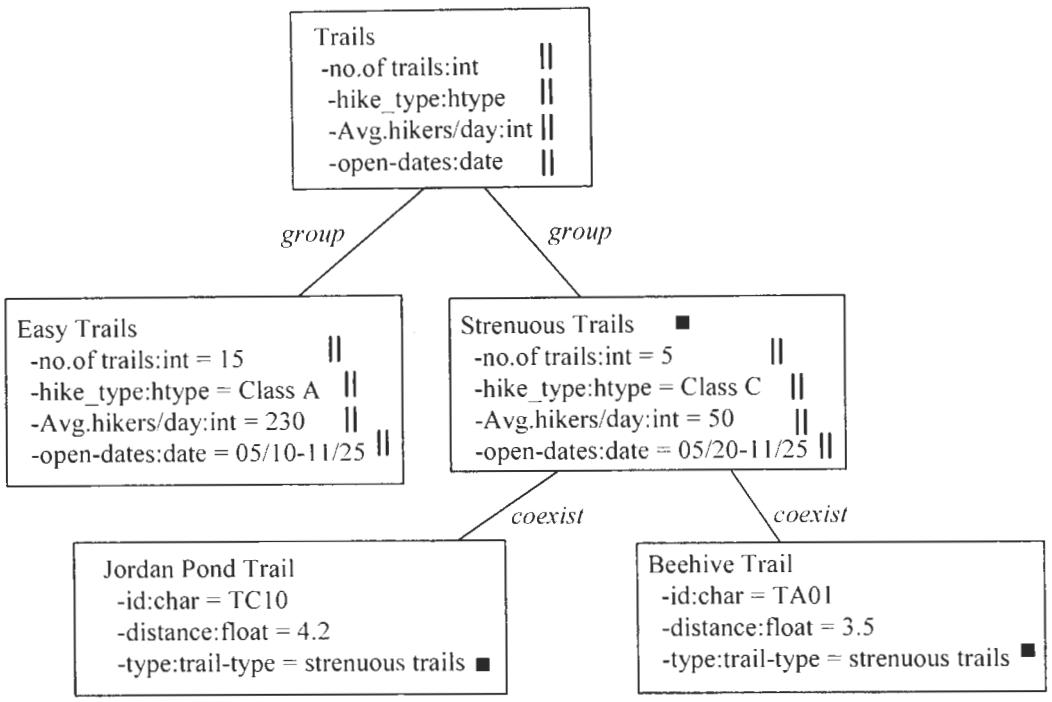


Figure 5.4 Amalgamation of objects applying a *coexist* followed by a *group*: ■- common attribute values and || - common attributes of objects.

Consider the composition of a *group* with a *coexist* operator. Since *group* is applied to instance-class or class-class, the result of a *group* is a class, consisting of only attributes of objects. The resultant object does not contain attribute values. A *group* cannot be followed with a *coexist* operator and therefore, *group* ⊗ *coexist*, composition of a *group* with a *coexist* operator cannot exist. For a *coexist* ⊗ *group*, consider a *coexist* operator from *A* to *B* (i-i) and a *group* operator from *B* to *C* (i-c). Object *B* is an instance-of class *C* (e.g., *Strenuous trails* is an instance-of *Trails*) (Figure 5.4). An object that is member-of an object is also be a member of its class. Hence, *coexist* ⊗ *group* yields a *coexist*, i.e., object *A* is also a member-of the class *C*. For example, *Beehive Trail* is a member of *Trails* (Figure 5.4).

5.1.3 Compositions with compose

A *compose* operator is modeled by spatial containment, connectivity, and nearness among objects. Compositions with *compose* are evaluated based on these spatial relations of objects. Therefore, the *compose* operator is categorized into three sub-operators *compose[Contained]*, *compose[Connected]* and *compose[Near]* operators. We first present compositions of the kinds of *compose* operators with each other and then in combination with the other operators.

A *compose[Contained]* operator from object *A* to *B* models the containment relation among objects, for example, *Sand Beach* is contained within the *Acadia Park*. Spatial containment has a transitivity property. Thus, if object *A* is contained in *B* and object *B* is contained in *C*, then *A* is contained in *C*. Hence, a *compose[Contained]* operator is transitive and its composition yields a *compose[Contained]* (Table 5.2). The transitivity property holds for objects that are classes as well as instances of classes.

A *compose[Connected]* operator from *A* to *B* and from *B* to *C* that defines a spatial connectivity among the objects is also transitive (Table 5.2). Thus, composition of *compose[Connected]* with *compose[Connected]* yields a *compose[Connected]*. However, a *compose[Near]* operator defining nearness between two objects, is not transitive. For example, consider object *A* near *B* and object *B* near *C*. The composition may result in object *A* near *C* or *A* farther away from *C*. Hence, *compose[Near]* $\otimes$ *compose[Near]* is undetermined (Table 5.2). A value undetermined for a composition implies that there can be more than one result of the composition. More information about the objects involved in the composition would be required in order to arrive at a single value of the composition.

Let us consider the composition of a *compose[Contained]* with other kinds of *compose*, such as a *compose[Contained]* operator from *A* to *B* with a *compose[Connected]* operator from *B* to *C*. The result of the composition leads to several possible relations from *A* to *C*. For example, object *A* may be near *C* or *A* may be connected to *C* (Table 5.2). There is no definite relation that can be derived from *A* to *C*. Hence, composition *compose[Contained]* ⊗ *compose[Connected]* is undetermined.

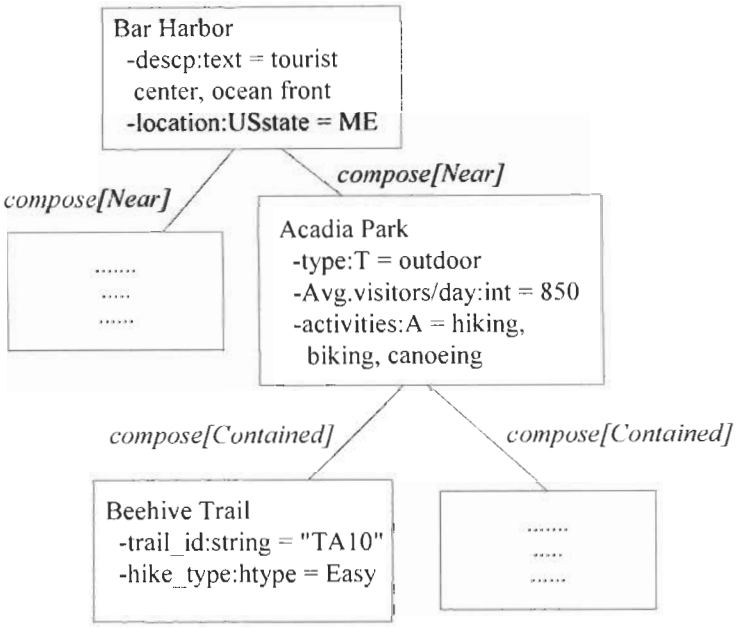


Figure 5.5 Composition of *compose[Contained]* with *compose[Near]* yields an undetermined composition.

Likewise, the compositions of the other kinds of *compose* operators with each other do not imply any definite relation from *A* to *C* and are undetermined. For example, *Beehive Trail* contained in *Acadia Park*, need not necessarily be near *Bar Harbor*, though *Acadia Park* is near *Bar Harbor* (Figure 5.5).

The table (Table 5.2) lists the compositions of each kind of *compose* operator with other coarsening operators. A row in the table describes a composition. Figures are provided to support the rationale for deriving the result of the composition. The result is a single valid composition or an undetermined composition denoted by a ~. Spatial objects used in the example are regions and roads, represented as ellipses and lines respectively, and are labeled *A*, *B*, and *C*. In each row, the first part in the figure represents the first operator in the composition, the second part represents the second operator in the composition as well as the possible result of composition. The third part, if present, describes the other possible results of the composition. For example, Row 1, *compose[contained]* $\otimes$ *compose[contained]* is described in figure as *A* contained in *B* and *B* contained in *C*. The second figure also shows the result of the composition *A* contained in *C*. Similarly, all other compositions with the compose operators are presented.

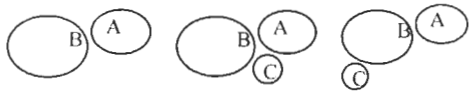
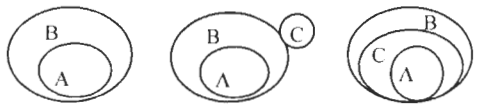
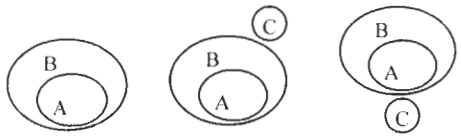
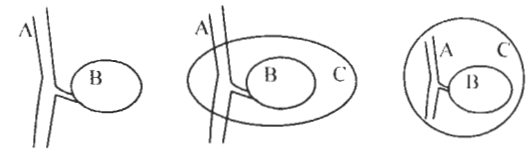
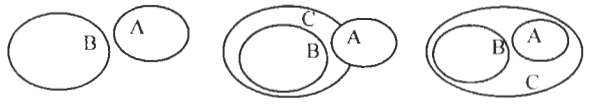
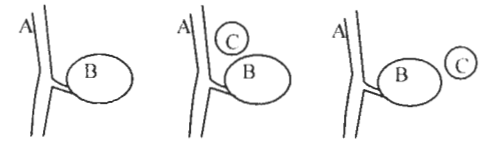
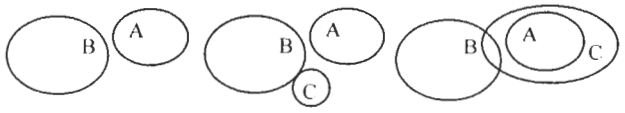
Composition	Rationale	Result
Compose[Contained ⊗ compose[Contained]		Compose [Contained]
Compose[Connected] ⊗ compose[Connected]		Compose [Connected]
Compose[Near] ⊗ compose[Near]		~
Compose[contained] ⊗ compose[connected]		~
Compose[contained] ⊗ compose[Near]		~
Compose[connected] ⊗ compose[Contained]		~
Compose[near] ⊗ compose[contained]		~
Compose[connected] ⊗ compose[near]		~
Compose[near] ⊗ compare[connected]		~

Table 5.2 Compositions of the *compose* operators with each other. ~ represents undetermined compositions.

Now, let us consider the compositions of the different compose operators with *group*, and *coexist*. For the composition of the *compose* operators with a *group*, we first consider the case with objects as instances of a class. Consider a *compose[Contained]* operator from *A* to *B* (i-i)and *group* operator from *B* to *C* (i-c). An object that is contained in another object, which is an instance, is also contained in its class. Hence, if *A* is in *B* and *B* is a *C*, then *A* is also in *C*. Therefore, the composition of a *compose[Contained]* with *group* results in a *compose[Contained]*. For example, consider the relations *Sand Beach* is contained in *Acadia Park* and *Acadia Park* is an instance-of a *Maine Attractions*. Applying the compositions, we can derive that *Sand Beach* is part-of *Maine_Attractions* by the *compose[Contained]* operator (Figure 5.6). This holds true for the other types of *compose* operations also. Thus, any *compose* with a *group* will result in the corresponding *compose* operator.

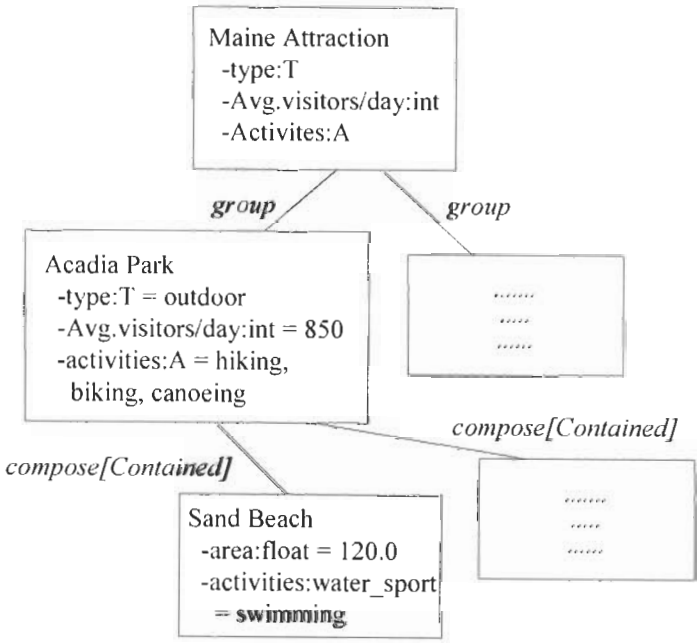


Figure 5.6 Composition of *compose[Contained]* with a *group* over instances yields a *compose[Contained]*.

The reverse case, composition of a *group* with *compose[contained]* yields different results depending on instance and classes of objects. Consider a *group* operator from *A* to *B* (i-c) and a *compose[Contained]* operator from *B* to *C* (c-c). Class *B* is contained in *C* and therefore an instance of the class *B*, *A*, is also contained in *C*. Hence, *group* ⊗ *compose[Contained]* yields a *compose[Contained]* operator from *A* to *C*. For example, *Echo Lake* is an instance of a *Lake* by a *group* operator and *Lake* is part-of the *Cadillac Mountain* by a *compose[Contained]* operator. This suggests that *Echo Lake* is also part-of the *Cadillac Mountain* by the *compose[Contained]* operator. The compositions of *group* with *compose[Connected]* and *compose[Near]* operators also yield the respective *compose* operators.

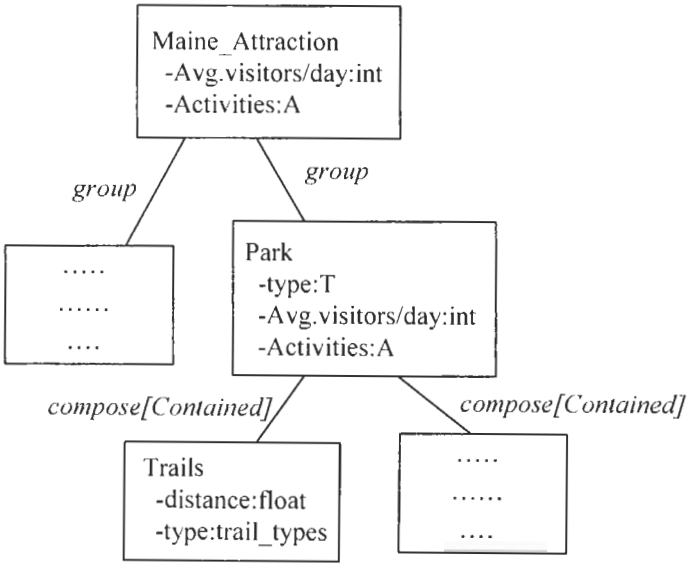


Figure 5.7 Composition *compose[Contained]* ⊗ *group* over classes yields an undetermined result.

Let us now consider the compositions of *compose* operators with *group* based on objects as classes. Let *compose[Contained]* be an operator from *A* to *B* (c-c) and *group*

an operator from B to C (c-c) (Figure 5.7). Object B is derived from class C, and can be referred to as a specialized class C that contains class A. Objects that are contained in B need not necessarily be a contained in its class C. The spatial containment of an object in B, therefore, may not hold true for C. Thus, *compose[Contained]* $\otimes$ *group* for classes is undetermined. For example, *Trails* are contained in *Park* and *Park* is a *Maine Attraction* (Figure 5.7). But *Trails* need not be contained in every *Maine Attraction*. This holds true for compositions of the other *compose* operators with *group* for objects as classes, and it is also undetermined.

Finally, compositions of the *compose* operators with a *coexist* operator are considered. A *coexist* operator operates only on attribute values. Hence, the composition of the *compose* operator and a *coexist* operator exists only for instances of classes. Consider a *compose[Contained]* operator from A to B (i-i) and a *coexist* operator from B to C (i-i). Only based on the spatial containment relation between A and B, not much can be inferred about the common attributes values of A with C. Therefore, the composition *compose[Contained]* $\otimes$ *coexist* is undetermined for instances of classes. Similarly, the compositions of a *compose[Connected]* and *compose[Near]* with a coexist leads to an undetermined result.

The reverse composition, *coexist* $\otimes$ *compose[Contained]*, for a *coexist* operator from A to B and a *compose[Contained]* operator from B to C is also undetermined. In this case too, the *compose[Contained]* operator from B to C only defines the spatial containment of B in C and does not express similarity of C with attribute values in A. Thus, A, B, and C may not have any attribute values in common. Hence, this composition is

undetermined, and this applies to the compositions with the other *compose* operators as well.

5.2 Inferences from Compositions

A complete set of all possible compositions is derived based on valid instance class pairs of operators. *Compose* is distinguished into three types, that is, *compose[Contained]*, *compose[Connected]*, and compose[*Near]*. Of the 324 compositions of operators there exist 160 possible compositions, of which there are 74 valid and determinable compositions and 86 undetermined compositions. The labels, *F*, *G*, *Ce*, *Cn*, *Co*, and *N* denote the coarsening operators *filter*, *group*, and *coexist* and the types of *compose* – contained, connected and near, respectively. The compositions are separated into two tables as compositions over classes (Table 5.3) and over instances (Table 5.4). The empty cell denotes that a composition for that particular sequence of operators does not exist and cannot be performed. ~ indicates that the composition is undetermined and may be established with additional information. Valid compositions are represented with corresponding letters of coarsening operators. A composition in the table is read as coarsening operator *A* (row) in composition with coarsening operator *B* (column) yields an operator that is represented by the corresponding cell value of the combining operators. For each operator, valid instance-class pairs are specified. For example, a *group* (i-c), is a *group* from an instance to a class, in composition with a *compose[contained]* (c-c), from a class to a class, is valid and the result of the composition is a *compose[contained]* operator (i-c).

			Filter		Group		Coexist		Compose			Compose		
			(ii)	(cc)	(ic)	(cc)	(ii)	(ic)	(cc) (ci)			(ii), (ic)		
									Cn	Co	N	Cn	Co	N
Filter	(cc)			F		G			Cn	Co	N			
Group	(cc)			G		G			Cn	Co	N			
Compose	(cc)	Cn		Cn		~			Cn	~	~			
		Co		Co		~			~	Co	~			
		N		N		~			~	~	~			
	(ci)	Cn	**Cn**		Cn		~	~				**Cn**	~	~
		Co	Co		**Co**		~	~				~	Co	~
		N	N		N		~	~				~	~	~

Table 5.3 Compositions of coarsening operators over classes. ~ signifies undetermined compositions.

			Filter		Group		Coexist		Compose			Compose		
			(ii)	(cc)	(ic)	(cc)	(ii)	(ic)	(ii), (ic)			(cc), (ci)		
									Cn	Co	N	Cn	Co	N
Filter	(i-i)		F		G		Ce	Ce	Cn	Co	N			
Group	(i-c)			G		G						Cn	Co	N
Coexist	(i-i)		Ce		Ce		Ce	Ce	~	~	~			
	(i-c)			Ce		Ce						~	~	~
Compose	(ii)	Cn	Cn		Cn		~	~	Cn	~	~			
		Co	Co		Co		~	~	~	Co	~			
		N	N		N		~	~	~	~	~			
	(ic)	Cn		Cn		~						Cn	~	~
		Co		Co		~						~	Co	~
		N		N		~						~	~	~

Table 5.4 Compositions of coarsening operators over instances. ~ signifies undetermined compositions.

The compositions of *compose* operators from i-i or i-c with the *compose* operator from i-i or i-c are equal. Similarly, compositions of the *compose* operator from i-i or i-c with the *compose* operator from c-c and c-i have equal values of compositions. Hence, the columns for *compose* operator have i-i and i-c in one column and c-c and c-i together in another.

The following inferences can be made from the table:

- From the compositions over classes it can be derived that there are 40 possible c-c compositions, of which 20 are valid. Also, there are 13 valid c-i compositions of 30 possible cases. Similarly from the compositions over instances, we can derive 17 valid (i-i) compositions from 40 possible and 24 valid (i-c) compositions from 50 possible cases. From these figures, the percentages of valid compositions over classes and instances can be derived. It is seen that the compositions over c-c and i-c are most with 50% and 48% valid compositions respectively.

Class-class	50%
Instance-class	48%
Class-instance	43.3%
Instance-instance	42.5%

Table 5.5 Percentage of valid compositions over instances and classes.

- *Filter* operator acts as an identity operator in compositions.
- Result of a composition can yield a *group* only by composing a *group* with itself or with *filter*.
- *Coexist* operator is obtained by composing *coexist* with a *filter*, *group*, or with itself.

- The table does not exhibit symmetry. Thus, the compositions of operators are not commutative. For example, *coexist* ⊗ *group* → *coexist*, whereas *group o coexist* does not exist.
- It can also be inferred from the table that the *group* and *filter* operators result in maximum number of valid compositions, whereas the *compose[near]* operator has the least number of valid compositions. Hence, *group* and *filter* are the most functional operators in a composition. And *compose[near]* is the least functional operator.

	Cn	Co	N
Cn	Cn	~	~
Co	~	Co	~
N	~	~	~

Table 5.6 Compositions of the detailed *compose* operators with each other. ~ represents undetermined compositions.

- The part-of relation among objects is typically regarded as transitive, implying part-of ⊗ part-of → part-of. However, the results of compositions of the different *compose* operators with each other are not identical (Table 5.6). This suggests that the different semantics of part-of lead to different results on composition. For example, *compose[contained]* and *compose[connected]* are transitive and their respective compositions are valid. However, *compose[near]* is not transitive and hence, the composition *compose[near]* ⊗ *compose[near]* is undetermined. Also, the composition of two different *compose* operators leads to an undetermined result.

The value of the undetermined composition can be either contained or connected or near or even be not part-of (i.e., null). For example, consider *contained*(Summit_trail, Cadillac mountain) and *near*(Cadillac mountain, Echo lake). Using the general notion of part-of as being transitive, from the example, we can say that *Summit_Trail* is part-of *Echo_Lake*. But, *Summit_Trail* is not contained in *Echo_lake*, it is not connected to *Echo_lake*, and it is not near *Echo_lake*. Hence, *Summit_Trail* is not part-of *Echo_lake*. Thus, part-of $\otimes$ part-of $\rightarrow$ part-of does not necessarily hold. The result of the composition that we obtain (i.e., *compose[contained]*$\otimes$ *compose[near]* is undetermined) is correct and acceptable. Therefore, detailed *compose* operators are needed to reveal correctly the results of compositions of coarsening operators. This proves our hypothesis that

> *Different semantics associated with object amalgamations yield correct results of the compositions of coarsening operators.*

5.3 Application of Compositions

The valid compositions can be used to reduce the sequence of operators in a path connecting two objects in a granularity graph. For example, consider the path {*Beehive Trail, group, Trails, filter, compose[Connected], Park_loop_road, compose[Contained], Acadia Park*} (Figure 5.8). The sequence of operators can be reduced as in Table 5.7. Thus, a shorter path of length 2 is obtained from *Beehive Trail* to *Acadia Park* (Figure 5.8).

group ⊗ filter ⊗ compose[Connected] ⊗ compose[Contained]
→ group ⊗ compose[Connected] ⊗ compose[Contained]
→ compose[Connected] ⊗ compose[Contained]

Table 5.7 Applying compositions of operators to arrive at a shorter path.

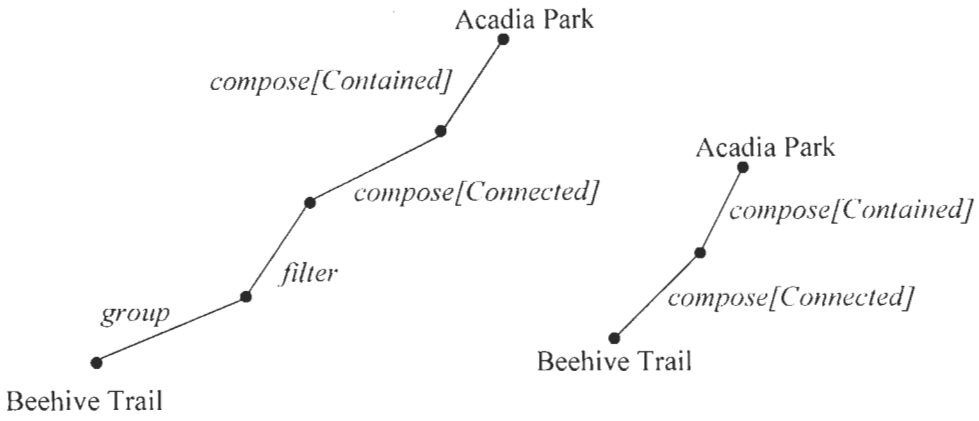


Figure 5.8 Simplifying the path from *Beehive Trails* to *Acadia Park*.

Also, applying the compositions, the original path (Figure 5.9) in the granularity graph can be replaced by the derived shorter paths. For example, a path in the graph {*Cadillac Mountain, filter, Cadillac Mountain, compose[contained], Mount Desert Island*}, can be simplified to {*Cadillac Mountain, compose[contained], Mount Desert Island*}, The number of objects in the graph is reduced, leading to a simplification of the granularity graph (Figure 5.10).

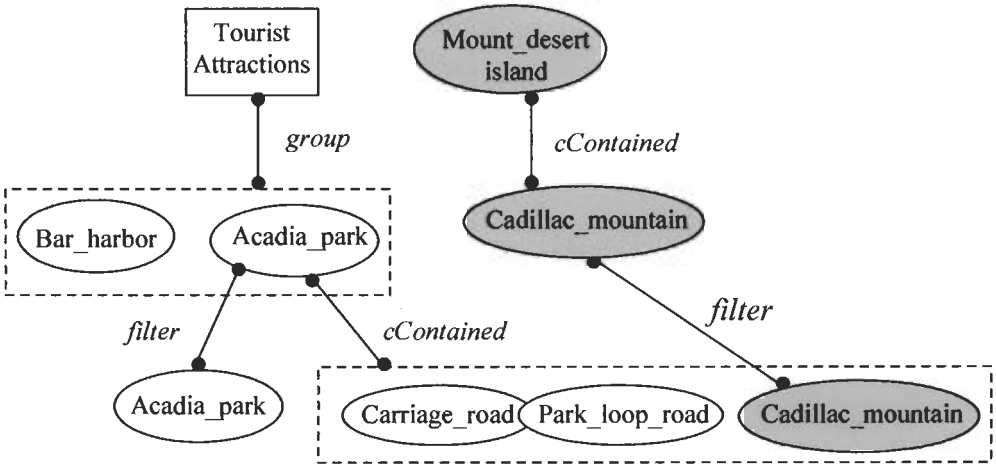


Figure 5.9 Granularity Graph with a selected path.

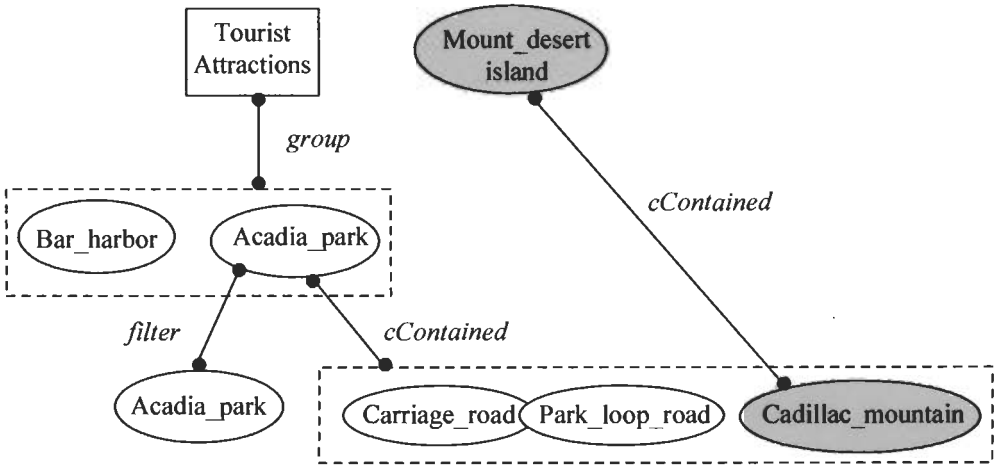


Figure 5.10 Simplified granularity graph applying compositions.

Composition of operators can also be used to derive the multiple ways of arriving at a granularity. Given *n* operators in a path, the compositions can be applied to derive up to *n-1* different ways to arrive at a granularity. For example, consider the path connecting *Beehive Trail* to *Acadia Park* (Figure 5.8). Applying the compositions, 3 different ways of arriving at *Acadia Park* is obtained (Table 5.8) consisting of 4, 3, and 2 operators respectively.

Path 1 - group ⊗ filter ⊗ compose[Connected] ⊗ compose[Contained]
Path2 - group ⊗ compose[Contained] ⊗ compose[connected]
Path3 – compose[Contained] ⊗ compose[Connected]

Table 5.8 Multiple ways to arrive at *Acadia Park* from *Beehive Trails.*

Multiple ways to arrive at a granularity is useful for retrieving the shortest path, or a path with a specific operator, or path with minimum number of different operators. If the cost of applying each operator can be evaluated, then the multiple paths can be used to obtain the most efficient way of arriving at a granularity.

The composition of operators exhibits the associative property. Thus, if *R1*, *R2*, *R3* is a sequence of operators connecting two objects at different granularities, then $(R1 \otimes R2) \otimes R3 = R1 \otimes (R2 \otimes R3)$. The composition of the operators can therefore, be applied to the sequence of operators in any order. For example, *group ⊗ compose[Contained] ⊗ filter*, yields the same result *compose[Contained]* though the compositions are applied in any order (Table 5.9).

(group ⊗ compose[Contained]) ⊗ filter compose[Contained] ⊗ filter compose[Contained]	group ⊗ (compose[Contained] o filter) group ⊗ compose[Contained] compose[Contained]

Table 5.9 Simplification of a sequence of operators using the associative property of compositions.

The associative property of the compositions is significant because it partially removes the need to apply the compositions in strict order from left to right. Applying this property, it is possible to retain a particular granularity of interest (e.g., *group*) in the graph while determining a shorter path (Figure 5.11).

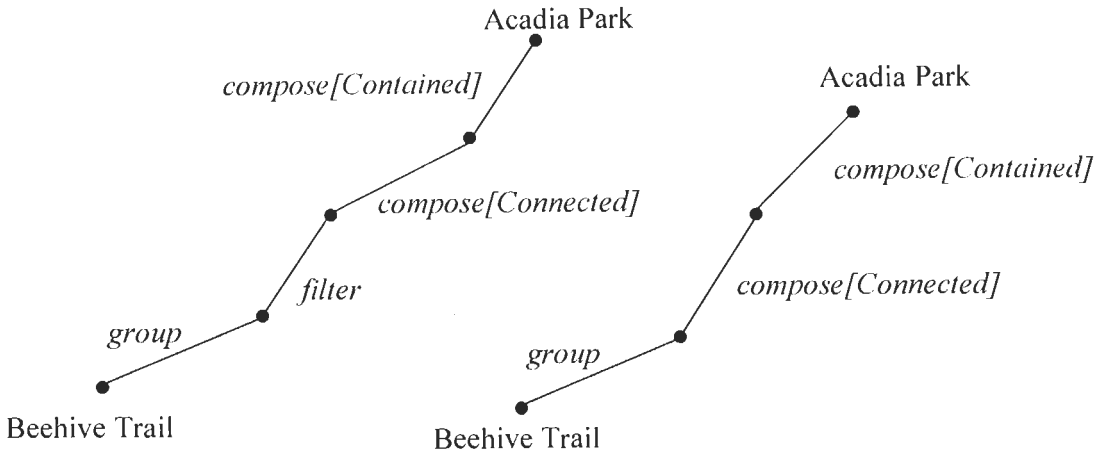


Figure 5.11 Deriving a path from *Beehive Trail* to *Acadia Park* with a *group* operator.

Another application of the operators is for determining how two objects at different levels of detail are related in the granularity graph. Consider the path from *Beehive Trail* to *Acadia Park*. There exist two paths (a) {*Beehive Trail, coexist, Strenuous Trails, group, Trails, compose[Contained], Acadia Park*} and (b) {*Beehive Trail, group, Trails, filter, Trails, compose[Contained], Acadia Park*}. Let us consider path *b*. The operators in path *b* are *group*, *filter*, and *compose[Contained]*. Applying the composition of operators (Table 5.10) to the sequence of operators in path *b*, we obtain that *Beehive Trail* is connected to *Acadia Park* by the *compose[Contained]* operator. Having reduced the sequence of operators to one, it is possible to directly relate *Beehive Trail* and *Acadia Park*, i.e., *Beehive Trail* is contained in *Acadia Park*.

group ⊗ filter ⊗ compose[Contained]
→ group ⊗ compose[Contained]
→ compose[Contained]

Table 5.10 Simplifying sequence of operators for relating object granularities.

Consider another example of a path from *Sand Beach* to *Tourist Attractions*; {*Sand Beach, compose[Connected], Park Loop Road, compose[Contained], Acadia Park, group, Tourist Attractions*}. Applying the valid compositions of operators, a shorter path (Table 5.11) from *Sand Beach* to *Tourist Attractions* with two operators, {*compose[Connected], compose[Contained]*} is obtained. This path suggests that, *Sand Beach* is connected to *Park Loop Road* and *Park Loop Road* is contained in the *Acadia Park*. Consider a second path between the same two objects consisting of {*compose[Contained], Acadia Park, filter, Acadia Park, group, Tourist Attractions*}. Based on this path, the composition can be applied to the operators resulting in *Sand Beach compose[Contained]* in *Tourist Attractions* (Table 5.11). The simplification obtained using two different paths do not contradict each other instead complement one other. Using both relations from the granularity graph, a more complete semantics of *Sand Beach* and the *Tourist Attractions* can be determined, i.e., *Sand Beach* and *Park Loop Road* are both contained in the *Tourist Attractions* and *Sand Beach* is connected to the *Park Loop Road*. The relations that are derived among objects can be stored into a knowledge base for providing reasoning as to how the objects at different granularities are related to each other.

Path a compose[Connected] ⊗ compose[Contained] ⊗ group → compose[Connected] ⊗ compose[Contained] Path b compose[Contained] ⊗ filter ⊗ group → compose[Contained] ⊗ group → compose[Contained]

Table 5.11 Two different paths connecting *Sand Beach* to *Tourist Attractions* yields analogous simplifications.

5.4 Summary

This chapter provides a detailed evaluation of the compositions of the coarsening operators. Compositions of coarsening operators are primarily required to collapse or simplify long sequences of operators in a granularity graph. We derive a complete set of 324 different compositions of the coarsening operators based on their valid instance-class pairs. The definitions for the composition of operators are presented and supported with several examples. Of the 324 compositions, 160 are possible compositions and there are 74 compositions that are valid and determinable (i.e., up to a maximum of 50% of the compositions). Therefore compositions must be exploited and used in simplifications. Several inferences are derived based on the compositions of operators. It is inferred that the different semantics associated with the coarsening operators play an important role in deriving correct results of the compositions. We support this with the detailed *compose* operators. Also, compositions enable in determining the most functional and least functional operators while arriving at a simplification.

This chapter also presents the several applications of the compositions. Compositions are used in deriving shorter paths in the granularity graph and for determining the multiple ways of arriving at a granularity. They can also be used for efficient retrieval of objects and for relating objects at multiple granularities in a granularity graph. The multiple simplifications obtained by applying the compositions support enhanced reasoning based on the object granularities.

The next chapter describes the implementation of a prototype using the algorithms discussed in Chapter 4 to test the browsing and compositions of operators.

Chapter 6

PROTOTYPE FOR CONSTRUCTING AND BROWSING A GRANULARITY GRAPH

This chapter describes the design and implementation of a prototype for modeling multiple granularities of spatial objects. The prototype allows building a granularity graph and enables shifting among granularities through browsing. The goal of the prototype is to demonstrate the construction of multiple granularities of spatial objects by applying the coarsening operators. Browsing operations on the graph for retrieving objects at finer or coarser granularities are also implemented. The following sections discuss the design of the prototype and implementation of data structures, coarsening operators, and the user interface. The working of the prototype is illustrated with an example.

6.1 Prototype Design and Specification

The prototype implements a step-wise building of a granularity graph. The design of the prototype is separated into two components: the user-interface and graph builder (Figure 6.1). The user-interface facilitates input to the application, display of a granularity graph, construction of the graph, and generation of browse results. The graph builder implements the coarsening operators. A set of objects from an information space is given

as input to the model. These objects are stored in a list structure. Coarsening operators are applied to the objects in the list resulting in amalgamated objects. The amalgamated objects are displayed in the granularity graph and appended to the list of objects. Coarsening operators are now applied to the amalgamated objects added to the list, arriving at another new set of amalgamated objects. The process is repeated until objects can no longer be amalgamated or the user requirements are satisfied. The prototype is implemented in Visual C++ (6.0), with a GUI. A viewer window is provided for displaying the granularity graph and operations on the graph. The main building blocks of the model are the objects and the functions implementing the coarsening operators.

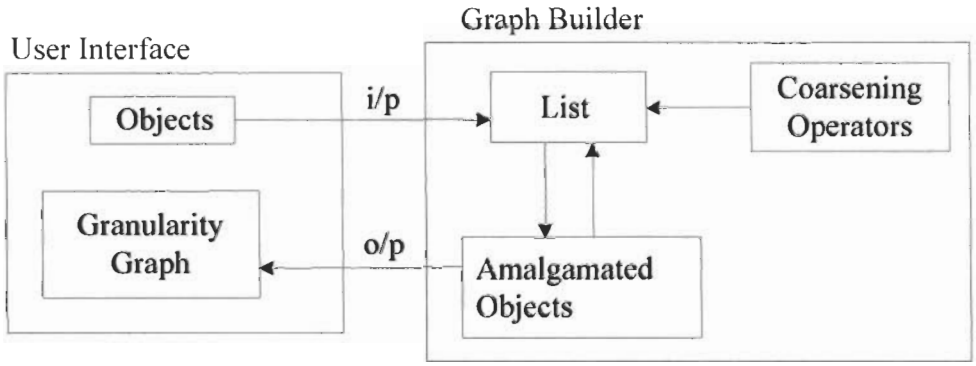


Figure 6.1 The prototype architecture: user-interface and graph builder.

6.1.1 Objects

The prototype models objects at different levels of details. A structure *GObject* is defined to store an object. Every object consists of a label *oName* for displaying the object, number of attributes of objects *oNum_attributes*, array of attribute names, attribute values, and relations with other objects *GOAttrib*. In a granularity graph, objects are connected to other objects at coarser granularities through a coarsening operator. Thus, every object can have knowledge about the objects at a finer granularity as well as the

corresponding coarsening operator connecting the objects. *GOLink* models the coarsening operator and the objects at finer granularities to an object, where *GOLink::oOpr* represents the coarsening operator and *GOLink::oChildIds* are the objects at a finer granularity (Figure 6.2). Other data members of an object, such as system id *oId*, the level of the object in the granularity graph *oLevel*, and the bounding rectangle *oRect* are private data members used for displaying the object in a granularity graph.

```
enum GOperator {Group =0,
Compose,Coexist, Filter, N};

template <class T>
struct GOAttrib
{
    CString oAttName;
    T       oAttValue;
};

struct GOLink
{
    GOperator oOpr;
    vector <CString> oChildIds;
};
```

```
class GObject: public CObject
{
    public:
        CString oName;
        int oNum_attributes;
        vector <GOAttrib> oAtt;
        GOLink oLink;

    private:
        int oId;
        int oLevel;
        RECT oRect;
};
```

Figure 6.2 Structure of an object, *GObject*.

For traversing the graph, it is required to iterate through the stored objects. Iterating the objects in the graph requires objects to be stored as a collection that can be accessed sequentially or by pointers. Therefore, we derive the object class *GObject* from the base class *CObject* enabling access to a collection of objects. *CObject* is the root class for *CObList*, which supports ordered lists of type *CObject*. *CObList* lists behave like doubly linked lists.

6.1.2 Coarsening operators

Each of the coarsening operators is implemented as a function. A class *GraphBuilder* is defined to handle the operators (Figure 6.3). Each operator function accepts an array of objects for amalgamation. Objects are selected through the GUI and an array of objects is passed to the operator function. The function compares the structure of the objects for attributes and values that are required to satisfy the operator condition and returns the object ids that satisfy the condition in a structure of type *GOLink*. For example, if two objects *Beehive Trail* (sysid:int = 1, id:char = T01, distance:float = 3.5, type:ttype = easy trail) and *Bowl Trail*(sysid:int = 2, id:char = T12, distance:float = 3.0, type:ttype = strenuous trail) are passed to the function, then comparing the structure of the objects based on the operators, the function returns a structure *GOLink* with values *GOLink::cChildIds* = {1,2} and *GOLink::oOpr* = *Group*. A new amalgamated object is created consisting of objects at finer granularities from *GOLink::oChildIds* and connected by the operator *GOLink::oOpr*. For example, *GOLink GraphBuilder::Group(GObjects* ob)* accepts an object array and returns a *GOLink* structure with object *ids* having similar attribute names.

Figure 6.3 Class *GraphBuilder*.

The user can add a label for the new amalgamated object. Additional attribute names and values for the object can be added and stored in the object list. The object gets added as a new node to the granularity graph and is connected to other objects with its respective coarsening operator.

6.1.3 Granularity graph

A granularity graph modeling multiple object granularities is implemented as a collection of objects using a linked list structure. Objects at each level are stored in a linked list derived from *CObList* (Figure 6.4). A variable of type *POSITION* is a key for the list. The *POSITION* variable can be used as an iterator to traverse a list sequentially. Objects can be inserted very fast at the list head, at the tail, or at a known *POSITION*. A sequential search is necessary to look up an element by value or index. This search can be slow if the list is long. As, an alternative, a *CMapStringToOb*, a dictionary collection class that maps unique *CString* objects to *CObject* pointers is used (Figure 6.4). Once a *CString-CObject** pair (element) is inserted into the map, an object can be efficiently retrieved using a string value as a key. It is also possible to iterate through the elements in the map.

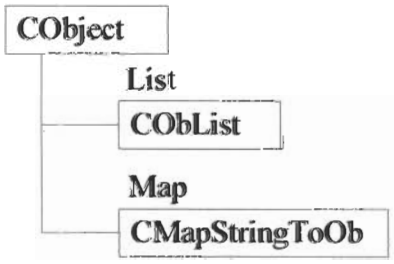


Figure 6.4 Classes for implementing the graph structure.

6.2 The User Interface

Input to the prototype, interactive building of a granularity graph, and browsing operations on the graph are the main features of the user interface. These tasks are provided as menu items in the application (Figure 6.5). The interface consists of the granularity graph window for displaying the graph, a set of coarsening operators used for building the graph, and an attribute list window for displaying the attributes and relations associated with a selected object. Selection of objects and operators on the graph is enabled through mouse clicks in the viewer window. The following sections describe the functions supported by the user interface.

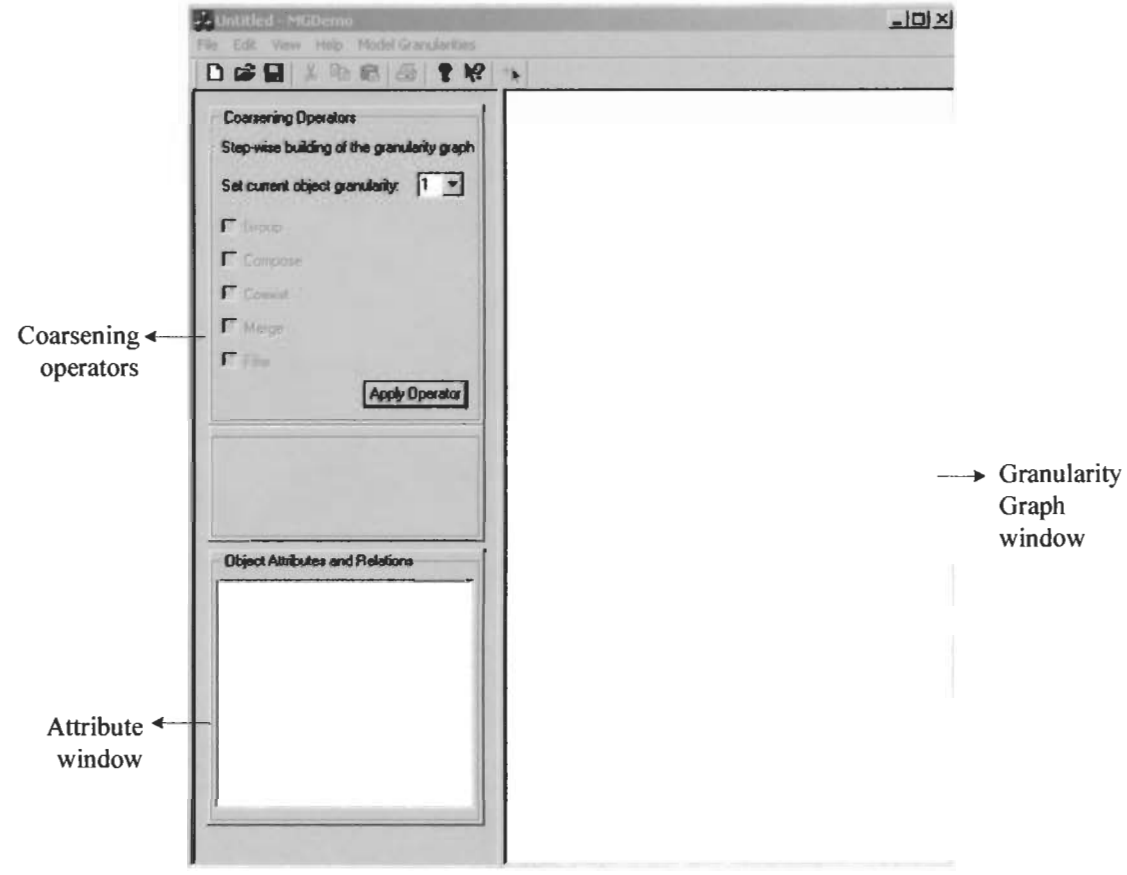


Figure 6.5 User interface of the prototype.

6.2.1 Creating a new granularity graph

Objects in an information space can be stored in a Microsoft Access Database (.mdb) file. The input file consists of 4 tables, namely *Object* - stores the objects, each object has a name and system id, *Rel1* – stores the attributes of objects, modeled using an is-a relations among objects, *Rel2* – a table to store part-of relations among objects, *Rel3* – stores the member-of relation among objects. The table *Rel1*, *Rel2*, and *Rel3* stores corresponding relations of objects using the object id in table *Object*. The object file name *(.mdb)* is passed as input for creating a new granularity graph through the menu item *Create a GG*. The application reads the set of objects from the file and displays them in the viewer window, as the first level of objects in the granularity graph. These are the leaf nodes of the granularity graph. Additional levels of objects can be added to the graph by applying the coarsening operators.

On saving the file, a text (*.gg*) file consisting of the objects in an information space is created. The values for objects are written in the format satisfying the object structure. Each object begins with the keyword *object* followed by the name, number of objects, list of attributes, and list of corresponding attribute values.

If the granularity graph already exists, then the user can open the graph *(.gg)* file through the menu item *Open a GG* and perform operations on the graph.

6.2.2 Applying coarsening operators

From the objects displayed in the viewer window, the user can select objects of interest by clicking the mouse inside the rectangular object area. The selected objects are highlighted and their attributes are displayed in the object attribute box. As the objects

are selected, the coarsening operators that can be applied are enabled in the graph builder dialog, provided on the left of the viewer window. The user can then select an operator that is enabled by checking the box against the operator. For each operator that can be successfully applied to the objects, a new amalgamated object is created. The new object with its values is added to the set of objects in the text file and to the granularity graph. The operator used in the amalgamation is represented by the corresponding color of the edge. These steps can be repeated to add further levels in a graph.

6.2.3 Browsing object granularities

Browsing operations for the graph (Chapter 4) are implemented to support retrieval of objects that are at different granularities. In the user interface, browsing operations can be chosen though the menu item *Granularity Graph – Browse*. We have implemented two of the browse operations: *getChildOp(B, A)* and *getDescendants(B, A)*. The operation *getChildOp* retrieves the fine-grained granularities that are adjacent to an object by applying a specific operator. The *getDescendants* operation enables retrieval of all the objects at a finer granularity to an object, until the leaf nodes are reached. The user selects a browsing operation and the result of the operation is displayed on the graph in the viewer window.

6.3 Illustration of the Prototype

This section illustrates the use of the prototype with an example. We use Acadia National Park as our information space of objects. Several hiking trails in the park are given as the input set of objects (e.g., input file *Acadia_Trails.mdb*) using the menu item *Create-GG*.

A granularity graph with the leaf nodes is created. The different *trails* objects from the file are displayed in the graph viewer window as the leaf nodes in the graph (Figure 6.6).

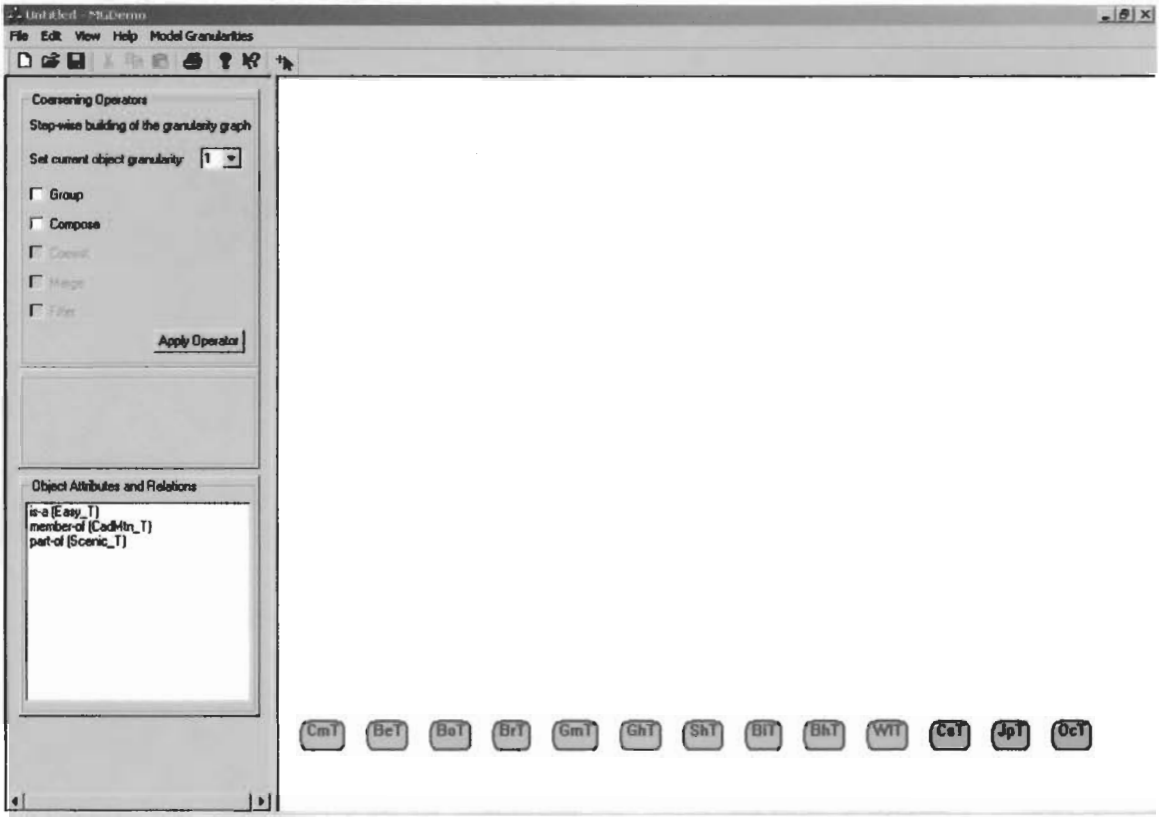


Figure 6.6 Selecting three *trails* for amalgamation from the set of objects.

A *Trail* can be selected and its corresponding attributes can be viewed in the attribute display window on the left window (Figure 6.6). Selected *trails* are highlighted in a green color in the graph window. Multiple *trails* can be selected for applying the coarsening operators (Figure 6.6). On selecting multiple trails, the coarsening operators that can be applied to the trails become available in the operator window. One or all of the operators can be selected by clicking in the checkbox against the operator and applied to objects.

On selecting an operator, an operator dialog box will pop up that displays the combining objects and the resultant amalgamated object (Figure 6.7). A suitable label can

be given to the amalgamated objects in a text box in the dialog. Additional attributes and values of objects can also be added to the newly created amalgamated object.

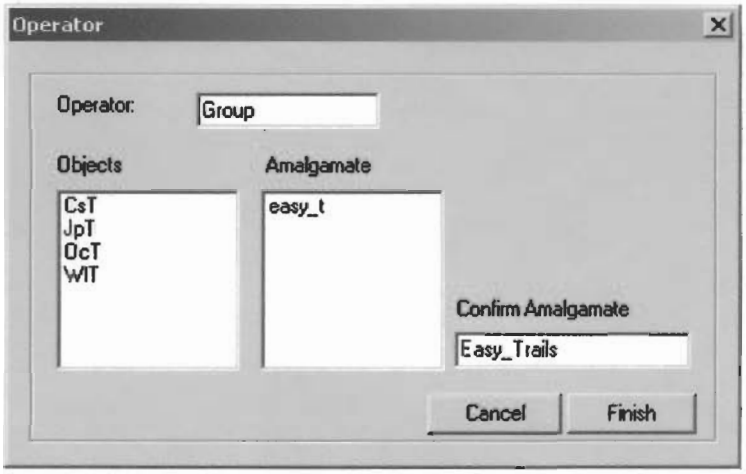
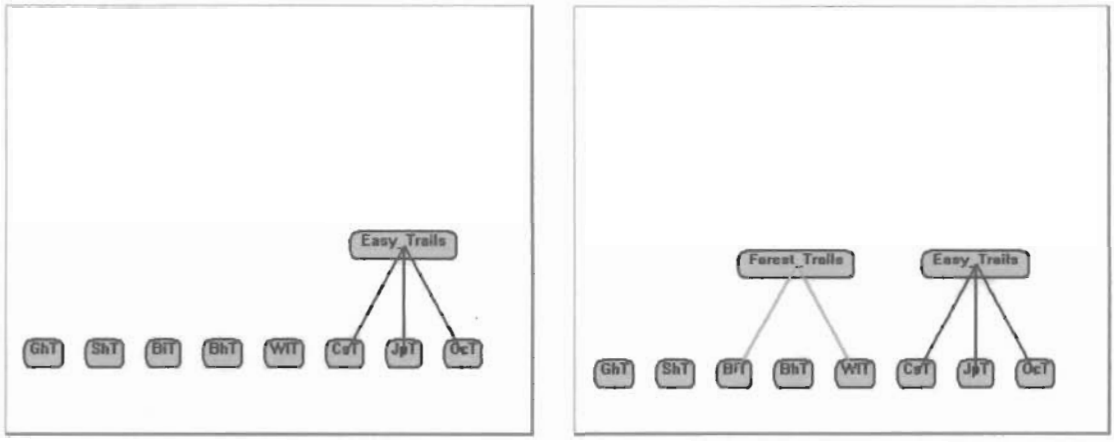


Figure 6.7 Creating an amalgamated object *Easy_Trails* for the selected objects by applying the *group* operator.

The resulting amalgamated object is added to the graph at a new level, connecting the combining objects with the corresponding coarsening operator (Figure 6.8). The edges in the graph, i.e., coarsening operators, are color coded in the application for providing a better graph visualization: *group* –red, *compose* –orange and *coexist* – purple.



(a) (b)

Figure 6.8 Step-wise building of a granularity graph. (a) Creating an amalgamated object *Easy Trails* by the *group* operator and (b) adding amalgamated object *Forest Trails* by applying the *compose* operator.

Other objects from the first level in the graph can be selected resulting in more amalgamated objects. Once all possible operators are applied to objects in level 1, a level 2 set of objects is constructed in the graph. Operators can now be applied to objects in level 2 to result in higher level of amalgamated objects. The process is repeated until a level with a single object is obtained. A granularity graph for Acadia Park with 4 levels of granularities is shown (Figure 6.9).

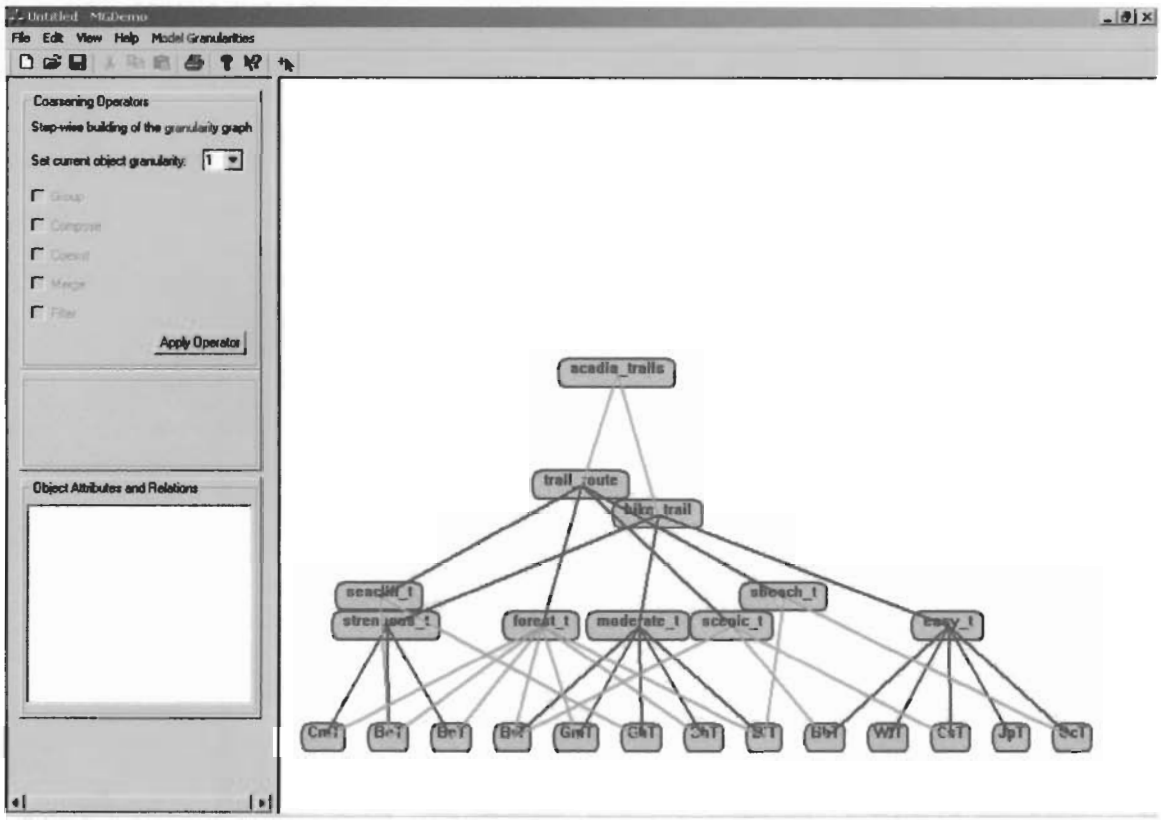


Figure 6.9 Granularity graph for the Acadia Trails.

Once the graph is constructed, browsing operations can be applied to the graph for retrieving finer or coarser granularities of objects. Two unary browsing operations *getChildOp* and *getDescendants* are implemented in this prototype. Browse operations can be selected from the menu item *Granularity graph – Browse*. On selecting a browse operation, a node in the graph must be selected for applying the browsing.

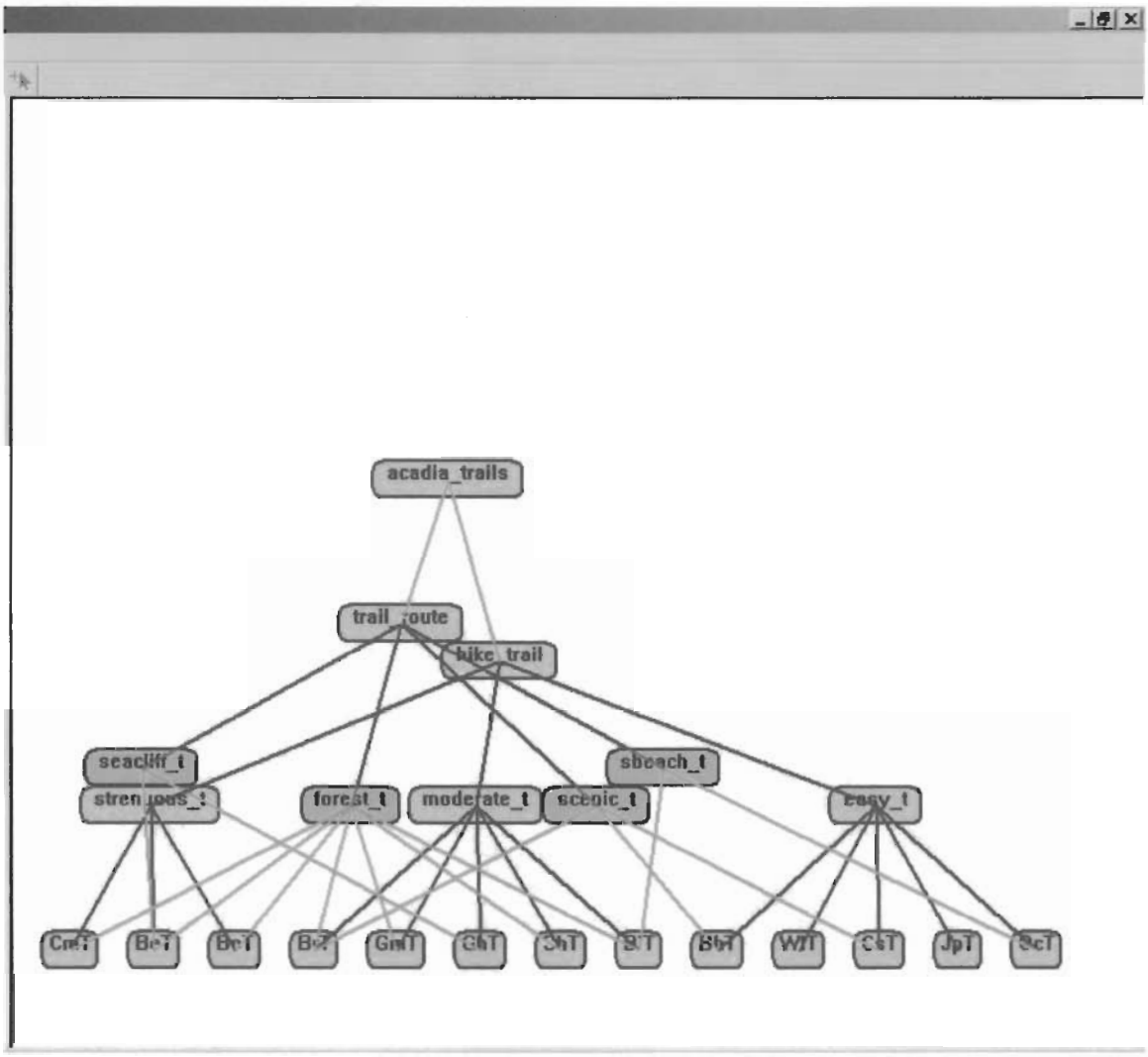


Figure 6.10 Result of the browse operation *getChlidOp* on *Trail_Routes* based on the *group* operator.

For example, the *getChilpOp* operation is applied to the object *Trail Routes* based on the *group* operator (Figure 6.10). The operation retrieves the objects that are at a finer granularity in the preceding level to *Trail Routes*, (i.e., *Sea-cliff Trails*, *Forest Trails*, *Scenic Trails*, *Sand Beach Trails)*. The result of the browse operation is highlighted in the graph. Similarly, the operation *getDescendant* can be applied to objects to retrieve all finer granularities of an object. For example, *getDescendants* when applied to the *Hike*

Trails retrieves all the finer granularities that are connected to *Hike trails*, displayed as orange rectangles in the graph (Figure 6.11).

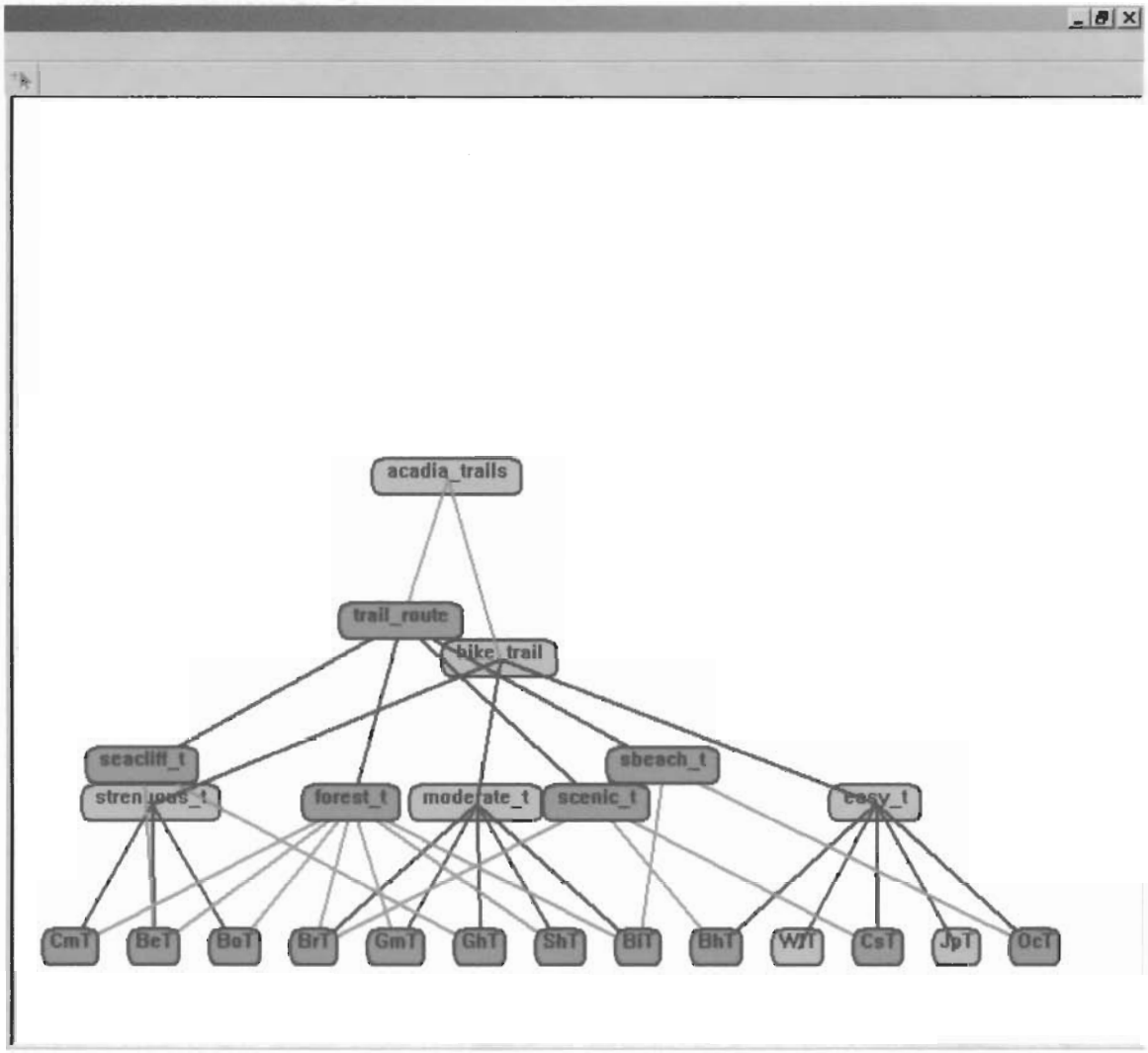


Figure 6.11 Result of the browse operation *getDescendants* on *Hike Trail*.

6.4 Summary

This chapter described the prototype implementation for constructing and browsing a granularity graph. The prototype design and specification, and the class structures were discussed to understand the data flow and the interaction between the application

program and the user. The prototype was also used as a test bed for deriving a framework of multiple granularities and investigating the application of the granularities.

The next chapter concludes this thesis with a summary and future recommendations to be carried out based on this research work.

Chapter 7

CONCLUSIONS AND FUTURE WORK

The focus of this thesis is to model multiple granularities of spatial objects and perform shifts among different granularities. An approach for modeling objects at different granularities has been developed with an understanding that spatial objects' attributes and relation with other spatial objects can be exploited to result in coarser granularities. The approach captures the different semantics associated with combining objects that lead to multiple granularities and presents a categorization of coarsening operators based on these semantics.

Multiple granularities of objects can be organized into a granularity graph. Such a granularity graph can be used for retrieving granularities of objects at finer or coarser granularities with respect to an object. It is also possible to analyze the multiple ways of arriving at a granularity and to determine relations among objects at different granularities in the graph. This chapter summarizes the thesis work and presents conclusions. The future directions for research based on this work are highlighted with recommendations.

7.1 Summary

There exist different granularities of objects, each suited for a particular purpose. In the process of reasoning about our information space, it is required to make available a means for performing shifts among the multiple granularities. In this thesis, we identify multiple granularities of objects and formalize shifts among them.

This thesis pursues an object-oriented approach for modeling multiple granularities of objects based on the concept of coarsening. An object is modeled as a structure consisting of attributes, attribute values, and relations with other objects. Based on the attributes and relations of objects, the different semantics of how objects can be combined resulting in coarser granularities are identified. As a result, four coarsening operators, *filter*, *group*, *compose*, and *coexist* are defined.

Applying the operators to objects recursively results in amalgamated or coarser granularities of objects. The multiple granularities of objects are organized in a framework, a granularity graph. A granularity graph is comprised of objects at different granularities related to each other by coarsening operators. Using the granularity graph, several browsing operations are defined. Browsing a granularity graph refers to making translations either to coarser granularities or to more detailed granularities in the graph. Browse operations on a graph are categorized as unary and binary operations. Unary operations can be applied to an object to retrieve its finer or coarser granularities of objects. Binary operations are used to determine the common coarser or finer granularities to two objects. For example, *getAllAncestors(X,M,N)* is a binary browse operation to obtain all the coarser granularities of objects to objects M and N. The

operations also enable the retrieval of objects that are connected to each other based on a particular coarsening operator.

This thesis also presents the compositions of coarsening operators. We derive a complete set of all possible compositions of the operators, consisting of 74 valid compositions. Compositions effectively collapse a sequence of operators into a simpler, reduced sequence. Thus, compositions can be used for determining a shorter path connecting two objects in the graph. The compositions also play an important role to determine the multiple ways of arriving at a granularity and to arrive at a desired granularity. The compositions exhibit an associative property. Using this property, the compositions can be applied anywhere in a sequence, providing more flexibility to find the multiple paths to arrive at a granularity. Compositions are used to find the relation among objects at multiple granularities in a granularity graph. The different relations that can be obtained by applying the compositions provide enhanced reasoning using the object granularities with regard to how the objects are connected in the graph. The prototype developed complements the approach by supporting the construction of a granularity graph and enabling browsing through multiple object granularities.

7.2 Conclusions

Different semantics are involved when creating coarser granularities of objects. We define a set of coarsening operators based on these semantics to derive amalgamated objects. From this foundation, solutions can be presented for research questions, such as is it possible to combine two objects in order to arrive at a coarser granularity? or what are the different ways in which an object can be combined with other objects?

The granularity graph is a rich structure modeling the multiple granularities. Browsing operations on the graph enable the retrieval of finer and coarser granularities of objects. Queries related to multiple granularities, such as what are all the objects that are at a finer granularity to an object? or what is a coarser granularity of an object? can be answered using the browsing operations.

Another major contribution of this thesis is the composition of coarsening operators. Compositions of operators are required for simplifying long sequences of operators connecting two objects in a granularity graph. Compositions of operators are derived based on their applicable instance-class pairs and the semantics associated with object amalgamations. The result of a composition is either a single convincing result or an undetermined result. The case when an undetermined result occurs, there can exist multiple results of the composition, such as one of the operators in the composition, or nothing. Thus, composition of a *compose* operator with itself cannot be always be generalized to *compose*. The hypothesis of this thesis is defined as: different semantics associated with object amalgamations yield correct results of the compositions of coarsening operators. We support our hypothesis by describing the compositions of the different *compose* operators. It is observed that detailed *compose* operators reveal correctly the results of compositions of coarsening operators.

Composition of operators enables one to reduce the sequence of n operators connecting two objects in a granularity graph up to a single operator. The simplification of the sequence of operators to a single operator, provided there are no undetermined relations between the granularities, indicates that it is possible to determine the relation between any two objects at different granularities in the graph.

Applying the compositions of operators to a path consisting of n operators, between two objects, it is possible to derive n-1 ways of arriving at a coarser granularity. Thus, the compositions complement the multiple ways of arriving at a granularity of objects and can also be used to find the different sequences of coarsening operators that lead to a coarser granularity from an object.

7.3 Future Work

This section lists a set of possible future research tasks that are enabled by this work.

- **Extending the set of coarsening operators**

The set of coarsening operators capturing multiple granularities is rich but not necessarily complete. There may be other ways of combining objects, for example, objects can combine to evolve in to a new object and objects can merge into another object. Evolution of objects results in a new object. The properties of the objects can be completely different from the combining objects and need not be determined by the structure of combining objects. Alternatively, dynamic objects exhibit the semantics of merging, for example, a *car* merging into a *traffic*. These semantics are temporal in nature and will need additional information about the objects. Several questions will need to be addressed. How will the temporality in the structure of objects modify the granularities? Can we integrate spatio-temporal objects into the granularity graph? What is the effect of including these objects in the granularity graph? Will the associative property of the compositions hold for these cases?

- **Computing undetermined compositions of the coarsening operators**

The compositions of operators had 86 undetermined compositions out of the 160 possible compositions. An extension of this thesis is to successfully reduce the number of

undetermined compositions. Are there any other attribute or relations of objects that when captured, can reduce the undetermined relations? What are all the possible values of operators that correspond to the undetermined compositions? Is it possible to list the different values that represent an undetermined composition?

- **Visualization of multiple granularities**

This thesis discussed only one possible model for multiple granularities of objects and the usefulness of the models in translating among the granularities. When modeling granularities in a GIS, the effectiveness of the shifts in the granularities is portrayed only by the spatial representation of the granularities of objects. Many functions need to be investigated to accommodate the spatial representation of objects. This opens the door to some challenging research questions for associating a spatial representation with objects over multiple granularities. Can we use this model to determine the relation among granularities and retrieve the corresponding spatial representations from another stored source? Is it possible to convey multiple granularities only by displaying the spatial representation of the fine-grained objects and derive methods for approximating the representation of coarser objects? Can this model be used as a meta-data or for relational indexing of objects to arrive at the corresponding spatial granularities?

BIBLIOGRAPHY

K. Beard (1990) Constraints on rule formation. in: B. Buttenfield and R. McMaster (Eds.), *Map Generalization: Making Rules for Knowledge Representation*. pp. 121-135, Longmans, London, UK.

B. Bederson and J. Hollan (1994) Pad++: a zooming graphical interface for exploring alternate interface physics. in: *ACM Symposium on User Interface Software and Technology (UIST)*, pp. 17-26.

M. Brodie, J. Mylopoulos, and J. Schmidt, Eds. (1984) *On conceptual modeling, perspectives from artificial intelligence, databases, and programming languages*. Springer-Verlag, New York, NY.

B. Buttenfield and J. Delotto (1989) *Multiple representations*. National Center for Geographic Information and Analysis (NCGIA), UCSB Santa Barbara, CA, Technical Report 89-3.

B. Buttenfield and R. McMaster, Eds. (1991) *Map generalization: Making Rules for Knowledge Representation*. John Wiley & Sons, New York, NY.

C. Dyreson and R. Snodgrass (1995) Temporal granularity. in: R. Snodgrass (Ed.), *The TSQL2 Temporal Query Language*. pp. 347-394, Kluwer Academic Publishers, Norwell, MA.

M. Egenhofer (1993) A model for detailed binary topological relationships. *Geomatica* 47(3&4): 261-273.

M. Egenhofer and A. Frank (1992) Object-oriented modeling for GIS. *Journal of the Urban and Regional Information Systems Association* 4(2): 3-19.

A. Frank and S. Timpf (1994) Multiple representations for cartographic objects in a multi-scale tree: An intelligent graphical zoom. *Computers and Graphics* 18(6): 823-829.

A. Frank, G. Volta, and M. McGranaghan (1997) Formalization of families of categorical coverages. *International Journal of Geographical Information Science* 11(3): 215-231.

G. Furnas (1986) Generalized fisheye views. in: M. Mantei and P. Orbeton (Eds.), *Human Factors in Computing Systems CHI'86*, New York, NY, pp. 16-23.

N. Guarino (1994) The ontological level. in: R. Casati, B. Smith, and G. White (Eds.), *Philosophy and the Cognitive Science*, Vienna, pp. 443-456.

N. Guarino and C. Welty (2000) A formal ontology of properties. in: R. Dieng and O. Corby (Eds.), *12th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pp. 97-112.

S. Guptill (1990) Multiple representations of geographic entities through space and time. in: K. Brassel (Ed.), *4th International Symposium on Spatial Data Handling*, Zurich, pp. 859-868.

T. Hadzilacos and N. Tryfona (1997) An extended entity-relationship model for geographic applications. *SIGMOD Record* 26(3): 24-29.

H. Hamilton, R. Hilderman, L. Li, and J. Randall (1999) Generalization lattices. *2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD)*: 328-336.

H. Hamilton and D. Randall (1999) Heuristic selection of aggregated temporal data for knowledge discovery. in: I. Imam, Y. Kodratoff, A. El-Dessouki, and M. Ali (Eds.),

Multiple Approaches to Intelligent Systems,12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE). Lecture Notes in Computer Science 1611, pp. 714-723, Springer Verlag, Cairo, Egypt.

M. Hammer and D. McLeod (1981) Database description with SDM: a semantic database model. *Association for Computing Machinery: Transactions on Database Systems* 6(3): 351-386.

R. Hilderman, H. Hamilton, and N. Cercone (1999) Data mining in large databases using domain generalization graphs. *Journal of Intelligent Information Systems* 13(3): 195-234.

J. Hobbs (1990) Granularity. in: D. Weld and J. Kleer (Eds.), *Readings in Qualitative Reasoning about Physical Systems*, San Mateo, CA, pp. 542-545.

K. Hornsby and M. Egenhofer (1998) Identity-based change operations for composite objects. *8th International Symposium on Spatial Data Handling*: 202-213.

K. Hornsby and M. Egenhofer (1999) Shifts in detail through temporal zooming. in: A. Toja, A. Cammelli, and R. Wagner (Eds.), *Tenth International Workshop on Database and Expert Systems Applications*. pp. 487-491, IEEE, Computer Society, Florence, Italy.

K. Hornsby and M. Egenhofer (2002) Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence* 36: 177-194.

G. Kosters, B. Pagel, and H. Six (1996) GeoOOA: object-oriented analysis for geographic information systems. in: *IEEE International Conference for requirements Engineering*, Colorado Springs, pp. 245-253.

S. Lamy, A. Ruas, A. Demazeau, M. Jackson, W. Mackaness, and R. Weibel (1999) The application of agents in automated map generalization. in: *19th International Cartographic Conference*, Ottawa, Canada, pp. 1225-1234.

J. Mackinlay, G. Robertson, and S. Card (1991) The perspective wall: detail and context smoothly integrated. in: *ACM CHI Conference on Human Factors in Computing Systems*, New Orleans, LA, pp. 173-180.

S. Madria, M. Mohania, and J. Roddick (1998) *A query processing model for mobile computing using concept hierarchies and summary databases*. Center for Advanced Information Systems, Technical Report 16.

D. Maier and J. Stein (1981) Development and implementation of an object-oriented DBMS. *Association for Computing Machinery: Transactions on Database Systems* 6(3): 167-184.

R. McMaster and K. Shea (1992) *Generalization in digital cartography*. Association of American Geographers, Washington, DC.

J. Muller, J. Lagrange, and R. Weibel, Eds. (1995a) *GIS and generalization: methodology and practice*. Taylor & Francis, London, UK.

J. Muller, R. Weibel, J. Lagrange, and F. Salge (1995b) Generalization: state of the art and issues. in: J. Muller, J. Lagrange, and R. Weibel (Eds.), *GIS and generalization: methodology and practice*. pp. 3-17, Taylor and Francis, London, UK.

P. Oosterom and J. Bos (1989) An object-oriented approach to the design of geographic information systems. *Computing and Graphics* 13: 409-418.

D. Ormsby and W. Mackaness (1999) The development of phenomenological generalization within an object oriented paradigm. *Cartography and Geographical Information Systems* 26: 70-80.

E. Puppo and G. Dettori (1995) Towards a formal model for multiresoultion spatial maps. in: M. Egenhofer and J. Herring (Eds.), *Advances in Spatial Databases SSD'95, 4th International Symposium. Lecture Notes in Computer Science* 951, pp. 152-169, Springer Verlag, Portland, ME.

R. Ramakrishnan (1997) *Database management systems*. McGrawHill,

R. Read, D. Fussell, and A. Silberschatz (1992) A multi-resolution relational data model. in: *Proceedings of the 18th Conference on Very Large Databases*, Vancouver.

U. Schiel (1989) Abstractions in semantic networks: axiom schemata for generalization, aggregation and grouping. in: (Eds.), *ACM SIGART Bulletin*. pp. 25-26, ACM Press, Brazil.

B. Smith and A. Varzi (1997) Fiat and bona fide boundaries. in: S. Hirtle and A. Frank (Eds.), *Spatial Information Theory: A Theoretical Basis for GIS*, Laurel Highlands, Pennsylvania, pp. 103-119.

J. Smith and D. Smith (1977) Database abstractions: aggregation. *Communications of the ACM* 20(6): 405-413.

J. Stell and M. Worboys (1998) Stratified map spaces: a formal basis for multi-resolution spatial databases. in: T. Poiker and N. Chrisman (Eds.), *Eighth International Symposium on Spatial Data Handling*, Vancouver, Canada, pp. 180-189.

J. Stell and M. Worboys (1999) Generalizing graphs using amalgamation and selection. in: R. Güting, D. Papadias, and F. Lochovsky (Eds.), *Advances in Spatial Databases, 6th International Symposium, SSD'99*, Hong Kong, China, pp. 19-32.

M. Stone, K. Fishkin, and E. Bier (1994) The movable filter as a user interface tool. in: *Human Factors in Computing Systems (CHI)*, Boston, MA.

B. Stroustrup (1991) *The C++ programming language*. Addison-Wesley,

M. Tanaka and T. Ichikawa (1988) A visual user interface for map information retrieval based on semantic significance. *IEEE Transactions on Software Engineering* 14(5): 666-670.

S. Timpf (1999) Abstractions, levels of detail, and hierarchies in map series. in: C. Freska and D. Mark (Eds.), *Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science. Lecture Notes in Computer Science* 1661, pp. 125-139, Springer Verlag, Stade, Germany.

S. Timpf and A. Frank (1998) *Geographic zooming*. Department of GeoInformation, Technical University of Vienna, Technical Report.

R. Weibel and G. Dutton (1999) Generalizing spatial data and dealing with multiple representations. in: P. Longley, M. Goodchild, D. Maguire, and D. Rhind (Eds.), *Geographical Information Systems*, New York, pp. 125-155.

G. Wiederhold, S. Jajodia, and W. Litwin (1991) Dealing with granularity of time in temporal databases. in: R. Andersen, J. Bubenko, and A. Solvberg (Eds.), *3rd International Conference on Advanced Information Systems Engineering (CAiSE)*, Trondheim, Norway, pp. 124-140.

M. Winston, R. Chaffin, and D. Herrmann (1987) A taxonomy of part-whole relations. *Cognitive Science* 11: 417-444.

M. Worboys (1994) Object-oriented approaches to geo-referenced information. *International Journal of Geographical Information Systems (IJGIS)* 8(4): 385-399.

BIOGRAPHY OF THE AUTHOR

Chitra Ramalingam was born in Kerala, India on October 23, 1976. She received her undergraduate degree, Bachelor of Engineering in Computer Engineering, from Bombay University, India, in 1998. Thereafter, she worked at the Center of Studies in Resources Engineering (CSRE), one of the departments with active research in GIS, at the Indian Institute of Technology (IIT), Bombay, for two years. At CSRE, she held the post of a software developer and research associate and was involved in developing a geographic information system, called GRAM++. She then joined the University of Maine's Spatial Information Science and Engineering program as a master's candidate in the fall of 2000. Here she worked as a graduate research assistant with the National Center for Geographic Information and Analysis (NCGIA). Chitra is a candidate for the Master of Science degree in Spatial Information Science and Engineering from The University of Maine in December, 2002.